

Mobile Services Project
“Event Guide”
Report

Ramunas Laucius
Svetlana Voronkova

Introduction

“Event Guide” is a peer-to-peer application that allows users to create events, invite friends to them via sms and keep track of events that user has been invited to. The application also provides a possibility to view/search for events by date and category, or represent upcoming events on the map.

Main features of the “Event guide”:

- Create new event,
- Choose friends from phonebook list and send invitation to them via sms,
- Cancel created event ,
- View events according to different properties.

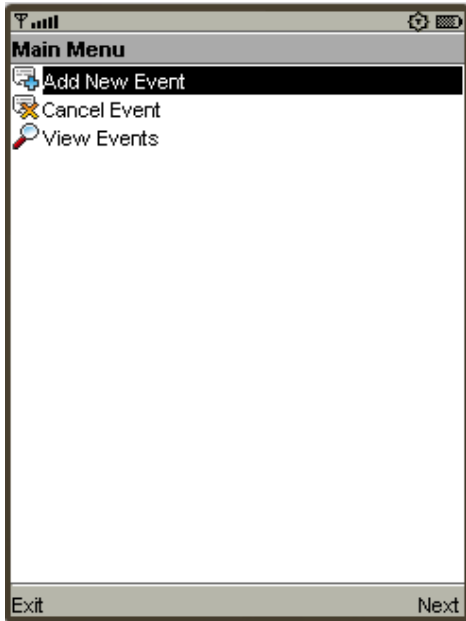
User can view events by following properties:

- Show all events,
- Show today’s events,
- Show tomorrow’s events,
- Show all events from category “party”,
- Show all events from category “cinema”,
- Show all events from category “birthday”,
- Show events that are in 10 km radius from user location.

Interactivity and Navigation

“Event Guide” is a usability-focused application, with user friendly interfaces and intuitive navigation. In order to test this, lets consider the following scenario: try to create a new event and send invitations to some of your friends. The first screen displays the main menu with the commands: Add new event, View events, Cancel event (Picture 1). After choosing “Add new event” the user is redirected to the new event insertion form (Picture 2). In this form user can specify name, address, date, time, and type of the event, there is also a possibility to add a short description. After pressing the button “Submit” the created event is automatically saved on local device storage and the user is asked if he wants to invite some friends. If user chooses not to invite any friends now, he is redirected to the main menu. In the other case the list of friends from phonebook is displayed, and user can choose whom to invite to an event (Picture 3). When the user presses button “Send”, sms with the invitation is sent to all selected friends and user is redirected to main menu.

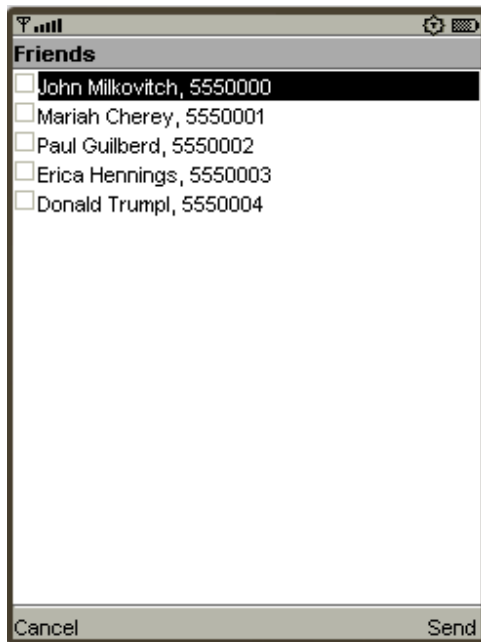
Note: after each background task is executed (e.g. sending messages or querying over the wireless network), user is notified by alert dialog, showing status of the task and asking user’s confirmation. This makes user aware of what services are accessed and what is their result, leaving no unknown information for the user.



Picture 1. Main menu



Picture 2. Event insertion form



Picture 3. Friends List



Picture 4. List of events viewing properties

As a second scenario let's try to view all events from category "Party". First of all user has to choose from main menu the command "View events", after he is redirected to the list of all

possible properties, by which event can be viewed (Picture 4). Let's choose the filtering by "Parties". The list of all party events now is displayed (Picture 5). Now after selecting one event and pressing "Next", that event is shown on the map with event location and event description displayed below the map (Picture 6). By pressing the button "Invite", user will be redirected to the friend list and will have a possibility to invite more friends to the viewed event (Picture 3).



Picture 5. List of all parties



Picture 6. Map view

Navigation between all screens is linked logically so that there are no "dead points" when the user ends up in a screen with no navigation out.

Network usage

The application is using two types of data-transmission networks: SMS's are sent using usual WMA service, and maps along with geocoding service uses Http connections over the net.

All WMA transactions are per-user basis, and are made in two situations – when inviting friends to a new (executed during transition between screen in Picture 3 and Picture 1) or notifying them about cancelled event (reached from screen in Picture 1, selecting "Cancel Event" and confirming selected event).

Http connections, on the other hand, are seamlessly integrated to the application and uses minimal bandwidth (except downloading image from Yahoo service). Http traffic is being generated when the event is saved to database by querying its latitude and longitude information from Yahoo geocoding API. The other Http access point is when selecting specific events from one of the filtered views (Picture 4). Images are not cached, thus generating some extra network

traffic per event view. If there is no connection to the Internet, events' geolocation and map view are ignored and not shown, displaying only textual information from local storage.

The Structure of the Code

The application uses the following API's:

- Personal Information Management (PIM) , for retrieving contact information from device's memory. It is used for read-only purpose only and thus is no security threat whatsoever.
- Location API, for detecting nearest events within specific radius (implemented as 10km).
- Yahoo maps API (external, link: <http://developer.yahoo.com/maps/>, namely, Rest API and Geocoding service), for displaying and marking event of interest on the map.
- KXml (external, link: <http://kxml.sourceforge.net/kxml2/>), for parsing XML messages used in maps API.

The graphics for application's UI are provided from Deviant Art website (<http://deviantart.com>).

All application code lies subdivided in two packages – the main package (“hello”) where we have Midlet as a controller for UI elements, and the utility package (“hello.helper”) containing extra functionality which involves repetitive tasks, like constructing Alert components, querying remote Http services on Yahoo, etc. The following list briefly describes main components of our application (also refer to Picture 7 for UML class diagram of this application):

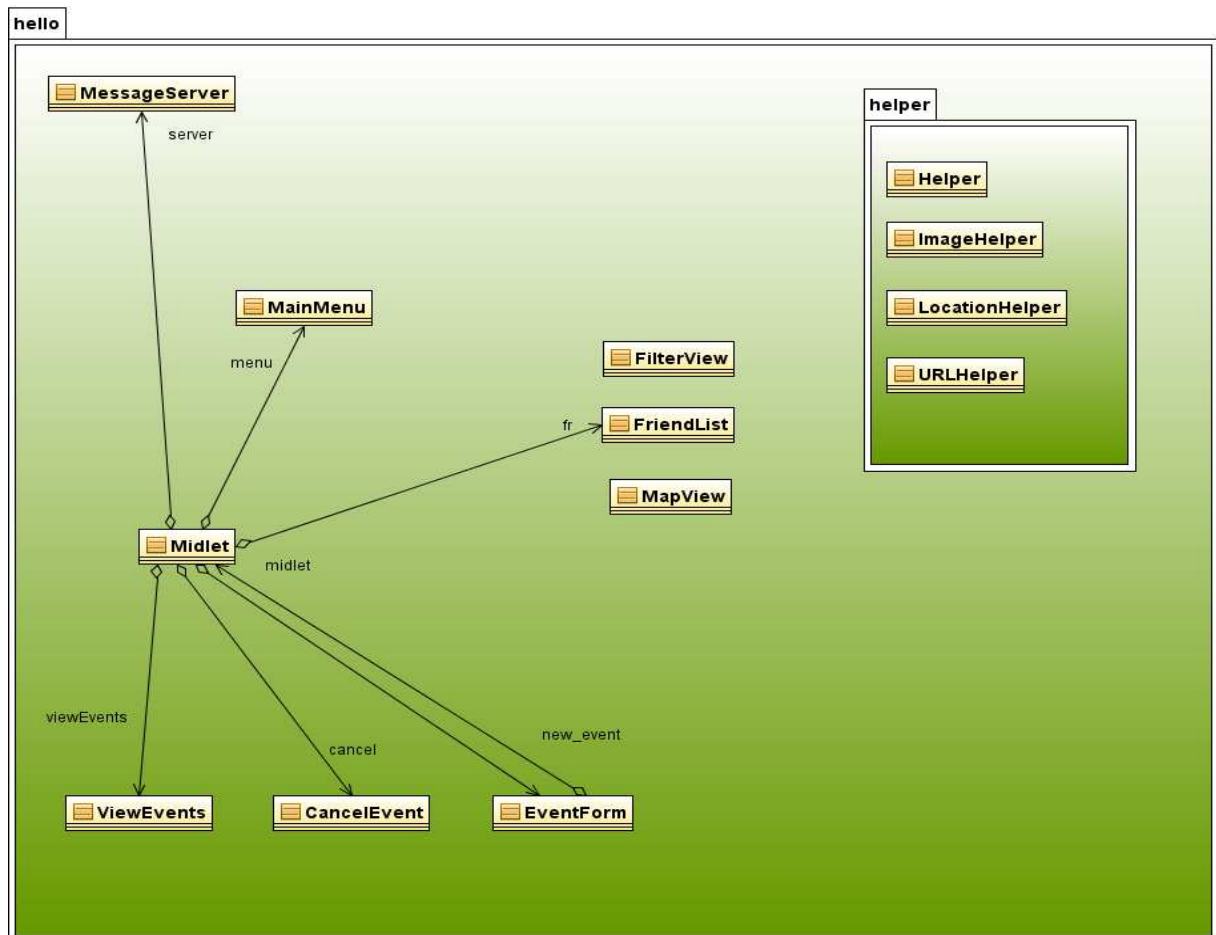
hello package

- Midlet – main entry point to our application since it extends MIDlet base class. It acts as a controller responsible for navigation between screens and actions executed in between transition. Most of helper classes from *hello.helper* package are used in Midlet's listener methods.
- MessageServer – server which listens for incoming SMS messages from your friends. It is a handler for received SMS's when we launch application deployed using OTA.
- MainMenu, EventForm, ViewEvents, CancelEvent, FilterView, MapView – primary and secondary menus and lists. Those are basically lcdui components extending either Form or List classes from J2ME SDK. As stated, transitions between these elements are controlled in Midlet class.
- FriendList – the component using PIM API to extract contact info from phonebook. It is used in event creation/editing chain of screens.

hello.helper package

- Helper – most basic helper for creating Alert controls from J2ME SDK.

- LocationHelper – retrieves information about latitude/longitude of the event using Geocoding API, also parses XML from Yahoo services.
- ImageHelper – retrieves URL for map image using Yahoo maps API, also parses XML retrieved from Yahoo services.
- URLHelper – reads byte content of specified URL.



Picture 7: UML class diagram

The Major Technical Problems and Recommendations

The major technical problems that we faced during the development of the project were:

- Difficulty with sending the SMS message that could run the application on other cell phone when receiving it (involving deployment through OTA).
- We had also some problems with emulator:

- Friends Contact list had to be re-populated every time before running an application;
- The records are deleted from the Record Store after re-running an application, making it difficult and time consuming to test.
- Technical Problems with displaying the maps:
 - Google does not have an API for J2ME, the only possible way (as far as we know) to get Google maps running is by having intermediate server which parses Javascript and provides plain images available through Http to our mobile application, which is far beyond the scope of this project;
 - Yahoo maps – to display one event on map two requests are needed. As the main limitation we can describe that it's not possible to display more than one event marker on the map;

By summarizing the difficulties mentioned above, we conclude with several recommendations which could make this application more efficient:

- Caching map images on local storage if it is possible, since network bandwidth is (usually) paid.
- Intermediate server component would provide way more flexibility to this type of social application, e.g. letting every user to know who is attending specific event, which is very costly to do with usual SMS messaging. Another use would be of more interactive maps, e.g. Google provided ones.
- Using non-blocking sockets would provide more efficient messaging communications.
- Testing application on multiple vendor phones, since emulator provides only a single running perspective (only few options available, which make nearly no difference).