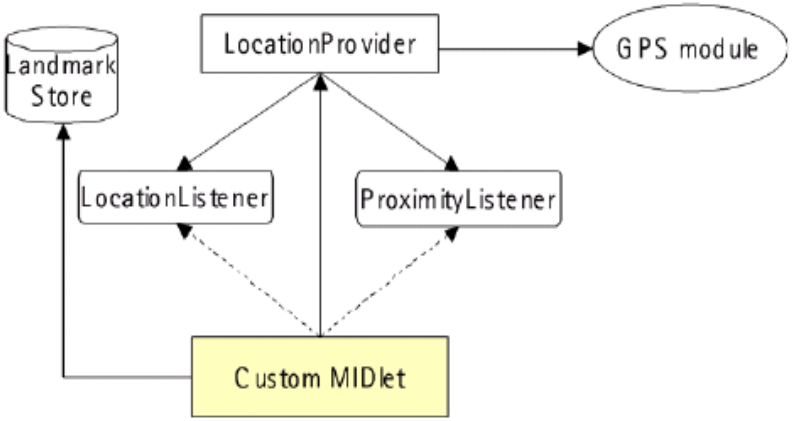
Progetti sviluppati all'interno del corso di Sistemi Mobili e Pervasivi a.a. 2006/2007

Road Blog

- Road Blog di Marco Salvatori
- Road Blog è un sistema per la gestione di un fotoblog con la possibilità di georeferenziare automaticamentele foto scattate. L'applicazione si compone di una midlet j2ME e di una parte server per la visualizzazione del blog.
- L'applicazione j2ME consente di scattare delle foto e di aggiungere dei commenti che poi verranno pubblicati nel blog. Una volta terminato il viaggio, viene generato un file XML che segue le specifiche di Google Maps per poter visualizzare le foto nelle mappe di Google. La midlet utilizza le Location API (jsr179) per la ricezione delle coordinate da un dispositivo GPS. Le possibilità della location API si possono riassumere essenzialmente in 4 punti, anche se le applicazioni e i possibili sviluppi che contiene ogni singolo punto sono veramente vaste (Geocoder, Routing, Mapping, Mobile Positioning). In questo caso specifico vengono utilizzate per il Mobile Positioning e per il tracking degli spostamenti durante un determinato viaggio.

Road Blog

- Cuore delle API è la classe LocationProvider che permette di ottenere, in modo del tutto trasparente (per quando riguarda il sistema di localizzazione utilizzato, a basso livello, dall'apparato), informazione sulla posizione dell'apparato in un dato momento. Si possono settare diversi criteri di ricerca delle coordinate, a seconda della precizione, della potenza, o del costo.
- Per ottenere le informazioni sulla posizione si può, in alternativa alla lettura diretta, utilizzare il listener LocationListener al quale viene notificato il cambiamento di locazione con l'invocazione del metodo locationUpdated.
- Per la memorizzazione dei parametri di setup dell'applicazione si utilizza l'RMS (Record Management Store), invece il salvataggio delle coordinate e delle foto scattate vier fatto direttamente sul filesystem, con l'uti delle File Connection API (jsr 75) per permettere all'utente di visualizzare le fo direttamente dal cellulare con qualsiasi visualizzatore di immagini.
- L'invio dei dati avviene tramite POST HTTP ad una servlet che li memorizza e li organizza per la visualizzazione.



Road Blog

- Screenshot

Schermata iniziale

Visualizzazione delle coordinate

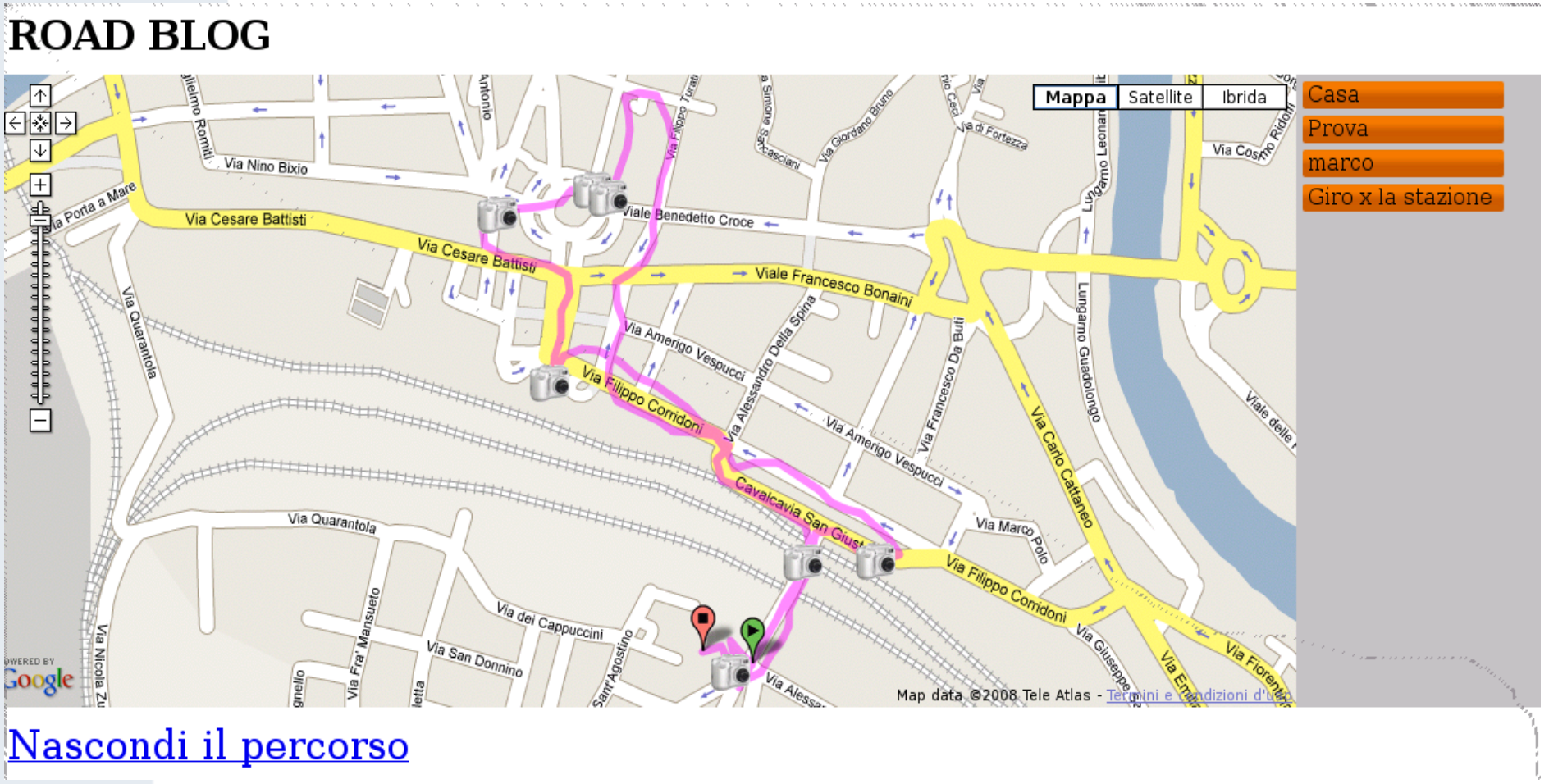
Fotografia

Road Blog

- Applicazione Web
- L'applicazione web è composta da una servlet che riceve i dati (foto piú un file coordinates.kml) dal dispositivo mobile e li memorizza in una cartella specifica.
- Inoltre c'è una servlet che genera un menú composto da tutti i viaggi inviati dal cellulare. Quando si seleziona un viaggio viene mostrata una mappa di Google Maps con i marker delle foto scattate e opzionalmente il tragitto effettuato durante l'utilizzo dell'applicazione.
- Per la gestione della mappa sono stati scritti degli script Javascript che, tramite Ajax, legge le informazioni nel file kml memorizzato e gestisce tutti gli eventi della mappa.
- Come supporto ai javascript è stata scritta un servlet che legge il file kml memorizzato nel filesystem e lo invia in uno stream alla pagina.

Road Blog

- Visualizzazione del percorso



Nascondi il percorso

Road Blog

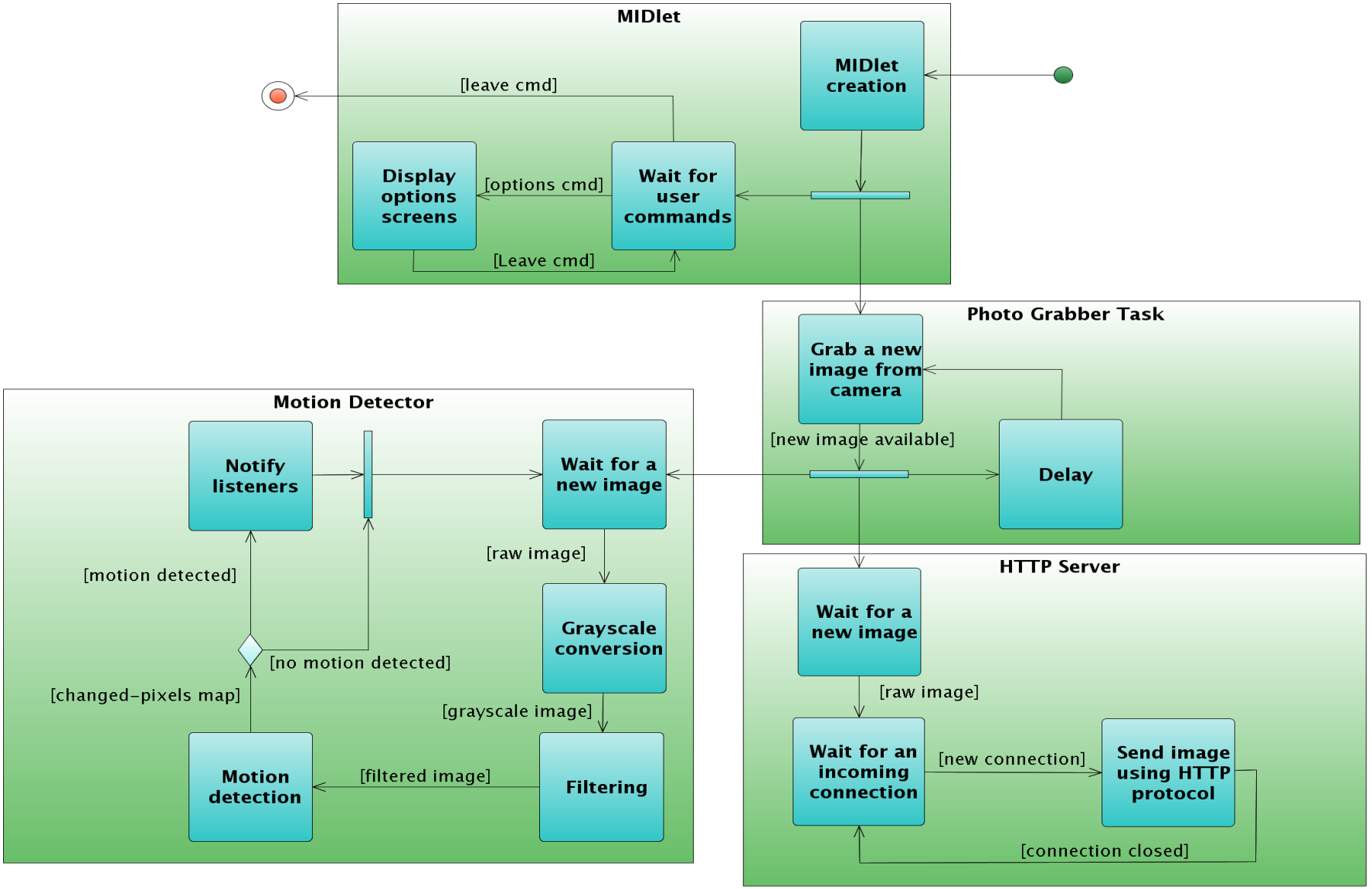
- Visualizzazione delle foto all'interno dei tooltip

Motion Detection

- Motion Detection di Danilo Levantesi
- L'applicazione nasce con lo scopo di poter utilizzare il proprio cellulare per monitorare i movimenti in un ambiente.
- La quasi totalità dei cellulari dispone di un dispositivo di acquisizione immagini e di potenza di calcolo sufficiente per elaborare le immagini acquisiti, il tutto ad un prezzo paragonabile o addirittura inferiore a quello di ip-webcam.
- L'idea è quella di posizionare il proprio cellulare in un determinato ambiente e di ricevere notifiche quando qualcosa nell'ambiente cambia. Il concetto di notifica può essere un SMS, un MMS con l'immagine che ha fatto scattare l'allarme oppure il semplice inviare l'immagine ad un server.
- Una semplice estensione è quella di poter sfruttare l'applicazione per il controllo da remoto, integrando un microserver HTTP in grado di fornire l'immagine attualmente acquisita.

Motion Detection

- Realizzazione



Motion Detection

- MIDlet
  - All'avvio la MIDlet principale MotionDetectorMIDlet, oltre a creare le classi per l'interfaccia grafica, si preoccupa di:
    - ottenere un riferimento di tipo VideoControl per poter acquisire immagini
    - far partire un task periodico PhotoGrabberTask che, ad intervalli regolari, acquisisca immagini
- PhotoGrabberTask
  - E' compito di questo task inviare l'immagine scattata sia al server http che al manager per la rilevazione del movimento.
- HTTPServer
  - La classe HTTPServer, che estende la classe Thread:
    - rimane in attesa di connessioni
    - invia l'immagine al client utilizzando il protocollo HTTP, ignorando lo stream di dati ricevuto dal client (per rendere l'operazione più veloce ed usare meno risorse)

Motion Detection

- MotionDetector
- La classe MotionDetector, che estende la classe Thread:
  - rimane in attesa di una nuova immagine (fornita attraverso il metodo newImage(byte[] rawCopy, short width, short height)
  - la trasforma in formato RGB
  - la converte in scala di grigi attraverso la conversione RGB=>YUV
  - esegue un eventuale filtraggio attraverso il metodo ImageFilter.filter
  - chiama il metodo MotionDetectionAlgorithm.motionDetection in modo da ottenere una mappa dei pixel cambiati e determina la percentuale dei pixel cambiati
  - se la percentuale è superiore ad una soglia impostata dall'utente, invia una copia dell'immagine ad ogni eventuale classe MotionDetectedAction interessata

Motion Detection

- ImageFilter
  - Le classi derivate da ImageFilter implementano un algoritmo di filtraggio dell'immagine. Attualmente sono presenti i seguenti filtri:
    - MedianFilter: filtro basato sulla mediana del valore dei pixel adiacenti al pixel considerato

Motion Detection

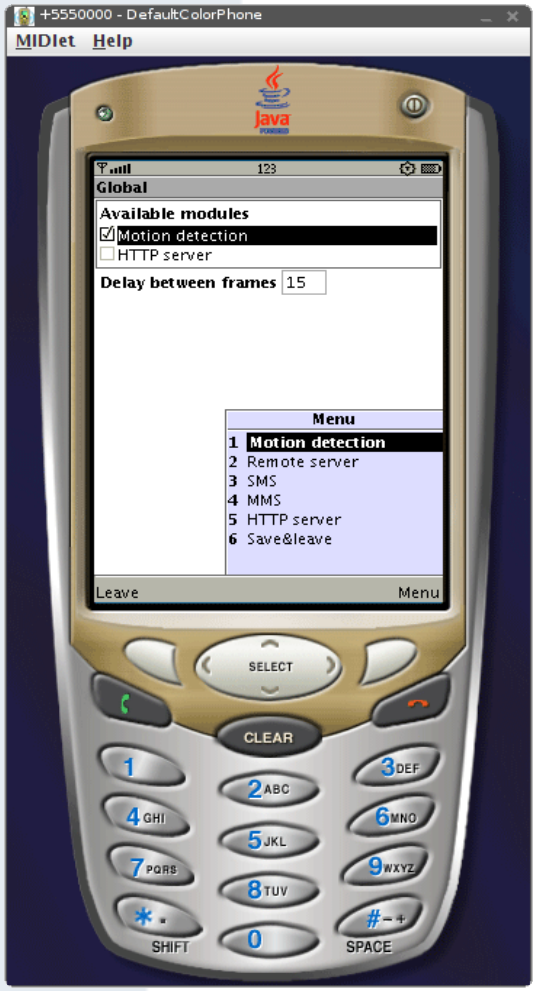
- MotionDetectionAlgorithm
  - Le classi derivate da MotionDetectionAlgorithm implementano un algoritmo di motion detection. Attualmente sono presenti i seguenti algoritmi:
    - SimpDiffAlgorithm: un pixel risulta in movimento se è dal relativo nell'immagine di riferiment
    - SimpDiffGNAlgorithm: la luminosità dell'immagine viene prima normalizzata (bwData[i] = (short)((refSigma/sigma)*(bwData[i] - mean) + refMean)), poi viene calcolato il valore della soglia con un algoritmo iterativo (si cerca il valore che rende massima la varianza relativa Vr=varianza/media, dove la varianza e la media sono calcolate sull'intensità dei pixel al di sopra della soglia in questione). Un pixel è in movimento se differisce dal relativo pixel di riferimento per un valore superiore alla soglia calcolata
    - MedMadAlgorithm: si ipotizza che il rumore sia bianco: si calcola la mediana MED e la deviazione assoluta della mediana MAD della differenza tra l'immagine di riferimento e la nuova immagine. La soglia T = MED + 2.5×1.4826×MAD viene usata per determinare i pixel in movimento. Viene inoltre applicato un filtro di post-processing basato su isteresi, in modo da eliminare pixel spuri.

Motion Detection

- MotionDetectedAction
  - Le classi derivate da MotionDetectedAction sono incaricate di inviare un messaggio di errore. Attualmente sono previsti i seguenti tipi di notifica:
    - SMS (MotionDetectedSMSAction)
    - MMS (MotionDetectedMMSAction)
    - Invio a server remoto (MotionDetectedRemoteAction)
- Comunicazione fra thread
  - E' stato implementato un meccanismo basato su eventi (classi derivate da MotionDetectorEvent) per la notifica dello stato dei vari thread alla MIDlet (in modo da avere un riscontro visivo delle operazioni in corso).

Motion Detection

- Screenshot



Opzioni generali

SMS

Motion detection

Necropoli

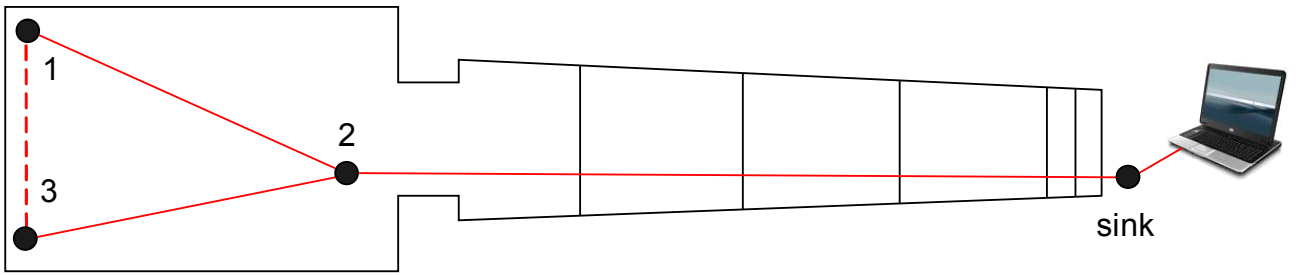
- Necropoli etrusca di Daniel Cesarini
- The Etruscan necropolis of Tarquinia is a well-known archaeological site. Situated on a ridge close to the ancient city, the necropolis contains more than one hundred painted tombs, dating from the 6th to the 4th century BCE. Tarquinia tomb frescos (Figure 1) are well preserved in many cases. Temperature and humidity are two important parameters of the tomb that should be monitored in order to preserve tomb frescos.

Necropoli

- A wireless sensor network has been deployed inside a tomb. The network is composed of TmoteSky nodes, by MoteIV. Three nodes were placed inside the tomb to gather humidity and temperature readings, and periodically communicate the values to the sink node. The sink node is placed outside the tomb, and is connected to a laptop.

Necropoli

- The structural characteristics of the tomb limited the transmission capabilities of the nodes: due to stone walls, ground, and shape of the tomb, only one node could directly communicate with the sink node. A detailed map of the monitored tomb, the placement of the nodes and the generated multi-hop network are shown in the figure.



Necropoli

- Temperature and humidity of the tomb were monitored for a period of 28 hours. The monitoring application has been developed for the TinyOS operating system. Nodes placed inside the tomb execute the fuzzy aggregator and BMAC+. The sink executes TOSBase, a library application of TinyOS, together with BMAC+, in order to bridge the wireless network to a laptop. According to the number of transmission envisioned with the analytical model of the monitoring application, the low power listening mode of BMAC+ was set up to 300 milliseconds in order to correctly balance energy consumption between transmissions and receptions. To evaluate the energy-saving performance of the whole monitoring application, sensor nodes included also a program that seamlessly included in the packets some statistics about the activity of the radio transceiver. Specifically, the following parameters were included into the packets: number of sent/received packets, total active time of the radio transceiver. At the end of the experiment, more than ten thousands packets were received by each node, and the results confirmed the validity of models and assumptions: even if the number of transmitted packets is almost double than the number of received packets, the activity of the radio transceiver is equally balanced between transmission and reception.

LastFMobile

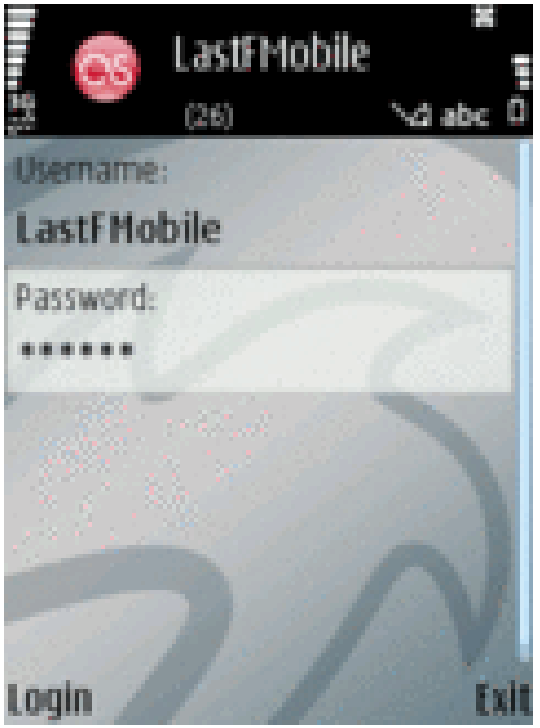
- LastFMobile di Luca Niccolini
- Last.fm è un servizio web che, grazie ad appositi plugin per i più diffusi lettori musicali, raccoglie informazioni riguardo ai gusti musicali dei suoi utenti. Grazie a questa funzione, denominata 'scrobbling', visitando il sito www.last.fm si può visualizzare il proprio profilo, visualizzare una lista di utenti con gusti musicali compatibili, conoscere nuovi artisti grazie al sistema di 'raccomandazione', informarsi sui concerti degli artisti preferiti, ascoltare radio personalizzate e altro ancora.

LastFMobile

- LastFMobile è un software per dispositivi mobili che funge da client per l'ascolto delle radio personalizzate di Last.fm. Per poter accedere ai servizi base è necessario effettuare una registrazione gratuita.
- Le radio attualmente supportate da LastFMobile sono le seguenti:
  - Neighbours - La radio degli utenti con profilo simile
  - Similar Artists - La radio degli artisti simili ad un artista specificato
  - Global tag - La radio della musica con etichetta specificata
  - Recommended - La radio con la musica raccomandata da Last.fm
  - Fans - La radio dei maggiori ascoltatori di un artista specificato
  - Personal - La tua radio personale
  - Loved - La radio con la tua musica preferita
- Queste ultime due stazioni sono accessibili soltanto da quegli utenti che hanno sottoscritto un abbonamento di (attualmente) 2,50€ mensili.

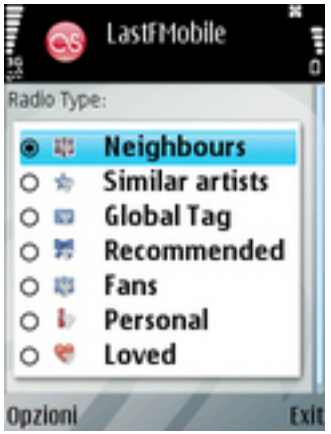
LastFMobile

- L'applicazione si compone di tre schermate (forms): una per il login, una per la scelta della stazione radio (tuning) e una per l'ascolto della stazione scelta.
- La form per il login si presenta nel modo classico:

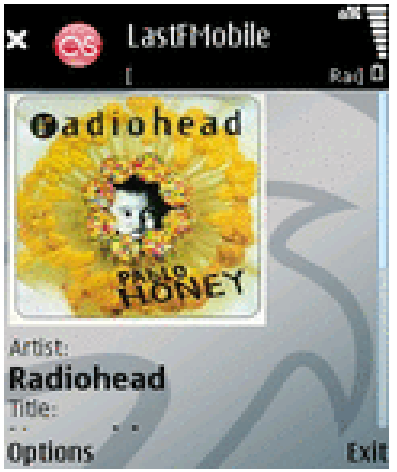


LastFMobile

- La form per il tuning varia in funzione del tipo di utente che ha effettuato il login:



- La form visualizzata durante l'ascolto della radio contiene il titolo della stazione e le informazioni sulla traccia corrente:



LastFMobile

- E' possibile inoltre di regolare il volume e di inviare a Last.fm i comandi Love e Ban che servono a specificare che si preferisce una certa canzone oppure che non la si vuole più ascoltare. Il proprio profilo musicale viene automaticamente aggiornato sulla base delle tracce ascoltate e dei comandi love e ban.

LastFMobile

- Requisiti
  - Il software è stato sviluppato utilizzando la piattaforma J2ME, compatibilmente con le specifiche CLDC1.1 e MIDP2.0. Dipende dalle librerie MMAPI e SATSA-CRYPTO che su alcuni dispositivi potrebbero non essere presenti.
- Note
  - La riproduzione dello stream audio presenta dei salti ad intervalli regolari. Questo problema è dovuto alle correnti implementazioni delle MMAPI che allo stato attuale non consentono la riproduzione di stream mp3 via http. La soluzione adottata prevede l'utilizzo di due istanze dell'oggetto javax.microedition.media.Player che si alternano facendo in modo che il buffer di una venga riempito mentre l'altra istanza è nello stato di play e viceversa. La frequenza dei salti nella riproduzione è funzione della dimensione del buffer.

Progetti sviluppati all'interno del corso di Sistemi Mobili e Pervasivi a.a. 2005/2006

Prossimo autobus per...

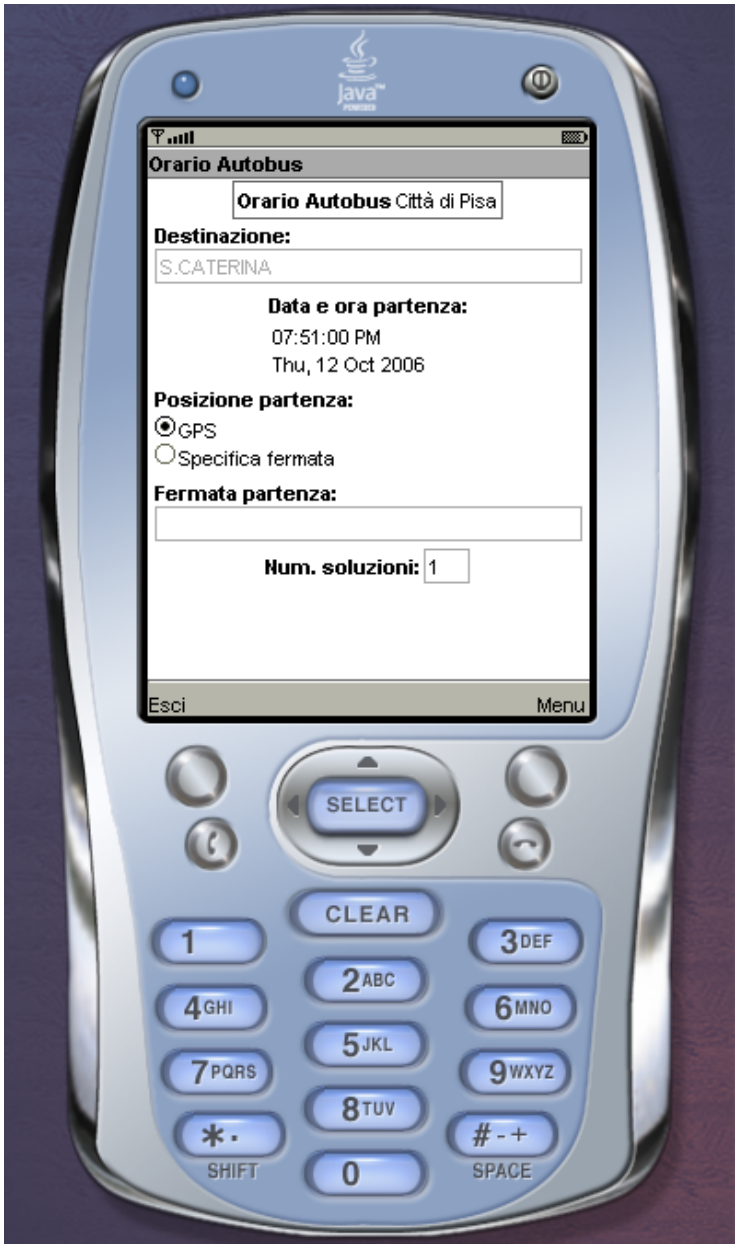
- Prossimo autobus di Ludovico Cavedon e Luca Foschini
- Ricerca del percorso più veloce in autobus tra 2 fermate.
- Previsti cambi di linea e tratte a piedi.
- Fermata di partenza specificabile anche tramite localizzazione GPS.
- Calcolo dei percorsi sulla base di informazioni aggiornate in tempo-reale sul ritardo attuale degli autobus.
- Test & tuning effettuato sulla rete urbana pisana

Prossimo autobus per...

- Struttura client-server
- Piattaforma: CLDC-1.1 MIDP-2.0
- Localization API (JSR-179) opzionale.
- Servlet su Tomcat.
- Comunicazione di dati binari su HttpConnection per:
  - richiesta utente (posizione partenza ed arrivo, data e ora);
  - soluzioni di viaggio trovate;
  - aggiornamento della lista fermate sul client.

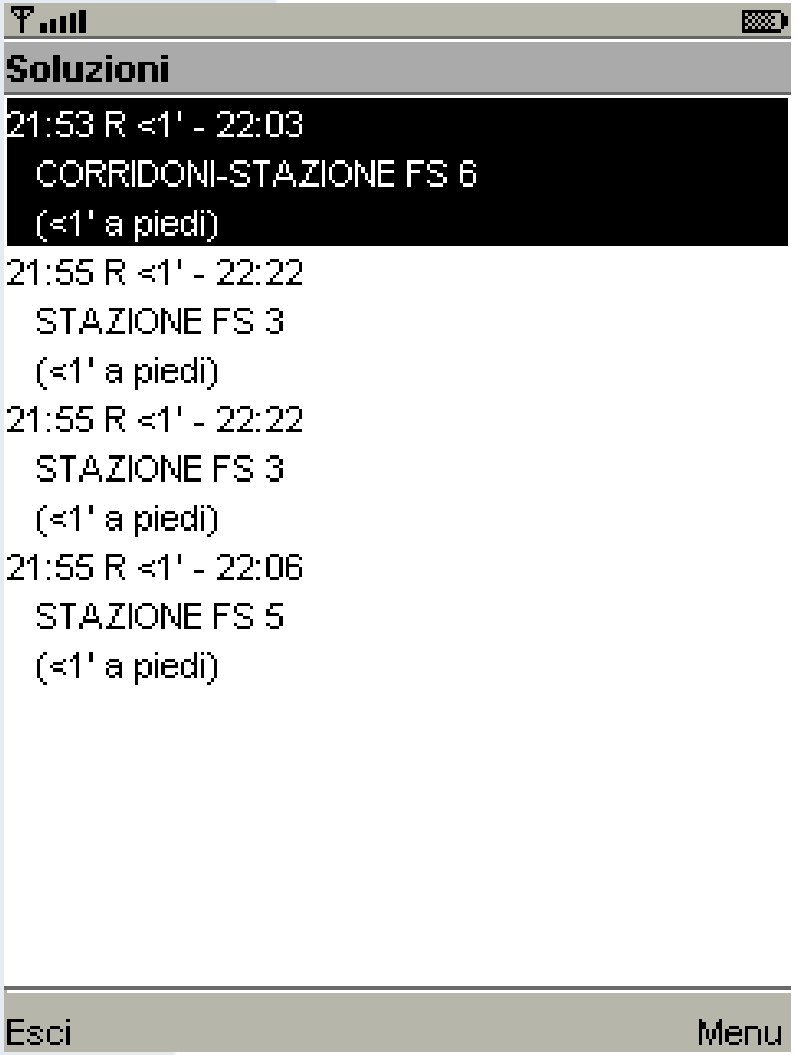
Prossimo autobus per...

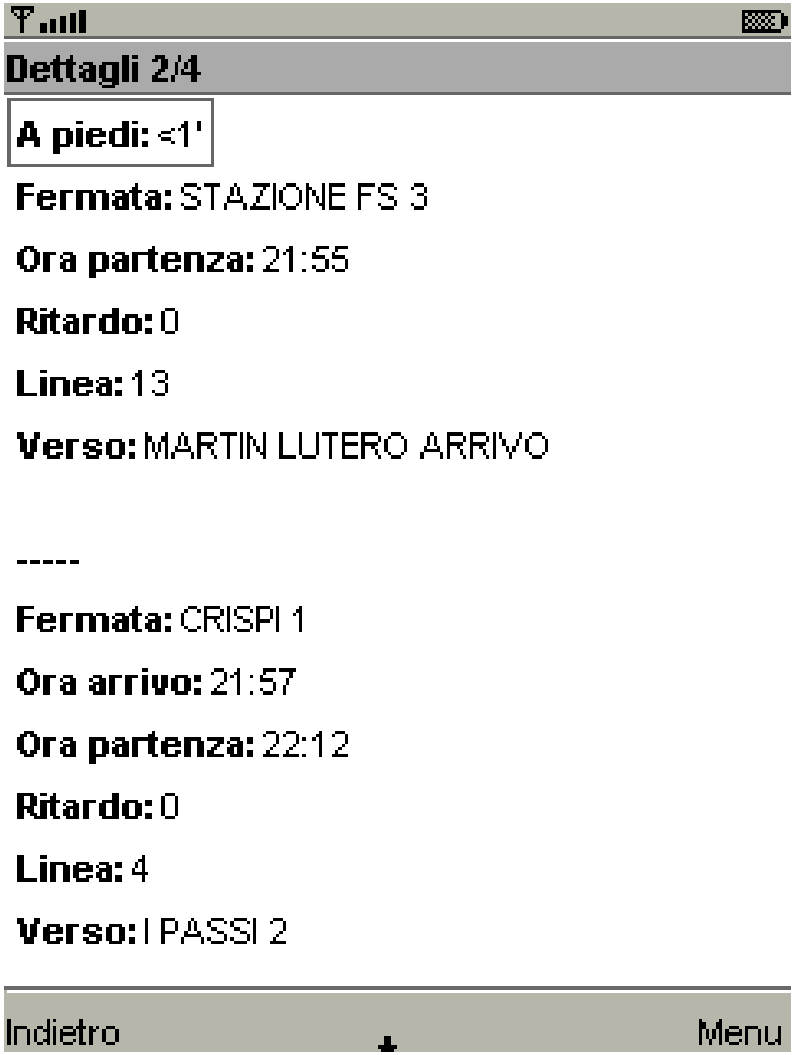
- Midlet:
  - raccolta dati dall'utente;
  - invio dei dati alla servlet via Internet;
  - ricezione e visualizzazione degli n percorsi più veloci;
  - thread dedicato alla comunicazione con la servlet.
- La posizione di partenza può essere ricevuta dal GPS.
  - viene indicata la fermata da cui parte il primo autobus.
  - la prima tratta del percorso è a piedi.
  - Requisiti del client: supporto per le Localization API (JSR-179); disponibilità di un dispositivo di localizzazione GPS.



Prossimo autobus per...

- Screenshot risultati



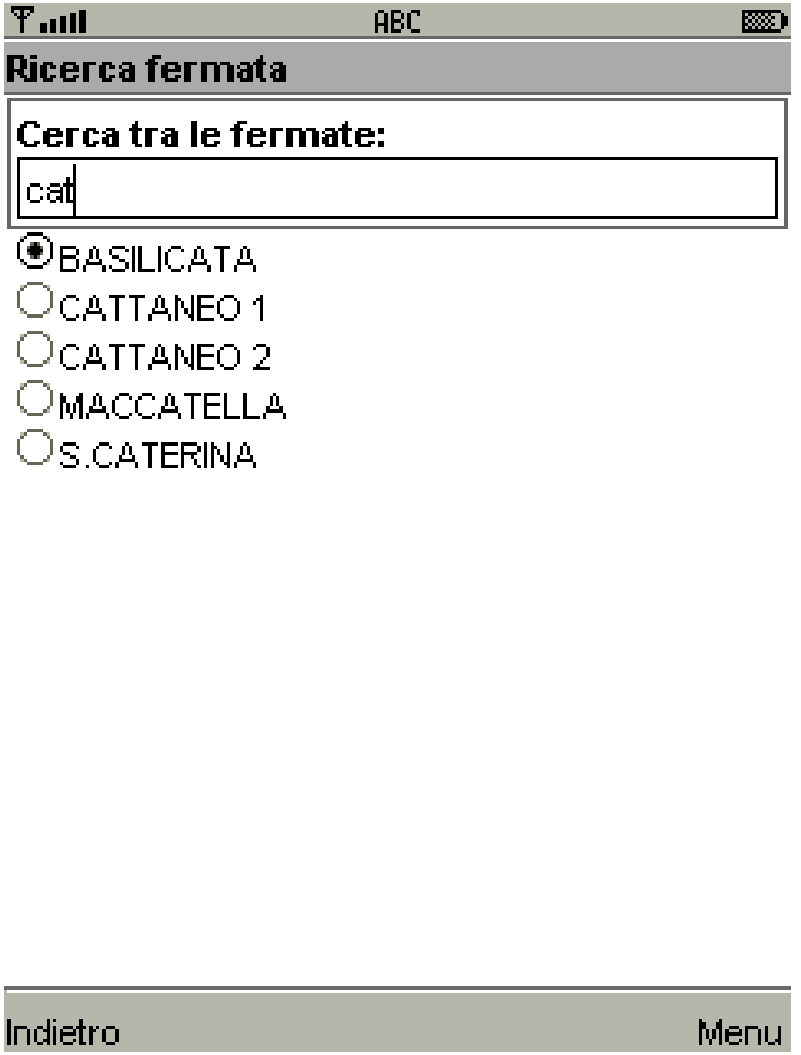


Prossimo autobus per...

- Comunicazione:
- Protocollo dedicato per minimizzare i dati scambiati (< 1KB).
- Trasmissione solo dei dati necessari tramite serializzazione ad-hoc delle classi.
- Il client possiede una lista di fermate:
  - trasmissione dell'ID delle fermate invece che della stringa
  - lista con versioning, aggiornabile tramite connessione ad un'ulteriore servlet.

Prossimo autobus per...

- Lista fermate
  - Residente nel client (10 KB).
  - Funzione di ricerca di sottostringhe.
  - Aggiornabile via rete.
  - Dotata di numero di versione per evitare il download se non necessario.



Prossimo autobus per...

- Grafo della rete degli autobus
- Applicazione di Dijkstra modificato sul grafo, dove:
  - nodi = fermate
  - archi = tratte di autobus da una fermata all'altra:
  - autobus di diversa linea sulla stessa tratta danno origine ad archi diversi;
  - autobus della stessa linea ad orari diversi sulla stessa tratta, danno origine ad archi diversi;
  - tra ogni coppia di fermate entro un certo raggio è possibile percorrere un ulteriore tratta a piedi.
- Fermate e tratte memorizzate in tabelle MySQL.
- Accesso tramite MySQLConnector.
- Ad ogni iterazione di Dijkstra viene interrogato il DB, correggendo i dati con i ritardi dei vari autobus:
- Ottimizzazione possibile: precare i dati sul grafo in RAM o almeno effettuare caching.

Prossimo autobus per...

- Correzione degli orari sulla base del ritrdo degli autobus
- Correzione dell'orario previsto di arrivo/partenza degli autobus con l'effettivo ritardo dell'autobus.
- Si suppone che l'autobus conservi il ritardo accumulato fino a quel momento.
- Si suppone tale informazione sia disponibile tramite un qualche sistema di tracciamento degli autobus (in fase di test i ritardi sono generati casualmente).

Prossimo autobus per...

- Post-processing del percorso
- Problema: l'algoritmo finora descritto può proporre soluzione “anomale” in cui si scende da un autobus per poi risalirci ad un'altra fermata
- Soluzione: il “best path” viene rielaborato dalla fermata di destinazione cercando di massimizzare la permenenza sullo stesso autobus

- Soluzioni successive
  - Calcolate supponendo che venga “perso” il primo autobus della solzione precedente
  - Ottenute eliminando dal grafo il primo arco delle soluzioni precedenti

St@tino

- St@atino di Paolo Caggiari
- Realizzazione di un'applicazione mobile in grado di sostituire gli attuali sistemi POS per la verbalizzazione e l'archiviazione degli statini d'esame.
- Struttura del progetto: una parte client e una parte server.
- Lato client: una MIDlet gestisce l'interazione con l'utente e, a seguito di un'autenticazione con codice docente e PIN, offre le seguenti funzionalità:
    - selezione di un esame tra quelli in cui il docente loggato figura come presidente di commissione
    - verbalizzazione di uno o più statini
    - memorizzazione sull'RMS locale o invio immediato del singolo statino creato al database remoto
    - annullamento di statini memorizzati localmente e non ancora inviati
    - sincronizzazione col database centrale, con invio in blocco di tutti gli statini locali

St@tino

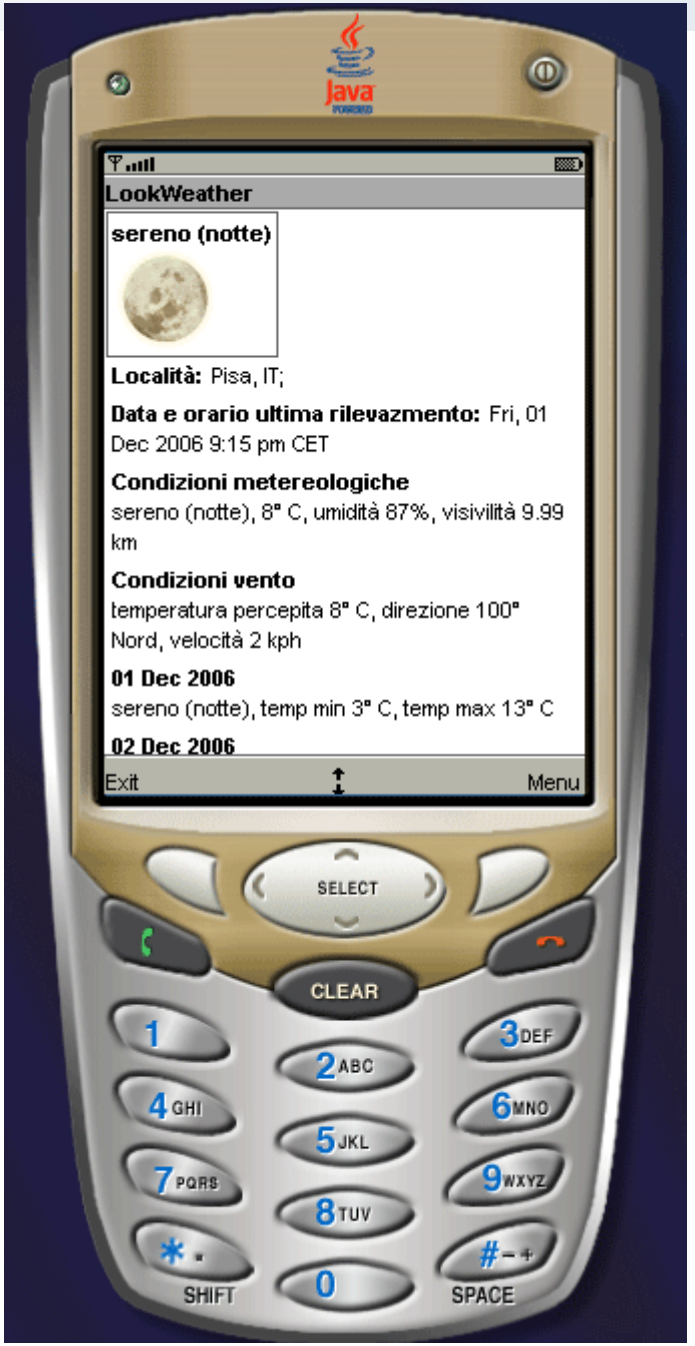
- Realizzazione di un'applicazione mobile in grado di sostituire gli attuali sistemi POS per la verbalizzazione e l'a
- Lato server
  - La parte lato server, completamente trasparente all'utente finale, consiste in una SERVlet che interagisce col sistema su due interfacce:
    - Locale: col database MySQL sul quale risiedono le informazioni relative a docenti, studenti, esami e statini archiviati. Tutte le operazioni di lettura e manipolazione dei dati sfruttano le funzionalità offerte dalle API JDBC.
    - Remota: con la MIDlet sul terminale mobile del docente, tramite connessioni HTTP.

St@atino

- Screenshot

LookWeather

- LookWeather di Francesco Giurlanda
- LookWeather è un applicativo per dispositivi mobili, realizzato in tecnologia java2me in grado di reperire informazioni sulle condizioni metereologiche in qualsiasi località sfruttando un servizio fornito su Internet del portale Yahoo!
- Nell'applicativo sono state implementate operazioni di selezione della località di cui si desidera conoscere le condizioni metereologiche, aggiunta e salvataggiodi nuove località e possibilità di impostazione delle unità di misura.



SIS.C.O.S.

- SIS.C.O.S. di Giovanni Ledda
- L'applicazione SIS.C.O.S. e' rivolta a tutte le organizzazioni (civili e non) che basano il proprio lavoro su attivita' di squadra che necessitano un coordinamento delle loro operazioni in tempo reale. Ci si riferisce ad esempio a squadre di vigili del fuoco o della protezione civile, corpi di tutela ambientale, squadre di anti-abigeato, e via dicendo. In particolare l'applicazione sviluppata nel progetto viene incontro all'esigenza di disporre in un unico device (telefono cellulare, palmare, etc.) di piu' funzionalita' ottenibili altrimenti solo a patto di disporre di piu' strumenti ingombranti e non sempre utilizzabili a causa di inevitabili limiti operativi.

SIS.C.O.S.

- SIS.C.O.S. (Sistema di Coordinamento per Operazioni di Squadra) consiste in una applicazione Java distribuita avente lo scopo di coordinare l'azione congiunta di piu' squadre che operano in ambiente aperto e sono dislocate in punti differenti di tale ambiente; ogni squadra, ovviamente, dovra' essere dotata di un dispositivo mobile sul quale sara' installata l'applicazione. Il supporto fornito dall'applicazione alle attivita' di ogni squadra viene implementato attraverso tre funzionalita' generali:
  - visualizzazione della posizione delle squadre nello scenario operativo mediante una mappa 2D
  - servizio di messaggeria istantanea tra squadre
  - visualizzazione delle informazioni tattiche sulle altre squadre.

SIS.C.O.S.

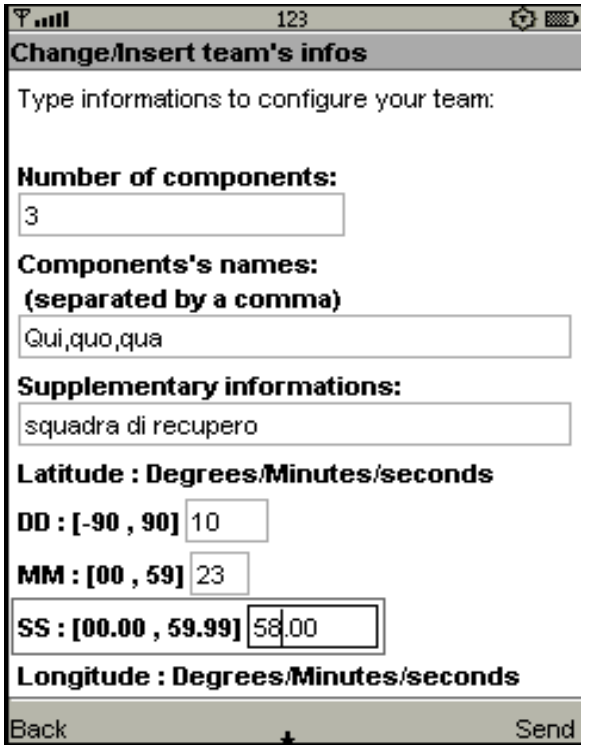
- Mediante l'utilizzo delle Location API (JSR 179) l'applicazione consente di rilevare la posizione geografica (espressa in coordinate geografiche) del dispositivo di squadra, e di mostrarla su una mappa a colori insieme alla posizione delle altre squadre operanti nell'ambiente circostante. La mappa relativa al settore operativo viene preliminarmente caricata su un server, per poi essere scaricata e visualizzata nel dispositivo al momento in cui la squadra desidera ottenere la disposizione tattica dell'intero gruppo all'interno dello scenario. Il risultato finale visualizzato nel display sara' una mappa di dimensioni 250 X 280 pixel nella quale, per ogni pixel relativo al punto geografico in cui staziona una squadra, sara' presente il numero simbolico rappresentante tale squadra.

SIS.C.O.S.

- Ogni squadra puo' inviare un messaggio testuale ad una qualsiasi delle squadre attive in un determinato istante. I messaggi sono preventivamente archiviati in una mailbox di squadra sul server e ogni squadra puo' decidere in seguito di scaricarli localmente sul proprio dispositivo. All'atto della ricezione di un messaggio, il server, mediante un Alert, avvisa la squadra destinataria della presenza di un nuovo messaggio nella sua mailbox remota, specificando nell'Alert la squadra che lo ha inviato.

SIS.C.O.S.

- Nel momento in cui la squadra si attiva nello scenario, effettua un procedura di login attraverso la quale comunica al server delle informazioni generali sul suo stato : numero e nomi dei componenti, posizione geografica, informazioni generali. I dati relativi a ogni squadra possono essere scaricati da una qualsiasi delle altre squadre sul proprio dispositivo.



Cardiofrequenzimetro

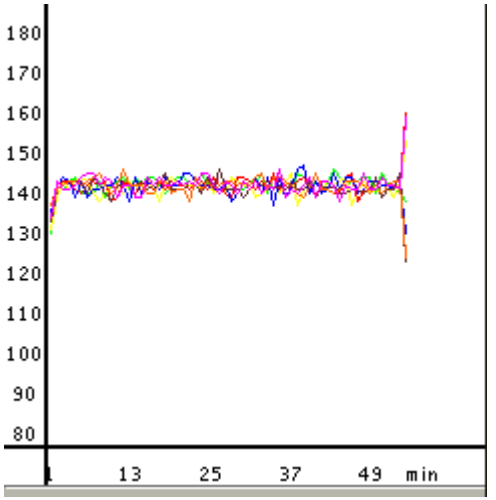
- Cardiofrequenzimetro di Bruno Rocco
- Il progetto prevede la realizzazione di un cardiofrequenzimentro bluetooth. Un cardiofrequenzimentro reale è composto da due moduli: un trasmettitore realizzato come fascia toracica con due elettrodi per la misurazione e l'invio delle pulsazioni cardiache a un ricevitore da polso che riceve le pulsazioni e le visualizza. Non essendo possibile realizzare fisicamente il trasmettitore, quest'ultimo è stato simulato tramite una midlet in connessione bluetooth.
-

Cardiofrequenzimentro

- Il ricevitore è un'altra midlet con le seguenti funzionalita:
  - Cardio: permette di visualizzare la pulsazione inviata dal trasmettitore e il tempo di allenamento. Una volta terminato l'allenamento visualizza il tempo totale e la pulsazione media ed è possibile viasualizzare un grafico dell'andamento temporale delle pulsazioni, salvare i risultati dell'allenamento e inviare questi ad una servlet remota.
  - Impostazioni Utente: prevede l'inserimento di nome, cognome, età, altezza, peso e in particolare pulsazione media d'allenamento. Questa permette al cardio l'emissione di un segnale sonoro e visivo se la pulsazione istantanea supera quella impostata.
  - Ultimi Dieci Allenamenti: mostra i risultati degli ultimi dieci allenamenti.
  - Calcolo Fitness Zone: sulla base dell'età e del tipo di allenamento, calcola la pulsazione massima e il range di pulsazioni entro quali bisogna allenarsi.
  - Fitness Session: prevede di creare delle sessioni di max 7 allenamenti della stessa durata, aprendo una sessione c'è la possibiltà di abilitare la funzionalità di cardio e in particolare di visualizzare un grafico comparativo dei precedenti allenamenti.

Cardiofrequenzimetro

- Screenshot



EBayChecker

- EbayChecker di Vitullo
- Funzionalità
  - L'applicazione consente di eseguire delle ricerche sul noto sito di compravendita eBay attraverso una midlet Java MIDP 2.0 che gira su un dispositivo mobile dotato di opportuna Java Virtual Machine; come sul sito, la ricerca viene effettuata specificando una o più parole chiave di interesse.
  - Per una data ricerca è possibile specificare il numero di risultati desiderati in una singola schermata (da 2 a 10); per ogni oggetto trovato vengono fornite informazioni quali:
    - Titolo esteso dell'inserzione
    - Prezzo corrente dell'oggetto
    - Tempo rimasto alla scadenza dell'inserzione
    - Testo descrittivo dell'oggetto

EBayChecker

- Struttura
- L'applicazione consta di 2 componenti: un server e un client.
  - Il client è una midlet Java MIDP 2.0 che inoltra le sue richieste verso il componente server seguendo un protocollo di comunicazione molto essenziale che mira a preservare le limitate risorse dei dispositivi mobili.
  - Il componente server gestisce le richieste di 1 o più client attraverso un meccanismo di multithreading. Il server gira su un calcolatore dotato di risorse di elaborazione sufficienti a gestire i numerosi e complessi oggetti dell'ambiente di sviluppo di eBay per Java.
    - Oltre alla gestione dei thread e all'invocazione delle API eBay per l'esecuzione dei servizi, il server mette a disposizione un'embrionale console di comando per il monitoraggio run-time e provvede al caching e alla paginazione dei risultati di una ricerca.

EBayChecker

- Per ogni utente diverso, il server riserva spazio sufficiente a conservare i risultati di un utente (il quale può richiedere i risultati stessi una pagina per volta). Tale meccanismo è stato implementato per ridurre il numero di API eBay invocate (sulle quali vige una tariffa economica se l'applicazione gira in ambiente di produzione - vedere prossimo paragrafo). Da notare, infatti, che la paginazione dei risultati può anche essere effettuata da eBay, ma ogni nuova pagina richiesta prevede l'invocazione di una funzione API (e quindi il pagamento della tariffa).

EBayChecker

- Sandbox vs Ambiente di produzione
  - L'ambiente di produzione è il sito di commercio elettronico eBay dove si concludono transazioni di compravendita reali e che prevedono reali scambi di denaro tra le parti coinvolte.
  - L'ambiente Sandbox è una copia esatta dell'ambiente di produzione, con la differenza che tutti i fondi monetari sono fittizi. E' possibile registrarvi un numero arbitrario di utenti, e ciascun utente può simulare la messa in vendita o l'acquisto di un oggetto.
  - Questo è l'ambiente che eBay mette a disposizione per il testing delle applicazioni. Il formato delle API e delle risposte del server è identico a quello dell'ambiente di produzione; ciò che cambia è l'indirizzo URL del server a cui vengono inoltrate le richieste.
  - Un'applicazione correttamente funzionante su Sandbox può passare ad operare nell'ambiente di produzione previa certificazione da parte di eBay.

Ticketless

- Ticketless di Chiara Boldrini e Iacopo Iacopini
- Realizzazione di un sistema ticketless in ambito ferroviario attraverso due principali componenti:
  - biglietteria virtuale che consente al viaggiatore l'acquisto dei biglietti tramite telefono cellulare
  - sistema di obliterazione virtuale che consente al controllore del treno di convalidare i tagliandi dei viaggiatori.
- Il sistema è suddiviso in una parte client e una parte server.

Ticketless

- Lato client: consiste di due applicazioni, una per il telefono del viaggiatore e una per il telefono del controllore
- L'applicazione consente al viaggiatore di:
  - interrogare un database con gli orari dei treni
  - acquistare biglietti
  - ottenere informazioni meteo sulla città di destinazione
  - ottenere la traduzione di un testo inserito
- L'applicazione consente al controllore di:
  - scaricare la lista dei biglietti acquistati per il treno su cui si appresta a prendere servizio
  - obliterare i biglietti
  - in maniera tradizionale, controllando visivamente che il codice del biglietto del cliente appartenga alla lista dei biglietti acquistati per quel treno
  - in maniera automatica attraverso una connessione bluetooth con il dispositivo del viaggiatore

Ticketless

- La parte lato server è composta da:
  - 2 servizi di interrogazione di un database remoto, uno dedicato al viaggiatore per l'acquisto dei biglietti e la consultazione dell'orario dei treni e uno dedicato al controllore per scaricare la lista dei biglietti acquistati per un determinato viaggio
  - 1 servizio di connessione al web service client che richiede informazioni meteo
  - 1 servizio di connessione al web service client che richiede il servizio di traduzione

Happy tourist

- Happy tourist di Anrea Cappelli e Francesca Mancini
- Happy Tourist è un'applicazione J2ME sviluppata per favorire il soggiorno di un turista in una città.
- Il software consente all'utilizzatore un agevole spostamento all'interno della stessa mettendo a disposizione una cartina sulla quale viene visualizzata la posizione dell'utente, ricavata dalla decifratura di un tag.
- Tramite il cellulare vengono ottenute informazioni riguardanti i servizi principali, quali ristorazione, pernottamento, locali, oltre a offrire la possibilità di vedere quali siano i musei, o i luoghi da visitare.
- Happy Tourist si pone quindi come una guida turistica interattiva, fornendo informazioni di prima utilità, consentendo all'utente di acquisire conoscenze sulla città e al tempo stesso stimolarne la curiosità partecipando ad un gioco domanda/risposta virtuali, il cui intento è anche quello di tracciare un percorso guidato alla visita della località turistica.

Happy tourist

- Quando il visitatore giunge nella città (in corrispondenza di aeroporti, stazioni, porti,..) è prevista la presenza di un access-point che consente all'utente di scaricare via OTA il file Jad dell'applicativo nella propria lingua e la successiva installazione del software.
- Sarà compito degli enti locali posizionare i tag all'interno della città, e prevedere una copertura wireless delle aree di interesse, in modo da non far sopportare al turista costi aggiuntivi per la connessione.
- Si prevede inoltre l'utilizzo di dispositivi "intelligenti", in grado di eseguire in automatico la commutazione fra la rete 802.11 e la rete GSM (alcuni apparecchi di questo tipo sono già in commercio).
- Per poter usufruire del programma, occorre compilare un form d'iscrizione specificando i dati personali, comprensivi della mail, che potrà essere utilizzata per l'invio di gadget o materiale informativo richiesto dall'utente.

Happy tourist

- Happy Tourist consente di:
  - Visualizzare la posizione dell'utente sulla mappa della città. Tale posizione è aggiornata all'ultima foto di un tag
  - Elencare hotel/ristoranti/musei/pub e ottenerne una breve descrizione
  - Fotografare un tag e fare il download di contenuti multimediali (quali file audio, foto) relativi all'oggetto a cui il tag appartiene
  - Partecipare ad un gioco rispondendo a domande di cultura generale, e sulla base del punteggio totalizzato, ricevere dei premi (brochure, news sulla città, gadget, ingressi ridotti in musei o altri locali)
  - Ottenere informazioni sulla presenza di mezzi pubblici nelle vicinanze
  - Memorizzare nella sezione "Preferiti" qualunque locale di particolare interesse, per poterne recuperare le informazioni (posizione, descrizione,..) senza doversi riconnettere al server
  - Prenotare un ristorante attraverso un meccanismo di request-response. Tale prenotazione si realizza in due fasi: una prima volta l'utente invia la sua richiesta al server, successivamente riceverà la relativa conferma. Questo meccanismo fa uso del Push Registry.
- Prevede delle servlet per la realizzazione del servizio richiesto oltre a effettuare l'interrogazionedi una base di dati

Ferormone

- Ferormone di Luca Frosini e Luca Galassi
- Permette l'invio di un messaggio di testo tramite bluetooth basando il protocollo di consegna sulla tecnica dei ferormoni utilizzato nel mondo animale.
- La consegna del messaggio, oltre che in modo diretto (mittente-destinatario), avviene tramite una terza persona che incontra frequentemente il destinatario.
- A tal fine ogni dispositivo ha una propria lista di cosiddetti device amici, cioè device incontrati frequentemente.
- E' disponibile una piccola rubrica dove conservare i propri contatti, in particolare per poter selezionare il bluetooth address del destinatario.
- E' necessario che i dispositivi dispongano della piattaforma J2ME CLDC 1.1 e MIDP 2.0 con estensioni API for Bluetooth Wireless Tecnology (JSR 82).

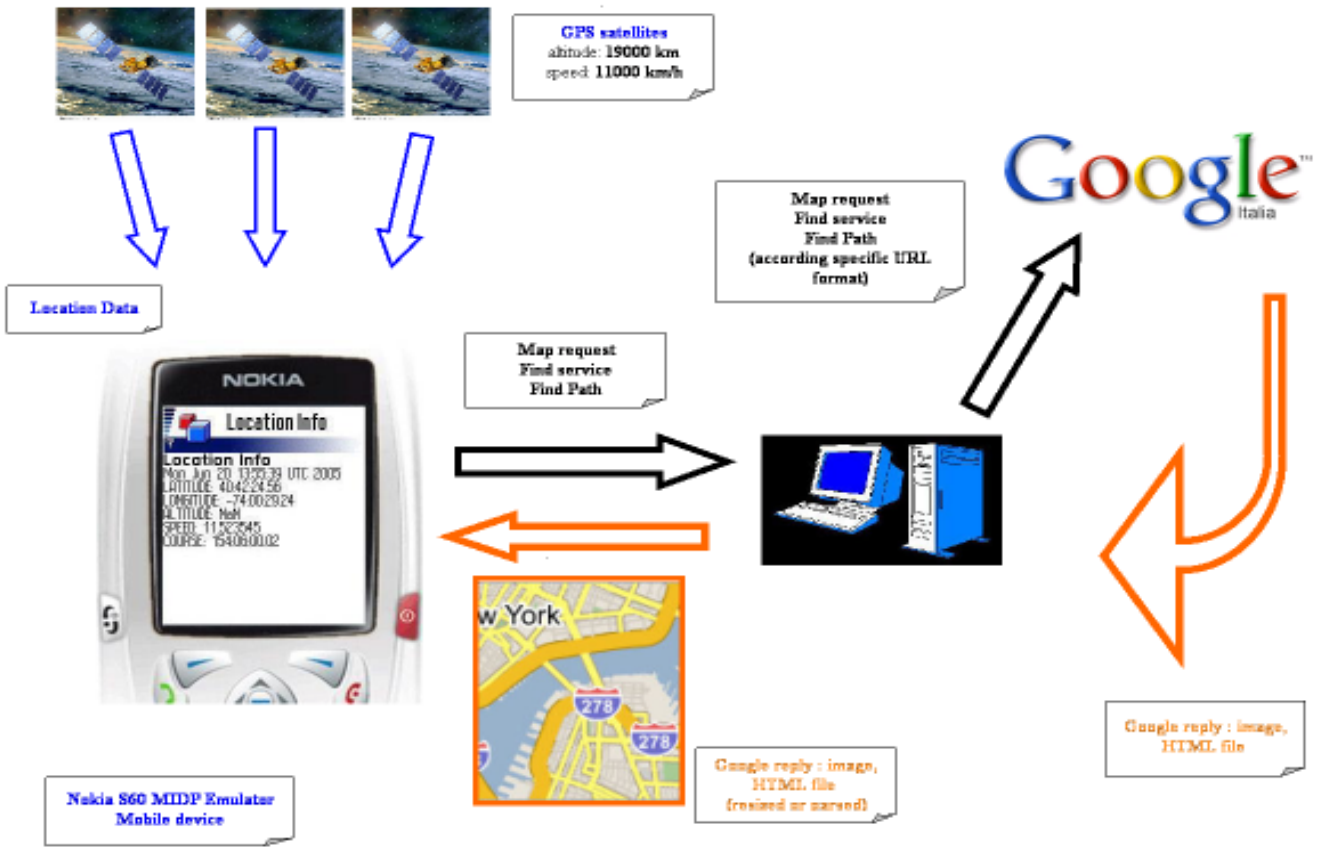
Mobile Google Maps

- Mobile Google Maps di AndreaPacini
- Il sistema consente di utilizzare alcuni dei servizi forniti dalle mappe di Google su un dispositivo mobile. Per il funzionamento complessivo e' necessario che il dispositivo mobile supporti le Location API (JSR 179) e sia dotato di un meccanismo di posizionamento geografico, quale ad esempio un ricevitore GPS (Global Positioning System).
- Le funzionalita' messe a dispozione da Mobile Google Maps sono:
  - visualizzazione delle proprie coordinate geografiche (latitudine, longitudine e altezza), recupero e visualizzazione della mappa corrispondente dal servizio google maps
  - navigazione manuale sulle mappe adiacenti alla posizione corrente (con funzioni di zoom in, zoom out e visualizzazione satellitare)
  - ricerca di servizi in base alla posizione geografica (es. "ristoranti a New York", oppure "negozi di abbigliamento a Chicago")
  - ricerca del percorso tra due localita' distinte (es. "da Seattle a Vancouver" , oppure "da Brooklyn Bridge , New York a Broadway 22, New York")

Mobile Google Maps

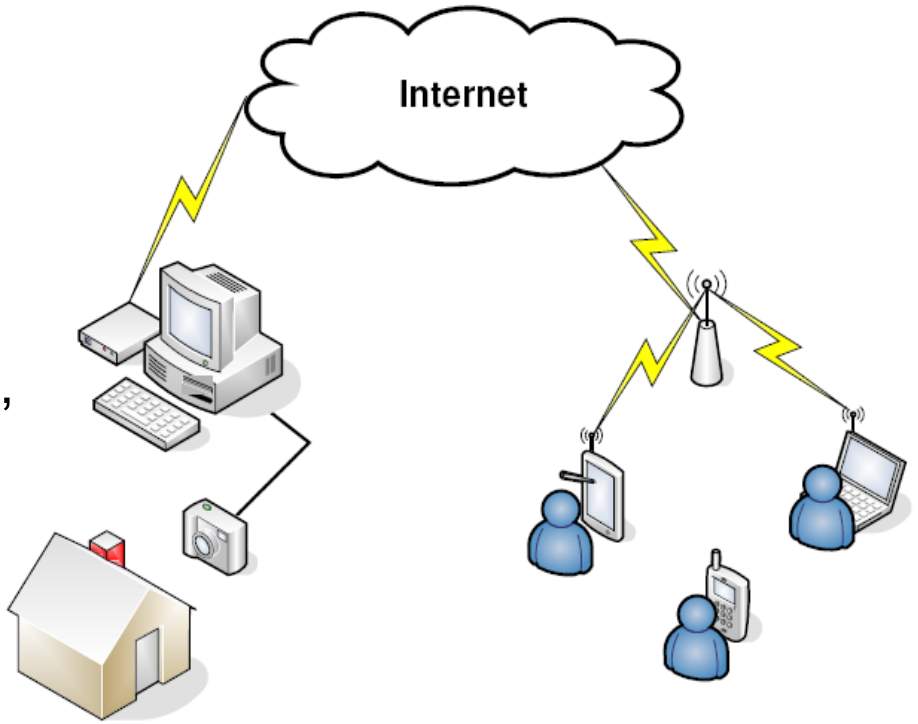
- Il programma e' diviso in due parti:
  - la midlet che recupera le informazioni di localizzazione dal sistema di posizionamento e , attraverso un interfaccia utente, consente di accedere ai servizi sopra citati.
  - una servlet che riceve le informazioni sulla posizione del dispositivo o le stringhe di ricerca e le invia opportunamente formattate al server di Google.
- Le mappe inviate da Google vengono ridimensionate dalla servlet in modo da adattarsi alla midlet; analogamente la servlet fa un parsing delle risposte di Google per le ricerche estraendo le informazioni significative che vengono rinviate alla midlet. Il sistema e' stato eseguito utilizzando l' emulatore Nokia Prototype 2.0 S60 MIDP Emulator esso possiede un emulatore interno di ricevitore GPS che legge i dati GPS nel formato NMEA da un file dell' emulatore stesso.

Mobile Google Maps



Panoptes

- Panoptes è un sistema di videosorveglianza che permette il controllo remoto di abitazioni, uffici, corridoi, piazze, etc.
- Il sistema si basa su tre componenti principali:
  - la componente di acquisizione video, composta a sua volta da un insieme (la cui dimensione è definibile dall'utente) di telecamere, webcam e camere IP;
  - una componente di elaborazione delle immagini e di archiviazione;
  - una componente di accesso alle immagini in archivio e "live".



Panoptes

- Ogni periferica di acquisizione video è gestita da un processo di management (Java) indipendente che si interfaccia da un lato verso la periferica di acquisizione video, dall'altro verso il sottosistema di archiviazione delle immagini. L'interfaccia lato periferica è realizzata mediante libreria di I/O (JMF, Java Media Framework) mentre l'interfaccia lato archiviazione è realizzata in Java-RMI.
- Le funzionalità del sottosistema di acquisizione video sono:
  - supporto di webcam locali (via JMF), webcam IP (mediante protocollo HTTP) o sistemi di acquisizione proprietari (via FileSystem);
  - indipendenza del sistema di acquisione dai dispositivi gestiti (il dispositivo deve essere riconosciuto dal sistema operativo e la periferica deve essere registrata presso il JMF-manager (Video4Linux o VideoForWindows));
  - algoritmo di motion detection in grado di rilevare cambiamenti nell'immagine acquisita e di informare in tempo reale l'utente (via email o MMS) dell'avvenuto cambiamento inviando copia dell'immagine catturata (allegata all'email o nel corpo del MMS);
  - algoritmo distribuito di fault control che monitorizza la costante presenza di tutte le sorgenti di acquisizione; in caso di anomalie si informa l'utente (via email o MMS) fornendo copia dell'ultima immagine disponibile per il device irraggiungibile.

Panoptes

- La componente di archiviazione ed elaborazione delle immagini ottiene in ingresso i frame catturati dai singoli sottosistemi di acquisizione, le archivia ed esporta un'interfaccia uniforme (per teminali fissi e mobili) di consultazione (rende disponibili anche le immagini prodotte in tempo reale). Questa componente fornisce inoltre i gateway per la notifica di eventi verso l'utente (via MMS o email) e gestisce l'algoritmo distribuito di fault control.
- Le funzionalità messe a disposizione sono:
  - archiviazione delle immagini prodotte basata su MySQL via JDBC;
  - interfacciamento per la notifica di eventi verso server SMTP (notifica per email) o centro servizi MMS (notifica MMS);
  - gestione delle liste di dispositivi funzionanti (basata su echo-reply UDP) e notifica di anomalie;
  - accesso alle immagini memorizzate sul server SQL via Java-Servlet e protocollo HTTP.

Panoptes

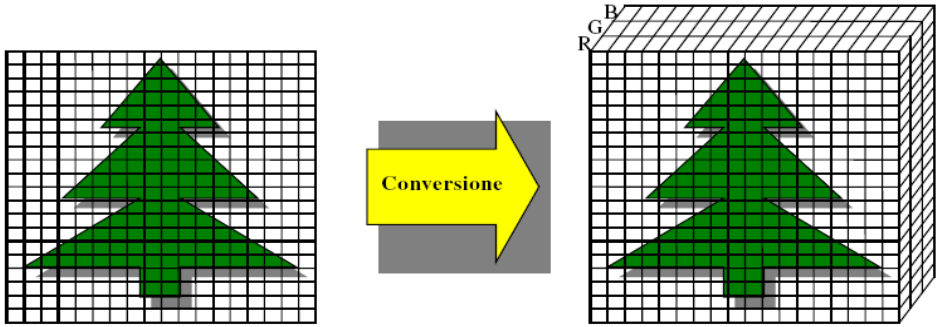
- La componente di accesso permette all'utente di consultare le immagini memorizzate.
- L'accesso alle immagini memorizzate avviene via protocollo HTTP.
- Sono forniti due diversi applicativi di accesso: uno per dispositivi mobili basato su J2ME e uno per dispositivi fissi costituito da un'applet da visualizzare in un web browser.
- Dai dispositivi l'utente può:
  - autenticarsi al sistema di accesso;
  - ottenere l'elenco delle periferiche di acquisizione video;
  - consultare la history delle immagini memorizzate in Panoptes;
  - consultare l'ultima immagine prodotta da un device, aggiornando l'immagine visualizzata impostando un intervallo di refresh.

Panoptes

- Il sistema di motion detection implementato è una versione semplificata e rivisitata di un ben noto algoritmo in letteratura e utilizzato in molti applicativi commerciali e opensource.
- I dati in ingresso all'algoritmo sono :
  - un numero n di immagini, n≥2
  - un valore che esprime la sensibilità ai movimenti dell'algoritmo
- Il dato in uscita dall'algoritmo è un valore di tipo booleano che indica se nell'analisi dell'immagine è stato riscontrato movimento.

Panoptes

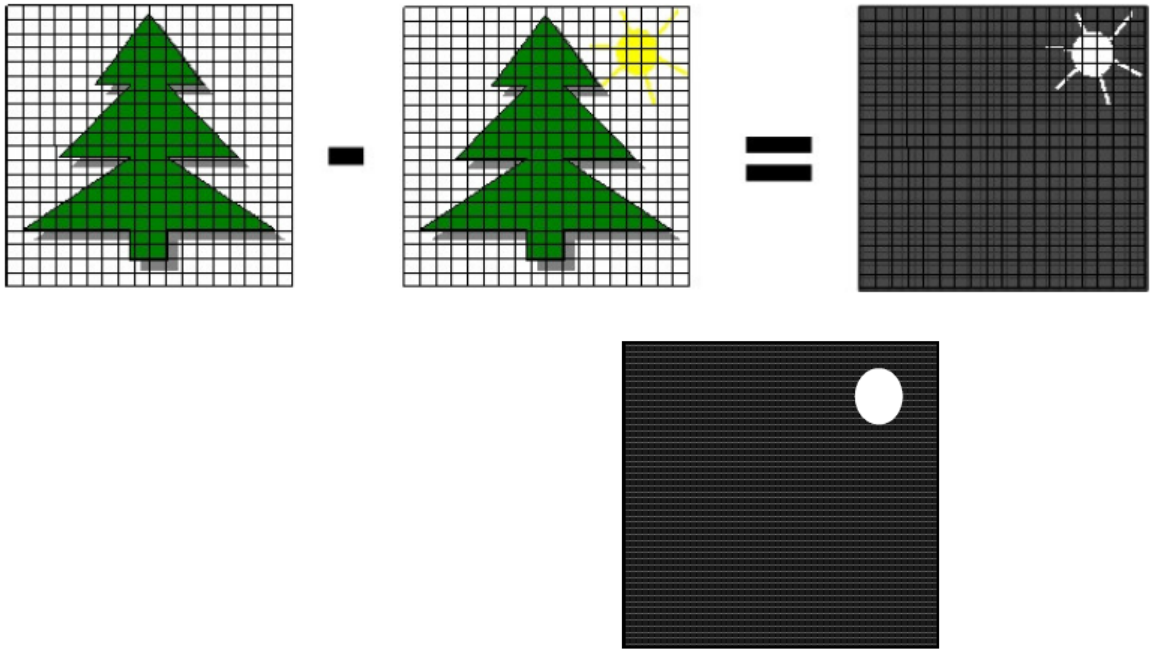
- Prese due immagini (che si suppongono avere stessa risoluzione e formato), l'algoritmo esegue i seguenti passi:
  - le immagini vengono convertite in RGB e i valori memorizzati in tre matrici;



  - Le matrici ver ento. Una nuova matrice di iden emorizzazione del risultato. Se due colori distano (sottrazione) per meno di un valore soglia, il risultato è allora pari a 0x00, altrimenti il risultato è pari a 0xFF.
  - La matrice risultato viene convertita in un'immagine in scala di grigio: le aree scure sono parti dell'immagine per le quali non sono state riscontrate differenze, le aree chiare sono parti dell'immagine per le quali si sono identificati cambiamenti.

Panoptes

- I punti della matrice vengono aggregati a formare aree più grandi: se un pixel chiaro è confinante con un altro pixel chiaro, i due pixel vengono aggregati a formare un'area più grande. Se un pixel chiaro è "isolato" (non ha pixel chiari confinanti) questo pixel viene sostituito da un pixel scuro. Il risultato di questa operazione è un immagine in cui sono state rimosse le differenze "locali" (non influenti dal punto di vista dei cambiamenti globali nell'immagine).
- Le aree chiare delle immagini di dimensioni inferiori ad un valore di soglia sono rimosse e sostituite con colore scuro.



Panoptes

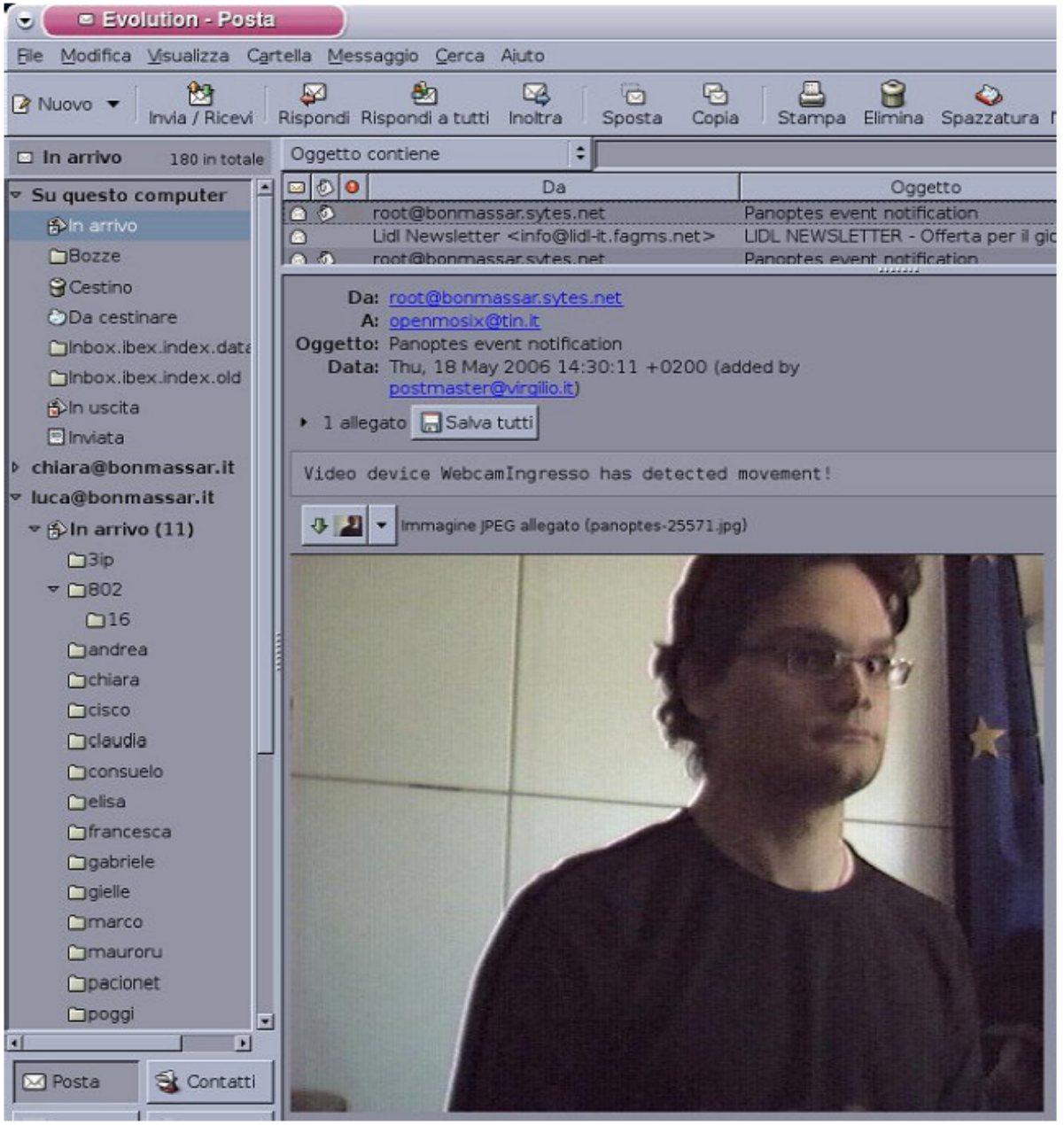
- Ogni nuova immagine prodotta dalla sorgente video viene confrontata con n immagini prodotte in precedenza dalla sorgente attraverso il precedente algoritmo:

```
while(i<=n){
        //Recupero dalla history di un frame da confrontare
        ImmagineConfronto = ArchivioImmagini[i];
        numeroPixelBianchi = eseguiAlgoritmo(ImmagineRiferimento, ImmagineConfronto);
        //Valutazione pesata sul tempo di creazione dell'immagine
        pixelBianchiTotali += (numeroPixelBianchi/((n-1)/i));
        i++;
}
```

- Questi valori sono tra loro aggregati pesandoli in rapporto al tempo di creazione delle immagini: il confronto tra frame temporalmente distanti avrà, nel risultato complessivo, un peso inferiore rispetto al confronto tra frame temporalmente adiacenti.

Panoptes

- Notifica di movimento
- Il sistema è in grado di avvertire l'utente attraverso posta elettronica o MMS.

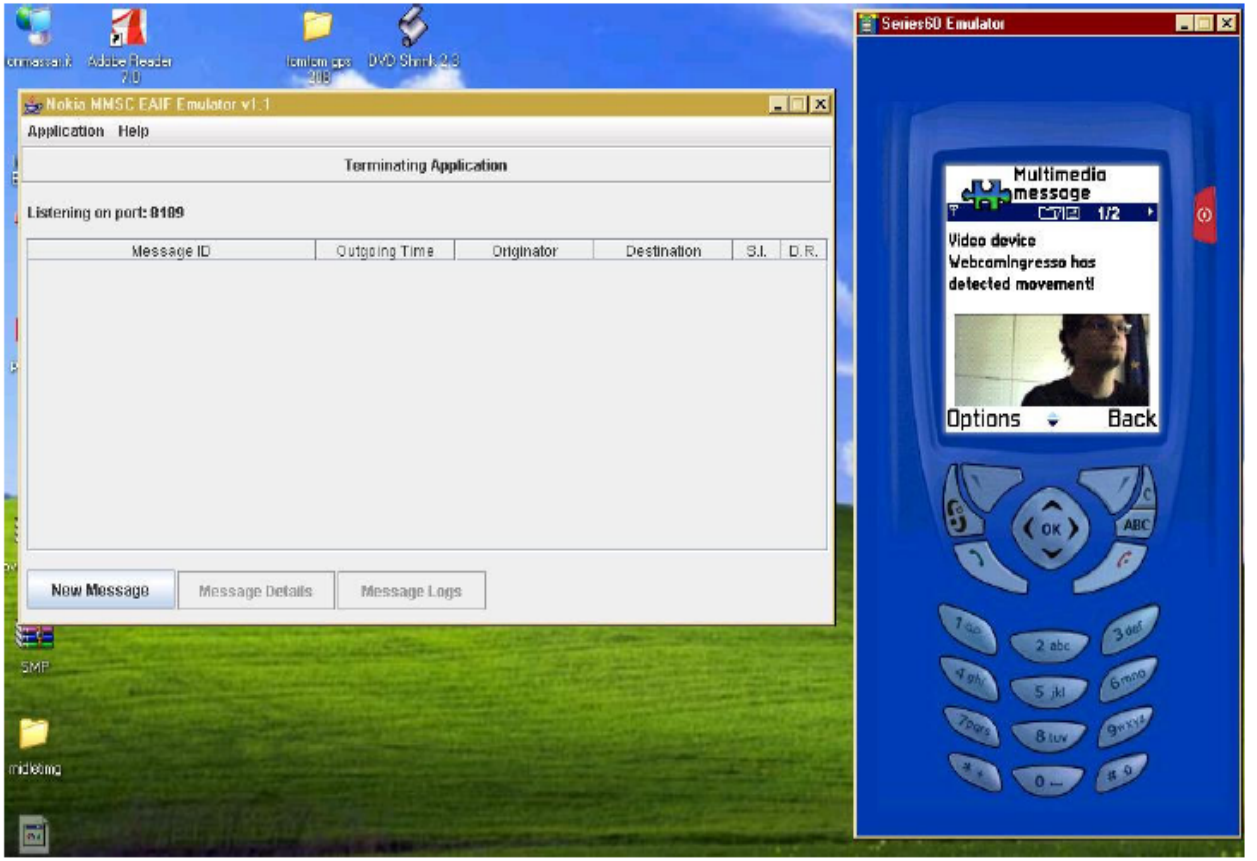


Panoptes

- Per la notifica di eventi mediante MMS è stato usato l'emulatore di centro servizi MMS Nokia e dell'emulatore di terminale N60.

- Una delle sorgenti video ha individuato movimento nelle proprie immagini e ha segnalato l'evento al collettore delle immagini.

- Il collettore delle immagini ha quindi inviato un messaggio di notifica via MMS all'emulatore del centro servizi, il quale a sua volta ha reinstradato il messaggio MMS verso il terminale.



Panoptes

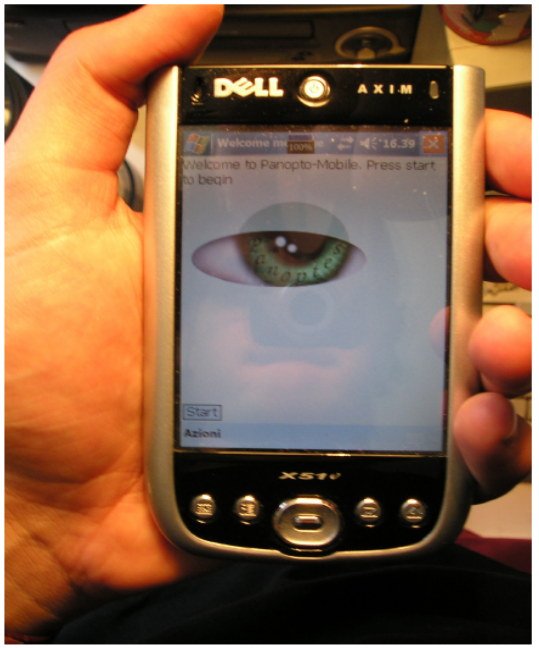
- L'utente può accedere alle immagini acquisite dalle telecamere attraverso un terminale mobile.
- La prima volta che il sistema viene usato è necessario introdurre username, password, e url del servizio.
- I dati vengono salvati in memoria persistente.

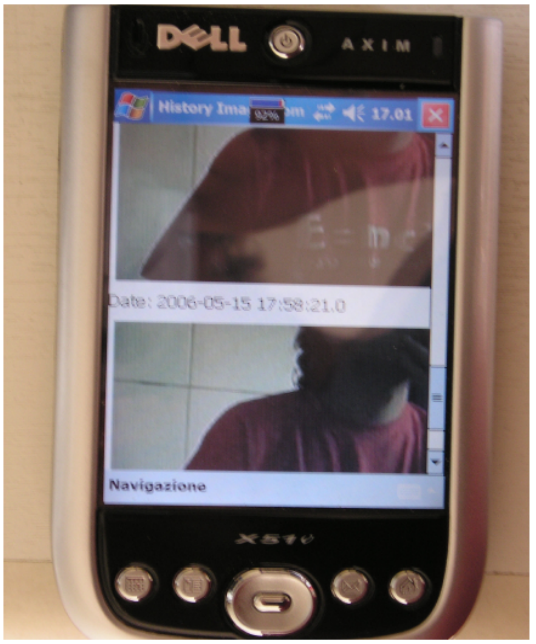
Panoptes

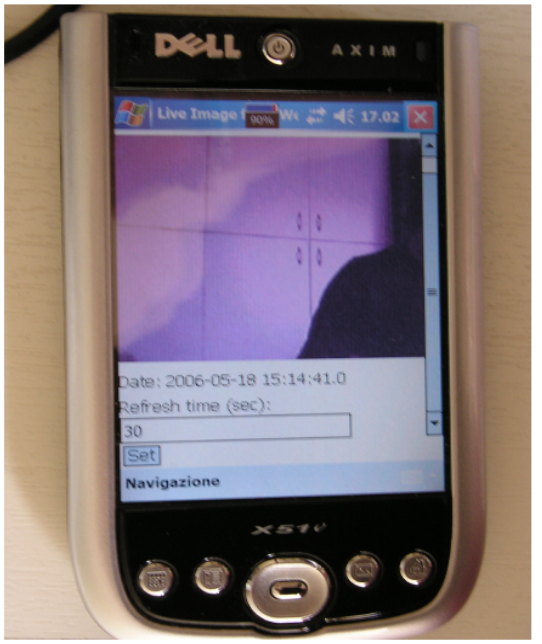
- Si può scegliere se visualizzare immagini “live” o archiviate.
- Nel caso di immagini “live”, l’applicazione richiede all’utente di selezionare una tra le sorgenti video disponibili.
- L’applicazione risponde mostrando a terminale l’immagine istantanea ricevuta da tale dispositivo. L’utente può selezionare un intervallo di refresh.

Panoptes

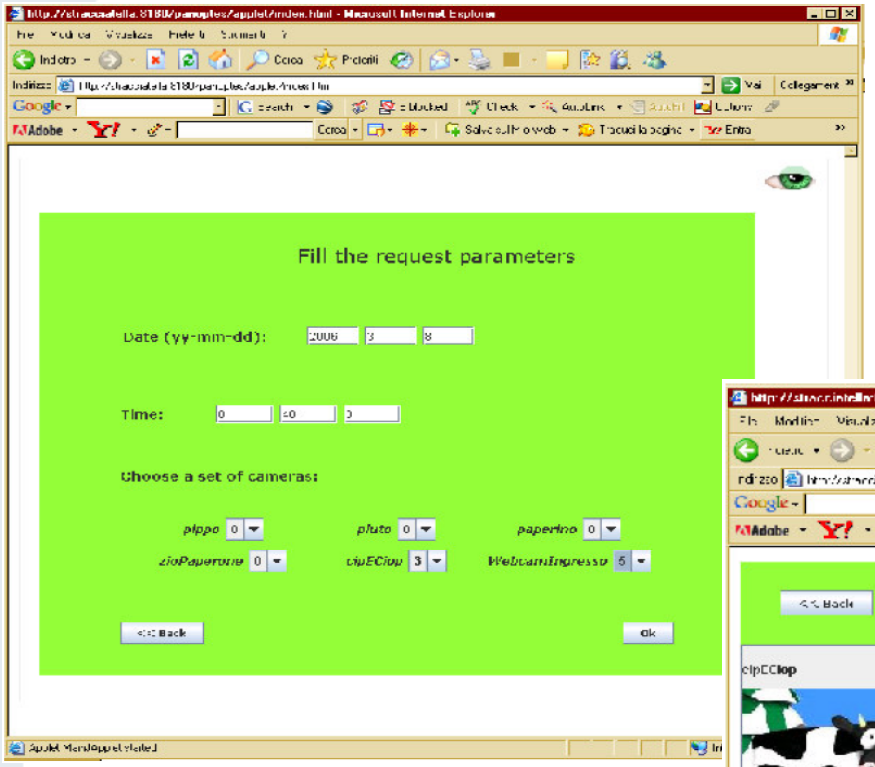
- Nel caso in cui si vogliano visualizzare le immagini archiviate è possibile selezionare la data, l'ora ed il numero di immagini.
- In questo caso in esecuzione su un dispositivo reale (PocketPC Dell Axim 51v, Windows Mobile 2003 e Ibm WebSphere MicroEnvironment 5.7.10).

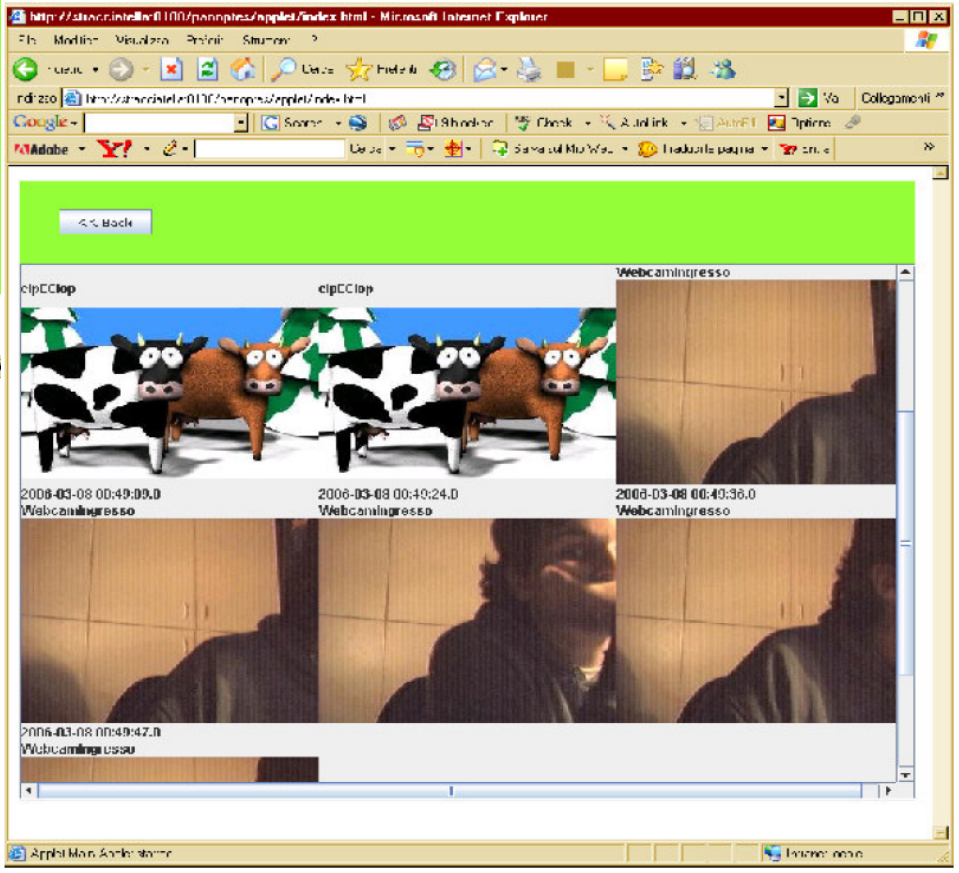






Panoptes





- Le immagini possono essere accedute anche attraverso un comune browser.