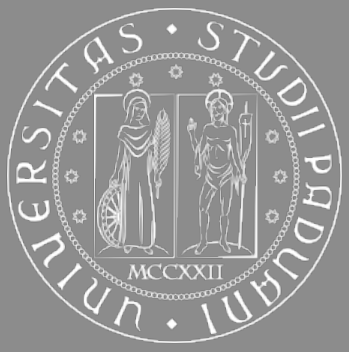


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO DI MATEMATICA

Laurea in Informatica

Il progetto "App ARPAV"

Laureando	*Giacomo Lorigiola*
Relatore	*Dott. Mauro Conti*
Agenzia ospitante	ARPAV
Tutor dell'agenzia	*Dott. Luca Menini*
Data	13.12.2012

Anno accademico 2012-2013

Alla mia famiglia,
e a tutti coloro che mi hanno sempre sostenuto.

Sommario

L'ARPAV è l'agenzia della Regione Veneto che si occupa della prevenzione e del monitoraggio ambientale. Di recente ha sentito l'esigenza di aprirsi verso il mondo della telefonia mobile e per questo motivo ha avviato un ampio progetto denominato *App ARPAV*, volto alla realizzazione di applicazioni per gli smartphones. Grazie a queste app, l'ARPAV vuole mettere a disposizione dei cittadini dei comodi strumenti, che permettano la visualizzazione dei dati ambientali da essa monitorati.

Lo stage svolto dallo studente presso l'agenzia ARPAV nella sede di Padova ha avuto come obbiettivo l'avanzamento di questo ampio progetto, sviluppando alcune applicazioni per la piattaforma mobile Android. Nella prima metà dello stage, lo studente si è occupato del miglioramento dell'esistente applicazione *ARPAV Meteo*, che ha lo scopo di fornire agli utenti informazioni meteo relative ai comuni veneti. Le modifiche apportate sono state concordate con l'agenzia, sulla base di osservazioni maturate durante il precedente periodo di utilizzo dell'applicazione e si sono concluse con il rilascio dei nuovi aggiornamenti nell'Android Market. Nella seconda metà dello stage, lo studente ha lavorato nella realizzazione di una nuova applicazione *ARPAV Idro*, che ha l'obbiettivo di fornire informazioni riguardanti l'idrometria e la pluviometria dei principali bacini presenti nel territorio veneto. I dati sono visualizzati sotto forma di grafici per renderne più chiara e veloce la loro lettura. Anche in questo caso, l'applicazione è stata rilasciata nell'Android Market per essere disponibile agli utenti.

L'esperienza di stage svolta è stata per me molto positiva: ho potuto approfondire le mie conoscenze personali, in particolar modo in ambito di sviluppo per la piattaforma Android, realizzando degli importanti strumenti innovativi utilizzabili gratuitamente. Il lavoro prodotto durante lo stage è stato ritenuto dall'agenzia molto soddisfacente ed esaudisce ampiamente le aspettative attese; pertanto ritengo che il mio contributo apportato presso l'ARPAV sia da ritenersi più che positivo.

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 L'agenzia ARPAV

A seguito di un referendum dell'Aprile del 1993 che ha abrogato le competenze del Servizio Sanitario Nazionale e delle ULSS nel campo del controllo e della prevenzione ambientale, il Parlamento con la Legge 61 del 1994 ha affidato tali compiti ad apposite "Agenzie Regionali", che diventano i centri deputati alla vigilanza e controllo ambientale in sede locale.
In Veneto, ARPAV, viene istituita con la Legge Regionale n°32 del 18 ottobre 1996 e diventa operativa il 3 ottobre 1997.

1.1.1 Obbiettivi

L'ARPAV è un'agenzia regionale che si occupa del monitoraggio ambientale, svolgendo attività di tutela, controllo e protezione dell'ambiente. Tra i settori di pertinenza dell'ARPAV troviamo: acqua, suolo, aria, rifiuti solidi e liquidi, radioattività ambientale, rischi di incidenti e disastri ambientali e molto altro.
L'ARPAV nella sua operatività ha due obiettivi strettamente connessi:

- la **protezione**, attraverso i controlli ambientali che tutelano la salute della popolazione e la sicurezza del territorio;
- la **prevenzione**, attraverso la ricerca, la formazione, l'informazione e l'educazione ambientale.

ARPAV realizza i propri obiettivi utilizzando competenze tecnico-scientifiche che ne diventano caratteristica distintiva.

1.1.2 L'innovazione: il progetto *"App ARPAV"*

Attraverso il sito web dell'ARPAV (`www.arpa.veneto.it`) sono disponibili tutte le informazioni pubbliche dei monitoraggi ambientali, come il meteo, i bollettini, i livelli idrometrici e pluviometrici, l'inquinamento dell'aria, dell'acqua ecc.
Di recente però, in seguito all'enorme dilagare negli ultimi anni degli smartphones e della diffusione dell'accesso ad internet nei cellulari, l'agenzia ARPAV ha deciso di porre maggiore attenzione anche a queste nuove tecnologie. Infatti l'esistenza di un solo portale web accessibile da browser per la visualizzazione dei dati ambientali messi a disposizione dell'agenzia, risultava scomodo e poco efficiente per un costante accesso attraverso smartphones.
Da qui è nata l'esigenza di realizzare apposite app per dispositivi mobili, che permettessero la visione di tali dati garantendone ove necessario una rielaborazione, come per esempio la realizzazione di grafici, supportando così tutte le funzionalità disponibili dal portale web.
Tutto ciò si è concretizzato con il progetto *App ARPAV*, che prevede lo sviluppo di applicazioni per smartphones dedicate ai dati ambientali prodotti dall'agenzia in diretta.
Attualmente le applicazioni smartphones attive messe a disposizione dall'ARPAV sono:

- *ARPAV Meteo*:
  che prevede la consultazione dei dati meteo dei comuni veneti, la lettura dei bollettini e la visualizzazione di immagini radar;
- *ARPAV Balneazione:*
  che permette di consultare la situazione della balneabilità delle coste venete;
- *ARPAV Idro*:
  che permette di visualizzare attraverso grafici il livello idrometrico e pluviometrico dei bacini e delle località venete.

Il codice delle app sopra citate è disponibile sotto licenza GPL in apposite repository reperibili all'interno dell'account GitHub proprio dell'agenzia ARPAV: `https://github.com/venetoarpa`. Questa scelta di adottare una licenza di tipo open-source è derivante dal fatto che il fine dell'agenzia è quello di poter divulgare apertamente i propri dati ambientali da essa monitorati, così da poter essere più facilmente raggiungibili anche attraverso differenti canali di divulgazione non necessariamente propri dell'agenzia.

1.2 Android: passato e presente

Android Inc. è stata fondata in California nell'ottobre 2003 da Andy Rubin, Rich Miner, Nick Sears e Chris White e, secondo le parole di Rubin, era volta allo sviluppo "... di dispositivi cellulari più consapevoli della posizione e delle preferenze del loro proprietario".
La società operò in segreto fino al 17 agosto 2005, anno in cui fu acquistata dalla Google grazie alla quale potè entrare nel mercato della telefonia mobile.
Nei successivi anni Robin e il suo team iniziarono a sviluppare un software per dispositivi mobili basato sul kernel Linux; nel novembre 2007 si ebbe la presentazione ufficiale del neonato "robottino verde" e il rilascio del "Software Development Kit" (Android SDK). L'anno successivo fu lanciata la prima versione Android 1.0, Apple Pie.

Dal 2007 ad oggi sono state rilasciate nuove e più potenti versioni Android, sino a giungere alle recenti 2.3 Gingerbread, 3.0 Honeycomb, 4.0 Ice Cream Sandwitch e 4.1 Jelly Bean. Durante questo periodo di tempo Android si è ampiamente diffuso tra i dispositivi cellulari, sino a diventare il software mobile più popolare nel mercato, spiazzando il vecchio colosso Symbian, i più recenti Blackberry e Windows Phone e persino il maggior competitor di sempre iOS.
Il grafico sottostante (vedi Figura ***1.1***) mostra come negli USA la diffusione degli smartphones Android abbia superato il 50% del mercato, raggiungendo picchi del 60% nei primi mesi del 2012.

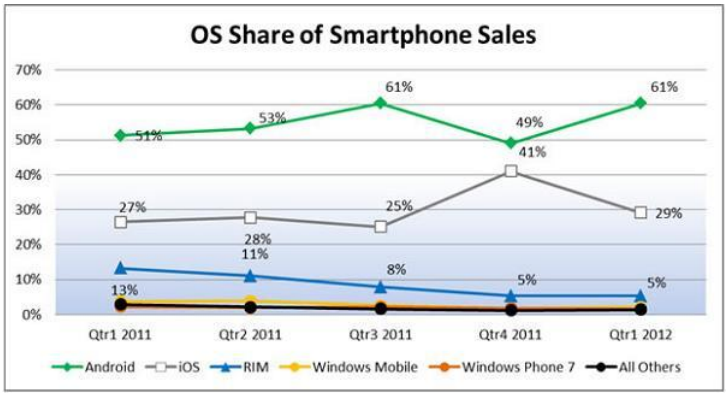


Figura 1.1: Diffusione piattaforme mobili negli USA

1.3 Organizzazione dei contenuti

Nei capitoli a seguire vengono presentate dapprima le tecnologie utilizzate durante lo stage (Capitolo ***2***); successivamente vengono descritte le librerie adottate nello sviluppo delle app (Capitolo ***2.3***); poi viene illustrato nel dettaglio il lavoro di aggiornamento per l'app *ARPAV Meteo* svolto nella prima metà dello stage, (Capitolo ***3***); in fine viene trattata la realizzazione dell'app *ARPAV Idro* (Capitolo ***4***).

Per evitare ridondanze tutti i termini e gli acronimi che necessitano di definizione e che sono presenti nel seguente documento, saranno seguiti da una "g" ad apice (e.g. App^g) alla loro prima occorrenza e saranno riportati con le rispettive spiegazioni nell'apposito Capitolo ***6***.

2 Ambiente e strumenti di lavoro

2.1 Strumenti utilizzati

Per la realizzazione delle applicazioni Android e per la stesura della relativa documentazione, sono stati utilizzati i seguenti strumenti di lavoro:

- **Eclipse IDE**
  Lo strumento base per lo svolgimento dello stage è stato l'ambiente di sviluppo software integrato Eclipse IDE. Questo strumento può essere utilizzato per la produzione di software di vario genere grazie alla sua ampia flessibilità, determinata dall'uso di appositi plug-in che consistono in componenti software ideate per uno specifico scopo.
  Il software Eclipse è liberamente scaricabile dal sito `http://www.eclipse.org`.

- **Android SDK e Android Development Tools (ADT) per Eclipse**
  L'Android Software Development Kit (SDK) è un framework per la realizzazione di applicazioni per Android e consiste in un insieme di strumenti software per lo sviluppo.
  Grazie al plug-in Android Development Tools (ADT) per Eclipse IDE, è possibile utilizzare l'Android SDK all'interno dell'ambiente di sviluppo Eclipse.
  Durante lo svolgimento dello stage è stato utilizzato l'Android SDK nell'ultima versione disponibile (v.20), con il supporto a Jelly Bean.

- **Git e GitHub**
  Per il sistema di versionamento sono stati adottati Git e GitHub; quest'ultimo è un servizio di web hosting e d'interfaccia. E' stata fatta questa scelta poiché l'agenzia ARPAV adottava già questo servizio di repository.

- **Cola Git GUI**
  GUI[g] desktop per un semplice e veloce utilizzo del sistema di versionamento git.

- **Latex e TextMate**
  Latex è un linguaggio di markup liberamente disponibile, particolarmente indicato per l'elaborazione di documenti scientifici con elevati livelli di qualità.
  Per la redazione e la correzione ortografica dei documenti in LaTex è stato utilizzato l'IDE TextMate.

- **LucidChart**
  Per la realizzazione dei diagrammi UML[g] è stato utilizzato il servizio online gratuito LucidChart, che consiste in una piattaforma molto completa per la costruzione di svariati tipi di

diagrammi.

- **Smartphone e tablet**
  Durante lo svolgimento dello stage, oltre ai simulatori offerti dall'Android SDK, sono stati utilizzati anche diversi dispositivi mobile per avere una verifica di quanto prodotto.
  I dispositivi adottati per i test disponevano di differenti caratteristiche hardware e software, che hanno permesso di poter verificare il corretto funzionamento e comportamento di quanto sviluppato.
  Sono stati pertanto utilizzati i seguenti dispositivi mobile:
  - Htc Incredible S (Gingerbread v.2.3.5)
  - Htc Wildfire (Froyo v.2.2)
  - Samsung Galaxy S Plus (Gingerbread v.2.3.6)
  - Samsung Galaxy S II (Ice Cream Sandwitch v.4.0)
  - Samsung Galaxy Tablet 10.1 (Ice Cream Sandwitch v.4.0)

2.2 Android SDK e Android Development Tools

Le applicazioni di Android sono sviluppate all'interno del framework Android SDK, ossia di una struttura dati specifica. Utilizzando l'ambiente di sviluppo Eclipse con il plug-in Android Development Tools (ADT), questa struttura del framework risulta essere molto chiara: alla creazione di ogni nuovo "Progetto Android", lo sviluppo ha inizio a partire dalla struttura base del framework che viene automaticamente creata.
Il framework è caratterizzato da una dualità: vi sono parti dinamiche scritte in Java[g] e parti statiche scritte in XML[g]. Tipico delle parti statiche sono quelle caratteristiche che non cambiano durante l'esecuzione dell'applicazione, come per esempio il colore dello sfondo; mentre per quanto riguarda le parti dinamiche, queste consistono nella parte esecutiva, come per esempio la gestione degli eventi.

Suddivisione del framework

Il framework è suddiviso in cartelle e sottocartelle con struttura gerarchica ad albero, che determinano una rigida ma ordinata organizzazione dei file. Nello specifico all'interno del framework sono presenti le seguenti cartelle basi:

- **src:**
  questa cartella serve per contenere tutti i package e i rispettivi file .java con il codice sorgente dell'applicazione in linguaggio Java.
- **gen:**
  gen contiene tutti i file "R.java", ossia i file contenenti gli indici delle risorse Android create durante lo sviluppo. Si tratta di file generati automaticamente e, come tali, non devono essere modificati. Questi file vengono ricreati ogni volta che si cambia qualcosa nella directory res, ad esempio, quando si aggiunge un file di immagine o XML.
- **res**
  questa cartella è dedicata alle risorse dell'applicazione. Ogni tipo di risorsa deve essere dichiarata nel rispettivo file XML in base al tipo di risorsa, fatta eccezione per le immagini che devono invece essere inserite nelle corrispondenti cartelle drowable.
  Al suo interno sono presenti le seguenti sottocartelle:
  - **anim**
    cartella dedicata alla dichiarazione delle animazioni utilizzate dall'applicazione.
  - **drawable**
    cartella utilizzata per il contenimento di tutte le immagini utilizzate dall'app[g].

  - **layout**
    cartella contenente i file XML per la creazione dei layout dell'applicazione relativi alle rispettive *viste*.
  - **menu**
    cartella contente i file XML dediti alla dichiarazione dei menu relativi ad ogni specifica *vista* dell'applicazione.
  - **value**
    cartella utilizzata per la dichiarazione di risorse generiche come le stringhe, e gli stili utilizzati dall'applicazione, come per esempio le definizioni dei colori.

- **assets**
  questa cartella presenta una funzionalità simile alla cartella res, poiché anche in questo caso la cartella serve per il contenimento di risorse necessarie all'applicazione. La differenza consiste però nel fatto che in questa cartella, qualsiasi risorsa venga in essa inserita, può mantenere il suo formato di file originale. Successivamente si può accedere a questi file attraverso la classe *AssetManager* messa a disposizione dalla libreria Android, che permette la lettura di tali file come stream di bytes.

- **bin**
  classe contenente tutti i file compilati, cioè i file .class di Java. In questa cartella sono inoltre contenute delle copie di tutte le immagini presenti nelle risorse.

  - **res/drowable**
    cartella che contiene le immagini utilizzate dall'applicazione.
  - **AndroidManifest.xml**
    file che descrive l'applicazione al dispositivo in cui viene eseguita. Il Manifest elenca la lista delle necessità del programma per poter operare nel sistema: per esempio la richiesta di connessione alla rete, l'interazione con la rubrica del dispositivo e molto altro.

- **libs**
  questa cartella serve per l'inserimento di eventuali librerie Java esterne, utilizzate dall'applicazione.

Le Activity Android

Il componente principale messo a disposizione dal framework è l'*Activity* che può essere descritta come la rappresentazione di una schermata/interfaccia utente. L'Activity presenta un ciclo di vita a sé stante gestito dalla piattaforma Android. E' quest'ultima infatti ad occuparsi delle schermate (e quindi, le varie Activity) mantenendone in primo piano (foreground) una sola alla volta e portando in secondo piano (background) le altre. Viene così generata una sorta di stack delle Activity.
Pertanto in generale un'applicazione Android è composta da più Activity richiamabili tra loro. Tecnicamente una Activity è una classe java che estende (anche indirettamente) la classe android.app.Activity.
In Figura ***2.1*** viene descritto il ciclo di vita di una Activity, detto anche "Android Activity lifecycle". I metodi illustrati sono gestiti da Android stesso, e possono essere sovrascritti per poter eseguire operazioni personalizzate a seguito di un particolare stato.

- onCreate(): questo metodo è invocato quando l'activity viene lanciata per la prima volta;
- onStart( ): indica che l'activity sta per essere avviata e quindi visualizzata all'utente;
- onResume( ): viene invocato quando l'activity è visibile all'utente;
- onPause( ): viene invocato prima che l'activity vada in background, di norma avviene quando è stata avviata un'altra activity;

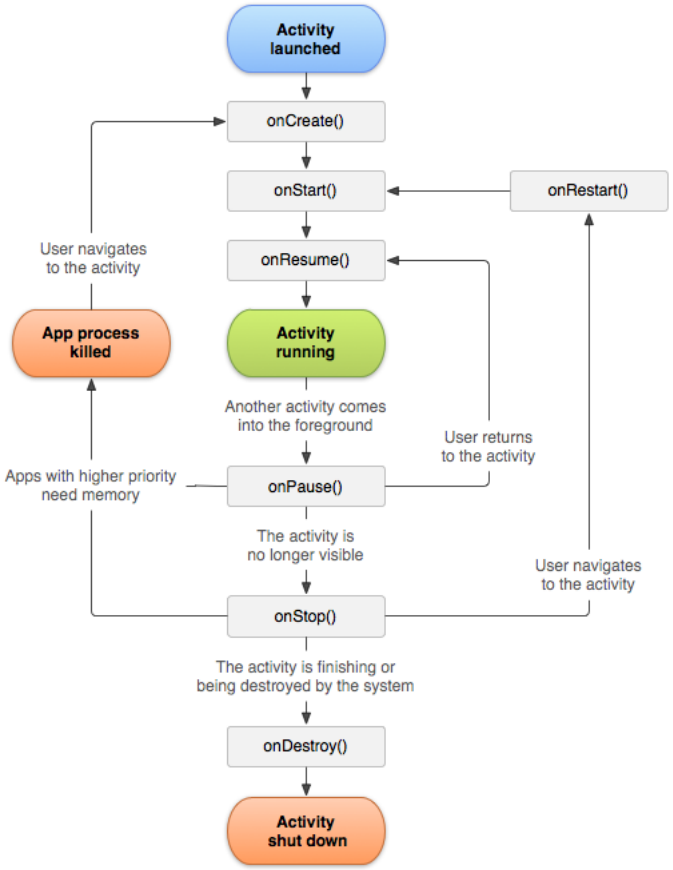


Figura 2.1: Android Activity lifecycle

- onStop( ): richiamato quando l'activity non è più visibile all'utente. In caso di scarsa memoria il metodo non viene invocato e l'Activity viene eliminata direttamente;
- onRestart( ): invocato quando una Activity deve ritornare visibile all'utente;
- onDestroy( ): richiamato giusto prima che l'activity venga distrutta. In caso di scarsa memoria il metodo non viene invocato e l'Activity viene eliminata direttamente.

Emulatori

Per un test pratico costante delle applicazioni, è possibile utilizzare degli emulatori Android: all'interno di quest'ultimi l'applicazione creata può essere eseguita simulandone il reale comportamento in un dispositivo fisico. Grazie all'Android Development Tools è possibile creare e avviare emulatori personalizzati (vedi Figura ***2.2***) direttamente dall'ambiente di sviluppo Eclipse.

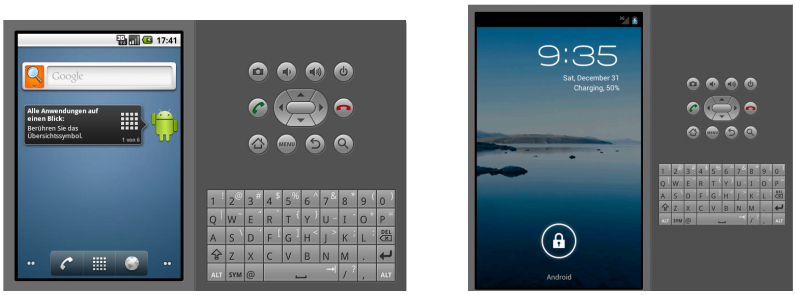


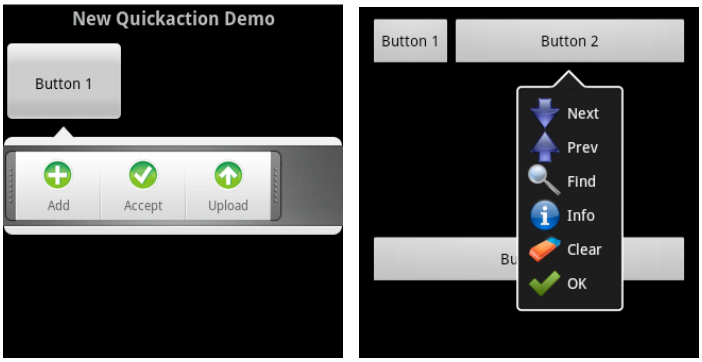
Figura 2.2: Emulatori Android

2.3 Librerie usate

Durante lo sviluppo per Android, è stato ritenuto opportuno utilizzare delle librerie che permettessero una più veloce e appropriata realizzazione di alcuni requisiti.
In questo capitolo verranno pertanto trattate le librerie utilizzate.

2.3.1 NewQuickAction e NewQuickAction3D

NewQuickAction è una libreria Android che permette la veloce realizzazione di *Quick Action Dialogs*, che consistono in rapidi menù richiamabili da bottone. Essi possono sostituire o integrare la normale funzione di menù di Android.
Esistono due varianti della libreria che si differenziano nelle loro grafiche, come visualizzabile in Figura ***2.3***.



(a) NewQuickAction (b) NewQuickAction3D

Figura 2.3: Libreria - NewQuickAction

Le librerie sono state scaricate dalle apposite repository presenti su GitHub e dispongono di documentazioni e applicazioni demo per illustrarne il funzionamento. Esse sono state sviluppate da Lorensius W. L. T con il contributo di Kevin P. e presentano licenza open-source Apache 2.0.

2.3.2 AChartEngine

Libreria per Android che permette la realizzazione di grafici personalizzabili attraverso l'inserimento di appositi valori. Dispone di vari tipi di grafici: lineari, a barre, circolari ecc. Vengono riportati alcuni esempi in Figura ***2.4***.
La libreria è stata utilizzata nella sua corrente versione 1.0.0 e dispone di apposita documentazione, codice sorgente, applicazione demo di prova e file di libreria java *.jar*.
E' stato possibile ricavare tali informazioni attraverso l'apposito sito dedicato `www.achartengine.org`.
La libreria è stata sviluppata dalla 4ViewSoft Company e dispone di licenza open-source Apache 2.0.

2.3.3 MapViewBalloons

Libreria molto diffusa che si integra con la libreria di Google Maps per Android (com.google.android.maps), e che permette la visualizzazione di informazioni relative agli elementi posizionati su di una mappa.
La visualizzazione delle informazioni avviene attraverso l'apertura di un "fumetto" come indicato in Figura ***2.5***.

Questa libreria è stata utilizzata nella realizzazione di app Android molto diffuse come: Foursquare, FriendCaster for Facebook, Brasileirao e molte altre.
La libreria dispone di apposito sito internet `www.jgilfelt.github.com/android-mapviewballoons`, dal quale è stato possibile ricavare tutte le informazioni necessarie e i link alla repository in GitHub

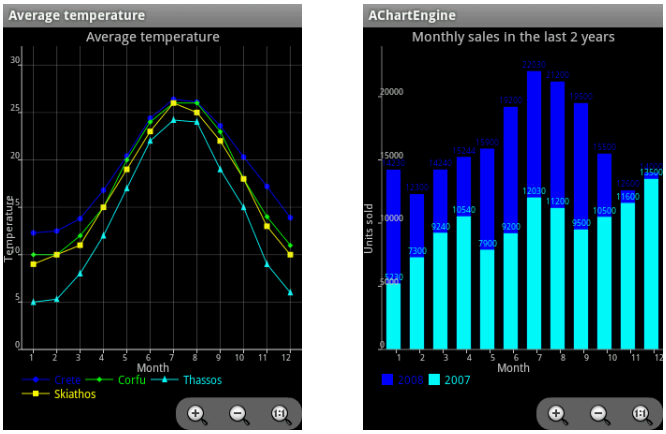


Figura 2.4: Libreria - AChartEngine

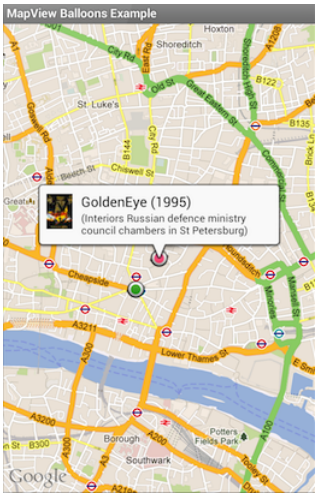


Figura 2.5: Libreria - MapView Balloons

dove risiede il codice.
La libreria è stata realizzata da Jeff G. e dispone di licenza open-source Apache 2.0.

2.3.4 Altre librerie

Di seguito vengono elencate altre librerie minori utilizzate:

- com.google.android.maps (Google Maps Android API)
- javax.xml.parsers
- org.apache.http
- org.w3c.dom

3 ARPAV Meteo

In questo capitolo viene presentato e descritto il lavoro svolto durante la prima parte dello stage, che è consistito in una fase di manutenzione e miglioramento dell'applicazione *ARPAV Meteo*.

3.1 Descrizione

L'ARPAV , grazie alla sua attività di monitoraggio ambientale, mette a disposizione dei cittadini i dati rilevati, attraverso servizi informativi gratuiti disponibili nel portale ARPAV accessibile dal sito `www.arpa.veneto.it`. Tra questi troviamo il "Meteo ARPAV" che permette la visualizzazione delle previsioni meteo della Regione Veneto. Grazie al progetto *App ARPAV* questo servizio da alcuni mesi può essere consultato anche attraverso l'apposita applicazione per Android, *ARPAV Meteo*, che può essere scaricata gratuitamente dall'Android Market[9].
L'applicazione è suddivisa in tre sezioni, *Meteo*, *Bollettini* e *Radar*, che di seguito vengono descritte nel dettaglio.

Sezione *Meteo*

In questa sezione, visualizzabile in Figura ***3.1***, l'applicazione permette di conoscere lo stato del cielo, le temperature e le precipitazioni previste per i prossimi 4 giorni.
E' possibile consultare le principali informazioni meteo relative ai comuni di interesse.
Quest'ultimi sono selezionabili in base alla provincia di appartenenza e possono essere aggiunti ai preferiti.

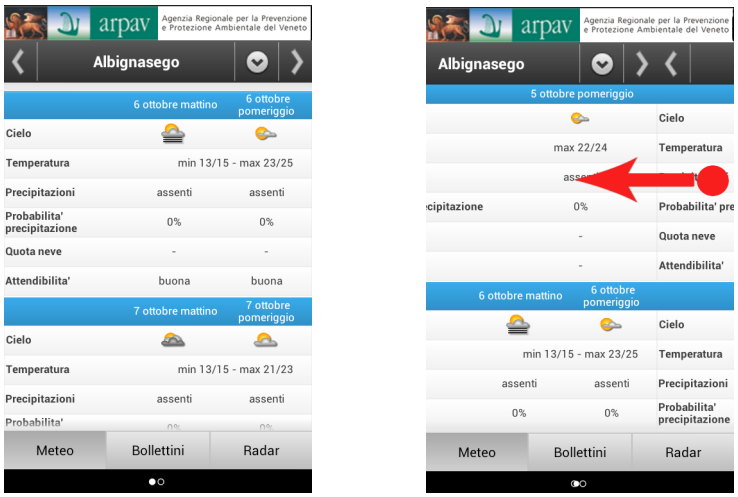


Figura 3.1: Sezione Meteo

Sezione *Bollettini*

La sezione visualizzabile in Figura ***3.2***, sono disponibili 3 bollettini: regionale, dolomiti e pianura; Essi riguardano le previsioni meteo dei prossimi quattro giorni, che sono suddivise in mattina e pomeriggio.

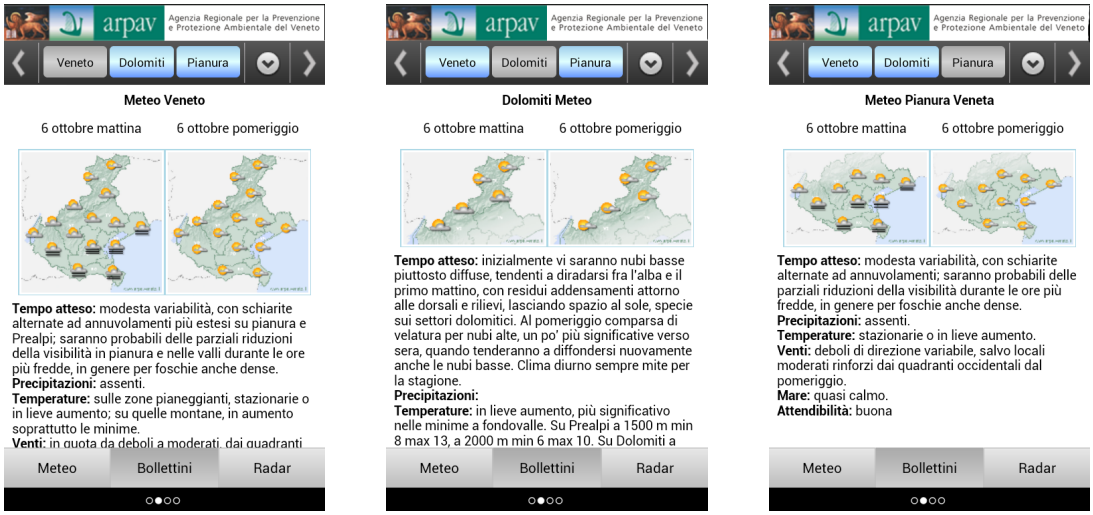


Figura 3.2: Sezione Bollettini

Sezione *Radar*

In questa sezione, illustrata in Figura ***3.3***, è possibile visualizzare la situazione meteorologica nel Veneto, monitorata attraverso i radar ARPAV di Teolo e Concordia Sagittaria.
Queste immagini, che possono essere visualizzate anche a schermo intero con possibilità di zoom e scroll, permettono di vedere la situazione delle nubi con il rispettivo grado di precipitazioni.

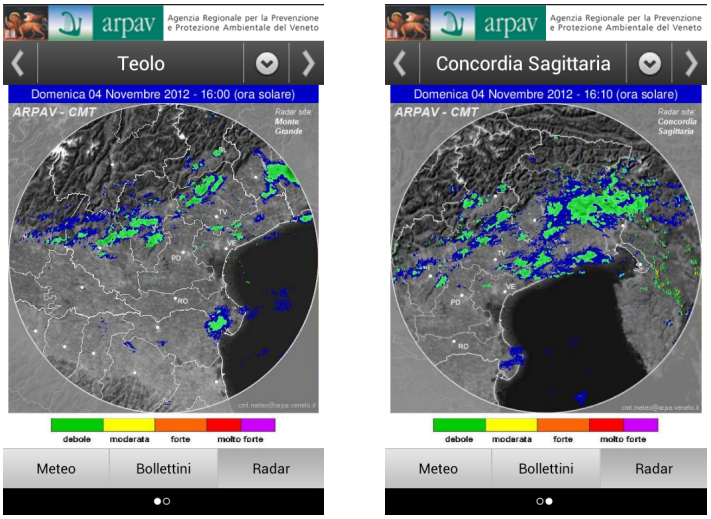


Figura 3.3: Sezione Radar

L'applicazione necessita della rete per eseguire gli aggiornamenti del meteo e per scaricare le immagini, ma può essere utilizzata anche in assenza della rete, visualizzando i dati caricati nell'ultimo aggiornamento.

3.2 Analisi delle modifiche

I miglioramenti e le modifiche apportate all'applicazione *ARPAV Meteo* sono stati inizialmente decisi in accordo con l'agenzia, sulla base di osservazioni ed esigenze maturate durante la fase di utilizzo dell'applicazione, in seguito al suo rilascio avvenuto il 10 maggio 2012.
Durante questo periodo si è potuto sperimentare ampiamente l'applicazione ricavandone importanti osservazioni; gli stessi utenti hanno potuto recensire l'app nel Market.
Pertanto, gli aggiornamenti dell'applicazione sono stati decisi sulla base di:

1. nuove esigenze da parte dell'agenzia: essa richiedeva di poter adattare l'applicazione in modo da poter essere utilizzata nei dispositivi tablet, in particolare il *Samsung Galaxy 10.1*.

2. osservazioni derivanti dall' esperienza maturata dal sottoscritto e da responsabili ARPAV attraverso l'utilizzo dell'applicazione a partire dal suo rilascio: si sono potute osservare delle lacune, risolvibili con dei miglioramenti puntuali che verranno definiti nel dettaglio durante la fase di analisi.
   Le osservazioni sono:

   - l'applicazione non presenta alcun tipo di aggiornamento automatico delle informazioni meteo;
   - in mancanza di connessione dati, l'applicazione non avvisa l'utente durante un aggiornamento;
   - la navigazione tra le sezioni dell'applicazione va migliorata e resa più intuitiva, e va migliorato il menù;
   - vanno migliorati i layout in generale;
   - esigenza di poter ingrandire le immagini radar.

3. analisi delle recensioni presenti nell'Android Market rilasciate dagli utenti, che hanno scaricato ed utilizzato l'applicazione; gli utenti hanno osservato:

   - il non corretto funzionamento dell'applicazione in dispositivi Nexus e tablet a causa dell'assenza del menù;
   - la mancanza di una navigazione non sempre intuitiva, in particolar modo per quanto riguarda gli scrolls laterali;
   - la possibilità di migliorare i layout;
   - la possibilità di introdurre un widjet dell'applicazione;

3.3 Analisi dei requisiti

In questo capitolo vengono descritti i requisiti che l'applicazione *ARPAV Meteo* dovrà soddisfare. Essi sono descritti graficamente tramite diagrammi dei casi d'uso nel linguaggio UML[g].

3.3.1 Contesto d'uso e funzionalità del prodotto

L'applicazione, già presente nell'Android Market, doveva rimanere disponibile agli utenti e doveva essere mantenuta aggiornata durante il periodo di realizzazione delle modifiche.
Non è previsto alcun tipo di limitazione di utenza per l'uso dell'applicazione in quanto questa non prevede restrizioni derivanti dal contenuto, in base alla *Classificazione dei contenuti* (`http://support.google.com/googleplay/android-developer/support/bin/answer.py?hl=it&answer=188189`) fornita da Google.

3.3.2 Requisiti software

L'applicazione ha le seguenti dipendenze dai sistemi operativi mobile Android:

- Android 1.6+

E' stato scelto di mantenere questa dipendenza, originariamente decisa dagli sviluppatori che hanno creato l'applicazione, così da poter continuare a supportare un'ampia fascia di dispositivi Android. Infatti per la realizzazione delle nuove modifiche, sono state utilizzate appositamente librerie esterne e classi delle API Google che supportavano tale dipendenza.

3.3.3 Use Case e descrizioni

Di seguito sono riportati gli Use Case (UC^g) relativi alle funzioni principali dell'applicazione. Ogni diagramma UC è integrato con una spiegazione nella quale ne verranno indicati gli attori e i rispettivi scenari nei quali essi si possono trovare.

Uc Generale

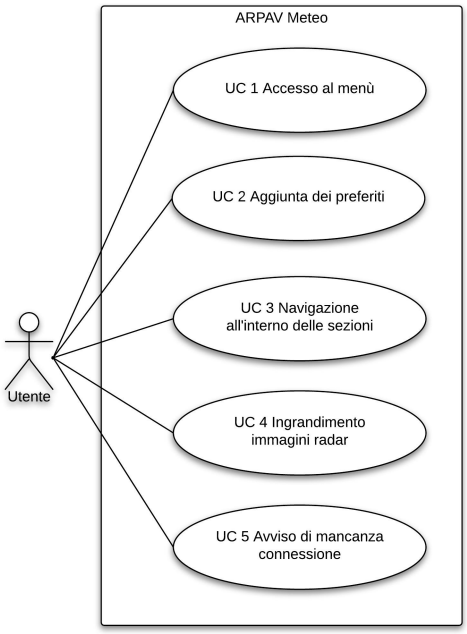


Figura 3.4: Uc generale

- **Attori coinvolti**:
  - *Utente*: utente generico utilizzatore dell'applicazione.
- **Precondizioni**:
  L'utente deve aver installato correttamente l'applicazione nel proprio dispositivo Android.
- **Postcondizione**:
  Iterazione conclusa, l'applicazione *ARPAV Meteo* è in uno stato stabile e pronta per una nuova iterazione.
- **Scenario principale**:
  - L'utente può accedere al menù dell'applicazione anche da dispositivi mobili che non prevedono il pulsante hardware di menù, come per esempio i tablet e i dispositivi Nexus. [Uc 1]

- L'utente può aggiungere i comuni ai preferiti, attraverso una scorciatoia accessibile dallo schermo. [Uc 2]
- L'utente può navigare all'interno delle sezioni, attraverso degli appositi bottoni che effettuano lo swipe laterale per lo scorrimento delle pagine. Inoltre nella sezione Bollettini l'utente può navigare all'interno delle sottosezioni Veneto, Pianura e Dolomiti, attraverso appositi bottoni. [Uc 3]
- L'utente può visualizzare le immagini radar a schermo intero ed effettuarne lo zoom. [Uc 4]
- L'utente visualizza un messaggio di mancanza di connessione del dispositivo, nel caso in cui stia eseguendo un'operazione che richieda l'utilizzo della rete. [Uc 5]

3.3.4 Classificazione dei requisiti

Di seguito viene riportato l'elenco dei requisiti da aggiungere all' applicazione *ARPAV Meteo* che sono stati definiti in accordo con il tutor dell'agenzia Dott. Luca Menini.
Essi sono organizzati in forma gerarchica e ciascuno può avere più sotto-requisiti.
Per soddisfare il requisito di livello superiore, è necessario che ciascuno dei sui sotto-requisiti sia stato realizzato.

La notazione utilizzata è la seguente:

<F|P|Q|V|I><o|d|z>-X<.Y<.Z> >

dove le seguenti sigle indicano:

F: requisito funzionale

V: requisito di vincolo

I: requisito di interfacciamento

o: obbligatorio

d: desiderabile

z: opzionale

X: requisito di primo livello

Y: sotto-requisito

Z: sotto-requisito di un sotto-requisito

I campi Y e Z possono essere assenti.

3.3.5 Requisiti funzionali

Requisiti funzionali obbligatori

Tabella 3.1: *ARPAV Meteo*: Requisiti funzionali obbligatori

Codice	Descrizione	Fonte
Fo-1	L'applicazione deve poter essere utilizzata interamente nei dispositivi che non dispongono del pulsante hardware per il menù	Proponente
Fo-1.1	L'applicazione deve permettere all'utente di accedere al menù anche nei dispositivi che non prevedono pulsante hardware del menù	Proponente
Fo-1.2	L'applicazione deve poter essere utilizzata interamente nei dispositivi tablet	Proponente
Fo-1.3	L'applicazione deve poter essere utilizzata interamente nei dispositivi Nexus	Proponente
Fo-2	L'applicazione deve permettere all'utente di aggiungere i comuni preferiti	Proponente
Fo-2.1	L'applicazione deve disporre di un pulsante di scorciatoia nella sezione Meteo per aggiungere i preferiti	Proponente
Fo-3	L'applicazione deve permettere all'utente di poter navigare all'interno delle sezioni attraverso pulsanti che effettuano lo swipe laterale	Proponente
Fo-4	L'applicazione deve permettere all'utente l'ingrandimento delle immagini radar	Proponente
Fo-4.1	L'applicazione deve permettere all'utente di poter visualizzare le immagini radar a schermo intero	Proponente
Fo-4.2	L'applicazione deve permettere all'utente di eseguire zoom sulle immagini radar	Proponente
Fo-5	L'applicazione deve avvisare l'utente in caso di mancanza di connessione del dispositivo alla rete internet	Proponente
Fo-5.1	L'applicazione deve avvisare l'utente in caso di mancanza di connessione del dispositivo alla rete internet, in caso esso stia eseguendo un'operazione che richieda la connessione internet	Proponente

3.3.6 Requisiti di vincolo

Requisiti di vincolo obbligatori

Tabella 3.2: *ARPAV Meteo*: Requisiti di vincolo obbligatori

Codice	Descrizione	Fonte
Vo-1	L'applicazione deve essere sempre connessa alla rete per poter visualizzare i dati aggiornati	Interno

3.3.7 Requisiti di interfacciamento

Requisiti di interfacciamento obbligatori

Tabella 3.3: *ARPAV Meteo*: Requisiti di interfacciamento obbligatori

Codice	Descrizione	Fonte

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

Io-1	L'applicazione deve avere un'interfaccia fluida e facilmente utilizzabile	Proponente
Io-1.1	L'applicazione deve avere dei layout fluidi ed intuibili all'utilizzo dell'utente	Proponente
Io-1.2	Nella sezione Bollettini l'applicazione deve permettere all'utente di visualizzare in quale sottosezione si trova	Proponente

3.3.8 Tracciamento

Di seguito sono riportate le tabelle di tracciamento tra Use Case - requisito e viceversa.
I requisiti di interfacciamento non sono stati tracciati da alcun Use Case, in quanto non sono rappresentabili da nessuno di essi, ma derivano dalle richieste del proponente o da decisioni interne.

Tracciamento Use Case - Requisiti

Tabella 3.4: *ARPAV Meteo*: Tracciamento Use Case - Requisiti

Codice Use Case	**Nome Use Case**	**Requisito**
Uc 1	Accesso al menù	Fo-1 Fo-1.1 Fo-1.1 Fo-1.2 Fo-1.3
Uc 2	Aggiunta dei preferiti	Fo-2 Fo-2.1
Uc 3	Navigazione all'interno delle sezioni	Fd-3
Uc 4	Ingrandimento immagini radar	Fd-4 Fo-4.1 Fo-4.2
Uc 5	Avviso di mancanza connessione	Fd-5 Fo-5.1

Tracciamento Requisito - Use Case

Tabella 3.5: *ARPAV Meteo*: Tracciamento Requisito - Use Case

Codice Requisito	**Codice Use Case**
Fo-1	Uc 1
Fo-1.1	Uc 1
Fo-1.2	Uc 1
Fo-1.3	Uc 1
Fo-2	Uc 2
Fo-2.1	Uc 2
Fo-3	Uc 3
Fo-4	Uc 4
Fo-4.1	Uc 4
Fo-4.2	Uc 4
Fo-5	Uc 5
Fd-5.1	Uc 5

3.4 Organizzazione del progetto

Di seguito viene presentata la suddivisione temporale del progetto, e il modello di ciclo di vita adottato.

3.4.1 Pianificazione del lavoro

Per la realizzazione delle modifiche all'app *ARPAV Meteo*, è stata pianificata una suddivisione temporale secondo il diagramma di Gantt presentato in Figura ***3.5***: esso prevede come giorno di inizio il 12/07/2012 e termine il 10/08/2012. Complessivamente il periodo per la realizzazione degli aggiornamenti dell'app coprirà metà delle ore totali utili per lo svolgimento dello stage. Prevedendo 8 ore lavorative giornaliere, le ore totali risultano essere 160.

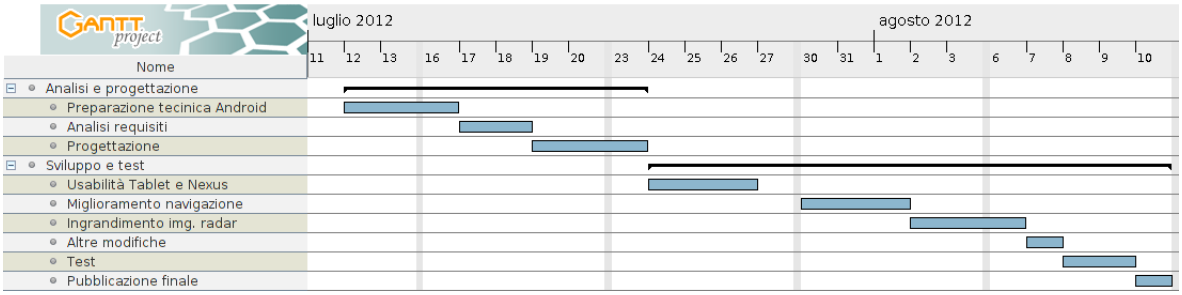


Figura 3.5: Diagramma Gantt

3.5 Dettaglio delle modifiche

Di seguito vengono analizzate nel dettaglio le modifiche apportate all'applicazione.

Usabilità in tablet e dispositivi Nexus

Come emerso dalle recensioni nell'Android Market, l'applicazione era in parte inutilizzabile in dispositivi tablet o Nexus, in quanto questi non dispongono di un apposito pulsante per accedere al menù; in questi dispositivi era pertanto impossibile aggiungere le località ai preferiti per poterne visualizzare il meteo.
Tale problema si verificava poiché a partire dalla versione di Android 3.0 (API level 11) è stata introdotta la *Action Bar*. Si tratta di una barra superiore visualizzabile nelle viste di un' applicazione, che fornisce accesso ad opzioni che variano in base al contesto, con lo scopo di aiutare l'utente nella navigazione e di includere il pulsante per accedere al menù direttamente nello schermo.
Con l'introduzione di questa barra, si è potuto dare avvio alla produzione di dispositivi mobile privi del pulsante hardware per il menù, come appunto i tablet e gli smartphone Nexus.
Infatti, prima di Android v.3.0 (Honeycomb) tutti i dispositivi Android erano dotati di un pulsante hardware per il menù. Da questa versione Android in poi, come specificato dalla stessa Google, viene rimossa qualsiasi dipendenza del sistema operativo dai tasti fisici, come già avvenuto in tablet e Nexus in cui i vecchi pulsanti hardware sono integrati nello schermo (vedi Figura ***3.6***).

Figura 3.6: Bottoni nello schermo

Per poter introdurre la *Action Bar* all'interno dell'app *ARPAV Meteo* , era però necessario modificare la versione minima di Android supportata dall'applicazione; in questa maniera si sarebbe dovuto passare dalla versione API 4.0 alla API 11.0. Tale modifica avrebbe però limitato

pesantemente il numero di dispositivi supportati, considerato che il numero di installazioni dell'app in dispositivi Android precedenti alla versione API 11.0 era più della metà. Questi dati sono visualizzabili in Figura ***3.7*** ricavata dall'*Android Console*[9], che riporta le statistiche sulle versioni Android in cui è stata installata l'applicazione.

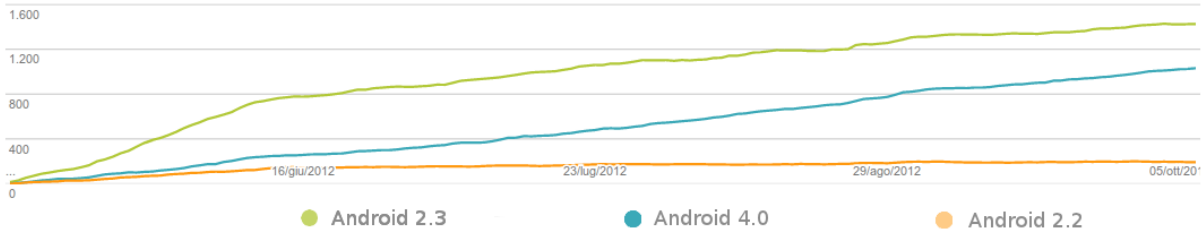


Figura 3.7: Installazioni attive per versione di Android

Non potendo quindi utilizzare la *Action Bar*, ho pertanto simulato il suo funzionamento inserendo un bottone nelle viste che fungesse da menù, come visualizzato in Figura ***3.8***.
Attraverso tale bottone è possibile aprire un'apposita tendina di menù, che è stata realizzata attraverso l'utilizzo della libreria NewQuickAction (per ulteriori informazioni fare riferimento al Capitolo ***2***), implementando le funzioni del menù *Aggiorna* e *Preferiti*.

Figura 3.8: Inserimento menù

In questo modo si può avere un accesso diretto al menù, attraverso una scorciatoia sempre visibile nello schermo e l'applicazione diviene così utilizzabile interamente anche nei dispositivi tablet e Nexus.

L'inserimento del pulsante preferiti come scorciatoia

Per agevolare l'aggiunta dei comuni ai preferiti (vedi Figura ***3.9***) ho aggiunto un pulsante di "scorciatoia" nella sezione *Meteo*, che è la sezione principale visualizzata all'avvio di *ARPAV Meteo*. In particolare questo bottone, con relativa spiegazione sovrastante, aiuta l'utente ad aggiungere i preferiti durante il suo primo utilizzo dell'applicazione.

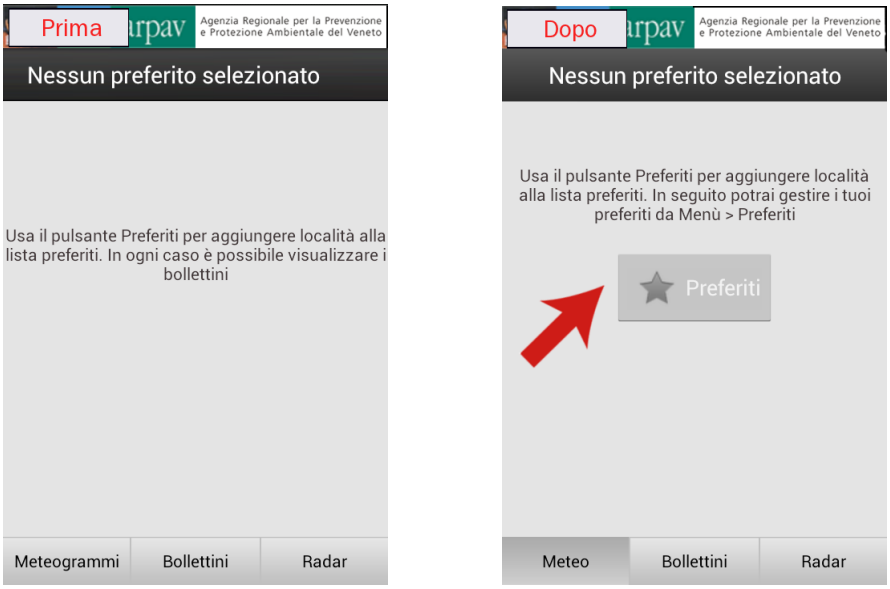


Figura 3.9: Inserimento pulsante Preferiti

Il miglioramento della navigazione tra le sezioni

Le tre sezioni dell'applicazione, *Meteo*, *Bollettini* e *Radar* dispongono di scroll laterale, ottenibile attraverso *swipe*, che consiste nel movimento del dito da sinistra verso destra o viceversa. In questo modo all'interno di una sezione è possibile navigare tra le varie sottosezioni.
Inizialmente questa navigazione era intuibile solamente attraverso un indicatore di pagina (vedi Figura ***3.10***) presente in ogni sezione, il quale però non era facilmente visibile e intuibile.

Figura 3.10: Indicatore di pagine

Per agevolare lo scroll laterale e renderlo più intuitivo, ho inserito degli appositi pulsanti in ogni sezione (vedi Figura ***3.11***), mediante i quali è possibile spostarsi tra le sottosezioni con maggiore facilità.

Figura 3.11: Bottoni di "swipe"

Il miglioramento della navigazione nella sezione *Bollettini*

La sezione *Bollettini* presenta nella barra superiore tre pulsanti, attraverso i quali si può accedere alle sottosezioni dei bollettini del Veneto, delle Dolomiti e della Pianura.
Tali bottoni però oltre ad essere poco leggibili, non evidenziavano la posizione in cui ci si trovava, rendendo la navigazione non immediata e disorientando l'utente.
Per questo ho deciso di migliorare la grafica e di disattivare il bottone relativo al bollettino corrente, rendendolo non cliccabile e di colore differente rispetto agli altri due (vedi Figura ***3.12***).

Figura 3.12: Pulsanti sezione *Bollettini*

L'ingrandimento delle immagini radar

Come richiesto dall'agenzia ARPAV, le immagini radar non erano visibili a schermo intero ma come normali immagini statiche.
Pertanto ho reso queste immagini cliccabili attraverso l'inserimento di un "listener",

```
radarImage.setOnClickListener(new View.OnClickListener(){});
```

In questo modo il tocco dell'utente sull'immagine radar viene captato e viene eseguito il codice che avvia un' apposita *activity*, per la visualizzazione dell'immagine a schermo intero.
L'immagine è inoltre ridimensionabile attraverso il *pinch to zoom*[g], così l'utente può visualizzare più nel dettaglio la localizzazione delle nubi e la concentrazione della pioggia (vedi Figura ***3.13***).

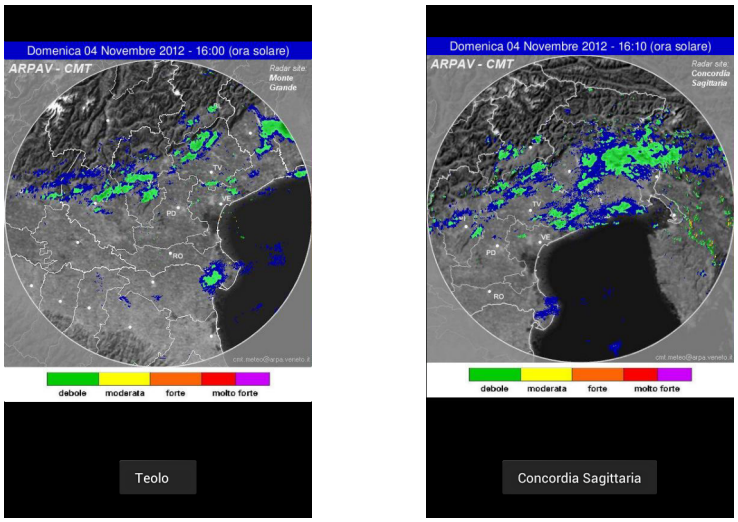


Figura 3.13: Immagini radar a schermo intero

Avviso di mancanza di connessione

L'applicazione è in grado di funzionare anche in mancanza di connessione, e al suo avvio visualizza le previsioni meteo caricate durante l'ultimo aggiornamento dei dati.
Nel caso in cui però l'utente volesse aggiornare le previsioni meteo, l'applicazione originariamente non lo avvisava dell'eventuale mancanza di una qualsiasi connessione alla rete, facendo attendere inutilmente.
Ho pertanto aggiunto un apposito metodo che viene invocato ogni qual volta l'applicazione esegua un'operazione che necessiti dell'utilizzo della rete, verificandone il suo stato. Il metodo ritorna un il booleano "true" se il dispositivo presenta una connessione alla rete, altrimenti ritorna "false".

```
public static boolean isNetworkAvailable(Context context) {
    ConnectivityManager connectivityManager = (ConnectivityManager)
        context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager
        .getActiveNetworkInfo();

```

```
    if (activeNetworkInfo != null)
        return activeNetworkInfo.isConnected();
    return false;

}
```

In base a quanto ritornato da questo metodo, vengono eseguite operazioni differenti: se la connessione è disponibile, l'operazione richiesta potrà proseguire, altrimenti viene visualizzato sullo schermo un messaggio di non connessione alla rete. In particolare questa funzionalità è risultata molto utile nella sezione radar: infatti durante il caricamento delle riprese radar veniva visualizzata un'immagine di sfondo che comunicava all'utente di attendere il download delle stesse. Tale immagine compariva indipendentemente dalla presenza o meno delle connessione alla rete internet, in assenza della quale l'utente attendeva inutilmente. Ho pertanto creato due differenti immagini di sfondo relative alla presenza o meno della connessione.

Altre modifiche

Le ultime modifiche apportate all'applicazione riguardano alcuni accorgimenti per rendere la grafica più accattivante. Da primo ho aggiornato le icone nell'elenco dei preferiti e dei comuni, che a mio avviso oltre ad essere poco intuitive, erano spente (vedi Figura ***3.14***).

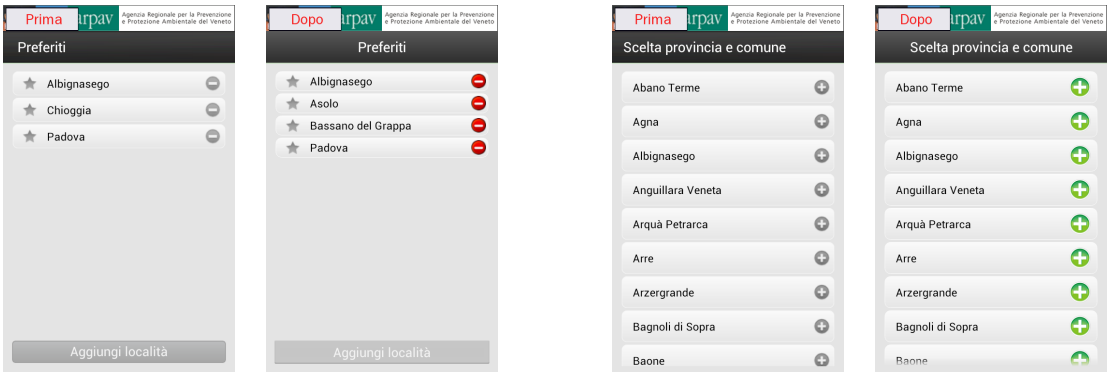


Figura 3.14: Modifiche icone

Ho posto inoltre attenzione alla grafica dei bottoni principali dell'applicazione, rendendoli più accattivanti creando degli appositi stili di colori che vengono applicati dinamicamente al variare dello stato del bottone: attivo, disattivo o cliccato (vedi Figura ***3.15***).

3.6 Pubblicazione nell'Android Market

Durante il periodo di sviluppo delle modifiche, al completamento di una serie di obiettivi, ho eseguito progressivamente l'aggiornamento dall'app nell'Android Market, passando così dalla versione 2.0 alla versione 2.5. In questo modo si è evitato di effettuare un unico aggiornamento finale dell'app, permettendo così di abbassare i rischi di fallimento che questo avrebbe potuto comportare; attraverso il rilascio di nuove versioni corrispondenti ad una serie limitata di modifiche, è più facile tenere sotto controllo eventuali problemi che l'aggiornamento può portare. Infatti, nonostante i numerosi tests pratici eseguiti sistematicamente in diversi dispositivi con differenti versioni di Android (per l'elenco dei dispositivi usati per i test pratici fare riferimento al Capitolo ***2***), è possibile che vi siano variazioni nei comportamenti e visualizzazioni dell'app date le numerose versioni Android disponibili in commercio.

Figura 3.15: Grafica pulsanti

3.7 Resoconto requisiti

Di seguito vengono elencati i requisiti prefissati, indicando per ognuno di essi se è stato soddisfatto o meno.

Tabella 3.6: *ARPAV Meteo*: Resoconto requisiti

Codice	**Descrizione**	**Esito**
Fo-1	L'applicazione deve poter essere utilizzata interamente nei dispositivi che non dispongono del pulsante hardware per il menù	Soddisfatto
Fo-1.1	L'applicazione deve permettere all'utente di accedere al menù anche nei dispositivi che non prevedono pulsante hardware del menù	Soddisfatto
Fo-1.2	L'applicazione deve poter essere utilizzata interamente nei dispositivi tablet	Soddisfatto
Fo-1.3	L'applicazione deve poter essere utilizzata interamente nei dispositivi Nexus	Soddisfatto
Fo-2	L'applicazione deve permettere all'utente di aggiungere i comuni preferiti	Soddisfatto
Fo-2.1	L'applicazione deve disporre di un pulsante di scorciatoia nella sezione Meteo per aggiungere i preferiti	Soddisfatto
Fo-3	L'applicazione deve permettere all'utente di poter navigare all'interno delle sezioni attraverso pulsanti che effettuano lo swipe laterale	Soddisfatto
Fo-4	L'applicazione deve permettere all'utente l'ingrandimento delle immagini radar	Soddisfatto
Fo-4.1	L'applicazione deve permettere all'utente di poter visualizzare le immagini radar a schermo intero	Soddisfatto
Fo-4.2	L'applicazione deve permettere all'utente di eseguire zoom sulle immagini radar	Soddisfatto
Fo-5	L'applicazione deve avvisare l'utente in caso di mancanza di connessione del dispositivo alla rete internet	Soddisfatto

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

Fo-5.1	L'applicazione deve avvisare l'utente in caso di mancanza di connessione del dispositivo alla rete internet, in caso esso stia eseguendo un'operazione che richieda la connessione internet	Soddisfatto

4 ARPAV Idro

In questo capitolo viene presentato e descritto il lavoro svolto durante la seconda parte dello stage, nella quale è stata progettata e sviluppata l'applicazione *ARPAV Idro*.

4.1 Descrizione

Tra le applicazioni da realizzare nel progetto *App ARPAV*, rientrava anche lo sviluppo di un'app che consentisse l'accesso ai dati delle misure idrometriche e pluviometriche degli ultimi tre giorni, relative a una serie di stazioni di rilevamento ARPAV distribuite sul territorio Veneto. Tale esigenza dell'agenzia è stata colmata durante la seconda metà dello stage, con la realizzazione e pubblicazione dell'app *ARPAV Idro*. L'obbiettivo è quello di poter dare ai cittadini informazioni immediate (dati in diretta) sul livello idrico dei fiumi e sulle quantità di precipitazioni cadute in una determinata zona. Questo servizio risulta essere utile soprattutto nei momenti di piogge intense ed alluvioni.

All'avvio dell'applicazione, dopo uno Splash Screen[g] iniziale, la navigazione parte dalla mappa della Regione Veneto in cui, in base alle coordinate geografiche, sono state localizzate le stazioni di rilevamento. Ogni stazione, in base ai sensori di rilevamento in essa presenti, è identificata da una rispettiva icona: le stazioni idrometriche che monitorano il livello dei bacini sono identificate da icone di color rosso, quelle pluviometriche che misurano il quantitativo di pioggia caduta da simboli di color blu, e le stazioni che prevedono entrambi i sensori di monitoraggio presentano simboli di color verde. La mappa consente all'utente di individuare la stazione d'interesse e alla selezione compare il nome della stazione e il bacino, come visualizzato in Figura ***4.1***.

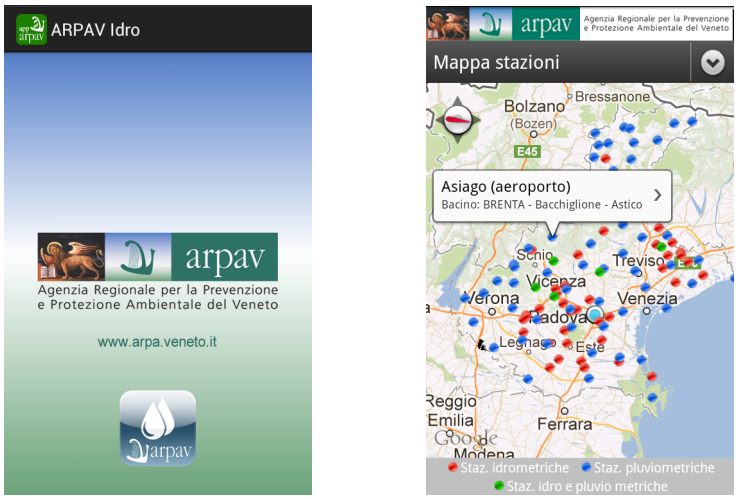


Figura 4.1: Avvio

Selezionando la stazione, dopo un veloce caricamento ed elaborazione dei dati, viene visualizzato il grafico riportante i dati relativi ad essa. Nel caso in cui la stazione selezionata presenti entrambi i sensori di rilevamento, viene chiesto all'utente quali valori desidera visualizzare. I grafici riportano nell'asse orizzontale il tempo ad intervalli orari, mentre nell'asse verticale sono riportati i valori in metri per i rilevamenti idrici e in millimetri per i rilevamenti pluviometrici. I grafici sono fluidi e scorrono in entrambi gli assi, inoltre possono essere zoomati per visualizzare più nel dettaglio i valori riportati. Per la realizzazione dei grafici, l'applicazione sfrutta dei dati in formato XML, realizzati da un sistema di monitoraggio proprio dell'agenzia ARPAV. In Figura ***4.2*** vengono illustrati i grafici.

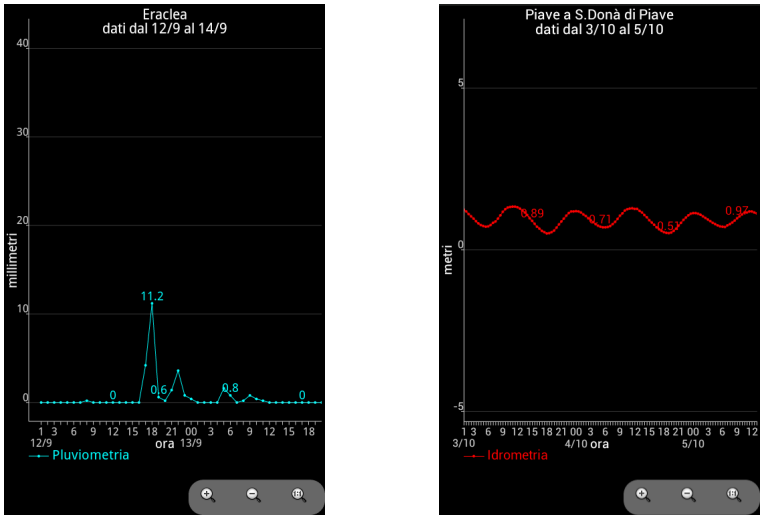


Figura 4.2: Grafici

Per la creazione della mappa, l'applicazione utilizza le apposite librerie di *Google Maps* per Android, e attraverso il sistema di localizzazione tramite GPS o rete mobile/wi-fi, permette di individuare nella mappa la propria posizione, aggiornandola se in movimento. Nel caso in cui l'utente nelle impostazioni del dispositivo non abbia acconsentito il rilevamento della propria localizzazione, l'app avvisa l'utente che nella mappa non potrà essere visualizzata la propria posizione. In seguito comunque, l'utente potrà in ogni momento modificare tale impostazione utilizzando la scorciatoia presente nel menù dell'applicazione (vedi Figura ***4.3***) **Attiva localizzazione**.
Grazie alla funzione di localizzazione è possibile navigare nella mappa e passare velocemente alla propria posizione zoomata, attraverso la funzione del menù **Mia posizione**. Ciò risulta utile per poter scegliere in maniera veloce le stazioni più vicine.
Nel caso in cui ci si trovi al di fuori dei confini della Regione Veneto, dove non sono presenti stazioni di monitoraggio ARPAV e l'utente richieda di visualizzare la propria posizione, un messaggio di avviso viene notificato e l'utente può scegliere se continuare o annullare l'operazione.
Dal menù è inoltre presente la funzione **Aggiorna**, utile in caso di problemi di connessione, che permette di ricaricare la mappa e le stazioni.

Figura 4.3: Menù

Come intuibile l'applicazione necessita di connessione ad internet quasi costante per poter accedere ai dati richiesti dall'utente e per poter caricare la mappa necessaria alla navigazione.

4.2 Analisi dei requisiti

In questa sezione vengono descritti i requisiti che l'applicazione *ARPAV Idro* dovrà soddisfare. Essi sono descritti graficamente tramite diagrammi dei casi d'uso nel linguaggio UML[g].

4.2.1 Contesto d'uso e funzionalità del prodotto

L'applicazione dovrà essere disponibile nell' *Android Market* e potrà essere scaricata ed utilizzata gratuitamente, da qualsiasi tipo di utenza in possesso di un dispositivo Android.
Non è previsto alcun tipo di limitazione di utenza per l'uso dell'applicazione, in quanto questa non prevede restrizioni derivanti dal contenuto, in base alla *Classificazione dei contenuti* (`http://support.google.com/googleplay/android-developer/support/bin/answer.py?hl=it&answer=188189`) fornita da Google.

4.2.2 Requisiti software

L'applicazione ha le seguenti dipendenze dai sistemi operativi mobile Android:

- Android 2.2+

E' stato scelto di supportare come versione minima Android 2.2 per evitare limitazioni nell'app; da questa versione di Android infatti nelle API Google sono state introdotte molte funzionalità aggiuntive, che potrebbero essere necessarie durante lo sviluppo dell'applicazione o durante una sua fase di manutenzione e upgrade in seguito al rilascio.
Alcune tra le principali funzionalità sono la ricezione di notifiche Push e il multi-touch. Va ricordato inoltre che è molto bassa (inferiore al 5%) la percentuale di dispositivi Android presenti nel mercato con una versione inferiore alla 2.2, e tale valore è destinato a diminuire progressivamente. Tale situazione è riassunta nel grafico in Figura *4.4*.

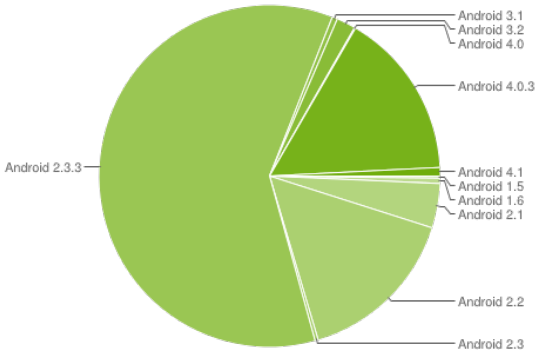


Figura 4.4: Diffusione versioni Android

4.2.3 Use Case e descrizioni

Di seguito sono riportati gli Use Case (UC[g]) relativi alle funzioni principali dell'applicazione. Ogni diagramma UC è integrato da una spiegazione nella quale vengono indicati gli attori e gli scenari nei quali essi si possono trovare.

Uc Generale

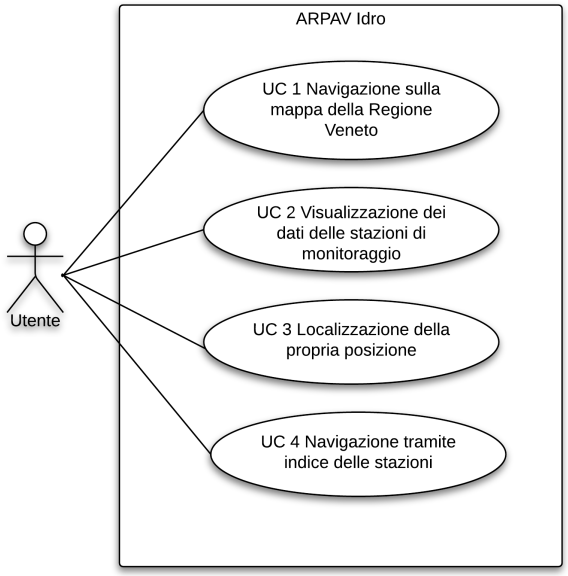


Figura 4.5: Uc generale

- **Attori coinvolti**:
  - *Utente*: utente generico utilizzatore dell'applicazione.
- **Precondizioni**:
  L'utente deve aver installato correttamente l'applicazione nel proprio dispositivo Android, e deve essere collegato alla rete internet.
- **Postcondizione**:
  Interazione conclusa, l'applicazione *ARPAV Idro* è in uno stato stabile e pronta per una nuova interazione.
- **Scenario principale**:
  - L'utente visualizza la mappa della Regione Veneto e vi naviga al suo interno. [Uc 1]
  - L'utente dopo aver scelto una stazione di interesse, ne visualizza i dati monitorati. [Uc 2]
  - L'utente visualizza la propria posizione sulla mappa. [Uc 3]
  - L'utente visualizza l'indice ordinato delle stazioni di monitoraggio. [Uc 4]

Uc 1: Navigazione sulla mappa della Regione Veneto

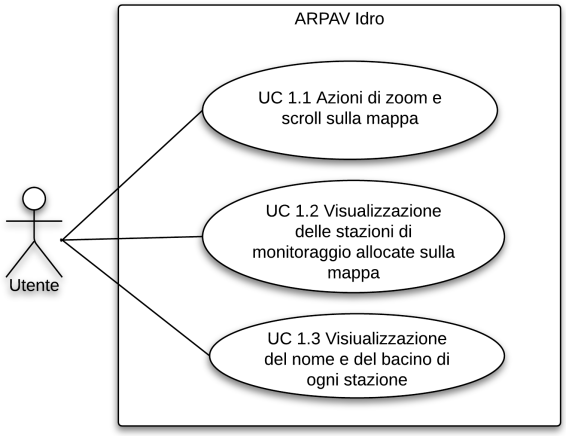


Figura 4.6: Uc 1: Navigazione sulla mappa della Regione Veneto

- **Attori coinvolti**:
  - *Utente*: utente generico utilizzatore dell'applicazione.
- **Precondizioni**:
  L'applicazione è in esecuzione e l'utente può interagire con essa.
- **Postcondizione**:
  Interazione conclusa, l'applicazione *ARPAV Idro* è in uno stato stabile e pronta per una nuova interazione.
- **Scenario principale**:
  - L'utente naviga sulla mappa eseguendo operazioni di zoom e scroll. [Uc 1.1]
  - L'utente visualizza con icone differenti le stazioni di monitoraggio allocate sulla mappa. [Uc 1.2]
  - L'utente cliccando l'icona di una stazione ne può visualizzare il nome e il rispettivo bacino di appartenenza. [Uc 1.3]

Uc 2: Visualizzazione dei dati delle stazioni di monitoraggio

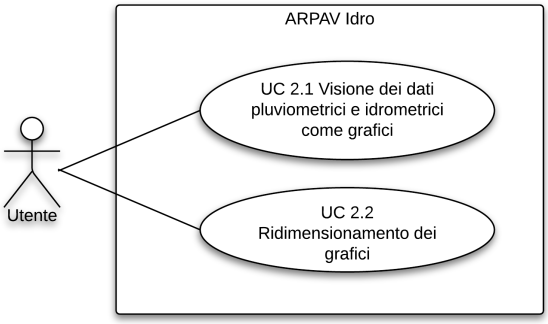


Figura 4.7: Uc 2: Visualizzazione dei dati delle stazioni di monitoraggio

- **Attori coinvolti**:
  - *Utente*: utente generico utilizzatore dell'applicazione.
- **Precondizioni**:
  L'applicazione è in esecuzione e l'utente può interagire con essa.
- **Postcondizione**:
  Interazione conclusa, l'applicazione *ARPAV Idro* è in uno stato stabile e pronta per una nuova interazione.
- **Scenario principale**:
  - L'utente può visualizzare in maniera veloce, (sotto forma di grafico) i livelli idrici e pluviometrici rilevati negli ultimi 3 giorni nelle proprie stazioni preferite. [Uc 2.1]
  - L'utente esegue operazioni di zoom e scroll sui grafici per vederne i dati più nel dettaglio. [Uc 2.2]

Uc 3: Localizzazione della propria posizione

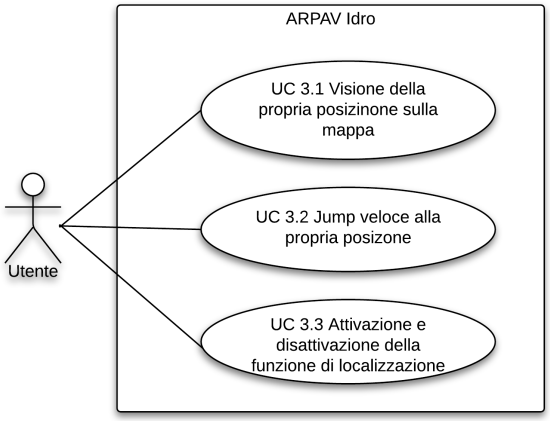


Figura 4.8: Uc 3: Localizzazione della propria posizione

- **Attori coinvolti**:
  - *Utente*: utente generico utilizzatore dell'applicazione.
- **Precondizioni**:
  L'applicazione è in esecuzione e l'utente può interagire con essa.
- **Postcondizione**:
  Interazione conclusa, l'applicazione *ARPAV Idro* è in uno stato stabile e pronta per una nuova interazione.
- **Scenario principale**:
  - L'utente visualizza sulla mappa la propria posizione secondo un' apposita icona. [Uc 3.1]
  - L'utente può passare velocemente alla propria posizione zoomata, attraverso l'apposita scorciatoia accessibile da menù. [Uc 3.2]
  - L'utente modifica le impostazioni sulla localizzazione, attivandola o disattivandola attraverso l'apposita scorciatoia del menù. [Uc 3.3]

4.2.4 Classificazione dei requisiti

Di seguito viene riportato l'elenco dei requisiti dell'app *ARPAV Idro* che sono stati definiti in accordo con il tutor dell'agenzia Dott. Luca Menini, in seguito ad una fase preliminare di analisi. Essi sono organizzati in forma gerarchica e ciascuno può avere più sotto-requisiti.
Per soddisfare il requisito di livello superiore, è necessario che ciascuno dei sui sotto-requisiti sia stato realizzato.

La notazione utilizzata è la seguente:

<F|P|Q|V|I><o|d|z>-X<.Y<.Z> >

dove le seguenti sigle indicano:

F: requisito funzionale

V: requisito di vincolo

I: requisito di interfacciamento

o: obbligatorio

d: desiderabile

z: opzionale

X: requisito di primo livello

Y: sotto-requisito

Z: sotto-requisito di un sotto-requisito

I campi Y e Z possono essere assenti.

4.2.5 Requisiti funzionali

Requisiti funzionali obbligatori

Tabella 4.1: *ARPAV Idro*: Requisiti funzionali obbligatori

Codice	Descrizione	Fonte
Fo-1	L'applicazione deve permettere all'utente di visualizzare la mappa della Regione Veneto	Proponente
Fo-1.1	L'applicazione deve permettere all'utente di navigare sulla mappa della Regione Veneto	Proponente
Fo-1.1.1	L'applicazione deve permettere all'utente di eseguire azioni di zoom e scroll sulla mappa	Proponente
Fo-2	L'applicazione deve permettere all'utente di visualizzare le stazioni di monitoraggio idrometriche e pluviometriche sulla mappa della Regione Veneto	Proponente
Fo-2.1	L'applicazione deve permettere all'utente di visualizzare sulla mappa le stazioni di monitoraggio mediante icone differenti in base al tipo di stazione	Proponente
Fo-3	L'applicazione deve permettere all'utente di poter scegliere una stazione di monitoraggio a partire dalla mappa della Regione Veneto, in cui sono state localizzate le stazioni di rilevamento	Proponente
Fo-3.1	L'applicazione deve permettere all'utente di poter cliccare l'icona di una stazione di monitoraggio presente sulla mappa	Proponente
Fo-3.2	L'applicazione deve permettere all'utente di visualizzare le informazioni relative ad una stazione: paese in cui si trova e bacino di appartenenza	Proponente
Fo-4	L'applicazione deve permettere all'utente di visualizzare i dati monitorati dalle stazioni	Proponente
Fo-4.1	L'applicazione deve permettere all'utente di visualizzare i dati relativi agli ultimi tre giorni	Proponente
Fo-4.2	L'applicazione deve permettere all'utente di visualizzare i dati sotto forma di grafico	Proponente
Fo-4.2.1	L'applicazione deve permettere all'utente di visualizzare i dati idrometrici sotto forma di grafico	Proponente
Fo-4.2.2	L'applicazione deve permettere all'utente di visualizzare i dati pluviometrici sotto forma di grafico	Proponente
Fo-4.2.3	L'applicazione deve permettere all'utente il ridimensionamento dei grafici	Proponente
Fo-5	L'applicazione deve avvisare l'utente in caso di mancanza di connessione del dispositivo alla rete internet	Proponente

Requisiti funzionali desiderabili

Tabella 4.2: *ARPAV Idro*: Requisiti funzionali desiderabili

Codice	Descrizione	Fonte
Fd-1	L'applicazione deve permettere all'utente la visione della propria posizione sulla mappa	Proponente
Fd-1.1	L'applicazione deve permettere all'utente la visualizzazione della propria posizione sulla mappa mediante apposita icona	Proponente
Fd-1.2	L'applicazione deve permettere all'utente la possibilità di visualizzare velocemente la propria posizione zoomata e centrata nella mappa	Proponente

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

Fd-1.3	L'applicazione deve permettere all'utente la possibilità di attivare o disattivare la funzione di rilevamento della propria posizione nel dispositivo mobile	Proponente

Requisiti funzionali opzionali

Tabella 4.3: *ARPAV Idro*: Requisiti funzionali opzionali

Codice	Descrizione	Fonte
Fz-1	L'applicazione deve permettere all'utente la visualizzazione mediante indice di tutti i bacini idrici monitorati	Proponente
Fz-2	L'applicazione deve permettere all'utente la visualizzazione mediante indice di tutte le stazioni di monitoraggio di un particolare bacino idrico	Proponente

4.2.6 Requisiti di vincolo

Requisiti di vincolo obbligatori

Tabella 4.4: *ARPAV Idro*: Requisiti di vincolo obbligatori

Codice	Descrizione	Fonte
Vo-1	L'applicazione deve essere sempre connessa alla rete per poter accedere ai dati delle stazioni di monitoraggio e per caricare correttamente la mappa	Interno

4.2.7 Requisiti di interfacciamento

Requisiti di interfacciamento obbligatori

Tabella 4.5: *ARPAV Idro*: Requisiti di interfacciamento obbligatori

Codice	Descrizione	Fonte
Io-1	L'applicazione deve avere un'interfaccia fluida e facilmente utilizzabile	Proponente
Io-2	L'applicazione deve prevedere la visualizzazione della mappa della Regione Veneto con allocate le stazioni di monitoraggio	Proponente
Io-3	L'applicazione deve prevedere la visualizzazione dei dati di una stazione, attraverso un'interfaccia che ne visualizza i valori mediante grafici	Proponente

4.2.8 Tracciamento

Di seguito sono riportate le tabelle di tracciamento tra Use Case - requisito e viceversa.
I requisiti di interfacciamento non sono stati tracciati da alcun Use Case, in quanto non sono rappresentabili da nessuno di essi ma derivano dalle richieste del proponente o da decisioni interne.

Tracciamento Use Case - Requisiti

Tabella 4.6: *ARPAV Idro*: Tracciamento Use Case - Requisiti

Codice Use Case	**Nome Use Case**	**Requisito**
Uc 1	Navigazione sulla mappa della Regione Veneto	Fo-1 Fo-1.1 Fo-1.1.1
Uc 1.1	Azioni di zoom e scroll sulla mappa	Fo-1.1
Uc 1.2	Visualizzazione delle stazioni di monitoraggio allocate sulla mappa	Fo-2 Fo-2.1
Uc 1.3	Visualizzazione del nome e del bacino di ogni stazione	Fo-3.2
Uc 2	Visualizzazione dei dati delle stazioni di monitoraggio	Fo-4 Fo-4.1 Fo-4.2
Uc 2.1	Visualizzazione dei dati idrometrici e pluviometrici sotto forma di grafici	Fo-4 Fo-4.2 Fo-4.2.1 Fo-4.2.2
Uc 2.2	Ridimensionamento dei grafici	Fo-4.2.3
Uc 3	Localizzazione della propria posizione	Fd-1
Uc 3.1	Visione della propria posizione sulla mappa	Fd-1.1
Uc 3.2	Jump veloce alla propria posizione	Fd-1.2
Uc 3.3	Attivazione e disattivazione della funzione di localizzazione	Fd-1.3

Tracciamento Requisito - Use Case

Tabella 4.7: *ARPAV Idro*: Tracciamento Requisito - Use Case

Codice Requisito	**Codice Use Case**
Fo-1	Uc 1
Fo-1.1	Uc 1.1
Fo-1.1.1	Uc 1.1
Fo-2	Uc 1.2
Fo-3	Uc 1.2
Fo-3.2	Uc 1.3
Fo-4	Uc 2
Fo-4.1	Uc 2
Fo-4.2	Uc 2.1
Fo-4.2.1	Uc 2.1
Fo-4.2.2	Uc 2.1
Fo-4.2.3	Uc 2.2
Fd-1	Uc 3.1
Fd-1.2	Uc 3.2
Fd-1.3	Uc 3.3

4.3 Organizzazione del progetto

Di seguito viene presentata la suddivisione temporale del progetto e il modello di ciclo di vita adottato.

4.3.1 Pianificazione del lavoro

Per la realizzazione dell'app *ARPAV Idro*, è stata pianificata una suddivisione temporale secondo il diagramma di Gantt presentato in Figura ***4.9***, che vede come giorno di inizio il 20/08/2012 e termine il 14/09/2012. Complessivamente il periodo per la realizzazione dell'app occupa la metà delle ore totali utili per lo svolgimento dello stage. Prevedendo 8 ore lavorative giornaliere, il monte ore complessivo risulta essere di 160 ore.

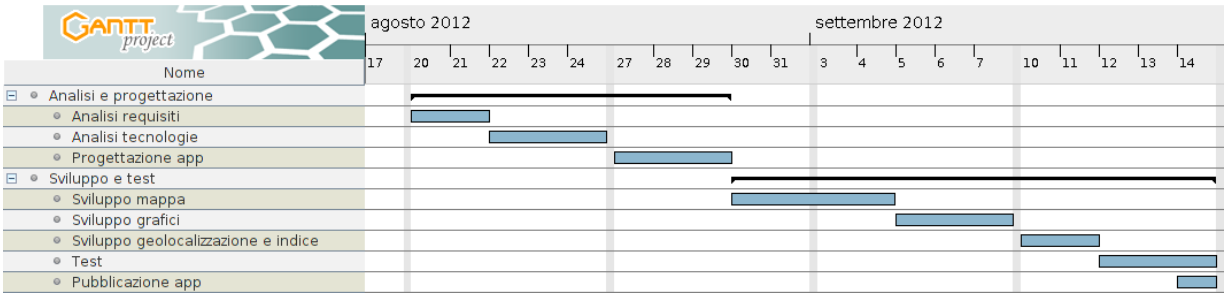


Figura 4.9: Diagramma Gantt

4.3.2 Modello di ciclo di vita

Per la realizzazione dell'app *ARPAV Idro* è stato adottato il modello di ciclo di vita incrementale, descritto in Figura ***4.10***.
Secondo il modello è prevista inizialmente la stesura dei requisiti e la realizzazione dell'architettura ad alto livello. E' stato possibile adottare questo modello di ciclo di vita poiché fin da subito con il tutor dell'agenzia sono stati delineati in modo chiaro i requisiti desiderati.
Il modello prevede uno sviluppo suddiviso per incrementi, ciascuno dei quali comporta la realizzazione di una parte delle funzionalità richieste, diminuendo così i rischi di fallimento.
I primi incrementi hanno lo scopo di soddisfare i requisiti fondamentali delineati in fase di analisi, i quali devono rimanere stabili. Vengono così realizzate dapprima le funzionalità basi dell'applicazione, e progressivamente nelle successive iterazioni sono soddisfatti i requisiti desiderabili e obbligatori.

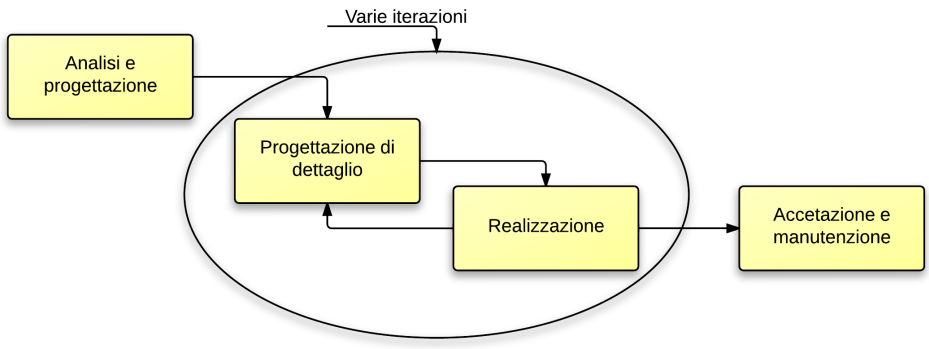


Figura 4.10: Modello incrementale

Iterazioni

In base al ciclo di vita adottato, per la realizzazione dell'app sono state prefissate tre iterazioni che vengono di seguito descritte:

- **I° iterazione**:
  l'iterazione prevede la progettazione di dettaglio e sviluppo dell'activity principale, che deve permettere la visualizzazione delle stazioni sulla mappa della Regione Veneto;
- **II° iterazione**:
  l'iterazione prevede la progettazione di dettaglio e lo sviluppo dell'activity per la visualizzazione dei grafici che riportano i dati monitorati da ogni stazione;
- **III° iterazione**:
  l'uscita dell'iterazione precedente dovrà mostrare l'app funzionante che soddisfa tutti i requisiti obbligatori. In quest'ultima iterazione verranno realizzati i requisiti desiderabili, che prevedono la geo-localizzazione sulla mappa visualizzata e i requisiti opzionali. Verranno inoltre realizzati i tests e la pubblicazione dell'app nell'Android Market.

4.3.3 Analisi dei rischi

In relazione al modello di ciclo di vita adottato, è possibile sviluppare con sicurezza un'app funzionante in uscita dalla seconda iterazione; va sottolineato però che eventuali ritardi delle prime due iterazioni implicano ritardi non recuperabili nell'ultima iterazione, ovvero una diminuzione del completamento dei requisiti opzionali. I rischi maggiori legati alla realizzazione dell'app sono dovuti in prevalenza alla mancanza di esperienza di sviluppo in ambito Android. Tale inesperienza può comportare tra l'altro errori nelle scelte architetturali, oppure errori nelle stime temporali delle varie fasi di sviluppo dell'app.
Di seguito vengono analizzati i maggiori rischi di fallimento individuati:

- **Ritardi**
  il rischio di ritardo potrebbe verificarsi come conseguenza strettamente connessa al verificarsi di altri rischi. Tra questi, i più importanti sono:
  - **Inesperienza nell'ambito Android**:
    La mancanza di esperienza del tirocinante, in ambito di progettazione e sviluppo di app per la piattaforma Android, può generare ritardi nelle prime due iterazioni; le quali prevedono lo sviluppo nel dettaglio e la realizzazione delle activity principali dell'app.
    Questo rischio presenta un'alta probabilità di occorrenza.
  - **Inadeguatezza delle tecnologie-librerie scelte**:
    Una scelta non accurata delle librerie da utilizzare potrebbe comportare ritardi. Tra queste troviamo per esempio le librerie per la visualizzazione della mappa e per la creazione dei grafici. Come mitigazione del rischio è prevista in fase di progettazione un'attenta analisi delle possibili tecnologie da adottare, così da poter effettuare una scelta il più possibile adeguata.
    Questo rischio presenta una media probabilità di occorrenza.
- **Malfunzionamento dell'app dovuto a dipendenze esterne**:
  Come uniche dipendenze esterne vi è il caricamento della mappa in tempo reale e la ricezione dei dati in formato XML attraverso i server ARPAV.
  In caso di malfunzionamento o spegnimento dei server ARPAV o modifiche ai file XML utilizzati dall'applicazione, si potrebbe verificare un suo scorretto funzionamento.
  Per mitigare questo rischio, l'app verrà realizzata in modo tale da poter gestire al meglio queste eccezioni; infatti una eventuale modifica radicale dei file XML e in particolare dei "tags" in essi contenuti, potrebbe comportare la non usabilità dell'applicazione. Questo rischio presenta una probabilità medio-bassa.

Ritardi verificatisi

Per la realizzazione dell'applicazione sono state eseguite tre iterazioni che hanno permesso la realizzazione di tutti i requisiti obbligatori e desiderabili. Durante la seconda iterazione si è verificato un lieve ritardo, dovuto alla necessità di dover adattare i grafici per la visualizzazione dei dati, a differenti dimensioni di schermi per smartphones e tablets. Questa personalizzazione ha richiesto infatti costanti verifiche pratiche dell'app, provocando una posticipazione della conclusione della seconda iterazione, e quindi un conseguente accorciamento della terza iterazione. Pertanto in quest'ultima fase si sono eseguiti i tests per la pubblicazione dell'applicazione nel Market ma non è stato possibile concludere alcuni requisiti opzionali, che prevedevano la visualizzazione delle stazioni di monitoraggio tramite indice ordinato.
I requisiti non soddisfatti sono:

- **Fz-1** L'applicazione deve permettere all'utente la visualizzazione di tutti i bacini idrici monitorati mediante indice;
- **Fz-2** L'applicazione deve permettere all'utente la visualizzazione mediante indice di tutte le stazioni di monitoraggio di un particolare bacino idrico.

4.4 Progettazione

Di seguito viene presentato il funzionamento dell'applicazione, in relazione alle sue dipendenze esterne e successivamente viene riportata l'architettura.

4.4.1 Funzionamento dell'app

L'applicazione prevede l'utilizzo della libreria *com.google.android.maps* (*Google Maps* per Android) per la visualizzazione della mappa; mentre utilizza dei file in formato XML, raggiungibili attraverso indirizzi pubblici messi a disposizione da server interni dell'ARPAV, per il reperimento dei dati delle stazioni di monitoraggio. Questo funzionamento è illustrato in Figura ***4.11***.

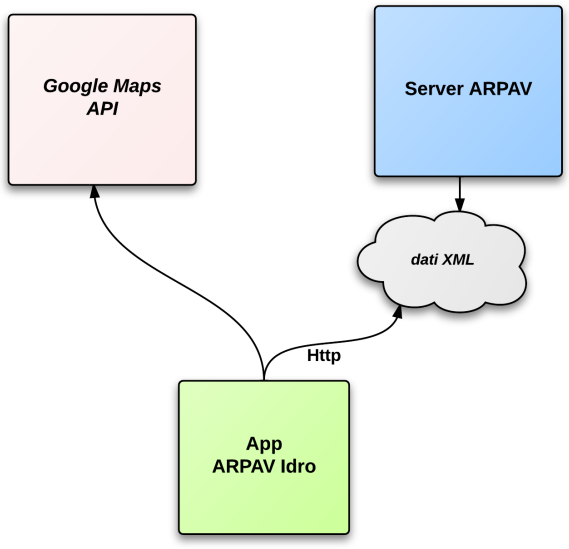


Figura 4.11: Schema funzionamento

I file XML sono così suddivisi: vi è un file principale che contiene solamente l'indice di tutte le stazioni con le loro informazioni di base: l'ID, il nome, il bacino di appartenenza, le coordinate geografiche, il tipo di stazione e l'indirizzo ove reperire i valori registrati dai sensori in esse contenute. Questo file viene scaricato ed elaborato all'avvio dell'app e permette la collocazione di tutte le stazioni sulla mappa. E' questo l'unico file il cui indirizzo dovrà sempre rimanere invariato per

permettere il corretto funzionamento dell'app. Un suo eventuale cambiamento comporta anche un aggiornamento dell'applicazione.
Di seguito viene presentato un esempio del tag *<STAZIONE>* contenuto nel file:

```
<STAZIONE>
    <ID>270</ID>
    <NOME>Adige a Albaredo</NOME>
    <BACINO>ADIGE</BACINO>
    <X>11.26639</X>
    <Y>45.31250</Y>
    <QUOTA>24</QUOTA>
    <LINK>http://www.arpa.veneto.it/upload_teolo/dati_xml/STAZ_270.xml</LINK>
    <TIPOSTAZ>IDRO</TIPOSTAZ>
</STAZIONE>
```

Il tag *<LINK>* contiene l'indirizzo al file XML specifico per ogni stazione, nel quale sono contenuti i dati monitorati dai sensori della stessa. Questo file secondario viene caricato ed elaborato, solamente se l'utente fa richiesta di visualizzare i dati della rispettiva stazione.
Di seguito in Figura ***4.12*** è presentato uno schema illustrativo della suddivisione dei file XML.

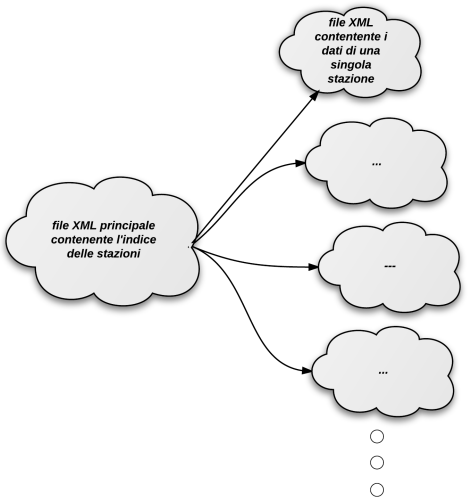


Figura 4.12: Schema file XML

Questa ripartizione dei file è stata da me decisa, in sostituzione ad un precedente file unitario nel quale erano contenute tutte le informazioni delle stazioni. Questo file risultava scomodo sia da scaricare, in quanto troppo pesante (sull'ordine di 1Mb), sia da elaborare poiché avrebbe rallentato di molto l'esecuzione dell'app. E' stato pertanto richiesto ai tecnici che gestiscono la generazione automatica dei file XML, la creazione di una nuova procedura che generasse i nuovi file. In questo modo è stato possibile garantire una fluidità dell'esecuzione dell'app, in quanto vengono scaricate di volta in volta solo le informazioni richieste dall'utente. Inoltre ogni file risulta essere molto leggero, tanto che i tempi di attesa durante il loro caricamento è quasi impercettibile all'utente: il file con l'indice delle stazioni pesa circa 29Kb, mentre quelli relativi ad ogni stazione sono intorno agli 8Kb.
Da ultimo è stato necessario richiedere la modifica del tipo di coordinate geografiche riportate nei tag *<X>* e *<Y>*, sostituendo le coordinate di tipo Gauss Boaga, con le attuali coordinate di tipo WGS 84. Google Maps infatti utilizza quest'ultimo tipo di coordinate per la localizzazione geografica.

4.4.2 Diagramma dei componenti

Grazie al framework per Android e alle componenti Activity da esso fornite, la progettazione delle applicazioni risulta essere relativamente semplice. Le Activity sono il componente fondamentale delle applicazioni Android, e in relazione al design pattern architetturale Model-View-Controller[g]

(MVC), rappresentano l'insieme di view e controller.
Per la realizzazione di *ARPAV Idro* sono state implementate tre classi che estendono la componente Activity: SplashScreenActivity, MapActivity e GraphActivity.
In Figura ***4.13*** viene presentato il diagramma UML dei componenti.

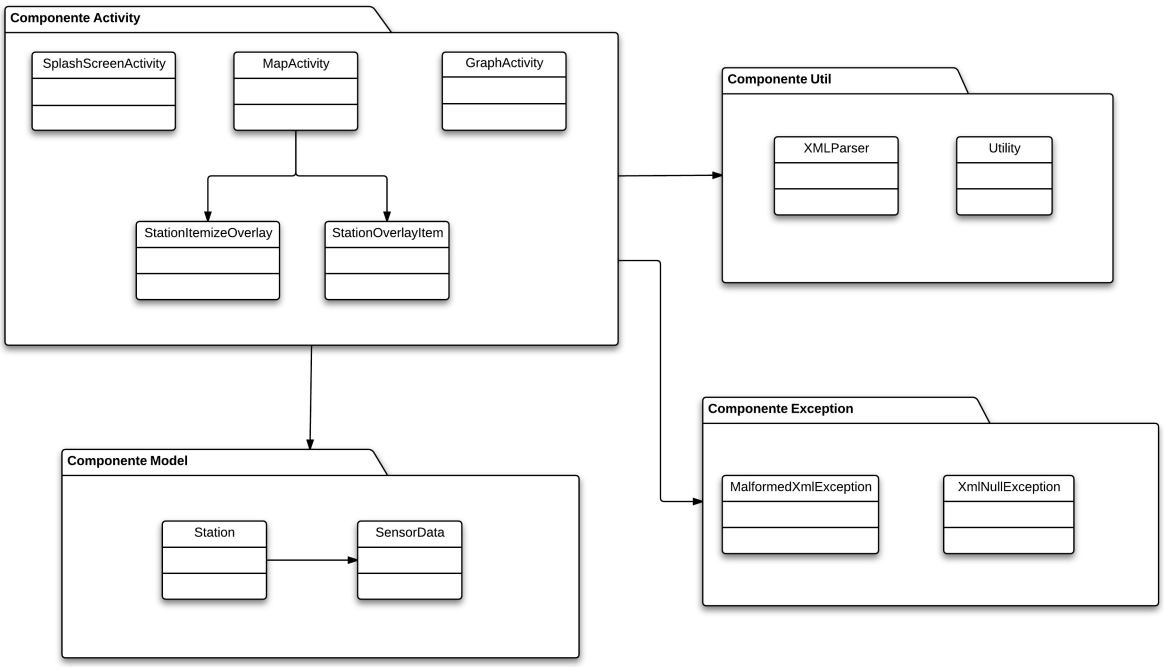


Figura 4.13: Diagramma dei componenti

4.4.3 Componente Activity

E' il componente che contiene le classi logiche fondamentali dell'applicazione, in particolare le activity. Di seguito viene presentata una descrizione di ogni singola classe.

it.arpav.mobile.apparvap.activity.SplashScreenActivity

Questa è l'activity iniziale che viene lanciata ad ogni avvio dell'applicazione e ha lo scopo di visualizzare una schermata introduttiva, contenente il logo ARPAV per un periodo di alcuni millisecondi. L'activity all'interno del suo metodo principale `onCreate()` avvia un thread che esegue una `wait(time)` per un determinato periodo di tempo `time`; questo permettere la visualizzazione temporizzata.

it.arpav.mobile.apparvap.activity.MapActivity

E' l'activity principale e viene avviata in seguito alla terminazione della *SplashScreenActivity*. MapActivity permette la visualizzazione della mappa della Regione Veneto, in cui sono allocate le stazioni di monitoraggio, ed estende la classe `MapActivity` messa a disposizione dalla libreria di Google Maps.
L'activity all'avvio controlla la presenza della rete e successivamente scarica ed elabora le informazioni sulle stazioni, attraverso il file XML contenente l'indice delle stazioni, appositamente generato dai server ARPAV. Quindi la mappa viene popolata posizionando su di essa le stazioni marcate con apposite icone.
L'activity verifica anche l'attivazione della localizzazione del dispositivo tramite la rete mobile o tramite Gps. Per fare questo utilizza un componente messo a disposizione dalle API Android, `android.location.LocationManager`, come illustrato nel codice seguente:

```
/**
 * check if Localization is active
 */
public boolean isLocalizationActive(){
    if(locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)
        || locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER))

        return true;
    return false;
}
```

Se la localizzazione è attiva, viene visualizzata nella mappa la posizione in cui ci si trova. E' anche possibile effettuare un ingrandimento della posizione corrente, attraverso la funzione `animateToMyPosition(GeoPoint)`; quest'ultima verifica se ci si trova all'interno dei confini della Regione Veneto attraverso la funzione `isUserOutLimits(GeoPoint)` e in caso negativo allerta l'utente con un messaggio.

```
/**
 * move the map to user position
 */
private void animateToMyPosition(GeoPoint myPosition){
    // check if user is out of Veneto limits
    if(isUserOutLimits(myPosition))
        showOutLimitsAlertDialog(myPosition);
    else{
        mapController.animateTo(myPosition);
        mapController.setZoom(11);
    }
}


/**
 * check if user is out of Veneto limits
 */
private boolean isUserOutLimits(GeoPoint myPosition){
    int lat = myPosition.getLatitudeE6();

    if( lat > latTopVeneto || lat < latBottomVeneto )
       return true;

    int lon = myPosition.getLongitudeE6();
    if( lon > lonRightVeneto || lon < lonLeftVeneto )
       return true;

    return false;
}
```

it.arpav.mobile.apparvap.activity.GraphActivity

Activity volta alla creazione dei grafici dei dati delle stazioni. La classe sfrutta la libreria AChartEngine che mette a disposizione svariate tipologie di grafici, personalizzabili attraverso apposite funzioni. Tra i principali campi dati dell'activity troviamo:

- `private String type`: specifica il tipo di grafico, pluviometrico o idrometrico;
- `private String stationName`: indica il nome della stazione a cui i valori fanno riferimento;
- `private String unitMeasurement`: specifica l'unità di misura dei valori;
- `private double[] value`: array di double che contiene i valori registrati.

it.arpav.mobile.apparvap.activity.StationOverlayItem

Per la creazione e collocazione di item personalizzati su di una Google Maps per Android, è necessario l'utilizzo di due appositi componenti forniti dalla libreria Google Maps Android API:

- `com.google.android.maps.ItemizedOverlay`
  classe necessaria per la creazione e gestione di item, che nel nostro caso rappresentano le stazioni di monitoraggio da posizionare sulla mappa. La classe è di tipo abstract e deve essere necessariamente estesa da un'altra classe, che la completa definendone tutti i suoi metodi astratti.

- `com.google.android.maps.OverlayItem`
  classe che consiste nei veri e propri item da collocare sulla mappa; quest'ultimi vengono manipolati dalla classe precedente ItemizedOverlay.

`StationOverlayItem` personalizza la classe `OverlayItem` estendendola. E' stato necessario fare ciò per poter "associare" ad ogni `OverlayItem` una determinata stazione di monitoraggio, mantenendone traccia con un riferimento.
La classe infatti contiene un campo dati `protected Station station` che fa riferimento ad un oggetto di tipo `Station`, contenente le informazioni della stazione che rappresenta. La classe dispone di un costruttore che riceve due parametri: il primo di tipo `GeoPoint`, classe fornita dalla libreria, contiene le coordinate di tipo WGS 84 in cui si trova la stazione, il secondo è di tipo `Station`. Di seguito vengono presentati il codice della classe contenente il costruttore e due funzioni per la gestione dell'oggetto `Station`:

```
public StationOverlayItem(GeoPoint point, Station station) {
    super(point, station.getName(), "Bacino:" + station.getReservoir());
    this.station = station;
}

public Station getStation() {
    return station;
}


public void setStation(Station station) {
    this.station = station;
}
```

it.arpav.mobile.apparvap.activity.StationItemizeOverlay

Questa classe ha lo scopo di creare e gestire le stazioni sulla mappa ed estende il componente `com.readystatesoftware.mapviewballoons.BalloonItemizedOverlay` fornito dalla libreria MapViewBaloon (si veda Capitolo *2.3*). A sua volta, `BalloonItemizedOverlay` estende la classe astratta `ItemizedOverlay` fornita dalla libreria Google Maps Android API. E' stato necessario l'utilizzo della libreria MapViewBaloon per poter visualizzare sulla mappa dei "fumetti", indicanti le informazioni di una rispettiva stazione quando questa viene selezionata.
La classe contiene il campo dati `private ArrayList<StationOverlayItem> mOverlays`, che serve per memorizzare tutti i riferimenti agli `StationOverlayItem` da disegnare sulla mappa.
All'interno della classe troviamo la funzione `public void addOverlay(StationOverlayItem)` che permette l'aggiunta di nuovi item all'`ArrayList<StationOverlayItem> mOverlays`. Inoltre la classe sovrascrive il metodo `protected boolean onBalloonTap(int, StationOverlayItem)` della libreria MapViewBaloon, che viene richiamato quando l'utente preme nel fumetto di una stazione. Il metodo quindi invoca il download e il parser del file XML contenente i dati della stazione, e successivamente avvia l'activity per la visualizzazione del grafico.

```
@Override
protected boolean onBalloonTap(int index, StationOverlayItem item) {

    pdToLoadStationData = ProgressDialog.show(context, context.getString(
        R.string.loading), context.getString(R.string.loadingData),true,true);
        Station station=item.getStation();
        if(! station.isDataLoaded() ){
            if(Util.isOnline(context)){
                DownloadStationDataTask dt = new DownloadStationDataTask();
                dt.execute(station);

            }
            else{
                pdToLoadStationData.dismiss();
                Toast.makeText(context, context
                    .getString(R.string.notOnlineNote), Toast.LENGTH_SHORT).show();
            }
        }
        else{
            pdToLoadStationData.dismiss();
            choiceSensorData(station);
        }

        return true;
}
```

Nel caso in cui una stazione registri sia il livello idrometrico che pluviometrico, verrà chiesto all'utente quali dati desidera visualizzare graficamente.

4.4.4 Componente Model

Il componente Model serve a memorizzare i dati necessari all'applicazione, ossia tutte le informazioni relative alle stazioni di monitoraggio e i valori registrati dai sensori in esse contenuti.

it.arpav.mobile.apparvap.model.Station

Classe rappresentante una stazione di monitoraggio. Essa contiene vari campi dati di tipo `String` ricavati dal file XML principale, che memorizzano le informazioni di una stazione. Sono presenti inoltre tutti i metodi pubblici necessari per gestire tali campi dati, come illustrato in Figura ***4.14***.

it.arpav.mobile.apparvap.model.SensorData

La classe rappresenta i sensori di rilevamento presenti in una stazione e memorizza al suo interno i valori registrati. Come già visto infatti, tra i campi dati delle classe `Station` vi sono dei riferimenti ad oggetti di tipo `SensorData`. Gli oggetti di questa classe vengono creati soltanto quando l'utente richiede la visualizzazione dei grafici relativi ad una stazione di monitoraggio e vengo pertanto caricati i dati relativi. In Figura ***4.15*** viene illustrata la classe.

4.4.5 Componente Util

Le classi di questa componente contengono funzionalità di supporto.

it.arpav.mobile.apparvap.util.XmlParser

Questa classe ha lo scopo di elaborare i file XML quando richiesto. E' stato deciso di utilizzare l'interfaccia basata sullo standard DOM, che permette di vedere il documento XML come se fosse un albero e di navigare tra i suoi nodi. E' stata fatta questa scelta principalmente in relazione alla frammentazione delle informazioni all'interno di molti file XML, ciò comporta che:

- ogni file contiene unicamente i dati della stazione a cui si riferisce;

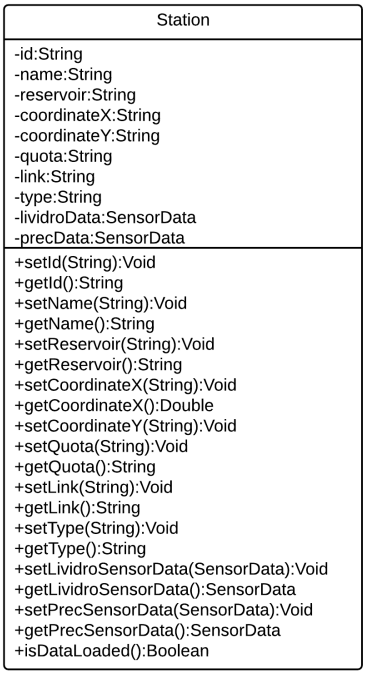


Figura 4.14: Classe Station

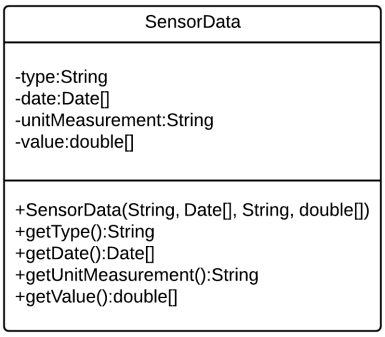


Figura 4.15: Classe SensorData

- ogni file è molto leggero e poco complesso.

Lo standard DOM prevede il caricamento completo in memoria di ogni singolo file XML in fase di elaborazione; poichè però questi ultimi sono stati frammentati, comportano l'utilizzo di pochissima memoria. Inoltre quando viene richiesto un file XML, questo contiene solamente le informazioni minime necessarie; quindi l'eventuale utilizzo dell'interfaccia SAX, che permette di ricercare velocemente un'informazione specifica all'interno di un documento XML, sarebbe stata inutile.
Tra i metodi principali della classe troviamo `public String getXmlFromUrl(String url)`; questo invia richieste HTTP GET ai server ARPAV, per richiedere i file XML contenenti le informazioni necessarie, come illustrato nel codice seguente:

```
/**
 * Reads an XML file from an url with Http request, and return a string of XML file
*/
public String getXmlFromUrl(String url) {
    String xml = null;

    try {
        // defaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(url);
        HttpResponse httpResponse = httpClient.execute(httpGet);
        HttpEntity httpEntity = httpResponse.getEntity();
        xml = EntityUtils.toString(httpEntity);
    } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
    } catch (ClientProtocolException e) {
            e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return xml;
}
```

Una volta ricevuto e salvato il file XML all'interno di una stringa, questo viene trasformato in elementi DOM, come illustrato nel codice seguente:

```
/**
 * Parsing XML content from string and getting DOM element
*/
public Document getDomElementFromString( String xml) {
    Document doc = null;
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    try {

        DocumentBuilder db = dbf.newDocumentBuilder();

        InputSource is = new InputSource();
        is.setCharacterStream(new StringReader(xml));
        doc = db.parse(is);

    } catch (ParserConfigurationException e) {
            System.out.println("XML parse error: " + e.getMessage());
            return null;
    } catch (IOException e) {
            System.out.println("I/O exeption: " + e.getMessage());
            return null;
    }

    return doc;
}
```

it.arpav.mobile.apparvap.util.Utility

Questa classe ha l'obbiettivo di ragruppare al suo interno svariati metodi statici di supporto, che possono essere utilizzati a livello globale da più di una stessa classe; tra questi troviamo per esempio il metodo che verifica se il dispositivo è connesso alla rete. Nella classe sono presenti anche molti campi dati statici che permettono di tenere traccia di tutte le stringhe identificate come parole chiavi fondamentali, utilizzate a livello globale. E' stato deciso di adottare una classe per tale scopo, così da poter ragruppare in un unico "contenitore" tutte queste informazioni e funzionalità.

4.4.6 Componente Exception

Questa componente contiene le classi necessarie alla gestione delle eccezioni dell'app.

it.arpav.mobile.apparvap.exception.MalformedXmlException

Eccezione sollevata nel caso in cui nel file XML non siano presenti i tags ricercati durante il parser.

it.arpav.mobile.apparvap.exception.XmlNullExceprion

Eccezione sollevata nel caso in cui il file XML ricercato non sia disponibile, per esempio per problemi interni ai server ARPAV.

4.5 Pubblicazione nell'Android Market

La pubblicazione nell'Android Market è avvenuta a conclusione dell'applicazione. Successivamente è stato eseguito un upgrade, poiché è stato necessario effettuare delle migliorie ai grafici per renderli più facilmente leggibili. Si è pertanto giunti alla corrente versione software v1.1.

4.6 Resoconto requisiti

Di seguito vengono elencati i requisiti prefissati, indicando per ognuno di essi se è stato soddisfatto o meno.

Tabella 4.8: *ARPAV Idro*: Resoconto requisiti

Codice	Descrizione	Esito
Fo-1	L'applicazione deve permettere all'utente di visualizzare la mappa della Regione Veneto	Soddisfatto
Fo-1.1	L'applicazione deve permettere all'utente di navigare sulla mappa della Regione Veneto	Soddisfatto
Fo-1.1.1	L'applicazione deve permettere all'utente di eseguire azioni di zoom e scroll sulla mappa	Soddisfatto
Fo-2	L'applicazione deve permettere all'utente di visualizzare le stazioni di monitoraggio idrometriche e pluviometriche sulla mappa della Regione Veneto	Soddisfatto
Fo-2.1	L'applicazione deve permettere all'utente di visualizzare sulla mappa le stazioni di monitoraggio mediante icone differenti in base al tipo di stazione	Soddisfatto
Fo-3	L'applicazione deve permettere all'utente di poter scegliere una stazione di monitoraggio a partire da una mappa della Regione Veneto in cui sono state localizzate le stazioni di rilevamento	Soddisfatto
Fo-3.1	L'applicazione deve permettere all'utente di poter cliccare l'icona di una stazione di monitoraggio presente sulla mappa	Soddisfatto

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

Fo-3.2	L'applicazione deve permettere all'utente di visualizzare le informazioni relative ad una stazione: paese in cui si trova e bacino di appartenenza	Soddisfatto
Fo-4	L'applicazione deve permettere all'utente di visualizzare i dati monitorati dalle stazioni	Soddisfatto
Fo-4.1	L'applicazione deve permettere all'utente di visualizzare i dati relativi agli ultimi tre giorni	Soddisfatto
Fo-4.2	L'applicazione deve permettere all'utente di visualizzare i dati sotto forma di grafico	Soddisfatto
Fo-4.2.1	L'applicazione deve permettere all'utente di visualizzare i dati idrometrici sotto forma di grafico	Soddisfatto
Fo-4.2.2	L'applicazione deve permettere all'utente di visualizzare i dati pluviometrici sotto forma di grafico	Soddisfatto
Fo-4.2.3	L'applicazione deve permettere all'utente il ridimensionamento dei grafici	Soddisfatto
Fo-5	L'applicazione deve avvisare l'utente in caso di mancanza di connessione del dispositivo alla rete internet	Soddisfatto
Fd-1	L'applicazione deve permettere all'utente la localizzazione sulla mappa della propria posizione	Soddisfatto
Fd-1.1	L'applicazione deve permettere all'utente la visualizzazione della propria posizione sulla mappa mediante apposita icona	Soddisfatto
Fd-1.2	L'applicazione deve permettere all'utente la possibilità di visualizzare velocemente la propria posizione zoomata e centrata nella mappa	Soddisfatto
Fd-1.3	L'applicazione deve permettere all'utente la possibilità di attivare o disattivare la funzione di rilevamento della propria posizione nel dispositivo mobile	Soddisfatto
Fz-1	L'applicazione deve permettere all'utente la visualizzazione di tutti i bacini idrici monitorati mediante indice	Non soddisfatto
Fz-2	L'applicazione deve permettere all'utente la visualizzazione di tutte le stazioni di monitoraggio di un particolare bacino idrico mediante indice	Non soddisfatto

4.7 Possibili estensioni

Di seguito vengono riportate le possibili estensioni applicabili all'app, che sono state ricavate sulla base di idee dell'agenzia e mie personali, in fase di stesura dei requisiti dell'applicazione.

4.7.1 Elenco ordinato delle stazioni

Questa estensione rientrava nei requisiti opzionali Fz-1 e Fz-2. Quest'ultimi prevedevano la visualizzazione delle stazioni attraverso un elenco ordinato e non sono stati soddisfatti. Di seguito viene descritto come tale estensione era stata da me pensata.
L'elenco risulterebbe accessibile tramite apposita sezione dedicata, nella quale le stazioni verrebbero raggruppate per bacino di appartenenza, come illustrato in Figura ***4.16***. In questo modo verrebbe agevolata la navigazione dell'utente all'interno dell'app, poiché risulterebbe più semplice sia ricercare una stazione di monitoraggio per bacino di appartenenza, sia visualizzare i livelli di uno stesso bacino in diverse zone di monitoraggio.
Ogni stazione inoltre verrebbe identificata con le differenti tipologie di icone in base ai sensori in essa presenti.

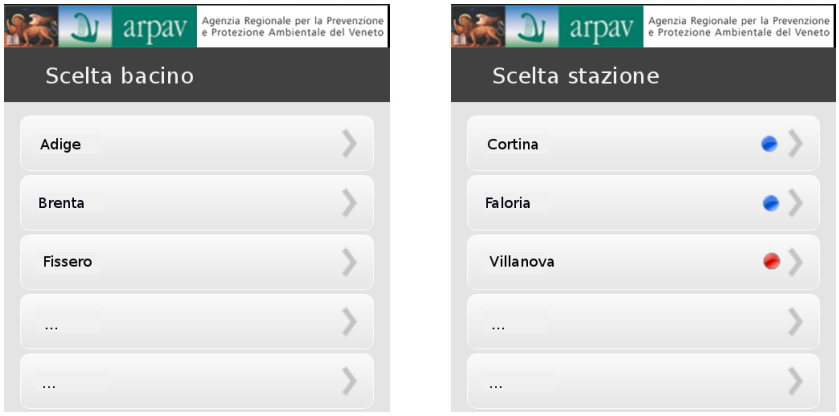


Figura 4.16: Estensione - Indice stazioni

4.7.2 Funzione di ricerca avanzata delle stazioni

Per una maggiore immediatezza nella ricerca di una specifica stazione desiderata, potrebbe risultare molto utile per l'utente avere a disposizione un' apposita funzione di ricerca veloce. Questa funzione potrebbe essere richiamabile sia attraverso un bottone dedicato visualizzabile sullo schermo, sia attraverso l'apposito pulsante hardware di ricerca, per i dispositivi che ne sono forniti. Per poter avviare la funzione di ricerca attraverso il bottone hardware, sarebbe necessario eseguire un override dell'apposito metodo:

```
public boolean onSearchRequested() {

    // Implementation of search

    return true;
}
```

Una volta attivata la funzione di ricerca, verrebbe visualizzata sullo schermo una barra superiore, contente un campo di testo in cui inserire la parola chiave e un pulsante di avvio ricerca, come visualizzato in Figura ***4.17***.

Figura 4.17: Estensione - Ricerca veloce

La ricerca della parola chiave verrebbe effettuata tra i campi dati `String name` e `String reservoir` degli oggetti `Station`. In questo modo sarebbe possibile ricercare non soltanto una specifica stazione, ma inserendo come parola chiave il nome di un bacino monitorato, verrebbero visualizzate tutte le stazioni di appartenenza di codesto bacino.
Al termine verrebbe riportato un elenco di tutte le stazione trovate.

4.7.3 Stazioni preferite

Nell'applicazione potrebbe essere aggiunta una sezione dedicata alle stazioni preferite. L'utente avrebbe quindi la possibilità di marcare le stazioni di maggior interesse come preferite, così da poterle visualizzare in stile elenco nell'apposita sezione *Preferiti*. In questo modo risulterebbe più facile e veloce accedere alle informazioni desiderate.
L'aggiunta di una stazione ai preferiti potrebbe avvenire in tre differenti modi:

1. **Attraverso la mappa**: una volta localizzata una stazione di interesse, sarebbe possibile aggiungerla ai preferiti attraverso un apposito pulsante stellato, presente all'interno del "fumetto" riportante le informazioni della stazione. Qualora una stazione fosse inserita tra i preferiti questo pulsante risulterebbe di colore giallo, altrimenti sarebbe grigio, come visualizzato in Figura ***4.18***.

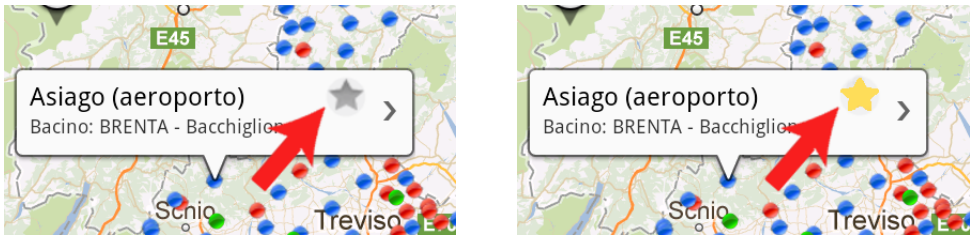


Figura 4.18: Estensione - Aggiunta preferiti tramite mappa

2. **Attraverso l'elenco ordinato delle stazioni**: dopo aver trovato la stazione di interesse, anche in questo caso sarà visibile un pulsante stellato che permetterà l'aggiunta o la rimozione della stazione dai Preferiti, come illustrato in Figura ***4.19***.

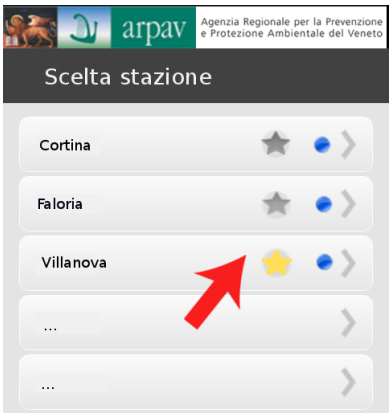


Figura 4.19: Estensione - Aggiunta preferiti tramite elenco

3. **Attraverso un pulsante scorciatoia nella sezione preferiti**: in questa sezione verrebbe visualizzato un pulsante per l'aggiunta di nuove stazioni preferite, il quale rimanderà all'elenco ordinato delle stazioni. Da qui si potrà aggiungere una stazione come illustrato nel punto precedente.

5 Conclusioni

5.1 Contributo

Durante lo stage ho portato a compimento quasi tutti gli obbiettivi prefissati, realizzando delle app che fanno parte di un ampio progetto per la diffusione dei dati sui monitoraggi ambientali, a mio avviso socialmente utile e molto ambizioso.
Per la buona riuscita dello stage, sono state fondamentali le nozioni di programmazione acquisite durante il corso di laurea, ed in particolar modo il linguaggio ad oggetti Java, che mi hanno permesso di iniziare da subito lo sviluppo del progetto in ambito Android.
A livello personale ho potuto approfondire le mie conoscenze sul formato XML, ampiamente utilizzato durante lo stage. Infatti i dati ARPAV relativi ai monitoraggi ambientali sono pubblicati in spazi web pubblici in formato XML, ed è attraverso questi indirizzi che le app sviluppate estraggono le informazioni necessarie. Inoltre è stato fatto ampio uso del linguaggio XML per la realizzazione dei layout delle app, per la stesura dei rispettivi file di configurazione e per la scrittura dei file indicanti le "risorse" delle app.
Ho potuto approfondire le mie conoscenze anche nell'utilizzo del sistema di versionamento Git e nella rispettiva piattaforma web GitHub, che permettono di tenere traccia permanente delle modifiche effettuate e di realizzare progetti in modo collaborativo tra più utenti.
Grazie a queste nozioni ho potuto soddisfare le esigenze richieste dall'agenzia, consolidando il suo impegno nella diffusione agli utenti dei dati ambientali in tempo reale, attraverso l'utilizzo della tecnologia Android.
Lo stage è stato supervisionato dal tutor Dott. Luca Menini ed ha avuto una durata totale di 320 ore. Oltre al tutor, all'interno dell'agenzia ho collaborato anche con un tecnico-programmatore, Matteo Setini, con il quale ho potuto interagire per la preparazione dei database e per alcune decisioni tecniche, che si sono presentate durante lo sviluppo.

5.2 Considerazioni personali

Lo stage svolto mi ha permesso di formarmi in un ambito a me nuovo: lo sviluppo di app per Android. Ho potuto infatti acquisire importanti competenze sulla programmazione Java applicata a dispositivi Android, attraverso l'utilizzo del framework Android SDK.
Infatti era mio desiderio iniziale poter svolgere un'esperienza di stage, che riguardasse la realizzazione di app per smartphones ed in particolare per i dispositivi Android. Questo desiderio era motivato sia da un mio interesse personale per questo sistema operativo basato sul Kernel[g] Linux ed in continua evoluzione, sia per l'ampia diffusione che quest'ultimo ha avuto e sta avendo nel mercato della telefonia mobile.
Pertanto la proposta di stage offertami dall'agenzia è stata ritenuta da me molto innovativa per le tecnologie adottate e molto valida in relazione ai prodotti che avrei dovuto realizzare. Infatti le applicazioni *ARPAV Idro* e *ARPAV Meteo* sono un importante strumento di divulgazione dei dati

ambientali, grazie alle quali gli utenti possono interagire beneficiando delle informazioni da esse trasmesse. La scelta dello stage quindi non è stata casuale, ma dettata dalle motivazioni sopra citate.
Durante lo stage il rapporto professionale con l'agenzia ed in particolare con il mio tutor è stato molto efficace e costruttivo così da permettere durante tutta la mia permanenza una chiara collaborazione.

6 Glossario

A

App
Abbreviazione di applicazione; il termine viene spesso riferito alle applicazioni per smartphone e tablet.

Android Console
Spazio web dedicato agli sviluppatori di app Android, attraverso il quale è possibile gestire la pubblicazione delle proprie app.

Android Market (Google Play)
Store unificato di applicazioni, sviluppato da Google per i dispositivi Android e per i sistemi desktop e notebook.

Android SDK
L'acronimo SDK che in inglese sta per "*software development kit*", consiste in un apposito kit di sviluppo, in questo caso per Android, che permette agli sviluppatori di creare applicazioni per la piattaforma Android.

G

GUI "Graphics User Interface"
In italiano Interfaccia Grafica Utente, è un tipo di interfaccia utente caratterizzata dall'utilizzo di oggetti grafici (e.g finestre, pulsanti, caselle di testo).

K

Kernel
Costituisce il nucleo di un sistema operativo. Si tratta di un software avente il compito di fornire ai processi in esecuzione sull'elaboratore un accesso sicuro e controllato all'hardware.

M

Model-View-Controller (MVC)
E' un design pattern architetturale molto diffuso nei linguaggi orientati agli oggetti. Il pattern è basato sulla separazione fra i componenti software in tre livelli: il Model che rappresenta i dati dell'applicazione, la View che rappresenta la visualizzazione degli oggetti e il Controller che rappresenta l'insieme di regole che permettono la trasformazione degli input sulle viste in modifiche del modello.

P

Pinch to zoom
Tipo di "gesture" per dispositivi multi-touch che consiste nel movimento di apertura e chiusura di due dita per effettuare zoom-in e zoom-out su un'immagine o una schermata ridimensionabile.

S

Spash Screen
Consiste nella schermata di apertura di una applicazione. Lo splash screen è solitamente costituito da una immagine introduttiva e altre informazioni sull'applicazione.

J

Java
Consiste in un linguaggio di programmazione orientato agli oggetti che ha come caratteristica principale la portabilità tra i sistemi operativi.

X

XML
E' un linguaggio di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.

U

UML
Linguaggio per la modellazione del Software. E' lo standard utilizzato nell'ingegneria del Software per descrivere un sistema informatico.

7 Bibliografia

Riferimenti Bibliografici

[a] Marko Gargenta, 2011. Learning Android. 1° ed. O'Reilly Media. Sebastopol, CA.

Riferimenti Web

[1] http://www.arpa.veneto.it/arpav/chi-e-arpav

[2] http://it.wikipedia.org/wiki/Android

[3]http://www.androiday.com/diffusione-smartphone-negli-usa-android-cresce-iphone-cala

[4] http://android-developers.blogspot.it/2012/01/say-goodbye-to-menu-button.html

[5] http://http://www.achartengine.org

[6] https://github.com/lorensiuswlt/NewQuickAction#newquickaction

[7] https://github.com/lorensiuswlt/NewQuickAction3D

[8] https://github.com/jgilfelt/android-mapviewballoons

[9] http://jgilfelt.github.com/android-mapviewballoons/

[10] http://www.androidiani.com/applicazioni/sviluppo/capire-e-programmare-le-activity-869