Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Sistema software per la gestione di informazioni context aware mediante dispositivi PDA dotati di GPS

P. Amato, G. Rizzo, P. Storniolo, A. Urso

Rapporto Tecnico N:09
RT-ICAR-PA-05-09 **ottobre 2005**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Sistema software per la gestione di informazioni context aware mediante dispositivi PDA dotati di GPS

P. Amato[2], G. Rizzo[1], P. Storniolo[1], A. Urso[1]

Rapporto Tecnico N:09 ***Data:***
RT-ICAR-PA-05-09 **ottobre 2005**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo Viale delle Scienze edificio 11 90128 Palermo

[2] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica Viale delle Scienze 90128 Palermo

Indice

Sommario

Nel presente documento viene proposto un sistema software, sviluppato in Java, in grado di localizzare e di guidare, utilizzando la tecnologia GPS, un utente all'interno di un determinato luogo dove viene applicato tale sistema e di gestire contenuti *context aware*.
Il **context aware** è una branca di ricerca in fase di sviluppo che consiste nel fornire determinati servizi ad un utente, in base al contesto di fruizione del servizio.
Il sistema si propone come obiettivo quello di guidare l'utente, durante la visita all'interno del luogo dove viene applicato tale sistema, lungo un percorso scelto dall'utente stesso e di fornirgli a richiesta tutte le informazioni disponibili sui *Punti di Interesse* raggiunti.
Sul computer palmare, connesso in rete, un'applicazione del sistema interroga un'antenna GPS bluetooth dalla quale ottiene la posizione geografica espressa in latitudine e longitudine.
Una volta ricevuti i dati sulla posizione, questi vengono inviati mediante pacchetti UDP ad una applicazione che risiede sul server che a sua volta li memorizza nel database.
Nel server, una servlet spedisce pagine HTML al computer palmare contenenti le informazioni circa la posizione dell'utente e le istruzioni da effettuare per potere completare il percorso scelto.

Capitolo 1

Panoramica sul Global Positioning System (GPS)

1.1 I satelliti GPS

Il sistema GPS è attualmente composto da 24 satelliti di cui tre satelliti latenti (vere e proprie "ruote di scorta") della classe NAVSTAR blocco 2As prodotti dalla Rockwell International.
La vita media di un satellite GPS è di 7 anni e mezzo. I satelliti GPS orbitano a 20197 km. dalla terra (poco più del 50% della distanza per l'orbita geostazionaria) conseguentemente impiegano circa 12 ore per compiere un'orbita completa attorno al pianeta.
La velocità di spostamento rispetto al suolo terrestre è di circa 3000Km/h, mentre la velocità relativa riferita al piano dell'orbita è di circa 11000Km/h.

1.2 Il Control Segment

Il Control Segment è composto da cinque stazioni di sorveglianza sparse su tutta la terra :

- Colorado Springs (Stati Uniti);
- Isole Hawaii (Oceano Pacifico);
- Ascension Island (Oceano Atlantico);
- Isola Diego Garcia (Oceano Indiano);
- Isola Kwajalein (Oceano Pacifico).

La stazione di Colorado Springs è la master alla quale vengono confluiti tutti i dati dalle altre.
Lo scopo di queste stazioni è di controllare la posizione del satellite; questo calcolo viene fatto sfruttando il fatto che i satelliti non si trovano su orbite geostazionarie, e quindi passano sopra ad ogni stazione di controllo due volte al giorno.
Ad ogni passaggio la stazione di controllo calcola la corretta posizione del satellite, eventuali correzioni vengono trasmesse al satellite in modo d'aggiornare la propria posizione che è una delle informazioni fondamentali che il satellite irradia sulla terra.

1.3 Cenni sul funzionamento del GPS

Il principio su cui si basa il sistema GPS è il calcolo della distanza tra i satelliti e il ricevitore GPS, e da ciò conoscendo la posizione assoluta di tali satelliti ricavare la posizione del ricevitore sulla superficie terrestre in termini di latitudine, longitudine e altezza nel caso di rilevazioni tridimensionali. Il calcolo della distanza viene fatto calcolando la moltiplicazione tra il tempo che il segnale emesso dal satellite impiega per raggiungere il ricevitore e la velocità di propagazione delle onde elettromagnetiche.
Poiché gli intervalli temporali in gioco sono molto brevi, nell'ordine dei centesimi di secondo, sui satelliti sono montati ed utilizzati orologi atomici che offrono un'elevatissima precisione. Inoltre il sistema funziona in maniera passiva ovvero non si ha nessuna richiesta da parte del ricevitore al satellite.

I satelliti GPS emettono due tipi di segnali, chiamati C/A (Corse/Aquisition) e P (Precition). Il primo è accedibile da chiunque nella modalità SPS (Standard Positioning System) mentre il secondo è riservato all'impiego militare nella modalità PPS (Precise Positioning System).
La precisione orizzontale è nell'ordine dei 20 metri per la modalità SPS e superiore per la PPS, intorno ai 5 metri.
La trasmissione dei segnali viene effettuata attraverso l'utilizzo di due frequenze UHT: la L1 (1575.42 MHz) per la modalità SPS e la L2 (1227.60 MHz) oltre alla L1 per la modalità PPS.
Una volta calcolata la distanza dal satellite, si considera una sfera immaginaria che ha come centro il satellite stesso è come raggio la distanza e si calcola l'intersezione con la superficie terrestre; questa è rappresentata da una circonferenza che rappresenta il luogo dei punti in cui si trova il ricevitore.
A questo punto se sono visibili almeno tre satelliti dal ricevitore, il sistema GPS permette di calcolare la latitudine e la longitudine del ricevitore applicando una formula di triangolazione al triangolo ottenuto come intersezione tra la superficie terrestre e le tre sfere immaginarie aventi come centro il satellite e come raggio la distanza tra il satellite e il ricevitore; inoltre se sono visibili almeno quattro satelliti si può calcolare anche l'altezza.

1.4 Standard NMEA 0183

Lo standard NMEA (National Marine Electronics Association) 0183 è un protocollo tra dispositivi digitali.
Lo standard contempla i segnali elettrici, il protocollo e la temporizzazione della trasmissione ed i specifici formati dati, detti "sentences" o "frasi", operanti su un bus dati seriale, tipicamente RS 232.
Nell'ambito del sistema ogni bus dati può avere un unico elemento trasmittente (talker) e molti riceventi (listeners) .
Le frasi NMEA sono costituite da caratteri ASCII, caratterizzate da un prefisso, una serie di campi ed una checksum finale.
Ogni frase NMEA inizia con il carattere "$" e termina con la sequenza CR LF.
I singoli campi sono separati dalla virgola.
Una frase NMEA può contenere fino a 82 caratteri inclusi i delimitatori "$" e CR LF.
Se i dati per un campo non sono disponibili, il campo viene omesso pur rimanendo le virgole che lo delimiterebbero senza nessuna aggiunta di spazi tra loro.
Tutte le "sentences" iniziano con un prefisso costituito da sei caratteri. Il primo è rappresentato dal delimitatore "$" seguito da due caratteri che identificano l'entità che genera le sentences; nel caso del GPS, tutte le frasi iniziano con "$GP".
I successivi tre caratteri indicano il tipo di frase.
I campi che costituiscono la frase possono essere di lunghezza variabile e contengono le informazioni che vengono ricevute dal satellite come: il tempo, la latitudine, la longitudine, la velocità, ecc.
Alla fine di ogni frase è presente il campo relativo alla checksum, utile per rimediare ad eventuali errori avvenuti durante la trasmissione del messaggio.
Il checksum è l'exclusive OR a 8-bit di tutti i caratteri della frase NMEA esclusi i delimitatori "$" e "*", dove quest'ultimo rappresenta il carattere iniziale del campo checksum.

Capitolo 2

Sistema proposto

Lo scopo del progetto è la realizzazione di un servizio capace di gestire contenuti web context aware, che si adatti al contesto di fruizione del servizio e quindi alla posizione geografica dell'utente che utilizza il servizio stesso.
La posizione geografica dell'utente viene rilevata attraverso l'uso di ricevitori GPS.
L'idea di base del progetto è stata quella di realizzare un sistema capace di guidare all'interno di una specifica area un utente in modo da fornirgli tutte le indicazioni, in dipendenza della posizione geografica e del percorso scelto, necessarie a potergli permettere di effettuare un percorso durante il quale visitare tutti i POI (Punti di Interesse) che appartengono a tale percorso e inoltre, una volta che l'utente si trovi nell'area d'interesse di un POI avere la possibilità di potere accedere alle risorse relative a tale POI.
Il sistema presuppone che l'utente in questione sia dotato di un computer palmare su cui viene eseguita l'applicazione che comunica con il dispositivo GPS mediante il protocollo Bluetooth. Questa applicazione permette di ottenere le informazioni circa la posizione geografica dell'utente espressa in termini di latitudine e longitudine.
Inoltre si presuppone che la zona dove si utilizzerà tale sistema sia coperta da una rete Wireless.
Ciascun computer palmare che usufruirà del servizio avrà un indirizzo IP, che verrà opportunamente assegnato nel momento in cui l'utente entra nell'area coperta dal servizio.
Un tipico scenario è il seguente: l'utente entra nell'area coperta dal servizio, effettua la fase di registrazione nella postazione d'accesso dove gli viene consegnato il ricevitore GPS; a questo punto si connette, attraverso il Web Browser, al sito dell'ente che gestisce l'area e si scarica il software Pocket Visit vers. 1.1, necessario per inviare la posizione al server, avvia il programma, e attende l'indicazione che la connessione al server è andata bene; a questo punto seleziona la modalità con cui vuole usufruire del servizio; la modalità FREE, dove l'utente riceve solo informazioni relative alla posizione, o la GUIDE che oltre a fornire informazioni sulla posizione guida l'utente dando passo dopo passo tutte le indicazioni per potere completare un percorso prestabilito all'interno dell'area in questione.
Al completamento del percorso l'utente consegna il ricevitore GPS e registra l'uscita.

2.1 Architettura del sistema proposto

Il sistema realizzato è un sistema distribuito in cui distinguiamo un server centrale e quattro diversi tipi di client (User PDA, Administration PDA, Access Terminal, Administration Terminal).

Il sistema proposto è costituito da sette sottosistemi:

- Server;
- AdministrationServer;
- Servlet;
- SystemManage;
- UserManage;
- Pocket Visit vers. 1.1;
- setPOI vers. 1.1.

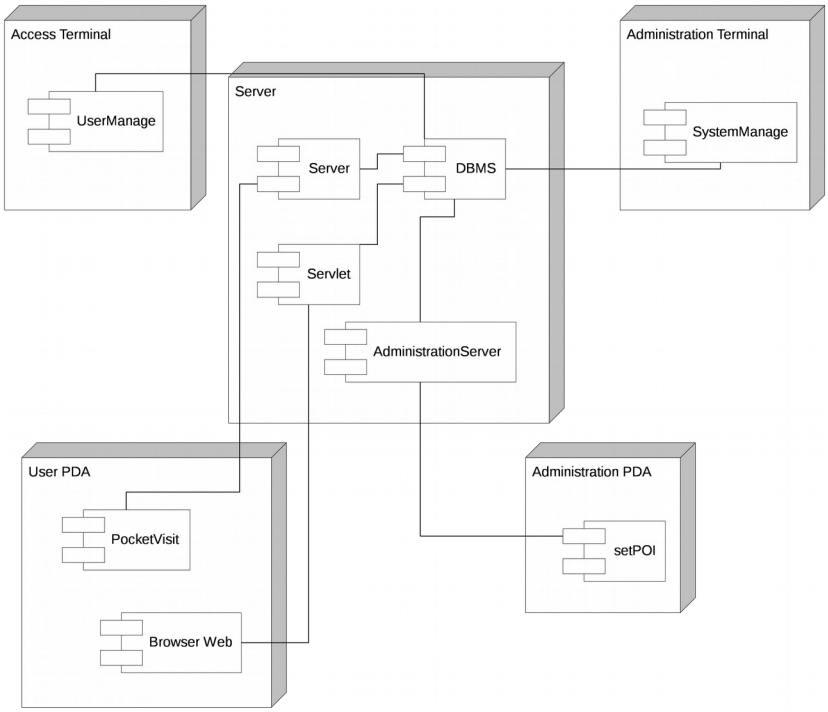
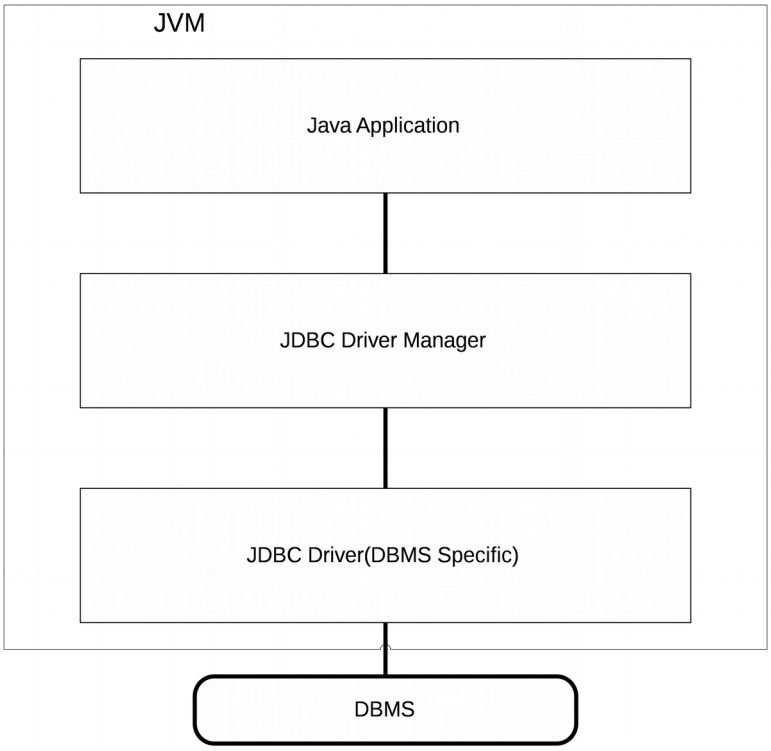


Figura 2.1 Diagramma di deployment

I sottosistemi Server, AdministrationServer e Servlet vengono installati nel server.
Il sottosistema SystemManage viene installato sulla macchina client Administration Terminal che comunica con il server, tramite una connessione TCP/IP, per tutte le operazioni che devono essere effettuate al DBMS.
Il sottosistema UserManage viene installato sulla macchina client Access Terminal che comunica con il server, tramite una connessione TCP/IP, per tutte le operazioni che devono essere effettuate al DBMS.
Il sottosistema Pocket Visit vers. 1.1 viene installato su un computer palmare client User PDA e comunica con il server tramite connessione UDP.
Il sottosistema setPOI vers. 1.1 viene installato su un computer palmare client Administration PDA e comunica con il server tramite connessione UDP.
La comunicazione col database viene effettuata utilizzando il protocollo JDBC(Java Database Connectivity), che permette alle applicazioni scritte completamente in Java di accedere in modo uniforme a basi di dati relazionali[riferimento a libro di database].
Dal punto di vista architetturale JDBC è suddiviso in due strati principali(vedi figura): il primo che fornisce una interfaccia verso il programmatore, il secondo di livello più basso che fornisce invece una serie di API per i produttori di drivers verso database relazionali e nasconde all'utente i dettagli del driver in uso. Questa caratteristica rende la tecnologia indipendente rispetto al motore relazionale con cui il programmatore deve comunicare.
I driver utilizzabili con JDBC sono di quattro tipi: bridge jdbc/odbc; driver nativo; middleware server e driver java.
La soluzione adottata nel sistema è quella con driver java ovvero l'uso di un driver specifico per il particolare DBMS.
I driver JDBC vengono caricati dall'applicazione in modo dinamico mediante un'istruzione del tipo Class.forName(package.class). Una volta che il driver (classe java) viene caricato è possibile connettersi al database tramite il driver manager utilizzando il metodo getConnection() che richiede in input la URL della base dati ed una serie di informazioni necessarie ad effettuare la connessione,come la password e l'username.



Architettura JDBC utilizzata

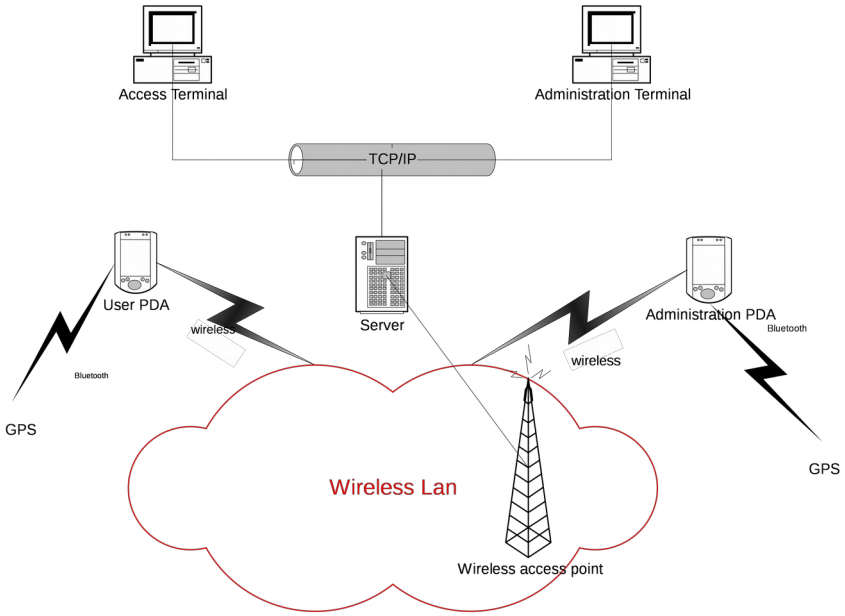


Figura 2.2 Architettura sistema proposto

Nelle sezioni seguenti vengono descritte in dettaglio tutte le parti del sistema proposto.

2.2 Parte server del sistema proposto

La parte server del sistema proposto è costituita da tre sottosistemi :

1. Il sottosistema Server gestisce la fase di ricezione della posizione, mediante pacchetti UDP, da parte di tutti i client di tipo User PDA ed effettua la registrazione di tali dati nel DBMS.

2. Il sottosistema AdministrationServer gestisce la fase di ricezione dei dati di configurazione dei POI, mediante pacchetti UDP, da parte del client di tipo Administration PDA ed effettua la registrazione di tali dati nel DBMS.

3. Il sottosistema Servlet è costituito da tre Servlet è si occupa di tutto il meccanismo di gestione del context aware, ovvero di spedire ai client pagine HTML in base alla posizione.

Si suppone che sulla macchina server siano installati: il database MySQL, la JVM versione 1.4 o più recente con i relativi driver JDBC per MySQL e il server Web Apache Tomcat.
Nella fase di testing si è utilizzato un notebook Asus, Intel Centrino 1,5 GHz con 40 Gb di hard disk; naturalmente nel caso in cui il sistema debba gestire un numero elevato di utenti la macchina su cui va installato la parte server del sistema proposto dei possedere dei requisiti molto superiori rispetto a quella utilizzata in fase di testing sia dal punto di vista della potenza di calcolo che della dimensione dell'hard disk.

2.2.1 Sottosistema Server

Il sottosistema Server è costituito dalle classi che implementano un server UDP in ascolto, su una porta configurabile, per i pacchetti provenienti dagli *n* utenti registrati e presenti all'interno del parco che contengono le informazioni sulla posizione geografica dei vari utenti.
Questo sottosistema, si occupa di gestire la ricezione di molti pacchetti e per fare fronte a problemi di concorrenza si è adottata la tecnica del multithreading ovvero si associa ad ogni richiesta di servizio da parte di un client un thread; in questo modo il server è sempre libero in attesa di richieste di servizio da parte dei client.
E' il thread che si occupa di ricevere il pacchetto e di effettuare la connessione al database per eseguire la query d'inserimento. Anche qui si è presentato un problema di prestazioni poiché stabilire una connessione al database è un'operazione molto complessa, infatti è necessario caricare il driver del database, stabilire una comunicazione bidirezionale sicura fra l'applicazione che deve accedere ai dati e il DBMS, eseguire la procedura di autenticazione e allocare le risorse necessarie per gestire le query successive. Quindi, l'esecuzione di questa procedura è molto onerosa e richiede alcuni secondi per essere portata al termine. Nel contesto dell'applicazione Server che deve gestire molte connessioni, l'overhead dovuto a ciò sarebbe molto pesante visto che l'obiettivo del sistema è quello di fornire in maniera interattiva le informazioni all'utente.
La soluzione adottata a questo problema è la tecnica del pool di connessioni ovvero ogni volta che il server termina l'esecuzione della query di inserimento della posizione del client, la connessione non viene chiusa ma viene inserita in cache; al successivo pacchetto che arriva il server prima di effettuare una nuova connessione guarda nella cache se c'è gia una connessione esistente e disponibile e la utilizza evitando l'operazione di creazione di una nuova connessione.
L'applicazione Server è costituita da quattro classi Java:

1. *ConnectionPool*: Classe che implementa il meccanismo del connection pool permettendo di memorizzare le connessione utilizzate in una coda e quindi di riutilizzarle senza dovere effettuare tutta la fase di connessione molto onerosa dal punto di vista del tempo.

2. *ConnectioPoolException*: Classe che si occupa della gestione delle eccezioni generate dalla classe ConnectionPool.

3. *Server*: Classe che implementa un server UDP in ascolto alla porta 9999. Ad ogni richiesta di sevizio da parte di un client associa un oggetto della classe thread.

4. thread: Classe che gestisce il servizio richiesto dal client ovvero quello di effettuare l'inserimento della posizione geografica del client nel database.

2.2.2 Sottosistema AdministrationServer

Il sottosistema AdministrationServer è costituito dall'omonima classe che implementa un server UDP in ascolto sulla porta 10000 di pacchetti provenienti dal client di tipo Administration PDA; tali pacchetti contengono le informazioni necessarie per la configurazione dei POI.
Una volta che riceve i pacchetti relativi ad un POI effettua la connessione con il database e successivamente la query d'inserimento.
Il sottosistema AdministrationServer è stato implementato con una sola classe poiché questa funzionalità del sistema non presenta problemi di concorrenza in quanto vi è un solo client e quindi non si è ritenuto necessario l'utilizzo del multithreading e di conseguenza non si è implementata la tecnica del pool di connessioni visto che le connessioni sono relative ad un solo client e con una frequenza bassa.

2.2.3 Sottosistema Servlet

Il sottosistema Servlet è costituito da tre servlet java, che implementano il meccanismo di contex aware; infatti le servlet sono invocabili dal computer palmare di ogni client di tipo User PDA, naturalmente dopo l'avvenuta connessione con il ricevitore GPS. Ad ogni richiesta HTTP del client le servlet rispondono inviando pagine web in base alla posizione dell'utente.
Le pagine guida e quelle info sulla posizione vengono inviate in maniera automatica utilizzando la tecnica del *client pull*.
Le servlet che costituiscono il sottosistema Servlet sono:

- HomePageServlet;
- ServletFree;
- ServletGuide.

2.2.3.1 HomePageServlet

La servlet HomePageServlet viene invocata dal client una sola volta, precisamente all'ingresso nel sito dell'ente che gestisce il sistema in un'apposita area.
Le funzionalità di questa servlet sono:

- Conteggiare i client che hanno usufruito del servizio fino al momento della connessione di tale client;
- Permettere di prelevare il software Pocket Visit vers 1.1 necessario per il corretto utilizzo del sistema;
- Assegnare ad ogni client un cookie utile per identificare il client in modo che le successive richieste HTTP effettuate da quel client stesso abbiano, nel campo *set-cookie*, lo stesso valore;

- Permette all'utente di scegliere tra la modalità FREE o GUIDE di utilizzo del servizio, invocando la servlet ServletFree per la modalità FREE e la servlet ServletGuide per la modalità GUIDE.

La figura 2.3 illustra una Home Page d'esempio dalla quale è possibile accedere al servizio cliccando o su "*Entra in modalità FREE*" o su "*Entra in modalità GUIDE*".

Figura 2.3 Esempio di Home Page

2.2.3.2 ServletFree

La servlet ServletFree è la servlet che si occupa della spedizione delle pagine web ai client, in base al loro contesto (posizione).
Le attività della servlet sono:

1. caricamento delle informazioni relative a tutti i POI presenti nel DBMS mediante l'esecuzione della query di selezione; attività che viene fatta solo all'inizio dell'istanziazione della servlet ed è valida per tutto il ciclo di vita della servlet;
2. ad ogni richiesta del client effettua una query al database per ottenere la posizione corrente del client; calcola la distanza di quest'ultimo da tutti i POI e controlla se il client si trova nell'area d'interesse di qualche POI;
3. spedisce le risorse relative al contesto del client mediante l'uso di pagine HTML.

In base alla posizione del client si hanno differenti casi:

- *client in area non interessata da POI*: in questo caso viene spedita una pagina HTML che visualizza la posizione corrente del client e indica la distanza dal POI più vicino; un esempio è mostrato in figura 2.4;

- *client in area interessata da un solo POI*: in questo caso viene spedita una pagina HTML che avvisa il client che si trova nell'area di interesse di un POI, un link alle risorse relative al POI in cui si trova il client; una pagina web d'esempio è mostrata in figura 2.5;
- *client in area interessata da n POI* : in questo caso viene inviata una pagina HTML che contiene la denominazione, la distanza e un link alle risorse relative ad ognuno dei POI che interessano il client; una pagina web d'esempio è mostrata in figura 2.6.

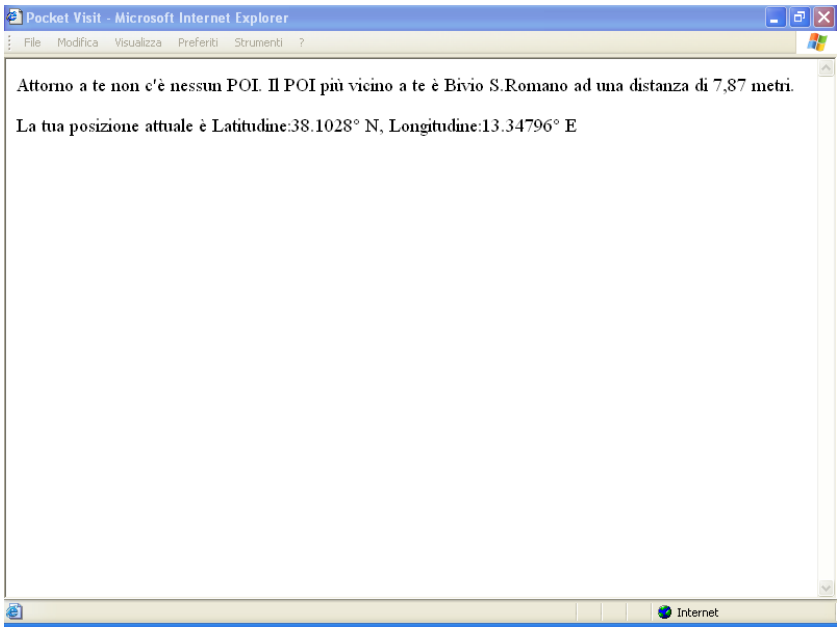


Figura 2.4 Client in area non interessata da POI.

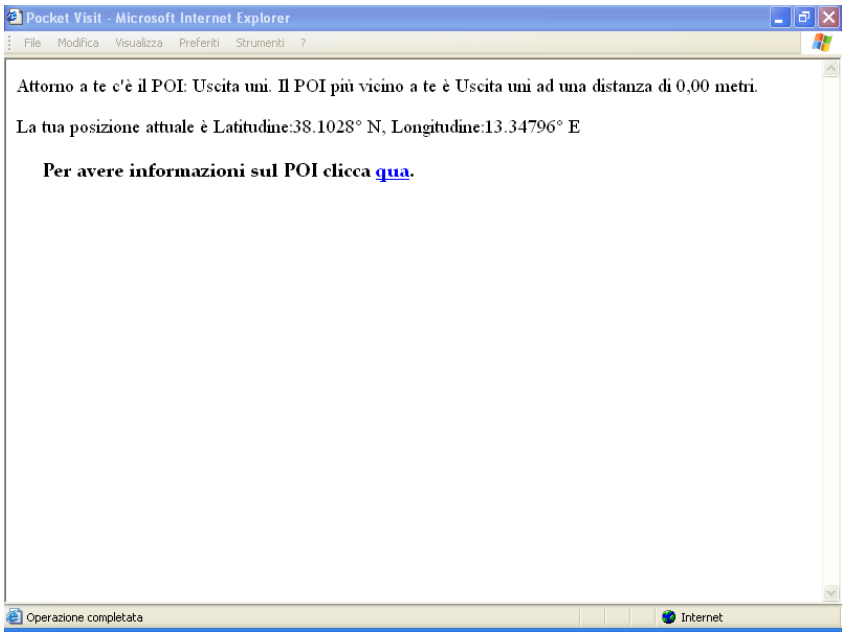


Figura 2.5 Client in area interessata da un solo POI.

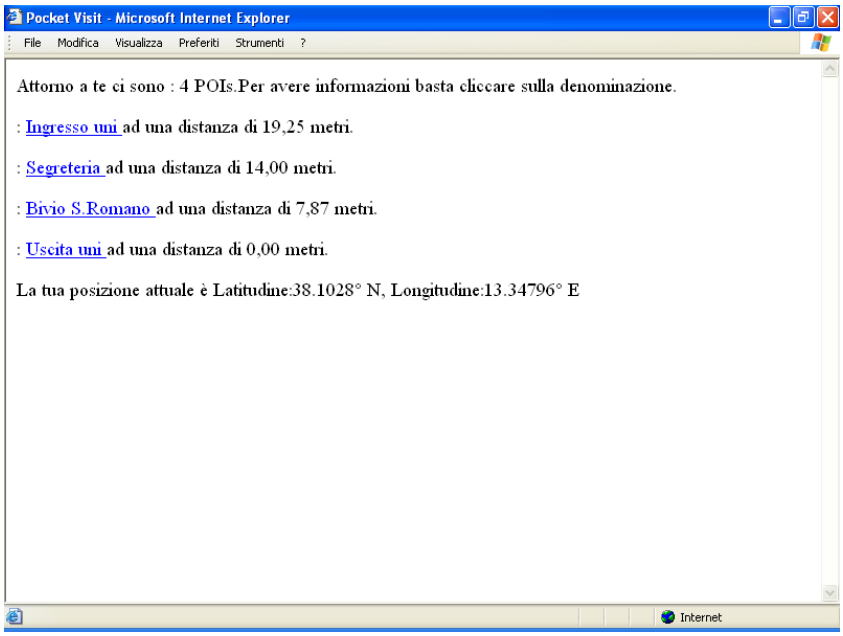


Figura 2.6 Client in area interessata da n POI.

2.2.3.3 ServletGuide

La servlet ServletGuide è la servlet che si occupa della spedizione delle pagine web ai client, in base al loro contesto (posizione) e al percorso scelto.
Le attività della servlet sono:

1. caricamento delle informazioni relative a tutti i POI presenti nel DBMS mediante l'esecuzione della query di selezione; attività che viene fatta solo all'inizio dell'istanziazione della servlet ed è valida per tutto il ciclo di vita della servlet;
2. alla prima richiesta del client la servlet mostra una pagina web, vedi figura 2.7, con tutti i possibili percorsi che si possono effettuare all'interno dell'area, scelto il percorso la servlet effettua una query nel DBMS per la creazione del traguardo corrente per tale client;
3. alle successive richieste di ogni client effettua una query al database per ottenere la posizione corrente e il traguardo corrente del client; controlla se il client ha terminato il percorso e in tal caso visualizza una pagina web come in figura 2.8, altrimenti calcola la distanza del client dal POI che deve essere raggiunto e controlla se il client si trova nell'area d'interesse di tale POI;
4. spedisce le risorse relative al contesto del client mediante l'uso di pagine HTML.

In base alla posizione del client e del traguardo corrente si hanno diversi casi:

- *Utente all'inizio del percorso*: la servlet controlla se l'utente è appena entrato nell'area servita dal servizio e invia una pagina web contenente i percorsi disponibili e in seguito alla scelta dell'utente una pagina web che spiega il servizio GUIDE, vedi figura 2.9;
- *Utente lungo percorso*: la servlet invia, ad intervalli di tempo Δt, pagine web contenenti le indicazioni da seguire per raggiungere il POI successivo, un esempio di tale pagina è mostrato in figura 2.10;

- *Utente nell'area d'interesse di un POI di tipo itinerario*: la servlet invia una pagina web indicando che è stato raggiunto un POI itinerario e indica l'azione da effettuare prima di continuare il percorso, un esempio di tale pagina è mostrato in figura 2.11;
- *Utente nell'area d'interesse di un POI di tipo interesse*: la servlet mostra una pagina web che indica al client che si trova nell'area d'interesse di un POI interesse e un relativo link per accedere alle risorse relative a tale POI, un esempio di tale pagina è mostrato in figura 2.12;
- *Utente alla fine del percorso*: la servlet invia una pagina web nella quale informa il client che il percorso è stato ultimato, un esempio di tale pagina è mostrato in figura 2.8.

Per approfondire il funzionamento della servlet in relazione alla gestione del percorso vedere il capitolo relativo alla creazione dei percorsi.

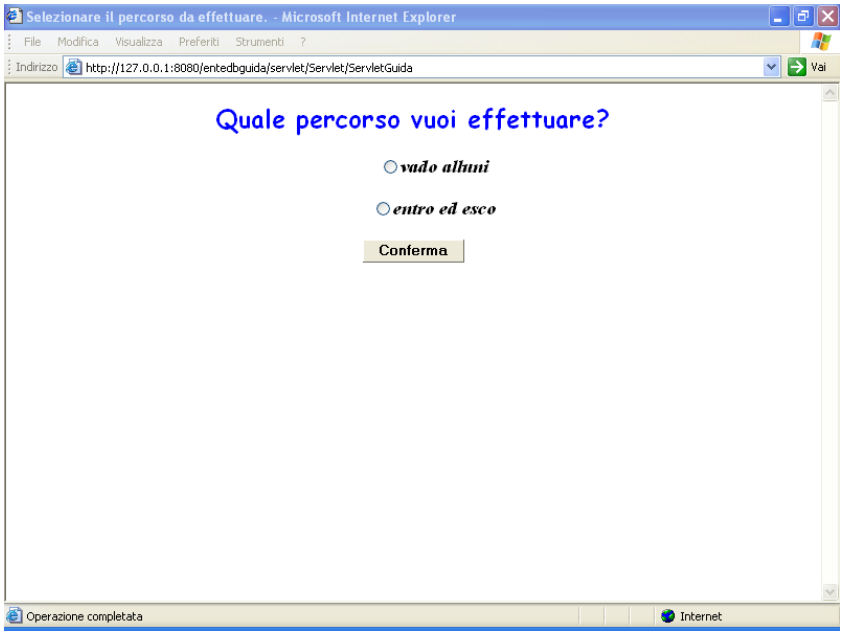


Figura 2.7 Pagina web di scelta del percorso

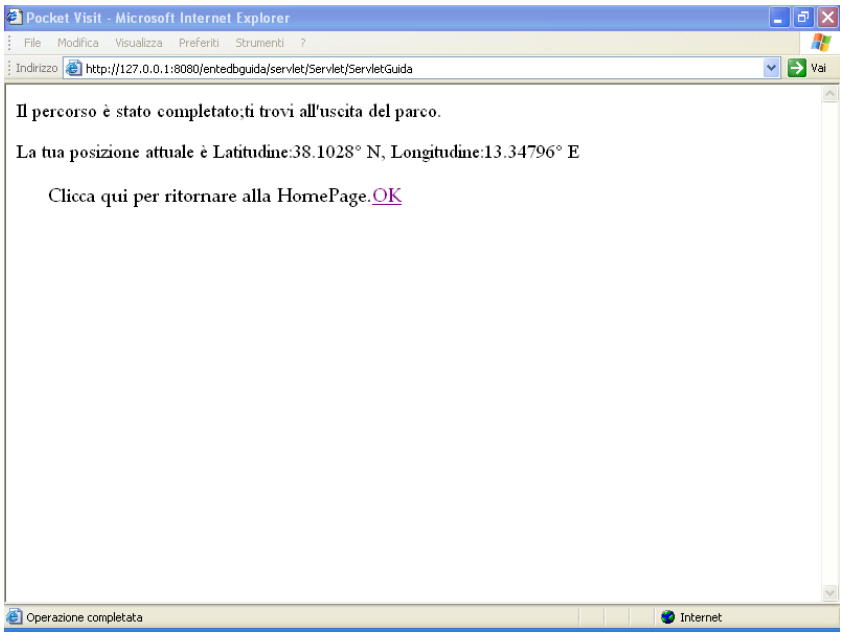


Figura 2.8 Pagina web fine percorso

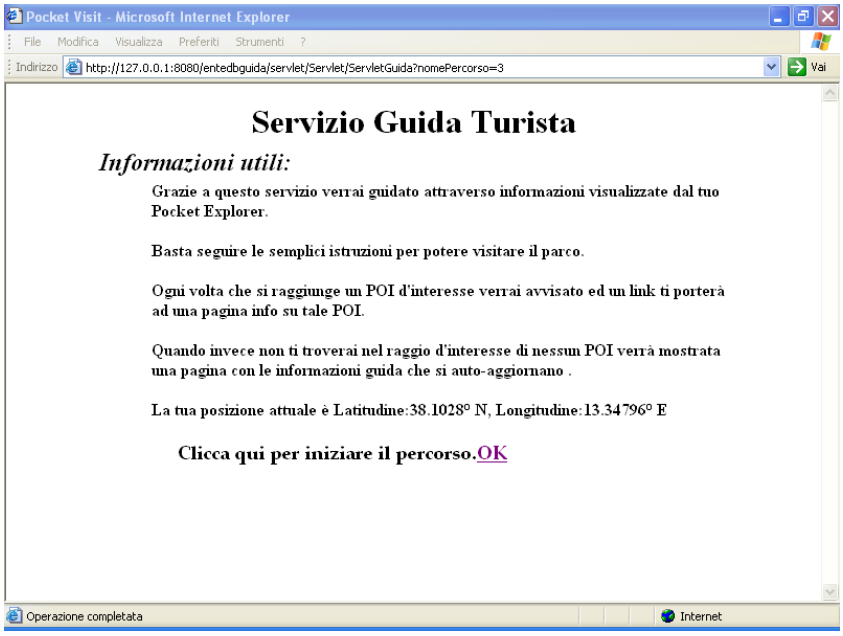


Figura 2.9 Pagina web info su servizio GUIDE

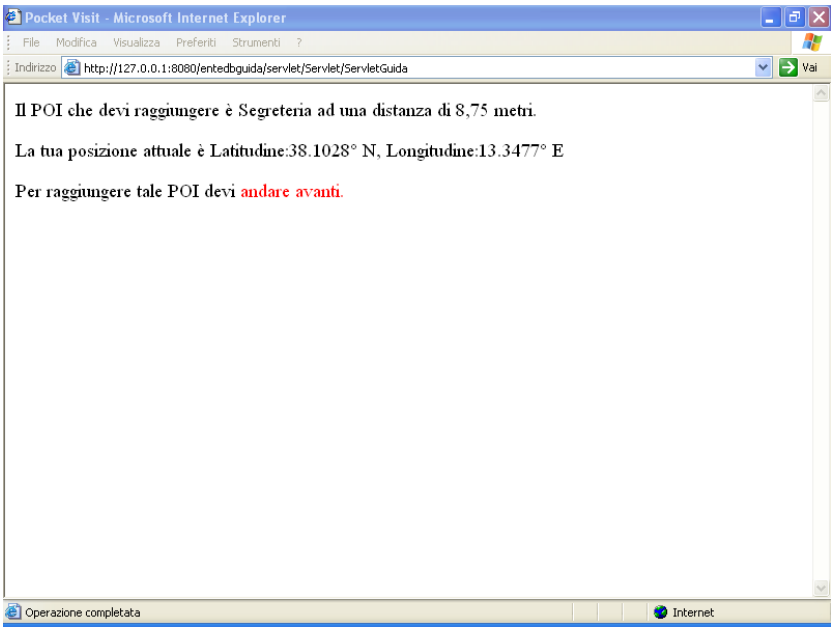


Figura 2.10 Pagina web con istruzioni guida

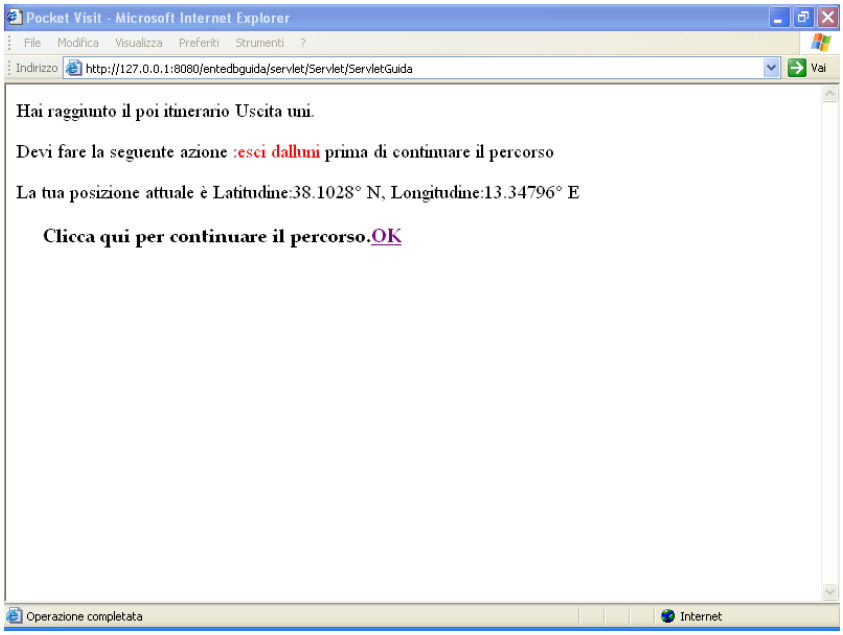


Figura 2.11 Client in area d'interesse POI itinerario

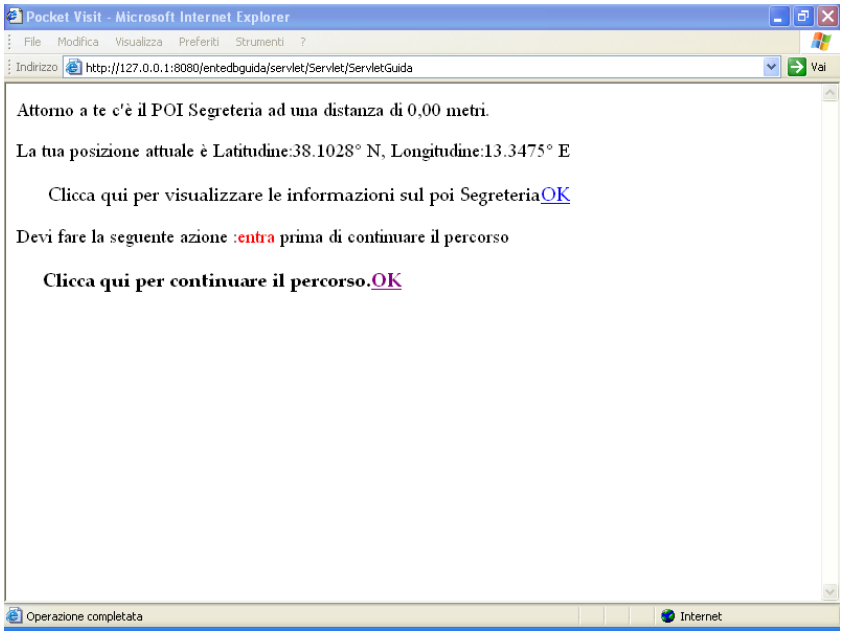


Figura 2.12 Client in area interesse POI interesse

2.3 Parte Client del sistema proposto

La parte client del sistema presenta diverse tipologie di client sia dal punto di vista funzionale che per l'hardware richiesto.
Il sistema proposto prevede la presenza di quattro tipologie di client:

1. *Client User PDA*: i client di questo tipo possono essere più di uno e sono i client che inviano la loro posizione al server e interrogano il server stesso per avere informazioni in base alla loro posizione;
2. *Client Administration PDA*: ha la sola funzione di configurare i dati relativi ad un POI inviando questi dati al server che li registra nel DBMS;
3. *Client Access Terminal*: è il client che permette di registrare nel DBMS l'ingresso e l'uscita di ogni utente dall'area in cui è attivo il sistema;
4. *Client Administration Terminal*: è il client che permette di effettuare tutte le operazioni utili alla corretta gestione del sistema, tra cui quella della creazione dei percorsi.

Nei seguenti paragrafi verranno analizzati in dettaglio tutte le tipologie di client.

2.3.1 Client User PDA

Il client User PDA è costituito da un computer palmare dotato di un ricevitore GPS.

L’applicazione che viene installata è Pocket *Visit vers 1.1* che è un adattamento dell’applicazione *Pocket Visit vers 1.0* descritta in [1], ovvero è stata modificata la sola classe *SerialConnection* permettendo all’applicazione di potere gestire la comunicazione con il Server per l’invio, ad intervalli di tempo Δt , della posizione geografica del client mediante connessione UDP.
Si rimanda a [1] per un maggiore approfondimento.
Una volta che la comunicazione è avvenuta dal computer palmare si può accedere, attraverso il browser web, al sito dell’ente e potere usufruire del servizio nelle modalità precedentemente descritte.
Nella fase di testing è stato utilizzato un computer palmare HP ipaq 5550 avente come sistema operativo Windows Pocket Pc 2003 e un’antenna GPS TOMTOM Bluetooth.
Inoltre per far funzionare l’applicazione, che è interamente scritta in linguaggio JAVA, il palmare è stato dotato della JVM sviluppata dalla NSICOM chiamata CRE-ME *vers. 3.25*; la scelta è ricaduta su questa JVM poiché supporta la libreria javax.comm necessaria per il funzionamento dell’applicazione.

2.3.2 Client Administration PDA

Il client Administration PDA è costituito da un computer palmare dotato di un ricevitore GPS.
L’applicazione che viene installata è *setPOI vers 1.1* che è un adattamento dell’applicazione *setPOI vers 1.0* descritta in [1], ovvero è stata modificata la sola classe *setPOI* per permettere di potere inserire la tipologia del POI e inoltre permettere all’applicazione di inviare, una volta terminata la configurazione del POI, mediante una connessione UDP alla porta 10000 all’AdministrationServer i dati relativi al POI in modo da potere essere registrato nel DBMS; se la registrazione è avvenuta con successo il client riceve un messaggio che visualizza ciò.
Si rimanda a [1] per un maggiore approfondimento.
Nella fase di testing è stato utilizzato un computer palmare HP ipaq 5550 avente come sistema operativo Windows Pocket Pc 2003 e un’antenna GPS TOMTOM Bluetooth.
Inoltre per far funzionare l’applicazione, che è interamente scritta in linguaggio JAVA, il palmare è stato dotato della JVM sviluppata dalla NSICOM chiamata CRE-ME *vers. 3.25*; la scelta è ricaduta su questa JVM poiché supporta la libreria javax.comm necessaria per il funzionamento dell’applicazione.

2.3.3 Client Access Terminal

Il client Access Terminal non necessita di particolare hardware, un normale personal computer è più che sufficiente. L’applicazione che viene installata è UserManage; questa applicazione permette di registrare all’ingresso l’utente inserendo tutti i dati e assegnando un’antenna GPS.
L’applicazione effettua una connessione al DBMS ed esegue la query di inserimento nel caso dell’ingresso e di aggiornamento, del campo ora uscita, nel caso dell’uscita dell’utente.
Nella fase di testing è stato utilizzato un personal computer con processore P4 a 1600 MHz dotato di connessione a internet necessaria per la comunicazione con il DBMS che risiede nel server.
Per il corretto funzionamento dell’applicazione, nella macchina deve essere presente una JVM vers 1.4 o superiore ed inoltre poiché si effettuano query al DBMS deve essere presente il driver ODBC per il DBMS a cui si fa riferimento.
Nella fase di testing è stata utilizzata la Java Virtual Machine vers 1.4.2_05 e i driver JDBC per il DBMS MySQL (MySQL Connector Java 3.1.4).

2.3.4 Client Administration Terminal

Il client Administration Terminal non necessita di particolare hardware, un personal computer comune è più che sufficiente.
L'applicazione che viene installata è SystemManage; questa applicazione permette di effettuare molteplici funzioni che sono:

- *Tracciamento percorso:* viene tracciato il percorso di un utente di cui si conosce l'IP e l'ora in cui era presente nell'area gestita dal sistema; il risultato è visualizzato in una finestra che contiene tutte le posizioni e la relativa data che si trovano memorizzate nel DBMS.
- *Mostrare situazione attuale:* viene visualizzato l'IP e la relativa posizione geografica di tutti gli utenti presenti nell'area gestita dal sistema;
- *Gestione POI:* permette di potere inserire, modificare e cancellare nel DBMS i POI mediante finestre grafiche;permette di inserire il raggio d'interesse del POI cosi da determinare la propria area d'interesse; inoltre si ha la possibilità di configurare per ogni POI l'url che si riferisce alle risorse relative a tale POI;
- *Gestione Antenne:* permette di potere inserire modificare e cancellare nel DBMS le Antenne mediante finestre grafiche;
- *Associare Utente a posizione:* permette di effettuare off-line l'aggiornamento del campo idutente di tutte le posizioni registrate nel DBMS per un dato giorno, che viene inserito dall'amministratore, utile a facilitare le ricerche, come per esempio il tracciamento di un percorso, poiché si può utilizzare l'idutente invece dell'indirizzo IP come chiave di ricerca;
- *Gestione Percorso:* permette di creare, modificare e cancellare un percorso nel DBMS. Per creare un percorso bisogna selezionare da un'apposita finestra nell'ordine in cui devono essere raggiunti i POI, e bisogna selezionare per ogni POI inserito nel percorso l'azione che bisogna effettuare al raggiungimento di tale POI;
- *Gestione AzionePOI:* permette di inserire, modificare e cancellare l'azionePOI dal DBMS, dove per azionePOI si identifica l'azione che l'utente deve fare una volta raggiunto un determinato POI; questa funzionalità è stata inserita in modo che le azioni possano essere configurare in relazione alle esigenze e alla tipologia dell'area in cui il sistema dovrà funzionare.

Nella fase di testing è stato utilizzato un personal computer con processore P4 a 1600 MHz dotato di connessione a internet necessaria per la comunicazione con il DBMS che risiede nel server.
Per il corretto funzionamento dell'applicazione, nella macchina deve essere presente una JVM vers 1.4 o superiore ed inoltre poiché si effettuano query al DBMS deve essere presente il driver ODBC per il DBMS a cui si fa riferimento.
Nella fase di testing è stata utilizzata la Java Virtual Machine vers 1.4.2_05 e i driver JDBC per il DBMS MySQL (MySQL Connector Java 3.1.4).

Capitolo 3

Algoritmo utilizzato per la creazione dei percorsi

Iniziamo col dire che la peculiarità del sistema proposto sta propriamente nel fornire all'utente del sistema delle informazioni in base al contesto (posizione) ed inoltre in base al percorso che è stato scelto dall'utente stesso.
Il percorso è costituito da una catena di anelli, come mostrato in figura 3.1, dove ogni anello unisce due POI, quello iniziale chiamato *POI_from (POI di provenienza)* e quello finale *POI_to (POI da raggiungere)*.

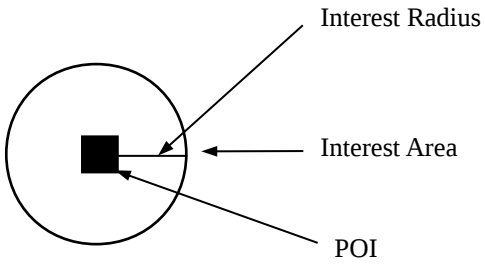


Figura 3.1 Rappresentazione di un POI

Ogni POI ha la caratteristica di avere un raggio d'interesse (vedi figura 3.1) che determina un'area d'interesse; il raggio è configurabile dall'amministratore in base alle caratteristiche del POI; il sistema ogni volta che un utente si trova dentro l'area di interesse di un POI considera il POI raggiunto dall'utente.
All'interno del percorso i POI possono essere di due tipologie:

- *POI di tipo Interesse*: sono i POI che hanno un certo interesse per cui debbano essere raggiunti e quindi visitati; inoltre il sistema prevede che per ogni POI di tipo interesse sia possibile accedere a delle risorse multimediali;
- *POI di tipo Itinerario*: sono i POI che vengono utilizzati per la creazione di percorsi univoci e sono normalmente posti in luoghi dove l'utente deve effettuare delle scelte sul tragitto (esempio bivi, edifici, tipologie di strade particolari, ponti, scale, ecc.); la dislocazione di questi POI è fatta in maniera tale che le indicazioni che si inviano all'utente siano necessarie per guidarlo e portarlo al completamento del percorso.

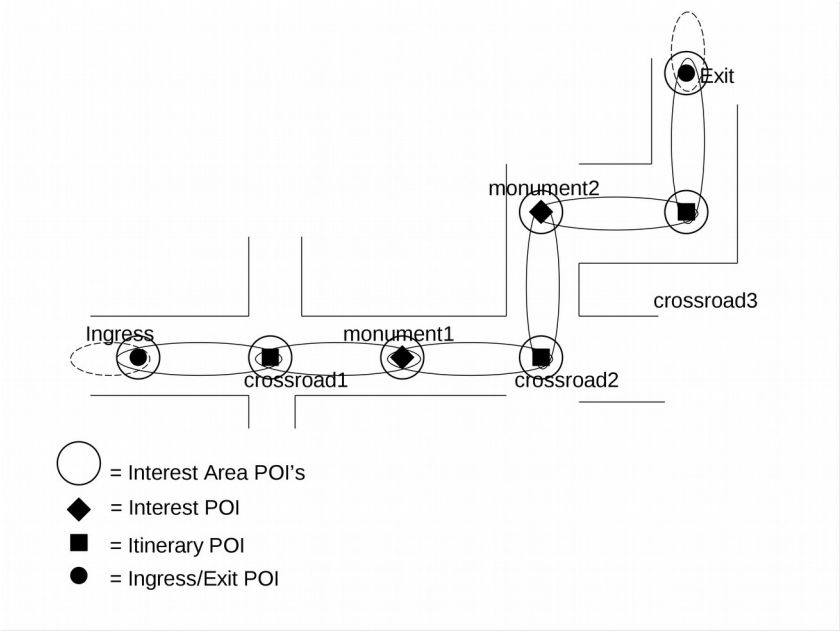


Figura 3.2 Esempio della creazione di un percorso

La figura 3.2 mostra l'applicazione dell'algoritmo per la creazione di un semplice percorso, immaginando che le linee rette rappresentino i bordi di una strada; come possiamo vedere il percorso non è altro che una catena d'anelli che unisce l'Ingress POI con l'Exit POI permettendo all'utente di visitare due Interest POI. Inoltre vediamo l'uso di tre Itinerary POI utili per potere guidare in modo corretto l'utente.
Attorno ad ogni POI c'è un cerchio che rappresenta l'area d'interesse del POI, infatti il sistema ritiene raggiunto il POI ogni volta che l'utente si trova in tale area d'interesse.
Come vediamo ogni anello unisce due POI, durante il quale l'utente sarà guidato con pagine web che gli indicano di andare avanti, finché l'utente non raggiunge l'area d'interesse del *POI_to*, se il *POI_to* e di tipo itinerario, come nel caso di *crossroad2*, all'utente sarà indicato di "*svoltare a sinistra"* prima di continuare il percorso; mentre se è di tipo Interesse, come nel caso di *monument2*, all'utente sarà data la possibilità d'accedere alle risorse su tale POI e inoltre sarà indicato di "*svoltare a destra*" prima di continuare il percorso.
Fanno eccezione gli anelli tratteggiati in quanto sono costituiti da un solo POI. Questi anelli hanno l'obiettivo di permettere al sistema di individuare quando l'utente si trova all'ingresso (quando il *POI_from* ha valore *null*) e quando si trova all'uscita (quando il *POI_to* ha valore *null*).
Il sistema di volta in volta tiene memoria del solo anello corrente, chiamato traguardo corrente, perché la sola conoscenza del POI di provenienza (POI_from) e del POI da raggiungere (POI_to) permettono al sistema di potere guidare l'utente; naturalmente al raggiungimento del POI_to, il traguardo corrente sarà aggiornato, ponendo come POI_from il POI appena raggiunto e come POI_to il POI_to dell'anello il cui POI_from è quello appena raggiunto; ovviamente nel momento che il POI_to presente nell'anello ha valore *null* vuol dire che l'utente ha completato il percorso.

Capitolo 4

Calcolo della distanza tra l'utente e i POI

Il calcolo della distanza tra l'utente e un POI viene effettuato utilizzando il teorema di Eulero (o del coseno). Infatti si considera il triangolo sferico ABP (vedi figura 4.1) e la distanza ortodromica *d* è pari a :

$\cos d = \cos C_A \cos C_B + \operatorname{sen} C_A \operatorname{sen} C_B \cos\Delta\lambda$

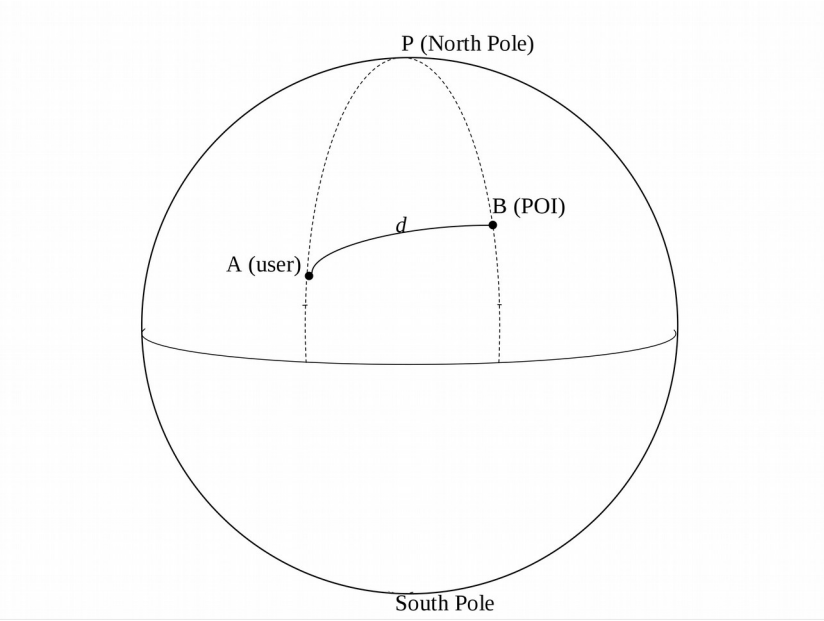


Figura 4.1

La formula è naturalmente algebrica, cioè occorrerà badare ai segni di ciascun termine, rispettando la convenzione di aver scelto come terzo vertice del triangolo ortodromico, il polo dell'emisfero del punto di partenza. In base a tale convenzione, considerando che il punto di partenza e quello di arrivo sono sempre nello stesso emisfero, e tenendo presente che le coordinate, espresse in termini di latitudine e longitudine, dei punti A e B sono [φ1, λ1] e [φ2, λ2] abbiamo:

$\cos d^\circ = \cos(90^\circ-\varphi 1)\cos(90^\circ-\varphi 2) + \sin(90^\circ-\varphi 1)\sin(90^\circ-\varphi 2)\cos(\lambda 1-\lambda 2)$

dove d° rappresenta la distanza in gradi; per ottenere la distanza in Km si moltiplica per il raggio medio della terra pari a 6367.45 km il termine $\arccos(\cos(d^\circ))$.

Capitolo 5

Gestione dei dati persistenti

Per la gestione dei dati persistenti il sistema utilizza un DBMS. Il DBMS è interrogabile via rete dai vari terminali client dopo avventa autenticazione.
Sono previste due modalità d'accesso al DBMS:

- *Modalità client*: il client può solamente effettuare query di inserimento e di selezione, non ha i permessi per potere modificare o cancellare dati dal DBMS; i client che utilizzano questa modalità sono: User PDA, Administration PDA, Access Terminal;
- *Modalità root*: il client può effettuare tutte le operazioni possibili ai dati memorizzati nel DBMS; il client che utilizza questa modalità è Administration Terminal.

Nella fase di testing il DBMS utilizzato è MySQL vers. 4.0.21.

5.1 Descrizione testuale ERD

Il database dovrà contenere un archivio, che ci permetta di gestire la registrazione dell'ingresso e dell'uscita di un utente dal parco, dove per parco si intende la zona dove viene utilizzato il sistema proposto; inoltre ci deve permettere di registrare le informazioni relative ad un percorso che è costituito da vari anelli e di tenere nota per ogni utente del traguardo corrente.
Ogni utente è caratterizzato da nome, cognome, tipo di documento, numero di documento, oraIngresso, oraUscita, IDantenna , IP .
Ogni antenna è caratterizzata da IDantenna, stato, data.
Ogni posizione è caratterizzata da latitudine, longitudine, dataRilevazione, IPutente, IDutente.
Ogni POI è caratterizzato da latitudine, longitudine, url, data, stato, tipologia, raggioInteresse, denominazione, IDpoi .
Ogni Percorso è caratterizzato da IDpercorso e nomePercorso.
Ogni Anello è caratterizzato da IDpoiTo, IDpoiFrom, azione, azionePoi, IDnodo e IDpercorso.

5.2 Identificazione delle entità e delle relazioni

Dalla descrizione testuale si sono identificati le seguenti entità, relazioni ed attributi :

- L'entità **Utente** è in relazione "assegnare" con l'entità antenna, in relazione "occupare" con l'entità posizione, in relazione "appartenere" con l'entità traguardoCorrente; gli attributi dell'entità utente sono: nome, cognome, tipoDocumento, numeroDocumento, oraIngresso, oraUscita, IDantenna, IP, IDutente.
- L'entità **Antenna** è in relazione "assegnare" con l'entità utente; gli attributi dell'entità antenna sono: IDantenna, data, stato.
- L'entità **Posizione** è in relazione "occupare" con l'entità utente; gli attributi dell'entità posizione sono: latitudine, longitudine, dataRilevazione, IPutente, IDutente.

- L’entità **POI** è in relazione “costituire”con l’entità anello; gli attributi dell’entità POI sono: latitudine, longitudine, url, data, stato, tipologia, raggioInteresse, denominazione, IDpoi.
- L’entità **Percorso** è in relazione “costituire” con l’entità anello; l’entità percorso ha i seguenti attributi: IDpercorso e nomePercorso.
- L’entità **Anello** è in relazione “costituire” con l’entità POI, in relazione “costituire” con l’entità traguardoCorrente, in relazione “costituire” con l’entità percorso; l’entità anello ha i seguenti attributi: IDpoiFrom, IDpoiTo, azione, azionePoi, IDnodo e IDpercorso.
- L’entità **TraguardoCorrente** è in relazione “appartenere” con l’entità utente, in relazione “costituire” con l’entità anello; l’entità traguardoCorrente ha i seguenti attributi: IDtraguardoCorrente, IPutente, IDpoiFrom, IDpoiTo, azionePOI, dataRegistrazione.
- L’entità **AzionePoi** ha i seguenti attributi: IDazione, azionePOI.

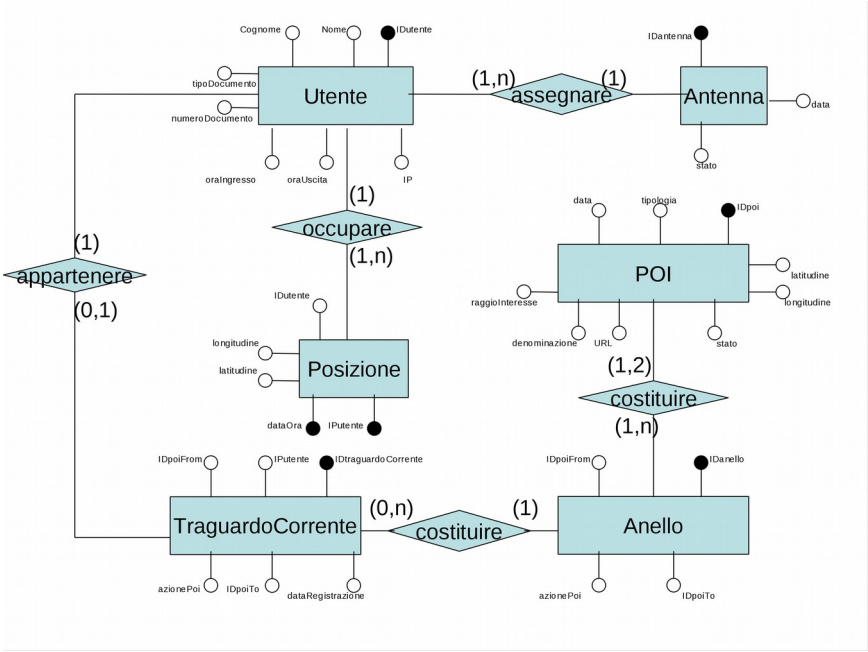
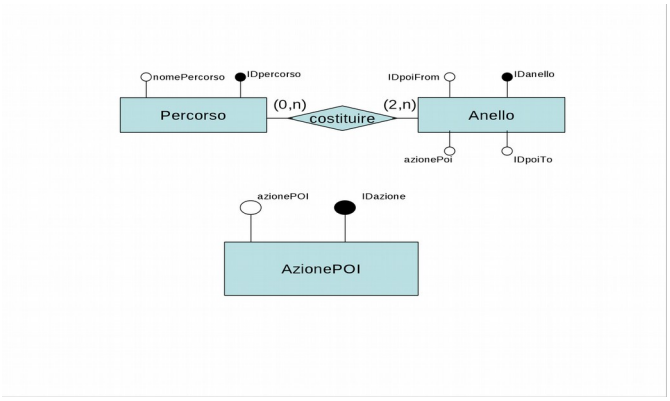


Figura 5.1 Diagramma Entità-Relazioni



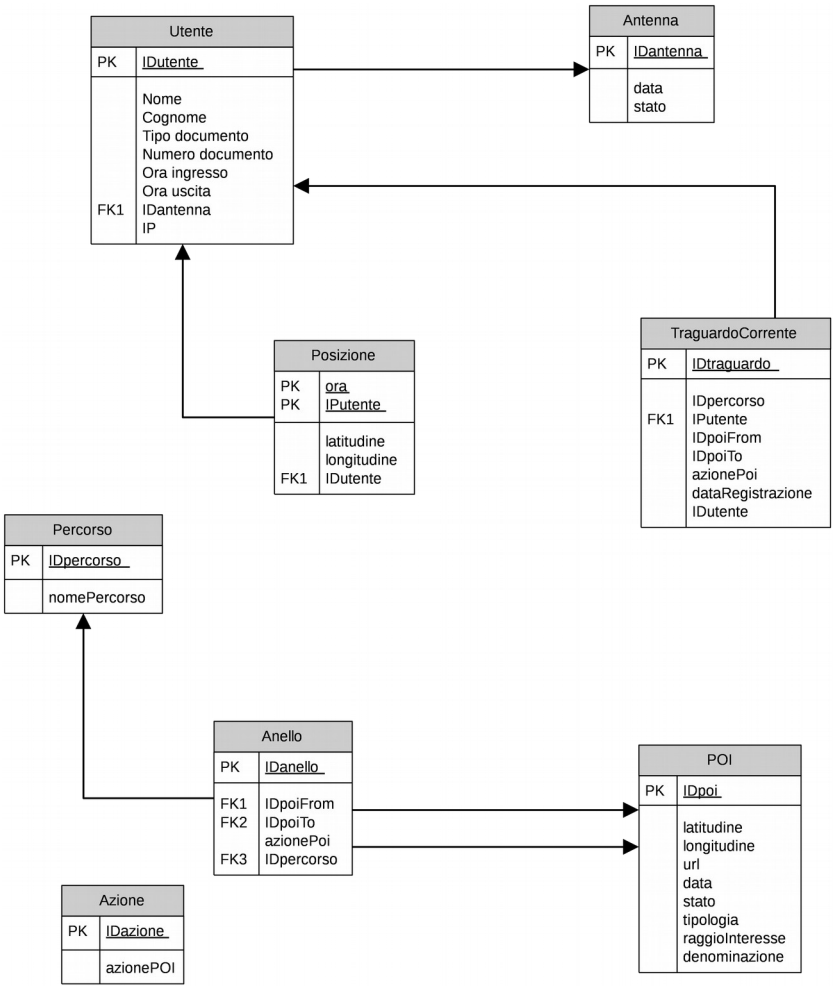


Figura 5.2: Diagramma delle relazioni fra le tabelle .

Guardando il diagramma mostrato in *figura 5.2* si può notare la mancanza della relazione tra TraguardoCorrente e Anello.
Questa è una scelta implementativa effettuata per migliorare le prestazioni dal punto di vista della velocità, poiché in questo modo si risparmia una query al database ogni volta che si deve recuperare il traguardoCorrente; perché in questo modo si hanno già tutti i dati dell'anello relativo a tale traguardo.
Infatti se si mantenesse la relazione fra le due tabelle, nella tabella traguardoCorrente si avrebbe come campo l'IDanello al posto di IDpoiFrom, IDpoiTo e azionePOI e quindi per recuperare queste informazioni si dovrebbe effettuare una ulteriore interrogazione.
Questa ridondanza di dati non ha un elevato costo poiché per ogni utente corrisponde una sola riga nella tabella traguardoCorrente.

Conclusioni

Il sistema descritto in questo documento consiste in un sistema capace di localizzare e guidare all'interno di un'area gli utenti presenti in tale area e muniti di computer palmare con il relativo ricevitore GPS.
Il sistema gestisce contenuti contex aware spedendo in maniera dinamica informazioni ai vari moduli client in relazione alla posizione e al percorso scelto.
Queste informazioni hanno lo scopo non solo di dare la possibilità al client di accedere a risorse sui vari POI che visita durante il percorso, ma principalmente hanno l'obiettivo di guidare,dando indicazioni dettagliate, l'utente affinché possa visitare l'area in base ad un percorso prestabilito.
Il sistema consiste in due parti: *client* e *server;* è un sistema distribuito e per questo si sono adottate delle scelte che hanno tenuto conto dei problemi di concorrenza e dell'obiettivo dell'interattività; infatti si sono adottate tecniche come il multithreading e il connection pool.
Il pregio del sistema è la sua adattabilità, infatti è compito dell'amministratore dell'ente, utilizzando l'applicazione SystemManage, configurare tutte le peculiarità che caratterizzano l'area in cui deve essere utilizzato il sistema.
Infine, effettuando opportune modifiche alla classe SerialConnection delle applicazioni Pocket Visit vers. 1.1 e setPOI vers. 1.1 è possibile utilizzare il sistema con diverse marche di ricevitori GPS.

Appendice A

Codice sorgente Java

Premessa

Di seguito viene riportato il codice Java relativo alle classi più importanti.
Per quanto riguarda le applicazioni Pocket Visit vers. 1.1 e setPOI vers. 1.1 vengono riportate le classi modificate rispetto alle versioni precedenti descritte in [1].

A.1 Applicazione Pocket Visit vers. 1.1

A.1.1 SerialConnection.java

```
package PocketVisit;
import javax.comm.*;


import java.io.*;
import java.awt.TextArea;
import java.awt.event.*;
import java.util.TooManyListenersException;
import java.util.StringTokenizer;
import java.net.*;
import java.lang.Exception;


/**
 * La classe SerialConnection implementa SerialPortListener e CommPortOwnershipListener.
 * La SerialPortEventListener propaga eventi di porta seriale mentre l'interfaccia
*CommPortOwnershipListener
 * fa si che quando una porta è aperta, un evento CommPortOwnership di tipo PORT_OWNED
 * (porta occupata) sarà propagato. Quando una porta è chiusa, un evento
 * CommPortOwnership di tipo PORT_UNOWNED (porta libera) sarà propagata.
 * Ogni volta che riceve dati dalla porta li invia al Server ad intervalli di tempo pari in secondi al
*valore del parametro int intervallo.
 * @author Pietro Amato
 *
 */
public class SerialConnection implements SerialPortEventListener,
                CommPortOwnershipListener { /*La SerialPortEventListener
                                        propaga eventi di porta seriale mentre
                                        l'interfaccia CommPortOwnershipListener
                                        fa si che quando una porta è aperta, un
                                        evento CommPortOwnership di tipo PORT_OWNED
                                        (porta occupata) sarà propagato. Quando una
                                        porta è chiusa, un evento
                                        CommPortOwnership di tipo PORT_UNOWNED
                                        (porta libera) sarà propagata */

   private PocketVisit parent; //Oggetto PocketVisit
   private TextArea messageAreaOut; //TextArea dei messaggi in uscita che passiamo
                                //dalla classe PocketVisit
   private TextArea messageAreaIn; //TextArea dei messaggi in uscita che passiamo dalla
                                //classe PocketVisit
   private SerialParameters parameters; //oggetto della classe SerialParameters che rappresenterà
                                  // i parametri che passiamo
   private OutputStream os; //Flusso in uscita
   private InputStream is; //Flusso in ingresso
   String old;//Serve per evitare errori dovuti a mancanze di informazioni dall'antenna GPS

  private CommPortIdentifier portId; /*CommPortIdentifier è un gestore di comuniczioni di porte
                                CommPortIdentifier è la classe centrale per il controllo
```

```
                                dell'accesso alle porte ci comunicazione*/
   private SerialPort sPort; // una porta di comunicazione seriale RS-232.
   private boolean open; //Booleano

   private long intervallo = 3000 ;//indica il numero di millisecondi che passano tra l'invio di
un  //pacchetto con il successivo al server

   int flag;//Controlla se la connessione è avvenuta con successo

   /*
   Crea un oggetto SerialConnection e inizializza le variabili passate come parametri
   parent= Oggetto PocketVisit.
   parameters= Un oggetto SerialParameters.
   messageAreaIn= Il TextArea che visualizza i messaggi in ingresso dalla porta seriale
   */
   /*Nel messageAreaOut ci sono i messaggi che vengono mandati fuori la porta seriale*/
   public SerialConnection(PocketVisit parent, /*COSTRUTTORE che accetta un oggetto
PocketVisit,
                                i parametri, la TextArea di input(Area dove i
                                mex arrivano) e la TextArea di output */
            SerialParameters parameters,
            TextArea messageAreaOut,
            TextArea messageAreaIn) {
   old=new String("");
   flag=0;
   this.parent = parent; //Inizializza parent a parent di tipo SerialDemo
   this.parameters = parameters;
//    this.messageAreaOut = messageAreaOut;
   this.messageAreaIn = messageAreaIn;
   open = false; //è un booleano

  }

   /*Tenta di aprire una connessione seriale e flussi usando i parametri della porta
   seriale (oggetto di tipo SerialParameters). Se la connesisone non va a buon fine,
   esso torna la porta attraverso un'eccezione*/

  public void openConnection() throws SerialConnectionException {  //Metodo che apre
//                                          una connessione

   // Ottiene un oggetto CommPortIdentifier (javax.comm) per la porta che si vuole aprire.
   try {
   /*Gestione delle porte di comunicazione. CommPortIdentifier è una classe centrale per
  il controllo all'accesso alle porte di comunicazione.*/
      portId =

CommPortIdentifier.getPortIdentifier(parameters.getPortName());/*getPortIdentifier(String)
                                          è un metodo che ottiene
                                          un oggetto CommPortIdentifier
                                          mediante il nome della porta */

   } catch (NoSuchPortException e) {
      throw new SerialConnectionException(e.getMessage());
   }

   //Apre la porta rappresentata dall'oggetto CommPortIdentifier, viene dato all'apertura della
 // chiamata un timeout relativamente lungo di 30 sec per permettere un'applicazione differente di
//     rilasciare la porta se l'utente la vuole
   try {
      sPort = (SerialPort)portId.open("PocketVisit", 30000); /*il metodo open() della classe
                                       CommPortIdentifier
                                       apre una porta di comunicazione
                                       e accetta il nome dell'applicazione
                                       e un intero che indica il timeout (msec).
                                       IL METODO OPEN OTTIENE LA COMPLETA
                                       PADRONANZA DELLA PORTA da qualche
                                       altra applicazione
                                       , è propagato un evento
                                       PORT_OWNERSHIP_REQUESTED usando il
                                       meccanismo dell'evento
                                       CommPortOwnershipListener.
                                       C'è un InputStream ed un outputStream
                                       associato a ogni porta. Dopo che una porta
                                       è aperta mediante il metodo open,
                                       tutte le chiamate al metodo getInputStream
                                       ritornerà lo stesso stream fino a che è
                                       chiamato il metodo close*/
```

```
    messageAreaIn.append("    ***************************\n ");
                messageAreaIn.append("    *      PocketVisit v1.0 \"Client\"   *\n ");
                messageAreaIn.append("    *         ICAR-CNR - Palermo      *\n ");
                messageAreaIn.append("    *************************** \n\n\n");
    messageAreaIn.append(" Attendere conferma connessione \n con antenna GPS....");
    messageAreaIn.append("\n Se tale conferma non dovesse essere data,\n assicurarsi che la
connessione Bluetooth \n possa avvenire.. \n\n\n ");

    } catch (PortInUseException e) { // l'evento di porta in uso lanciando un'eccezione se in uso
       throw new SerialConnectionException(e.getMessage());
    }

    // Setta i parametri della connessione, se essi non si settano, la porta viene chiusa e viene
//    lanciata un'eccezione
    try {
       setConnectionParameters(); //
    } catch (SerialConnectionException e) {
       sPort.close();
       throw e;
    }

    //Apre i flussi di ingresso e uscita per la connessione. Se essi non si aprono, viene lanciata
//    un'eccezione
    try {
//       os = sPort.getOutputStream();
       is = sPort.getInputStream();

    } catch (IOException e) {
       sPort.close();
       throw new SerialConnectionException("Errore apertura flussi I/O");
    }


    // Aggiunge questo oggetto come un ascolto di evento per la porta seriale.
    try {
       sPort.addEventListener(this);
    } catch (TooManyListenersException e) {
       sPort.close();
       throw new SerialConnectionException("troppi listeners aggiunti");
    }

    // Setta notifyOnDataAvailable a true per permettere event driven di input.
    sPort.notifyOnDataAvailable(true);

    // Setta notifyOnDataAvailable a true per permettere event driven di break.
    sPort.notifyOnBreakInterrupt(true);

    try {
       sPort.enableReceiveTimeout(30);
    } catch (UnsupportedCommOperationException e) {
    }


    portId.addPortOwnershipListener(this);

    open = true;
    }

    public void setConnectionParameters() throws SerialConnectionException{ /*Metodo di
utilità
                                                    che setta i parametri di connessione
                                                    utilizzando l'oggetto parameters.
                                                    Se il settaggio fallisce ritorna
                                                    l'oggetto parameters
                                                    ai suoi settaggi originali e
                                                    lancia l'eccezione*/

    // Salva lo stato dei parametri prima di tentare il settaggio.
    int oldBaudRate = sPort.getBaudRate(); //salva il BaudRate
    int oldDatabits = sPort.getDataBits();
    int oldStopbits = sPort.getStopBits();
    int oldParity   = sPort.getParity();
    int oldFlowControl = sPort.getFlowControlMode();

    // Setta i parametri di connessione, se il set fallisce ritorna l'oggetto parameters
    // allo stato originale.
```

```
   try {
      sPort.setSerialPortParams(parameters.getBaudRate(), //Setta i parametri della porta seriale
//                                                        passando tutti i parametri.Ricordiamo che
                  parameters.getDatabits(),            //sPort è un oggetto CommPortIdentifier
                  parameters.getStopbits(),
                  parameters.getParity());
   } catch (UnsupportedCommOperationException e) { //Gestisce l'eccezione di non suporto dei
parametri
      parameters.setBaudRate(oldBaudRate); //Riporta parameters al suo stato originale
      parameters.setDatabits(oldDatabits);
      parameters.setStopbits(oldStopbits);
      parameters.setParity(oldParity);
      throw new SerialConnectionException("Parametro non supportato");
   }

   // Setta il controllo di flusso dal settaggio contenuto in parameters
   try {
      sPort.setFlowControlMode(parameters.getFlowControlIn()
                  | parameters.getFlowControlOut());
   } catch (UnsupportedCommOperationException e) { //lancia l'eccezione se non supportato
      throw new SerialConnectionException("Flusso di controllo non supportato");
   }
   }//Fine metodo setConnectionparameters

   /**
   Chiude la porta e pulisce gli elementi associati.
   */
   public void closeConnection() { //Metodo che chiude la connessione
   // se la porta è già chiusa esce dalla funzione.
   flag=0;
   if (!open) {
      return;
   }

   // Contolla che sPort ha referenza per evitare un NPE.
   if (sPort != null) {//Se sPort si riferisce ad una porta
      try {
      // chiudi gli i/o streams.
//          os.close(); //Chiude gli Stream in ingresso e uscita
         is.close();
      } catch (IOException e) {
      System.err.println(e);
      }

      // chiude la porta.
      sPort.close(); //Chiude la porta

      // Rimuove l'ascoltatore di padronanza della porta (ownership).
      portId.removePortOwnershipListener(this);
   }

   open = false; //Dichiara la porta chiusa
   }// fine metodo closeConnection

   /*
   Rapporta lo stato di apertura della porta.
   ritorna true se la porta è aperta, false se la porta è chiusa.
   */
   public boolean isOpen() {
   return open;
   }

   /* Testa gli eventi SerialPortEvents. I due tipi di SerialPortEvents che questo programma è
capace
    di ascoltare sono DATA_AVAILABLE e BI. Durante il DATA_AVAILABLE il buffer della
porta è letto
     fino al suo svutamento, quando non sono disponibili più dati e sono passati 30 msec il
metodo
      ritorna. Quando un evento BI accade, la parola BREAK RECEIVED è scritta nel
messageAreaIn. */

   public void serialEvent(SerialPortEvent e) {
   // Crea una StringBuffer e int per ricevere dati in input.
   StringBuffer inputBuffer = new StringBuffer();
   int newData = 0;
   String toSend;
   String toSent=new String("");
```

```
    String confronto=new String("$GPGGA");
    String confronto1=new String("$GPGLL");
    String confronto2=new String("$GPRMC");

    // determina il tipo di evento.
    //if(!e.getEventType()) System.out.println("Non connesso");
    switch (e.getEventType()) {

        // Legge dati fino a quando è ritornato -1. Se è ricevuto \r sostituisce
        // \n per un corretto maneggio dell newline.
        case SerialPortEvent.DATA_AVAILABLE:
            while (newData != -1) {
                try {
                    newData = is.read();
                if (newData == -1) {
                break;
                }
                if ('\r' == (char)newData) { //Ha finito di trasmettere la riga
//                  System.out.println(inputBuffer+"\n\n\n");
                    StringTokenizer st = new StringTokenizer(new String(inputBuffer),",");
               toSend=new String("");//Svuota la String precedente
               String tag=new String(st.nextToken());
                while(st.hasMoreTokens()){
                    if(tag.equals(confronto)){ /*Mediante queste righe si vengono presi soltanto 4
elementi:

                                a) Latitudine della posizione attuale
                                b) Emisfero della posizione attuale (Nord o Sud)
                                c) Longitudine della posizione attuale
                                d) Verso della posizione attuale (Esto o Ovest)
                                */
                                st.nextToken();
                                 for(int i=0;i<4;i++){
                                                                toSend=toSend.concat(st.nextToken()
+" ");
                                    }
                                old=new String(toSend);
                                break;
                               }else if(tag.equals(confronto1)){ /*Mediante queste righe si vengono presi
soltanto 4 elementi:
                                a) Latitudine della posizione attuale
                                b) Emisfero della posizione attuale (Nord o Sud)
                                c) Longitudine della posizione attuale
                                d) Verso della posizione attuale (Esto o Ovest)


   */
                                 for(int i=0;i<4;i++){
                                                                toSend=toSend.concat(st.nextToken()
+" ");
                                    }
                                //System.out.println(toSend);
                                old=new String(toSend);
                                break;
            }else if(tag.equals(confronto2)){ /*Mediante queste righe vengono presi soltanto 4
elementi:

                                a) Latitudine della posizione attuale
                                b) Emisfero della posizione attuale (Nord o Sud)
                                c) Longitudine della posizione attuale
                                d) Verso della posizione attuale (Esto o Ovest)
                                */
                                 st.nextToken();
                                 st.nextToken();
                                 for(int i=0;i<4;i++){
                                                                toSend=toSend.concat(st.nextToken()
+" ");

                                    }
                                //System.out.println(toSend);
                                old=new String(toSend);
                                break;
            }
            else{
                                toSend=new String(old);
//                              System.out.println("Continua ciclo");
                                break;
           }
```

```
            }
            toSent=toSend;
            //StringTokenizer s=new StringTokenizer(toSend);
            //while(s.hasMoreElements())
//              System.out.println(toSent);
            //System.out.println(toSend);//Non si sa perchè ma toSend è formato da 4 righe uguali; a
noi interessa soltanto la prima riga
              inputBuffer.append('\n');
                                                    //Bisogna spedire al Server lo la Stringa toSend
che contiene le coordinate del client, poi il server se lo gestisce

              } else {
                 inputBuffer.append((char)newData);
              }
              } catch (IOException ex) {
                 System.err.println(ex);
                 return;
              }

           }
         try{
                                        conn(toSent);

                                       }
        // Append received data to messageAreaIn.
        finally{
//        messageAreaIn.append(new String(inputBuffer)); //Appende i dati in ingresso nel
//                                         messageAreaIn
        break;
                               }

    }

}

    /* Se è ricevuto un evento PORT_OWNERSHIP_REQUESTED è creato un box di dialogo
domandando
     all'utente se essi sono disposti a cedere la porta*/

    public void ownershipChange(int type) {
    if (type == CommPortOwnershipListener.PORT_OWNERSHIP_REQUESTED) {
       PortRequestedDialog prd = new PortRequestedDialog(parent);
    }
    }

/**
                 * Metodo che invia la posizione che riceve come parametro in ingresso al Server.
                 *
                 * @param arg Rappresenta la posizione del client
                 * @throws Exception
                 * @throws IOException
                 */
                public   void conn(String arg) throws Exception, IOException{

                                                                        if (flag==0){

    messageAreaIn.append("********************************** \n\n\n");


    messageAreaIn.append("Benvenuti su PocketVisit v1.0 \n for Pocket PC! \n \n Connessione
all'antenna\n GPS TOMTOM \n avvenuta con successo. \n \n Adesso è possibile aprire\n Pocket
Internet Explorer! ...\n\n");


    flag++;
                                                                                }
                                                                        DatagramSocket
datSocket;
                                        byte[] data=new byte[1024];
                                        data =arg.getBytes(); //Converte il messaggio in
un array di byte
                                        datSocket=new DatagramSocket(9998);
                                        //crea il pacchetto
                                        InetAddress
address=InetAddress.getByName("147.163.3.205"); //
    InetAddress myAddress=InetAddress.getByName("Pocket_Pc_1.mshome.net");
                                        DatagramPacket sendPacket= new
DatagramPacket(data, data.length,/*InetAddress.getLocalHost()*/address,9999);
```

```
                                                datSocket.send(sendPacket);
                                                datSocket.close();
                                                //messageAreaIn.append("Pacchetto Spedito a " +
address.getHostAddress() +" contenente: " + arg + "\n");
//                                              System.out.println("Io ho l'indirizzo: " +
myAddress.getHostAddress());
                                                //si invia il  pacchetto al server ogni intervallo
secondi
                                                Thread.sleep(intervallo);

                                        }
}
```

A.2 Applicazione setPOI vers. 1.1

A.2.1 setPOI.java

```
package setPOIs;
import javax.comm.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
import java.text.DecimalFormat;
import java.net.*;



/**
 *
 * Classe interfaccia che permette di inserire negli appositi campi la tipologia e il nome
 * del POIe cliccando il bottone 'Salva' si inviano i dati al database affinchè avviene la
 * registrazione del POI
 * @author Pietro Amato
 *
 */
public class setPOIs extends Frame implements ActionListener {

    final int HEIGHT = 500;
    final int WIDTH  = 460;

    private MenuBar mb;
    private Menu fileMenu;
    private Menu helpMenu;

    private MenuItem exitItem;
    private MenuItem aboutItem;

    private Button openButton;
    private Button stopButton;
    private Button addButton;

    private Panel buttonPanel;

    private Panel messagePanel;

    private TextArea messageAreaOut;
    private TextArea messageAreaIn;

    private ConfigurationPanel configurationPanel;
    private SerialParameters parameters;
    private SerialConnection connection;

    private Properties props = null;


    public static void main(String[] args) {

   setPOIs set_pois = new setPOIs();
   set_pois.setVisible(true);
   set_pois.repaint();
```

```
}


 public setPOIs(){
super("setPOIs for PocketVisit v1.0");

parameters = new SerialParameters();
parameters.setBaudRate("4800");


addWindowListener(new CloseHandler(this));

mb = new MenuBar();

fileMenu = new Menu("File");
helpMenu = new Menu("Help");

exitItem = new MenuItem("Exit");
exitItem.addActionListener(this);
fileMenu.add(exitItem);

aboutItem=new MenuItem("About PocketVisit");
aboutItem.addActionListener(this);
helpMenu.add(aboutItem);


mb.add(fileMenu);
mb.add(helpMenu);

setMenuBar(mb);

messagePanel = new Panel();
messagePanel.setLayout(new GridLayout(1, 1));

messageAreaOut=new TextArea();

messageAreaIn = new TextArea();
messageAreaIn.setEditable(false);
messagePanel.add(messageAreaIn);

add(messagePanel, "Center");

configurationPanel = new ConfigurationPanel(this);

buttonPanel = new Panel();

openButton = new Button("Vai>>");
openButton.addActionListener(this);
buttonPanel.add(openButton);

stopButton = new Button("<<Ferma");
stopButton.addActionListener(this);
stopButton.setEnabled(false);

buttonPanel.add(stopButton);

addButton=new Button("Salva POI");
addButton.addActionListener(this);
addButton.setEnabled(false);
buttonPanel.add(addButton);

Panel southPanel = new Panel();

GridBagLayout gridBag = new GridBagLayout();
GridBagConstraints cons = new GridBagConstraints();

southPanel.setLayout(gridBag);

cons.gridwidth = GridBagConstraints.REMAINDER;
gridBag.setConstraints(configurationPanel, cons);
cons.weightx = 1.0;
southPanel.add(configurationPanel);
gridBag.setConstraints(buttonPanel, cons);
southPanel.add(buttonPanel);

add(southPanel, "South");
```

```
   connection = new SerialConnection(this, parameters,
                                                messageAreaOut, messageAreaIn);
   setConfigurationPanel();

   Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

   setLocation(/*screenSize.width/9*/ 5,
                    /*screenSize.height/9*/ 25);

   setSize(WIDTH/2 , HEIGHT/2 );
   }


   public void setConfigurationPanel() {
   configurationPanel.setConfigurationPanel();
   }

   public void actionPerformed(ActionEvent e) {
   String cmd = e.getActionCommand();

   if (cmd.equals("Exit")) {
      shutdown();
   }

   if (cmd.equals("About PocketVisit")){
                        AlertDialog al=new AlertDialog(this, "About PocketVisit", "setPOIs for
Pocket Visit v1.0", "Proprietà di: ICAR di Palermo", "Istituto di Calcolo e Reti ad alte
prestazioni");
                }

   if (cmd.equals("Vai>>")) {
      openButton.setEnabled(false);
      Cursor previousCursor = getCursor();
      setNewCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
      configurationPanel.setParameters();
      try {
                connection.openConnection();
      } catch (SerialConnectionException e2) {
                AlertDialog ad = new AlertDialog(this,
                                                "Error Opening Port!",
                                                "Error opening port,",
                                                e2.getMessage() + ".",
                                                "Select new settings, try again.");
         openButton.setEnabled(true);
                setNewCursor(previousCursor);
                return;
      }
      portOpened();
      setNewCursor(previousCursor);
   }


   if (cmd.equals("<<Ferma")) {
      portClosed();
   }

   if(cmd.equals("Salva POI")){
                try{
                salva(); //Questo metodo deve aprire il file esistente di temp.txt dove vi è
memorizzata la posizione corrente ricevuta dall'antennina
                                //leggere e memorizzare il contenuto del file einviarlo al
serverAmministrazione con la denominazione e la tipologia del POI
   }catch(IOException e4){}

   }

                }


   public void portOpened() {
   openButton.setEnabled(false);
   stopButton.setEnabled(true);
   addButton.setEnabled(true);
   configurationPanel.denField.setEditable(true);
```

```
}


public void portClosed() {
connection.closeConnection();
openButton.setEnabled(true);
stopButton.setEnabled(false);
addButton.setEnabled(false);
  configurationPanel.denField.setEditable(false);

}


private void setNewCursor(Cursor c) {
setCursor(c);
messageAreaIn.setCursor(c);
messageAreaOut.setCursor(c);
}

private void shutdown() {
connection.closeConnection();
System.exit(1);
}



class ConfigurationPanel extends Panel implements ItemListener {

private Frame parent;
private Label den; //denominazione del POI da aggiungere
private Label tip;
private Label portNameLabel;
private Choice portChoice;//tipologia del POI da aggiungere

public TextField denField;
public Choice tipChoice;


public ConfigurationPanel(Frame parent) {
   this.parent = parent;

   setLayout(new GridLayout(3, 3));

   den=new Label("POI Denominazione:",Label.LEFT);
   add(den);

   denField=new TextField("",20);
   denField.setEditable(false);
   add(denField);

   tip=new Label("POI Tipologia:",Label.LEFT);
   add(tip);

   tipChoice = new Choice();
   tipChoice.add("Itinerario");
   tipChoice.add("Interesse");
   tipChoice.addItemListener(this);
   add(tipChoice);

   portNameLabel = new Label("Port Name:", Label.LEFT);
   add(portNameLabel);

   portChoice = new Choice();
   portChoice.addItemListener(this);
   add(portChoice);
   listPortChoices();
   portChoice.select(parameters.getPortName());

}


public void setConfigurationPanel() {
   portChoice.select(parameters.getPortName());

}
```

```
    public void setParameters() {
        parameters.setPortName(portChoice.getSelectedItem());

    }


    void listPortChoices() {
        CommPortIdentifier portId;

        Enumeration en = CommPortIdentifier.getPortIdentifiers();

        // iterate through the ports.
        while (en.hasMoreElements()) {
            portId = (CommPortIdentifier) en.nextElement();
            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                portChoice.addItem(portId.getName());
            }
        }
        portChoice.select(parameters.getPortName());
    }


  public void itemStateChanged(ItemEvent e) {

    if (connection.isOpen()) {

            if (e.getItemSelectable() == portChoice) {

                AlertDialog ad = new AlertDialog(parent, "Porta aperta!",
                                                         "La porta non può essere",
                                                         "cambiata",
                                                         "mentre una porta è aperta.");


                setConfigurationPanel();
                return;
            }

            setParameters();
            try {

                connection.setConnectionParameters();
            } catch (SerialConnectionException ex) {

                AlertDialog ad = new AlertDialog(parent,
                        "Configurazione non supportata!",
                        "Parametro di configurazione non supportato,",
                        "seleziona un nuovo valore.",
                        "Ritorno alla precedente configurazione.");
                setConfigurationPanel();
            }
    } else {

            setParameters();
    }
  }
  }


  class CloseHandler extends WindowAdapter {

  setPOIs sd;

  public CloseHandler(setPOIs sd) {
    this.sd = sd;
  }

  public void windowClosing(WindowEvent e) {
    sd.shutdown();
  }
  }

/**
 * Metodo della classe che permette di inviare i dati al ServerAmministrazione relativi ad un POI
 * @throws IOException
 * @throws FileNotFoundException
 */
```

```
private void salva() throws IOException, FileNotFoundException{

      //Legge le coordinate correnti, la denominazione dal TextField e la tipologia dal Choice
      BufferedReader                  filebuf =
          new BufferedReader(new FileReader("temp.txt"));
      String posizione = filebuf.readLine(); //legge coordinate nuovo POI
      filebuf.close();  // chiude il file

      String newDen=configurationPanel.denField.getText();
      String newTip=configurationPanel.tipChoice.getSelectedItem();

String message = posizione+" "+newTip+" "+newDen;

//si invia il messaggio al ServerAmministrazione

DatagramSocket datSocket;
byte[] data=new byte[1024];
data =message.getBytes(); //Converte il messaggio in un array di byte
datSocket=new DatagramSocket(9999);

//crea il pacchetto
//al posto dell'indirizzo locale bisogna inserire quello del serverAmministrazione
InetAddress address=InetAddress.getByName("127.0.0.1");
//InetAddress myAddress=InetAddress.getByName("Pocket_Pc_1.mshome.net");

DatagramPacket sendPacket= new DatagramPacket(data,
data.length,/*InetAddress.getLocalHost()*/address,10000);
datSocket.send(sendPacket);


byte[] data1=new byte[2];
DatagramPacket receivePacket = new DatagramPacket(data1,data1.length );
datSocket.receive(receivePacket);
String messaggio = new String(receivePacket.getData());

if( messaggio.equalsIgnoreCase("ok")  ){
   //si visualizza un JoptopnPane che avviasa l'utente dell'avvenuta registrazione
    AlertDialog ad = new AlertDialog(this, "POI registrato","La registrazione è",
                    "avvenuta",
                    "con successo.");
}

else{
   //si visualizza un JOptionPane dell'errore
   AlertDialog ad = new AlertDialog(this, "POI non registrato","Errore durante",
                    "l'esecuzione",
                    "della registrazione.");
}


datSocket.close();
      }
     }
```

A.3 Applicazione AdministrationServer

A.3.1 ServerAmministrazione.java

```
package ServerAmministrazione;

import java.net.*;
import java.io.*;
import java.util.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.text.*;
```

```
/**
 * Classe che implementa un server UDP che sta in ascolto alla porta 10000.
 * Serve per ricevere i pacchetti che contengono le informazioni dei POIs,
 * ovvero la latitudine,longitudine,denominazione ed opzionalmente anche la tipologia;
 * una volta ottenuto il pacchetto effettua la connessione con il database e opera la
 * query di inserimento di un nuovo POI.
 *
 * @author Pietro Amato
 *
 *
 */

public class ServerAmministrazione{
   /**
    * Socket  in ascolto alla porta 10000
    *
    */
   private DatagramSocket datSocket;

   /**
    * Attributo della classe utilizzato per memorizzare la latitudine espressa in gradi del POI
    */
   double latitudine;
   /**
    * Attributo della classe utilizzato per memorizzare la longitudine espressa in gradi del POI
    */
   double longitudine;
   /**
    * Attributo della classe utilizzato per memorizzare l'eventuale tipologia del POI
    */
   String tipologia ;
   /**
    * Attributo della classe utilizzato per memorizzare la denominazione del POI
    */
   String denominazione ;
   /**
    * @see java.sql.Connection
    */
   private Connection con;
   /**
    * @see java.sql.Statement
    */
   private Statement stmt;
   /**
    * Attributo della classe di tipo String;rappresenta url del database
    */
   static String url ="jdbc:mysql://localhost/entedbguida";
   /**
    * Attributo della classe di tipo String;rappresenta il nome del driver del database
    */
   static String driver = "com.mysql.jdbc.Driver";
/**
    * Metodo che gestisce l'attesa dei pacchetti da parte dei client
    * @throws UnknownHostException
    */
public void waitForPacket() throws UnknownHostException{

                try{
                                                datSocket= new DatagramSocket(10000);
                                                System.out.println("Server UDP in ascolto alla
porta 10000");
                                        }
                                        catch(SocketException e){
                                                        e.printStackTrace();

                                                System.exit(1);
                                                }

                                                while(true){
                                                        //Riceve pacchetto, visualizza
contenuto e salva in un file

                                                        try{
                                                                //imposta pacchetto
                                                                byte data[]=new byte[100];
                                                                DatagramPacket
receive=new DatagramPacket(data, data.length);
```

```
   datSocket.receive(receive); //attende il pacchetto
                                                                        //synchronized(this){
                                                                        //Controlla a quale client
appartiene il pacchetto

//LA STRINGA DI SOTTO E' DA METTERE, ERA STATA SOSTITUITA CON LA
SUCCESSIVA PER FARE LE PROVE IN MACCHINA LOCALE
                                                                        String
IPHost=(receive.getAddress()).toString().substring(1);//Indirizzo IP del client
                                                                        //String IPHost=new
String("127.0.0.1");
                                                                        //Visualizza informazioni
sul pacchetto ricevuto

   System.out.println("Pacchetto ricevuto da:\n" +
                                                                        "Host: " + IPHost +"\n" +
                                                                        "Contenente la Stringa: " +
new String(receive.getData(),0,receive.getLength()));

                                        synchronized(this){//una volta iniziata la sessione viene
eseguita senza essere interrotta
                                                                        String str=new
String(receive.getData(),0,receive.getLength()); //Stringa contenente le coordinate del client
                                                                        if(str==null || str.equals(""))
continue; //Non dovrebbe mai accadere, ma se dovesse accadere lo ignora
      //Si suddivide in token str ottenendo 4 tokens
      StringTokenizer divided=new StringTokenizer(str);

//lat è la stringa xxxx.xxxx
      String lat=divided.nextToken();

      //Si suddivide ulteriormente in token con argomento "."
      StringTokenizer latitude=new StringTokenizer(lat,"."); //Si ottengono i token xxxx e xxxx

      //Il token xxxx lo si deve dividere in due parti xx xx corrispondenti ai gradi ed ai minuti
primi
      String temp=new String(latitude.nextToken());
      String latgrad=temp.substring(0,2);
      String latprimi=temp.substring(2);
      double gradi=Double.parseDouble(latgrad);//Gradi
      double minutiPrimi=(Double.parseDouble(latprimi));//Minuti primi
      double millesimiPrimi=(Double.parseDouble(latitude.nextToken()))/10000;//millesimi di
minuti primi
      double decGrad=(minutiPrimi+millesimiPrimi)/60;
      latitudine=gradi+decGrad; //Latitudine dell'antenna GPS espressa in gradi


//    si elimina il token relativo alla direzione della latitudine (N = Nord,S = Sud)
      divided.nextToken();

//La stessa cosa per la longitudine
//lon è la stringa xxxxx.xx
      String lon=divided.nextToken();

      //Si suddivide ulteriormente in token con argomento "."
      StringTokenizer longitude=new StringTokenizer(lon,"."); //Si ottengono i token xxxxx e xx

       //Il token xxxxx lo si deve dividere in due parti xxx xx corrispondenti ai gradi ed ai minuti
primi
      String temp2=new String(longitude.nextToken());
      String longrad=temp2.substring(0,3);
      String lonprimi=temp2.substring(3);
      gradi=Double.parseDouble(longrad);//Gradi
      minutiPrimi=(Double.parseDouble(lonprimi));//Minuti primi
      millesimiPrimi=(Double.parseDouble(longitude.nextToken()))/10000;//millesimi di primi
      decGrad=(minutiPrimi+millesimiPrimi)/60;
      longitudine=gradi+decGrad;//longitudine dell'antenna GPS espressa in gradi


//    si elimina il token relativo alla direzione della longitudine (E = Est,O = Ovest)
      divided.nextToken();

      //si recupera la tipologia del POI
      tipologia = divided.nextToken();

      //si recupera la denominazione del POI
```

```
    try{
            denominazione = divided.nextToken();
     }
//gestisce la possibilità che la denominazione non venga inserita,non dovrebbe accadere,se
accade la denominazione viene settata a null
     catch(NoSuchElementException ne){
            denominazione = null;
     }

     /*
     * Per convenienza le coordinate sono espresse in gradi con 5 cifre di precisione
     */
     DecimalFormat fiveDigits=new DecimalFormat("0.00000");
     latitudine=Double.parseDouble(fiveDigits.format(latitudine).replace(',','.'));
     longitudine=Double.parseDouble(fiveDigits.format(longitudine).replace(',','.'));

//       System.out.println(latitudine+"   "+longitudine);

     try {

            java.sql.Timestamp oraRilevazione = new Timestamp((new
java.util.Date()).getTime());
            String ora = oraRilevazione.toString();
            Class.forName(driver);

            con = DriverManager.getConnection(url,"client","client");
            stmt = con.createStatement();
            int num  = stmt.executeUpdate("INSERT INTO
POI(latitudine,longitudine,data,tipologia,denominazione)
VALUES('"+latitudine+"','"+longitudine+"','"+ora+"','"+tipologia+"','"+denominazione+"' )");



            //          ritorna al client un messaggio che avvisa dell'avvenuta registrazione
del POI


            //          IPAddress contiene l'indirizzo del client
                  InetAddress IPAddress = receive.getAddress();

                  //          port e' il numero di porta del client
                  int port = receive.getPort();


                  //          ack sara' il contenuto del pacchetto che inviamo al client
                  String ack = new String("ok");

                  //          convertiamo ack da string a byte[] e lo assegniamo a
sendData
                  byte[]      sendData = ack.getBytes();

                  //          si costruisce il pacchetto da inviare
                  DatagramPacket sendPacket = new DatagramPacket(sendData
,sendData.length ,

  IPAddress ,port );

                  //          si invia il pacchetto
                  datSocket.send(sendPacket);

     }
            catch (Exception e){
                  System.err.println("Errore");
                  //ritorna un messaggio della non avvenuta registrazione del POI
                  //IPAddress contiene l'indirizzo del client
                        InetAddress IPAddress = receive.getAddress();

                        //port e' il numero di porta del client
                        int port = receive.getPort();

                        //posizione sara' il contenuto del pacchetto che inviamo al
client
                        String nack = new String("error");

                        //convertiamo posizione da string a byte[] e lo assegniamo a
sendData
                        byte[]      sendData = nack.getBytes();
```

```
                                        //si costruisce il pacchetto da inviare
                                        DatagramPacket sendPacket = new
DatagramPacket(sendData ,sendData.length ,


   IPAddress ,port );

                                        //si invia il pacchetto
                                        datSocket.send(sendPacket);
                }

                try {

                        stmt.close();
                   con.close();
                   System.out.println("Chiusura connessione");

                } catch (SQLException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                }

                        }

                                                                }
                                                                catch(IOException e2){

   System.out.print(e2.toString());
                                                                        e2.printStackTrace();
                                                                }

                                                        }

                                        }

/**
 * Metodo main della classe
 * @param args
 * @throws IOException
 */
public static void main (String args[]) throws IOException {

                                                new ServerAmministrazione().waitForPacket();

                                        }
                        }
```

A.4 Applicazione Server

A.4.1 Server.java

```
package Server;
import java.net.*;
import java.io.*;



/**
 * Classe che implementa un server UDP che sta in ascolto alla porta 9999.
 * Riceve i pacchetti che contengono la posizione del client.
 *
 * @author Pietro Amato
 *
 */

public class Server{

   /**
    * Attributo della classe Server di tipo DatagramSocket.
    *
```

```
 */
private DatagramSocket datSocket;



/**
 * Attributo della classe di tipo ConnectionPool,rappresenta il pool di
 * connessioni che viene utilizzato per gestire le connessioni al database
 */
ConnectionPool conPool = null;

public void waitForPacket() throws UnknownHostException{

        try{
                datSocket= new DatagramSocket(9999);
                System.out.println("Server UDP in ascolto alla
porta 9999");
        }
        catch(SocketException e){
                e.printStackTrace();

                System.exit(1);
                }
        try {
                conPool = ConnectionPool.getConnectionPool();
        } catch (ConnectionPoolException e1) {
                // TODO Auto-generated catch block
                System.out.println("errore generato dal pool");
        }
                while(true){

                        byte data[]=new byte[100];
                        DatagramPacket receive=new
DatagramPacket(data, data.length);
                        try {
                                datSocket.receive(receive);
                        } catch (IOException e2) {
                                // TODO Auto-generated
catch block
                                e2.printStackTrace();
                        }
                        new
thread(datSocket,receive,conPool).start();

                }

        }
    public static void main (String args[]) throws IOException {

                new Server().waitForPacket();

        }
    }
```

A.4.2 ConnectionPool.java

```
package Server;
import java.util.*;
import java.sql.*;

/**
 * Classe che gestisce un pool di connessioni al database in maniera da
 * migliorare le prestazioni poichè si risparmi il tempo di connessione al database
 * se già c'è una connessione libera
 *
 * @author Pietro Amato
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class ConnectionPool {
/**
 * Variabile che gestisce l'unica istanza di ConnectionPool
 */
```

```
private static ConnectionPool connectionPool = null;
/**
 * Vettore che memorizza le connessioni libere al database
 */
private Vector freeConnections;
  /**
  * Stringa che memorizza l'url del database
  */
private String dbUrl = "jdbc:mysql://localhost/entedbguida";
/**
 * Parametro che memorizza il driver del database
 */
private String dbDriver= "com.mysql.jdbc.Driver";
  /**
   * Costruttore della classe ConnectionPool
   * @throws ConnectionPoolException
   */
private ConnectionPool() throws ConnectionPoolException {

    freeConnections = new Vector();  // Costruisce la coda delle connessioni libere
    loadDriver();                                                    // Carica il driver del
database
  }

   // .
/**
 * Metodo privato che carica il driver per l'accesso al database.
 * In caso di errore durante il caricamento del driver solleva
 * un'eccezione di tipo ConnectionPoolException
 * @throws ConnectionPoolException
 */
private void loadDriver() throws ConnectionPoolException {
    try {
      java.lang.Class.forName(dbDriver);
    }
    catch (Exception e) {
      throw new ConnectionPoolException();
    }
  }

/**
 * Metodo statico e sincronizzato che ritorna un oggetto di tipo ConnectionPool
 * che è stato appena creato.
 * In caso d'errore si ha un'eccezione di tipo ConnectionPoolException
 * @return
 * @throws ConnectionPoolException
 */
public static synchronized ConnectionPool getConnectionPool()
  throws ConnectionPoolException {

    if(connectionPool == null) {
      connectionPool = new ConnectionPool();
    }
    return connectionPool;
  }

  // Il metodo getConnection restituisce una connessione libera prelevandola
  // dalla coda freeConnections oppure se non ci sono connessioni disponibili
  // creandone una nuova con una chiamata a newConnection
/**
 * Metodo che restituisce una connessione libera prelevandola dalla coda
 * freeConnections o se non ci sono connessioni disponibili ne crea una
 * nuova con una chiamata a newConnection
 * @return Oggetto di tipo Connection che rappresenta la connessione libera
 * @throws ConnectionPoolException
 */
public synchronized Connection getConnection() throws ConnectionPoolException {

    Connection con;

    if(freeConnections.size() > 0) {      // Se la coda delle connessioni libere non è vuota
      con = (Connection)freeConnections.firstElement();  // Preleva il primo elemento
      freeConnections.removeElementAt(0);                // e lo cancella dalla coda
      try {
        if(con.isClosed()) {          // Verifica se la connessione non è più valida
          con = getConnection();    // Richiama getConnection ricorsivamente
        }
```

```
      }
      catch(SQLException e) {           // Se c'è un errore
        con = getConnection();        // richiama getConnection ricorsivamente
      }
    }
    else {                            // se la coda delle connessioni libere è vuota
      con = newConnection();            // crea una nuova connessione
    }
    return con;                       // restituisce una connessione valida
  }

  // Il metodo newConnection restituisce una nuova connessione
/**
 *
 * Metodo che crea una nuova connessione con il database e la ritorna
 * @return ggetto di tipo Connection che rappresenta la connessione appena creata
 * @throws ConnectionPoolException
 */
private Connection newConnection() throws ConnectionPoolException {
    Connection con = null;

    try {
      con = DriverManager.getConnection(dbUrl,"client","client");  // crea la connessione
    }
    catch(SQLException e) {                     // in caso di errore
      throw new ConnectionPoolException();      // solleva un'eccezione
    }
    return con;                               // restituisce la nuova connessione
  }

  // Il metodo releaseConnection rilascia una connessione inserendola
  // nella coda delle connessioni libere
/**
 * Metodo che rialascia una connessione che riceve come parametro e la inserisce
 * nella coda delle connessioni libere ovvero in freeConnections
 * @param con Rappresenta la connessione da liberare e da inserire in freeConnections
 */
public synchronized void releaseConnection(Connection con) {
    freeConnections.add(con);   // Inserisce la connessione nella coda
  }
}
```

A.4.3 thread.java

```
package Server;
import java.net.*;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.Timestamp;
import java.text.DecimalFormat;
import java.util.StringTokenizer;


/**
 * Classe thread che gestisce il servizio richiesto dal client ovvero quello di
 * effettuare l'inserimento della posizione del client nel database
 *
 * @author Pietro Amato
 *
 */
public class thread extends Thread
{
   /**
    * Parametro di tipo Connection
    */
   private Connection con = null;
   /**
    * Parametro di tipo DatagramPacket
    */
   private DatagramPacket receive;
   /**
    * Parametro di tipo ConnectionPool
    *
    */
```

```
 private ConnectionPool conPool;
 /**
  * Latitudine del client corrente
  */
 double latitudine; //Latitudine corrente dell'antenna GPS espressa in gradi
 /**
  * Longitudine del client corrente
  */
 double longitudine;//Longitudine corrente dell'antenna GPS espressa in gradi

 /**
  * Parametro di tipo Statement
  */
 private Statement stmt ;

 /**
  * Parametro di tipo DatagramSocket
  */
 DatagramSocket sock;


/**
 * Costruttore che accetta la DatagramSocket gia' creata
 */
public thread(DatagramSocket s,DatagramPacket receive1,ConnectionPool conPool1)
{
  conPool = conPool1;
  sock=s;
  receive = receive1;
}

/*
 * Metodo run che gestisce l'esecuzione del servizio richiesto dal client
 */
public void run()
{

        //Riceve pacchetto, visualizza contenuto e salva in un file

        try{

                //imposta pacchetto
                //byte data[]=new byte[100];
                //DatagramPacket receive=new DatagramPacket(data, data.length);
                //sock.receive(receive); //attende il pacchetto
                //synchronized(this){
                //Controlla a quale client appartiene il pacchetto

//LA STRINGA DI SOTTO E' DA METTERE, ERA STATA SOSTITUITA CON LA
SUCCESSIVA PER FARE LE PROVE IN MACCHINA LOCALE
                String IPHost=(receive.getAddress()).toString().substring(1);//Indirizzo
IP del client
//              String IPHost=new String("127.0.0.1");

                //Visualizza informazioni sul pacchetto ricevuto
                System.out.println("Pacchetto ricevuto da:\n" +
                "Host: " + receive.getAddress().toString().substring(1)+"\n" +
                "Contenente la Stringa: " + new
String(receive.getData(),0,receive.getLength()));

                synchronized(this){//una volta iniziata la sessione viene eseguita senza
essere interrotta
                        String str=new
String(receive.getData(),0,receive.getLength()); //Stringa contenente le coordinate del client
                        if(str==null || str.equals("")) ; //Non dovrebbe mai accadere,
ma se dovesse accadere lo ignora

                        //Si suddivide in token str ottenendo 4 tokens
                        StringTokenizer divided=new StringTokenizer(str);

                        //lat è la stringa xxxx.xxxx
                        String lat=divided.nextToken();

                        //Si suddivide ulteriormente in token con argomento "."
                        StringTokenizer latitude=new StringTokenizer(lat,"."); //Si
ottengono i token xxxx e xxxx
```

```
                                        //Il token xxxx lo si deve dividere in due parti xx xx
corrispondenti ai gradi ed ai minuti primi
                                        String temp=new String(latitude.nextToken());
                                        String latgrad=temp.substring(0,2);
                                        String latprimi=temp.substring(2);
                                        double gradi=Double.parseDouble(latgrad);//Gradi
                                        double minutiPrimi=(Double.parseDouble(latprimi));//Minuti
primi
                                        double
millesimiPrimi=(Double.parseDouble(latitude.nextToken()))/10000;//millesimi di minuti primi
                                        double decGrad=(minutiPrimi+millesimiPrimi)/60;
                                        latitudine=gradi+decGrad; //Latitudine dell'antenna GPS
espressa in gradi

                                        //La stessa cosa per la longitudine
                                        divided.nextToken();//Emisfero della posizione attuale(indica
il verso di positività degli assi cartografici

                                        //lon è la stringa xxxxx.xx
                                        String lon=divided.nextToken();

                                        //Si suddivide ulteriormente in token con argomento "."
                                        StringTokenizer longitude=new StringTokenizer(lon,"."); //Si
ottengono i token xxxxx e xx

                                        //Il token xxxxx lo si deve dividere in due parti xxx xx
corrispondenti ai gradi ed ai minuti primi
                                        String temp2=new String(longitude.nextToken());
                                        String longrad=temp2.substring(0,3);
                                        String lonprimi=temp2.substring(3);
                                        gradi=Double.parseDouble(longrad);//Gradi
                                        minutiPrimi=(Double.parseDouble(lonprimi));//Minuti primi

  millesimiPrimi=(Double.parseDouble(longitude.nextToken()))/10000;//millesimi di primi
                                        decGrad=(minutiPrimi+millesimiPrimi)/60;
                                        longitudine=gradi+decGrad;//longitudine dell'antenna GPS
espressa in gradi

                                        /*
                                         * Per convenienza le coordinate sono espresse in gradi con 5
cifre di precisione
                                         */
                                        DecimalFormat fiveDigits=new DecimalFormat("0.00000");

  latitudine=Double.parseDouble(fiveDigits.format(latitudine).replace(',','.'));

  longitudine=Double.parseDouble(fiveDigits.format(longitudine).replace(',','.'));

                                        try {

                                                java.sql.Timestamp oraRilevazione = new
Timestamp((new java.util.Date()).getTime());

                                                String ora = oraRilevazione.toString();

                                                con = conPool.getConnection();

                                                stmt = con.createStatement();

                                                int num  = stmt.executeUpdate("INSERT INTO
posizione(latitudine,longitudine,ora,IPturista)
VALUES('"+latitudine+"','"+longitudine+"','"+ora+"','"+IPHost+"' )");

                                        }
                                        catch (NoClassDefFoundError e){
                                                System.err.println("Errore");

                                        }

                                }//fine synchronized

                        }//fine try principale

                                catch(Exception e2){
                                        System.out.print(e2.toString());
                                        e2.printStackTrace();
                                }
```

```
                }//fine run()
}//fine class
```

A.5 Applicazione Servlet

A.5.1 HomePageServlet.java

```
package Servlet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

/**
 *
 * Servlet che mostra la pagina di default del sito dell'ente.
 * @author Pietro Amato
 *
 *
 */
public class HomePageServlet extends HttpServlet{
   /**
    *
    * Attributo costante della servlet; rappresenta l'indirizzo della macchina dove è installata
l'applicazione Servlet.
    */
   final String indirizzoServer = "http://127.0.0.1:8080/";

   /**
    * Attributo di tipo intero ;è un contatore di utenti collegati durante un giorno
    */
   int count=0; //Contatore di utenti collegati durante il giorno

   /**
    * Metodo get che gestisce le chiamate di tipo get dei client;
    * memorizza un cookie e spedisce una pagina html in cui viene
    * visualizzato un numero che indica quanti visitatori ci sono
    * stati fino a quel momento.
    * @see javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse)
    */
   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{
                count++;
                Cookie myCookie=null;
                String nome="mioCookie";
                String valore="1";
                myCookie=new Cookie(nome,valore);
                myCookie.setMaxAge(1000);
                response.addCookie(myCookie);
                PrintWriter out=response.getWriter();
                //Spedisce pagina HTML

                response.setContentType("text/html");
                String hostName=request.getRemoteHost();
                out.println("<html>");
                out.println("<head>");
                out.println("<meta http-equiv=\"Content-Language\" content=\"it\">");
                out.println("<meta http-equiv=\"Content-Type\" content=\"text/html;
charset=windows-1252\">");
                out.println("<meta name=\"GENERATOR\" content=\"Microsoft FrontPage
5.0\">");
                out.println("<meta name=\"ProgId\" content=\"FrontPage.Editor.Document\">");
                out.println("<meta name=\"Microsoft Theme\" content=\"ice 011\">");
                out.println("</head>");
                out.println("<body>");
                out.println("<p align=\"center\"></p>");
                out.println("<p align=\"center\"><b><font size=\"5\">");
//              out.println("<img border=\"0\" src=\"images/animated.gif\" align=\"left\"
width=\"48\" height=\"55\"></font><font size=\"6\">Home Page dell'ente." + "\n Oggi sei il
visitatore n." + count + "&amp;quot;</i></font></b></p>");
                out.println("</font><font size=\"6\">Home Page dell'ente." + "\n Oggi sei il
visitatore n." + count + "&amp;quot;</i></font></b></p>");
```

```
                out.println("<p align=\"center\">");
                out.println(" </p>");
                out.println("<p align=\"center\"><font size=\"1\">");
                out.println("<!--webbot bot=\"Timestamp\" S-Type=\"EDITED\" S-Format=\"%d/
%m/%Y %H.%M\" --></font></p>");
                out.println("<p align=\"center\"><b><font size=\"5\">");
//              out.println("<img border=\"0\" src=\"images/Pocket.jpg\" width=\"50\"
height=\"70\" hspace=\"30\"></font></b></p>");
                out.println("</font></b></p>");
                out.println("<p align=\"center\">");
//              out.println("<img border=\"0\" src=\"images/PocketLogo.jpg\" width=\"87\"
height=\"33\" vspace=\"-20\"></p>");
                out.println("</p>");
                out.println("<p align=\"center\"> </p>");
                out.println("<p align=\"left\">Benvenuto "+ hostName +". Ricorda che per
utilizzare questo servizio bisogna installare sul proprio palmare il software PocketVisit v1.0 for
Pocket Pc scaricabile da <b>");
                out.println("<a href=\""+ indirizzoServer
+"entedbguida/src/PocketVisit.jar\">&amp;quot;<i>qui</i>&amp;quot;.");
                out.println("</a>.");
                out.println("Nello stesso sito è possibile anche scaricare una documentazione in
formato pdf riguardo l'utilizzo dello stesso. </b></p>");
                out.println("<p align=\"left\">PocketPc v.1.0 è un software che permette di fare
conoscere al ");
                out.println("nostro web server il punto in cui sei nella mappa caricata al fine di
poterti ");
                out.println("inviare informazioni &amp;quot;automaticamente&amp;quot; riguardo ciò che
stai guardando.</p>");
                out.println("<p align=\"left\">Si può accedere al sito in due modalità: free o
guide.<b>");
                out.println("<p align=\"left\">La modalità free è quella in cui vengono visualizzate
solo informazioni mentre nella modalità guide viene guidata la visita all'interno del parco.<b>");
//              out.println("<p align=\"center\"><img border=\"0\" src=\"images/PocketLogo.jpg\"
width=\"150\" height=\"70\"></p>");
                out.println("<p align=\"center\"></p>");
                out.println("<p align=\"center\"> </p>");
                out.println("<p align=\"center\"><i><b><font size=\"5\" face=\"Palatino
Linotype\">Buona visita !!</font></b></i></p>");
                out.println("<p align=\"center\"> </p>");
                out.println("<p align=\"center\"><b>");
//      l'ho modificata per provare in locale       out.println("<a
href=\"http://169.254.69.4:8080/Segesta/SegestaServlet\">ENTRA</a></p>");
                out.println("<a href=\""+ indirizzoServer
+"entedbguida/servlet/Servlet/ServletFree\">ENTRA in modalità FREE</a></p>");
                out.println("<p align=\"center\"> </p>");
                out.println("<p align=\"center\"><b>");
                out.println("<a href=\""+ indirizzoServer
+"entedbguida/servlet/Servlet/ServletGuida\">ENTRA in modalità GUIDE</a></p>");
        out.println("<p align=\"center\"> </p>");
                out.println("<p align=\"center\"> </p>");
                out.println("<p align=\"left\"><font size=\"2\">Servizio offerto in collaborazione
con </font> <a href=\"http://www.cere.pa.cnr.it/\">");
                out.println("<font size=\"2\">ICAR ");
                out.println("Istituto di Calcolo e Reti ad Alte Prestazioni di
Palermo.</font></a></p>");
                out.println("<p align=\"center\"><b> </b></p>");
                out.println("<p align=\"center\"> </p>");
                out.println("<p align=\"center\"> </p>");
                out.println("</body>");
                out.println("</html>");
                out.close();

    }
}
```

A.5.2 ServletFree.java

```
package Servlet;

import javax.servlet.*;
```

```
import javax.servlet.http.*;
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.text.DecimalFormat;

/**
 *
 * Questa servlet serve per gestire le richieste del client.
 * Ad ogni richiesta la servlet risponde inviando una pagina HTML
 * che contiene le informazioni circa la posizione del client.
 *
 * @author Pietro Amato
 *
 *
 */

public class ServletFree extends HttpServlet{

                        /**
                         *
                         * Attributo della servlet di tipo double[][];serve per memorizzare le
                         * coordinate dei POIs in radianti;ogi riga rappresenta un POI mentre
nelle due colonne sono memorizzati
                         * la latitudine e la longitudine.
                         * E' dichiarato static poichè una volta istanziato non varia più;questo
                         * vuol dire che se si modificano i POIs,affinchè queste modifiche siano
                         * effettuate bisogna riavviare il serverweb Tomcat
                         */
                        static double POIcoordinateRadius[][]= null;
                        /**
                         *
                         * Attributo della classe costante,indica il raggio della terra in Km;
                         * ricordiamo che il raggio della terra non è costante e ha il suo massimo
pari a 6378.14 KM all'equatore
                         * e il suo minimo pari a 6356.75 Km in corrispondenza dei poli .
                         *
                         */
                        final double EARTH_RADIUS = 6371.0;

                        /**
                         *
                         * Attributo della servlet di tipo double[][];serve per memorizzare le
coordinate dei POIs
                         * in gradi,minuti,secondi ovvero nel formato in cui si ricevono le
coordinate dal GPS.
                         * E' dichiarato static poichè una volta istanziato non varia più;questo
                         * vuol dire che se si modificano i POIs,affinchè queste modifiche siano
                         * effettuate bisogna riavviare il serverweb Tomcat
                         */
                        static double POIcoordinate[][]= null;//Contiene le coordinate dei POI
                        /**
                         *
                         * Attributo della servlet di tipo double[];serve per memorizzare il raggio
di interesse di ogni
                         * POIs.
                         * POIraggio[n] ci restituisce il raggio d'inderesse dell'ennesimo POI.
                         */
                        static double POIraggio[]= null;

                        /**
                         *
                         * Attributo della servlet di tipo String[];serve per memorizzare la
denominazione dei POIS.
                         * denominazione[n] ci restituisce la denominazione dell'ennesimo POI.
                         */
                        static String denominazione[]= null;
                        /**
                         *
                         * Attributo della servlet di tipo String[];serve per memorizzare le URL
dei POI.
                         * urlPOI[n] ci restituisce l'URL dell'ennesimo POI.
                         */
```

```
                        static String urlPOI[]= null;
                        /**
                         *
                         * Attributo della classe di tipo int ;serve per memorizzare il numero di
POI.
                         */
                        int numeroRighe = 0;
                /**
                 *
                 * Attrituto costante della servlet; sta ad indicare il numero di colonne delle matrici
                 * POIcoordinateRadius e POIcoordinate.
                 */
                final int numeroColonne = 5;
                /**
                 *
                 * Attributo della classe di tipo int;permette di settare l'intervallo di tempo in
secondi tra due aggionamenti della pagina html effettuati dalla servlet
                 * in maniera automatica;è il cosidetto "Server Push".
                 */
                final int serverPushIntervallo = 10;
                /**
                 *
                 * Attributo costante della servlet; rappresenta l'url della servlet stessa ed è
utilizzata per effettuare il "Server Push".
                 */
                final String urlServlet =
"http://127.0.0.1:8080/entedbguida/servlet/Servlet/ServletFree";//rappresenta url di questa
servlet,serve per il refresh del server push

                /**
                 * Attributo della classe di tipo String;rappresenta url del database
                 */
                static String url ="jdbc:mysql://localhost/entedbguida";
                /**
                 * Attributo della classe di tipo String;rappresenta il nome del driver del database
                 */
                static String driver = "com.mysql.jdbc.Driver";
  /**
   * Il metodo init viene eseguito una sola volta per tutto il ciclo di vita della servlet;
   * serve per effetuare il caricamento dei POIs; ovvero per recuperare tutte le informazioni
   * sui POIs dal Database.
   * Infatti ci si connette al database ,si effetttua la query SELECT
latitudine,longitudine,raggioInteresse,denominazione,url FROM POI WHERE stato = 'ON'
   * ottenendo tutti i POIS attivi e per ogni POI la sua latitudine,longitudine,raggio di
interesse,denominazione,url.
   *
   *
   * @see javax.servlet.GenericServlet#init()
   */
public void init(){

                Connection con = null ;
                Statement stmt = null ;
                ResultSet rs;
                //si ci connette al database
                //si fa una select latitudine,longitudine,raggioInteresse,descrizione from POI where
stato = attivo
                //il ridultato della select si memorizza in POI[n][4],dove n è il numero di records o
di POI disponibili

                try {

                Class.forName(driver);
                con = DriverManager.getConnection(url,"client","client");
                stmt = con.createStatement();
                rs = stmt.executeQuery("SELECT
latitudine,longitudine,raggioInteresse,denominazione,url FROM POI WHERE stato = 'ON'");

                rs.last();
                numeroRighe = rs.getRow();
                POIcoordinate = new double[numeroRighe][2];
                POIcoordinateRadius=new double[numeroRighe][2];
                POIraggio = new double[numeroRighe];
                denominazione=new String[numeroRighe];
                urlPOI = new String[numeroRighe];
                rs.first();
                int i = 0;
```

```
	do{//	for(int i = 0 ; i < numeroRighe ; i++){
				POIcoordinate[i][0]= rs.getDouble(1);
				POIcoordinate[i][1]= rs.getDouble(2);
				POIraggio[i]= (rs.getInt(3));
				denominazione[i]=rs.getString(4);
				urlPOI[i]=rs.getString(5);
				//rs.next();
				i++;
				}
	while(rs.next());
		//		si rilasciano le risorse
		rs.close();
		stmt.close();
		con.close();
				}
		catch (Exception e){
				System.err.println("Errore");
			}

		try {

				stmt.close();
		   con.close();
		   System.out.println("Chiusura connessione");

		} catch (SQLException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
		}


		//facciamo anche una volta sola la conversione delle coordinate dei poi in radianti
		for(int i=0;i<numeroRighe;i++){ //porta le coordinate dei POIs in radianti
					POIcoordinateRadius[i][0]=(POIcoordinate[i]
[0]*(Math.PI/180));
					POIcoordinateRadius[i][1]=(POIcoordinate[i]
[1]*(Math.PI/180));
		}
   }
   /**
    * La servlet una volta istanziata è in attesa della chiamata di un client.
    * Questo metodo è utilizzato dalla Servlet per gestire la chiamata da parte del client di tipo
GET HTTP.
    *
    * @see javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse)
    */
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

				String nextStr; //Coordinate geografiche del client rispetto al sistema di
riferimento cartografico
				double latitudineClient = 0 ;//Latitudine espressa in gradi
				double longitudineClient = 0 ;//Longitudine espressa in gradi
				double latitudineClientRadius = 0; //Latitudine espressa in radianti
				double longitudineClientRadius = 0;//Longitudine espressa in radianti
				Connection con = null ;
				Statement stmt = null ;
				ResultSet rs ;
				String IPAddr = request.getRemoteAddr(); //Indirizzo IP del client che
effettua la richiesta get

				int count=0;

					/*
					* Adesso si portano le coordinate dei POIs in radianti
					*/

					double distanze[]=new double[numeroRighe];//contiene la
distanza in metri del client dal ogni POI
					double minimo;//Minimo delle distanze


//a questo punto si fa una query al database per ottenere la posizione del client dove l'ora è max
//utilizzando l'ip del cliente come parametro e l'ora max siamo sicuri di ottenere una
//sola posizione
```

```
                                        try {

                                                java.sql.Timestamp oraRilevazione = new
Timestamp((new java.util.Date()).getTime());
                                        String ora = oraRilevazione.toString();
                                    Class.forName(driver);
                                        con = DriverManager.getConnection(url,"client","client");
                                        stmt = con.createStatement();
                                        rs = stmt.executeQuery("SELECT latitudine,longitudine
FROM posizione WHERE IPturista = '"+IPAddr+"'AND ora <= '"+ora+"'");

                                        rs.last();//ci permette di recuperare l'ultima posizione del
client ovvero la più recente
                                        latitudineClient = rs.getDouble(1);
                                        longitudineClient = rs.getDouble(2);
                                        //String data = rs.getString(3);
                                    //      si rilasciano le risorse
                                        rs.close();
                                        stmt.close();
                                        con.close();
                                                }
                                        catch (Exception e){
                                                System.err.println("Errore");
                                            }

                                        try {

                                                stmt.close();
                                           con.close();
                                           System.out.println("Chiusura connessione");

                                        } catch (SQLException e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }


                                        /*
                                        * Adesso è possibile calcolare il punto di interesse più vicino
                                        */

                                        latitudineClientRadius=(latitudineClient*Math.PI)/180;
                longitudineClientRadius=(longitudineClient*Math.PI)/180;

                //calcolo le distanze da tutti i POIs
                                        for(int i=0;i<numeroRighe;i++){


//si utilizza d = acos [ sin(lat1)*sin(lat2)+cos(lat1)*cos(lat2)*cos(lon1-lon2) ]*raggioTerra
                                                distanze[i] = Math.acos
( Math.sin(latitudineClientRadius) * Math.sin(POIcoordinateRadius[i][0]) +
Math.cos(latitudineClientRadius) * Math.cos(POIcoordinateRadius[i][0]) *
Math.cos(longitudineClientRadius-POIcoordinateRadius[i][1]) ) * 6371;
                                                //

distanze[i]=Math.acos((Math.cos(latitudineClientRadius)*Math.cos(longitudineClientRadius)*
Math.cos(POIcoordinateRadius[i][0])*Math.cos(POIcoordinateRadius[i][1])) +
(Math.cos(latitudineClientRadius)*Math.sin(longitudineClientRadius)
*Math.cos(POIcoordinateRadius[i][0])*Math.sin(POIcoordinateRadius[i][1]))+
(Math.sin(latitudineClientRadius)*Math.sin(POIcoordinateRadius[i][0])))*EARTH_RADIUS;

                                        }

                                        minimo=distanze[0];
                                        for(int i=1;i<numeroRighe;i++){

                                        /*
                                        * Determina la minima distanza
                                        */
                                        minimo=Math.min(distanze[i],minimo);
                                }

                                        /*
                                        * Determina l'indice del minimo delle n distanze
                                        */
```

```
int finder=0;
for(int i=0;i<numeroRighe;i++)
	if(distanze[i]==minimo) finder=i; //finder
identifica il POI a distanza minore

int indiciPOI[] = new int[numeroRighe];//contiene l'indice di
riga per identificare i POIs
int numeroPOI = 0 ;//indica quanti POI interressano la
posizione del client

double raggioKm[]= null;
raggioKm = new double[numeroRighe];
//portiamo i raggi in Km
for(int i=0;i<numeroRighe;i++){
	raggioKm[i] = POIraggio[i]/1000;

}

//controlliamo se siamo all'interno del raggio di interesse di
qualche POI
	for(int i=0;i<numeroRighe;i++){
		if( raggioKm[i]>= distanze[i]){
			indiciPOI[numeroPOI] = i;
			numeroPOI++;
		}

	}

ServletOutputStream out;
out = response.getOutputStream();
response.setContentType("text/html");

DecimalFormat twoDigits=new DecimalFormat("0.00");


	//primo caso: non ci sono poi che interessano la posizione,ovvero numeroPOI = 0 ;
	//lo si deve pensare meglio
		if( numeroPOI == 0 ){
			//avvisare il client di cio e dire il punto
di interesse più
			//vicino e a quale distanza utilizzando
finder e minimo
			//ricordarsi che gl'indici vanno da 0 a n
			//allora si controlla il più vicino di tipo
itinerario

			out.println("<html>");
			out.println("<head>");
			out.println("<meta http-
equiv=\"Content-Language\" content=\"it\">");
			out.println("<meta http-
equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\">");
//			serve per fare il server push
CONTENT=\"serverPushIntervallo; indica che l'aggionamento avviene ogni
'serverPushIntervallo' secondi
			out.println ("<meta http-
equiv=\"Refresh\" CONTENT=\""+serverPushIntervallo+";URL="+urlServlet+"\"> ");

			out.println("<title>Pocket
Visit</title>");
			out.println("</head>");
			out.println("");
			out.println("<body>");
			out.println("");
			out.println("<p><font size=\"4\">
Attorno a te non c'è nessun POI. Il POI più vicino a te è "+denominazione[finder] +" ad una
distanza di "+twoDigits.format((distanze[finder]*1000)) +" metri.");
			out.println("</font></p>");
			out.println("<p><font size=\"4\">La
tua posizione attuale è Latitudine:"+latitudineClient+"° N, Longitudine:"+longitudineClient+"°
E</font></p>");
			out.println("");
			//qui va inserito il ling alla servlet in
maniera che aggiorna tutto
			//out.println("<p><a
href=\"http://127.0.0.1/Segesta/SegestaServlet\" target=\"_top\">OK</a></p>");
			out.println("</body>");
```

```
                                                out.println("");
                                                out.println("</html>");
                                                out.flush();
                                        out.close();
                                        try { Thread.sleep(5000); } catch
(InterruptedException e) { }

                                }

                                //secondo caso: c'è un solo POI che interessa la
posizione del client

                                else if(numeroPOI == 1){
                                        //avvisare il client di cio e dire che si
trova in prossimita del
                                        //denominazione[indiciPOI[0]] alla
distanza distanze[indicePOI[0]]e l'url è url[indicePOI[0]]
                                        //inoltre si deve implemetare il
controllo sul fatto che se non è
                                        //il POI più vicino lo si deve dire



                                        out.println("<html>");
                                        out.println("<head>");
                                        out.println("<meta http-
equiv=\"Content-Language\" content=\"it\">");
                                        out.println("<meta http-
equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\">");
// serve per fare il server push CONTENT=\"6; indica che l'aggionamento avviene ogni 6
secondi
                                        //quando si è nell'area di interesse di
almeno un poi l'intervallo viene raddoppiato perpermettere
                                        //di potere accedere alla pagina
                                        out.println ("<meta http-
equiv=\"Refresh\" CONTENT=\""+(2*serverPushIntervallo)+";URL="+urlServlet+"\"> ");

                                        out.println("<title>Pocket
Visit</title>");
                                        out.println("</head>");
                                        out.println("");
                                        out.println("<body>");
                                        out.println("");
                                        out.println("<p><font size=\"4\">
Attorno a te c'è il POI: "+denominazione[indiciPOI[0]]+". Il POI più vicino a te è
"+denominazione[indiciPOI[0]] +" ad una distanza di
"+twoDigits.format((distanze[indiciPOI[0]]*1000)) +" metri.");
                                        out.println("</font></p>");
                                        out.println("<p><font size=\"4\">La
tua posizione attuale è Latitudine:"+latitudineClient+"° N, Longitudine:"+longitudineClient+"°
E</font></p>");
                                        out.println("");
                                        //qui va inserito il ling alla servlet in
maniera che aggiorna tutto
                                        out.println("<p><FONT
face=\"Palatino Linotype\" size=4>     <B>Per avere
informazioni sul POI clicca <a href="+urlPOI[indiciPOI[0]]+">qua</a>.</B></FONT></p>");
                                        out.println("</body>");
                                        out.println("");
//                                      out.println("<P><FONT
face=\"Palatino Linotype\" size=4>     <B>Clicca qui per
continuare.<A href=\""+urlServlet+"\" target=_top>OK</A></B></FONT></P>");
                                        out.println("</html>");
                                        out.flush();
                                out.close();



                                try { Thread.sleep(5000); } catch
(InterruptedException e) { }



                                }

                                //terzo caso:ci sono 2 o più POIS che interessano il
client
```

```
                                        else if(numeroPOI > 1 ){
                                                //avvisare di ciò il client e mostrare una
pagina in cui vi è la denominazione
                                                //di tutti i poi le relative url e le relative
distanze
                                                //tutto ciò si fa utilizzando i vettori con
indice quello contenuto
                                                //in indiciPOI
                                                String POIs = null;

                                                out.println("<html>");
                                                out.println("<head>");

                                                out.println("<meta http-
equiv=\"Content-Language\" content=\"it\">");
                                                out.println("<meta http-
equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\">");
                                                //serve per fare il server push
CONTENT=\"6; indica che l'aggionamento avviene ogni 6 secondi
                                                out.println ("<meta http-
equiv=\"Refresh\" CONTENT=\""+(2*serverPushIntervallo)+";URL="+urlServlet+"\"> ");
                                                out.println("<title>Pocket
Visit</title>");
                                                out.println("</head>");
                                                out.println("");
                                                out.println("<body>");
                                                out.println("");
                                                out.println("<p><font size=\"4\">
Attorno a te ci sono : "+numeroPOI+" POIs.Per avere informazioni basta cliccare sulla
denominazione.");
                                                out.println("</font></p>");
                                                for(int i = 0 ; i < numeroPOI ; i++){

                                                out.println("<p><font size=\"4\"> : <a
href="+urlPOI[indiciPOI[i]]+">"+denominazione[indiciPOI[i]]+" </a>ad una distanza di
"+twoDigits.format((distanze[indiciPOI[i]]*1000)) +" metri.");
                                                out.println("</font></p>");

                                                }
                                                out.println("<p><font size=\"4\">La
tua posizione attuale è Latitudine:"+latitudineClient+"° N, Longitudine:"+longitudineClient+"°
E</font></p>");
                                                out.println("");
                                                out.println("</body>");
                                                out.println("");
                                                //out.println("<P><FONT
face=\"Palatino Linotype\" size=4>     <B>Clicca qui per
continuare.<A href=\""+urlServlet+"\" target=_top>OK</A></B></FONT></P>");
                                                out.println("</html>");
                                                out.flush();
                                        out.close();


                                        try { Thread.sleep(5000); } catch
(InterruptedException e) { }

                                }

    }
}
```

A.5.3 ServletGuide.java

```
package Servlet;

import javax.servlet.*;
```

```
import javax.servlet.http.*;
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.text.DecimalFormat;

/**
 * Questa servlet serve per gestire le richieste del client.
 * Ad ogni richiesta la servlet risponde inviando una pagina HTML
 * che contiene le informazioni circa la posizione del client.
 *
 * @author Pietro Amato
 *
 *
 */

public class ServletGuida extends HttpServlet{

                        /**
                         *
                         * Attributo della servlet di tipo double[][];serve per memorizzare le
                         * coordinate dei POIs in radianti;ogi riga rappresenta un POI mentre
nelle due colonne sono memorizzati
                         * la latitudine e la longitudine.
                         * E' dichiarato static poichè una volta istanziato non varia più;questo
                         * vuol dire che se si modificano i POIs,affinchè queste modifiche siano
                         * effettuate bisogna riavviare il serverweb Tomcat
                         */
                        static double POIcoordinateRadius[][]= null;
                        /**
                         *
                         * Attributo della classe costante,indica il raggio della terra in Km;
                         * ricordiamo che il raggio della terra non è costante e ha il suo massimo
pari a 6378.14 KM all'equatore
                         * e il suo minimo pari a 6356.75 Km in corrispondenza dei poli .
                         *
                         */
                        final double EARTH_RADIUS = 6371.0;
                        /**
                         *
                         * Attributo della servlet di tipo double[][];serve per memorizzare le
coordinate dei POIs
                         * in gradi,minuti,secondi ovvero nel formato in cui si ricevono le
coordinate dal GPS.
                         * E' dichiarato static poichè una volta istanziato non varia più;questo
                         * vuol dire che se si modificano i POIs,affinchè queste modifiche siano
                         * effettuate bisogna riavviare il serverweb Tomcat
                         */
                        static double POIcoordinate[][]= null;//Contiene le coordinate dei POIs
                        /**
                         *
                         * Attributo della servlet di tipo double[];serve per memorizzare il raggio
di interesse di ogni
                         * POIs.
                         * POIraggio[n] ci restituisce il raggio d'inderesse dell'ennesimo POI.
                         */
                        static double POIraggio[]= null;
                        /**
                         *
                         * Attributo della servlet di tipo String[];serve per memorizzare la
denominazione dei POIS.
                         * denominazione[n] ci restituisce la denominazione dell'ennesimo POI.
                         */
                        static String denominazione[]= null;
                        /**
                         *
                         * Attributo della servlet di tipo String[];serve per memorizzare della
tipologia dei POIS.
                         * tipologia[n] ci restituisce la denominazione dell'ennesimo POI.
                         */
                        static String tipologia[]= null;
                        /**
                         *
```

```
                         * Attributo della servlet di tipo String[];serve per memorizzare l' id  dei
POIS.
                         * IDpoi[n] ci restituisce l'id  dell'ennesimo POI.
                         */
                        static int IDpoi[]= null;
                        /**
                         *
                         * Attributo della servlet di tipo String[];serve per memorizzare le URL
dei POIs.
                         * urlPOI[n] ci restituisce l'URL dell'ennesimo POI.
                         */
                        static String urlPOI[]= null;
                        /**
                         *
                         * Attributo della servlet di tipo String[];serve per memorizzare l' id  dei
POIS.
                         * IDpoi[n] ci restituisce l'id  dell'ennesimo POI.
                         */
                        static int IDpercorso[]= null;
                        /**
                         *
                         * Attributo della servlet di tipo String[];serve per memorizzare le URL
dei POIs.
                         * urlPOI[n] ci restituisce l'URL dell'ennesimo POI.
                         */
                        static String nomePercorso[]= null;
                        /**
                         *
                         * Attributo della classe di tipo int ;serve per memorizzare il numero di
POIs.
                         */
                        int numeroPercorsi = 0;

                        /**
                         *
                         * Attributo della classe di tipo int ;serve per memorizzare il numero di
POIs.
                         */
                        int numeroRighe = 0;
                /**
                 *
                 * Attrituto costante della servlet; sta ad indicare il numero di colonne delle matrici
                 * POIcoordinateRadius e POIcoordinate.
                 */
                final int numeroColonne = 5;
                /**
                 *
                 * Attributo della classe di tipo int;permette di settare l'intervallo di tempo in
secondi tra due aggionamenti della pagina html effettuati dalla servlet
                 * in maniera automatica;è il cosidetto "Server Push".
                 */
                final int serverPushIntervallo = 10;
                /**
                 *
                 * Attributo costante della servlet; rappresenta l'url della servlet stessa ed è
utilizzata per effettuare il "Server Push".
                 */
                final String urlServlet =
"http://127.0.0.1:8080/entedbguida/servlet/Servlet/ServletGuida";//rappresenta url di questa
servlet,serve per il refresh del server push
                /**
                 *
                 * Attributo costante della servlet; rappresenta l'url dell'homePage del sito dell'ente
                 */
                final String homePage = "http://127.0.0.1:8080/entedbguida/";
                /**
                 * Attributo di tipo String che rappresenta l'url del database
                 */
                static String url ="jdbc:mysql://localhost/enteDBguida";
                /**
                 * Attributo di tipo String che rappresenta il nome del driver del database
                 */
                static String driver = "com.mysql.jdbc.Driver";

  /**
   * Il metodo init viene eseguito una sola volta per tutto il ciclo di vita della servlet;
   * serve per effetuare il caricamento dei POIs; ovvero per recuperare tutte le informazioni
```

```
    * sui POIs dal Database.
    * Infatti ci si connette al database ,si effetttua la query SELECT
latitudine,longitudine,raggioInteresse,denominazione,url FROM POI WHERE stato = 'ON'
    * ottenendo tutti i POIS attivi e per ogni POI la sua latitudine,longitudine,raggio di
interesse,denominazione,url.
    *
    *
    * @see javax.servlet.GenericServlet#init()
    */
public void init(){

                //modifica da effettuare
                //mostrare una pagina html in cui il turista seleziona il percorso
                //una volta selezionato il percorso bisogna caricare tutti i poi appartenenti a quel
percorso
                //poi si prosegue nel modo precedente ovvero con un solo percorso

                //bisogna creare il file guida nel palmare

                Connection con = null ;
                Statement stmt = null ;
                ResultSet rs;

                //ci si connette al database
                //si fa una select latitudine,longitudine,raggioInteresse,descrizione from POI where
stato = attivo
                //il ridultato della select si memorizza in POI[n][4],dove n è il numero di records o
di POI disponibili
                try {

                Class.forName(driver);
                con = DriverManager.getConnection(url,"client","client");
                stmt = con.createStatement();
                rs = stmt.executeQuery("SELECT
latitudine,longitudine,raggioInteresse,denominazione,url,IDpoi,tipologia FROM POI WHERE
stato = 'ON'");
                rs.last();
                numeroRighe = rs.getRow();
                POIcoordinate = new double[numeroRighe][2];
                POIcoordinateRadius=new double[numeroRighe][2];
                POIraggio = new double[numeroRighe];
                denominazione=new String[numeroRighe];
                tipologia=new String[numeroRighe];
                urlPOI = new String[numeroRighe];
                IDpoi = new int[numeroRighe];
                rs.first();
                int i = 0;

                do{
                        POIcoordinate[i][0]= rs.getDouble(1);
                        POIcoordinate[i][1]= rs.getDouble(2);
                        POIraggio[i]= (rs.getInt(3));
                        denominazione[i]=rs.getString(4);
                        urlPOI[i]=rs.getString(5);
                        IDpoi[i]=rs.getInt(6);
                        tipologia[i]=rs.getString(7);
                        i++;
                }
        while(rs.next());

                //si rilasciano le risorse
        rs.close();
        stmt.close();
        con.close();
                }//fine try

                catch (Exception e){
                        System.err.println("Errore__1");
                }

                try {

                        stmt.close();
                   con.close();
                   System.out.println("Chiusura connessione__1");

                }
```

```
		catch (SQLException e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}
		//facciamo anche una volta sola la conversione delle coordinate dei poi in radianti

		for(int i=0;i<numeroRighe;i++){ //porta le coordinate dei POIs in radianti
			POIcoordinateRadius[i][0]=(POIcoordinate[i][0]*(Math.PI/180));
			POIcoordinateRadius[i][1]=(POIcoordinate[i][1]*(Math.PI/180));
		}//fine for
	}//fine metodo init
	/**
	 * La servlet una volta istanziata è in attesa della chiamata di un client.
	 * Questo metodo è utilizzato dalla Servlet per gestire la chiamata da parte del client di tipo
GET HTTP.
	 *
	 * @see javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse)
	 */
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{

		String nextStr; //Coordinate geografiche del client rispetto al sistema di riferimento
cartografico

		double latitudineClient = 0;//Latitudine espressa in gradi

		double longitudineClient = 0;//Longitudine espressa in gradi

		double latitudineClientRadius = 0; //Latitudine espressa in radianti

		double longitudineClientRadius = 0;//Longitudine espressa in radianti

		String dataRegistrazione = null;

		int IDpoiFrom;

		int IDpoiTo;

		int IDPercorsoScelto = 0;

		String infoGuida;

		String infoGuidaPOI;

		Connection con = null ;

		Statement stmt = null ;

		ResultSet rs ;

		String IPAddr = request.getRemoteAddr(); //Indirizzo IP del client che effettua la
richiesta get

		int count=0;

				/*
				* Adesso si portano le coordinate dei POIs in radianti
				*/

		double distanze[]=new double[numeroRighe];//contiene la distanza in metri del
client dal ogni POI

		double minimo;//Minimo delle distanze

//a questo punto si fa una query al database per ottenere la posizione del client dove l'ora è max
//utilizzando l'ip del cliente come parametro e l'ora max siamo sicuri di ottenere una
//sola posizione

		try {

			java.sql.Timestamp oraRilevazione = new Timestamp((new
java.util.Date()).getTime());

			String ora = oraRilevazione.toString();

			Class.forName(driver);
```

```
                        con = DriverManager.getConnection(url,"client","client");

                        stmt = con.createStatement();

                        rs = stmt.executeQuery("SELECT latitudine,longitudine FROM
posizione WHERE IPturista = '"+IPAddr+"'AND ora <= '"+ora+"'");

                        rs.last();//ci permette di recuperare l'ultima posizione del client ovvero la
più recente

                        latitudineClient = rs.getDouble(1);

                        longitudineClient = rs.getDouble(2);

                }//fine try

                catch (Exception e){

                        System.err.println("Errore__2");

                }//fine catch


                ServletOutputStream out;

                out = response.getOutputStream();

                response.setContentType("text/html");

                DecimalFormat twoDigits=new DecimalFormat("0.00");

                //si effettua una select per ottenere il traguardo corrente del turista

                try {

                        Timestamp dataIngresso = null;

                        java.sql.Timestamp oraRilevazione = new Timestamp((new
java.util.Date()).getTime());

                        String ora = oraRilevazione.toString();

                        Class.forName(driver);

                        con = DriverManager.getConnection(url,"client","client");

                        stmt = con.createStatement();

                        rs = stmt.executeQuery("SELECT oraIngresso FROM turista where IP =
'"+IPAddr+"' AND oraUscita is NULL");

                        rs.next();

                        dataIngresso = rs.getTimestamp(1);

                        rs = stmt.executeQuery("SELECT
IDpoiFrom,IDpoiTo,azione,azionePoi,IDpercorso FROM traguardoCorrente WHERE IPturista =
'"+IPAddr+"'AND dataRegistrazione > '"+dataIngresso+"' AND dataRegistrazione <=
'"+ora+"'");

                        rs.last();//ci poniamo su l'ultima Row,si ha lo stesso risultato con
rs.first(),o con rs.next(),in quanto rs punta ad una sola row

                        if(rs.getRow()==0){//non c'è un traguardo corrente

                                //il turista è appena entrato nel parco o ha appena deciso di
essere guidato
                                //qui abbiamo due casi:
                                //1:se il turista è all'ingresso allora idpoifrom=0
                                //2:se il turista è all'interno del parco ed ha appena deciso di
essere guidato allora
                                //idpoito!=0 e il percorso non può essere deciso ma solo
all'interno di quelli disponibili

                        //calcoliamo il poi più vicino
                                latitudineClientRadius=(latitudineClient*Math.PI)/180;
```

```
longitudineClientRadius=(longitudineClient*Math.PI)/180;

//calcolo le distanze da tutti i POIs

for(int i=0;i<numeroRighe;i++){

	//si utilizza d = acos
[ sin(lat1)*sin(lat2)+cos(lat1)*cos(lat2)*cos(lon1-lon2) ]*raggioTerra

	distanze[i] = Math.acos
( Math.sin(latitudineClientRadius) * Math.sin(POIcoordinateRadius[i][0]) +
Math.cos(latitudineClientRadius) * Math.cos(POIcoordinateRadius[i][0]) *
Math.cos(longitudineClientRadius-POIcoordinateRadius[i][1]) ) * 6371;

	//distanze[i]=Math.acos((Math.cos(latitudineClient
Radius)*Math.cos(longitudineClientRadius)*Math.cos(POIcoordinateRadius[i]
[0])*Math.cos(POIcoordinateRadius[i][1])) +
(Math.cos(latitudineClientRadius)*Math.sin(longitudineClientRadius)
*Math.cos(POIcoordinateRadius[i][0])*Math.sin(POIcoordinateRadius[i][1]))+
(Math.sin(latitudineClientRadius)*Math.sin(POIcoordinateRadius[i][0])))*EARTH_RADIUS;


}//fine for

minimo=distanze[0];

for(int i=1;i<numeroRighe;i++){
		/*
		* Determina la minima
distanza
		*/

	minimo=Math.min(distanze[i],minimo);


}//fine for

	 /*
	  * Determina l'indice del minimo delle n distanze
	  */

int finder=0;

for(int i=0;i<numeroRighe;i++)

	if(distanze[i]==minimo) finder=i; //finder
identifica il POI a distanza minore


int IDpoiPiuVicino = IDpoi[finder];

ResultSet rs1 ;

int poiFrom;

String azione;

String azionePoi;


//selezioniamo il nodo il cui IDpoiTo == IDpoiPiuVicino per
creare il traguardo corrente
//con più percorsi si possono avere più risultati
rs1 = stmt.executeQuery("Select
IDpoiFrom,azione,azionePoi,IDpercorso FROM nodo WHERE IDpoiTo =
'"+IDpoiPiuVicino+"'");
int IDpoiControllo = 0;
while(rs1.next()){
	//controlliamo se siamo all'ingresso del parco e
	//quindi mostrare al turista la pagina di scelta di
tutti percorsi
	if(rs1.getInt(1)==0){
		IDpoiControllo = 1;
		//return;
	}
}
```

```
                                        if((request.getParameter("nomePercorso"))==null){
                                                IDPercorsoScelto = 0;
                                        }
                                        else{
                                                IDPercorsoScelto =
Integer.parseInt(request.getParameter("nomePercorso"));
                                        }

                                        //bisogna controllare IDpoiControllo;se è uguale a 1 il turista
si trova all'ingresso del parco
                                        //altrimenti si trova all'interno del parco

                                        if(IDpoiControllo == 1){
                                                //bisogna gestire il caso in cui il turista è
all'ingresso del parco

                                                try {

                                                Class.forName(driver);
                                                con =
DriverManager.getConnection(url,"client","client");
                                                stmt = con.createStatement();
                                                rs = stmt.executeQuery("SELECT
IDpercorso,nomePercorso FROM percorso");
                                                rs.last();
                                                numeroPercorsi = rs.getRow();
                                                IDpercorso = new int[numeroPercorsi];
                                                nomePercorso=new String[numeroPercorsi];

                                                rs.first();
                                                int i = 0;

                                                do{
                                                        IDpercorso[i]= rs.getInt(1);
                                                        nomePercorso[i]= rs.getString(2);
                                                        i++;
                                                }
                                            while(rs.next());

                                                }//fine try

                                                catch (Exception e){
                                                        System.err.println("Errore__1");
                                                }
                                        /*
                                                try {

                                                        stmt.close();
                                                   con.close();
                                                   System.out.println("Chiusura
connessione__1");

                                                }
                                                catch (SQLException e1) {
                                                        // TODO Auto-generated catch block
                                                        e1.printStackTrace();
                                                }
                                                */
                                                /*
                                                String nomePercorsoScelto =
(String)JOptionPane.showInputDialog(null,"Selezionare il percorso da effettuare:","Selezionare
Percorso",JOptionPane.INFORMATION_MESSAGE,null,nomePercorso,nomePercorso[0]);
                                        for(int j = 0 ; j < numeroPercorsi; j++){
                                                System.out.println("Siamo dentro il for,controllo
se funziona il break");


   if(nomePercorso[j].compareTo(nomePercorsoScelto)==0){
                                                        IDPercorsoScelto=IDpercorso[j];
                                                        System.out.println("Siamo dentro
l'if,controllo se funziona il break");
                                                        break;

                                                }
                                        }

                                                */
```

```
                                        if(IDPercorsoScelto == 0){//viene mostrata questa
pagina prima che l'utente ha scelto il percorso
                                        response.setContentType("text/html");
                                        out.println("<html>");
                                        out.println("<head>");
                                        out.println("<meta http-equiv=\"Content-Type\"
content=\"text/html; charset=windows-1252\">");
                                        out.println("<title>Selezionare il percorso da
effettuare.</title>");
                                        out.println("</head>");
                                        out.println("<body>");
                                        out.println("<p align=\"center\"><font
face=\"Comic Sans MS\" size=\"5\" color=\"#0000FF\"><strong>Quale percorso vuoi
effettuare?</strong></font></p>");
                                        out.println("<form>");//
action=\"ServletGuida\"METHOD=GET>");
                                        out.println("<dl>");
                                        for(int j = 0 ; j < numeroPercorsi; j++){
                                                out.println("<dd><p
align=\"center\">");
                                                out.println("<input type=\"radio\"
name=\"nomePercorso\" value=\""+IDpercorso[j]+"\"><b><i><font
size=\"4\">"+nomePercorso[j]+"</i></b></dd>");
                                                out.println("<dd>");
                                        }
                                        out.println("</dl>");
                                //out.println("<p align=\"center\"><input
type=submit></form>");
                                        out.println("<p align=\"center\"><INPUT
type=submit value=\"Conferma\" style=\"border-style:ridge; border-width:1; padding:0; font-
family: Arial (fantasy); font-weight: bold; \"></FORM>");
                                        out.println("</body>");
                        out.println("</html>");
                        out.flush();
                        out.close();
                                        }
                        IDPercorsoScelto =
Integer.parseInt(request.getParameter("nomePercorso"));


                        //System.out.println("Speriamo che funzioni \n
nomepercorso:"+nomePercorso);
                        //System.out.println("modifica");

                                        //recuperiamo il nodo che appartiene al percorso
scelto
                        rs1 = stmt.executeQuery("Select IDpoiFrom,azione,azionePoi
FROM nodo WHERE IDpoiTo = '"+IDpoiPiuVicino+"' AND IDpercorso =
'"+IDPercorsoScelto+"'");

                        //recuperiamo i valori per la creazione del traguardo corrente
                        rs1.next();

                        poiFrom = rs1.getInt(1);

                                        azione = rs1.getString(2);

                                        azionePoi = rs1.getString(3);


                    }
                    else{//bisogna gestire il caso in cui il turista è all'interno del
parco

                                        rs1.last();//ci poniamo su l'ultima Row
                                        int numeroPercorsiDisponibili = rs1.getRow();

                                        int IDpercorsiDisponibili[] = new
int[numeroPercorsiDisponibili];


                                        rs1.first();
                                        int j = 0;

                                        do{
                                                IDpercorsiDisponibili[j]=rs1.getInt(4);
```

```
                                                j++;
                                        }
                                        while(rs1.next());

                                        String query = "'"+IDpercorsiDisponibili[0]+"'";
                                        for(int i = 1 ; i < numeroPercorsiDisponibili;i++){
                                                query = query.concat(" OR IDpercorso
= '"+IDpercorsiDisponibili[i]+"'");
                                                }


                                        try {

                                        Class.forName(driver);
                                        con =
DriverManager.getConnection(url,"client","client");
                                        stmt = con.createStatement();
                                        rs = stmt.executeQuery("SELECT
IDpercorso,nomePercorso FROM percorso WHERE IDpercorso = "+query+"");

                                        rs.last();
                                        numeroPercorsi = rs.getRow();
                                        IDpercorso = new int[numeroPercorsi];
                                        nomePercorso=new String[numeroPercorsi];
                                 rs.first();
                                        int i = 0;

                                        do{
                                                IDpercorso[i]= rs.getInt(1);
                                                nomePercorso[i]= rs.getString(2);
                                                i++;
                                        }
                                    while(rs.next());

                                        }//fine try

                                        catch (Exception e){
                                                System.err.println("Errore__1");
                                        }
                                /*
                                        try {

                                                stmt.close();
                                           con.close();
                                           System.out.println("Chiusura
connessione__1");

                                        }
                                        catch (SQLException e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }
                                        */

                                        /*
                                        String nomePercorsoScelto =
(String)JOptionPane.showInputDialog(null,"Selezionare il percorso da effettuare:","Selezionare
Percorso",JOptionPane.INFORMATION_MESSAGE,null,nomePercorso,nomePercorso[0]);
                                    for(int jj = 0 ; jj < numeroPercorsi; jj++){
                                                System.out.println("Siamo dentro il for,controllo
se funziona il break");


   if(nomePercorso[jj].compareTo(nomePercorsoScelto)==0){
                                                IDPercorsoScelto=IDpercorso[jj];
                                                System.out.println("Siamo dentro
l'if,controllo se funziona il break");
                                                break;

                                        }
                                }

                                         *
                                *
                                */
                                        if(IDPercorsoScelto == 0){//viene mostrata questa
pagina prima che l'utente ha scelto il percorso
```

```
                                        response.setContentType("text/html");
                                        out.println("<html>");
                                        out.println("<head>");
                                        out.println("<meta http-equiv=\"Content-Type\"
content=\"text/html; charset=windows-1252\">");
                                        out.println("<title>Selezionare il percorso da
effettuare.</title>");
                                        out.println("</head>");
                                        out.println("<body>");
                                        out.println("<p align=\"center\"><font
face=\"Comic Sans MS\" size=\"5\" color=\"#0000FF\"><strong>Quale percorso vuoi
effettuare?</strong></font></p>");
                                        out.println("<form>");
                                        out.println("<dl>");
                                        for(int jj = 0 ; jj < numeroPercorsi; jj++){
                                                out.println("<dd><p
align=\"center\">");
                                                out.println("<input type=\"radio\"
name=\"nomePercorso\" value=\""+IDpercorso[jj]+"\"><b><i><font
size=\"4\">"+nomePercorso[jj]+"</i></b></dd>");
                                                out.println("<dd>");
                                        }
                                        out.println(" </dl>");
                                        out.println("<p align=\"center\"><INPUT
type=submit value=\"Conferma\" style=\"border-style:ridge; border-width:1; padding:0; font-
family: Arial (fantasy); font-weight: bold; \"></FORM>");
                                        out.println("</body>");
                                out.println("</html>");
                                out.flush();
                                out.close();
                                        }
                                IDPercorsoScelto =
Integer.parseInt(request.getParameter("nomePercorso"));
                                        //System.out.println("Speriamo che funzioni \n
nomepercorso:"+nomePercorso);
                                //recuperiamo il nodo che appartiene al percorso scelto

                                        rs1 = stmt.executeQuery("Select
IDpoiFrom,azione,azionePoi FROM nodo WHERE IDpoiTo = '"+IDpoiPiuVicino+"' AND
IDpercorso = '"+IDPercorsoScelto+"'");
                                //recuperiamo i valori per la creazione del traguardo corrente
                                rs1.next();
                                poiFrom = rs1.getInt(1);
                                        azione = rs1.getString(2);
                                        azionePoi = rs1.getString(3);
                                }



                                        if(IDPercorsoScelto!=0){//viene mostrata questa pagina solo
dopo che l'utente ha scelto il percorso
                                        //creazione del traguardo corrente

                                        java.sql.Timestamp dataRilevazione = new Timestamp((new
java.util.Date()).getTime());

                                        String data = dataRilevazione.toString();

                                        int n =stmt.executeUpdate("Insert into
TraguardoCorrente(IPturista,dataRegistrazione,IDpoiFrom,IDpoiTo,azione,azionePoi,IDpercorso
)values('"+IPAddr+"','"+data+"','"+poiFrom+"','"+IDpoiPiuVicino+"','"+azione+"','"+azionePoi+"
','"+IDPercorsoScelto+"')");
                                        System.out.println("Insert into
TraguardoCorrente(IPturista,dataRegistrazione,IDpoiFrom,IDpoiTo,azione,azionePoi,IDpercorso
)values('"+IPAddr+"','"+data+"','"+poiFrom+"','"+IDpoiPiuVicino+"','"+azione+"','"+azionePoi+"
','"+IDPercorsoScelto+"'");
                                        System.out.println("NUMERO
INSERIMENTI:"+n+"indirizzo ip :"+IPAddr);
                                        //si crea una pagina e la si invia al client dove si visualizza la
distanza dal poi piu vicino e l'info guida ovvero l'azione
                                        /*<body>

                                        out.println("<div align="center">");
                                        out.println("<center>");
                                        out.println("<table border="0" width="80%">");
                                        out.println("<tr>");
```

```
                                        out.println("<td align="center" colspan="2"><b><font
size="6">Servizio Guida Turista</font></b></td>");
                                        out.println(" </tr>");
                                        out.println("<tr>");
                                        out.println("<td colspan="2"></td>");
                                        out.println("</tr>");
                                        out.println("<tr>");
                                        out.println("<td colspan="2"><b><i><font
size="5">Informazioni utili:</font></i></b></td>");
                                        out.println("</tr>");
                                        out.println("<tr>");
                                        out.println("<td width="5%"></td>");
                                        out.println("<td width="60%"><b>Grazie a questo servizio
verrai guidato attraverso informazioni visualizzate dal tuo Pocket Explorer. </b>");
                                        out.println("<P><b>Basta seguire le semplici istruzioni per
potere visitare il parco. </b></P>
                                        out.println("<P><b>Ogni volta che si raggiunge un POI
d'interesse verrai avvisato ed un link ti porterà ad una pagina info su tale POI. </b></P>
                                        out.println("<P><b>Quando invece non ti troverai nel raggio
d'interesse di nessun POI verrà mostrata una pagina con le informazioni guida che si auto-
aggiornano .</b></P>
                                        <P><b>La tua posizione attuale è Latitudine:38.10456° N,
Longitudine:13.34629° E</b></P>
                                        out.println("<P><FONT face="Palatino Linotype"
size=4>     <B>Clicca qui per iniziare il percorso.<A
href="http://127.0.0.1:8080/entedbguida/servlet/Servlet/ServletGuida"
target=_top>OK</A></B></FONT></P>");
                                            out.println("<p> </td>");
                                          out.println("</tr>");
                                         out.println("</table>");
                                        out.println("</center>");
                                        out.println("</div>");

                                        </body>*/


                                        out.println("<html>");
                                        out.println("<head>");
                                        out.println("<meta http-equiv=\"Content-Language\"
content=\"it\">");
                                        out.println("<meta http-equiv=\"Content-Type\"
content=\"text/html; charset=windows-1252\">");
                                        out.println("<title>Pocket Visit</title>");
                                        out.println("</head>");
                                        out.println("");
                                        out.println("<body>");
                                        out.println("");
                                        out.println("<div align=\"center\">");
                                        out.println("<center>");
                                        out.println("<table border=\"0\" width=\"80%\">");
                                        out.println("<tr>");
                                        out.println("<td align=\"center\" colspan=\"2\"><b><font
size=\"6\">Servizio Guida Turista</font></b></td>");
                                        out.println(" </tr>");
                                        out.println("<tr>");
                                        out.println("<td colspan=\"2\"></td>");
                                        out.println("</tr>");
                                        out.println("<tr>");
                                        out.println("<td colspan=\"2\"><b><i><font
size=\"5\">Informazioni utili:</font></i></b></td>");
                                        out.println("</tr>");
                                        out.println("<tr>");
                                        out.println("<td width=\"5%\"></td>");
                                        out.println("<td width=\"60%\"><b>Grazie a questo servizio
verrai guidato attraverso informazioni visualizzate dal tuo Pocket Explorer. </b>");
                                        out.println("<P><b>Basta seguire le semplici istruzioni per
potere visitare il parco. </b></P>");
                                        out.println("<P><b>Ogni volta che si raggiunge un POI
d'interesse verrai avvisato ed un link ti porterà ad una pagina info su tale POI. </b></P>");
                                        out.println("<P><b>Quando invece non ti troverai nel raggio
d'interesse di nessun POI verrà mostrata una pagina con le informazioni guida che si auto-
aggiornano .</b></P>");
                                        out.println("<P><b>La tua posizione attuale è
Latitudine:"+latitudineClient+"° N, Longitudine:"+longitudineClient+"° E</b></P>");
                                        out.println("<P><FONT face=\"Palatino Linotype\"
size=4>     <B>Clicca qui per iniziare il percorso.<A
href=\""+urlServlet+"\" target=_top>OK</A></B></FONT></P>");
```

```
                                out.println("<p> </td>");
                                out.println("</tr>");
                                out.println("</table>");
                                out.println("</center>");
                                out.println("</div>");
                                out.println("</html>");

                                out.flush();

                                out.close();

                                try { Thread.sleep(5000); } catch (InterruptedException e) { }

                        }//fine if creazione traguardo corrente

                        }//fine if


                        else{//se c'è un traguardo corrente


                                IDpoiFrom = rs.getInt(1);

                                IDpoiTo = rs.getInt(2);

                                infoGuida = rs.getString(3);

                                infoGuidaPOI = rs.getString(4);

                                IDPercorsoScelto = rs.getInt(5);

                                //si effettua il controllo se si è terminato il percorso

                                if(IDpoiTo == 0 ){


        //si visualizza pagina che dice che il percorso è finito

                                                out.println("<html>");
                                                out.println("<head>");
                                                out.println("<meta http-equiv=\"Content-
Language\" content=\"it\">");
                                                out.println("<meta http-equiv=\"Content-Type\"
content=\"text/html; charset=windows-1252\">");
                                                out.println("<title>Pocket Visit</title>");
                                                out.println("</head>");
                                                out.println("");
                                                out.println("<body>");
                                                out.println("");
                                                out.println("<p><font size=\"4\"> Il percorso è
stato completato;ti trovi all'uscita del parco.");
                                                out.println("</font></p>");
                                                out.println("<p><font size=\"4\"> ");
                                                out.println("</font></p>");
                                                out.println("<p><font size=\"4\">La tua posizione
attuale è Latitudine:"+latitudineClient+"° N, Longitudine:"+longitudineClient+"°
E</font></p>");
                                                out.println("");
                                                out.println("</body>");
                                                out.println("");
                                                out.println("<p><font face=\"Palatino Linotype\"
size=\"4\">      Clicca qui per ritornare alla HomePage.<a
href=\""+homePage+"\" target=\"_top\">OK</a></p>");
                                                out.println("");
                                                out.println("</html>");
                                                out.flush();
                                                out.close();

                                                try { Thread.sleep(5000); } catch
(InterruptedException e) { }


                                }//fine if


                                else{
```

```
                                        //calcoliamo se il turista si trova dentro il raggio
d'interesse del POI con id = IDpoiTo;

                                        //recuperiamo la latitudine e longitudine del
suddetto poi

                                        int j = 0;

                                        while(IDpoiTo != IDpoi[j]){

                                                j++;

                                        }


   latitudineClientRadius=(latitudineClient*Math.PI)/180;


   longitudineClientRadius=(longitudineClient*Math.PI)/180;


                                        //calcolo la distanza dal poi

                                        double distanza = Math.acos
( Math.sin(latitudineClientRadius) * Math.sin(POIcoordinateRadius[j][0]) +
Math.cos(latitudineClientRadius) * Math.cos(POIcoordinateRadius[j][0]) *
Math.cos(longitudineClientRadius-POIcoordinateRadius[j][1]) ) * 6371;


                                        //controlliamo se siamo all'interno del raggio di
interesse di qualche POI

                                        if( (POIraggio[j]/1000) >= distanza){//siamo
all'interno del poi,si divide per mille per portare il raggio in Km

                                                //si aggiorna il traguardo corrente con il
nodo che ha come IDpoiFrom = IDpoiTo

                                                //si recupera il nodo

                                                ResultSet rs2 ;

                                                int poiTo;

                                                String azione;

                                                String azionePoi;

                                                rs2 = stmt.executeQuery("Select
IDpoiTo,azione,azionePoi FROM nodo WHERE IDpoiFrom = '"+IDpoiTo+"' AND IDpercorso =
'"+IDPercorsoScelto+"'");

                                                System.out.println("ciao
IDpoiFrom"+IDpoiFrom+POIraggio[j]);

                                                rs2.next();

                                                if(rs2.getInt(1)== 0){

                                                        poiTo = 0;

                                                }

                                                else{

                                                        poiTo = rs2.getInt(1);

                                                }//fine else


                                                azione = rs2.getString(2);

                                                azionePoi = rs2.getString(3);


                                                //aggiornamento del traguardo corrente
```

```
                                                    java.sql.Timestamp dataRilevazione =
new Timestamp((new java.util.Date()).getTime());

                                                    String data =
dataRilevazione.toString();

                                                    stmt.executeUpdate("Update
TraguardoCorrente set dataRegistrazione= '"+dataRilevazione+"',IDpoiFrom
='"+IDpoiTo+"',IDpoiTo = '"+poiTo+"',azione = '"+azione+"',azionePoi = '"+azionePoi+"'
WHERE IPturista = '"+IPAddr+"' AND IDpoiTo = '"+IDpoiTo+"' AND dataRegistrazione >
'"+dataIngresso+"' AND dataRegistrazione <= '"+dataRilevazione+"' AND IDpercorso =
'"+IDPercorsoScelto+"'");

                                                    //controllare il tipo del poi in cui si
trova il turista per decidere di che tipo

                                                    //(info o guida)deve essere la pagina da
inviare al turista

                                                    String tipo = tipologia[j];


   System.out.println("tipologia"+tipologia[j]);

                                                    System.out.println("tipo:"+tipo);



   if(tipo.compareToIgnoreCase("interesse")== 0 ){

                                                            //si visualizza una pagina in
cui si avvisa il turista che si trova nei pressi del poi

                                                            //con denominazione[j],e un
link che porta a info

                                                            out.println("<html>");
                                                            out.println("<head>");
                                                            out.println("<meta http-
equiv=\"Content-Language\" content=\"it\">");
                                                            out.println("<meta http-
equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\">");
                                                            out.println("<title>Pocket
Visit</title>");
                                                            out.println("</head>");
                                                            out.println("");
                                                            out.println("<body>");
                                                            out.println("");
                                                            out.println("<p><font
size=\"4\"> Attorno a te c'è il POI "+denominazione[j] +" ad una distanza di
"+twoDigits.format((distanze[j]*1000)) +" metri.");
                                                            out.println("</font></p>");
                                                            out.println("<p><font
size=\"4\">La tua posizione attuale è Latitudine:"+latitudineClient+"° N,
Longitudine:"+longitudineClient+"° E</font></p>");
                                                            out.println("");

                                                            //qui va inserito il link al
pagina info del poi

                                                            out.println("<p><font
face=\"Palatino Linotype\" size=\"4\">&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; Clicca qui per
visualizzare le informazioni sul poi "+denominazione[j]+"<a href=\""+urlPOI[j]+"\"
target=\"_top\">OK</a></font></p>");
                                                            out.println("</body>");
                                                            out.println("");


   if( infoGuidaPOI.compareToIgnoreCase("nessuna")!= 0 ){


   out.println("<p><font size=\"4\">Devi fare la seguente azione :<font
color=\"#FF0000\">"+infoGuidaPOI+"</font> prima di continuare il percorso</font></p>");
                                                                    out.println("");
```

```
                                                        }//fine if
                                                        out.println("<P><FONT
face=\"Palatino Linotype\" size=4>     <B>Clicca qui per
continuare il percorso.<A href=\""+urlServlet+"\" target=_top>OK</A></B></FONT></P>");

                                                        //out.println("<p><font
face=\"Palatino Linotype\" size=\"4\">      Clicca qui per
continuare il percorso.<a href=\""+urlServlet+"\" target=\"_top\">OK</a></p>");

                                                        out.println("");
                                                        out.println("</html>");
                                                        out.flush();
                                                        out.close();

                                                        try { Thread.sleep(5000); }
catch (InterruptedException e) { }


                                                }//fine if


                                                else{//allora è di tipo itinerario

                                                        //si visualizza la pagina
guida la turista dicendo di fare questa azione infoGuidaPOI(se c'è ) ,che si trova nel poi itinerio
denominazione[j]

                                                        out.println("<html>");
                                                        out.println("<head>");
                                                        out.println("<meta http-
equiv=\"Content-Language\" content=\"it\">");
                                                        out.println("<meta http-
equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\">");
                                                        out.println("<title>Pocket
Visit</title>");
                                                        out.println("</head>");
                                                        out.println("");
                                                        out.println("<body>");
                                                        out.println("");
                                                        out.println("<p><font
size=\"4\"> Hai raggiunto il poi itinerario "+denominazione[j] +".");
                                                        out.println("</font></p>");


   if( infoGuidaPOI.compareToIgnoreCase("nessuna")!= 0 ){


   out.println("<p><font size=\"4\">Devi fare la seguente azione :<font
color=\"#FF0000\">"+infoGuidaPOI+"</font> prima di continuare il percorso</font></p>");
                                                                out.println("");


                                                        }               //fine if

                                                        out.println("<p><font
size=\"4\">La tua posizione attuale è Latitudine:"+latitudineClient+"° N,
Longitudine:"+longitudineClient+"° E</font></p>");
                                                        out.println("");
                                                        out.println("</body>");
                                                        out.println("");
                                                        out.println("<P><FONT
face=\"Palatino Linotype\" size=4>     <B>Clicca qui per
continuare il percorso.<A href=\""+urlServlet+"\" target=_top>OK</A></B></FONT></P>");

                                                        //out.println("<p><font
face=\"Palatino Linotype\" size=\"4\">      Clicca qui per
continuare il percorso.<a href=\""+urlServlet+"\" target=\"_top\">OK</a></p>");


                                                        out.println("</html>");
                                                        out.flush();
                                                        out.close();

                                                        try { Thread.sleep(5000); }
catch (InterruptedException e) { }

                                                }//fine else
```

```
                                        }//fine if

                                        else{       //non siamo all'interno dell'area d' di un
poi,ma c'è sempre un traguardo

                                                    //si visualizza la pagina guida al turista
dicendo che l'azione da fare è infoGuida che

                                                    //il poi da raggiungere dista distanza e
si chiama denominazione[j]

                                                    out.println("<html>");
                                                    out.println("<head>");
                                                    out.println("<meta http-
equiv=\"Content-Language\" content=\"it\">");
                                                    out.println("<meta http-
equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\">");

                                                    //serve per fare il server push
CONTENT=\"serverPushIntervallo; indica che l'aggionamento avviene ogni
'serverPushIntervallo' secondi

                                                    out.println ("<meta http-
equiv=\"Refresh\" CONTENT=\""+serverPushIntervallo+";URL="+urlServlet+"\"> ");
                                                    out.println("<title>Pocket
Visit</title>");
                                                    out.println("</head>");
                                                    out.println("");
                                                    out.println("<body>");
                                                    out.println("");
                                                    out.println("<p><font size=\"4\"> Il
POI che devi raggiungere è "+denominazione[j] +" ad una distanza di
"+twoDigits.format((distanza*1000)) +" metri.");
                                                    out.println("</font></p>");
                                                    out.println("<p><font size=\"4\">La
tua posizione attuale è Latitudine:"+latitudineClient+"° N, Longitudine:"+longitudineClient+"°
E</font></p>");
                                                    out.println("");

                                                    //qui va inserito il link alla servlet in
maniera che aggiorna tutto

                                                    out.println("<p><font size=\"4\"> Per
raggiungere tale POI devi <font color=\"#FF0000\">"+infoGuida+".");
                                                    out.println("</font></p>");
                                                    out.println("</body>");
                                                    out.println("");
                                                    out.println("</html>");
                                                    out.flush();
                                                    out.close();

                                                    try { Thread.sleep(5000); } catch
(InterruptedException e) { }

                                        }//fine else

                                }//fine if-else controllo fine percorso

                        }//fine else

                }//fine try

                catch (Exception e){

                        System.err.println("Errore__3");
                        System.err.println(e.getMessage()+"/n"+e.getLocalizedMessage()
+"/n"+e.getCause());
                }//fine catch

                try {

                        stmt.close();

                        con.close();

                        System.out.println("Chiusura connessione__3");
```

```
                }//fine try

                catch (SQLException e1) {

                        // TODO Auto-generated catch block

                        e1.printStackTrace();

                }//fine catch

    }//fine metodo doGet

} //fine servlet
```

A.6 Applicazione SystemManage

A.6.1 InterfacciaDataBase.java

```
package packageAmministrazione;

import java.util.Vector;
import java.sql.*;
import javax.swing.DefaultListModel;
import javax.swing.JOptionPane;
import javax.swing.table.AbstractTableModel;


/**
 *
 * Classe che implementa l'interfaccia al database
 * @author Pietro Amato
 *
*/
public class InterfacciaDataBase extends AbstractTableModel {

   Connection          connection;
   Statement           statement;
   PreparedStatement   preparedStatement;
   ResultSet           resultSet;
   String[]            columnNames = {};
   Vector                      rows = new Vector();
   ResultSetMetaData   metaData;

   private String url = "jdbc:mysql://169.254.95.238:3306/entedbguidabis";
   private String driverName = "com.mysql.jdbc.Driver";

   //costruttore che effettua la connessione
   public InterfacciaDataBase(String user, String passwd) {
      try {

         Class.forName(driverName);

         System.out.println("Opening db connection");


         try{
                connection = DriverManager.getConnection(url, user, passwd);
         }
         catch(ExceptionInInitializerError e){

                JOptionPane.showMessageDialog(null,"Connessione negata.\nVerificate
username e password","Attenzione",JOptionPane.WARNING_MESSAGE);
                Login l1 = new Login();
         }
         catch(NullPointerException e){

                JOptionPane.showMessageDialog(null,"Connessione negata.\nVerificate username
e password","Attenzione",JOptionPane.WARNING_MESSAGE);
                Login l1 = new Login();
         }
         catch(NoClassDefFoundError e){
```

```
                JOptionPane.showMessageDialog(null,"Connessione negata.\nVerificate username
e password","Attenzione",JOptionPane.WARNING_MESSAGE);
                Login l1 = new Login();
            }

            statement = connection.createStatement();
        }
        catch (ClassNotFoundException ex) {

            System.err.println("Cannot find the database driver classes.");
            System.err.println(ex);
        }
        catch (SQLException ex) {

            System.err.println("Cannot connect to this database.");
            System.err.println(ex);
        }

        JOptionPane.showMessageDialog(null,"L'utente è stato autenticato","Accesso
consentito",JOptionPane.INFORMATION_MESSAGE);

    }

    /**
     * Metodo che esegue le query al database
     * @param query
     */
    public void executeQuery(String query) {
        if (connection == null || statement == null) {

                JOptionPane.showMessageDialog(null,"Non c'è il database per eseguire la
query","Attenzione",JOptionPane.WARNING_MESSAGE);

            System.err.println("There is no database to execute the query.");

            return;
        }
        try {
            resultSet = statement.executeQuery(query);
            metaData = resultSet.getMetaData();

            int numberOfColumns =  metaData.getColumnCount();
            columnNames = new String[numberOfColumns];
            // Get the column names and cache them.
            // Then we can close the connection.
            for(int column = 0; column < numberOfColumns; column++) {
                columnNames[column] = metaData.getColumnLabel(column+1);
            }

            // Get all rows.
            rows = new Vector();
            while (resultSet.next()) {
                Vector newRow = new Vector();
                for (int i = 1; i <= getColumnCount(); i++) {
                newRow.addElement(resultSet.getObject(i));
                }
                rows.addElement(newRow);
            }
            //  close(); Need to copy the metaData, bug in jdbc:odbc driver.
            fireTableChanged(null); // Tell the listeners a new table has arrived.
        }
        catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

            System.err.println(ex);
        }
    }

 //serve per eseguire la query e ritornare la tabella con la posizione attuale dei turisti presenti


/**
 * Metodo che crea la tabella che recupera la situazione attuale ovvero
 * l'ip e la posizione di tutti i turisti presenti nel parco
 *
```

```
 */
public void creaTabellaSituazioneAttuale()     {

//              si deve fare una query che mi ritorna tutti gli IP dei turisti la cui ora uscita is null

                //a questo punto per ogni turista presente si va a recuperare  con rs.last e si richiede
l'ultima posizione
                //e per tutti i turisti si memorizza in un vettore

                //Select ip,IDantenna from turista where oraUscita is null;

   //           Connection con = null;

   //           Statement stmt = null;
   //           PreparedStatement stmt1 = null;
  //            ResultSet rs = null;
  //            ResultSet rs1 = null;
                int numeroTuristiPresenti = 0;
                String[] ipTuristiPresenti = null;
/*
                try             {
                                //    Caricamento del driver
                                Class.forName(driverName);
//                              Apertura della connessione
                                //String url = ("jdbc:mysql://localhost/enteDB2?user=root");
                                con = DriverManager.getConnection(url);

   }
                catch (ClassNotFoundException ex) {

                JOptionPane.showMessageDialog(null,"Inpossibile trovare i driver per il
database.","Errore",JOptionPane.ERROR_MESSAGE);

      System.err.println("Cannot find the database driver classes.");
      System.err.println(ex);
   }


                catch (SQLException ex) {

                JOptionPane.showMessageDialog(null,"Inpossibile connettersi al
database.","Attenzione",JOptionPane.WARNING_MESSAGE);

      System.err.println("Cannot connect to this database.");
      System.err.println(ex);
   }
                */

                try {

//                  Esecuzione dell'interrogazione

                                statement = connection.createStatement();

                  resultSet = statement.executeQuery("select IP from turista where oraUscita is
null");


                    //si costruisce il vettore che conterra il risultato da visualizzare
         rows = new Vector();

         while(resultSet.next()){

                numeroTuristiPresenti++;

         }
  if(numeroTuristiPresenti == 0) {
   JOptionPane.showMessageDialog(null,"Non c'è nessun utente presente nel
parco.","Attenzione",JOptionPane.WARNING_MESSAGE);
    return ;
  }


         ipTuristiPresenti = new String[numeroTuristiPresenti];

         resultSet.first();
         int i = 0;
```

```
        do {

                ipTuristiPresenti[i]= resultSet.getString(1);

                i++;
        } while (resultSet.next());

                 }//fine try

                catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

         System.err.println(ex);
      }


                for(int e = 0 ; e < numeroTuristiPresenti; e++){

                try {

                                preparedStatement = connection.prepareStatement("Select
posizione.ipTurista,posizione.ora,posizione.latitudine,posizione.longitudine from posizione
where posizione.ipturista = ?");
         //per tutti i turisti presenti recuperare la posizione attuale

                                String ip = new String(ipTuristiPresenti[e].toString());

                                preparedStatement.setString(1,ipTuristiPresenti[e].toString());

                  resultSet = preparedStatement.executeQuery();

                  if(e == 0){
                                 metaData = resultSet.getMetaData();

        int numberOfColumns =  metaData.getColumnCount();
        columnNames = new String[numberOfColumns];
        // Get the column names and cache them.
        // Then we can close the connection.
        for(int column = 0; column < numberOfColumns; column++) {
           columnNames[column] = metaData.getColumnLabel(column+1);
        }

                  }

                  Vector newRow = new Vector();
                   //ci permette di selezionare la posizione attuale

                  //questo if serve per tutti i turisti registrati ma che non hanno la posizione
disponibile
                if ( !resultSet.last()){
                                newRow = new Vector();
                                newRow.addElement(ip);
                                for (int i = 2; i <= getColumnCount(); i++) {
                        newRow.addElement("non disponibile");
                }
                                }
   else{
                 //allochiamo la memoria per una nuova riga temporanea
              newRow = new Vector();
              //aggiungiamo tutte le colonne che andranno visualizzate per ogni turista presente

              for (int i = 1; i <= getColumnCount(); i++) {
                                newRow.addElement(resultSet.getObject(i));
                }

   }//fine else
          /* newRow.addElement(rs.getObject(1));
             newRow.addElement(rs.getObject(2));
             newRow.addElement(rs.getObject(3));
             newRow.addElement(rs.getObject(4));
             */
             //aggiungiamo la riga temporanea all'oggetto row
             rows.addElement(newRow);
             System.out.println(rows.toString());
```

```
                        }
                        catch (SQLException ex) {
                        JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione
della query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

                    System.err.println(ex);
                }


        }

        //  close(); Need to copy the metaData, bug in jdbc:odbc driver.
        fireTableChanged(null); // Tell the listeners a new table has arrived.


            }



/**
 * Metodo che permette di associare l'idturista alla posizione,in maniera da poter effettuare
 * delle selezioni utilizzando come chiave di ricerca l'id del turista
 */
public void associaTuristaPosizione(){

    try {

//      Esecuzione dell'interrogazione
        statement = connection.createStatement();
        int numeroAggiornamenti = statement.executeUpdate("UPDATE posizione,turista SET
posizione.IDturista = turista.IDturista WHERE turista.oraIngresso <= posizione.ora AND
turista.oraUscita >= posizione.ora AND posizione.IPturista = turista.IP");
        }
        catch (SQLException ex) {

                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);
            System.err.println(ex);
        }

   }


/**
 * Metodo che ritorna la lista delle antenne registrate nel database
 * @return DefaultListModel
 */
public DefaultListModel recuperaListaAntenna(){

   DefaultListModel data = null ;
    data = new DefaultListModel();

try {

//  Esecuzione dell'interrogazione
   statement = connection.createStatement();
    resultSet = statement.executeQuery("select IDantenna from antenna");

  while(resultSet.next()){
     data.addElement((resultSet.getString(1)));
                }
  }

  catch (SQLException ex) {
    JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della query.Verificare i
dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

     System.err.println(ex);
  }
 return data;
}


/**
 * Metodo che ritorna la lista delle azioni registrate nel database
 * @return
```

```
 */
public DefaultListModel recuperaAzioni(){

   DefaultListModel data = null ;
    data = new DefaultListModel();

try {

//  Esecuzione dell'interrogazione
   statement = connection.createStatement();
    resultSet = statement.executeQuery("select azionePOI from azione");

  while(resultSet.next()){
     data.addElement((resultSet.getString(1)));
                }
  }

  catch (SQLException ex) {
   JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della query.Verificare i
dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

     System.err.println(ex);
  }
 return data;
}



/**
 * Metodo che ritorna la lista dei POI con la sola denominazione
 * @return
 */
public DefaultListModel recuperaListaPOI(){

   DefaultListModel data = null ;
    data = new DefaultListModel();


try {

//  Esecuzione dell'interrogazione
  statement = connection.createStatement();
  resultSet = statement.executeQuery("select denominazione from POI");

  while(resultSet.next()){
     data.addElement((resultSet.getString(1)));
                }
  }

  catch (SQLException ex) {
   JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della query.Verificare i
dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

     System.err.println(ex);
  }
  return data;
}


/**
 * Metodo che ritorna la lista dei percorsi costituita dai nomi dei percorsi
 * @return
 */
public DefaultListModel recuperaListaPercorsi(){

   DefaultListModel data = null ;
    data = new DefaultListModel();


try {

//  Esecuzione dell'interrogazione
  statement = connection.createStatement();
  resultSet = statement.executeQuery("select nomePercorso from Percorso");

  while(resultSet.next()){
     data.addElement((resultSet.getString(1)));
```

```
                }
  }

  catch (SQLException ex) {
    JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della query.Verificare i
dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

      System.err.println(ex);
  }
  return data;
}


/**
 * Metodo che permette di modificare i dati relativi ad un'antenna
 * @param data Parametro che contiene tutti i dati relativi alle antenne
 * @param antenna Parametro che rappresenta l'id dell'antenna da cancellare
 */
public void modificaAntenna(Object data[][],int antenna){

     try {
//     Esecuzione dell'interrogazione
       preparedStatement = connection.prepareStatement("Update antenna set stato =? WHERE
IDantenna = ?");
       String IDantenna=new String(data[antenna][0].toString());
       String stato=new String(data[antenna][1].toString());
       preparedStatement.setString(1,stato);
       preparedStatement.setString(2,IDantenna);
       preparedStatement.executeUpdate();
       }

     catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

         System.err.println(ex);
     }
   }


/**
 * Metodo che permette di modificare i dati relativi ad un'azionePOI
 * @param data Parametro che contiene tutti i dati relativi alle azionePOI
 * @param azionePOI Parametro che rappresenta l'id dell'azionePOI da cancellare
 */
public void modificaAzionePOI(Object data[][],int azionePOI){

     try {
//     Esecuzione dell'interrogazione
       preparedStatement = connection.prepareStatement("Update azione set azionePOI =?
WHERE IDazionePOI = ?");
       String IDazionePOI=new String(data[azionePOI][0].toString());
       String stato=new String(data[azionePOI][1].toString());
       preparedStatement.setString(1,stato);
       preparedStatement.setString(2,IDazionePOI);
       preparedStatement.executeUpdate();
       }

     catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

         System.err.println(ex);
     }
   }


/**
 * Metodo che cancella l'antenna che ha l'idantenna uguale al parametro idAntenna
 * @param idAntenna
 */
public void cancellareAntenna(String idAntenna){

   try {

//     Esecuzione dell'interrogazione
     statement = connection.createStatement();
```

```
    int resultSet = statement.executeUpdate("delete from antenna where IDantenna =
"'+idAntenna+"'");
        }

    catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

          System.err.println(ex);
      }


}


/**
 * Metodo che cancella l'azionePOI che ha l'idazionePOI uguale al parametro azionePOI
 * @param azionePOI
 */
public void cancellareAzionePOI(String azionePOI){

   try {

//     Esecuzione dell'interrogazione
    statement = connection.createStatement();

    int resultSet = statement.executeUpdate("delete from azione where azionePOI =
"'+azionePOI+"'");
        }

    catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

          System.err.println(ex);
      }


}



/**
 * Metodo che permette di cancellare un POI che ha la denominazione uguale al
 * parametro denominazione
 * @param denominazione
 */
public void cancellarePOI(String denominazione){

   try {

//     Esecuzione dell'interrogazione
    statement = connection.createStatement();

    int resultSet = statement.executeUpdate("delete from POI where denominazione =
"'+denominazione+"'");
        }

    catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

          System.err.println(ex);
      }


}


/**
 * Metodo che ritorna la lista delle antenne con tutti i propri attributi
 * @return
 */
public Object[][] recuperaListaAntennaAttributi(){
```

```
	Object[][] data =null;

	try {

		//Esecuzione dell'interrogazione
		statement = connection.createStatement();
		resultSet = statement.executeQuery("select IDantenna,stato from antenna");

		int numeroR = 0;

		while(resultSet.next()){
			numeroR++;
		}

		//Elaborazione del risultato

		resultSet.first();

		data = new Object[numeroR][2];

		for(int i = 0; i < numeroR ;i++){
			for(int e = 0; e < 2; e++){

								data[i][e] = new String(resultSet.getString(e+1));
								}
			resultSet.next();
		}

	}

	catch (SQLException ex) {

		JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

		System.err.println(ex);
	}

	return data;
}


/**
 * Metodo che ritorna la lista delle azionePOI con i propri attributi
 * @return
 */
public Object[][] recuperaListaAzionePOI(){


	Object[][] data =null;

	try {

		//Esecuzione dell'interrogazione
		statement = connection.createStatement();
		resultSet = statement.executeQuery("select IDazione,azionePOI from azione");

		int numeroR = 0;

		while(resultSet.next()){
			numeroR++;
		}

		//Elaborazione del risultato

		resultSet.first();

		data = new Object[numeroR][2];

		for(int i = 0; i < numeroR ;i++){
			for(int e = 0; e < 2; e++){

								data[i][e] = new String(resultSet.getString(e+1));
								}
			resultSet.next();
		}
```

```
    }

    catch (SQLException ex) {

                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

                System.err.println(ex);
    }

    return data;
}


/**
 * Metodo che ritorna la lista delle azionePOI con la sola denominazione
 * @return
 */
public Object[] recuperaAzionePOI(){


    Object[] data =null;

    try {

                //Esecuzione dell'interrogazione
                statement = connection.createStatement();
                resultSet = statement.executeQuery("select azionePOI from azione");

                int numeroR = 0;

                while(resultSet.next()){
                        numeroR++;
                }

                //Elaborazione del risultato

                resultSet.first();

                data = new Object[numeroR];

                for(int i = 0; i < numeroR ;i++){

                                                data[i] = new String(resultSet.getString(1));

                        resultSet.next();
                }

    }

    catch (SQLException ex) {

                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

                System.err.println(ex);
    }

    return data;
}



/**
 * Metodo che permette di modificare un POI che ha l'idPOI uguale al parametro poi
 * @param data Rappresenta la lista dei POI con i propri attributi
 * @param poi Rappresenta l'id del poi da modificare
 */
public void modificaPOI(Object data[][],int poi){

     try {
//     Esecuzione dell'interrogazione
       preparedStatement = connection.prepareStatement("Update POI set denominazione
=?,latitudine = ?,longitudine=?,url =?,stato=?,tipologia=?,raggioInteresse=? WHERE IDpoi
= ?");
       String denominazione = new String(data[poi][0].toString());
         String latitudine = new String(data[poi][1].toString());
```

```
        String longitudine = new String(data[poi][2].toString());
        String url = new String(data[poi][3].toString());
        String stato = new String(data[poi][4].toString());
        String tipologia = new String(data[poi][5].toString());
        String raggioInteresse = new String(data[poi][6].toString());
        String idPOI = new String(data[poi][7].toString());

        preparedStatement.setString(1,denominazione);
        preparedStatement.setString(2,latitudine);
        preparedStatement.setString(3,longitudine);
        preparedStatement.setString(4,url);
        preparedStatement.setString(5,stato);
        preparedStatement.setString(6,tipologia);
        preparedStatement.setString(7,raggioInteresse);
        preparedStatement.setString(8,idPOI);

        preparedStatement.executeUpdate();
      }

    catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

        System.err.println(ex);
    }
  }



/**
 * Metodo che ritorna la lista dei POI con tutti i propri attributi
 * @return
 */
public Object[][] recuperaListaPoiAttributi(){


  Object[][] data =null;

  try {

   //Esecuzione dell'interrogazione
   statement = connection.createStatement();
   resultSet = statement.executeQuery("select
denominazione,latitudine,longitudine,url,stato,tipologia,raggioInteresse,IDpoi from POI");

   int numeroR = 0;

   while(resultSet.next()){
                numeroR++;
            }

   //Elaborazione del risultato

   resultSet.first();

   data = new Object[numeroR][8];

   for(int i = 0; i < numeroR ;i++){
            for(int e = 0; e < 8; e++){
                if(resultSet.getString(e+1)==null){
                    data[i][e]="";
                }else{
                                data[i][e] = new String(resultSet.getString(e+1));
                                System.out.println(data[i][e]);
                                }

            }
            resultSet.next();
   }
   }



  /*for(int i = 0; i < numeroR ;i++){
            for(int e = 0; e < 8; e++){
                if(rs.getString(e+1)==null){
                    data[i][e]="";
```

```
                        }else{
                                data[i][e] = new String(rs.getString(e+1));
                                System.out.println(data[i][e]);
                                }
  *
  *
  * */




  catch (SQLException ex) {

                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

                System.err.println(ex);
  }

  return data;
}


/**
 * Metodo che permette di creare il primo nodo(anello)del percorso con i dati che riceve conme
parametri in ingresso
 * @param IDpoiTo Rappresenta l'id del poi da raggiungere
 * @param azionePoi Rappresenta l'azione da effettuare al raggiungimento del POI
 * @param idPercorso Rappresenta l'id del percorso a cui appartiene il nodo
 */
public void creareNodo(String IDpoiTo,String azionePoi,int idPercorso){
   try {

//      Esecuzione dell'interrogazione
      statement = connection.createStatement();
      //int resultSet = statement.executeUpdate("Insert into
nodo(IDpoiFrom,IDpoiTO,azionePoi,IDpercorso)
values('0','"+IDpoiTo+"','"+azionePoi+"','"+idPercorso+"')");
      int resultSet = statement.executeUpdate("Insert into nodo(IDpoiTO,azionePoi,IDpercorso)
values('"+IDpoiTo+"','"+azionePoi+"','"+idPercorso+"')");

         }

      catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);
          System.err.println(ex);
       }
   }


/**
 * Metodo che permette di creare un percorso dandogli come nome il parametro in ingresso;
 * ritorna l'id del percorso creato
 * @param nomePercorso
 * @return L'id del percorso creato
 */
public int creaPercorso(String nomePercorso){
   int idPercorso = 0;
   try {
//      Esecuzione dell'interrogazione
      statement = connection.createStatement();
      int resultSet = statement.executeUpdate("Insert into percorso(nomePercorso)
values('"+nomePercorso+"')");
         }

      catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);
          System.err.println(ex);
       }
      try {
//      Esecuzione dell'interrogazione
      statement = connection.createStatement();
      resultSet = statement.executeQuery("select IDpercorso from percorso where nomePercorso
='"+nomePercorso+"'");
```

```
      while(resultSet.next()){
         idPercorso = resultSet.getInt(1);
                                  }
    }

      catch (SQLException ex) {
                  JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);
           System.err.println(ex);
       }
      return idPercorso;
    }

/**
 * Metopdo che permette di creare l'ultimo nodo(anello) del percorso
 * con i dati che soi ricevono come parametri in ingresso
 * @param IDpoiFrom Rappresenta l'id del nodo
 * @param idPercorso Rappresenta l'id del percorso del nodo
 */
public void creareNodo(String IDpoiFrom,int idPercorso){
    try {

//      Esecuzione dell'interrogazione
      statement = connection.createStatement();
      //int resultSet = statement.executeUpdate("Insert into nodo(IDpoiFrom,IDpoiTO,IDpercorso)
values('"+IDpoiFrom+"','0','"+idPercorso+"')");
      int resultSet = statement.executeUpdate("Insert into nodo(IDpoiFrom,IDpercorso)
values('"+IDpoiFrom+"','"+idPercorso+"')");

          }

      catch (SQLException ex) {
                  JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);
           System.err.println(ex);
       }
    }

/**
 * Metodo che permette di creare un nodo ,che non sia il primo o l'ultimo,del percorso
 * con i dati che si ricevono come parametri d'ingresso
 * @param IDpoiFrom Rappresenta l'id del nodo di partenza
 * @param IDpoiTo Rappresenta l'id del nodo d'arrivo
 * @param azione
 * @param azionePoi Rappresenta l'azione da effettuare al raggiungimento del poi
 * @param idPercorso Rappresenta l'id del percorso
 */
public void creareNodo(String IDpoiFrom,String IDpoiTo,String azione,String azionePoi,int
idPercorso){
    try {

//      Esecuzione dell'interrogazione
      statement = connection.createStatement();
      int resultSet = statement.executeUpdate("Insert into
nodo(IDpoiFrom,IDpoiTO,azione,azionePoi,IDpercorso)
values('"+IDpoiFrom+"','"+IDpoiTo+"','"+azione+"','"+azionePoi+"','"+idPercorso+"')");

          }

      catch (SQLException ex) {
                  JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);
           System.err.println(ex);
       }
    }

/**
 * Metodo che permette di cancellare il percorso che ha il nome
 * uguale al parametro in ingresso
 * @param nomePercorso
 */
public void cancellarePercorso(String nomePercorso){
int IDpercorso = 0;
    try {

//      Esecuzione dell'interrogazione
```

```
    statement = connection.createStatement();

    resultSet = statement.executeQuery("select IDpercorso from percorso where nomePercorso
="'+nomePercorso+"'");

    while(resultSet.next()){
       IDpercorso = resultSet.getInt(1);
                              }


    int resultSet = statement.executeUpdate("delete from nodo where IDpercorso =
"'+IDpercorso+"'");

    int resultSet1 = statement.executeUpdate("delete from Percorso where nomePercorso =
"'+nomePercorso+"'");
        }

    catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

        System.err.println(ex);
     }
}

//ritorna false se si sono avuti problemi altrimenti true

/**
 * Metodo che permette di eseguire una query di aggiornamento nel database
 * @param query Rappresenta la query da eseguire
 */
public void execute(String query) {
     if (connection == null || statement == null) {
                JOptionPane.showMessageDialog(null,"Non c'è il database per eseguire la
query","Attenzione",JOptionPane.WARNING_MESSAGE);

        System.err.println("There is no database to execute the query.");
        return;
     }
     int numero = 0;
     try {
        numero = statement.executeUpdate(query);
        JOptionPane.showMessageDialog(null,"Operazione effettuata con
successo","Informazione",JOptionPane.INFORMATION_MESSAGE);

        }

     catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

        System.err.println(ex);
     }
  }

/**
 * Metodo che esegue la query che riceve come parametro in ingresso e ritorna
 * l'ora d'uscita del turista selezionato
 * @param query
 * @return
 */
public String executeQueryNotVisual(String query) {

            String oraUscita = null;

     if (connection == null || statement == null) {
                JOptionPane.showMessageDialog(null,"Non c'è il database per eseguire la
query","Attenzione",JOptionPane.WARNING_MESSAGE);

        System.err.println("There is no database to execute the query.");
        return null;
     }

     try {
        ResultSet rs =  statement.executeQuery(query);
        while (rs.next()){
        oraUscita = rs.getString("oraUscita");
```

```
                                        }
            }

        catch (SQLException ex) {
                    JOptionPane.showMessageDialog(null,"Errore durante l'esecuzione della
query.Verificare i dati inseriti.","Attenzione",JOptionPane.WARNING_MESSAGE);

            System.err.println(ex);
        }
    return oraUscita;
    }


/**
 * Metodo che chiude la connessione con il database
 * @throws SQLException
 */
public void close() throws SQLException {

        System.out.println("Closing db connection");
        resultSet.close();
        statement.close();
        connection.close();

    }

    protected void finalize() throws Throwable {
        close();
        super.finalize();
    }

    //////////////////////////////////////////////////////////////////////
    //
    //            Implementation of the TableModel Interface
    //
    //////////////////////////////////////////////////////////////////////

    // MetaData

/*
 * @see javax.swing.table.TableModel#getColumnName(int)
 */
public String getColumnName(int column) {
        if (columnNames[column] != null) {
            return columnNames[column];
        } else {
            return "";
        }
    }


    /* (non-Javadoc)
     * @see javax.swing.table.TableModel#getColumnClass(int)
     */
    public Class getColumnClass(int column) {
        int type;
        try {
            type = metaData.getColumnType(column+1);
        }
        catch (SQLException e) {
            return super.getColumnClass(column);
        }

        switch(type) {
        case Types.CHAR:
        case Types.VARCHAR:
        case Types.LONGVARCHAR:
            return String.class;

        case Types.BIT:
            return Boolean.class;

        case Types.TINYINT:
        case Types.SMALLINT:
        case Types.INTEGER:
            return Integer.class;
```

```
    case Types.BIGINT:
        return Long.class;

    case Types.FLOAT:
    case Types.DOUBLE:
        return Double.class;

    case Types.DATE:
        return java.sql.Date.class;

    default:
        return Object.class;
    }
}

public boolean isCellEditable(int row, int column) {
    try {
        return metaData.isWritable(column+1);
    }
    catch (SQLException e) {
        return false;
    }
}

public int getColumnCount() {
    return columnNames.length;
}

// Data methods

public int getRowCount() {
    return rows.size();
}

public Object getValueAt(int aRow, int aColumn) {
    Vector row = (Vector)rows.elementAt(aRow);
    return row.elementAt(aColumn);
}

public String dbRepresentation(int column, Object value) {
    int type;

    if (value == null) {
        return "null";
    }

    try {
        type = metaData.getColumnType(column+1);
    }
    catch (SQLException e) {
        return value.toString();
    }

    switch(type) {
    case Types.INTEGER:
    case Types.DOUBLE:
    case Types.FLOAT:
        return value.toString();
    case Types.BIT:
        return ((Boolean)value).booleanValue() ? "1" : "0";
    case Types.DATE:
        return value.toString(); // This will need some conversion.
    default:
        return "\""+value.toString()+"\"";
    }

}

public void setValueAt(Object value, int row, int column) {
    try {
        String tableName = metaData.getTableName(column+1);
        // Some of the drivers seem buggy, tableName should not be null.
        if (tableName == null) {
            System.out.println("Table name returned null.");
        }
        String columnName = getColumnName(column);
        String query =
```

```
                "update "+tableName+
                " set "+columnName+" = "+dbRepresentation(column, value)+
                " where ";
            // We don't have a model of the schema so we don't know the
            // primary keys or which columns to lock on. To demonstrate
            // that editing is possible, we'll just lock on everything.
            for(int col = 0; col<getColumnCount(); col++) {
                String colName = getColumnName(col);
                if (colName.equals("")) {
                    continue;
                }
                if (col != 0) {
                    query = query + " and ";
                }
                query = query + colName +" = "+
                    dbRepresentation(col, getValueAt(row, col));
            }
            System.out.println(query);
            System.out.println("Not sending update to database");
            // statement.executeQuery(query);
        }
        catch (SQLException e) {
            //     e.printStackTrace();
            System.err.println("Update failed");
        }
        Vector dataRow = (Vector)rows.elementAt(row);
        dataRow.setElementAt(value, column);

    }
}
```

A.7 Applicazione UserManage

A.7.1 RegistraTurista.java

```
package packageBiglietteria;

import java.sql.Timestamp;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import packageAmministrazione.InterfacciaDataBase;

/**
 * Classe interfaccia che permette di inserire
 * nei rispettivi campi i dati relativi alla registrazione dell'ingresso di un turista.
 * Per effettuare la registrazione bisogna cliccare il bottone registra mentre per uscire
 * il bottone indietro.
 *
 * @author Pietro
 *
 */
public class RegistraTurista extends JDialog {



  InterfacciaDataBase dataBase;

   private javax.swing.JPanel jContentPane = null;

   private JLabel nomeL = null;
   private JLabel cognomeL = null;
   private JLabel tipoDocumentoL = null;
   private JLabel numeroDocumentoL = null;
   private JLabel indirizzoIPL = null;
   private JLabel IDantennaL = null;
   private JTextField nomeF = null;
   private JTextField cognomeF = null;
   private JTextField tipoDocumentoF = null;
   private JTextField numeroDocumentoF = null;
   private JTextField indirizzoIPF = null;
   private JTextField IDantennaF = null;
   private JButton registraButton = null;
   private JButton indietroButton = null;

   /**
    * This is the default constructor
    */
   public RegistraTurista() {
            super();
            initialize();
   }
   /**
    * This method initializes this
    *
    * @return void
    */
   private void initialize() {
            this.setLocation(480, 300);
            this.setTitle("Registrazione ingresso turista");
            this.setContentPane(getJContentPane());
            this.setSize(666, 295);
            this.setVisible(true);
            this.addWindowListener(new java.awt.event.WindowAdapter() {
                     public void windowClosing(java.awt.event.WindowEvent e) {
                              new GestioneTuristi();
                              System.out.println("windowClosing()"); // TODO Auto-
generated Event stub windowClosing()
                     }
            });
   }
   /**
    * This method initializes jContentPane
    *
    * @return javax.swing.JPanel
    */
   private javax.swing.JPanel getJContentPane() {
            if(jContentPane == null) {
                     IDantennaL = new JLabel();
                     indirizzoIPL = new JLabel();
                     numeroDocumentoL = new JLabel();
```

```
                        tipoDocumentoL = new JLabel();
                        cognomeL = new JLabel();
                        nomeL = new JLabel();
                        jContentPane = new javax.swing.JPanel();
                        jContentPane.setLayout(null);
                        nomeL.setBounds(15, 15, 200, 25);
                        nomeL.setText("Nome:");
                        nomeL.setName("nomeL");
                        cognomeL.setBounds(15, 45, 200, 25);
                        cognomeL.setText("Cognome:");
                        cognomeL.setName("cognomeL");
                        tipoDocumentoL.setBounds(15, 75, 200, 25);
                        tipoDocumentoL.setText("Tipo documento:");
                        tipoDocumentoL.setName("tipoDocumentoL");
                        numeroDocumentoL.setBounds(15, 105, 200, 25);
                        numeroDocumentoL.setText("Numero documento:");
                        numeroDocumentoL.setName("numeroDocumentoL");
                        indirizzoIPL.setBounds(15, 135, 200, 25);
                        indirizzoIPL.setText("Indirizzo IP:");
                        indirizzoIPL.setName("indirizzoIPL");
                        IDantennaL.setBounds(15, 165, 200, 25);
                        IDantennaL.setText("IDantenna:");
                        IDantennaL.setName("IDantennaL");
                        jContentPane.add(nomeL, null);
                        jContentPane.add(cognomeL, null);
                        jContentPane.add(tipoDocumentoL, null);
                        jContentPane.add(numeroDocumentoL, null);
                        jContentPane.add(indirizzoIPL, null);
                        jContentPane.add(IDantennaL, null);
                        jContentPane.add(getNomeF(), null);
                        jContentPane.add(getCognomeF(), null);
                        jContentPane.add(getTipoDocumentoF(), null);
                        jContentPane.add(getNumeroDocumentoF(), null);
                        jContentPane.add(getIndirizzoIPF(), null);
                        jContentPane.add(getIDantennaF(), null);
                        jContentPane.add(getRegistraButton(), null);
                        jContentPane.add(getIndietroButton(), null);
                }
                return jContentPane;
}
/**
 * This method initializes nomeF
 *
 * @return javax.swing.JTextField
 */
private JTextField getNomeF() {
                if (nomeF == null) {
                        nomeF = new JTextField();
                        nomeF.setBounds(226, 15, 409, 25);
                }
                return nomeF;
}
/**
 * This method initializes cognomeF
 *
 * @return javax.swing.JTextField
 */
private JTextField getCognomeF() {
                if (cognomeF == null) {
                        cognomeF = new JTextField();
                        cognomeF.setBounds(226, 45, 409, 25);
                }
                return cognomeF;
}
/**
 * This method initializes tipoDocumentoF
 *
 * @return javax.swing.JTextField
 */
private JTextField getTipoDocumentoF() {
                if (tipoDocumentoF == null) {
                        tipoDocumentoF = new JTextField();
                        tipoDocumentoF.setBounds(226, 75, 409, 25);
                }
                return tipoDocumentoF;
}
/**
```

```
	 * This method initializes numeroDocumentoF
	 *
	 * @return javax.swing.JTextField
	 */
	private JTextField getNumeroDocumentoF() {
		if (numeroDocumentoF == null) {
			numeroDocumentoF = new JTextField();
			numeroDocumentoF.setBounds(226, 105, 409, 25);
		}
		return numeroDocumentoF;
	}
	/**
	 * This method initializes indirizzoIPF
	 *
	 * @return javax.swing.JTextField
	 */
	private JTextField getIndirizzoIPF() {
		if (indirizzoIPF == null) {
			indirizzoIPF = new JTextField();
			indirizzoIPF.setBounds(226, 135, 409, 25);
		}
		return indirizzoIPF;
	}
	/**
	 * This method initializes IDantennaF
	 *
	 * @return javax.swing.JTextField
	 */
	private JTextField getIDantennaF() {
		if (IDantennaF == null) {
			IDantennaF = new JTextField();
			IDantennaF.setBounds(226, 165, 409, 25);
		}
		return IDantennaF;
	}
	/**
	 * This method initializes registraButton
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getRegistraButton() {
		if (registraButton == null) {
			registraButton = new JButton();
			registraButton.setBounds(154, 205, 147, 31);
			registraButton.setText("Registra");


registraButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.Bev
elBorder.RAISED));
			registraButton.addActionListener(new java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {

				registra();

					System.out.println("actionPerformed()"); // TODO
Auto-generated Event stub actionPerformed()
				}
			});
		}
		return registraButton;
	}
	/**
	 * This method initializes indietroButton
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getIndietroButton() {
		if (indietroButton == null) {
			indietroButton = new JButton();
			indietroButton.setBounds(350, 205, 147, 31);
			indietroButton.setText("Indietro");


indietroButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.Bev
elBorder.RAISED));
			indietroButton.addActionListener(new java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {
```

```
                                                indietro();
                                                System.out.println("actionPerformed()"); // TODO
Auto-generated Event stub actionPerformed()
                                        }
                                });
                }
                return indietroButton;
    }
     public void indietro(){

                this.setVisible(false);

                new GestioneTuristi();

    }

    public void registra(){

          String nome = new String(nomeF.getText());
          String cognome = new String(cognomeF.getText());
          String tipoDocumento = new String(tipoDocumentoF.getText());
          String numeroDocumento = new String(numeroDocumentoF.getText());
                String IP = new String(indirizzoIPF.getText());
                String IDantenna = new String(IDantennaF.getText());
                java.sql.Timestamp ora = new Timestamp((new java.util.Date()).getTime());
                String oraIngresso = ora.toString();

          Login.dataBase.execute("INSERT INTO
turista(nome,cognome,tipoDocumento,numeroDocumento,oraIngresso,IP,IDantenna)
VALUES( '"+nome+"','"+cognome+"','"+tipoDocumento+"','"+numeroDocumento+"','"+oraIngr
esso+"','"+IP+"','"+IDantenna+"')");

                        this.setVisible(false);

                new GestioneTuristi();

          }

}  //  @jve:decl-index=0:visual-constraint="55,22"
```

Appendice B

Codice SQL

```
create database entedbguida;

GRANT INSERT,SELECT,UPDATE ON provaparticolare.* TO
client@localhost IDENTIFIED BY 'client';

use entedbguida;

create table antenna(
IDantenna char(6) not null,
stato char(2) default 'ON',
data datetime,
primary key(IDantenna)
)type = innoDB;

create table turista(
IDturista int(6) not null AUTO_INCREMENT,
IP char(16) not null,
nome varchar(20) not null,
cognome varchar(20) not null,
tipoDocumento varchar(30) not null,
numeroDocumento varchar(15) not null,
oraIngresso datetime not null,
oraUscita datetime ,
IDantenna char(6) not null references antenna(IDantenna),
Index(IDantenna),
foreign key(IDantenna) references antenna(IDantenna)
on update cascade,
primary key(IDturista)
)type = innoDB;

create table posizione(
latitudine decimal(11,6) not null,
longitudine decimal(10,6) not null,
ora datetime not null,
IDturista int(6) references turista(IDturista),
Index(IDturista),
IPturista char(16) not null,
foreign key(IDturista) references turista(IDturista)
on update cascade,
primary key(IPturista,ora)
)type = innoDB;

create table POI(
latitudine decimal(11,6) not null,
longitudine decimal(10,6) not null,
url varchar(255),
data datetime not null,
stato char(2) default 'ON',
tipologia varchar(30),
raggioInteresse decimal(12,2),
```

```
denominazione varchar(30) not null,
IDpoi int(5) not null AUTO_INCREMENT,
primary key(IDpoi)
)type = innoDB;

create table azione(
IDazione int(6) not null AUTO_INCREMENT,
azionePOI varchar(30) not null,
primary key(IDazione)
)type = innoDB;

create table percorso(
IDpercorso int(6) not null AUTO_INCREMENT,
nomePercorso varchar(20) not null,
primary key(IDpercorso)
)type = innoDB;

create table nodo(
IDpoiFrom int(5) references poi(IDpoi),
Index(IDpoiFrom),
IDpoiTo int(5) references poi(IDpoi),
Index(IDpoiTo),
azione varchar(30) not null,
azionePoi varchar(50) default 'nessuna',
IDanello int(5) not null AUTO_INCREMENT,
IDpercorso int(6) not null references percorso(IDpercorso),
Index(IDpercorso),
foreign key(IDpercorso) references percorso(IDpercorso)
on update cascade,
foreign key(IDpoiTo) references poi(IDpoi)
on update cascade,
foreign key(IDpoiFrom) references poi(IDpoi)
on update cascade,
primary key(IDanello)
)type = innoDB;

create table traguardoCorrente(
IDpercorso int(6) not null references percorso(IDpercorso),
Index(IDpercorso),
IPturista char(16) not null,
IDpoiFrom int(5) not null,
dataRegistrazione datetime not null,
IDpoiTo int(5) not null,
azione varchar(30) not null,
azionePoi varchar(50) default 'nessuna',
IDtraguardo int(5) not null AUTO_INCREMENT,
foreign key(IDpercorso) references percorso(IDpercorso)
on update cascade,
primary key(IDtraguardo)
)type = innoDB;
```

Bibliografia

[1] Rapporto Tecnico N.:12 RT-ICAR-PA-04-12.

[2] Sun Microsystems, “Java Servlets API”.

[3] Sun Microsystems, “Java JDBC API”.

[4] Sun Microsystems, “Java 1.1.8 API”.

[5] Sun Microsystems, “Java 1.4.1 API”.

[6] P. Atzeni,S. Ceri, S. Paraboschi, R. Torlone, Basi di dati2 ED, McGraw-Hill

[7] Vario materiale su Java dal sito http://www.mokabyte.com

[8] Jason Hunter, William Crawford, Java Servlet Programming 2Ed., O’Really 2001.