Università degli studi di Trento
Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea (Triennale) in informatica

elaborato finale

Guida turistica per dispositivi mobili: Progettazione e sviluppo dell'applicativo con tecnologia J2ME

Relatore
Prof. Paolo Giorgini

Candidato
Erik Bertoletti

Anno Accademico 2007/2008

Indice

Introduzione

Negli ultimi anni si è assistito ad un incredibile sviluppo tecnologico, in particolar modo nel mondo delle telecomunicazioni e dei dispositivi mobili come i cellulari, smartphone e PDA; basti pensare che uno dei primi cellulari, nato negli anni '40, occupava il bagagliaio di un'automobile e le uniche funzioni disponibili erano comporre il numero e parlare. Ora alcuni modelli stanno nel palmo della mano e hanno capacità di calcolo paragonabili a quelle di un pc di qualche anno fa.
Oltre a questo non bisogna dimenticare il gran numero di periferiche di cui sono muniti oggi la maggior parte dei dispositivi, sempre più preposti alla connettività mobile.
In questo settore una delle tecnologie che ha subito una notevole evoluzione è quella GPS; nata come strumento militare, si può trovare al giorno d'oggi racchiusa in piccoli dispositivi elettronici, detti ricevitori. I "vecchi" navigatori satellitari come Tom Tom, Nuvi, ecc. sono stati pian piano affiancati anche da questi dispositivi mobili.
Le previsioni degli esperti del settore dicono che entro la fine del 2008, ad essere dotati di GPS, saranno circa il 25% dei cellulari WCDMA (rete di terza generazione). Le ragioni di questa evoluzione sono sostanzialmente 4:

- La prima è riconducibile agli inviti che le autorità preposte mandano ormai da tempo a operatori produttori sulla necessità di mettere a punto soluzioni per le emergenze. Tra queste, la possibilità di localizzare, grazie alla tecnologia GPS, i terminali dai quali provengono chiamate d'emergenza.

- La seconda ragione è che gli operatori CMDA stanno intensificando le attività di laboratorio per integrare la tecnologia nei nuovi terminali.

- Il terzo motivo è di ordine funzionale ed economico: dotati di GPS, i nuovi terminali permetterebbero all'utente l'utilizzo di tutta una serie di servizi proposti dai vari operatori di telefonia mobile che farebbero incrementare i loro introiti (come succede adesso con gli SMS e gli MMS e con la tecnologia GPRS).

- E infine, la richiesta di servizi di localizzazione satellitare da parte dell' utenza.

Naturalmente esistono in commercio software in grado di sfruttare questa tecnologia: alcuni sono prodotti commerciali, quindi a pagamento, mentre se ne possono trovare molti opensource, sviluppati nella maggior parte dei casi con tecnologia j2me (quindi compatibili con la maggior parte dei cellulari ormai in commercio), piuttosto che per sistemi operativi symbian e Windows Mobile (scritti generalmente in linguaggio C o Python). Si differenziano l'uno dall'altro soprattutto per il principio di funzionamento e i prerequisiti hardware richiesti. Ad esempio, alcuni usano le connessioni GPRS per connettersi alla rete internet e scaricare le informazioni; altri semplicemente sono compatibili solo con certi tipi di cellulari.

Scopo di questa tesi è elaborare un sistema che riesca a colmare le lacune lasciate da queste applicazioni, nello specifico sotto il punto di vista economico e di portabilità.
Quindi quello che ci si aspetta da questa applicazione è che l'utente riesca ad usarla in maniera efficace, riuscendo ad ottenere tutte le informazioni turistiche sul luogo di interesse in maniera veloce, ma soprattutto economica.
La particolarità di questo sistema consiste nell'usare direttamente le coordinate geografiche del punto in cui l'utente si trova e, in base a queste e a preferenze impostate dall'utente stesso sul software, fornire informazioni turistiche sulla zona; il tutto indipendentemente dall'uso di connessioni GPRS e simili, quindi con un deciso abbattimento dei costi e quasi in tempo reale.
Il linguaggio utilizzato per sviluppare l'applicazione è java, più precisamente la versione micro edition di java sviluppata dalla Sun Microsystems appositamente per dispositivi mobili, che garantisce un'ampia compatibilità indipendentemente dal modello o dalla marca del dispositivo, in quanto la java virtual machine è installata di default in quasi tutti i cellulari e dispositivi mobili (in seguito verrà discusso il perchè questo non sia completamente corretto).
L' idea di base è che un utente, quale può essere la stessa agenzia del turismo, possa creare dei database di informazioni della propria città, formattati in una determinata maniera (che verrà esposta in seguito) con l'uso di un software dedicato su piattaforma PC (discusso in un'altra tesi), da poter inserire nella memoria del telefono, nel caso specifico di quest'applicazione nella memory card. L'idea della memory card come supporto principale è dovuta ad una questione pratica come l'uso commerciale che può farne l'agenzia vendendola o noleggiandola all'utenza. Una sorta di libro in formato elettronico.

Questa tesi si propone di esporre nel Capitolo 1 qual'è lo stato attuale delle guide turistiche per dispositivi mobili e le tecnologie usate per lo sviluppo di questa applicazione. Nel capitolo 2 verrà esposto il problema di partenza, e la possibile soluzione analizzando i requisiti che essa dovrà avere e la sua implementazione. Nel capitolo 3 saranno invece esposti i problemi sorti durante il testing dell'applicazione con uno schema che visualizza la velocità di funzionamento in differenti occasioni di utilizzo; verranno inoltre esposti i possibili sviluppi futuri.

Capitolo 1

Panorama e tecnologie di sviluppo

In questo capitolo si parlerà di quello che è lo stato dell'arte attuale in fatto di guide turistiche per dispositivi mobili e delle principali tecnologie usate per lo sviluppo dell' applicazione; verranno descritte di fatto la tecnologia Bluetooth, il funzionamento dei dispositivi GPS e il loro protocollo di trasmissione e gli strumenti informatici usati per l'implementazione, quali J2ME e XML.

1.1 Panorama della guida turistica per dispositivi mobili

Lo scenario attuale in fatto di applicazioni di tipo turistico per dispositivi mobili si presenta ricco di prodotti, commerciali o meno, dotati di svariate caratteristiche.
Per quanto riguarda la piattaforma Personal Computer, i software più diffusi in questo ambito sono Google maps, google earth, Microsoft Autoroute (esistono svariate altri software simili, ma questi possono essere considerati i più completi): sono applicazioni molto sofisticate, possiedono funzionalità avanzate che permettono di tracciare un itinerario da un qualsiasi punto del mondo ad un altro, piuttosto che visualizzare la via più corta da percorrere, o più semplicemente visualizzare la mappa della zona che ci interessa.

Risulta chiaro che per un turista che vuole visitare una città, i suoi musei, le sue piazze, un software di questo tipo è praticamente inutile, non solo per l'inutilità delle funzioni che presenta, ma all'atto pratico anche per l'ingombro che esso rappresenta, dato che per funzionare necessita quanto meno

di un computer portatile.
Con l'avvento dei cellulari le cui caratteristiche hardware e software sono cresciute esponenzialmente, specie negli ultimi anni, sempre più aziende e software house, nonchè liberi programmatori, hanno pensato al porting su questi dispositivi di questo tipo di applicazione. A tutt'oggi se ne possono reperire un numero indefinito, e si differenziano tra di loro per requisiti hardware e funzionalità offerte.
La diffusione sempre più capillare di nuove tecnologie di trasmissione dati, come ad esempio il GPRS ha fatto si che questi software evolvessero in termini di prestazioni e funzionalità, a scapito dell' economicità dell'applicazione. Un servizio GPRS (General Packet Radio Service) permette la trasmissione di dati attraverso la rete GSM, con una velocità media tra i 4 e i 5 kB/s; questo servizio ha un costo, che dipende dall'operatore mobile con il quale si ha stipulato il contratto. Alcuni gestori telefonici hanno adottato per il GPRS una politica di tariffe basse rispetto ai precedenti sistemi di trasferimento dati attraverso reti GSM, noti con le sigle CSD (Circuit Switched Data) e HSCSD (High Speed Circuit Switched Data).
Per l'accesso a internet, molti di questi gestori non offrono tariffe di tipo flat (cioè indipendenti dal tempo di connessione), con la significativa eccezione della T-Mobile negli USA, in cui la tariffazione è basata sul volume di dati scambiato (solitamente arrotondata ai 100 kbyte). Le tariffe applicate dai vari gestori variano enormemente (da 1 a 20 euro per megabyte). Negli USA la T-Mobile offre un abbonamento a 30 $ al mese senza limitazioni di tempo. Anche la AT&T Wireless offre abbonamenti di tipo flat.

Solitamente utilizzando una tipologia di cellulare con tecnologia
GPRS/EDGE/UMTS non collegato ad altro terminale (PC, portatile o palmare) sono accessibili solo siti WAP ovvero siti appositamente progettati per i cellulari: si tratta di siti semplificati, non contenenti applet java, pagine dinamiche o applicazioni in HTML4.
Connettendo il cellulare ad un altro terminale, tramite Bluetooth o cavo USB 2.0 (che non rallenta la connessione nel tratto finale dal momento che supporta velocità di trasmissione fino a 480 Mbit/s), si stabilisce una normale connessione in cui sono disponibili i medesimi servizi e protocolli accessibili con il modem analogico o via ADSL. Il cellulare funziona da modem, e tutti i bit vengono inviati al terminale per la decodifica e la visualizzazione.

La maggior parte delle applicazioni di guida turistica, fanno largo uso di questo servizio, o per lo scaricamento dello stesso programma direttamente dal sito (magari quando il cellulare in questione non ha la possibilità di connettersi al pc), o per il download delle informazioni turistiche, piuttosto che

delle mappe, ecc.
Con l'avvento dei ricevitori GPS, si sono aggiunte applicazioni con nuove funzionalità, che sfruttano i dati ottenuti dal satellite per la localizzazione del punto in cui l'utente si trova.

Una delle ultime novità in fatto di localizzazione (nel linguaggio J2ME) è l'uso delle *Location API*; Grazie ad esse la posizione di un dispositivo mobile può essere determinata nei seguenti modi:

- Usando la rete del telefono mobile: l'ID del telefono mobile può essere utilizzato per determinare la stazione base di trasmissione (BTS) con il quale il dispositivo comunica ed identificare quindi la posizione della stazione BTS. Con questa tecnica la posizione viene determinata in maniera molto grossolana: un cellulare GSM per intenderci potrebbe trovarsi entro il raggio di 10 Km dalla stazione BTS con la quale comunica.

- Sfruttando il sistema GPS (Global Positioning System) costituito da una rete di 24 satelliti che calcola i ritardi di propagazione del segnale fra un satellite e l'altro per determinare la posizione geografica del dispositivo. Tale sistema è costoso in quanto il dispositivo mobile deve essere dotato di un ricevitore GPS ma garantisce una buona precisione (fra 4 e 40 metri).

- Sfruttando una rete locale: ad esempio all'interno di un edificio si potrebbe installare una rete di dispositivi Bluetooth e calcolare tramite triangolazioni la posizione del dispositivo mobile.

  Naturalmente la precisione è un fattore molto importante per alcuni tipi di applicazioni e meno per altre: un'applicazione che ha lo scopo di guidare un turista all'interno di una città potrebbe ad esempio accontentarsi di un margine di errore di qualche decina di metri ma lo stesso margine di errore potrebbe risultare troppo grande per applicazioni di tipo scientifico.

Come verrà descritto in seguito, purtroppo non tutti i cellulari dispongono di questa libreria; ciò pone naturalmente un vincolo in fatto di portabilità del codice.

Si possono elencare alcuni esempi di questo tipo di applicazioni, caratterizzati tutti da specifiche diverse:

CELLMAP Si tratta di una guida turistica per cellulari che mette a disposizione dell' utente tutte le informazioni utili sulla visita e sulla permanenza in una determinata località turistica in maniere rapida ed economica.
l' utente si collega con il proprio cellulare ad un sito internet specifico e scarica sul telefonino la guida turistica con tutte le informazioni sulla città e/o località turistica.
Dopo averla scaricata ed installata l' utente può accedere con facilità a tutta una serie di informazioni turistiche come ad esempio:

- Elenco di Hotel, B&B, Camping, ecc
- Elenco di ristoranti
- Tempo libero, come Discoteche, concerti, pubs, Sport e Cinema
- Offerte Last Minute di ogni tipo
- News
- Manifestazioni
- Informazioni sullo shopping e altro

Questo è un software che viene anche venduto da alcune agenzie per il Turismo di alcune località, come ad esempio Livigno; nello specifico vengono venduti i database con le informazioni turistiche.
L'utente può ottenere la guida scaricandola da un sito tramite l'immissione di una password ottenuta da una carta tipo ´gratta e vinci´, venduta dallo steso sviluppatore, ottenendo così un certo introito; poi periodicamente può collegarsi ad un sito specifico e scaricare gli aggiornamenti direttamente sul cellulare; risulta chiaro che in questo caso si deve essere in possesso di una connessione dati per fare ciò, con i costi che essa comporta.
Secondo lo sviluppatore l'unico requisito tecnico è che il cellulare sia abilitato WAP, cioè appunto in grado di connettersi alla rete Internet senza l'ausilio di un PC. In seguito, a parte nel caso degli aggiornamenti, non necessita di altre connessioni.

TRAVELS GUIDE Travels Guide è una serie di software, sviluppati in FlashLite per cellulari flashlite enabled , completi di mappa della città, mappa della metropolitana, percorsi dei bus e di molte altre informazioni essenziali quali polizia, pronto soccorso e principali collegamenti come aeroporti stazione ferroviaria, ecc.
Come promozione è disponibile gratuitamente la Travels Guide Vienna, con le seguenti caratteristiche (comuni a tutte le altre località):

Fig. 1.1: CellMap

Mappa della Città comprensiva del centro storico e dei principali dintorni, per un'area di circa 5000x4000 mt.

Mappa della Metropolitana E' possibile tramite apposita funzione visualizzare/nascondere la mappa della città e quella della metropolitana a piacimento.

Archivio di oltre 500 vie per trovare immediatamente l'indirizzo al quale si è interessati.

Più di 70 luoghi turistici da visitare completi di mappe, descrizione della zona, ecc.

Oltre 300 punti di interesse tra cui alberghi, discoteche, ristoranti, ecc.

l'applicazione richiede però delle specifiche non comuni a tutti i cellulari, quali un sistema operativo Symbian e un Plugin Macromedia FlashLite 2.0 per cellulari; quindi prerequisiti non di poco conto, visto che allo stato attuale sono si sempre più numerosi i cellulari e smartphone dotati di queste caratteristiche, però non sono ancora così diffusi in relazione al totale dei dispositivi in uso al momento dall'utenza me-

dia.

Possiede comunque la particolarità che non necessita di connessioni alla rete internet per il suo download e installazione, in quanto si può trasferire il pacchetto di installazione direttamente dal pc al telefono tramite cavo o connessione bluetooth (oppure scaricarlo direttamente sul cellulare tramite wi-fi, se dotato di questo accessorio); la consultazione viene poi fatta con la stessa modalità di una guida turistica cartacea, facilitata comunque dalle varie opzioni di ricerca per punti di interesse che automatizzano e velocizzano di conseguenza tutto il processo.

ITALIA MOBILE Guida Italia Mobile è uno strumento informativo che consente la ricerca di Hotels, Ristoranti, Agriturismi, ecc, su tutto il territorio nazionale che si installa su telefoni cellulari, smartphone e palmari.
La Guida Mobile seleziona le strutture turistiche secondo criteri di ricerca personalizzati (Regione, Località, Tipologia).
Essendo scritto in linguaggio java, richiede che il cellulare sia equipaggiato con una virtual machine e ne esistono varie versioni per soddisfare le diverse esigenze di memoria dei dispositivi.

La particolarità dell'applicazione è che non usa nessun strumento esterno di collegamento, come GPS o Bluetooth, però contiene tutte le informazioni in un unico file; in questo modo se l'utente si trova in una zona e vuole sapere cosa c'è nei dintorni, deve ricercare a mano le informazioni di suo interesse.

Questo, grafica e funzioni varie a parte, ha un principio di funzionamento molto simile al precedente, con la particolarità che essendo scritto in linguaggio java (j2ME nello specifico), è compatibile con una fetta più grossa di dispositivi. Google Maps Mobile 2.0 Beta promette di offrire la localizzazione del dispositivo anche in situazioni in cui le funzionalità GPS non siano disponibili, ovvero al chiuso o su smartphone non provvisti di tale periferica.

GOOGLE MAPS MOBILE La versione mobile della nota applicazione per pc di google, Google Maps: essa è entrata a far parte degli strumenti comunemente utilizzati dagli utenti per individuare località, indirizzi o informazioni simili: utilizzando un pc provvisto di connessione a banda larga si ha a propria disposizione un supporto cartografico

notevole. Alla semplice cartografia Google ha aggiunto nel tempo da disponibilità di foto aeree e di vari altri strati tematici.
Da tempo si parla di una versione del servizio Google Maps dedicata ai dispositivi mobile e una prima versione di tale software è in circolazione da parecchi mesi. Google ha recentemente annunciato la disponibilità di una versione aggiornata di tale strumento la cui novità principale è la funzione My Location ideata per mostrare sulle mappe di Google la posizione del terminale mobile.
Nel caso in cui lo smartphone sia dotato di ricevitore GPS Google Maps Mobile proietta le informazioni relative alla posizione sulla mappa accessibile attraverso i servizi online. In questa situazione la localizzazione è assai precisa, per contro il consumo energetico dell'unità GPS non è trascurabile e disponibilità del servizio è limitata solo agli spazi aperti.

Premendo il tasto 0 sullo smartphone con Google Maps Mobile 2.0 in esecuzione, lo smartphone effettua un'interrogazione al servizio online di Google. In questa fase lo smartphone trasmette anche l'identificativo della cella telefonica attraverso cui è connesso alla rete Internet. I server di Google gestiscono un apposito database al cui interno sono presenti le informazioni relative alla localizzazione delle singole celle e ritrasmettono allo smartphone tali dati.

Google Maps Mobile 2.0 non effettua alcuna triangolazione di segnale e non valuta la potenza del segnale radio ricevuto per determinare la distanza: tutto avviene interrogando un database remoto.
Per poter usufruire di questa applicazione è necessario possedere uno smartphone di ultima generazione: tutti i più recenti sistemi Nokia, alcuni BlackBerry e taluni dispositivi Sony Ericsson e Motorola possono essere utilizzati per utilizzare Google Maps Mobile 2.0.
Una lista dettagliata è disponibile a questo indirizzo. Google Mobile Maps 2 può essere installato anche su sistemi Microsoft Windows Mobile e Palm OS mentre sui dispositivi Apple iPhone non è disponibile la funzione My location.

Queste sono solo alcune delle applicazioni di questo tipo; da qui si può capire come sia un mondo abbastanza vario, quello composto dalle applicazioni mobili, in particolare queste ad uso turistico.
Di fatto è difficile fare un punto della situazione generale, la tipologia da utilizzare dipende molto dalle necessità e dal prodotto che un utente possiede. Infatti pur avendo molte caratteristiche simili alcune applicazioni possiedono

Fig. 1.2: Una schermata di Google Mobile 2.0

requisiti diversi, e molte invece pur avendo requisiti uguali si differenziano nella modalità d'uso e nei servizi che possono offrire.

1.2 Java 2 Micro Edition

La rivoluzione introdotta dall'avvento del linguaggio Java con il suo slogan Write Once, Run Everywhere (Scrivi una volta, esegui dovunque), si era limitata fino a qualche anno fa al mondo dei PC (grazie a J2SE) e a quello delle applicazioni server (con J2EE).
L'esplosione del mercato dei terminali mobili, PDA e telefoni cellulari, ha fatto sì che Sun si spingesse anche su queste piattaforme con la tecnologia denominata J2ME ovvero Java Micro Edition. Configuration e Profile.
La sfida tecnologica è stata quella di definire un set di API, da mettere a disposizione degli sviluppatori, che soddisfacesse le minime disponibilità di oggetti che:

- hanno limitate capacità di elaborazione;
- hanno limitate capacità di memoria;
- hanno schermi di ridotte dimensioni (a volte non a colori) e con poca risoluzione.

L'offerta del mercato è inoltre variegata e presenta terminali molto diversi in termini di funzionalità e caratteristiche tecniche (si pensi, ad esempio, alle differenza di potenza di calcolo e di display tra un telefono cellulare e un palmare).
Per supportare una così vasta gamma di terminali, Sun ha introdotto il concetto di Configuration che può identificarsi con la definizione delle librerie base (core) e delle caratteristiche di una Java Virtual Machine (JVM) che risiede in un sistema con determinate potenzialità. In particolare sono definite due diverse Configuration:

1. Connected Device Configuration (CDC) che nella versione attuale (la 1.1) è presente in terminali con le seguenti caratteristiche minime:

   - 512 kbyte di memoria RAM disponibile
   - 256 kbyte di memoria per il runtime delle applicazioni Java
   - Connettività in rete possibilmente continua ed a larga banda

2. Connected Limited Device Configuration (CLDC) che nella versione attuale (la 1.1) è presente in terminali con le seguenti caratteristiche minime:

   - 192 kb di memoria per eseguire applicazioni Java
   - Un processore a 16 o 32 bit
   - Basso consumo di energia (normalmente derivante da batterie)
   - Connettività in rete (normalmente wireless) con banda limitata e accesso a intermittenza.

L'applicazione sviluppata è basata sulla configurazione CLDC, la cui particolare JVM è stata denominata KVM (K Virtual Machine). È stato poi necessario rendere ancora più flessibile questa piattaforma introducendo il concetto di Profile che rappresenta l'insieme di librerie che consentono la gestione delle interfacce utente, la cattura e la gestione degli eventi, la memorizzazione persistente, le connessioni e la gestione dei timer su device di una determinata tipologia. Il Profile che considereremo è il Mobile Information Device Profile (MIDP) giunto alla versione 2.1.

Possiamo considerare in pratica la Configuration come l'insieme delle librerie per la gestione delle operazioni di basso livello; viceversa il Profile è utilizzato per la gestione delle operazione di alto livello.

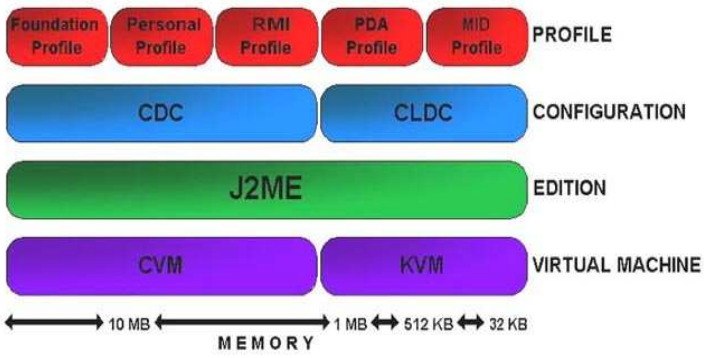

Fig. 1.3: Configurazioni e profili di J2ME

1.3 Organizzazione di un tipico programma j2me

1.3.1 MIDP (Mobile Information Device Profile)

Il Mobile Information Device Profile è un profilo Java 2 Micro Edition che si basa sulla configurazione CLDC (Connected Limited Device Configuration) ed è diretto allo sviluppo di applicazioni Java per dispositivi wireless (telefoni cellulari, smartphone, two-way-pagers, palmari). La CLDC, seguendo la filosofia J2ME lascia irrisolti alcuni aspetti fondamentali per la realizzazione di applicazioni reali; le librerie MIDP come vedremo si integrano ed estendono quelle della configurazione permettendo la realizzazione, la gestione e l'esecuzione degli applicativi java. In particolare il MIDP si occupa di:

- gestire il ciclo di vita delle applicazioni (caricamento - esecuzione - distruzione delle applicazioni)
- gestire l'interfaccia utente (dispositivi di input/output)
- salvataggio persistente dei dati
- networking (implementazione dei protocolli)

Questi quattro punti sono implementati in altrettanti packages, rispettivamente:

- javax.microedition.midlet

- javax.microedition.lcdui
- javax.microedition.rms
- javax.microedition.io

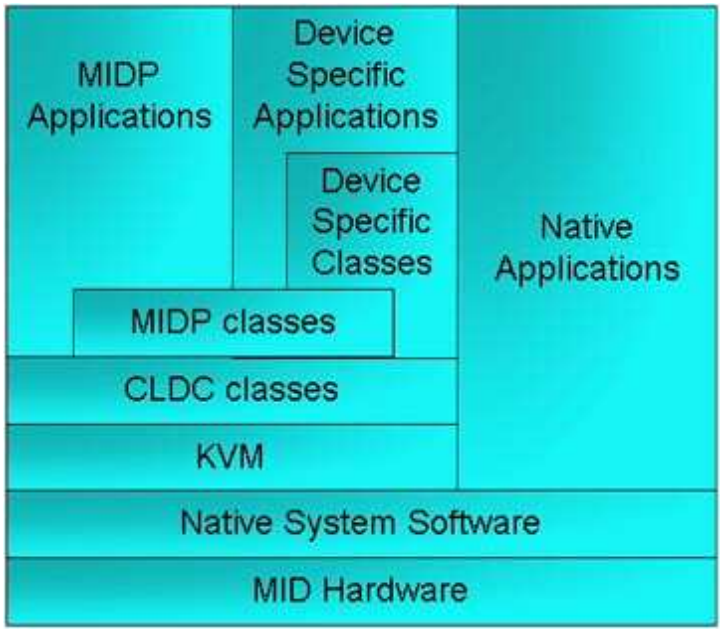

Fig. 1.4: Architettura MIDP

In figura 1.4 è mostrata la struttura dell'architettura MIDP: alla base dell'architettura troviamo, ovviamente, l'hardware del dispositivo host e il software nativo, cioè il sistema operativo e le librerie native. Al di sopra di questi due livelli si posa tutta l'architettura J2ME. La KVM (Kilobyte Virtual Machine) fornisce l'ambiente runtime per l'esecuzione delle applicazioni Java, mentre CLDC e MIDP forniscono il set completo di API Java di cui il programmatore potrà disporre. Come è possibile osservare oltre alle librerie Java standard (CLDC-MIDP) possono essere disponibili librerie proprietarie specifiche per un dato dispositivo che se da un lato possono aumentare potenzialità e prestazioni dall'altro pregiudicano la portabilità del codice (device-specific applications). Per fare un esempio esistono librerie Siemens utilizzabili sul modello Siemens SL45i che permettono l'invio di messaggi SMS:

import com.siemens.mp.io.sms.*;
un applicativo che fa uso di questa porzione di codice potrebbe essere relegato a funzionare esclusivamente su questo o altri modelli Siemens (ad esempio non funziona su Nokia e Motorola).

1.3.2 Gestione del ciclo di vita delle applicazioni : midlet

Un'applicazione per dispositivi MIDP è chiamata MIDlet ed è qualcosa di simile ad un'Applet java. Un MIDlet non possiede un metodo main, ma deve estendere la classe javax.microedition.midlet.MIDlet ed implementare i suoi metodi astratti relativi allo stato in cui il MIDlet stesso può trovarsi:

protected abstract void startApp() segnala al MIDlet che è stato posto nello stato di attività

protected abstract void pauseApp() segnala al MIDlet il suo inserimento nello stato di pausa

protected abstract void destroyApp (boolean unconditional) impone al MIDlet di terminare le proprie attività ed entrare nello stato destroyed.

Il cambiamento di stato di un MIDlet è controllato dall'AMS Application Management Software (talvolta chiamato anche MIDlet Management Software o Java Application Manager) questo altro non è che un'applicazione software che fa parte dell'implementazione del MIDP stesso e svolge le funzioni di gestore dei MIDlet essendo responsabile della loro installazione, esecuzione e rimozione. Più precisamente l'AMS:

- garantisce la possibilità di installare e disinstallare MIDlet nel/dal dispositivo
- prepara l'ambiente di esecuzione per il MIDlet (KVM, librerie CLDC-MIDP, librerie di terzi, file JAD, risorse esterne, ecc)
- cattura gli errori che si possono verificare sia in fase di installazione del MIDlet che durante la sua esecuzione

In figura 1.5 è riportato il diagramma dei possibili stati di un MIDlet. Nel momento in cui esso è pronto per essere eseguito l'AMS ne crea un'istanza e pone il MIDlet nello stato di pausa. A questo punto con la chiamata del

metodo startApp () il MIDlet passa allo stato attivo e riceve tutte le risorse di cui necessita per il funzionamento. A questo punto si possono verificare due condizioni: il MIDlet può essere distrutto oppure può entrare nello stato di pausa. In questo caso rilascia temporaneamente le risorse di cui disponeva e diventa inattivo. Una nuova chiamata al metodo startApp () può riportare il MIDlet dallo stato di pausa a quello di attività. Il MIDlet può comunque decidere della propria sorte, attraverso i metodi:

notifyPaused() - il MIDlet informa l'AMS che è entrato nello stato di pausa

resumeRequest() - il MIDlet richiede all'AMS la possibilità di rientrare in attività

notifyDestroyed() - il MIDlet informa l'AMS che verrà distrutto

esso può decidere il proprio cambiamento di stato.

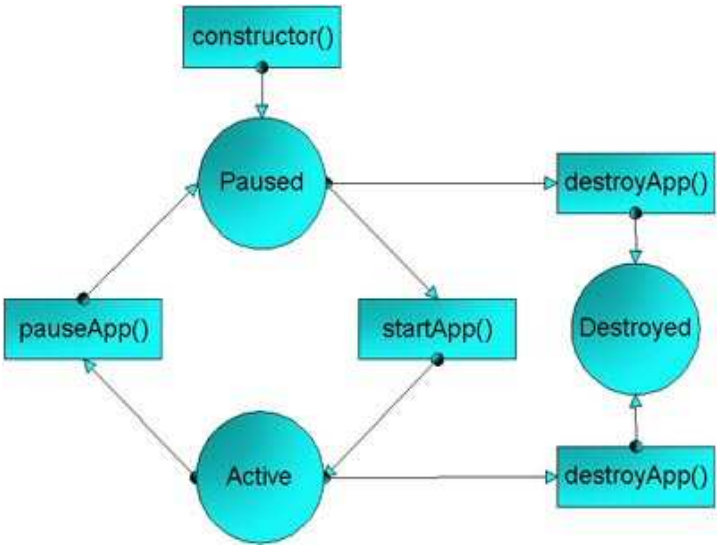

Fig. 1.5: Architettura MIDP

1.4 Librerie J2ME (JSR)

Per ora abbiamo visto diverse componenti del mondo J2ME.La KVM che è la virtual machine,CLDC che è la configurazione con cui vengono implementate

delle funzionalità base del linguaggio Java e il profilo MIDP che permette (insieme a CLDC) di avere un ambiente di sviluppo completo per le applicazioni. A tutta questa serie di componenti dobbiamo aggiungere gli Optional Package e le librerie di terze parti.Gli Optional Package nascono da JSR (Java Specification Request) e permettono la definizione di un determinato tipo di funzionalità che una device puo' supportare oppure no.
Un esempio di Optional Package sono le WMA (Wireless Messaging API),API per la gestione della messaggistica.Praticamente esistono tutta una serie di Optional Package che arricchiscono piano piano la piattaforma di sviluppo.
C'è da dire che mentre una configurazione e un profilo sono necessari alla composizione di un ambiente J2ME, l'implementazione di un Optional Package all'interno di un determinato dispositivo è a discrezione del produttore, il quale puo' decidere quali funzionalità includere nella piattaforma.
A differenza di librerie JAR, che talvolta possiamo includere nei nostri programmi J2SE, gli Optional Package non possono essere inseriti direttamente dallo sviluppatore in un telefonino. Ovvero, se un cellulare non possiede le API per la gestione della connessione Bluetooth, noi non potremo includerle in nessuna maniera.
Molto importanti sono le librerie di terze parti, ovvero librerie sviluppate basandosi sulle funzionalità offerte da CLDC e MIDP che, a differenza degli Optional Package possono essere incluse all'interno di un nostro progetto. Un esempio di libreria è kXML,che permette il parsing di XML in applicazioni J2ME.

Di seguito un elenco delle principali JSR che un dispositivo mobile può possedere, elencarle tutte è impossibile per motivi di spazio e comunque quelle usate in questa tesi sono solo una piccolissima parte:

- **JABWT - JSR 82** : JABWT (Java BlueTooth Wireless Technology) sono delle API che descrivono completamente le interazioni che è possibile fare da Java con una radio Bluetooth. Queste API sono disponibili sia per la versione J2ME, dove vengono implementate dai vari costruttori, sia nella versione J2SE con librerie ch aderiscono allo standard.

- **J2ME Web Services API - JSR 172**: Anche nel mondo embedded sono sbarcati i WebServices. Infatti grazie a questa JSR è possibile implementare un client che sfrutta un WebServices. Alla base di questa API ci sono le librerie per il parsing dell'XML e per l'invio di richieste stile RPC (JAXP e JAX-RPC). Diversi toolkit mettono a disposizione dei tool per creare le classi stub per l'utilizzo dei WebServices inserendo

semplicemente l'url o il file WSDL. In questo modo vengono create le classi che si occupano di richiamare il servizio. Sono utili comunque anche i package relativi al parsing XML, che permettono di avere un parser di default senza ricorrere a librerie esterne.

Questa libreria è usata dalla maggior parte di applicazioni di guida turistica sviluppate in J2ME, in quanto usano connessioni GPRS per connettersi a determinati server internet per scaricare le pagine web con tutte le informazioni relative al luogo.

- **Mobile 3D Graphics API - JSR 184**: J2ME è il re incontrastato sui dispsitivi mobili per quanto riguarda i giochi. Infatti parte del suo successo lo deve proprio a questa fascia di mercato, che ha convinto piano piano tutti i produttori a includere una JVM nei propri dispositivi. Chiaramente essendo i giochi una parte importante dello sviluppo J2ME non poteva mancare una libreria per il rendering 3D. Grazie a questa API abbiamo ha a dispozione una serie di classi/interfacce che rendono possibile e molto agevole lo sviluppo di applicazioni 3D su dispositivi J2ME.

- **PIM e FileConnection - PDA Optional Packages - JSR 75**: Questa API apre la strada verso nuovi orizzonti per i programmi J2ME. Infatti chi ha iniziato a sviluppare dalle prime versioni ha accolto questa nuova API con molto entusiasmo. Praticamente è ora possibile interagire con il filesystem del dispositivo (ram o memory card esterne) e con le classiche informazioni che possiamo avere su un celllare come rubrica, todo, note.

- **Location API - JSR 179**: Questa JSR definisce un API ad alto livello per il reperimento di informazioni geografiche tramite Java. Leggendo dettagliatamente il documento di specifica di questa JSR ci si rende conto che lo scopo finale è di fornire una potentissima API per la localizzione. Sui dispositivi mobili questa API si scontra chiaramente con le implementazioni dei differenti produttori e soprattutto sulle informazioni reperibili a livello di rete cellulare GSM/GPRS/UMTS. Su alcuni dispositivi Motorola è possibile usufruire del sistema A-GPS (Assisted GPS) per avere queste informazioni di localizzazione. Questa è una di quelle API che è molto legata alla coppia dispositivo/rete quindi c'è bisogno di documentarsi ulteriormente quando si vuole implentare un programma che la utilizzi.

All'inizio pensavo di usare questo sistema per il reperimento delle coordinate, ma ho scoperto poi che il nokia 6630 non implementa nativamente questa libreria e come spiegato prima non c'è modo di inserirla da parte dello sviluppatore.

- **SIP API - JSR 180**: SIP (Session Initiation Protocol) è un protocollo standard pensato per gestire le connessioni riguardanti comunicazioni multimediali su Internet. Praticamente questo è uno standard che viene utilizzato per la realizzazione di programmi VoIP (Voice over IP). Quello che di solito viene ignorato è che SIP gestisce soltanto le connessioni, mentre lo scambio di dati multimediali, come ad esempio la voce, deve essere realizzato con altri mezzi. Comunque troviamo a disposizione questa API negli ultimi modelli di dispositivi, utile per realizzare interessanti programmi di comunicazione.

- **Scalable 2D Vector Graphics API - JSR 226**: Questa specifica definisce un API per il rendering 2D su dispositivi mobili. Si basa principalmente sulla Java 2D API disponibile per la versione J2SE e permette inoltre la possibilità di gestire immagini nel formato SVG (Scalable Vector Graphics).

Questo è solo un piccolo elenco delle librerie esistenti, tanto per dare un'idea di massima di quello che si può fare con questa tecnologia al giorno d'oggi; da considerare il fatto che l'elenco non è fermo, ma in continua espansione e più potenti diventeranno le risorse hardware dei dispositivi e più performanti diventeranno queste librerie.

1.5 Bluetooth

Ideata da Ericsson, che nel 1994 lanciò una campagna di ricerca per studiare interfacce radio di basso costo e consumo, destinate a collegare i telefonini con il resto del mondo digitale. L'evoluzione vera e propria si ebbe all'inizio del 1998, quando partì l'iniziativa Bluetooth Special Interest Group (SIG), consorzio che comprende la stessa Ericsson insieme a Nokia, Ibm, Lucent, Motorola Nokia, Toshiba, Intel, 3Com e di altre 1400 compagnie, tutte unite dall'obiettivo di sviluppare uno standard per l'interconnessione via onde radio.
All'interno del panorama Wireless (connessioni senza fili) il Bluetooth è sicuramente una tecnologia che rivoluzionerà il mercato della connessione delle reti wireless, principalmente per i bassi costi di trasmissione su cui essa si

basa e soprattutto per la possibilità di far comunicare qualunque tipo di dispositivo wireless attraverso onde radio (telefoni, stampanti, notebook, PDA, impianti HiFi, tv,computer, PC, cellulari, elettrodomestici, device,etc) Per fare ciò ciascun dispositivo deve possedere all'interno di un chip, integrato, in grado di trasmettere e ricevere informazioni nell'etere usando un ricetrasmettitore che opera nella frequenza di 2,4 GHz, frequenza assegnata per usiindustriali.

Questi piccoli network wireless (reti domestiche) sono generalmente chiamati piconet. Un piconet è costituito da due o più periferiche che condividono un canale di comunicazione utilizzando Bluetooth , fino ad un massimo di 8 dispositivi. La frequenza di 2,4 Ghz è sotto le frequenze UHF radio amatoriali che vanno dai 5,65 - 5,85 Ghz.
Il sistema di comunicazione bluetooth è progettato per funzionare anche in ambienti con forte presenza di interferenze.La velocità di comunicazione è prossima ad 1 Mbps anche con piccole potenze nell'ordine di alcuni milli-Watt, impiegando la tecnologia TDD (Time Division Duplex, gestione del traffico asimmetrico).
Bluetooth permette infatti di gestire sia i dati sia la voce, utilizzando una trasmissione a pacchetto su rete radio per i dati e una modalità connection-oriented per la voce. I dispositivi dotati di questa tecnologia comunicano dunque tra loro creando e riconfigurando dinamicamente (la configurazione cambia infatti automaticamente quando si inserisce o si elimina un dispositivo) delle reti ad hoc dette picoreti ,composte da un massimo di otto nodi; più picoreti possono a loro volta interconnettersi, aumentando le possibilità di espansione.
Tutto questo è possibile grazie al service discovery protocol (SDP)che permette ad un dispositivo Bluetooth di determinare quali sono i servizi che gli altri apparecchi presenti nella picorete mettono a disposizione.Tale protocollo può fungere sia da server (ossia può essere interrogato da un altro dispositivo e rispondere con i propri servizi) sia da client (interrogando gli altri dispositivi) e ogni apparecchio dispone delle informazioni relative ai servizi di cui è capace e dei protocolli supportati.
Quando un dispositivo si inserisce per la prima volta in una picorete, inoltre, effettuerà una scansione di tutti i nodi presenti per capire come può interagire con essi.

Una piccola rete Bluetooth può supportare un collegamento punto a punto (point to pont) e multipunto (multipoint). Tale modalità di interconessione dinamica consente di sincronizzare i dati tra due apparecchi Bluetooth automaticamente, ad esempio un pda con un notebook, sfruttando l'SDP per

distanze comprese tra 10 - 100 metri.
In un collegamento tutti gli apparecchi Bluetooth connessi sono generalmente in modalità standby, cioè di attesa, seguendo un ciclo di scansione ad intervalli di tempo al fine di verificare la presenza di eventuali altri dispositivi.La scansione effettuata può essere di due tipi : **PS (Page Scan)** e **IS (Inquiry Scan)**.
La scansione PS consente la la ricerca di un collegamento con un altro apparecchio Bleutooth, che può risultare in modalità: connectable mode non-connectable mode.
La scansione IS simile alla precedente, permette di identificare la tipologia di apparecchi disponibili nella picorete, discoverable mode o non-discoverade mode, e di approntare i necessari protocolli per il collegamento.
Il Bluetooth opera nella gamma di frequenza dai 2,4 ai 2,483 Ghz, suddivisa in canali da 1 Mhz impiegando la tecnica FHSS*(Frequency Hopping Spread Spectrum, tecnologia che consente a più utenti di condividere lo stesso insieme di frequenze, cambiando automaticamente la frequenza di trasmissione fino a 1600 volte al secondo, al fine di una maggiore stabilità di connessione e di una riduzione delle interferenze tra canali di trasmissione).
Lo spectrum spreading consiste in una continua variazione di frequenza utilizzando una modulazione di frequency hopping.Gli hops corrispondono ai salti di frequenza all'interno della gamma assegnata (2,402 Ghz -2,480 Ghz, con salti di 1 Mhz, complessivamente 79 hops set). La tecnologia Bluetooth consente due principali modalità di collegamento tra unità master e slave:

1. **ACL**. Il collegamento ACL (Asynchronous Connectionless) consente la trasmissione dei dati (Td) con una modalità sincrona. La velocità di trasmissione dati nella modalità asimmetrica sarà 723 Kbps - 57,6Kbps nell'altra direzione, nella modalità simmetrica invece, sarà pari a circa 434 Kbps.

2. **SCO**. Il collegamento SCO (Synchronous Connection Oriented) consente una trasmissione radio (Tr) e una trasmissione Voce (Tv).La velocità di trasmissione voce sincrona bidirezionale sfrutta una codifica voce Continuous Variable Slope Delta Modulation (CVSD) , permettendo un bit rate di 64 Kbps. I dati di una piconet vengono trasmessi a pacchetti e sono composti da un AC (Access Code), da un H (Header), e da un P (Payload).

La modalità di trasmissione e ricezione dati può cambiare in relazione alle esigenze di comunicazione delle varie unità Bluetooth, passando per esempio da una sola comunicazione voce a una sola comunicazione dati.

Gli sviluppi del Bluetooth,al momento attuale, seguono le specifiche della versione Bluetooth 2.1+EDR (Enhanced Data Rate) che consente un incremento di velocità di trasmissione, un raggio d'azione da 10 a 100 metri (a seconda della classe di funzionamento),supporto simultaneo per slave a bassa e alta velocità, bassi costi, conformità con versione Bluetooth 1.0 e 2.0 e soprattutto, molto importante al giorno d'oggi, consumi energetici inferiori alle versioni precedenti.

In questa tesi l' uso del Bluetooth è indispensabile per il collegamento del ricevitore GPS al cellulare usato per lo sviluppo.

1.6 Global Position System (GPS)

Nel 1991 gli USA aprirono al mondo il servizio con il nome SPS (Standard Positioning System), con specifiche differenziate da quello militare denominato PPS (Precision Positioning System). In pratica veniva introdotta la cosiddetta Selective Availability (SA) che introduceva errori intenzionali nei segnali satellitari allo scopo di ridurre l'accuratezza della rilevazione, consentendo precisioni solo nell'ordine di 100-150 m. Il GPS è stato creato in sostituzione del precedente sistema, il Transit, quando gli USA, rinunciando alla Selective Availability, hanno reso il primo accurato quanto il secondo, supportandolo con una rete di 24 satelliti artificiali.
L'UE ha in progetto il completamento di una propria rete di satelliti, il Sistema di posizionamento Galileo, per scopi civili, fra i quali il GPS. Questo progetto ha un'evidente valenza strategica in quanto la rete americana è proprietà dei soli Stati Uniti d'America ed è gestita da autorità militari, che, in particolari condizioni, potrebbero decidere discrezionalmente e unilateralmente di ridurre la precisione o bloccare selettivamente l'accesso al sistema; la condivisione dell'investimento e della proprietà da parte degli stati utilizzatori garantisce continuità, accessibilità e interoperabilità del servizio.

1.6.1 Funzionamento

Il sistema di navigazione si articola nelle seguenti componenti:

- un complesso di 25 satelliti, di cui 3 non attivi divisi in gruppi di quattro su ognuno dei sei piani orbitali (distanti 60°fra loro e inclinati di 55°sul piano equatoriale)

- 2 cicli al giorno

- una rete di stazioni di tracciamento (tracking station)
- un centro di calcolo (computing station)
- due stazioni di soccorrimento (injection stations)
- un ricevitore GPS

I satelliti sono disposti su 6 piani orbitali inclinati di 55°rispetto al piano equatoriale (quindi non coprono le zone polari) a forma di ellissi a bassa eccentricità. Ogni piano orbitale ha 3 o 4 satelliti, e i piani sono disposti in modo tale che ogni utilizzatore sulla terra possa ricevere i segnali di almeno 5 satelliti. La loro quota è di 20200 km e compiono due orbite complete in un giorno siderale. Ciascun satellite emette sulle frequenze di 1,2 e 1,5 GHz derivate da un unico oscillatore ad alta stabilità. Lo scopo della doppia frequenza è quello di eliminare l'errore dovuto alla rifrazione atmosferica. Su queste frequenze portanti, modulate in fase, vengono emessi i messaggi di effemeride, ciascuno della durate di due minuti; essi iniziano e terminano ai minuti pari interi del GMT. Questi messaggi di effemeride contengono il segnale orario e i parametri orbitali del satellite. In tal modo il ricevitore GPS, mentre effettua il conteggio doppler, riceve i parametri dell'orbita da cui deriva la posizione del satellite: viene così a disporre di tutti gli elementi necessari a definire nello spazio la superficie di posizione.
In orbita vi sono 24 satelliti per la trasmissione di dati GPS, più 3 di scorta. Da questo si evince che da un punto del globo terrestre il ricevitore riesce a vedere solo la metà di essi, quindi 12. Ma non li vedrà mai tutti e 12 per via della loro inclinazione rispetto all'equatore. In più il ricevitore GPS stesso fa una discriminazione dei satelliti: preferisce quelli più perpendicolari possibile per questione di ricezione del timing in quanto il dato da quelli con più inclinazione arriverebbe con maggiore ritardo. Ogni satellite è dotato di 4 oscillatori ad altissima precisione, di cui 2 al cesio e 2 al rubidio; ha dei razzi per effettuare le correzioni di orbita.
Il tracciamento dei satelliti comprende tutte quelle operazioni atte a determinare i parametri dell'orbita. A ciò provvedono 4 stazioni principali dette appunto di tracciamento (main tracking stations) e un centro di calcolo (computing center), tutti situati in territorio USA, ed in particolare a Wahiova (Hawaii), Point Mogu (California), Prospect Harbur (Maine) e Rosemount (Minnesota). Point Mogu è anche sede del centro di calcolo. Ogni volta che ciascun satellite nel suo moto orbitale sorvola il territorio americano le stazioni di tracciamento ne registrano i dati doppler che vengono avviati al centro di calcolo e qui valorizzati per la determinazione dei parametri orbitali. Per risolvere questo problema è stato necessario venire in possesso di un

fedele modello matematico del campo gravitazionale terrestre. La costruzione di questo modello è stato uno dei problemi di più ardua soluzione nello sviluppo del progetto Transit da cui è derivato l'attuale Navstar. I risultati di questa indagine sul campo gravitazionale terrestre, che sono di vasta portata dal punto di vista geodetico, possono riassumersi in una immagine del globo nella quale vengono riportate le linee di eguale scostamento del Geoide (LMM) dall'ellissoide di riferimento APL. I satelliti, dotati di orologi atomici al cesio di grandissima precisione, che vengono sincronizzati dalla stazione americana di Colorado Spring ogni qual volta vi passano sopra, trasmettono in continuazione dati numerici che comprendono le proprie coordinate X, Y, Z, e l'istante esatto T di trasmissione.
Questi dati vengono elaborati a terra dal ricevitore il quale, confrontandoli con il proprio tempo locale, il quanto anche il ricevitore è dotato di orologio al quarzo di grande, anche se non di grandissima precisione, e conoscendo la velocità delle onde elettromagnetiche, deduce a che distanza si trova da ognuno dei satelliti di cui sta ricevendo il segnale. Infatti, noto l'istante T1 trasmesso dal satellite in cui è partito il segnale, e l'istante T2, indicato dall'orologio locale, in cui il segnale è stato ricevuto a terra, si conosce il tempo impiegato a percorrere la distanza dal satellite al ricevitore, ed essendo la velocità della luce c, nota, la distanza D del satellite dal ricevitore risulta:

$$D = c\ (T2 - T1)$$

La conoscenza della distanza da un solo satellite è un dato del tutto insufficiente per determinare la propria posizione, in quanto non è nota la posizione azimutale nè quella zenitale dello stesso, analogamente non è sufficiente conoscere la distanza da due satelliti; infatti, l'intersezione di due sfere di raggio noto, cioè le distanze calcolate, dà luogo ad un cerchio e non ad un punto. L'intersezione di tre sfere di raggio noto, invece, determina due punti, dei quali, invero, uno è di norma inaccettabile in quanto si trova ad altissima quota e risulta anche muoversi ad altissima velocità.
Soltanto l'intersezione di quattro sfere di raggio noto, invece, consente con certezza, di determinare una posizione univoca nello spazio, il che spiega perchè è necessario aspettare del tempo, anche se se tratta di minuti, per elaborare i dati, in quanto bisogna aspettare il passaggio di almeno quattro satelliti ed avere anche il tempo di effettuare numerosi calcoli ed approssimazioni successive. I dati del quarto satellite, infatti, oltre a rendere univoca la soluzione al sistema di quattro equazioni in quattro incognite, consentono di correggere il valore del tempo proprio del ricevitore per mezzo dei tempi dei quattro satelliti. Le quattro incognite da determinare sono X, Y, Z, T, cioè le tre coordinate indicanti la posizione geografica dell'utente fornito di ricevitore, più il tempo proprio che è indispensabile per determinare con

grande precisione le distanze dei satelliti, distanze che costituiscono i dati di partenza. La X rappresenta la longitudine, la Y rappresenta la latitudine, la Z, la quota sul livello del mare, e T il tempo proprio, corretto dagli orologi ad altissima precisione che orbitano sui satelliti.

1.6.2 Grado di precisione del rilievo

Il primo segnale consente la determinazione della propria posizione con la precisione di circa 300 metri, il secondo invece, con la precisione di 50 cm. Mentre il primo segnale è trasmesso in chiaro, il secondo, invece, è trasmesso in codice segreto e non è accessibile se non al Ministero della difesa degli Stati Uniti che lo utilizza esclusivamente per la propria sicurezza e non lo rende noto a tutti per evitare che possa essere utilizzato contro gli interessi degli Stati Uniti da criminali o da Stati nemici. Ogni satellite trasmette dei segnali ad alta frequenza verso terra che vengono ricevuti da un apposito apparecchio ricevitore delle dimensioni ridottissime di un comune cellulare e comprendente una particolare antenna regolabile che va rivolta verso l'alto. I ricevitori G.P.S. funzionano all'aperto, non è quindi possibile utilizzarli all'interno di un appartamento o in sotterranei, ma devono poter vedere il cielo libero su di se per ricevere il segnale satellitare.

1.6.3 Il GPS nell'utilizzo quotidiano

I moderni ricevitori GPS hanno raggiunto dei costi molto contenuti. Dopo il telefono cellulare stiamo assistendo alla diffusione di un nuovo cult: quello del navigatore satellitare personale. Il mercato offre ormai soluzioni a basso costo per tutti gli impieghi e per tutte le tasche che si rivelano efficaci non soltanto per la navigazione satellitare in sè e per sè, ma anche per usi civili, per il monitoraggio dei servizi mobili e per il controllo del territorio. Esistono varie soluzioni:

- Integrate: sono dispositivi portatili All-in-One che incorporano un ricevitore GPS, un display LCD, un altoparlante, il processore che esegue le istruzioni, date solitamente da un sistema operativo proprietario, uno slot per schede di memoria ove memorizzare la cartografia.

- Ibride: sono dispositivi portatili (PC, Palmari, SmartPhone) che, nati per scopi diversi, sono resi adatti alla navigazione satellitare attraverso il collegamento di un ricevitore GPS esterno (Bluetooth o via cavo) e l'adozione di un software dedicato, in grado di gestire la cartografia.

Con la diffusione capillare dei sistemi GPS, e il conseguente abbattimento dei costi dei ricevitori, molti produttori di telefoni cellulari hanno cercato di inserire un modulo GPS all'interno dei loro prodotti, aprendosi quindi al nuovo mercato dei servizi LBS (Location Base Services, servizi basati sul posizionamento). Tali servizi vengono sempre più sfruttati per offrire anche sul web dei servizi molto utili. Tuttavia, la relativa lentezza con cui un terminale GPS acquisisce la propria posizione al momento dell'accensione (in media, tra i 45 e i 90 secondi), dovuta alla necessità di cercare i satelliti in vista, ed il conseguente notevole impegno di risorse hardware ed energetiche, ha frenato in un primo momento questo tipo di abbinamento. Negli ultimi anni, però, è stato introdotto in questo tipo di telefoni il sistema Assisted GPS, detto anche A-GPS, con cui è possibile ovviare a tali problemi: si fanno pervenire al terminale GPS, attraverso la rete di telefonia mobile, le informazioni sui satelliti visibili dalla cella a cui l'utente è agganciato. In questo modo un telefono A-GPS può in pochi secondi ricavare la propria posizione iniziale, in quanto si assume che i satelliti in vista dalla cella siano gli stessi visibili dai terminali sotto la sua copertura radio. Tale sistema è molto utile anche come servizio d'emergenza, ad esempio per localizzare mezzi o persone ferite in seguito ad un incidente.

1.6.4 Calcolo della distanza tra due punti geografici

Il calcolo della distanza tra due punti di coordinate geografiche diverse non è così immediato come si può pensare, ciò è dovuto alla particolare forma sferica della terra, quindi serve una particolare formula matematica che tiene conto dei seguenti punti:

- La distanza chilometrica tra un parallelo e l'altro è sempre la stessa (circa 110,9467 Km) in qualsiasi punto della Terra. Per esempio, se - scendo dal decimo parallelo Nord fino all'equatore, viaggiando lungo un meridiano, potrò conoscere la distanza percorsa con un semplicissimo calcolo: 110,9467 x 10= 1.109,467 Km. Insomma, viaggiando lungo una qualsiasi circonferenza polare e assumendo che la terra è completamente sferica, per ogni percorso tra un parallelo e l'altro intercorre la stessa distanza di 110,9467 Km.

- La stessa cosa non accade se, a latitudini diverse, passiamo da un meridiano all'altro, cioè se viaggiamo lungo un parallelo. Infatti, percorrendo l'equatore (parallelo zero gradi) in tutta la sua circonferenza avremmo percorso una distanza di 40.075,0354 km; mentre percorrendo il 30°o il 60ij parallelo in tutta la circonferenza ciascun percorso sarà

evidentemente inferiore a quello equatoriale. Man mano che ci allontaniamo dall'equatore, sia verso nord che verso sud, la latitudine aumenta mentre diminuisce la distanza chilometrica tra un meridiano ed il successivo, tanto che tale distanza si azzera in corrispondenza dei poli. Quindi, passare da un meridiano all'altro a latitudini diverse, implica percorrere distanze diverse.

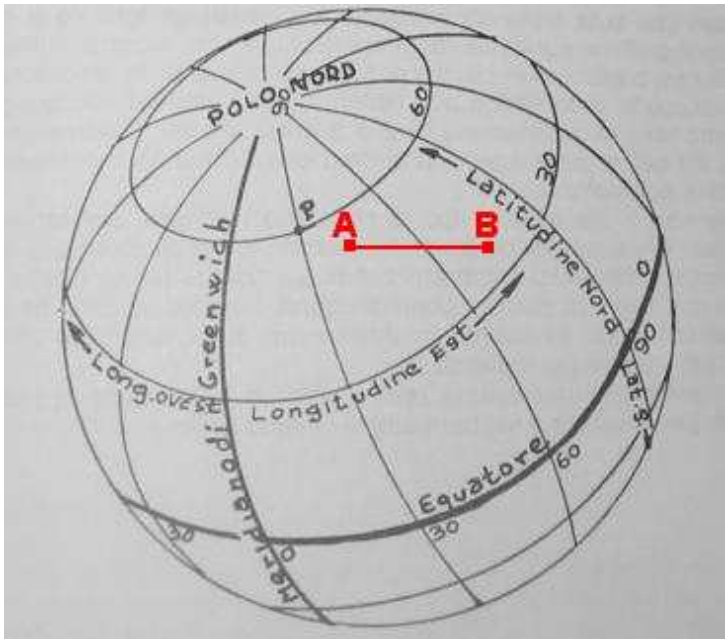

Fig. 1.6: Due punti qualsiasi

Quindi la corretta modalità di calcolo è la seguente:

Siano $\phi_s, \lambda_s, \phi_f, \lambda_f$ la latitudine (ϕ) e la longitudine (λ) rispettivamente dei punti A e B; sia ($\Delta\lambda$) la differenza tra le due longitudini e ($\Delta\hat{\sigma}$) la distanza angolare tra i due punti considerati (A,B), la distanza angolare tra A e B è data dalla formula:

$$\Delta\hat{\sigma} = \arctan\left(\frac{\sqrt{(\cos\phi_f \sin\Delta\lambda)^2 + (\cos\phi_s \sin\phi_f - \sin\phi_s \cos\phi_f \cos\Delta\lambda)^2}}{\sin\phi_s \sin\phi_f + \cos\phi_s \cos\phi_f \cos\Delta\lambda}\right)$$

Basta moltiplicare il risultato ($\Delta\hat{\sigma}$) x 6372 (il raggio della terra in km) per ottenere la distanza in km.

1.7 Standard NMEA-0183 nella comunicazione GPS

1.7.1 Descrizione dello Standard

Lo standard di interfacciamento NMEA 0183 definisce le specifiche per il segnale elettrico, il protocollo per la trasmissione dei dati e il tempo, e specifica il formato della frase per un bus dati seriale a 4800 baud. Ogni bus deve avere solo un dispositivo che comunica, ma può avere più ascoltatori. Formati della frase specifici sono comuni a entrambi NMEA 0183 e NMEA 0183-HS e sono definiti nello standard NMEA 0183. Lo standard è un documento protetto da copyright e disponibile esclusivamente presso NMEA. Molti software che forniscono le informazioni sul posizionamento in tempo reale ricevono ed elaborano i dati nel formato NMEA. Questi dati includono la soluzione PVT completa (PVT: position, velocity, time), elaborata da un ricevitore GPS. L'idea di NMEA è quella di spedire una linea di dati, chiamata frase, che contiene al suo interno la completa informazione, ed è indipendente da altre frasi. Ci sono frasi standard per ogni categoria di dispositivo, e esiste anche la possibilità di definire frasi proprietarie per essere utilizzate individualmente dalle compagnie.
Tutte le frasi conformi allo standard hanno un prefisso di due lettere che definisce il dispositivo che utilizza il tipo di frase in questione. Per i ricevitori GPS questo prefisso è GP. I formati di dati del sistema NMEA sono molto numerosi, e solo una parte molto limitata ha rilevanza nell'ambito del GPS, dove il sistema NMEA viene prevalentemente impiegato per trasmettere dati da un ricevitore GPS verso un computer, un palmare o un telefono cellulare bluetooth.

1.7.2 Sintassi di una frase NMEA 0183

Tutti i dati sono trasmessi sotto forma di frasi (sentences). Solo i caratteri ASCII stampabili sono permessi, più CR (carriage return) e LF (line feed). Ogni frase inizia con il simbolo $ e termina con <CR><LF>. Ci sono tre tipi base di frasi:

1. Frasi del trasmettitore (Talker Sentences)

   Il formato generale per una frase del trasmettitore è:

   $ttsss,d1,d2,....<CR><LF>

Le prime due lettere che seguono il simbolo $ (tt) rappresentano l'identificatore del dispositivo che trasmette. I successivi tre caratteri (sss) sono l'identificatore della frase, seguito da un numero variabile di campi dati, separati da virgole, seguiti a loro volta da un checksum opzionale, e terminate da un carriage return/line feed. I campi dati sono unicamente definiti per ogni tipo di frase. Un esempio di frase del trasmettitore è:

$HCHDM,238,M<CR><LF>

2. Frasi proprietarie (Proprietary Sentences)

   Lo standard permette ai produttori individuali di definire formati di frasi proprietarie. Queste frasi iniziano con $P, seguite da un ID di 3 lettere del produttore, seguito a sua volta da qualsiasi dato il produttore desideri, seguendo il formato generale delle frasi standard. Garmin e Magellan, due case produttrici di dispositivi GPS, utilizzano frasi proprietarie per i propri dispositivi. Per esempio una frase Garmin inizia con PGRM e le frasi Magellan iniziano con PMGN.

3. Frasi interrogative (Query Sentences)

   Una frase interrogativa è un mezzo che ha un ascoltatore per richiedere una particolare frase ad un trasmettitore. Il formato generale è:

   $ttllQ,sss,<CR><LF>

   I primi due caratteri del campo indirizzo sono l'identificatore del dispositivo che invia la richiesta, e i successivi due caratteri sono l'identificatore del dispositivo in ascolto che è interrogato. Il quinto carattere è sempre una Q, che definisce il messaggio come query (interrogazione). Il campo successivo (sss) contiene le tre lettere mnemoniche della frase che è stata richiesta. Un esempio di frase interrogativa è:

   $CCGPQ,GGA<CR><LF>

   dove il dispositivo CC (in questo caso un computer) sta richiedendo al dispositivo GP (una unità GPS) la frase GGA. Il GPS trasmetterà allora questa frase una volta al secondo fino a che non è fatta una nuova interrogazione.

1.7.3 Prefissi

I dispositivi collegabili tra loro, con sistema NMEA, ricadono in diverse categorie:

- Autopiloti.
- Dispositivi di comunicazione (radio e anche GPS).
- Sensori di direzione (bussole e giroscopi).
- Dispositivi LORAN.
- Sensori di velocità ed una categoria per altri trasduttori.

Tutte le sentenze NMEA, come detto, iniziano con un prefisso costituito dal carattere $, seguito da due caratteri che identificano l'entità che le genera, talker. Nell'ambito del GPS, le frasi iniziano con $GP, dove GP denota che essa è stata generata da un dispositivo GPS. I successivi tre caratteri del prefisso della frase indicano il tipo di frase.

Esempio 1:

una frase che inizia con $GPGLL è inviato da un dispositivo di tipo GPS (GP) ed è del tipo Geographic position, Longitude and Latitude (GLL).

1.7.4 Formati dei campi

La frase, dopo il prefisso, continua con una serie di campi, separati da virgole, che contengono le informazioni vere e proprie. Si noti che tra le varie versioni (e spesso tra diverse implementazioni e costruttori) le lunghezze dei campi possono variare o non essere presenti. Chi scrive un software di decodifica considera i campi di lunghezza variabile e si basa sulle virgole. Inoltre, alcune implementazioni includono gli zeri davanti ai numeri, altri no, e il numero di cifre decimali varia tra diverse implementazioni. Ecco i formati dei dati usati dalle frasi NMEA GPS:

- *[hhmmss.ss]*: Ore, minuti, secondi e centesimi di secondo
  (es: 132957.94 = 13:29:57.94).
- *[ddmmyy]*: Campo data giorno, mese, anno
  (es: 151005 = 15 ottobre 2005).

- *[A]*: Campo numerico di lunghezza fissa di un carattere.
- *[A-A]*: Campo alfanumerico di lunghezza variabile.
- *[llll.ll]*: Campo latitudine (es: 4531.47 = 45°31.47').
- *[yyyyy.yy]*: Campo longitudine (es: 00917.21 = 009°17.21').
- *[x]*: Campo numerico con n cifre intere.
- *[x.x]*: Campo numerico con n cifre intere e n cifre decimali (es: 123.45).
- *[n]*: Campo numerico, singola cifra.
- *[nn]*: Campo numerico, due cifre.
- *[nnnn]*: Campo numerico, quattro cifre.

Nota:

- Riferimenti a gradi si intendono sessagesimali.
- Riferimenti a UTC si intendono relativi all'ora universale, praticamente GMT, detto anche ora Zulu.

In effetti ci sono piccoli scostamenti (costanti e accuratamente monitorati) tra l'ora interna del sistema GPS e l'ora GMT, normalmente usata per gli orologi, attualmente intorno ai 13 secondi.

1.7.5 Checksum

Al termine di ciascuna frase NMEA è posto, dopo un asterisco, un checksum per poter individuare eventuali errori sulla linea di trasmissione tra emettitore e ascoltatore (talker e listener). Con le attuali tecnologie, una trasmissione a 4800 bps su una interfaccia RS232 è, a dir poco, affidabile, ma qualora la trasmissione avvenisse mediante un sistema più complesso (infrarosso, Bluetooth) o una catena più lunga (come nel campo della localizzazione veicolare, magari su GSM/GPRS o ponti radio) il checksum può essere utile per assicurare l'integrità della trasmissione. Il checksum è uno XOR a 8-bit (senza i bit di start o stop) di tutti i caratteri della frase NMEA, compresi i delimitatori ,, tra i delimitatori $ e * esclusi. Il valore esadecimale dei 4 bit più alti e bassi del risultato sono convertiti in due caratteri ASCII (0-9, A-F). Il carattere trasmesso per primo è quello più significativo. Questo checksum è eseguito di default dal ricevitore GPS.

1.8 File XML

Il linguaggio XML (eXtensible Markup Language) è stato introdotto per colmare la mancanza di uno standard che consenta ricerche intelligenti, scambio di dati, presentazioni adattabili e personalizzazioni.
Lo standard XML è un formato testuale, simile per molti aspetti al linguaggio HTML, ideato specificamente per memorizzare e trasmettere dati.
Internet non si può limitare a fornire l'accesso ad informazioni e fissare standard di visualizzazione, ma deve fissare uno standard per la comprensione delle informazioni, cosicchè i programmi software siano in grado di eseguire meglio le ricerche, di visualizzare e di manipolare informazioni nascoste in file di testo.
L' XML si descrive da solo, il che significa che comprende sia la struttura che la semantica dei dati contenuti nel file.
Il linguaggio XML fornisce una rappresentazione strutturale (struttura ad albero) dei dati che può essere ampiamente e facilmente implementata. Lo standard XML è costituito da un sottoinsieme di SGML ottimizzati per l' invio nel Web. Definito dal consorzio W3C (World Wide Web Consortium), il linguaggio XML assicura che i dati strutturati siano uniformi e indipendenti dalle applicazioni o dai produttori.
L'inizio di un file XML è costituito da una tag iniziale *(<title>)*, una tag finale *(</title>)* e da informazioni, comprese tra le due tag.
Diversamente dal linguaggio HTML, lo standard XML consente di includere un numero illimitato di tag, ciascuna indicante non l' aspetto, ma il significato di un elemento testuale. È compito dell' autore del documento stabilire il tipo di dati da utilizzare e i nomi delle tag che meglio si adattano.
Il linguaggio XML consente di utilizzare fogli di stile quali XSL (eXtensible Style Language), CSS (Cascading Style Sheets) e altre applicazioni per la visualizzazione ed elaborazione dei dati. Questo procedimento permette al linguaggio XML di poter tenere separati i dati dalle altre applicazioni di rappresentazione, permettendo così una perfetta integrazione dei dati provenienti anche da più fonti.
Per rendere un documento XML più completo solitamente si allega ad esso un file DTD (Document Type Definition), che definisce le regole utilizzate, quali ad esempio l'indicazione degli elementi presenti e la relazione strutturale esistente tra essi.
Anche se il linguaggio XML è totalmente personalizzabile esistono comunque delle regoli di sintassi.
Le più importanti sono:

- Tutti i file XML devono iniziare con la dichiarazione standard che specifica la versione XML e il set di caratteri utilizzati nel documento. Esempio 3: *<?xml version=1.0 encoding=iso-8859-1?>*.
- Tutti i documenti XML devono avere un elemento principale (root).
- Tutti gli elementi XML devono avere un tag di chiusura.
- I tag sono case sensitive.
- Tutti gli elementi XML devono essere nidificati correttamente.
- Gli attributi devono essere inclusi nel tag di apertura e devono essere scritti tra doppie virgolette.

I principali vantaggi dell' XML sono visibili in alcune applicazioni:

- Nelle applicazioni che manipolano una grossa quantità di dati, poichè resta in ogni modo flessibile ed estensibile.
- Nelle applicazioni in cui esiste un rischio di ridondanza dei contenuti.
- Qualora sia necessario uno scambio di una mole di dati elevata su piattaforme diverse.
- Nelle applicazioni in cui le informazioni siano facilmente disponibili per una grande quantità di client.

Naturalmente l' XML presenta anche dei limiti, questo non deve preoccupare poichè è un linguaggio in continua evoluzione. I principali svantaggi sono:

- Mancano trigger e indici per ricerche veloci.
- Non è possibile la Multi-utenza.
- Le funzioni di archiviazione non sono efficienti.
- Non esistono controlli sull'integrità dei dati.
- Non è implementato un controllo sulla sicurezza delle transizioni.
- Non sono supportate interrogazioni tra più documenti.

- è un linguaggio prolisso perchè richiede una coppia di tag per ciascun elemento.

A causa di queste lacune la ricerca nei file XML risulta lenta, ma serve comunque precisare che questo standard non è un linguaggio ideato per database. In questa applicazione riveste un ruolo molto importante perchè verrà usato come database di informazioni turistiche.

Capitolo 2

Progettazione e sviluppo dell'applicazione

In questo capitolo presenteremo l' analisi, l'architettura del' applicazione, e i dettagli della sua implementazione. Verranno descritte le principali classi che la compongono e la loro relazione.

2.1 Analisi dei requisti

Il sistema analizzato consiste in due parti logiche, quella relativa all'utente denominata Front-End e quelal relativa alla connessione GPS, denominata Back-End; questa applicazione richiede in realtà anche una parte server per la costruzione del database delle informazioni, ma viene trattata in un'altra tesi.
Il sistema è stato progettato tenendo conto delle esigenze di chi lo utilizzerà, quindi cercando di rendere semplice e intuitiva l'interfaccia e il processo che porterà alla visualizzazione delle informazioni desiderate.

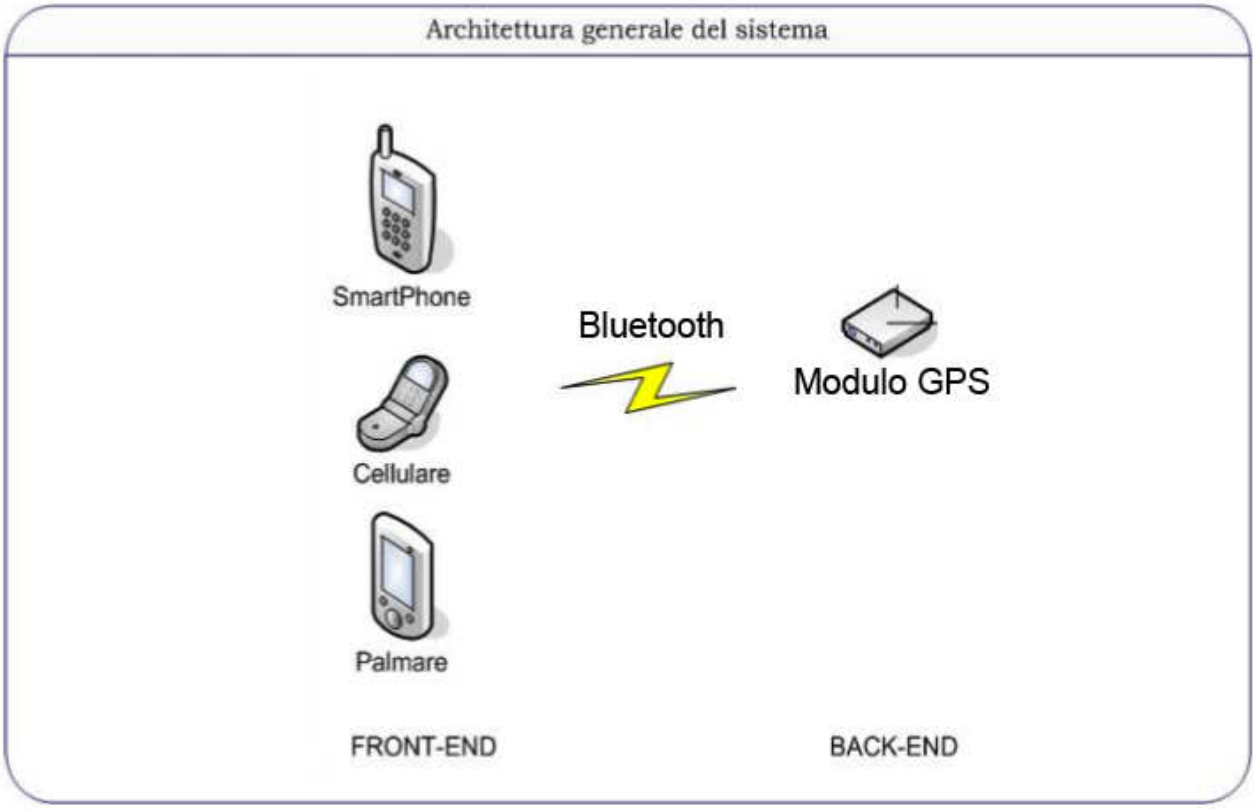

Fig. 2.1: Architettura generale del sistema

2.1.1 Sottosistema Front-End - Requisiti Funzionali

Lo scopo dell'applicazione Midlet da sviluppare è informare l'utente di determinate informazioni di tipo turistico. Tal informazioni saranno nello specifico i tag, cioè l'identificatore della tipologia del Punto di Interesse, i Punti di Interesse e gli articoli, gli elementi che compongono i punti di interesse, cioè quelli che danno informazioni specifiche su di essi.
L'utente deve poter usare l'applicazione in due casi:

- Nel caso in cui abbia la possibilità di connettersi al GPS; in questo caso viene denominata modalità ON-LINE.
- Nel caso in cui non abbia la possibilità di connettersi al GPS; in questo caso viene denominata modalità OFF-LINE.

È stata prevista la possibilità di ricercare i punti desiderati tramite la selezione dei tag e la selezione della distanza di ricerca nel caso in cui l'utente si trovi nella modalità ON-LINE, oppure solamente tramite la selezione dei tag nel caso in cui l'utente si trovi nella modalità OFF-LINE.
Quando l'utente cerca di connettersi al ricevitore GPS Bluetooth, non avendo a disposizione il suo ID (identificativo del dispositivo fisico, indispensabile nei collegamenti BT), deve avere la possibilità di farlo in maniera semplice e intuitiva; è previsto infatti che l'applicazione debba visualizzare tutti i dispositivi Bluetooth nelle vicinanze e sia in seguito l'utente a selezionare quello desiderato.
In caso di mancata connessione o nel caso in cui non venga visualizzato nessun dispositivo, l'utente deve poter riavviare la connessione senza dover avviare nuovamente l'applicazione.
Nel momento in cui riceve le coordinate e avvia la ricerca deve essere informato da un'alert nel caso in cui non ci sia un segnale Bluetooth oppure non ci siano Punti di Interesse con quelle caratteristiche nel raggio di ricerca impostato; in caso contrario l'utente deve poter selezionare i punti di interesse e visualizzare i suoi articoli, contenenti testo e immagini. Dopo ogni scelta deve poter tornare indietro di un passo alla volta e modficare le proprie decisioni.

N°	REQUISITI FUNZIONALI
1	L'utente deve poter selezionare la modalità di funzionamento scegliendo tra OFF-LINE (nel caso in cui non ci sia segnale GPS oppure voglia semplicemente visualizzare una panoramica dei Punti di Interesse) e modalità ON-LINE per la ricerca automatica dei Punti di Interesse desiderati
2	L'utente deve poter selezionare i tag di proprio interesse
3	L'utente deve poter impostare la distanza di ricerca dei Punti di Interesse
4	L'utente deve poter scegliere il dispositivo Bluetooth al quale collegarsi
5	Nel caso in cui la ricerca del dispositivo Bluetooth non vada a buon fine l'applicazione deve permettere l'avvio di una nuova ricerca senza dover riavviare l'applicazione
6	Nel caso in cui la connessione satellitare non vada a buon fine l'applicazione deve permettere l'avvio di una nuova connessione senza dover essere riavviata
7	Nel caso in cui non vi sia segnale GPS o Bluetooth l'applicazione deve avvisare l'utente tramite messaggio d'errore.
8	L'utente deve poter selezionare i punti di interesse e i loro articoli in modo da visualizzare il testo e le immagini

Fig. 2.2: Requisiti Funzionali

2.1.2 Sottosistema Front-End - Requisiti Non Funzionali

Il programma non è rivolto ad una particolare fascia d'utenza, ma è ideato per essere accessibile a chiunque possieda un cellulare o PDA o Smartphone. Il dispositivo in questione deve implementare una KVM, cioè una Java virtual Machine per la tecnologia J2ME, visto che è stato implementato usando questo linguaggio.
Il sistema dovrà essere di facile utilizzo da parte di chi non ha grande dimestichezza con i dispositivi mobili, quindi dovrà essere intuitivo nel settaggio delle impostazioni, nella ricerca dei dispositivi Bluetooth, e nella ricerca e visualizzazione dei Punti di Interesse.

Perchè l'utente possa usare completamente l'applicazione, il dispositivo dovrà essere munito di collegamento Bluetooth, e di un ricevitore GPS con lo stesso tipo di periferica, altrimenti potrà usare l'applicazione solo in modalità OFF-LINE.

N°	REQUISITI NON FUNZIONALI
1	L'applicazione deve essere sviluppata con ambiente di sviluppo J2ME, per garantire un'ampia compatibilità
2	Il software deve essere accessibile a dispositivi mobili (cellulari e PDA), che supportano la piattaforma Java Micro Edition
3	I dispositivi in gioco devono essere provvisti di connessione Bluetooth; in mancanza di questa l'applicazione può funzionare esclusivamente in modalità OFF-LINE
4	L'applicazione dovrà essere semplice da utilizzare e non dovrà richiedere particolare dimestichezza con i dispositivi mobili

Fig. 2.3: Requisiti Non Funzionali

2.1.3 Sottosistema Back-End - Requisiti Funzionali

Questa parte comprende il file xml contenente i dati che l'applicazione andrà a leggere e il sistema GPS; In quest'ultimo caso non è possibile entrare nel merito del sistema, in quanto si tratta di una tecnologia proprietaria già implementata e funzionante alla quale l'applicazione sviluppata si appoggia.
Gli unici due requisiti che deve avere questo sistema (nello specifico il dispositivo ricevitore) sono:

- Deve essere un dispositivo esterno al quale ci si deve poter collegare tramite segnale Bluetooth.
- Deve supportare lo standard ASCII NMEA.

Infatti l'applicazione è progettata per supportare solamente dispositivi esterni; tra gli sviluppi futuri è compreso sicuramente un'estensione ai dispositivi con chip GPS integrato. Il file XML invece deve essere formattato in una determinata maniera, visto che la funzione che lo parserizza è costruita su misura per un certo tipo di formattazione; dovrà contenere i seguenti tag: *<pi>*, *<tag>*, *<lat>*, *<long>*, *<articolo>*, *<img>*; altri dettagli in merito saranno esposti in seguito, nella sezione in cui si descrive l'implementazione.

N°	REQUISITI FUNZIONALI
1	Il file XML deve possedere un struttura ad hoc, indispensabile per la corretta gestione dei dati
2	I file XML deve contenere determinati tag
3	Il ricevitore deve supportare lo standard NMEA

Fig. 2.4: Requisiti Funzionali

2.1.4 Sottosistema Back-End - Requisiti Non Funzionali

Il file XML deve essere creato da un'apposita applicazione server, sviluppata appositamente per questo uso e funzionante su piattaforma PC. Il ricevitore GPS, come tutti gli altri dispositivi in gioco deve essere dotato di connessione Bluetooth, altrimenti non sarà possibile usufruire di tutte le funzionalità dell'applicazione. Al fine di una migliore ricezione del segnale anche in ambienti poco adatti a questo tipo di uso, il ricevitore dovrebbe essere dotato quantomeno di un chip di tipo Sirf3 a 44 canali, attualmente l'ultimo ritrovato tecnologico in fatto di prestazioni e capacità di ricezione.

N°	REQUISITI NON FUNZIONALI
1	Il file XML deve essere creato da un'applicazione esterna sviluppata ad hoc per questo tipo di uso
2	Il ricevitore deve essere provvisto di connessione Bluetooth

Fig. 2.5: Requisiti Non Funzionali

Riassumendo, la situazione delle possibilità concesse all' utente è sostanzialmente quella descritta in Figura 3.1:

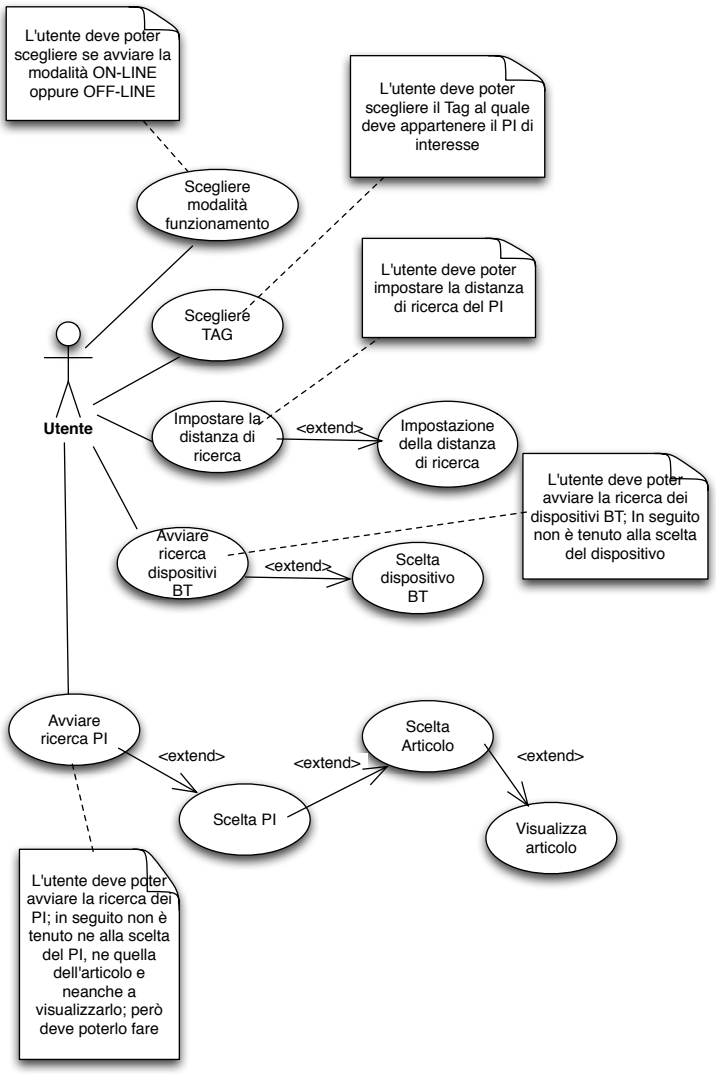

Fig. 2.6: USE CASE Diagram

2.2 Strumenti di sviluppo

La scelta è ricaduta su due strumenti di sviluppo principali (per quanto riguarda lo sviluppo del codice), molto famosi in ambito di programmazione

per dispositivi mobili (e non solo):

- Il linguaggio di programmazione J2ME, (la cui descrizione più dettagliata si può trovare nel capitolo 3); la motivazione principale, come già detto, è stata la portabilità di questo linguaggio, cosa di cui le altre alternative più ambite, come il linguaggio C, non sono caratterizzate, perchè legate al Sistema Operativo Symbian.

- Netbeans 6.1, l'IDE di sviluppo della Sun, progettato per la programmazione Java (e non solo), nel quale è stato incorporato il WTK (Wireless ToolKit), plugin appositamente creato per la programmazione J2ME, che permette all'IDE di creare progetti con le impostazioni di base predefinite utili alla creazione di un'applicazione MIdlet.

Mentre i due supporti hardware usati per il test sono stati i seguenti:

- NOKIA 6630, SmartPhone della nokia dotato di sistema operativo Symbian ma con una KVM incorporata, così da poter testare l'effettivo funzionamento in tempo reale dell'applicazione.

- Ricevitore GPS Bluetooth Cellular Line, sirf3 a 44 canali

Il WTK incorpora un emulatore, ma non è stato possibile usufruirne perchè non c'è la possibilità di farlo comunicare, tramite connessione Bluetooth, con un dispositivo esterno. f

2.3 Funzionamento dell'applicazione

2.3.1 Possibili Scenari di utilizzo

L'applicazione midlet qui sviluppata può essere considerata una versione client, supportata da una versione Server su piattaforma PC, la cui realizzazione viene discussa in un'altra tesi.
L'utilizzo del GPS come strumento principale, pone di fatto l'esigenza di usare l'applicazione in aree aperte, possibilmente evitando l'avvio del ricevitore in zone coperte, o circondate da edifici di una certa grandezza (come può esserlo ad esempio un grattacielo) o in zone coperte da vegetazione (infatti il segnale proveniente dai satelliti ha una potenza di circa 50 Watt, basterebbero anche pochi alberi per ostacolarne il passaggio); il problema non si pone più dal momento in cui avviene un allineamento tra il ricevitore e i satelliti, in quanto una volta tornati in zona aperta, dopo pochi secondi la connessione viene ripristinata.

Fig. 2.7: Schermata iniziale dell'applicazione

Quindi non è indicata per l'uso all'interno degli edifici, a meno che non si voglia usare in modalità OFF-LINE (Verrà descritto in seguito il suo funzionamento).
Il suo uso può essere ricondotto a due scenari tipici:

Scenario 1: Supponiamo che un turista voglia visitare una città, evitando di portarsi appresso una guida turistica su supporto cartaceo, un libro ad esempio; se possiede un cellulare, o uno smartphone può usare l'applicazione come guida turistica elettronica.
L'applicazione può essere scaricata dalla rete internet, oppure più semplicemente acquistata assieme alle memory card sulle quali vengono memorizzati i database presso l'agenzia turistica locale.
Il turista non deve far altro che avviare la ricerca dei dispositivi BT, selezionare quello desiderato (cioè il GPS) e dopo aver stabilito una connessione satellitare, può avviare la ricerca dei PI nel raggio impostato precedentemente con l'apposita funzione.

Scenario 2: Lo stesso turista, ad un certo punto della sua visita si trova all'interno della chiesa che vuole visitare ma vuole conoscere qualcosa di più sulla sua storia.

2.3.2 Funzionamento

Il suo funzionamento risulta all'atto pratico molto semplice e intuitivo. L'utente inserisce la memory card caricata del file xml contenente tutti i dati relativi alla città che intende visitare, come ad esempio musei, ristoranti,

pub, cinema, ecc, referenziati tutti tramite tag, come ad esempio cultura, intrattenimento, ecc, e le cartelle con i relativi file immagine (se ce ne sono); Tutti questi luoghi, chiamati nel file *pi*, possiedono degli articoli, che possono essere l'indirizzo, piuttosto che una descrizione del luogo o del monumento o dell'edificio, e così via; ogni articolo può avere oppure no delle immagini: nel file xml sono memorizzati solo i loro nomi, mentre i file, in formato jpeg, sono salvati in apposite cartelle, con i nomi dei Punti di interesse di appartenenza; il tutto memorizzato appunto nella memory card.
Come esempio pratico a supporto degli scenari di utilizzo precedentemente descritti, prendiamo la città di Trento (città nella quale sono stati eseguiti i test):

Scenario 1: il turista si trova in piazza Duomo e, non conoscendo niente della città, vuole sapere se c'è qualche edifico storico nelle vicinanze. Dopo aver avviato l'applicazione e la connessione GPS, non deve fare altro che selezionare il tag *chiese*, impostare il raggio di ricerca, ad esempio, a 100 metri, ed ecco che l'applicazione visualizzerà tutto ciò che si trova nel raggio di 100 metri catalogato con il tag *chiese*. Tra i vari PI, apparirà sicuramente quello relativo al Duomo (questo perchè nel file xml il Duomo è catalogato con questo tag).

Scenario 2: Lo stesso turista vuole visitare l'interno del Duomo; a questo punto se vuole avere più informazioni riguardo a questo Punto di Interesse non deve fare altro che avviare l'applicazione in modalità OFF-LINE (infatti all'interno degli edifici il ricevitore GPS è inutilizzabile) e agire esattamente come nel primo scenario, tralasciando i due passaggi in cui imposta la distanza di ricerca e in cui imposta il ricevitore GPS.

Ecco un semplice esempio di come è strutturato il file xml che funge da Database per l'applicazione:

```
<punti>
<pi>
<nome>Studio Andromeda</nome>
<coordinate>
<lat>46.068354</lat>
<lon>11.123467</lon>
</coordinate>
<tags>arte</tags>
<articoli>
<articolo>
```

```
<titolo>Cenni storici</titolo>
<testo>
testo dell'articolo</testo>
<immagini>
<img>andromeda1.jpg</img>
<img>andromeda2.jpg</img>
<img>andromeda3.jpeg</img>
</immagini>
</articolo>
<articolo>
<titolo>Personaggi</titolo>
<testo>
personaggi illustri come Michetti ci hanno visitati
</testo>
<immagini>
<img>andromeda5.jpg</img>
</immagini>
</articolo>
</articoli>
</pi>
</punti>
```

Come già detto, nella sezione $<img>$ ... $</img>$ non viene inserito direttamente il file immagine ma il suo nome; infatti il file si troverà nella cartella con path $/img/StudioAndromeda/nome-fil-img/$; quando dovrà essere visualizzato, il programma lo caricherà seguendo il precedente path. Di seguito viene descritto più nel dettaglio il funzionamento del sistema:

Avvio Applicazione All'avvio dell'applicazione, i dati vengono estrapolati dal documento XML mediante uno StringTokenizer costruito su misura; i dati sensibili vengono memorizzati in strutture dati dedicate, precisamente dei Vector, pronti per essere visualizzati al momento opportuno; non si fa uso di RMS, quindi alla chiusura dell' applicazione le strutture dati vengono resettate e riempite solamente al successivo riavvio.

Opzioni Dopo la canvas iniziale, appare una form di Benvenuto, dalla quale si può scegliere il comando opzioni. Dalla schermata delle opzioni si possono selezionare vari item:

Opzioni Tag in questa schermata l'utente può scegliere il tag al quale è interessato; non sono previsti tag predefiniti, ma il tutto è dinamico, cambiano a seconda di quello che viene letto dal documento xml.

Opzioni GPS in questa schermata è possibile fare la ricerca di dispositivi Bluetooth in generale; vengono visualizzati tutti i dispositivi trovati, poi sarà l'utente a selezionare quello corretto, che sarò appunto il ricevitore GPS.

Opzioni Distanza In questa schermata è possibile inserire la distanza di ricerca dei vari PI rispetto alle coordinate del punto in cui si trova il dispositivo; il valore è espresso in metri, la ricerca prende in consideraione il raggio dal punto suddetto.

Ricerca PI Con questo comando l'utente avvia la ricerca dei PI; verranno visualizzati tutti i PI presenti nel raggio di ricerca e corrispondenti al tag scelto precedentemente.

Visualizza Articoli Verranno visualizzati tutti gli articoli corrispondenti al PI selezionato.

Visualizza Articolo Verrà visualizzato il contenuto (testo e/o immagini) dell'articolo selezionato.

La modalità OFF-LINE è accessibile direttamente dalla schermata di Benvenuto, la dinamica è esattamente la stessa della modalità ON-LINE, solo che non vengono prese in considerazione le coordinate GPS.
Nel caso in cui non vi siano presenti PI, viene visualizzata un' Alert che informa di ciò l' utente. Stessa cosa nel caso non venga trovato un dispositivo Bluetooth, e nel caso non vi sia segnale dal satellite.

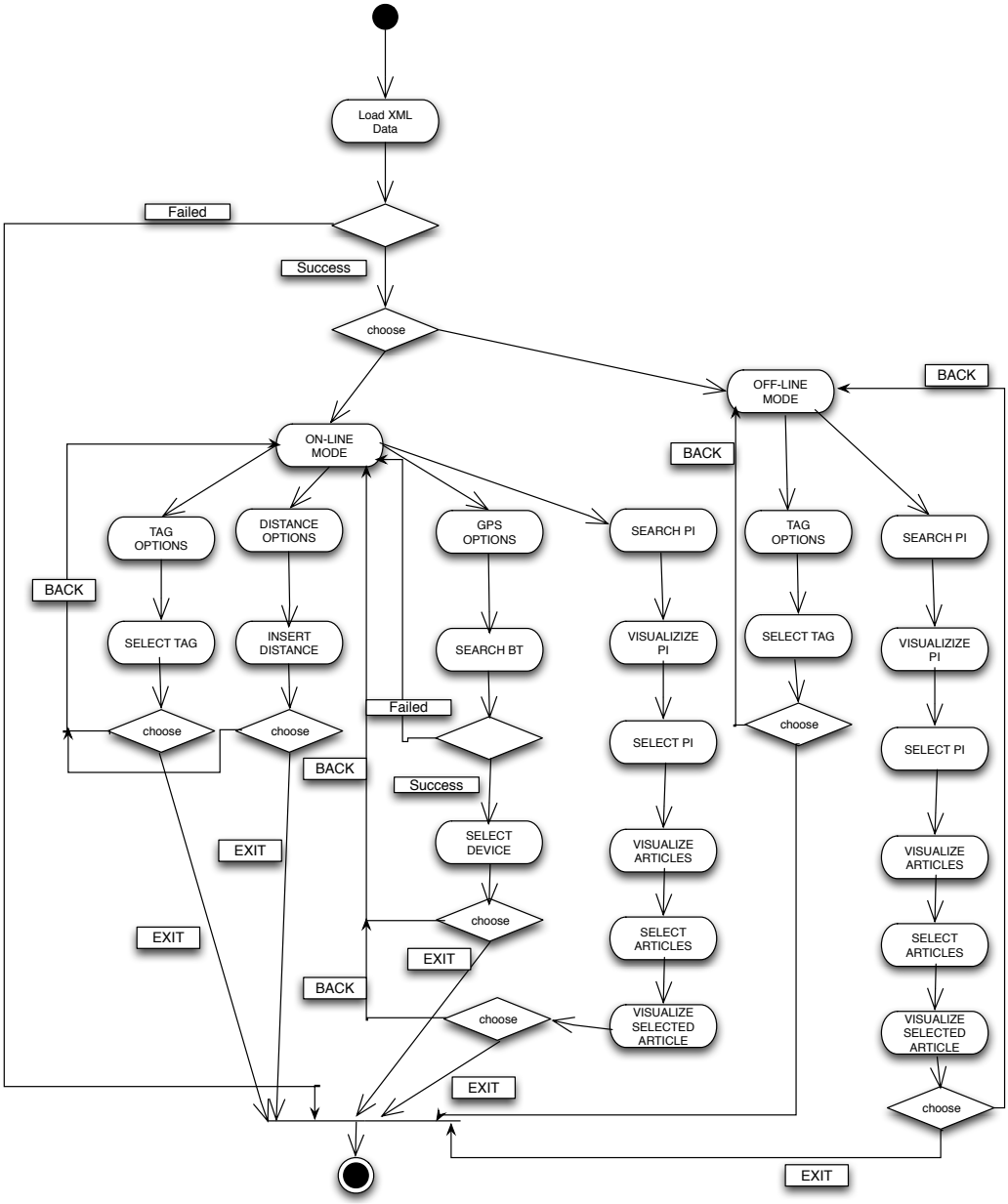

Fig. 2.8: Activity diagram dell'applicazione

2.4 Schema delle Classi

L'applicazione è composta da 13 classi, di seguito verranno descritte quelle principali contenenti i metodi di maggiore importanza per l'elaborazione dei dati. In più contiene un package esterno, mypapit.net.java, che comprende una libreria opensource per l'utilizzo della classe StringTokenizer, usata nel parsing del documento xml; trattasi di una trasposizione dei metodi della classe omonima presente in J2SE, essendo inesistente in J2ME come libreria nativa.
Queste classi contengono metodi e riferimenti alle tecnologie descritte nel capitolo 2; comune a tutte c'è naturalmente il linguaggio di programmazione J2ME, mentre specificherò per ognuna quale sono incluse nel codice.

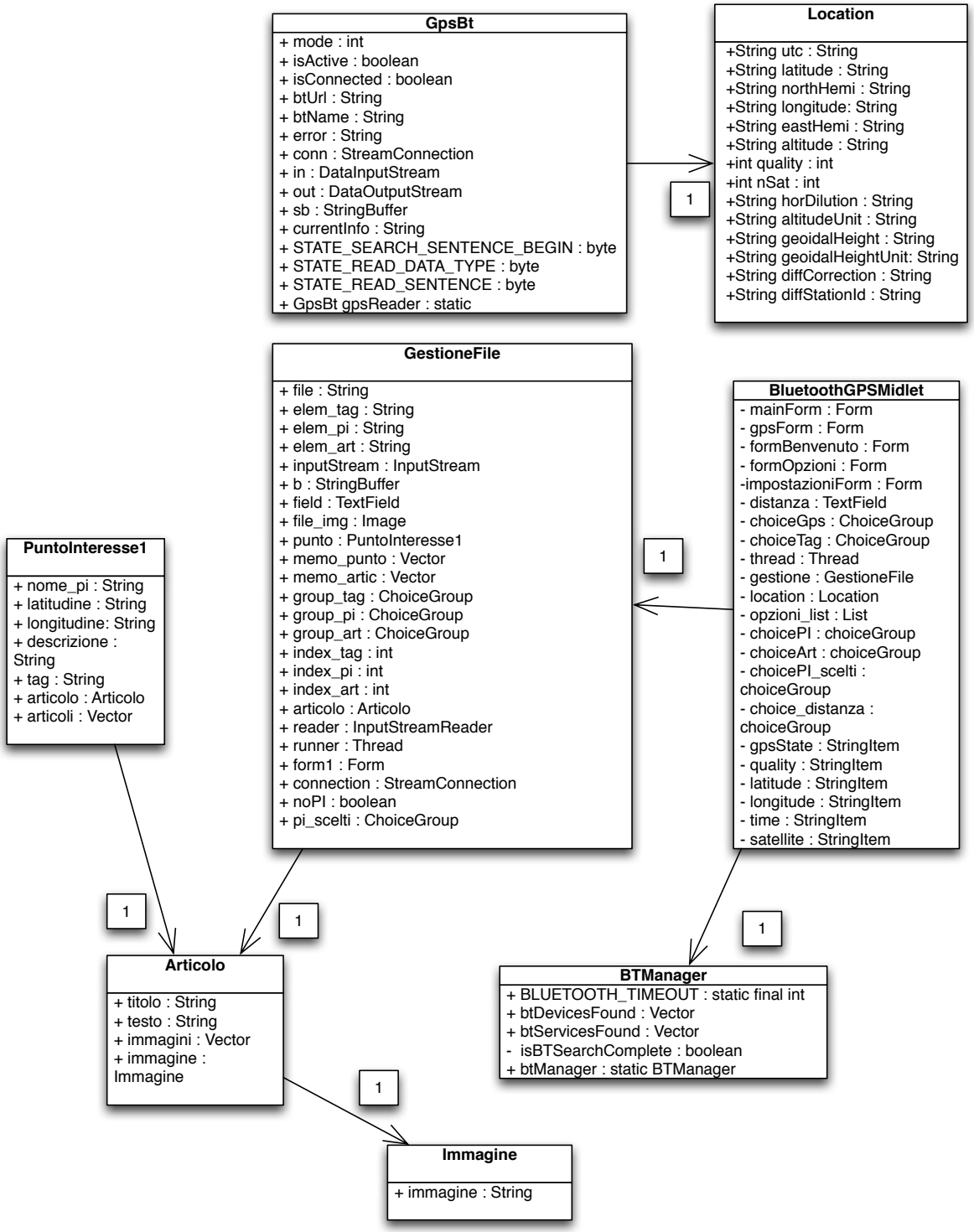

Fig. 2.9: Class Diagram

2.4.1 Classi per la gestione del GPS

Sono 3 le classi per la gestione del ricevitore GPS:

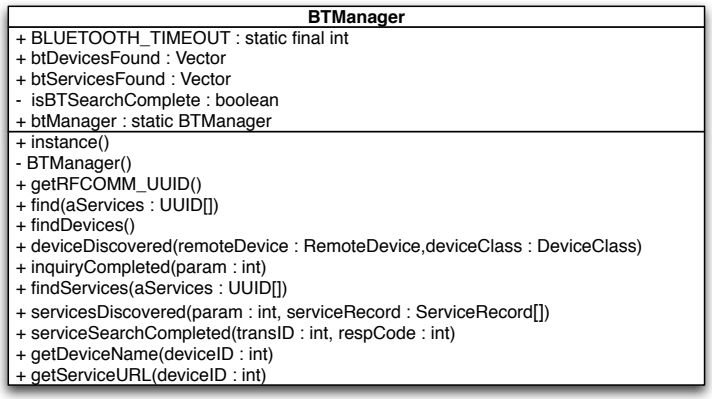

Fig. 2.10: classe BTManager

BTManager Questa classe contiene i metodi per ricercare qualsiasi dispositivo Bluetooth presente nelle vicinanze; ho optato per questo sistema perchè c'è la possibilità che l'utente non abbia a disposizione sempre lo stesso ricevitore GPS, in questa maniera vengono visualizzati sul display tutti i dispositivi bluetooth e l'utente potrà scegliere quello desiderato. Sostanzialmente crea un servizio Bluetooth sulla base di un URL. In questa classe si usa la tecnologia bluetooth, da vedere in particolare i riferimenti al protocollo RFCOMM.

```
public int findDevices() {
    try {
         btDevicesFound.removeAllElements();
      isBTSearchComplete = false;
      LocalDevice local = LocalDevice.getLocalDevice();
      DiscoveryAgent discoveryAgent = local.getDiscoveryAgent();
       discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
      while ((!isBTSearchComplete)) {
        synchronized (this) {
          this.wait(BTManager.BLUETOOTH_TIMEOUT);
```

```
        }
        if (!isBTSearchComplete) {
          discoveryAgent.cancelInquiry(this);
        }
      ...
          }
    return btDevicesFound.size();
  }
```

questo metodo avvia la ricerca dei dispositivi Bluetooth nel raggi d'azione del segnale; il metodo *startInquiry()* avvia la ricerca; finchè sta cercando non succede niente, poi quando trova un dispositivo, chiama il metodo *inquiryCompleted()*; successivamente viene avviata la ricerca dei servizi disponibili sul dispositivo trovato.
I due metodi di maggior interesse per l'applicazione sono i seguenti:

```
public String getDeviceName(int deviceID) {
    try {
      return ((RemoteDevice) btDevicesFound.elementAt(deviceID))
          .getFriendlyName(false);
    } catch (Exception e) {
      e.printStackTrace();
    }
    return "Error";
  }
```

e

```
public String getServiceURL(int deviceID) {
    try {
      return ((ServiceRecord) btServicesFound.elementAt(deviceID))
          .getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false)
    } catch (Exception e) {
      e.printStackTrace();
```

```
    }
    return "Error";
  }
```

Il primo, *getDeviceName()*, restituisce il nome del dispositivo, mentre il secondo, *getServiceURL* restituisce l'URL del dispositivo in formato RFCOMM; questo ha una notevole importanza, in quanto è grazie ad esso che il dispositivo mobile riuscirà a collegarsi al ricevitore GPS;

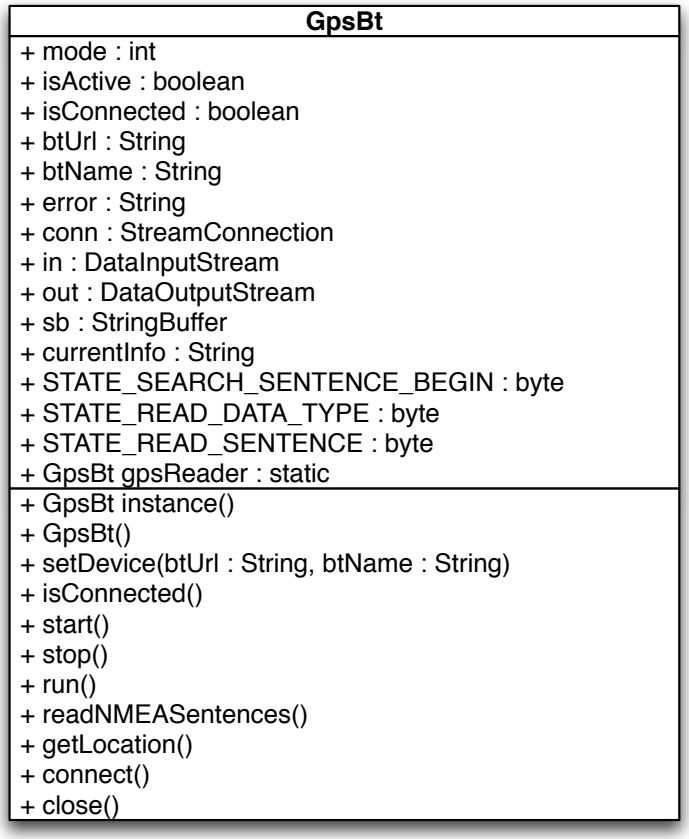

Fig. 2.11: classe GpsBt

GpsBt Questa classe contiene invece metodi più specifici per la comunicazione direttamente con il ricevitore GPS. In particolare il metodo *start* attiva un Thread, che stabilisce una comunicazione con il dispositivo locale il cui UUID è stato determinato precedentemente, tramite la selezione dal choicegroup da parte dell'utente del dispositivo, gestito dalla classe *BluetoothGPSMidlet* (graficamente) ed in background dalla classe descritta precedentemente (BTManager). In questa classe viene utilizzato il protocollo NMEA-0183 per l'estrazione delle informazioni relative alle coordinate geografiche.

Di seguito la spiegazione dei principali metodi:

```
public void run() {
    isActive = true;
    while (isActive) {
      try {
        if (!isConnected \&\& isActive) {
          connect();
        } else {
          readNMEASentences();
        }
     ...
      close();
    isActive = false;
  }
```

Il metodo *run*, attivato dalla chiamata del metodo *start()* avvia un Thread, che stabilisce una connessione bluetooth con il dispositivo selezionato, aprendo un canale *InputStream* tramite le funzioni $(StreamConnection)Connector.open(btUrl, Connector.READ\_WRITE)$ e *new DataInputStream(conn.openInputStream())* attraverso il quale viaggiano le informazioni sottoforma di byte. Il GPS attraverso questo canale invia una stringa, parsificata secondo lo standard NMEA da questo metodo:

```
public void readNMEASentences() {
    try {
      if (!isConnected) {
        return;
      }
      int size = in.available();
      if (size <= 0) {
        return;
      }
      for (int j = 0; j < size; j++) {
        int i = in.read();
        if (i != -1) {
          char l = (char) i;
          switch (mode) {
          case (STATE_SEARCH_SENTENCE_BEGIN): {
```

```
  if (l == '$') {
    mode = 1;
    sb.setLength(0);
  }
}
  break;
case (STATE_READ_DATA_TYPE): {
  sb.append(l);
  if (sb.length() == 6) {
    if (sb.toString().startsWith("GPGGA")) {
      mode = STATE_READ_SENTENCE;
      sb.setLength(0);
    } else {
      mode = STATE_SEARCH_SENTENCE_BEGIN;
      sb.setLength(0);
    }
  }
}
  break;
case (STATE_READ_SENTENCE): {
  sb.append(l);
  if ((l == 13) || (l == 10) || (l == '\$')) {
    mode = STATE_SEARCH_SENTENCE_BEGIN;
    synchronized (this) {
      currentInfo = new String(sb.toString());
    }
    Thread.sleep(1000);
  ....
    }
```

Nello specifico legge i dati nel pezzo di stringa compresa tra il prefisso GPGGA e i successivi 13 caratteri; in questa parte infatti sono presenti i dati relativi alle coordinate geografiche, latitudine e longitudine, assieme ad altre informazioni, quali la data, l'ora, il numero di satelliti al quale è connesso, l'indicazione dell'emisfero in cui si trova.

Location Questa classe contiene un solo metodo, per la parsificazione della stringa ottenuta dalla comuncicazione con il ricevitore GPS. In questo caso, viene tokenizzata la stringa di tipo $GPPGA, ottenendo i dati sensibili secondo le specifiche dello standard NMEA-0183. In questa

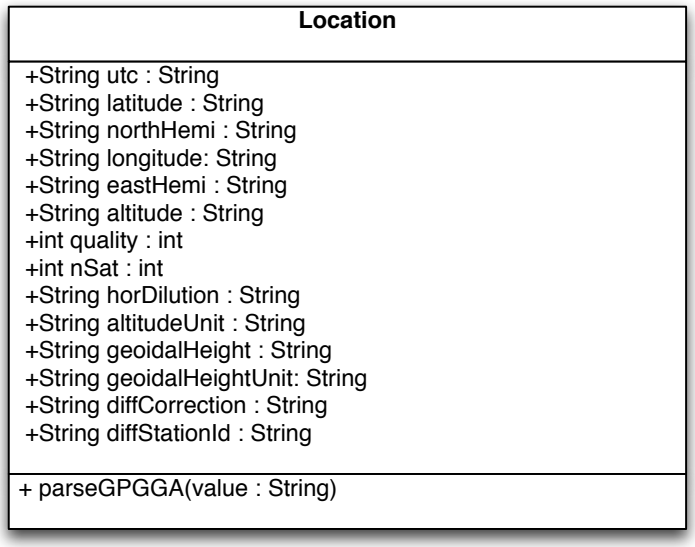

Fig. 2.12: classe Location

classe, oltre alle specifiche NMEA, viene usata anche una libreria esterna, chiamata *mypapit.java*, per l'uso della classe *StringTokenizer*; è una trasposizione delle omonime classi J2SE come libreria J2ME.

```
public void parseGPGGA(String value) {

    StringTokenizer tok = new StringTokenizer(value, ",");

    utc = tok.nextToken();
    latitude = tok.nextToken();
    northHemi = tok.nextToken();
    longitude = tok.nextToken();
    eastHemi = tok.nextToken();
    quality = Integer.parseInt(tok.nextToken());
    nSat = Integer.parseInt(tok.nextToken());
    horDilution = tok.nextToken();
    altitude = tok.nextToken();
    altitudeUnit = tok.nextToken();
    geoidalHeight = tok.nextToken();
    geoidalHeightUnit = tok.nextToken();
```

```
    diffCorrection = tok.nextToken();
    diffStationId = tok.nextToken();
  }
```

Nello specifico *value* è la stringa ottenuta dalla parer precedente.

2.4.2 La classe GestioneFile

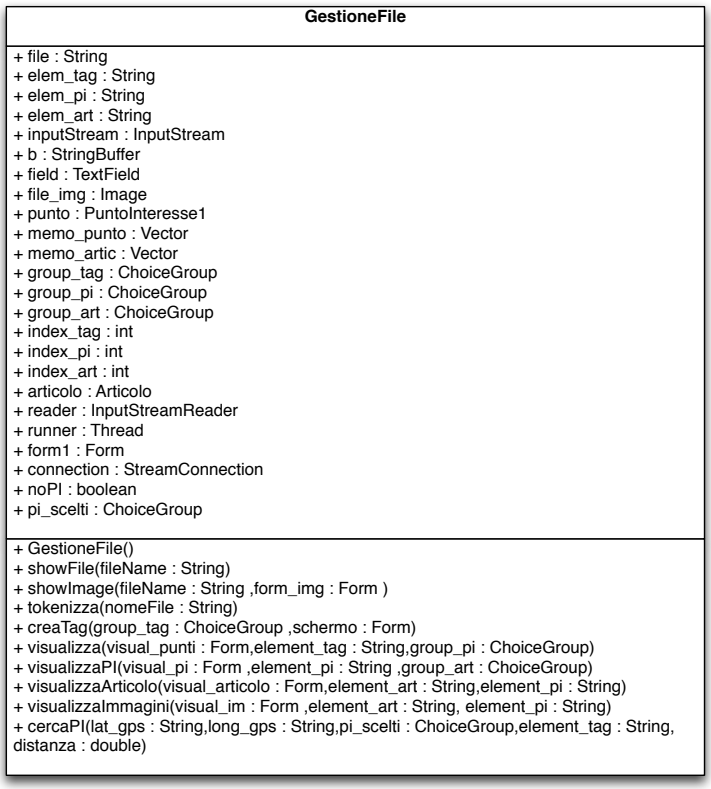

Fig. 2.13: classe GestioneFile

Questa è una delle classi più importanti dell'applicazione, in quanto contiene tutti i metodi per la gestione del file xml e dei dati in esso contenuti e anche quelli per la gestione delle coordinate geografiche. Per dettagli sulla tecnologia GPS vedere il paragrafo 2.6, in particolare la sezione 2.6.4. Di seguito elenco i metodi più importanti:

showFile() Si occupa della gestione del file xml, viene chiamato all'apertura dell'applicazione.

showImage() Si occupa della gestione delle immagini, viene chiamato solitamente quando si deve visualizzare un articolo (se questo contiene un immagine)

tokenizza() Questo è uno dei metodi principali, in quanto si occupa di parsificare il file xml per ottenere tutti i dati relativi ai PI, come il

Nome, il Tag, le sue coordinate geografiche, gli articoli ad esso correlati e relative immagini. La memorizzazione di queste informazioni viene gestita in maniera particolare, facendo uso di oggetti e strutture dati Vector, ma questo verrà spiegato in seguito.

visualizza() Questo metodo si occupa della ricerca e visualizzazione dei PI in un choicegroup, a seconda di quale Tag è stato scelto dall'utente; viene chiamato nel caso si utilizzi l'applicazione in modalità OFF-LINE.

visualizzaPI() Questo metodo si occupa della ricerca e visualizzazione di tutti gli articoli relativi al PI selezionato dal choiceGroup precedente.

visualizzaArticolo() Questo metodo si occupa della visualizzazione del testo dell'articolo selezionato.

visualizzaImmagini() Questo metodo si occupa della visualizzazione delle immagini relative all'articolo selezionato.

cercaPI() Questo metodo viene chiamato quando si avvia la ricerca del/dei PI che si trovano nella zona desiderata; da notare che contiene anche la funzione che calcola la distanza tra due punti geografici, utilizzando un particolare algoritmo, descritto nella sezione 3.5.4. La ricerca avviene considerando il Tag scelto e la distanza di ricerca immessa dall'utente nelle impostazioni, e naturalmente le coordinate geografiche del punto in cui si trova in quel momento; se la distanza tra le coordinate geografiche del PI segnate sul file xml e quelle ottenute dal ricevitore GPS è minore o uguale alla distanza di ricerca impostata, allora l'applicazione visualizza i punti di interesse relativi, altrimenti compare un'alert che informa l'utente che non sono presenti PI nel raggio desiderato.

Di particolare importanza è il seguente metodo:

```
distanza(double lat1, double longit1, double lat2, double longit2) {
  double distance = 0;
  lat1 = lat1 * Math.PI / 180;
  longit1 = longit1 * Math.PI / 180;
  lat2 = lat2 * Math.PI / 180;
  longit2 = longit2 * Math.PI / 180;
  dist_long = longit2 - longit1;
  dist_lat = lat2 - lat1;
  pezzo1 = Math.cos(lat2) * Math.sin(dist_long);
  pezzo11 = pezzo1 * pezzo1;
```

```
pezzo2 = Math.cos(lat1)*Math.sin(lat2)-Math.sin(lat1)*
 Math.cos(lat2)*Math.cos(dist_long);
pezzo22 = pezzo2 * pezzo2;
pezzo3 = Math.sin(lat1)*Math.sin(lat2)+Math.cos(lat1)*
 Math.cos(lat2)*Math.cos(dist_long);
pezzo4 = Float11.atan((Math.sqrt(pezzo11 + pezzo22)) / pezzo3);
distance = pezzo4 * 6372;
return distance;
  }
```

Questo metodo calcola la distanza in km tra due coordinate geografiche; le coordinate ottenute dal GPS e quelle fornite sul file xml sono in due formati diversi, quindi le prime 4 righe di codice hanno il compito di convertire quelle ottenute dal GPS nello stesso formato, così che l'algoritmo possa eseguire i calcoli correttamente.

2.4.3 La classe PuntoInteresse1

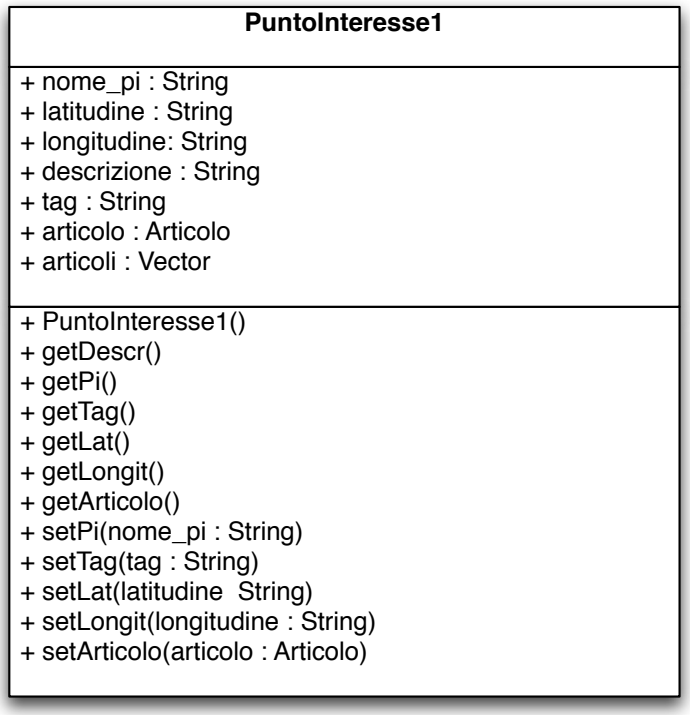

Fig. 2.14: classe PuntoInteresse1

Questa classe contiene solo metodi setter e getter, vengono chiamati nel metodo *tokenizza* decscritto nella sezione precedente, per memorizzare i dati in maniera opportuna; da sottolineare la presenza del metodo *setArticolo()*, il quale setta non una variabile semplice ma un oggetto di tipo *Articolo* (che verrà descritto in seguito); la scelta di questa procedura è dovuta al fatto che ogni Articolo può contenere più di una variabile, quindi la cosa più ovvia che mi è venuta in mente è stata quella di creare un oggetto Articolo con i suoi metodi. Ogni oggetto di tipo Articolo viene poi inserito in un Vector.

```
public class PuntoInteresse1 {

   public String nome_pi;
   public String latitudine, longitudine;
   public String descrizione;
```

```
 public String tag;
 public Articolo articolo;
 public Vector articoli = new Vector();
 public String immagine;

 public PuntoInteresse1() {
 }
 public String getDescr() {
     return descrizione;
 }
 public String getPi() {
     return nome_pi;
       }
....
public void setLongit(String longitudine) {
     this.longitudine = longitudine;
 }
 public void setArticolo(Articolo articolo) {
     this.articolo = articolo;
     articoli.addElement(this.articolo);
 }
}
```

2.4.4 La classe BluetoothGPSMIdlet

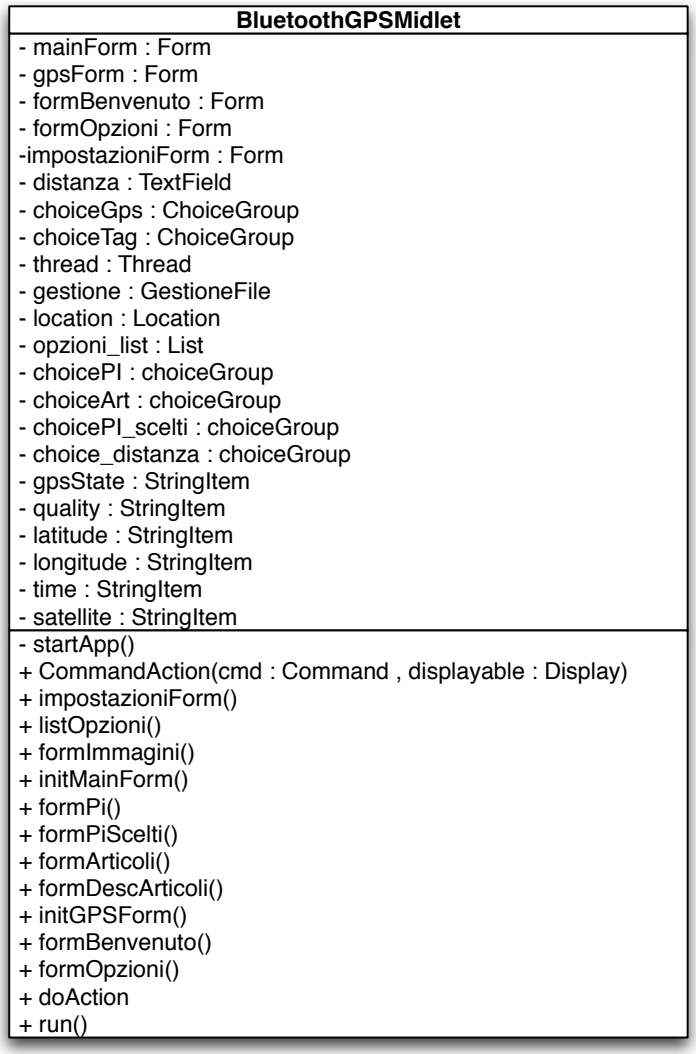

Fig. 2.15: classe BluetoothGPSMidlet

Questa classe contiene tutti i metodi per la parte grafica dell'applicazione, per la realizzazione quindi di tutte le form e i comandi delle form. In più contiene il metodo *doAction()*, che ha il compito di attivare un Thread per la comunicazione con il ricevitore GPS per la trasmissione al terminale dei dati relativi alle coordinate geografiche. Il thread in questione è sempre attivo, dal momento della sua accensione fino alla chiusura del'applicazione.

2.4.5 La classe Articolo

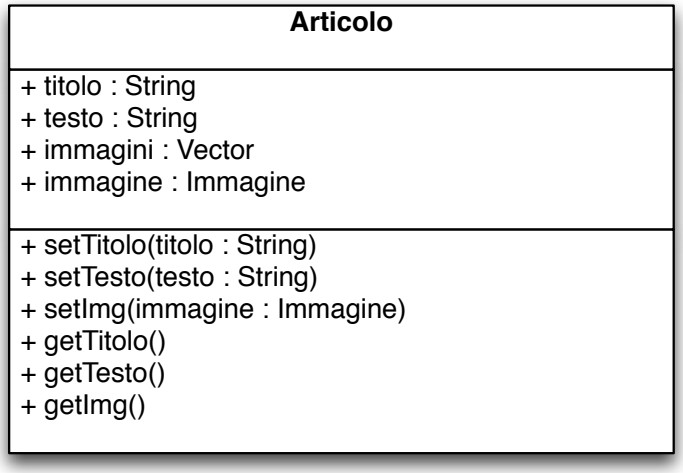

Fig. 2.16: classe Articolo

Come la classe PuntoInteresse1, contiene metodi setter e getter relativi a variabili che è previsto siano contenute in un articolo di un PI. Stesso discorso fatto per la classe PuntoInteresse1, Articolo contiene un metodo *setImg()* che salva le immagini (in questo caso il nome delle immagini) come un oggetto; scelta effettuata sempre per lo stesso motivo precedente, perchè un Articolo può contenere più di un'immagine. Ogno oggetto di tipo Immagine viene inserito in un Vector.

```
public class Articolo {

    public String titolo;
    public String testo;
    //public String immagine;
    public Vector immagini = new Vector();
    public Immagine immagine;

    public void setTitolo(String titolo) {
        this.titolo = titolo;
    }
    public void setTesto(String testo) {
```

```
        this.testo = testo;
    }
    public void setImg(Immagine immagine) {
        this.immagine = immagine;
        immagini.addElement(this.immagine);
    }
   .....
}
```

2.4.6 La classe Immagine

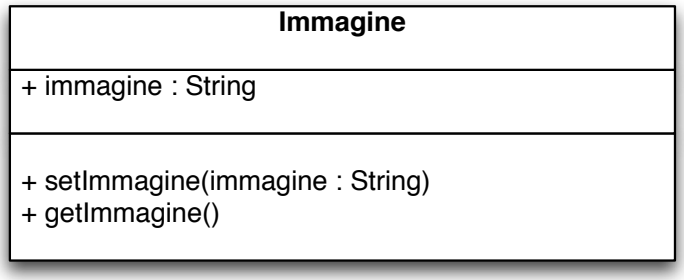

Fig. 2.17: classe Immagine

Questa classe contiene solo 2 metodi, *setImmagine* e *getImmagine*, per la memorizzazione del nome del file immagine che si dovrà cercare; Non è indispensabile indicare nel file xml l'intero path da seguire per trovare il file, perchè è impostato di default nel codice, in quanto ogni marca di dispositivi (a volte, per ogni marca anche modelli diversi) ha path diversi per quanto riguarda la memprizzazione dei file nel loro filesytem.

```
public class Immagine {

    String immagine;

    public void setImmagine(String immagine) {
        this.immagine = immagine;
    }
    public String getImmagine() {
      if (immagine == null)  immagine = "No Image";
        return immagine;
    }
}
```

2.4.7 Le classi Canvas

L'applicazione contiene alcune classi che estendono la classe Canvas e sono usate per comunicare un evento all'utente (come ad esempio la schermata

di errore che dice che non ci sono PI), o semplicemente usate all'avvio dell'applicazione (ad esempio i giochi in j2me per cellulari sono basati quasi esclusivamente sull'utilizzo di Canvas). La classe Canvas è una libreria che consente di creare un'interfaccia grafica a basso livello, dando come possibilità il disegno di forme geometriche bidimensionali semplici, come rettangoli , ellissi e l'importazione di immagini e scrittura di testi. Per contro, avere a disposizione solo strumenti grafici primitivi rende abbastanza difficoltoso creare interfacce complesse e gradevoli da usare. Comunque, in questa applicazione non verranno usate per rimpiazzare completamente la grafica del programma, ma solo per la schermata iniziale e per comunicare determinati eventi all'utente. In questa applicazione ce ne sono due:

La classe CanvasNoPI Questa classe contiene un Thread, che resta attivo per 5 secondi; viene chiamata quando la ricerca di PI in modalità ONLINE dà esito negativo.

La classe CanvasIniziale Anche questa classe contiene un Thread attivo per 5 secondi; viene chiamata all'avvio dell'applicazione e raffigura l'immagine di un castello, tanto per avere un buon impatto finale nell'uso del programma.

La classe CanvasNoBT Anche questa classe contiene un Thread attivo per 5 secondi; viene chiamata quando, durante la ricerca dei dispositivi Bluetooth, questa dà esito negativo.

Come esempio riporto il codice della classe *CanvasNoBt*, le altre due sono praticamente identiche a parte il messaggio di errore:

```
public class CanvasNoBT extends Canvas implements Runnable{

    BluetoothGPSMidlet canvas;

    public CanvasNoBT(BluetoothGPSMidlet canvas) {
        this.canvas = canvas;
        setFullScreenMode(true);
       Display.getDisplay(canvas).setCurrent(this);
    }

    protected void paint(Graphics g) {

        g.setColor(244,164,96);
        g.fillRect(0,0,g.getClipWidth(),g.getClipHeight());
```

```
        g.setColor(255,255,255);
        int i=Font.getDefaultFont().getHeight();
       g.drawString("no device found!",(g.getClipWidth()/2),
        (g.getClipHeight()/2)-i,g.BASELINE|g.HCENTER);
       g.drawString("controllare la",g.getClipWidth()/2,
        (g.getClipHeight()/2),g.BASELINE|g.HCENTER);
       g.drawString("connesione bluetooth",(g.getClipWidth()/2),
        (g.getClipHeight()/2)+i,g.BASELINE|g.HCENTER);
       try {
            g.drawImage(Image.createImage("/attention.png"),g.getClipWidth()/2,
             (g.getClipHeight()/2)+i,g.TOP|g.HCENTER);
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public void run() {
        repaint();
        serviceRepaints();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException ex) {
        }
        Display.getDisplay(canvas).setCurrent(canvas.initMainForm());
        }
}
```

Con la chiamata del metodo *repaint()* nel metodo *run()*, viene richiamato il metodo *paint()*, che ha il compito di ridisgnare la schermata secondo le specifiche della classe Graphics.

Capitolo 3

Test e Sviluppi futuri

In questo capitolo descriveremo i test eseguiti e i problemi riscontrati in quello che potrebbe essere un classico uso dell' applicazione.

3.1 Testing e problemi

Durante il testing dell'applicazione sono stati riscontrati alcuni problemi, dovuti più che altro dall'hardware utilizzato e meno dall'applicazione stessa; il programma è stato sviluppato principalmente per lo smartphone nokia 6630, con sistema operativo symbian; di seguito elenco i principali difetti trovati:

- L'applicazione deve leggere sempre file contenuti nella memory card del cellulare, quindi uno dei prerequisiti fondamentali è che il modello sia dotato di questo accessorio; al giorno d'oggi è cosa abbastanza normale averla in dotazione di serie, soprattutto nei modelli di fascia medio-alta.In generaleè indispensabile averla perchè è stato concepito per avere la possibilità di mettere in vendita memory card adibite solo a questo scopo; nessuno vieta di salvare i file appositi nella memoria interna del cellulare, ma questo prevede il rilascio di varie versioni dell'applicazione compatibili con i vari modelli (e/o marche) di cellulari, in quanto non tutti permettono l'accesso al proprio filesytem (vedi sezione 3.3)

- Il cellulare deve disporre delle librerie JSR-72 e JSR-85, ossia le API per la gestione del Bluetooth e del filesystem; questo purtroppo è possibile solo se il dispositivo è stato progettato per questo, in quanto sono impostazioni che vengono introdotte direttamente dalla casa madre durante la costruzione e messa a punto dello stesso.

- Nel caso di dispositivi con ricevitore GPS interno, invece si deve agire a livello di codice J2ME, in quanto è diverso l'approccio per un collegamento con un dispositivo Bluetooth esterno (infatti la connessione bluetooth è in generale unica per ogni dispositivo, poi i dati vengono elaborati dipendentemente da quello che lo sviluppatore vuole che il dispositivo faccia). Quindi anche in questo caso sarebbe opportuno il rilascio di varie versioni, cosa tra l'altro non rara parlando di applicazioni di questo tipo.

- La ricezione GPS dipende molto dal tipo di ricevitore usato (totalmente indipendente dall'applicazione in quanto questa legge solo una stringa), quindi è possibile pensare che, specie in città, non sia sempre possibile, o quantomeno difficoltoso, stabilire una connessione satellitare, con la ovvia conseguenza che non è possibile usufruire delle coordinate geografiche; per questo motivo sono state previste due modalità di funzionamento, ON-LINE e OFF-LINE.
  Nel corso dei test questo è successo molte volte, ma solo per quanto riguarda la prima connessione, cioè quando il ricevitore deve stabilire il primo collegamento e allineamento con i satelliti; generalmente, fatto questo, può succedere che trovandosi in in una zona d'ombra del segnale il ricevitore perda la connessione, ma una volta tornato in una zona più libera impiega pochi secondi per riottenerla.

- A volte la connessione bluetooth viene persa durante il normale collegamento tra dispositivo e ricevitore; questo non dipende direttamente ne dal software ne dall'hardware, ma dipende dalla tecnologia in se, nel senso che trattandosi di un collegamento via onde radio, è soggetto a interferenze varie di tipo ambientale.
  Non rappresenta in realtà un grosso problema in quanto è raro che succeda, in questo caso basta ripetere la procedura di collegamento.

- La velocità di ricerca dei PI è legata alla quantità di dati presente nel file, almeno questo è quanto è quanto succede provando con file di diversa dimensione; in ogni caso può passare da una visualizzazione dei dati quasi in tempo reale a una visualizzazione dopo pochissimi secondi nel caso di file contenente un centinaio di PI.
  In realtà ho potuto notare che sicuramente dipende molto anche dalle risorse hardware del dispositivo, in quanto se l'applicazione funziona in modalità OFF-LINE, la velocità è immediata indipendentemente dalla quantità di dati presenti, mentre in modalità ON-LINE, succede quanto detto sopra; probabilmente il fatto che ci sia un Thread perennemente

attivo (quello della connessione GPS) richiede risorse hardware non indifferenti nel caso vi siano più elaborazioni in corso.
Questa ipotesi è supportata da un'altro test effettutato con uno smartphone di ultima generazione e con risorse hardware decisamente più consistenti, con il risultato che la velocità di funzionamento del software è sensibilmente aumentata. Ho provato, in maniera molto empirica, a tenere traccia del tempo che impiega a caricare i dati; ho fatto diverse prove, utilizzando 3 file diversi contenenti 20, 50 e 150 PI, i cui risultati sono visibili in [Fig. 3.1].

GRANDEZZA FILE	AVVIO APPLICAZIONE	VISUALIZZAZIONE DATI
20 KB	Immediato	Immediata
40 KB	Immediata	2 secondi circa
160 KB	Immediata	3 secondi circa

Fig. 3.1: Risultati dei Test

Naturalmente questi dati sono molto indicativi, i tempi sono una media sui 3 test effettuati per ogni file, senza riavviare l'applicazione tra uno e l'altro (a parte naturalmente tra un cambio di file e l'altro), ma semplicemente tornando indietro nella procedura di visualizzazione. Lo stesso test effettuato su un smartphone molto recente ha dato invece risultati decisamente più positivi, infatti non ha mostrato rallentamenti degni di nota in nessun caso. Per quanto riguarda la compatibilità, sottolineo il fatto che è stato testato anche su un sony ericsson Z520, ma da subito si è visto che si tratta di uno di quei casi in cui l'applicazione non può per niente funzionare in quanto il cellulare è privo di una delle due librerie principali, cioè la JSR-75, quella che permette l'uso del filesystem; inoltre è privo anche dello slot per l'uso della memory card.

3.2 Conclusioni e sviluppi futuri

L'obbiettivo della tesi consiste nel progettare e sviluppare una guida turistica elettronica funzionante su dispositivi mobili, quali cellulari, palmari e smartphone; in questo elaborato sono stati descritti i vari passi effettuati, nonchè le tecnologie che hanno permesso ciò. Il prodotto segue le specifiche tecnologiche standard, come quelle espresse ad esempio dal consorzio NMEA per quanto riguarda la comunicazione GPS; sfrutta le librerie JSR-72 e JSR-85 per la comunicazione Bluetooth e la lettura del filesystem rispettivamente; utilizza il protocollo MIDP 2.0 per lo sviluppo con tecnologia J2ME.

L'applicazione risulta stabile nella maggior parte dei casi testati; i principali problemi riscontrati sono di natura ambientale, cioè in determinate situazioni risulta difficoltosa la connessione Bluetooth e/o GPS, ma è facilmente risolvibile tramite ripetizione della procedura di connessione.
Teoricamente, parafrasando anche il motto della Sun per quanto riguarda la portabilità del codice java, l'applicazione dovrebbe supportare un gran numero di dispositivi; purtroppo la politica tecnologica attivata dalle case costruttrici non permette una completa compatibilità, infatti non tutti i dispositivi implementano le librerie necessarie al suo completo funzionamento (specie nei modelli meno recenti); si auspica che in un futuro anche prossimo, visto l' andamento dell'innovazione tecnologica di questo tempo, un gran numero di dispositivi riescano a condividere senza tanti problemi la maggior parte delle applicazioni esistenti.
L' applicazione così come si presenta è da considerarsi comunque un prototipo funzionante al 100% ma sicuramente aperto a tutta una serie di sviluppi:

- per cominciare la grafica può essere riscritta utilizzando le Canvas, svincolandosi così dall'interpretazione che ha ogni dispositivo delle API grafiche di J2ME;

- si potrebbero inserire delle classi per gestire anche file audio e video, renderebbero più ricco il database di informazioni dei vari PI.

- Un'altra caratteristica passibile di miglioramenti, agendo nello specifico sulle strutture dati, è la velocità di visualizzazione dei dati, in particolare lavorando sugli algoritmi di ricerca.

I test effettuati su alcuni modelli hanno dato esiti positivi, su altri invece completamente negativi, cioè l'applicazione non poteva neanche essere avviata per mancanza di librerie, come ad esempio sul modello Sony Ericsson z520 e z530.
Questa applicazione può avere sicuramente riscontri positivi dall'utenza, in quanto risulta pratica da utilizzare, ma soprattutto economica e leggera nell'esecuzione; inoltre, la sua diffusione può essere facilitata dal fatto che ormai il cellulare è diventato un oggetto di massa; è difficile incontrare un turista che non ne possieda uno e un'applicazione di questo tipo non può fare altro che semplificare, automatizzare e velocizzare quello che è il gesto di sfogliare invece un libro alla ricerca delle informazioni, magari senza sapere neanche dove ci si trova in quel momento.

Bibliografia

[1] Sito web della Sun Microsystem dedicato a J2ME.
http://java.sun.com/javame/index.jsp

[2] J. Knudsen,
Wireless Java: Developing with J2ME, Second Edition, Apress, 2003

[3] Sito degli sviluppatori Sun dedicato alle specifiche delle JSR
http://java.sun.com/javame/reference/apis.jsp.

[4] Sito ufficiale sulla tecnologia Bluetooth
http://www.bluetooth.com.

[5] Forum ufficiale Nokia
http://www.forum.nokia.com.

[6] Kumar et al,
Bluetooth Application Programming with the Java APIs, First Edition, Morgan Kaufmann, 2004.

[7] Note sul Linguaggio J2ME e sulle principali JSR
http://mobilezoo.biz/j2me.php.

[8] Note sul calcolo delle coordinate GPS
http://www.vialattea.net/esperti/php/risposta.php?num=9366.

[9] J2ME Tutorial, Part 1: Creating MIDlets
http://today.java.net/pub/a/today/2005/02/09/j2me1.html.

[10] A.Moller, M. Schwartzbach
Introduzione a XML, Pearson Education, 2007