

ALMA MATER STUDIORUM · UNIVERSITÀ DI
BOLOGNA
CAMPUS DI CESENA

SCUOLA DI SCIENZE
Corso di Laurea in Ingegneria e Scienze Informatiche

**SINCRONIZZAZIONE DI UNA
APPLICAZIONE OFFLINE
CON I SISTEMI DI BACK-END
E ALLINEAMENTO DELLE
INFORMAZIONI ACQUISITE IN
CAMPO**

Relazione finale in
PROGRAMMAZIONE DI SISTEMI MOBILE

Relatore
Dott. MIRKO RAVAIOLI

Presentata da
LUCA PASCUCCI

Terza Sessione di Laurea
Anno Accademico 2014 - 2015

PAROLE CHIAVE

EBWorld

Java

Offline

XML

GIS

Alla mia famiglia

Indice

Introduzione	xi
1 EBWorld e gli applicativi esistenti	1
1.1 EBWorld s.r.l.	1
1.2 Geographic Information System	2
1.2.1 Modello dei dati	3
1.3 Gli applicativi esistenti	4
1.3.1 MapYou	4
1.3.2 Gestore MapYou	6
2 Tecnologie Utilizzate	9
2.1 Libreria JMTP	9
2.2 Android Debug Bridge	10
2.2.1 Sintassi	11
2.2.2 Principali funzionalità	11
2.3 Libreria JTattoo	12
2.4 Mercurial	13
3 Analisi	15
3.1 Descrizione generale	15
3.1.1 Prospettiva del prodotto desktop	15
3.1.2 Funzioni del prodotto desktop	15
3.1.3 Prospettiva del prodotto mobile	16
3.1.4 Funzioni del prodotto mobile	16

3.1.5	Caratteristiche dell'utente finale	16
3.1.6	Ambiente d'utilizzo dei progetti	16
3.1.7	Linguaggio e ambienti di sviluppo	17
3.2	Applicativi online o offline	17
3.2.1	Vantaggi e svantaggi online	17
3.2.2	Vantaggi e svantaggi offline	18
3.2.3	Scelta per applicativi sviluppati	18
3.3	Funzionalità del sistema desktop	19
3.3.1	Visualizzazione e selezione dispositivi	19
3.3.2	Preparazione dispositivo	20
3.3.3	Gestione file di configurazione	20
3.3.4	Visualizzazione e selezione mappe del computer desk- top e del dispositivo mobile	21
3.3.5	Inserimento e rimozione mappe	21
3.3.6	Gestione file di definizione progetti	22
3.3.7	Visualizzazione e selezione progetti	22
3.3.8	Esportazione o rimozione progetto	23
3.4	Moduli del sistema desktop	23
3.4.1	Modulo Dispositivi	24
3.4.2	Modulo Mappe	24
3.4.3	Modulo Progetti	26
3.5	Funzionalità del sistema mobile	27
3.5.1	Gestione file dei progetti	27
3.5.2	Visualizzazione e selezione progetti	27
3.5.3	Caricamento e salvataggio progetto	28
3.6	Moduli del sistema mobile	28
3.6.1	Modulo Progetti	28
4	Progettazione	31
4.1	Applicativo desktop	31
4.1.1	Architettura principale	31
4.1.2	Flusso di utilizzo dell'applicativo	36

4.1.3	Comunicazione tra applicativo desktop e dispositivo mobile	37
4.1.4	Manager	39
4.1.5	Struttura dei file XML	40
4.1.6	Workspace	46
4.1.7	Progettazione GUI	48
4.2	Applicativo mobile	51
4.2.1	Definizione modulo Progetti	51
4.2.2	Struttura dei file XML	53
4.2.3	Workspace	58
5	Implementazione	61
5.1	Rilevamento dispositivi mobile collegati	61
5.1.1	Rilevamento attraverso JMTP	61
5.1.2	Rilevazione attraverso ADB	62
5.1.3	Utilizzo combinato di JMTP e ADB	64
5.2	Gestione file XML	66
5.3	Realizzazione interfacce grafiche	69
	Conclusioni	71
	Ringraziamenti	75
	Bibliografia	79

Introduzione

Il 9 gennaio 2007 ha segnato l'inizio di un'enorme cambiamento in numerosi ambiti e settori partendo ovviamente dalle tecnologie mobili fino ad arrivare alla più semplice quotidianità, migliorando la cooperazione, lo scambio di informazioni e l'interazione tra persone. Durante lo sviluppo dei nuovi scenari creati da quel particolare evento, il mondo del lavoro è notevolmente cambiato, gli smartphone si sono diffusi per la loro praticità, affidabilità, velocità e semplicità di utilizzo. Impiegati in svariati modi, possono ad esempio essere utilizzati da un rappresentante per rimanere in contatto con la propria azienda, con clienti e fornitori oppure da un tecnico esterno per memorizzare gli interventi effettuati e poi presentarli in sede centrale.

Quest'ultimo caso rappresenta al meglio ciò che verrà spiegato nella tesi ed illustrato attraverso i progetti collegati, difatti lo scopo è di sviluppare due applicativi, uno desktop ed uno mobile che permettano ad un tecnico di preparare un dispositivo mobile contenente i dati necessari alla sua attività esterna, la possibilità di creare e salvare nuovi dati riguardanti i lavori svolti ed una volta concluse le operazioni di poterli importare in sede centrale.

Il software desktop richiesto deve fornire una interfaccia grafica semplice ed intuitiva che visualizzi i dispositivi collegati, i dati locali da importare sui dispositivi e quelli presenti sui dispositivi da esportare in locale. L'applicazione mobile deve fornire gli strumenti adatti per visualizzare ed interagire con i dati locali al dispositivo e crearne nuovi che identifichino l'operato dell'utente.

Gli aspetti visti sopra saranno descritti attraverso un'analisi del problema ed una progettazione architettuale. In particolare verranno evidenziati aspetti riguardanti la comunicazione tra l'applicativo desktop ed un dispositivo mobile collegato al computer che permetteranno la sincronizzazione dei contenuti.

La possibilità di effettuare questo percorso è stata concessa dall'azienda EBWorld s.r.l., che dal 1983 supporta lo sviluppo di grandi e piccole aziende fornendo piattaforme e applicazioni semplici ma potenti che sfruttano le informazioni geografiche dell'organizzazione per progettare e gestire reti tecnologiche sul territorio. Dato il grande sviluppo che l'azienda sta avendo, ricevendo anche la nomina di Cool Vendor 2015 in "Communications Service Provider Operational and Business Infrastructure" da Gartner, società internazionale leader nella consulenza strategica, nella ricerca e nell'analisi nel settore hi-tech; EBWorld ha espresso l'esigenza di un'applicazione che sostituisse una precedente versione realizzata su Windows Mobile ed utilizzasse i moderni dispositivi mobile disponibili sul mercato.

La sfida più importante è sicuramente quella di rendere l'applicativo desktop capace di visualizzare e gestire un dispositivo mobile collegato al PC, infatti non tutti i sistemi operativi, come ad esempio MAC OS X, riescono ad individuare un dispositivo Android e quindi garantire un collegamento stabile anche in presenza di una grossa mole di dati da interscambiare. Lo sviluppo di questo aspetto corrisponde al punto di forza dell'applicativo desktop.

Gli applicativi realizzati soddisfano tutti i requisiti iniziali richiesti dall'azienda, riuscendo a gestire senza alcun problema tutti i dispositivi Android collegati senza subire rallentamenti o perdite durante il trasferimento dati e dando la possibilità all'utente dell'applicativo mobile di visualizzare e creare i dati riguardanti gli interventi effettuati.

Il progetto desktop è stato realizzato interamente dal sottoscritto invece la parte mobile è stata sviluppata in team con il mio collega Filippo Nicolini, suddividendo attentamente i compiti tra i due. Questo elaborato di tesi tratta quindi dell'intera progettazione e implementazione dell'applicativo desktop e nella parte mobile dello sviluppo riguardante i dati creati dall'utente. Il mio collega si è occupato invece della realizzazione dell'interfaccia grafica con analisi della usability, dei workflow e del contesto operativo dell'utente.

La tesi è suddivisa in cinque capitoli:

- Nel primo capitolo verrà introdotta l'azienda EBWorld s.r.l. e fatta un'analisi sulle precedenti soluzioni sviluppate evidenziando in linea generale le loro caratteristiche.
- Il secondo capitolo espone le tecnologie utilizzate dagli applicativi e nello sviluppo software.
- Il terzo capitolo tratta l'analisi del problema, definendo le operazioni principali e mettendo in discussione la tematica dell'offline.
- La progettazione è descritta nel quarto capitolo, dove viene esposta la struttura finale dei progetti da sviluppare.
- Il quinto capitolo contiene i dettagli implementativi delle soluzioni sviluppate, focalizzandosi su alcune parti di codice molto importanti nei progetti.

Segue poi una breve conclusione.

Capitolo 1

EBWorld e gli applicativi esistenti

Questo capitolo introduce l'azienda EBWorld s.r.l., descrive sinteticamente le precedenti soluzioni sviluppate per comprendere meglio il progetto che si intende sviluppare.

1.1 EBWorld s.r.l.

EBWorld opera dal 1983 nel settore dell'informatica applicata all'impiego di informazioni geografiche e tecnologiche correlate ai processi di acquisizione, elaborazione e gestione di dati georiferiti. La consolidata competenza ed esperienza nella realizzazione di Sistemi Informativi Territoriali (SIT) e nell'integrazione di piattaforme GIS con sistemi ed applicazione, fanno di EBWorld il partner ideale per chi gestisce reti tecnologiche.

Importanti Aziende e Pubbliche Amministrazioni Italiane hanno scelto EBWorld per l'integrazione, la manutenzione e lo sviluppo dei loro sistemi IT. Dal 1995 EBWorld è Distributore e VAR per l'Italia dei prodotti Smallworld attualmente prodotti e distribuiti da GE Energy. È una dinamica azienda ICT

capace di dare forma e sostanza al più grande patrimonio di ogni Azienda: **l'informazione**.

1.2 Geographic Information System

Il GIS¹ è un sistema informativo computerizzato che permette l'acquisizione, la registrazione, l'analisi, la visualizzazione e la rappresentazione di informazioni derivanti da dati geografici (geo-referenziati) solitamente memorizzandole in strutture del tipo DBMS² che gestiscono anche la spazialità, o in singoli file.

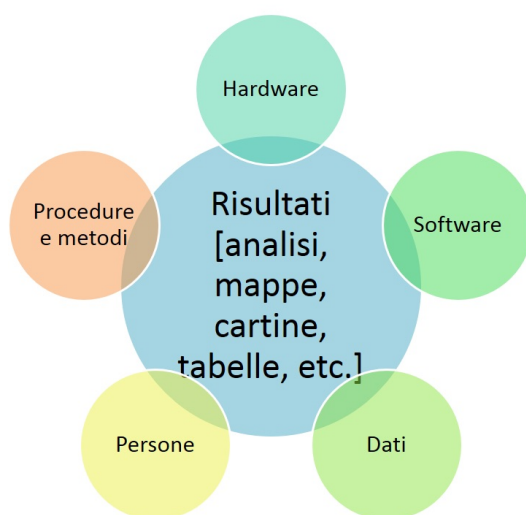


Figura 1.1: Componenti di un GIS

L'immagine raffigura le principali componenti di un sistema GIS:

- **Hardware:** costituito da computer, periferiche grafiche come plotter per la stampa di carte tematiche prodotte dalle elaborazioni GIS e server per la condivisione online di mappe. Le prestazioni hardware

¹Geographic Information System

²Database Management System

influiscono sulla capacità e velocità di elaborazione dei dati e variano a seconda della dimensione geografica del territorio che si intende memorizzare.

- **Software:** Esistono vari software GIS. La scelta di quale pacchetto utilizzare dipende dalle esigenze del progetto, dalla necessità di avere delle interfacce più o meno "user-friendly" come anche da vincoli di spesa.
- **Dati:** Elementi fondamentali del sistema, devono essere completi, possibilmente senza errori e forniti con una scala dimensionale. I dati maneggiabili con un sistema GIS possono essere di natura diversa difatti si possono introdurre dati vettoriali, dati raster, immagini da satellite ed anche documenti digitalizzati.
- **Persone:** L'utilizzatore del sistema GIS deve farsi carico di trovare i dati, deve progettare l'ambiente in cui costruire le mappe e l'analisi, deve progettare le metodologie e le procedure per analizzare i dati e deve scegliere il modo migliore per rappresentarle e comunicarle.
- **Procedure:** Utilizzo o creazione di linee guida al fine di trattare correttamente ed efficacemente i dati disponibili.
- **Analisi:** Risultato dell'aggregazione e trattamento dei dati.

1.2.1 Modello dei dati

Il modello dei dati rappresenta tutti gli oggetti che esistono nel mondo fisico (punti, linee, superficie, volumi, etc) ed è in grado di elaborare le loro combinazioni. Vengono memorizzate anche le informazioni riguardanti le relazioni tra gli oggetti (adiacenza, inclusioni, sovrapposizioni, etc.) ovvero la loro topologia. La caratteristica fondamentale di un GIS è la capacità di geo-referenziale i dati quindi ad ogni elemento vengono associate le sue coordinate spaziali reali del sistema di riferimento in cui realmente è situato l'oggetto.

1.3 Gli applicativi esistenti

Gli applicativi realizzati in precedenza dall'azienda sono stati mostrati prima di iniziare il percorso di analisi, progettazione e sviluppo necessario alla realizzazione degli obiettivi prefissati consentendo una migliore comprensione del lavoro da svolgere.

La motivazione principale che ha convinto EBWorld ad intraprendere questo percorso, è stata la modernizzazione di applicativi sempre maggiormente utilizzati per renderli compatibili con i moderni dispositivi disponibili sul mercato.

1.3.1 MapYou

MapYou, la soluzione mobile ideata da EBWorld nel 2008, è stata sviluppata per la piattaforma Windows Mobile rendendo disponibile uno strumento per la consultazione e la gestione di dati georeferenziati.

Le principali funzionalità riguardano:

- **Selezione Mappa** abilitando l'utente alla selezione della mappa da visualizzare, scegliendola tra quelle disponibili sul dispositivo.
- **Creazione Progetti** consentendo all'utente di inserire nuovi elementi sulla mappa che identifichino il suo operato o un futuro intervento da effettuare in quella determinata posizione
- **Ricerca Elementi** con la quale l'utente può cercare ed evidenziare uno o più oggetti presenti sulla mappa.
- **Utilizzo GPS o coordinate manuali** consentendo di posizionare in automatico la mappa nel punto in cui si trova o desidera l'utente.

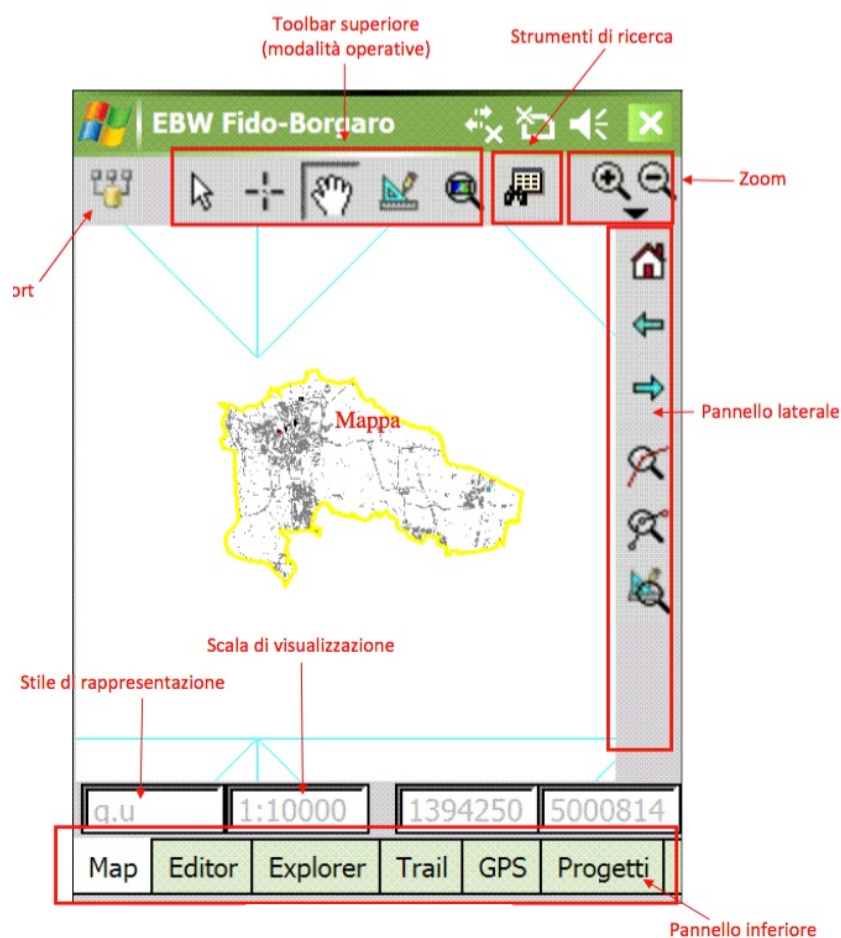


Figura 1.2: Schermata principale MapYou

La schermata principale dell'applicazione è costituita da cinque elementi principali:

1. **Mappa**, sulla quale è visualizzata l'area d'interesse;
2. **Toolbar superiore** contenete strumenti utili per la navigazione e l'interazione con la mappa;
3. **Pannello laterale**, a destra della mappa, le cui icone e collegate funzioni variano a seconda dello strumento selezionato nella toolbar superiore;

4. **Campi di informazioni** sotto la mappa, nei quali sono riportate le informazioni relative alla mappa visualizzata, ovvero la scala di visualizzazione e la scala nominale;
5. **Pannello inferiore** contenente i diversi TAB corrispondenti alle diverse sezioni dell'applicazione.

Durante la progettazione e lo sviluppo è stata tenuta in considerazione l'importanza di realizzare strumenti semplici garantendo pieno supporto alle funzionalità di navigazione e consultazione come anche di disegno e di progettazione, il tutto in presenza e in assenza di connettività internet.

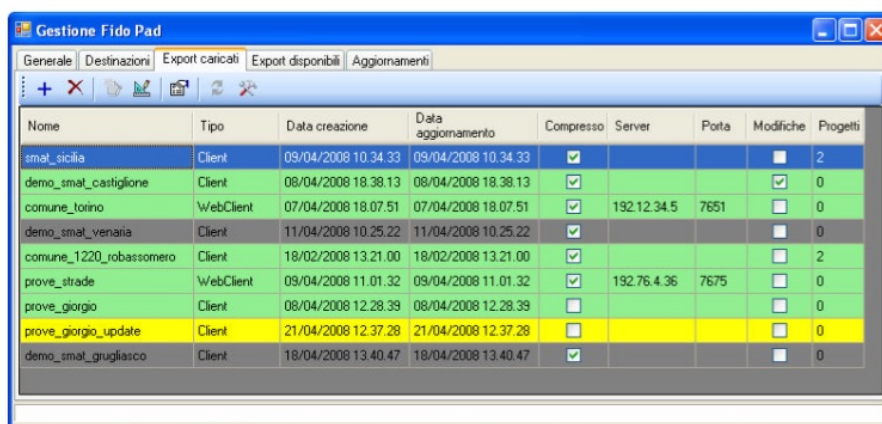
1.3.2 Gestore MapYou

In contemporanea allo sviluppo di MapYou è stata ideato il Gestore MapYou necessario alla preparazione del dispositivo mobile per il primo utilizzo.

L'applicazione si propone quale strumento di gestione di MapYou, in quanto consente:

- Il caricamento e l'aggiornamento dei dati che risiedono sul dispositivo mobile.
- L'esportazione degli aggiornamenti e dei progetti eseguiti sul campo dagli operatori.
- Il rinnovo della licenza d'uso installata sul dispositivo mobile.

I prerequisiti richiesti dall'applicativo riguardano il sistema operativo, utilizzabile su Windows XP o Windows 2000, la dotazione di una porta USB per il collegamento al dispositivo mobile e la possibilità di accedere ai dati delle mappe per il loro caricamento sul dispositivo mobile.



The screenshot shows a software window titled "Gestione Fido Pad". It has a menu bar with "Generale", "Destinazioni", "Export caricati", "Export disponibili", and "Aggiornamenti". Below the menu is a toolbar with icons for adding, deleting, and other functions. The main area contains a table with the following data:

Nome	Tipo	Data creazione	Data aggiornamento	Compresso	Server	Porta	Modifiche	Progetti
smat_sicilia	Client	09/04/2008 10.34.33	09/04/2008 10.34.33	<input checked="" type="checkbox"/>			<input type="checkbox"/>	2
demo_smat_castiglione	Client	08/04/2008 18.38.13	08/04/2008 18.38.13	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>	0
comune_torino	WebClient	07/04/2008 18.07.51	07/04/2008 18.07.51	<input checked="" type="checkbox"/>	192.12.34.5	7651	<input type="checkbox"/>	0
demo_smat_venaria	Client	11/04/2008 10.25.22	11/04/2008 10.25.22	<input checked="" type="checkbox"/>			<input type="checkbox"/>	0
comune_1220_robassomero	Client	18/02/2008 13.21.00	18/02/2008 13.21.00	<input checked="" type="checkbox"/>			<input type="checkbox"/>	2
prove_strade	WebClient	09/04/2008 11.01.32	09/04/2008 11.01.32	<input checked="" type="checkbox"/>	192.76.4.36	7675	<input type="checkbox"/>	0
prove_giorgio	Client	08/04/2008 12.28.39	08/04/2008 12.28.39	<input type="checkbox"/>			<input type="checkbox"/>	0
prove_giorgio_update	Client	21/04/2008 12.37.28	21/04/2008 12.37.28	<input type="checkbox"/>			<input type="checkbox"/>	0
demo_smat_grugliasco	Client	18/04/2008 13.40.47	18/04/2008 13.40.47	<input checked="" type="checkbox"/>			<input type="checkbox"/>	0

Figura 1.3: Esempio schermata Gestore MapYou

L'applicativo presenta cinque diverse schede:

1. **Generale** riportante alcune informazioni relative all'applicazione;
2. **Destinazioni**, in cui sono elencate le possibili destinazioni dei dati sul dispositivo mobile;
3. **Mappe caricate** contenente l'elenco di tutte le mappe presenti sul dispositivo mobile. Le funzionalità disponibili consentono all'utente di aggiungere o rimuovere mappe, di aggiornarle e modificarne le proprietà. Da questo pannello è possibile anche accedere alla funzionalità che consente di scaricare dal dispositivo mobile gli aggiornamenti effettuati sul campo e i progetti;
4. **Mappe disponibili**, in cui sono elencate le mappe disponibili per il caricamento sul dispositivo mobile;
5. **Aggiornamenti** attraverso il quale l'utente può rinnovare la licenza d'uso o installare la versione più recente dell'applicazione.

Capitolo 2

Tecnologie Utilizzate

In questo capitolo vengono descritte le tecnologie e librerie utilizzate nel corso dello sviluppo dei progetti.

2.1 Libreria JMTP

Per le ragioni progettuali che verranno esplicate nell'opportuno capitolo è stata di grande importanza nello sviluppo software la libreria JMTP, integrata con il linguaggio di programmazione Java.

JMTP è un progetto finalizzato a consentire un accesso di tipo MTP (Media Transfer Protocol) compatibile con lettori multimediali portatili ed è stato progettato con l'obiettivo di supportare i principali ambienti come Windows, Linux e Mac OS X utilizzando la stessa API.

Nato nel 2008 attraverso il primo rilascio di una beta, ora il progetto risulta archiviato senza ricevere ulteriori sviluppi ed aggiornamenti. L'ultima release, sempre in versione beta, risale al 2010. La libreria priva di una valida documentazione e di istruzioni per il corretto utilizzo ha creato un notevole impegno per il suo impiego nella realizzazione del progetto.

Le principali funzionalità fornite agli sviluppatori sono:

- Rilevazione dei dispositivi multimediali collegati.
- Navigazione dentro il filesystem della memoria del dispositivo.
- Inserimento di file multimediali nel dispositivo.

Questa libreria è stata largamente utilizzata nell'applicativo desktop per la parte di gestione di un dispositivo Android ad esso collegato.

2.2 Android Debug Bridge

Android Debug Bridge (ADB) è uno strumento a riga di comando che permette di comunicare con un dispositivo collegato o un'emulatore Android attivo. Si tratta di un programma client-server formato da tre componenti:

1. Un client, che viene eseguito sul computer che intende comunicare col dispositivo. È possibile richiamarlo attraverso una shell utilizzando un comando adb;
2. Un server eseguito come processo in background sul computer che gestisce la comunicazione tra il client e il demone adb in esecuzione su un emulatore o dispositivo;
3. Un demone eseguito come processo in background su ogni istanza di emulatore o dispositivo.

2.2.1 Sintassi

È possibile utilizzare i comandi adb da riga di comando o su di uno script. La sintassi generica è:

```
adb [-d|-e|-s <serialNumber>] <command>
```

Figura 2.1: Sintassi generica comando ADB

In presenza di un solo emulatore in esecuzione oppure un solo dispositivo collegato, il comando adb viene inviato di default a quel dispositivo; in caso contrario è necessario utilizzare l'opzione -d, -e oppure -s per specificare il dispositivo di destinazione.

2.2.2 Principali funzionalità

La gamma di comandi a disposizione degli sviluppatori, rende possibile numerose operazioni, tra le più importanti sono presenti:

- **Rilevazione dei dispositivi Android connessi** Genera una lista di dispositivi collegati al server ADB, specificando due diverse informazioni, il numero seriale del dispositivo ed il suo stato.
- **Installazione di una applicazione** Rende possibile l'installazione forzata di un file in formato apk, su un dispositivo specifico.
- **Gestione dei file del dispositivo** Attraverso i comandi pull e push è possibile copiare file dal computer al dispositivo o viceversa, quindi permettendo.
- **Utilizzo Wireless** Solitamente ADB è utilizzato attraverso un collegamento USB, tuttavia è possibile effettuare le stesse operazioni su Wi-Fi utilizzando la stessa rete.

Questi comandi sono stati utilizzati nell'applicativo desktop per ovviare alle mancanze della libreria JMTP.

2.3 Libreria JTattoo

JTattoo è stata sviluppata con l'intento di modificare l'interfaccia grafica standard offerta da JDK (Java Development Kit). La prima release pubblica è rilasciata nel Gennaio 2006 e ricevendo in seguito numerosi aggiornamenti sono stati sistemati vari problemi presenti in fase iniziale.

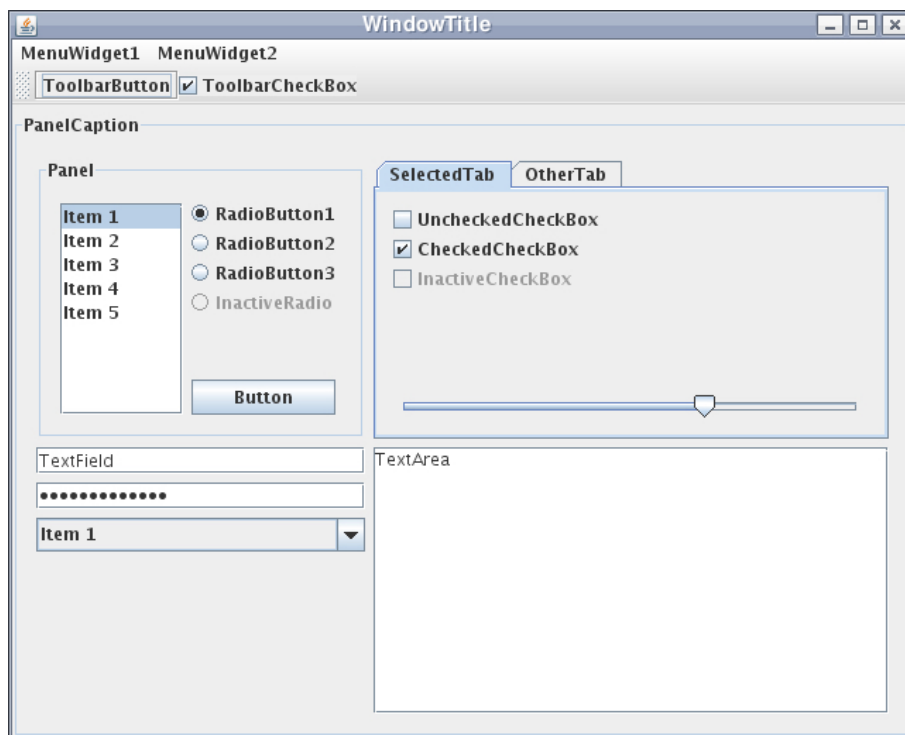


Figura 2.2: Esempio Swing, standard GUI Java

Consente ad uno sviluppatore il miglioramento dell'aspetto degli elementi grafici disponibili su Swing¹ utilizzando temi predefiniti composti da diversi aspetti grafici, offrendo anche la possibilità di personalizzare liberamente le

¹Libreria ufficiale per la realizzazione di interfacce grafiche in Java

proprietà della GUI inglobando successivamente queste modifiche nei temi già offerti da JTattoo.

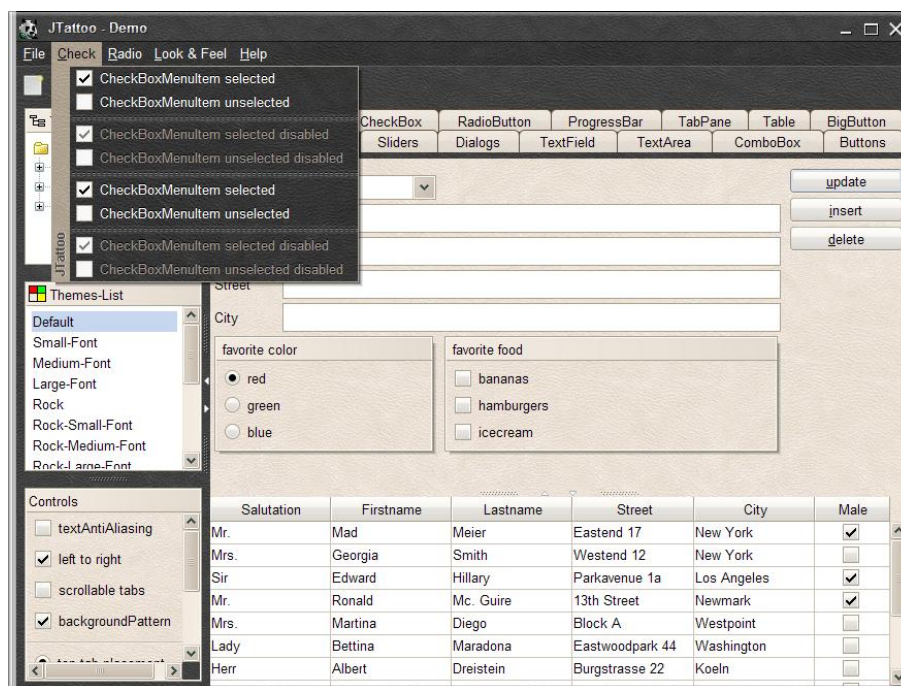


Figura 2.3: Tema predefinito TextureLookAndFeel

Questa libreria è stata utilizzata nell'applicativo desktop per modificare l'aspetto dell'interfaccia grafica standard di Java.

2.4 Mercurial

Mercurial è un software di controllo della versione del codice sorgente versatile e performante, distribuito gratuitamente ed efficientemente gestisce progetti di qualsiasi dimensione. È stato sviluppato quasi completamente in Python, con parti in C per motivi prestazionali, così rendendolo multiplatforma difatti è disponibile su tutte le principali piattaforme.

Nello sviluppo software è stato necessario utilizzarlo questo perché, trat-

tandosi di un lavoro in team, era fondamentale condividere il codice in modo corretto e tenere continuamente traccia di tutte le modifiche.

Tutto il codice sorgente dei due progetti è stato memorizzato su due repository Mercurial di www.bitbucket.org

Bitbucket

Bitbucket è un servizio di hosting web-based che permette di memorizzare in internet progetti che utilizzano Mercurial o Git. Fornisce la possibilità di creare repository pubbliche, che tutti possono vedere e nelle quali, normalmente, è possibile contribuire al codice, e repository private, come quelle utilizzate per questi progetti, dove soltanto chi ha l'accesso può leggere e modificare i file.

Capitolo 3

Analisi

In questo capitolo verranno descritti i requisiti che le applicazioni devono rispettare e le funzionalità che queste devono implementare.

3.1 Descrizione generale

3.1.1 Prospettiva del prodotto desktop

Il software progettato e sviluppato è un'applicazione per la preparazione e gestione di dispositivi Android collegati e dei dati utilizzati dall'applicazione mobile spiegata in seguito.

3.1.2 Funzioni del prodotto desktop

- Visualizzazione dei dispositivi mobile collegati
- Installazione dell'applicazione mobile
- Preparazione della directory di lavoro per l'applicazione mobile
- Gestione delle mappe presenti sul dispositivo mobile
- Esportazione dei progetti disponibili sul dispositivo mobile

3.1.3 Prospettiva del prodotto mobile

Il progetto sviluppato in team con il collega Filippo Nicolini consiste in un'applicazione mobile per la visualizzazione ed interazione con dati georiferiti locali al dispositivo. Di seguito verranno elencate tutte le funzionalità del sistema, nonostante alcune di esse siano state implementate dal collega. Nel paragrafo 3.5 verranno analizzate nel dettaglio le funzionalità attinenti alla tesi.

3.1.4 Funzioni del prodotto mobile

- Visualizzazione della mappa contenente i dati georiferiti
- Ricerca e visualizzazione di elementi contenuti nei dati georiferiti
- Inserimento di elementi grafici per misurazione e progettazione
- Visualizzazione di progetti esistenti sul dispositivo
- Creazione e salvataggio di progetti

3.1.5 Caratteristiche dell'utente finale

L'utente finale è generalmente un tecnico che effettua interventi esterni utilizzando l'applicativo mobile per interagire con le informazioni contenute e l'applicativo desktop per esportare progetti e lavori effettuati durante la giornata lavorativa. Dato che l'utente è già abituato all'utilizzo delle precedenti soluzioni prodotte da EBWorld, è necessario implementare le principali funzionalità in modo analogo alla versione precedente dell'applicazione desktop e mobile.

3.1.6 Ambiente d'utilizzo dei progetti

Il software desktop deve essere eseguibile, con buone prestazioni, su computer di fascia media di mercato che eseguono un sistema operativo Windows,

Mac o Linux. L'ambiente d'utilizzo per l'applicativo mobile dev'essere uno smartphone o un tablet Android di fascia media.

3.1.7 Linguaggio e ambienti di sviluppo

La scelta del linguaggio di programmazione è stata vincolata dalle richieste e dalla piattaforma di sviluppo utilizzata per l'applicativo mobile. Questo perché era necessario utilizzare un linguaggio che coincidesse ad entrambe fornendo anche il requisito di multi-piattaforma all'applicativo desktop.

Il linguaggio di programmazione scelto per implementare i progetti è Java con l'utilizzo di librerie esterne per migliorare e velocizzare lo sviluppo degli applicativi.

Per quanto riguarda la scelta degli ambienti di sviluppo si sono considerati gli IDE con maggiore diffusione ed ufficialità facendo ricadere la scelta su Android Studio risultante ambiente di sviluppo ufficiale per la piattaforma Android ed Eclipse per il progetto desktop in quanto principalmente utilizzato per lo sviluppo di applicativi Java.

3.2 Applicativi online o offline

Sempre un maggior numero di applicazioni sia desktop che mobile vengono portate online grazie all'enorme diffusione di Internet ed alla semplicità di connettersi ad esso. La tematica di applicativi online o offline è stata affrontata prima di avviare la progettazione e lo sviluppo. Di seguito vengono analizzati vantaggi e svantaggi dei due casi sopra citati.

3.2.1 Vantaggi e svantaggi online

Lo sviluppo di un applicazione collegata alla rete rende possibile l'inserimento di un vasto numero di funzionalità, dal più semplice collegamento ai

social network al più complesso sviluppo di un protocollo per interscambio dati. Un ambito da tenere in considerazione è la sicurezza, difatti un'applicativo in rete che ha l'accesso ai dati dell'utente potrebbe causare notevoli danni in caso di utilizzo da utenti non autorizzati. Un ulteriore elemento a favore è la gestione dei dati che può avvenire real-time rendendoli disponibili per ulteriori applicativi online sfruttando il collegamento alla rete. Queste possibilità possono sicuramente migliorare la sua utilità ma allo stesso tempo obbligano ad un aumento del tempo di progettazione, di implementazione e del relativo costo di sviluppo.

3.2.2 Vantaggi e svantaggi offline

Le caratteristiche del dispositivo sul quale si sviluppa un applicativo offline limitano notevolmente le funzionalità implementabili. Per la gestione dei dati tra differenti dispositivi è necessario instaurare un collegamento fisico e creare un protocollo di rilevazione e comunicazione in caso di presenza di differenti sistemi utilizzati dai dispositivi. L'ambito della sicurezza in un applicativo offline risulta allo stesso modo importante tuttavia va considerato che, a differenza di applicativi online, i soli dati contenuti sono locali e quindi in caso di utilizzo dello stesso da parte di un utente non autorizzato i danni risulterebbero limitati. La riduzione del numero di funzionalità disponibili per un'applicazione offline generalmente equivale ad un minor tempo di progettazione ed implementazione riducendo così il costo di realizzazione.

3.2.3 Scelta per applicativi sviluppati

L'analisi effettuata sulle condizioni di utilizzo dell'applicativo mobile ha fatto ricadere la scelta sullo sviluppo di applicativi offline. La principale motivazione difatti riguarda gli ambienti nei quali il tecnico esterno opera, normalmente sono scantinati di edifici, interventi sotto il manto stradale oppure anche luoghi fuori città. In queste situazioni è raramente assicurata la presenza di una connessione stabile alla rete e quindi la sincronizzazione dei

dati attraverso essa risulterebbe precaria. Oltre all'invio, anche il caricamento potrebbe avere rallentamenti o blocchi lasciando l'operatore senza dati da visualizzare bloccando così l'intervento. Attraverso l'utilizzo di un'applicativo offline con la persistenza dei dati in locale, le operazioni effettuate possono essere salvate in qualsiasi condizione con il solo obbligo di esportarle in sede centrale per inserirle nella sorgente dati.

3.3 Funzionalità del sistema desktop

Verrà analizzato il software desktop descrivendo nel dettaglio le funzionalità a cui deve provvedere, esplicando i requisiti funzionali che queste implicano.

3.3.1 Visualizzazione e selezione dispositivi

Descrizione: Il software deve essere in grado di controllare all'avvio quali sono i dispositivi connessi specificando quali sono già stati preparati al primo utilizzo, mostrarne la lista ed abilitare l'utente alla selezione di un dispositivo.

Requisiti funzionali

1. All'avvio il sistema deve provvedere alla verifica dei dispositivi connessi.
2. Il sistema deve controllare quali dispositivi risultano già pronti per il primo utilizzo
3. Il sistema deve fornire all'utente una lista di dispositivi connessi.
4. Il sistema deve fornire all'utente un'interfaccia per selezionare il dispositivo desiderato.

3.3.2 Preparazione dispositivo

Descrizione: Il sistema deve connettersi al dispositivo selezionato ed avviare l'installazione dell'applicazione mobile creando successivamente il workspace utilizzato dallo stesso applicativo.

Requisiti funzionali

1. Il sistema deve installare l'applicativo mobile presente in locale sul dispositivo selezionato.
2. Il sistema deve creare le cartelle principale all'interno della memoria interna del dispositivo selezionato.
3. Il sistema deve copiare i file fondamentali utilizzati dall'applicativo mobile dentro il dispositivo selezionato.

3.3.3 Gestione file di configurazione

Descrizione: Il sistema deve essere in grado di gestire interamente il file di configurazione in formato xml utilizzato dall'applicazione mobile per la corretta lettura dei dati archiviati sul dispositivo.

Requisiti funzionali

1. Il sistema deve caricare in modo corretto tutti i dati contenuti dal file di configurazione situato all'interno del dispositivo mobile.
2. Il sistema deve poter modificare suddetto file in base alle operazioni effettuate dall'utente.
3. Il sistema deve sostituire all'interno del dispositivo in gestione il precedente file di configurazione con quello modificato.

3.3.4 Visualizzazione e selezione mappe del computer desktop e del dispositivo mobile

Descrizione: Il sistema deve essere in grado di verificare quali mappe sono contenute localmente nel computer e nel dispositivo in gestione mostrandone una lista ed abilitare l'utente alla selezione di una o più mappe.

Requisiti funzionali

1. Il sistema deve provvedere autonomamente a verificare le mappe presenti sul computer in utilizzo e sul dispositivo in gestione.
2. Il sistema deve fornire all'utente una lista di mappe disponibili.
3. Il sistema deve fornire all'utente un'interfaccia per selezionare la mappa desiderata.

3.3.5 Inserimento e rimozione mappe

Descrizione: Il software deve essere in grado di inserire e rimuovere le mappe, selezionate dall'utente, dalla memoria del dispositivo mobile in gestione.

Requisiti funzionali

1. Il sistema deve memorizzare le mappe in inserimento ed in rimozione selezionate dall'utente.
2. Il sistema deve mostrare a schermo tutte le operazioni di modifica effettuate dall'utente.
3. Il sistema deve rimuovere il contenuto delle mappe selezionate dal dispositivo mobile.
4. Il sistema deve copiare il contenuto delle mappe selezionate all'interno del dispositivo mobile.

3.3.6 Gestione file di definizione progetti

Descrizione: Il sistema deve essere in grado di gestire interamente il file di definizione dei progetti in formato xml utilizzato dall'applicazione mobile per la corretta lettura dei progetti.

Requisiti funzionali

1. Il sistema deve caricare in modo corretto tutti i dati contenuti dal file xml situato all'interno del dispositivo mobile.
2. Il sistema deve poter modificare suddetto file in base alle operazioni riguardanti i progetti, effettuate dall'utente.
3. Il sistema deve sostituire all'interno del dispositivo, il precedente file xml con quello modificato.

3.3.7 Visualizzazione e selezione progetti

Descrizione: Il sistema deve essere in grado di controllare quali progetti sono contenuti dal dispositivo in gestione, mostrarne una lista ed abilitare l'utente alla selezione di uno o più progetti.

Requisiti funzionali

1. Il sistema deve provvedere autonomamente a verificare i progetti presenti sul dispositivo in gestione.
2. Il sistema deve fornire all'utente una lista di progetti disponibili.
3. Il sistema deve fornire all'utente un'interfaccia per selezionare il progetto desiderato.

3.3.8 Esportazione o rimozione progetto

Descrizione: Il software deve essere in grado di esportare o rimuovere progetti, selezionati dall'utente, prelevandoli dal dispositivo mobile in gestione.

Requisiti funzionali

1. Il sistema deve memorizzare i progetti in esportazione ed in rimozione selezionate dall'utente.
2. Il sistema deve mostrare a schermo tutte le operazioni di modifica effettuate dall'utente.
3. Il sistema deve rimuovere il contenuto dei progetti selezionati dal dispositivo mobile in gestione.
4. Il sistema deve esportare il contenuto dei progetti selezionati sul computer in uso.

3.4 Moduli del sistema desktop

Attraverso l'analisi delle funzionalità offerte dall'applicativo desktop è stato deciso uno sviluppo modulare, per permettere la massima scalabilità rendendo possibile future implementazioni di nuove funzioni. I moduli individuati sono tre:

- Dispositivi
- Mappe
- Progetti

3.4.1 Modulo Dispositivi

Lo scopo di questo modulo è di contenere tutte le funzionalità di gestione dei dispositivi. Si occupa quindi di rilevare i dispositivi mobile collegati, di analizzare il loro contenuto, di prepararli al primo utilizzo e fornire la visualizzazione della lista di dispositivi tra i quali scegliere quello desiderato.

Casi d'uso

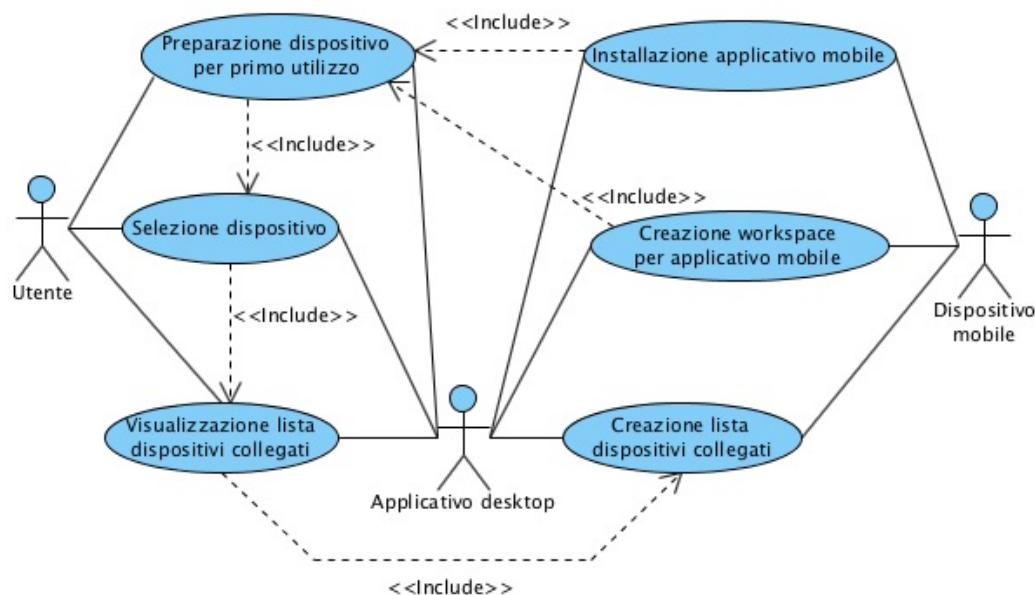


Figura 3.1: Diagramma dei casi d'uso del modulo dispositivi

Definite le funzionalità presenti nel modulo, il diagramma dei casi d'uso mostra graficamente le operazioni disponibili all'utente e le relative operazioni eseguite dall'applicativo desktop sui dispositivi mobile collegati.

3.4.2 Modulo Mappe

Questo modulo si occupa della gestione delle mappe. Qui sono disponibili tutte le funzionalità di lettura del file di configurazione contenuto dal

dispositivo in gestione, la ricerca delle mappe disponibili sul dispositivo e sul computer in utilizzo, l'inserimento o la rimozione di nuove mappe nel dispositivo. Si occupa quindi di tutte le modifiche apportate sulla lista delle mappe disponibili visualizzata a schermo comprendendo la modifica del file di configurazione necessario all'applicazione e del suo inserimento nel dispositivo.

Casi d'uso

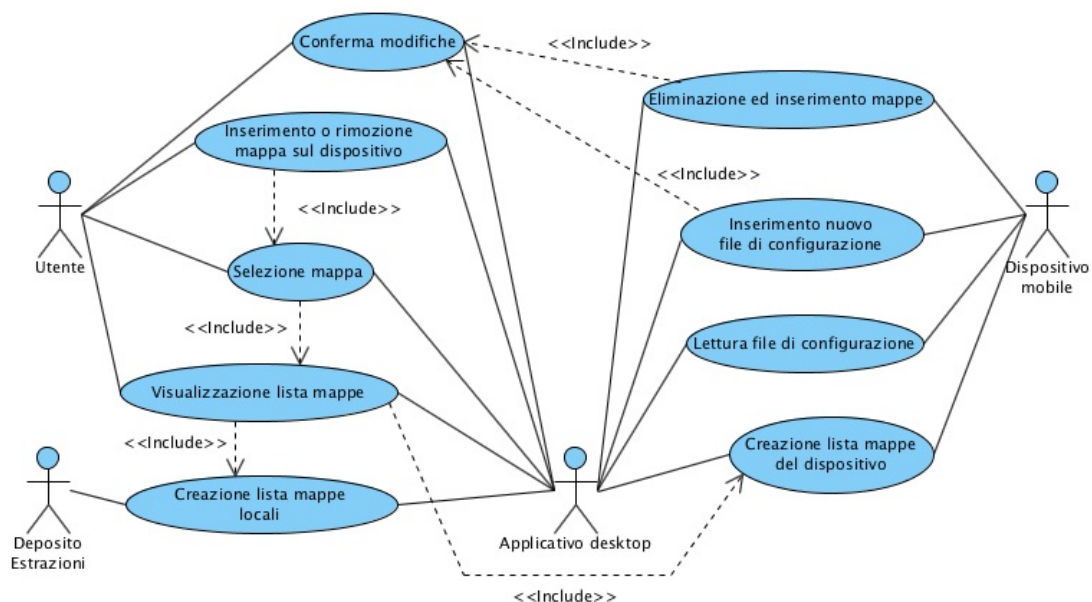


Figura 3.2: Diagramma dei casi d'uso del modulo mappe

Definite le funzionalità presenti nel modulo, il diagramma dei casi d'uso mostra graficamente le operazioni disponibili all'utente e le relative operazioni eseguite dall'applicativo desktop sul dispositivo mobile in gestione ed introduce la presenza di un deposito di estrazioni il cui contenuto e struttura verrà spiegato nel capitolo di progettazione.

3.5 Funzionalità del sistema mobile

Verranno descritte di seguito nel dettaglio le funzionalità, implementate per questa tesi, a cui l'applicativo mobile deve provvedere, esplicitando i requisiti funzionali che queste implicano.

3.5.1 Gestione file dei progetti

Descrizione: Il sistema deve essere in grado di gestire interamente il file dei progetti in formato .xml utilizzati dall'applicazione mobile.

Requisiti funzionali

1. Il sistema deve caricare in modo corretto tutti i dati contenuti dal file xml situato all'interno del dispositivo mobile.
2. Il sistema deve poter modificare suddetto file in base alle operazioni riguardanti i progetti, effettuate dall'utente.
3. Il sistema deve sostituire il precedente file con quello aggiornato.

3.5.2 Visualizzazione e selezione progetti

Descrizione: L'utente deve poter visualizzare quali progetti sono presenti all'interno del dispositivo e selezionare il progetto desiderato.

Requisiti funzionali

1. Il sistema deve provvedere a verificare i progetti memorizzati sul dispositivo.
2. Il sistema deve fornire all'utente una lista di progetti disponibili.
3. Il sistema deve fornire all'utente un'interfaccia per selezionare il progetto desiderato.

3.5.3 Caricamento e salvataggio progetto

Descrizione: Il sistema deve essere in grado di caricare correttamente il file xml contenente la struttura del progetto selezionato e salvarlo dopo ulteriori modifiche da parte dell'utente.

Requisiti funzionali

1. Il sistema deve caricare in modo corretto tutti i dati contenuti nella struttura del progetto in caricamento.
2. Il sistema deve poter modificare suddetto file in base alle operazioni effettuate dall'utente.
3. Il sistema deve sostituire il precedente file con quelli aggiornati.

3.6 Moduli del sistema mobile

A seguito della suddivisione del lavoro con il collega Filippo Nicolini, per lo sviluppo dell'applicativo mobile è stato identificato un solo modulo da implementare:

- Progetti

3.6.1 Modulo Progetti

Questo modulo si occupa della gestione dei progetti. Qui sono disponibili tutte le funzionalità di caricamento dei progetti disponibili nel dispositivo mobile, di caricamento e salvataggio della struttura del progetto in gestione. Si occupa quindi della memorizzazione di tutte le modifiche effettuate dall'utente ai progetti creati durante l'utilizzo dell'applicativo mobile.

Casi d'uso

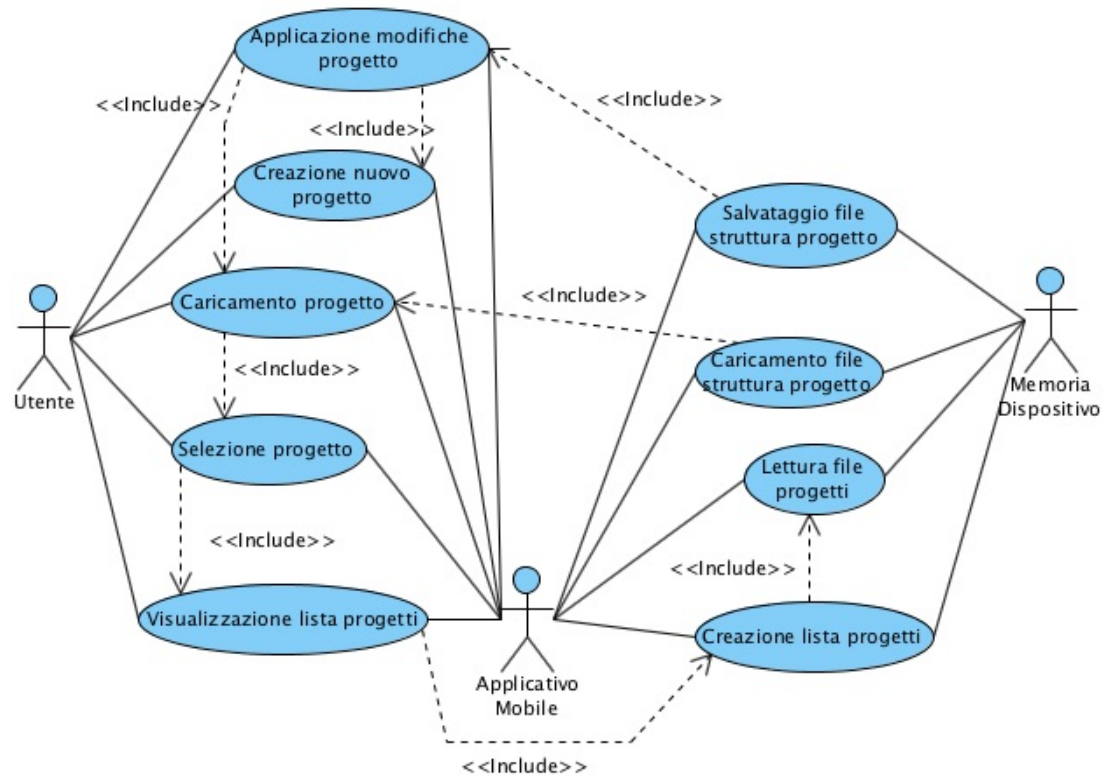


Figura 3.4: Diagramma dei casi d'uso del modulo progetti dell'applicativo mobile

Definite le funzionalità presenti nel modulo il diagramma dei casi d'uso mostra graficamente le operazioni disponibili all'utente e le relative operazioni eseguite dall'applicativo mobile sulla memoria interna la cui struttura e contenuto verrà spiegato nel capitolo di progettazione.

Capitolo 4

Progettazione

In questo capitolo verranno esposte le strutture dei progetti, indicando le scelte progettuali che hanno portato alla loro definizione.

4.1 Applicativo desktop

4.1.1 Architettura principale

L'architettura principale applicata è stata realizzata attraverso l'utilizzo del pattern Model-View-Controller molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione ad oggetti. La struttura del MVC è basata sulla separazione dei compiti fra i componenti software che interpretano tre ruoli:

- **MODEL:** fornisce i metodi per accedere ai dati utili all'applicazione.
- **VIEW:** visualizza i dati contenuti nel model e si occupa dell'interazione con utenti ed agenti.
- **CONTROLLER:** riceve i comandi dell'utente e li attua modificando lo stato degli altri due componenti.

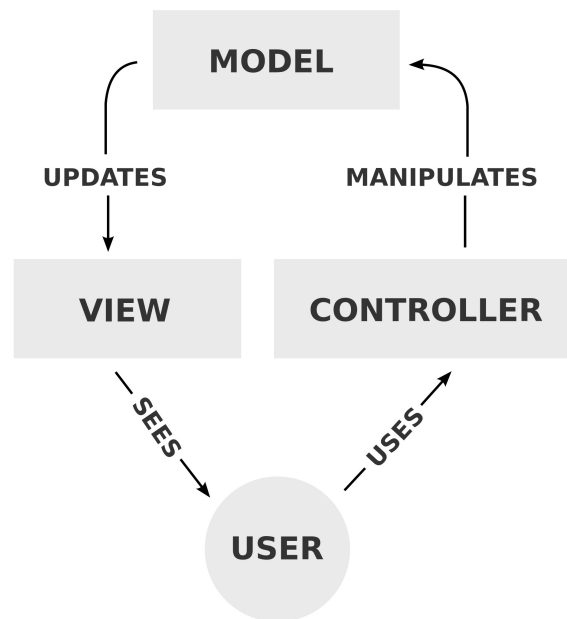


Figura 4.1: Interazione tra componenti del MVC

Il pattern MVC aumenta la complessità architetturale perché introduce classi aggiuntive dovute alla separazione tra model, view e controller; tuttavia i vantaggi risultano maggiormente d'impatto, difatti è possibile il ri-uso dei componenti del modello utilizzabili in differenti view, una semplice implementazione, testing e manutenibilità dei dati contenuti dal modello ed un facile supporto per nuovi tipi di client dato dal fatto che è sufficiente implementare nuove view e controller utilizzando i già presenti oggetti del modello.

Le motivazioni sull'utilizzo di questo pattern architetturale coincidono con i vantaggi descritti in precedenza dato che soddisfano il requisito di scalabilità richiesto dall'azienda.

Applicazione ai moduli

La suddivisione in moduli dell'applicativo, illustrata nella sezione 3.4, combinata all'utilizzo di questo pattern architetturale ha reso possibile una veloce strutturazione dell'applicativo illustrata nella figura seguente.

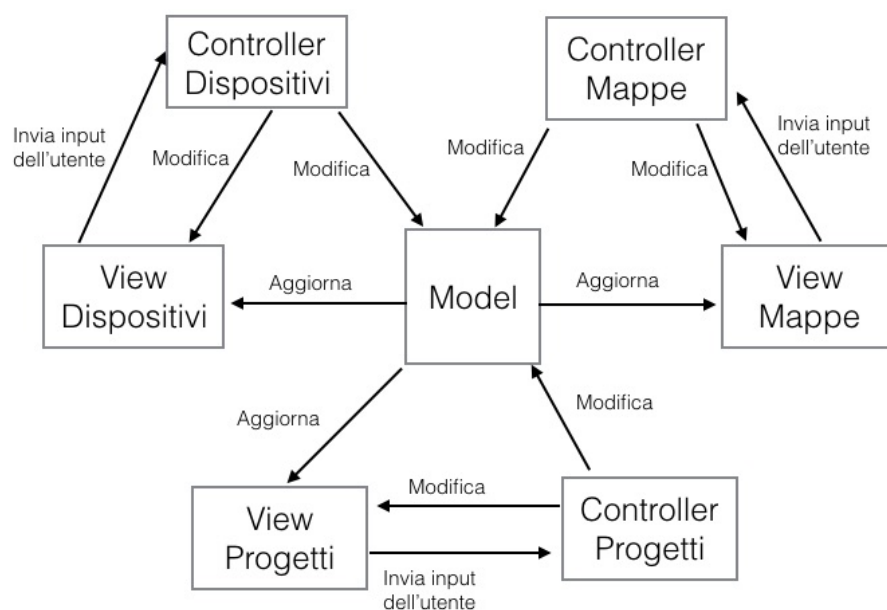


Figura 4.2: MVC applicato ai moduli

Classi ricavate

Le principali classi ricavate successive alla creazione dell'architettura principale sono le seguenti:

- **Model**: contenente i dati utilizzati dall'applicativo.
- **DevicesView**: interfaccia grafica che rende visibile all'utente la lista di dispositivi collegati con la possibilità di selezione e di preparazione di un dispositivo al primo utilizzo oppure il passaggio ai successivi moduli dell'applicativo¹.

¹Il flusso di utilizzo dell'applicativo desktop verrà illustrato nel paragrafo successivo.

- **ExportsView**²: interfaccia grafica che rende visibili la lista di mappe contenute nel dispositivo in gestione, la lista di mappe contenute dal computer in utilizzo con la possibilità di selezione ed inserimento nel dispositivo o rimozione dal dispositivo di una mappa.
- **ProjectsView**: interfaccia grafica che rende visibili la lista di progetti contenuti nel dispositivo in gestione con la possibilità di selezione ed esportazione o rimozione dal dispositivo di un progetto.
- **DevicesController**: classe che riceve gli input dell'utente passati dalla corrispondente view ed avvia le funzionalità di ricarica della lista dei dispositivi collegati, di preparazione al primo utilizzo installando l'applicazione mobile e creando il workspace sul dispositivo selezionato oppure avvia il successivo modulo selezionato dall'utente.
- **ExportsController**: controller che memorizza le operazioni effettuate dall'utente ed alla conferma avvia il trasferimento delle nuove mappe inserite dall'utente e la rimozione delle mappe eliminate dall'utente sul dispositivo in gestione.
- **ProjectsController**: controller che memorizza le operazioni effettuate dall'utente ed alla conferma avvia l'esportazione in locale dei progetti selezionati dall'utente e la rimozione dei progetti eliminati dall'utente sul dispositivo in gestione.

²Il nome export equivale ad estrazione. EBWorld utilizza questo termine per indicare la porzione della mappa presente in azienda estratta per un determinato cliente. D'ora in avanti il termine mappa ed estrazione si equivarranno.

Diagramma classi

Di seguito viene proposto il diagramma delle classi del modulo Dispositivi. I moduli mancanti presentano schemi simili strutturati sempre utilizzando il pattern architetturale MVC.

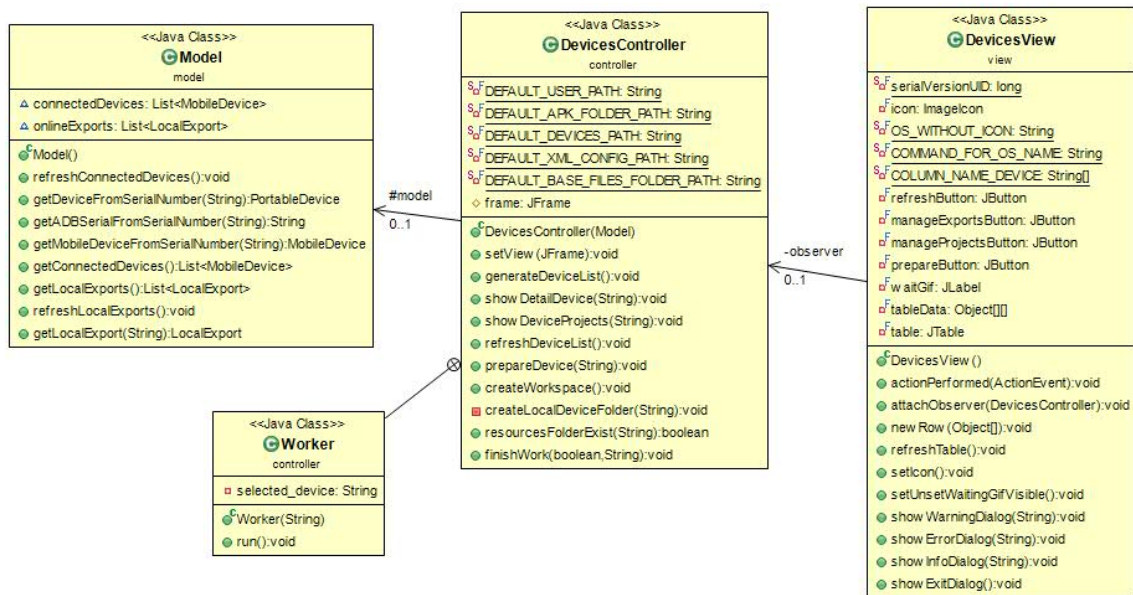


Figura 4.3: Diagramma classi del modulo Dispositivi

4.1.2 Flusso di utilizzo dell'applicativo

I moduli ricavati dall'analisi dell'applicativo desktop necessitano un collegamento anche se gestiscono ambiti diversi così da offrire all'utente un'esperienza di utilizzo lineare. Il seguente schema illustra il flusso di utilizzo dell'applicativo descrivendo in linea generale le operazioni effettuate per il passaggio tra i moduli.

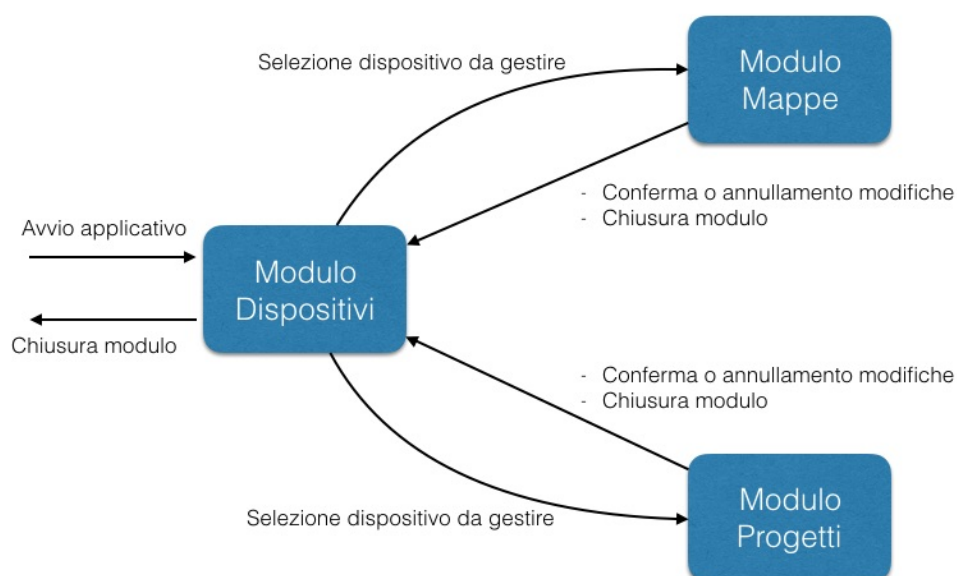


Figura 4.4: Flusso utilizzo dei moduli

Come si può notare dallo schema, il primo modulo utilizzabile dall'utente è il Modulo Dispositivi. La decisione è obbligata dal fatto che un dispositivo non può essere gestito in ambito mappe o progetti senza essere preparato al primo utilizzo perché risulterebbe mancante del workspace adatto alla memorizzazione di questi ultimi. Un ulteriore aspetto da tenere in considerazione riguarda la richiesta di un singolo dispositivo in gestione da parte dei moduli Mappe e Progetti. Per soddisfare questo requisito senza mostrare nuovamente l'intera lista dei dispositivi nei moduli Mappe e Progetti è stato implementato il passaggio del dispositivo mobile selezionato dal modulo Dispositivi agli altri.

4.1.3 Comunicazione tra applicativo desktop e dispositivo mobile

La principale e fondamentale funzionalità progettata e realizzata è la comunicazione tra l'applicativo desktop e il dispositivo mobile. Nelle fasi iniziali sono sorti problemi riguardanti l'impossibilità di lavorare su un dispositivo Android collegato attraverso un'applicativo java dato che non risultava visibile a quest'ultimo. Una parziale soluzione è stata ottenuta attraverso l'utilizzo dei comandi ADB illustrati nel paragrafo 2.2 ed inseriti all'interno dell'applicativo java, con i quali è possibile gestire in maniera completa un dispositivo Android collegato al computer. L'obbligo di tradurre ed estrarre informazioni utilizzabili dai risultati delle operazioni a riga di comando utilizzate, avrebbe complicato e rallentato eccessivamente lo sviluppo dei progetti, tuttavia attraverso l'inserimento della libreria JMTP descritta nel paragrafo 2.1, è stato raggiunto un buon compromesso. La mancanza di funzionalità nella libreria JMTP ha vincolato l'utilizzo di alcuni comandi ADB ripartendo le funzionalità in questa maniera.

Funzionalità utilizzate di JMTP

- Rilevazione dei dispositivi collegati
- Navigazione nel filesystem del dispositivo
- Creazione ed eliminazione di directory
- Trasferimento di file o cartelle dal computer al dispositivo

Funzionalità utilizzate di ADB

- Rilevazione dei dispositivi Android collegati
- Installazione del file contenente l'applicativo mobile sviluppato
- Trasferimento di file o cartelle dal dispositivo al computer

L'utilizzo combinato ha portato alla creazione della classe `MobileDevice` contenente il numero seriale estratto da ADB e l'oggetto della libreria JMTP che rappresenta il dispositivo mobile collegato. Questa necessità deriva dall'obbligo di inserire nel comando ADB il seriale del dispositivo sul quale va effettuata l'operazione. Il seguente diagramma rappresenta la definizione della classe `MobileDevice`.

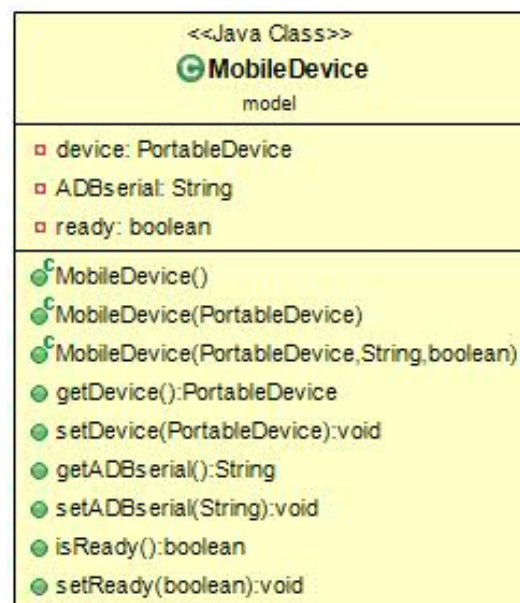


Figura 4.5: Diagramma della classe `MobileDevice`

4.1.4 Manager

L'analisi delle funzionalità presenti in ogni modulo ha riscontrato diversi funzioni in comune come ad esempio la lettura ed il salvataggio di file xml e la navigazione nel filesystem della memoria dei dispositivi collegati. Per questo motivo e per facilitare future funzionalità implementabili sono state create le seguenti classi per evitare la duplicazione del codice:

- **DeviceManger**: contenente tutti i metodi necessari alla comunicazione con un dispositivo mobile collegato al computer, alla navigazione nel filesystem del dispositivo, all'estrazione, rimozione o inserimento di file nel dispositivo.
- **FileXMLManger**: contenente i metodi necessari alla lettura di file xml utilizzati dall'applicativo desktop per analizzare il contenuto del dispositivo mobile ed al salvataggio dei file xml aggiornati in seguito alle modifiche apportate dall'utente.
- **ConfigManger**: classe contenente i metodi per gestire il file config.xml³ del dispositivo in gestione permettendo all'applicativo di memorizzare le modifiche effettuate dall'utente ed applicarle alla conclusione delle operazioni.
- **ProjectManger**: contenente i metodi per gestire il file progetti.xml del dispositivo in gestione permettendo all'applicativo di memorizzare le modifiche effettuate dall'utente ed applicarle alla conclusione delle operazioni.
- **LocalExportManger**: classe contenente i metodi necessari gestire le mappe contenute dal computer in uso.

³Nel sezione successiva verrà illustrata la struttura di tutti i file XML utilizzati dal applicativo desktop

4.1.5 Struttura dei file XML

L'analisi dell'applicativo desktop Gestore MapYou prodotto da EBWorld ha rivelato l'utilizzo di numerosi file xml utilizzati per la memorizzazione di diversi tipi di dati. XML (sigla di eXtensible Markup Language) è un metalinguaggio per la definizione di linguaggi di markup che consente di definire e controllare il significato degli elementi contenuti in un documento o testo.

Esempio.xml

```
1 <?xml version="1.0"?>
2 <libri>
3     <libro>
4         <autore>J.K.Rowlings</autore>
5         <titolo>Harry Potter e la pietra filosofale</titolo>
6         <editore>Salani</editore>
7     </libro>
8     <libro>
9         <autore>J.R.Tolkien</autore>
10        <titolo>Il signore degli Anelli</titolo>
11        <editore>Bonpiani</editore>
12    </libro>
13 </libri>
```

Il nome indica che si tratta di un linguaggio marcatore (markup language) estensibile in quanto permette di creare tag personalizzati come si può notare dal esempio sopra illustrato.

Nella progettazione sono stati creati file xml simili a quelli del precedente applicativo perché risultavano comprensibili e di facile utilizzo. Di seguito verranno illustrate e descritte le strutture dei file xml letti dall'applicativo desktop e la definizione delle classi java associate ad essi.

Current_export.xml

Current_export.xml è il file di definizione di una mappa, utilizzato dall'applicativo per determinare le estrazioni presenti nel computer in uso.

Current_export.xml

```
1 <Smallword_Export export_structure_version="1">
2   <Time>21/04/2008 11:50:01</Time>
3   <Base_Version_Id>20080421_115001</Base_Version_Id>
4   <Update_Id/>
5   <Raw_Folder_Up_To_Date/>
6   <Dati_Export>
7     <Name>Pesaro_25000_vettoriale</Name>
8     <Base_Version_Id build_time="12928" build_host="gio02">
9       20080421_115001
10    </Base_Version_Id>
11    <Update_Id/>
12    <Directories>270834</Directories>
13    <Files>665441</Files>
14    <Files_Size>504254917</Files_Size>
15  </Dati_Export>
16 </Smallword_Export>
```

La struttura ed il contenuto del file è stato definito in precedenza da EBWorld ed è presente in ogni estrazione con valori differenti per ogni mappa. L'applicativo utilizza i valori dei tag Time e Name e Base_Version_Id per identificare una estrazione memorizzandoli all'interno della classe LocalExport.

Config.xml

Config.xml è il principale file utilizzato dall'applicativo mobile per ricavare dati utili alla lettura delle mappe presenti nel dispositivo e dati generici utilizzati dall'applicativo per configurarsi.

Config.xml

```
1 <Config Internals="" Key="" NomeEseguibile="" Snapshot=""
  VersioneEseguibile="">
2   <Exports>
3     <Export CartellaDiLavoro="" Compattato=""
      DataAggiornamento="" DataExport=""
      FileExportStructure="" Nome="" PathExport=""/>
4     <Export CartellaDiLavoro="" Compattato=""
      DataAggiornamento="" DataExport=""
      FileExportStructure="" Nome="" PathExport=""/>
5   </Exports>
6 </Config>
```

Il file raffigurato contiene tre diversi tag:

- **Config:** Radice principale del file xml contenente attributi che identificano ad esempio la versione dell'applicativo mobile in utilizzo ed ulteriori valori utilizzati in fase di configurazione.
- **Exports:** Tag utilizzato come contenitore.
- **Export:** Tag rappresentativo di una estrazione o mappa all'interno del dispositivo mobile. Gli attributi al suo interno rappresentano tutti i dati necessari all'applicativo per conoscere il nome, la data di creazione e di aggiornamento, il percorso in memoria dei dati della mappa ed il percorso in memoria della cartella contenente i lavori e progetti riferiti alla stessa mappa.

Classi ricavate

Come definito dall'analisi, l'applicativo desktop deve essere in grado di gestire il contenuto del file descritto in precedenza, con le modifiche apportate dall'utente durante l'utilizzo del modulo Mappe. A conclusione della gestione delle mappe l'applicativo deve essere in grado di creare il nuovo file Config.xml e sostituirlo al precedente. Per rendere possibile queste operazioni sono state create le seguenti classi Java:

- **ConfigXML**: classe che raffigura l'intero file xml e contenente la lista di mappe contenute ed i valori degli attributi del tag Config.
- **ExportTag**: classe che raffigura il tag Export contenendo tutti i valori dei suoi attributi.

Diagramma Classi

Di seguito viene proposto il diagramma delle classi definite in precedenza.

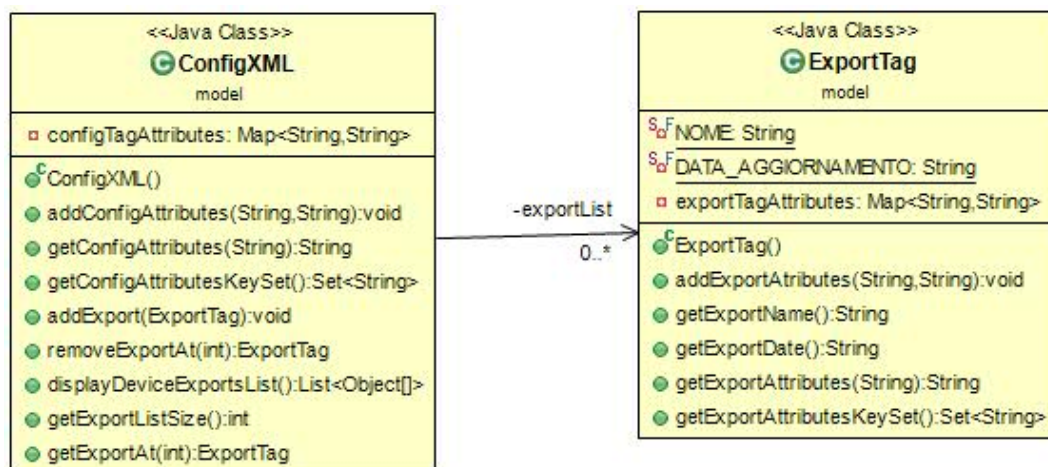


Figura 4.6: Diagramma delle classi ConfigXML e ExportTag

Progetti.xml

Progetti.xml è il file utilizzato dall'applicativo desktop per estrarre le informazioni dei progetti presenti all'interno del dispositivo mobile. Come illustrerà il paragrafo 4.2.3 all'interno del dispositivo mobile esiste un file xml di definizione dei progetti diverso collegato ad ogni mappa presente nel dispositivo.

Progetti.xml

```
1 <Projects ProgettoAttivo="">
2   <Project Nome="" DataProgetto=""/>
3   <Project Nome="" DataProgetto=""/>
4 </Projects>
```

Il file raffigurato contiene due tag:

- **Projects:** Radice principale del file xml che contiene il nome dell'ultimo progetto attivo utilizzato dall'applicativo mobile.
- **Project:** Tag rappresentativo di un progetto contenente il nome e la data corrispondente all'ultima modifica apportata da parte dell'utente.

Classi ricavate

Come definito dall'analisi, l'applicativo desktop deve essere in grado di gestire il contenuto del file descritto in precedenza, con le modifiche apportate dall'utente durante l'utilizzo del modulo Progetti. A conclusione della gestione dei progetti l'applicativo deve essere in grado di creare il nuovo file Progetti.xml e sostituirlo al precedente. Per rendere possibile queste operazioni sono state create le seguenti classi Java:

- **ProjectsXML:** classe che raffigura l'intero file xml e contenente la lista di progetti contenuti ed il valori dell'attributo ProgettoAttivo del tag Projects.
- **ProjectTag:** classe che raffigura il tag Project contenendo tutti i valori dei suoi attributi.

Diagramma Classi

Di seguito viene proposto il diagramma delle classi definite in precedenza.

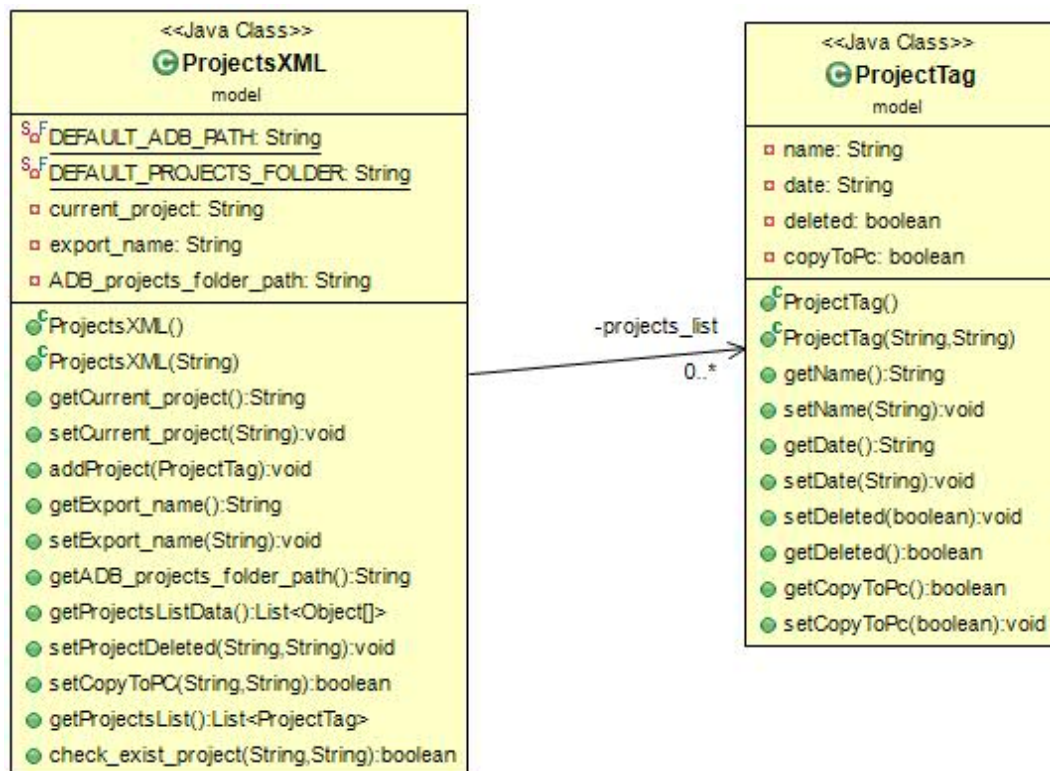


Figura 4.7: Diagramma delle classi ProjectsXML e ProjectTag

4.1.6 Workspace

Data la presenza di diversi file gestiti dall'applicativo è stata necessaria la creazione di un workspace locale utilizzato per differenti finalità spiegate in seguito.

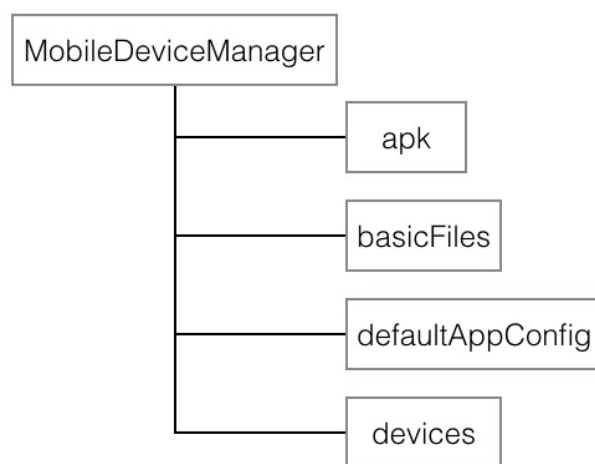


Figura 4.8: Struttura workspace locale

All'avvio dell'applicativo viene realizzato il workspace strutturato come in figura, nella cartella principale dell'utente collegato al computer in utilizzo. Nella cartella "apk" è presente il file di installazione dell'applicativo mobile che verrà utilizzato alla preparazione di un dispositivo mobile collegato. La cartella "basicFiles" contiene tutti i file che si vogliono inserire nella cartella principale del workspace dell'applicativo mobile, la cui copia verrà eseguita successivamente alla creazione dello stesso. La directory "defaultAppConfig" contiene i file xml definiti nel paragrafo precedente, utilizzati dall'applicativo nel caso risultino mancanti nel dispositivo mobile in gestione. L'ultima directory conterrà i file temporanei prelevati e creati per ogni dispositivo gestito dall'applicativo, organizzati in cartelle univoche nominate con il numero seriale del dispositivo.

Sorgente dati

Per completare la progettazione del workspace dell'applicativo desktop è necessario illustrare la struttura della directory utilizzata come sorgente dati delle estrazioni o mappe disponibili. È stata presa la decisione di dividerla dal workspace principale descritto in precedenza, perché generalmente la clientela di EBWorld utilizza una cartella di rete raggiungibile da tutti i computer in rete, come contenitore di mappe.

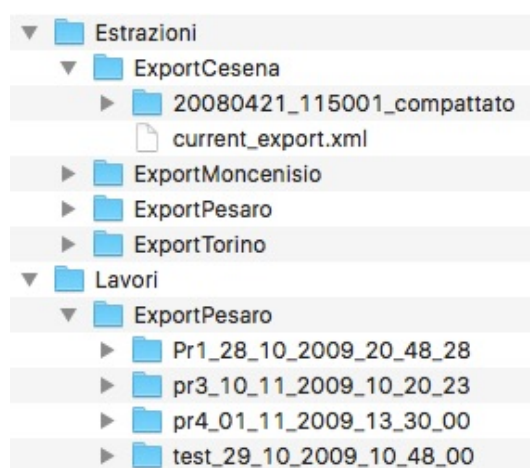


Figura 4.9: Struttura sorgente dati

Come si nota dall'esempio in figura, all'interno della directory principale sono presenti due cartelle. "Estrazioni" contiene i dati di tutte le mappe possedute dall'utente e la directory "Lavori" utilizzata dall'applicativo come cartella di destinazione dei progetti esportati dal dispositivo suddividendoli in cartelle nominate come la mappa nella quale sono stati realizzati. Sempre in figura viene mostrato il file `current_export.xml` definito nel paragrafo 4.1.5 grazie al quale l'applicativo definisce quali mappe sono presenti nella cartella "Estrazioni".

4.1.7 Progettazione GUI

La definizione dei view ossia interfacce grafiche utilizzate dall'utente per interagire con i dati contenuti dall'applicativo ed i dispositivi mobile collegati, ha avviato la realizzazione di numerosi mockup sempre più concreti fino ad arrivare alla soluzione ritenuta maggiormente chiara nell'esposizione dei dati e facilmente utilizzabile da parte dell'utente. Di seguito verranno mostrati i risultati.

Mockup DevicesView

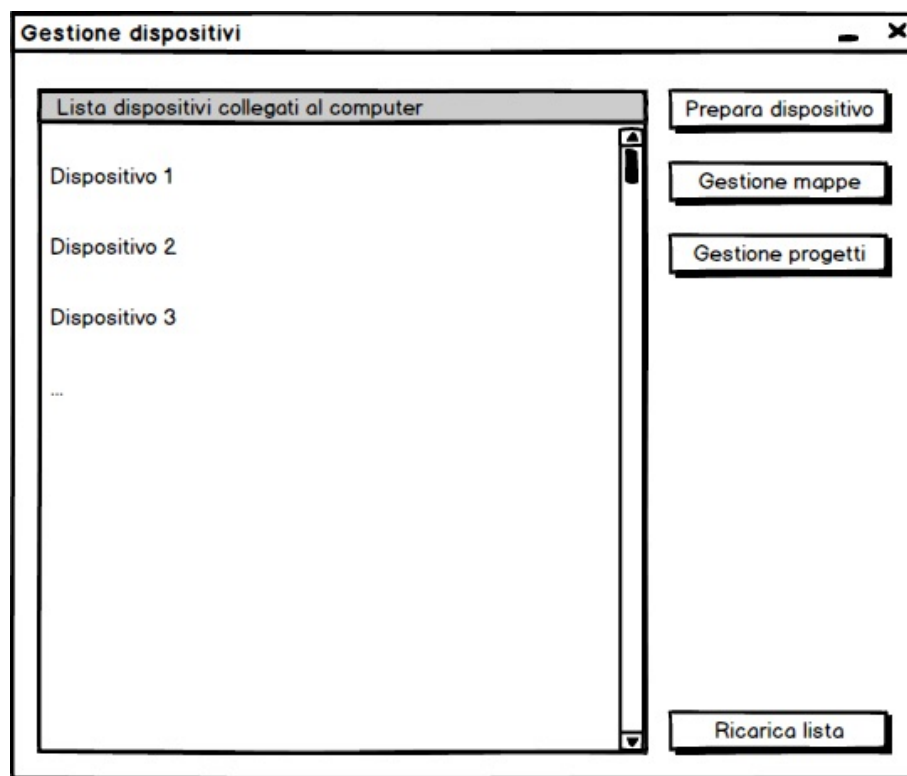


Figura 4.10: Mockup interfaccia grafica modulo Dispositivi

Il mockup in questione fa risaltare una disposizione molto chiara dei vari elementi che compongono la finestra mettendo subito in evidenza la lista dei dispositivi collegati e dando la possibilità all'utente di avviare la preparazio-

ne di un dispositivo oppure di entrare nei moduli successivi dell'applicativo attraverso l'utilizzo di bottoni posizionati a destra delle lista.

Mockup ExportsView

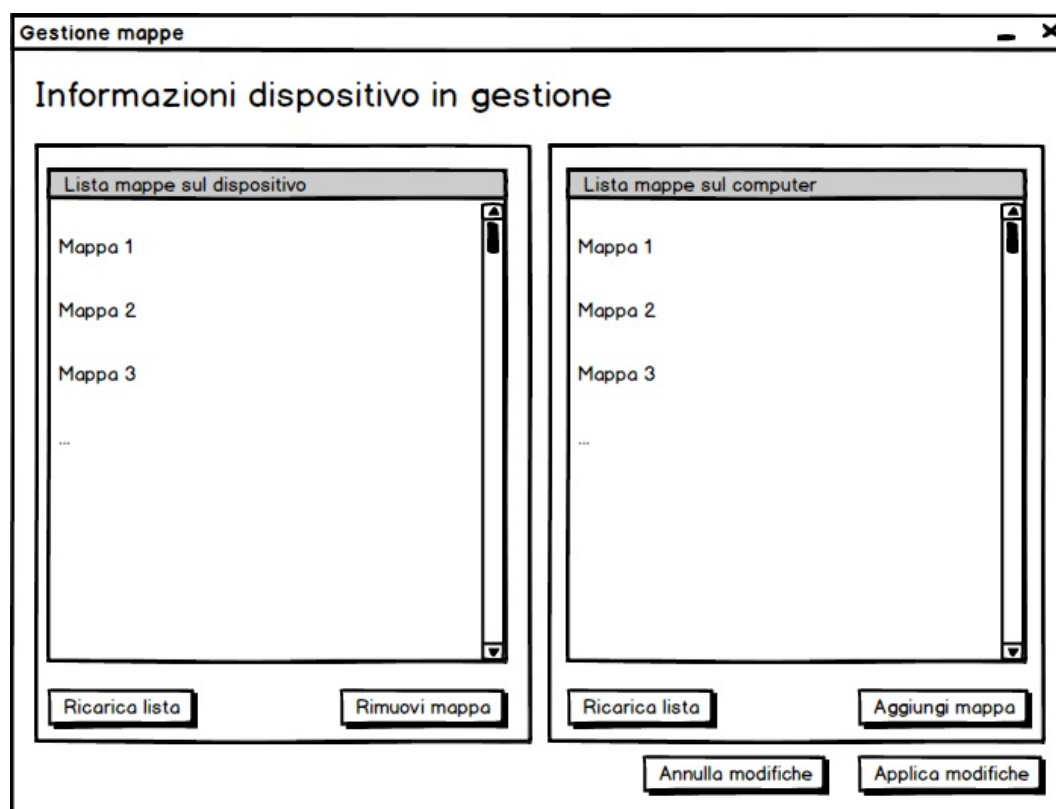


Figura 4.11: Mockup interfaccia grafica modulo Mappe

Il mockup superiore mette in evidenza la presenza di due liste, la prima a destra contenente le mappe presenti sul dispositivo e la seconda a sinistra contenente le mappe presenti sul computer in uso. Sono stati utilizzati due pannelli marcati con bordi evidenti per separare le due liste e gli associati bottoni collegati a specifiche funzionalità in modo da non confondere l'utente. Per agevolare l'utente, le due liste vengono aggiornate automaticamente visualizzando sempre lo stato successivo ad ogni operazione di inserimento o

rimozione. All'esterno dei due pannelli sono stati risposti due bottoni con i quali è possibile annullare o confermare le modifiche effettuate.

Mockup ProjectsView

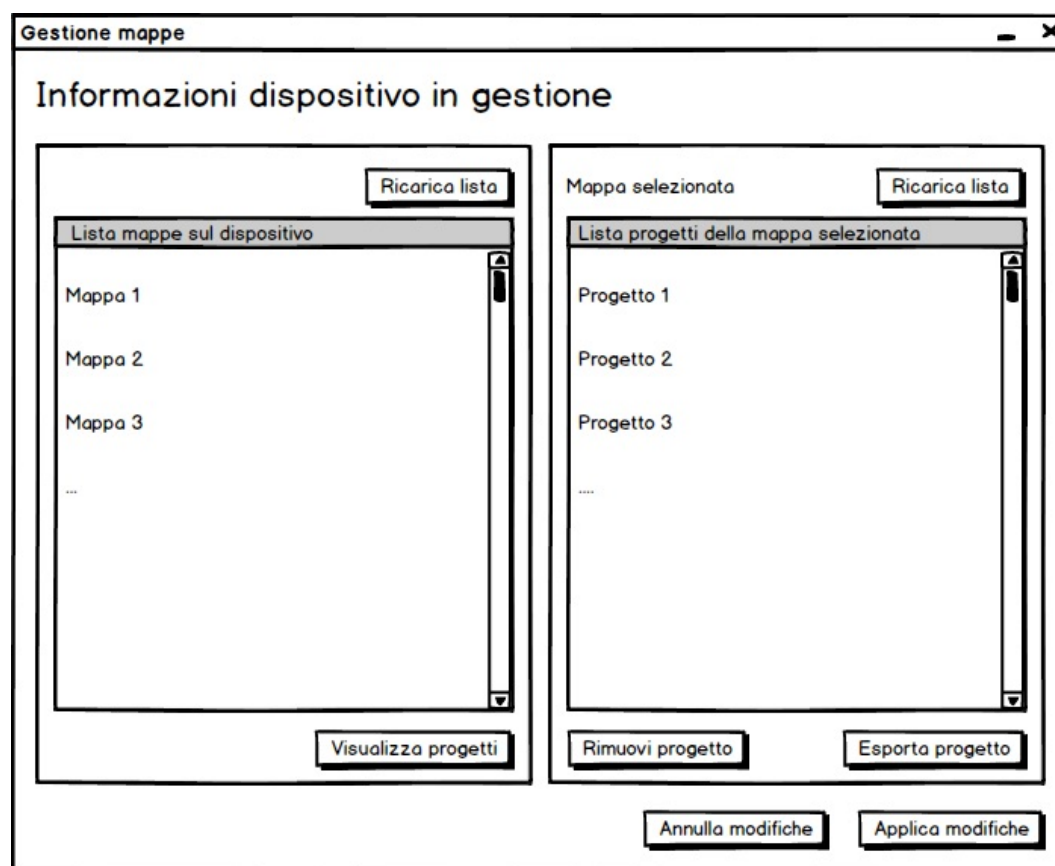


Figura 4.12: Mockup interfaccia grafica modulo Progetti

Il mockup in questione è composto da due pannelli delineati con bordi evidenti per separare la lista delle mappe presenti sul dispositivo alla lista dei progetti presenti nella mappa selezionata. Questa suddivisione avviene anche per le funzioni associate alle liste ad esempio i bottoni relativi all'eliminazione o esportazione dei progetti sono situati all'interno del pannello che contiene la lista dei progetti. Sono presenti due bottoni con i quali è possibile annullare o confermare le modifiche effettuate all'esterno dei due pannelli.

4.2 Applicativo mobile

4.2.1 Definizione modulo Progetti

L'analisi del modulo Progetti illustrato nel paragrafo 3.6.1 ha comportato la suddivisione delle principali funzionalità in tre settori:

- **Gestione lista progetti**
- **Gestione workspace di un progetto**
- **Gestione file struttura di un progetto**

Package Project_Manager

La definizione dei settori sopra illustrati ha generato le seguenti classi presenti nel package Project_Manager dell'applicativo mobile:

- **ProjectsXML**: contenente i metodi necessari alla lettura della lista dei progetti disponibili nella mappa attiva sull'applicativo ed il salvataggio della stessa lista in caso vengano creati nuovi progetti.
- **ProjectWorkspace**: classe contenente i metodi necessari alla creazione del workspace locale di un progetto così da consentire il salvataggio della struttura ed ulteriori file allegati al progetto in gestione.
- **ProjectStructureXML**: contenente i metodi necessari al caricamento della struttura di un progetto ricreando gli elementi presenti al suo interno e del salvataggio della stessa struttura con le modifiche apportate dall'utente.

Nella pagina seguente è presente il diagramma delle classi sopra elencate.

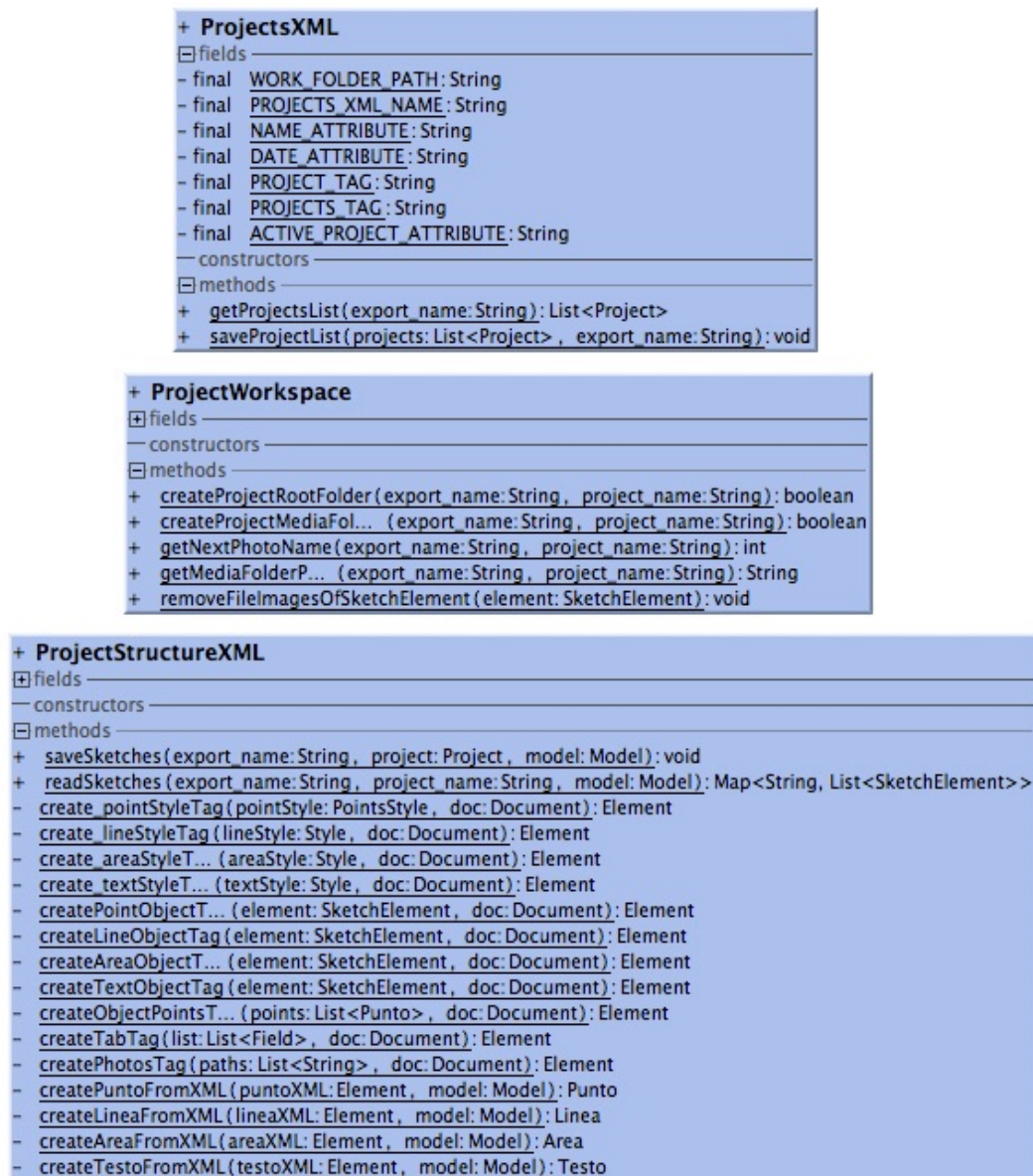


Figura 4.13: Diagramma classi del package Project_Manager

4.2.2 Struttura dei file XML

L'analisi dell'applicativo mobile MapYou come nell'applicativo desktop, ha rivelato la presenza di diversi file xml utilizzati dall'applicativo mobile per le seguenti finalità:

- **Config.xml**: dal quale vengono prelevati dati di configurazione per l'applicativo e le informazioni riguardanti le mappe presenti nella memoria del dispositivo.
- **Progetti.xml**: contiene la lista di progetti presenti nel dispositivo, specificando il nome e la data di ultima modifica.
- **Sketches_config.xml**: contiene la lista degli elementi disegnabili dall'utente definendo la loro struttura geometrica, il colore ed i valori attribuibili ad essi.
- **Sketches.xml**: rappresenta la struttura di un progetto contenendo la definizione degli elementi grafici utilizzati, la lista degli elementi disegnati specificando le coordinate ed i valori attribuiti dall'utente.

Il file sketches_config.xml non verrà illustrato perché utilizzato dal collega Filippo Nicolini per realizzare gli elementi grafici utilizzabili dall'utente e la struttura dei primi due file è stata descritta nel paragrafo 4.1.5.

Sketches.xml

Sketches.xml è il file utilizzato dall'applicativo per memorizzare la struttura di un progetto creato dall'utente e come spiegato in precedenza oltre a contenere la lista degli elementi presenti, definisce anche la loro struttura geometrica utilizzando la stessa struttura presente nel file Sketches_config.xml. La motivazione collegata a quest'ultima è la possibilità di rimozione di elementi grafici disegnabili, difatti l'utente durante l'utilizzo dell'applicativo potrebbe richiedere ad EBWolrd la creazione, la modifica o la rimozione di elementi di disegno. In caso di caricamento di un'elemento di disegno non

più presente nel file degli elementi grafici di default, la presenza della sua definizione nel file di struttura del progetto rende possibile ugualmente la sua visualizzazione ed utilizzo. Di seguito verrà definita la struttura del file Sketches.xml.

Struttura principale

```
1 <Sketches>
2   <Points>
3
4   </Points>
5   <Lines>
6
7   </Lines>
8   <Areas>
9
10  </Areas>
11  <Texts>
12
13  </Texts>
14 </Sketches>
```

L'immagine superiore rappresenta la struttura principale del file xml dove il tag "Sketches" conterrà l'intera definizione di un progetto. All'interno sono elencati quattro diversi tag che rappresentano le tipologie degli elementi disegnabili dall'utente definite in punti, linee, aree e testi.

Struttura interna generale elemento disegno

```
1 <Points>
2   <Styles>
3
4   </Styles>
5   <Objects>
6
7   </Objects>
8 </Points>
```

Ogni tipologia di elemento di disegno conterrà il tag `Styles` dove verranno inseriti gli stili degli elementi utilizzati nel progetto definendoli con la stessa struttura presente in `Sketches.config.xml`. Il tag `Objects` accoglierà ogni singola istanza di elemento inserito dall'utente.

Esempio di definizione dello stile di una linea

```
1 <Lines>
2   <Styles>
3     <style Name="Cavo" Pattern="" Color="0,0,255" Width="
4       3">
5       <tab>
6         <field Default="" Name="costruttore" Type="
7           text" ExternalName="Costruttore"/>
8         <field Default="" Name="fibre" Type="number"
9           ExternalName="Numero_fibre"/>
10        <field Default="" Name="note" Type="text"
11          ExternalName="Note"/>
12      </tab>
13    </style>
14  </Styles>
15  <Objects>
16    <Object>...</Object>
17    <Object>...</Object>
18  </Objects>
19 </Lines>
```

L'esempio in figura rappresenta la struttura utilizzata per memorizzare lo stile di un elemento di disegno di tipo linea dove sono presenti i tag `field` che corrispondono agli attributi modificabili dall'utente in ogni elemento dello stesso stile. In questo particolare caso l'utente può attribuire una stringa al campo costruttore, un valore numerico al campo identificato come fibre ed una stringa al campo note. Ad ogni elemento disegnabile dall'utente è collegato uno stile che obbligatoriamente contiene tag `field` simili a quelli in figura. Durante il salvataggio del progetto l'applicativo si occuperà di inserire univocamente gli stili utilizzati evitando la loro ripetizione anche in presenza di più elementi grafici con lo stesso stile. Al caricamento della struttura

del progetto selezionato dall'utente, l'applicativo provvederà all'estrazione degli stili presenti aggiungendoli a quelli già presenti nel caso risultassero mancanti.

Esempio di oggetto inserito nel progetto

```

1 <Lines>
2   <Styles>
3     <style Name="Cavo" ...>...</style>
4   </Styles>
5   <Objects>
6     <Object Style="Cavo">
7       <PointsItems>
8         <PointItem CoordX="1810501.0" CoordY="
9           4869303.0"/>
10        <PointItem CoordX="1810681.0" CoordY="
11          4869488.0"/>
12      </PointsItems>
13      <tab>
14        <field Name="costruttore" Default="telecom"
15          Type="text" ExternalName="Costruttore"/>
16        <field Name="fibre" Default="45" Type="number
17          " ExternalName="Numero_fibre"/>
18        <field Name="note" Default="Non_in_tensione"
19          Type="text" ExternalName="Note"/>
20      </tab>
21      <Photos>
22        <Photo Path="/storage/emulated/0/GisAndroid/
23          lavori/Pesaro_25000_vettoriale/progetti/
24          progetto3/media/0_20151119_122451.jpg"/>
25      </Photos>
26    </Object>
27  </Objects>
28 </Lines>

```

Come si può notare dall'immagine superiore ogni elemento inserito dall'utente corrisponderà ad un tag Object avente l'attributo Style che conterrà il nome dello stile dell'oggetto salvato.

All'interno del tag Object sono presenti i tag:

- **PointsItems**: utilizzato per definire la sezione dei punti che compongono l'elemento disegnato.
- **PointItem**: definisce le coordinate di ogni punto. Nel salvataggio di un elemento di tipo Punto le coordinate verranno direttamente inserite nel tag Object descritto in precedenza.
- **tab**: utilizzato per contenere gli attributi dell'elemento disegnato.
- **field**: definiscono gli attributi collegati all'oggetto.
- **Photos**: utilizzato per definire la sezione delle foto collegate all'elemento disegnato.
- **Photo**: contiene il percorso all'interno della memoria del dispositivo della foto collegata all'elemento disegnato.

Dalla lista si può notare il riutilizzo di tab e field corrispondenti agli stessi tag presenti nella definizione dello stile. L'unica differenza è l'attributo default che all'interno dei tag Styles conterrà i valori di default invece all'interno dei tag Objects conterrà i valori attribuiti dall'utente agli oggetti inseriti nel progetto. È stata effettuata questa scelta perché durante il caricamento di un progetto risulta più facile ricavare i valori specifici di ogni oggetto.

4.2.3 Workspace

Dato l'utilizzo da parte dell'applicativo mobile di differenti file è stata necessaria la progettazione di un workspace nella memoria interna del dispositivo. La creazione di tale workspace è eseguita come illustrato in precedenza dall'applicativo desktop durante la preparazione al primo utilizzo del dispositivo mobile collegato al computer.

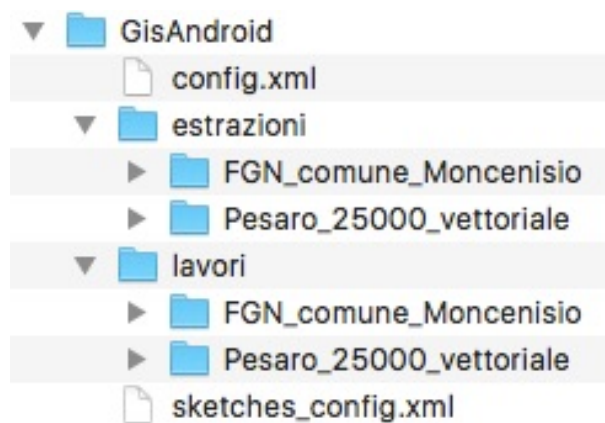


Figura 4.14: Struttura workspace dispositivo mobile

Come si può notare dall'immagine superiore è stata creata una cartella principale nominata `GisAndroid` contenente il file di configurazione `config.xml` utilizzato dall'applicativo per ricavare dati di configurazione e informazioni riguardanti le mappe disponibili in memoria presenti nella cartella `estrazioni` del workspace. La directory `lavori` conterrà i progetti creati dall'utente suddividendoli per mappa di appartenenza difatti al suo interno saranno presenti cartelle aventi lo stesso nome delle estrazioni disponibili.

Workspace progetti

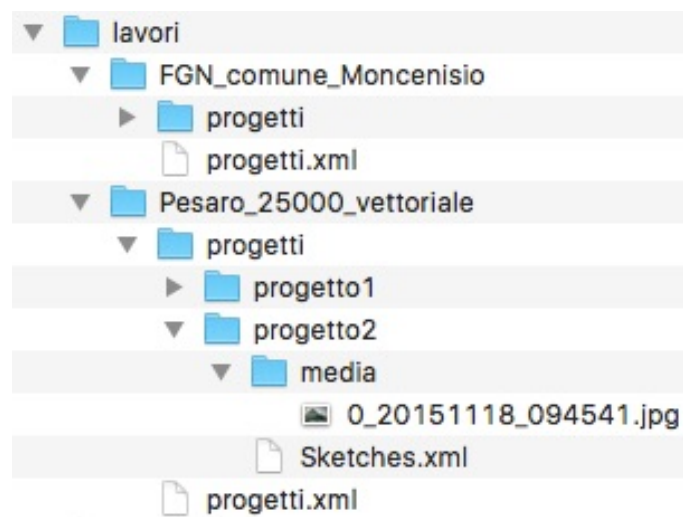


Figura 4.15: Struttura workspace progetti

L'esempio in figura rappresenta nello specifico il contenuto della directory "lavori" contenente i progetti creati dall'utente attraverso l'applicativo mobile. Come spiegato in precedenza è presente il file `progetti.xml` contenente la lista di progetti, per ogni cartella che rappresenta una mappa, all'interno di "lavori". All'interno della directory "progetti" vengono memorizzati i dati di ogni progetto creando inizialmente un cartella con il nome del progetto creato che conterrà il file `Sketches.xml` illustrato nel paragrafo 4.2.2 ed una cartella `media` dove salvare le foto collegate agli elementi del progetto.

Capitolo 5

Implementazione

In questo capitolo verranno descritte alcune implementazioni delle principali parti degli applicativi.

5.1 Rilevamento dispositivi mobile collegati

La maggior parte delle funzionalità dell'applicativo desktop riguardano la gestione della memoria di un dispositivo Android collegato al computer. Attraverso un semplice applicativo java in esecuzione risultava impossibile rilevare e quindi comunicare con essi. Come già spiegato nel paragrafo 4.1.3 con l'utilizzo combinato della libreria JMTP e i comandi ADB è stato possibile risolvere il problema descritto. Di seguito verrà illustrata nello specifico la soluzione utilizzata, evidenziando le differenze tra le due tecnologie.

5.1.1 Rilevamento attraverso JMTP

La libreria JMTP permette il rilevamento dei dispositivi multimediali collegati al computer attraverso una porta USB. Tra i dispositivi rilevabili sono compresi lettori MP3, videocamere, fotocamere, smartphone e qualsiasi altro dispositivo riconosciuto dal sistema operativo come multimediale.

Rilevamento con JMTP

```
1 public static List<PortableDevice> getConnectedDeviceJMTP() {  
2     PortableDeviceManager manager = new PortableDeviceManager();  
3     List<PortableDevice> list = new ArrayList<>();  
4     for(PortableDevice device : manager) {  
5         device.open();  
6         if (!device.getModel().contains("Apple") || !device.  
7             getModel().contains("Windows_Phone")) {  
8             list.add(device);  
9         }  
10        device.close();  
11    }  
12    return list;  
13 }
```

JMTP mette a disposizione la classe `PortableDeviceManager` capace di gestire tutti i dispositivi multimediali collegati ed attraverso una iterazione su di essa è possibile accedere ad ogni singola istanza di `PortableDevice`, classe che rappresenta un dispositivo. Prima di richiedere informazioni oppure accedere al contenuto di un dispositivo è importante aprire una connessione con esso attraverso il metodo `open()` e chiuderla utilizzando il metodo `close()`. Il codice in figura rappresenta il metodo utilizzato per rilevare attraverso JMTP solo i dispositivi multimediali Android collegati al computer.

5.1.2 Rilevazione attraverso ADB

Android Debug Bridge, illustrato nel paragrafo 2.2, ha differenza di JMTP è uno strumento specifico per la sola gestione di dispositivi Android. L'utilizzo è stato obbligato dalla mancanza di alcune funzionalità nella libreria JMTP come l'installazione di un applicativo Android su un dispositivo collegato al computer ed ha portato alla costruzione runtime dei comandi ADB specifici ad ogni dispositivo in gestione. Utilizzando comandi da terminale è stato necessario riuscire ad analizzare i risultati ottenuti contenenti dati utili all'applicazione desktop.

Rilevamento con ADB

```
1 private static final String ADB.DEVICES = "adb_devices";
2 private static final String ADB.S = "adb_s_";
3 private static final String ADB.RIL_SERIAL_NUMBER = "_shell_
  getprop_ril.serialnumber";
4
5 public static Map<String, String> getConnectedDevicesADB() {
6     Map<String, String> devices = new HashMap<>();
7     boolean checkStartList = false;
8     String line = "";
9     try {
10         Process proc = Runtime.getRuntime().exec(ADB.DEVICES);
11         BufferedReader reader = new BufferedReader(new
            InputStreamReader(proc.getInputStream()));
12         while((line = reader.readLine()) != null) {
13             if (checkStartList && line.length() > 0){
14                 int finish = line.indexOf(9);
15                 String ADBSerial = line.substring(0, finish);
16                 String RilSerial = getRilSerialADB(ADBSerial);
17                 if (RilSerial != null){
18                     devices.put(RilSerial, ADBSerial);
19                 }
20             }
21             if (line.startsWith("List")){
22                 checkStartList = true;
23             }
24         }
25     } catch (IOException e) {
26         e.printStackTrace();
27     }
28     return devices;
29 }
30
31 private static String getRilSerialADB(String ADBserial){
32     String adbRilSerial = ADB.S + ADBserial +
        ADB.RIL_SERIAL_NUMBER;
33     String line = "";
```

```
34     try {
35         Process proc = Runtime.getRuntime().exec("adbRilSerial");
36         BufferedReader reader = new BufferedReader(new
            InputStreamReader(proc.getInputStream()));
37         while((line = reader.readLine()) != null) {
38             return line;
39         }
40     } catch (IOException e) {
41         e.printStackTrace();
42     }
43     return null;
44 }
```

I metodi presenti nel frammento di codice in figura, vengono utilizzati per rilevare i dispositivi Android collegati al computer attraverso l'uso di comandi ADB che sono stati suddivisi tra parti statiche, memorizzate in variabili all'interno del progetto, e parti dinamiche passate ai metodi durante l'utilizzo dell'applicativo. È possibile vedere il filtraggio dei risultati ottenuti dai comandi ADB, difatti in entrambi i metodi sono presenti dei `BufferedReader` che attraverso il metodo `readLine()` gli estraggono. La finalità del secondo metodo verrà spiegato nella sezione successiva.

5.1.3 Utilizzo combinato di JMTP e ADB

JMTP e ADB presentano un importante limite al loro utilizzo congiunto che riguarda il collegamento dei dispositivi rilevati con il primo a quelli del secondo. Per l'applicativo desktop è obbligatorio riuscire ad effettuare questo collegamento visto che in sua mancanza, non riuscirebbe a soddisfare le principali funzionalità assegnategli. Le informazioni rese disponibili dalla classe `PortableDevice` come ad esempio il modello ed il codice seriale del dispositivo rappresentato hanno facilitato l'accoppiamento. Di seguito viene inserito il metodo che unisce la lista di `PortableDevice` ricavati tramite JMTP ai dispositivi rilevati attraverso ADB.

Abbinamento rilevamenti

```
1 public List<MobileDevice> mergeConnectedDevicesList() {  
2     List<MobileDevice> list = new ArrayList<>();  
3     List<PortableDevice> JMTPDevices = DeviceManager.  
        getConnectedDevice();  
4     Map<String,String> ADBDevices = DeviceManager.  
        getConnectedDevicesADB();  
5     for (PortableDevice device : JMTPDevices){  
6         if (ADBDevices.containsKey(device.getSerialNumber())){  
7             boolean ready = DeviceManager.checkDevice(device);  
8             list.add(new MobileDevice(device,ADBDevices.get(  
                device.getSerialNumber()),ready));  
9         }  
10    }  
11    return list;  
12 }
```

Dal codice in figura è possibile notare che l'abbinamento avviene tramite i numeri seriali dei dispositivi, però durante l'implementazione di tale metodo è stato riscontrato un diverso numero seriale ottenuto tramite JMTP da quello fornito con i comandi ADB, questo perché il valore del secondo rappresenta il numero seriale utilizzabile nei comandi ADB. La soluzione è stata raggiunta attraverso l'acquisizione del numero seriale di rilascio del dispositivo ADB prelevato in precedenza attraverso il metodo *getRilSerialADB(String ADBserial)*, corrispondente al valore ritornato dal metodo *getSerialNumber()* della classe *PortableDevice*. Il valore di ritorno del metodo in figura è una lista di *MobileDevice*, classe introdotta nel paragrafo 4.1.3, che rappresenta la lista dei dispositivi mobile collegati al computer.

5.2 Gestione file XML

Entrambi gli applicativi sviluppati implementano funzionalità per leggere, modificare e creare file XML. Esistono di numerose API per la gestione di questi file ma la scelta è ricaduta su DOM¹ perché permette la lettura, la scrittura e la manipolazione dell'intero file XML e preserva l'ordine degli elementi, molto utile durante la lettura dei punti che compongono aree e linee.

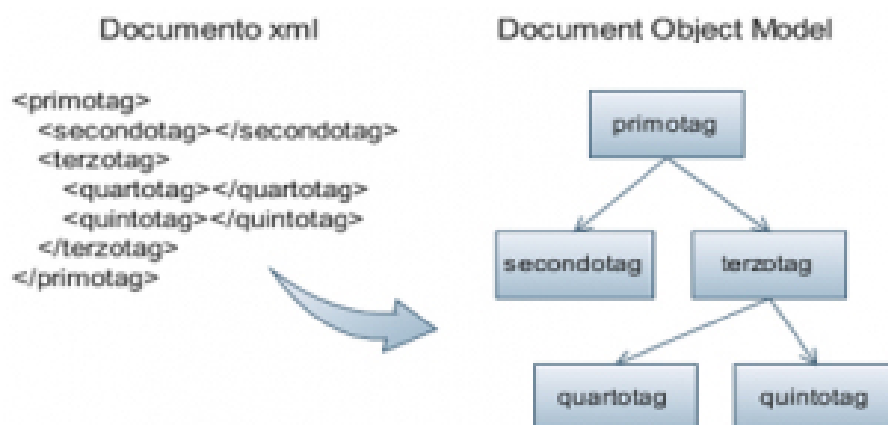


Figura 5.1: File xml con rappresentazione DOM

L'immagine raffigura un semplice documento xml e la sua rappresentazione DOM: come si può notare la rappresentazione ad albero prevede un nodo per ogni tag del documento e per ogni nodo tanti nodi-figlio quanti sono i tag nidificati all'interno dello stesso. Le classi coinvolte nell'elaborazione mediante DOM sono contenute nei seguenti package:

- **javax.xml.parsers**: contiene le classi `DocumentBuilder` e `DocumentBuilderFactory` che permettono di creare un parser DOM ed utilizzare quest'ultimo per ottenere una struttura ad albero rappresentativa del contenuto di un documento XML.

¹Document Object Model

- **org.w3c.dom**: definisce tutte le interfacce coinvolte nella rappresentazione di un documento XML.
- **javax.xml.transform**: per effettuare trasformazioni sul documento XML mediante XSLT².

Creazione e salvataggio progetti.xml

```
1 private static final String TAG.PROJECTS = "Projects";
2 private static final String TAG.PROJECT = "Project";
3 private static final String ATTRIBUTE_NAME = "Nome";
4 private static final String ATTRIBUTE_PROJECT_DATE = "
    DataProgetto";
5 private static final String ATTRIBUTE_CURRENT_PROJECT = "
    ProgettoAttivo";
6
7 public static boolean createProjectsXML (ProjectsXML
    projects_data, String path) {
8     try {
9         DocumentBuilderFactory docFactory =
            DocumentBuilderFactory.newInstance();
10        DocumentBuilder docBuilder = docFactory.
            newDocumentBuilder();
11        Document doc = docBuilder.newDocument();
12        Element rootElement = doc.createElement(TAG.PROJECTS);
13        doc.appendChild(rootElement);
14
15        if (projects_data.getCurrent_project() != null){
16            rootElement.setAttribute(ATTRIBUTE_CURRENT_PROJECT,
                projects_data.getCurrent_project());
17        } else {
18            rootElement.setAttribute(ATTRIBUTE_CURRENT_PROJECT,
                "");
19        }
20
21        for (ProjectTag proj: projects_data.getProjectsList()){
```

²EXtensible Stylesheet Language Trasformations che rendere possibile la trasformazione di un documento XML in un altro documento.

```
22         if (!proj.getDeleted()) {
23             Element project_elem = doc.createElement(
24                 TAG.PROJECT);
25             project_elem.setAttribute(ATTRIBUTE.NAME, proj.
26                 getName());
27             project_elem.setAttribute(ATTRIBUTE.PROJECT.DATE
28                 , proj.getDate());
29             rootElement.appendChild(project_elem);
30         }
31     }
32
33     TransformerFactory transformerFactory =
34         TransformerFactory.newInstance();
35     Transformer transformer = transformerFactory.
36         newTransformer();
37     DOMSource source = new DOMSource(doc);
38     StreamResult result = new StreamResult(new File(path));
39
40     transformer.transform(source, result);
41     return true;
42 } catch (ParserConfigurationException e) {
43     e.printStackTrace();
44 } catch (TransformerException e) {
45     e.printStackTrace();
46 }
47 return false;
48 }
```

Il metodo in figura presente in entrambi gli applicativi, permette la creazione del file progetti.xml ed è possibile suddividerlo in tre fasi. La prima istanzia un oggetto Document che rappresenta il file XML in creazione. La seconda realizza la struttura del XML e la riempie utilizzando i dati dei progetti contenuti dalla variabile *projects_data*. L'ultima fase attraverso il metodo *transform(Source xmlSource, Result outputTarget)* salva documento creato sul file creato nel percorso passato al metodo. Entrambi gli applicativi contengono metodi simili a quello in figura specifici per ogni file XML gestito.

5.3 Realizzazione interfacce grafiche

La presenza di mockup, realizzati durante la fase di progettazione, ha velocizzato la costruzione delle view dell'applicativo desktop composte dagli elementi grafici disponibili nella libreria Swing, appartenente alle Java Foundation Classes³. Data la possibilità di personalizzare il *look and feel*⁴ degli elementi grafici, grazie al controllo da parte di Swing sul rendering grafico dei propri componenti, è stata utilizzata la libreria JTattoo per applicare uno stile differente all'applicativo.

Modifica aspetto componenti Swing

```
1 private static final String ACRYLLOOKANDFELL = "com.jtattoo.  
   plaf.acryl.AcrylLookAndFeel";  
2  
3 private static void setGUI() {  
4     try {  
5         Properties props = new Properties();  
6         props.put("logoString", "EBWorld");  
7         AcrylLookAndFeel.setTheme(props);  
8         UIManager.setLookAndFeel(ACRYLLOOKANDFELL);  
9     } catch (ClassNotFoundException | InstantiationException |  
   IllegalAccessException | UnsupportedLookAndFeelException  
   exc) {  
10         Logger.getLogger("Unable_to_load_the_GUI");  
11     }  
12 }
```

Il metodo in figura permette la modifica dell'aspetto dei componenti grafici di Swing attraverso l'applicazione del tema *AcrylLookAndFeel* reso disponibile dalla libreria JTattoo. Dopo il suo utilizzo, il disegno e lo stile grafico dei componenti verranno emulati utilizzando valori presenti nel tema scelto. Di

³Caratteristiche percepite dall'utente di una interfaccia grafica, sia in termini di apparenza visiva che di modalità di interazione.

⁴Framework grafico per sviluppare interfacce grafiche multipiattaforma.

seguito verrà mostrata una view dell'applicativo successiva all'applicazione del tema utilizzato nel frammento di codice superiore.

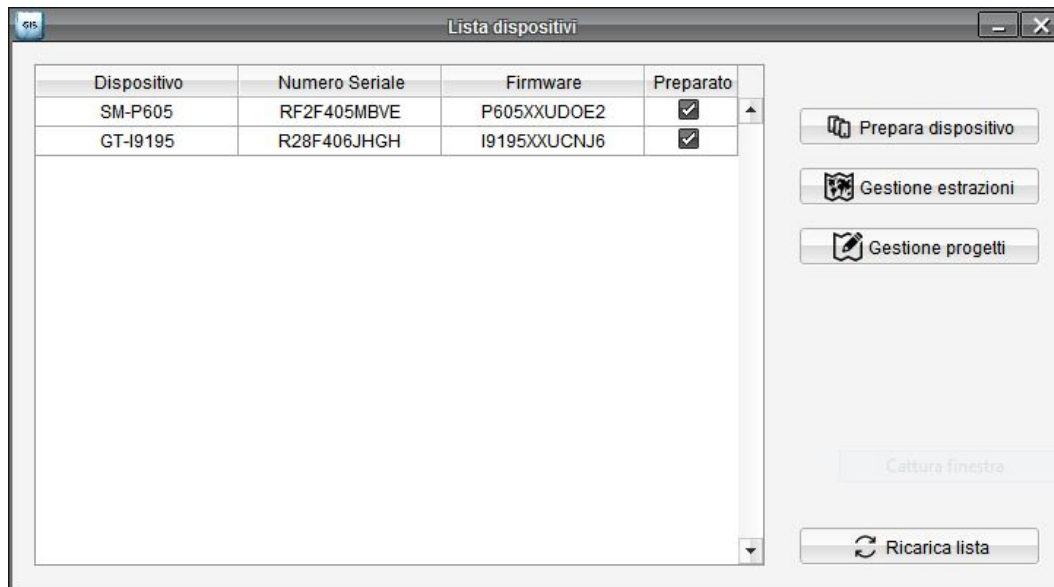


Figura 5.2: Risultato interfaccia grafica modulo Dispositivi

Conclusioni

È stato veramente interessante ed istruttivo analizzare, progettare e implementare questi applicativi soprattutto durante la strutturazione dell'applicativo desktop. I problemi che si sono dovuti affrontare sin dal principio creati dai limiti delle mie conoscenze hanno permesso di ampliare il mio bagaglio culturale per trovare soluzioni adeguate.

La possibilità di lavorare in azienda è stata stimolante in quanto ha permesso il confronto con altri colleghi, la risoluzione di problematiche più o meno complesse, lo sviluppo di abilità e capacità non apprendibili con corsi universitari e di avere un impatto con il mondo del lavoro e delle dinamiche di un'azienda.

Il team composto con il collega Filippo Nicolini è stato immediatamente produttivo attraverso un continuo scambio di opinioni, idee e domande rendendo possibile uno sviluppo senza interruzioni.

I sistemi creati hanno soddisfatto le aspettative ed hanno posto le basi per eventuali sviluppi futuri descritti nella sezione seguente.

Sviluppi futuri

I due applicativi sviluppati sono una prima versione di base, che comunque riescono a soddisfare tutte le operazioni fondamentali e sono stati creati con l'intenzione di aggiungere in futuro nuove funzionalità accrescendo pro-

gressivamente la loro utilità per l'utente finale. Per esempio l'applicativo desktop potrebbe fornire la possibilità di inserire un progetto locale nel dispositivo mobile collegato e l'applicativo mobile potrebbe rendere disponibile l'eliminazione di una o più mappe oppure uno o più progetti per liberare la memoria occupata.

Oltre a queste più semplici modifiche è necessario distinguere due prospettive maggiormente incisive in un futuro sviluppo, ovvero il collegamento dell'applicazione mobile ad Internet e la realizzazione di una parte di realtà aumentata.

Per quanto riguarda il primo caso, il funzionamento dell'applicazione potrebbe essere disponibile in presenza e in assenza di connettività internet. Numerose aziende supportano queste due modalità di utilizzo nelle loro applicazioni rendendo disponibili servizi e funzionalità in precedenza non possibili. In riferimento all'applicazione sviluppata, attraverso l'utilizzo di file scaricati e di una cache contenente parti scaricate in precedenza sarebbe possibile la visualizzazione di mappe archiviate in rete escludendo il problema della datazione dei dati archiviati in locale che potrebbero risultare non aggiornati. Nel caso l'applicazione risultasse mancante di un collegamento stabile si potrebbero utilizzare i dati locali disponibili sul dispositivo completando la mappa visualizzata. Un'ulteriore funzionalità sviluppabile attraverso questa prospettiva futura è la creazione di progetti e la loro sincronizzazione in rete rendendoli velocemente disponibili alla sede centrale ed altri tecnici esterni e consentendo una migliore interazione tra loro.

Per quanto riguarda la realizzazione di una realtà aumentata, questa prospettiva potrebbe risultare estremamente interessante. Infatti questo nuovo ambito si sta progressivamente diffondendo arricchendo la percezione umana della realtà mediante informazioni generalmente manipolate e convogliate elettronicamente. Per esempio si potrebbe pensare di fornire all'utente la

possibilità di visualizzare i dati georiferiti, contenuti nel dispositivo, attraverso l'utilizzo della fotocamera in combinazione con le informazioni rese disponibili dal giroscopio e la posizione geografica del dispositivo. Il risultato renderebbe possibile ad esempio la visualizzazione di elementi presenti nel sottosuolo oppure all'interno di mura ed attraverso il movimento del dispositivo e lo spostamento della posizione sarebbe possibile ricreare in 3D la precedente visualizzazione in 2D della mappa.

I possibili sviluppi futuri sono molteplici e possono aprire la strada a scenari molto interessanti ed estremamente innovativi.

Ringraziamenti

Ringrazio innanzitutto i miei genitori, per essermi stato vicino durante l'ultimo intenso periodo di questo percorso di studi e senza i quali non sarei mai arrivato fino a questo punto.

Ci tengo a ringraziare i compagni di corso Alessandro Bagnoli e Filippo Nicolini, con i quali ho affrontato numerosi esami sia lavorando in team per progetti che sopportandoci a vicenda durante la preparazione alle prove più impegnative.

Un grazie in particolare va all'azienda EBWorld per avermi permesso di sviluppare assieme a loro questo elaborato di tesi contribuendo alla crescita delle mie capacità spronandomi a dare sempre il meglio.

Elenco delle figure

1.1	Componenti di un GIS	2
1.2	Schermata principale MapYou	5
1.3	Esempio schermata Gestore MapYou	7
2.1	Sintassi generica comando ADB	11
2.2	Esempio Swing, standard GUI Java	12
2.3	Tema predefinito TextureLookAndFeel	13
3.1	Diagramma dei casi d'uso del modulo dispositivi	24
3.2	Diagramma dei casi d'uso del modulo mappe	25
3.3	Diagramma dei casi d'uso del modulo progetti	26
3.4	Diagramma dei casi d'uso del modulo progetti dell'applicativo mobile	29
4.1	Interazione tra componenti del MVC	32
4.2	MVC applicato ai moduli	33
4.3	Diagramma classi del modulo Dispositivi	35
4.4	Flusso utilizzo dei moduli	36
4.5	Diagramma della classe MobileDevice	38
4.6	Diagramma delle classi ConfigXML e ExportTag	43
4.7	Diagramma delle classi ProjectsXML e ProjectTag	45
4.8	Struttura workspace locale	46
4.9	Struttura sorgente dati	47
4.10	Mockup interfaccia grafica modulo Dispositivi	48

4.11	Mockup interfaccia grafica modulo Mappe	49
4.12	Mockup interfaccia grafica modulo Progetti	50
4.13	Diagramma classi del package Project_Manager	52
4.14	Struttura workspace dispositivo mobile	58
4.15	Struttura workspace progetti	59
5.1	File xml con rappresentazione DOM	66
5.2	Risultato interfaccia grafica modulo Dispositivi	70

Bibliografia

- [1] <http://www.ebw.it>
- [2] <https://docs.oracle.com/javase/8/docs/api/>
- [3] <http://developer.android.com/tools/help/adb.html>
- [4] <https://code.google.com/p/jmtp/>
- [5] <http://www.jtattoo.net>
- [6] <https://www.wikipedia.org>
- [7] <https://bitbucket.org>