Around me

Internet and Mobile Services Project

Philipp Hofer

Patrizia Gufler

Index

1) Introduction

The “Around Me” project was realized during the course of Internet and Mobile Services taught by prof. Ricci. “Around me” is an application for mobile devices supporting JavaME and equipped with a GPS sensor, which provides to the user a list of elements which he is interested in. The user gets easily information about restaurants, hotels, events etc. on a specified location. The information is retrieved from a web service provided by prof. Dario Cavada working on the http://www.visittrentino.it/ website. Preferred elements can be stored locally on the device, if desired also inserted in the local calendar *(for future work).*

2) Analysis

Mobile User Analysis

WHY using this app?

This app reveals to the user accurate and precise information he may be interested into in a fast and reliable way. So it is usable for exploring not well known locations or to become more acquainted with familiar places.

WHO uses the app?

This app is valuable for everybody that wants to discover quickly different points of interest. Is useful for families and young people to plan a free day; also business people can take advantage by searching an adequate restaurant or hotel for his working purposes. The major constraint is that the user cannot efficiently use the application when he is distracted. This means, a user that is occupied with another task, for example driving a car, cannot concentrate on the application. So, he cannot find the element which best fits to him.

WHERE is this app used?

Our application works best when the user can concentrate fully on navigating through the information adjusted at a time in a specific location. Therefore, it is convenient to have this mobile application, so the desired information is accessible not only from home but also in public places (train, office).

WHEN is this app used?

The intended moment to use this application is when the user is online, receives a GPS signal and has time to go precisely through the suggested elements of interest. So it is important that the user is not disturbed by something else, otherwise he is not able to choose the elements best fitted to his needs. Regardless of the season, week, day, time and location the user gets (customized)suggestions (*future work*) from the application for element (for example: hotel) selected. The app inspires the user during free and working time. In recommending it considers the time period of the day respectively weekday and weekend.
A user who drives a car has to be concentrated on the road and so he is not able to efficiently use the application.

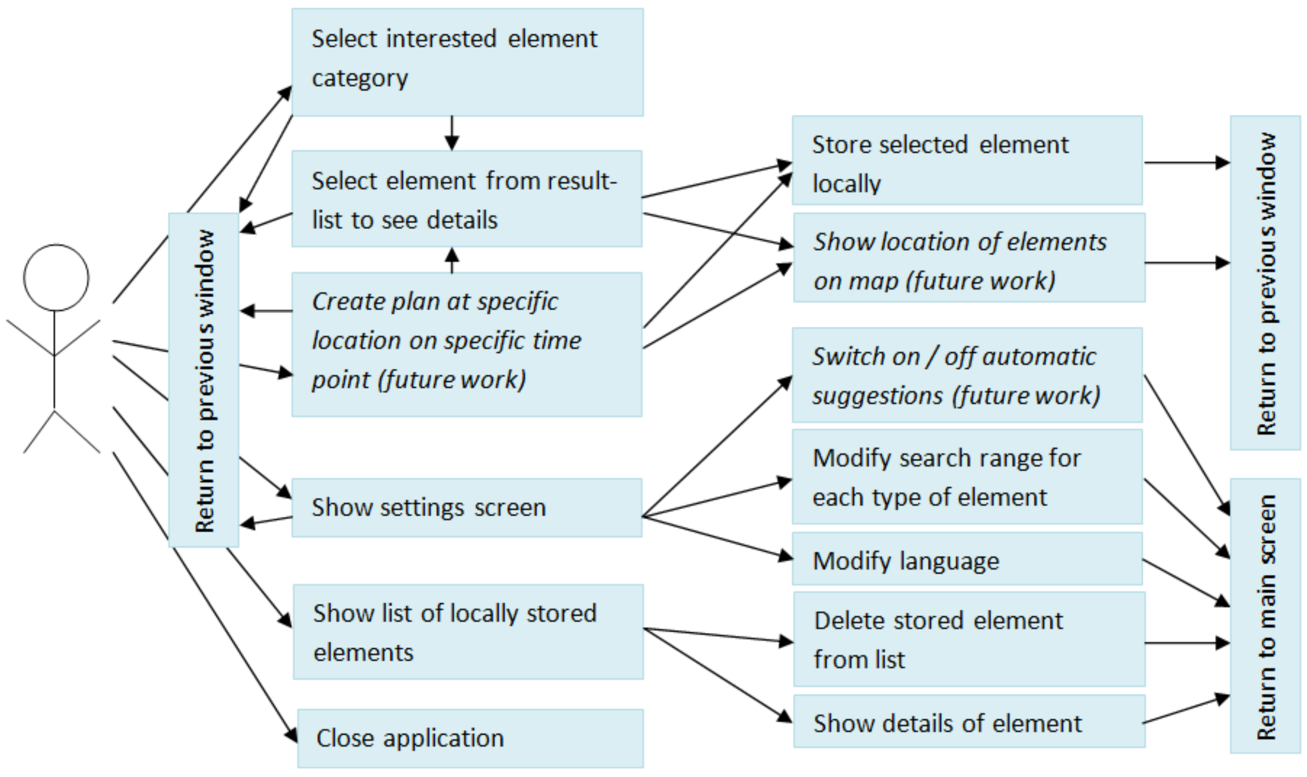
WHAT are the circumstances?

There are two main scenarios. In the first case the user plans (*future work*) to be in a specific location at some time point. In advance he wants to look at interested elements to get a picture. In the latter case the user is searching currently for nearby interested elements. The user can modify the size of the area which is considered for the search. During the search process the weather is taken into account.

Classification of this app

- Focused on: Informative context
- Side issue: Local context

Scenario Analysis



Select interested element category
Select element from result-list to see details
Create plan at specific location on specific time point (future work)
Show settings screen
Show list of locally stored elements
Close application
Return to previous window
Store selected element locally
Show location of elements on map (future work)
Switch on / off automatic suggestions (future work)
Modify search range for each type of element
Modify language
Delete stored element from list
Show details of element
Return to previous window
Return to main screen

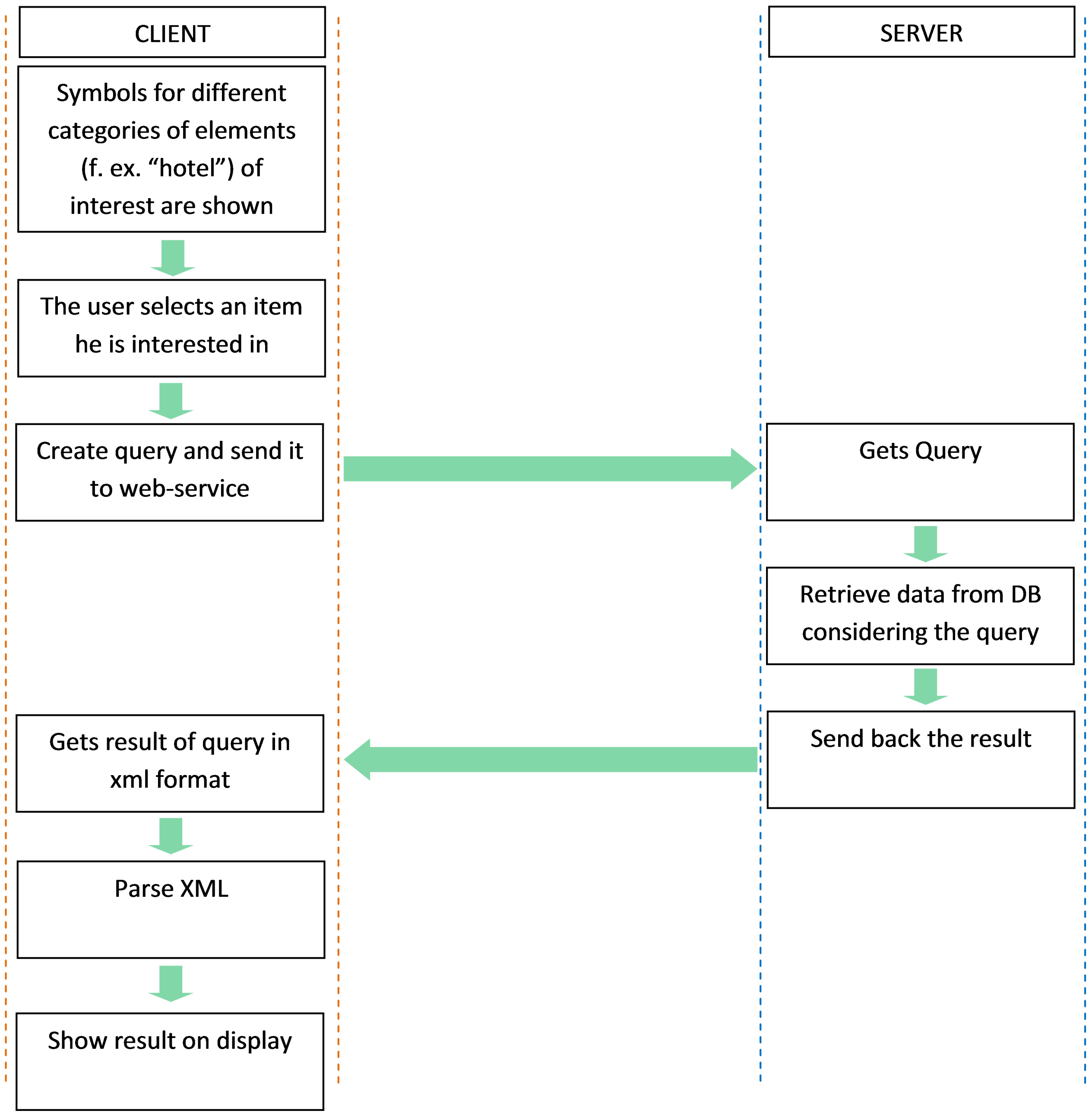
Usage Analysis

Screen and Interaction Analysis

Firstly, when the user runs the application the main screen with five different icons is shown. After selecting one item (for example “hotel”) a list of matching elements containing the most important information is displayed (ordered according to their distance). At this point the user can click at one item to get details about it. If the user wants to keep this information locally (/in offline mode), he must click the “Store” button.
In the main screen the user can open the menu where he can choose between:

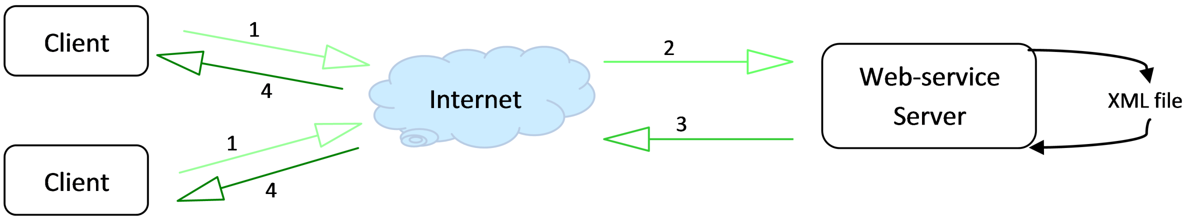
- *Planning (future work)*
    - *Set a timepoint*
    - *Insert a location (so not the current location is used in the search process)*
- Settings
    - Set the size of the area considered in the search process
    - Set the maximum elements displayed for each element type
    - Set the language of the application
    - *Activate / Deactivate personal suggestions (for free time slots)(future work)*
- Open stored elements

Environment Analysis


CLIENT
Symbols for different categories of elements (f. ex. “hotel”) of interest are shown
The user selects an item he is interested in
Create query and send it to web-service
Gets result of query in xml format
Parse XML
Show result on display
SERVER
Gets Query
Retrieve data from DB considering the query
Send back the result

3) Architectural Design

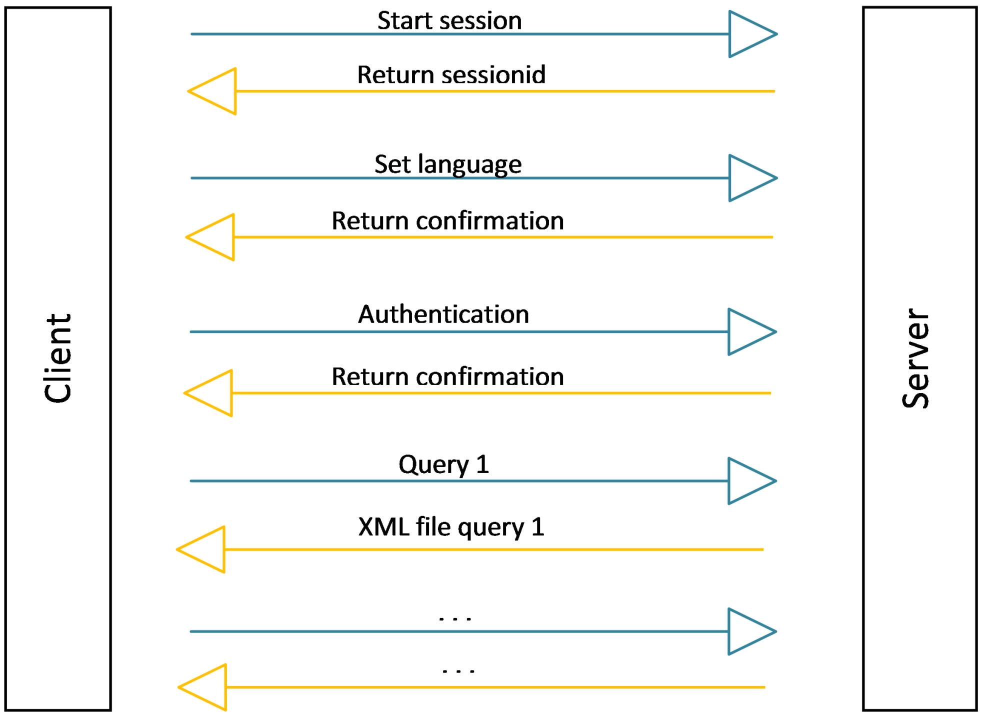
General information flow



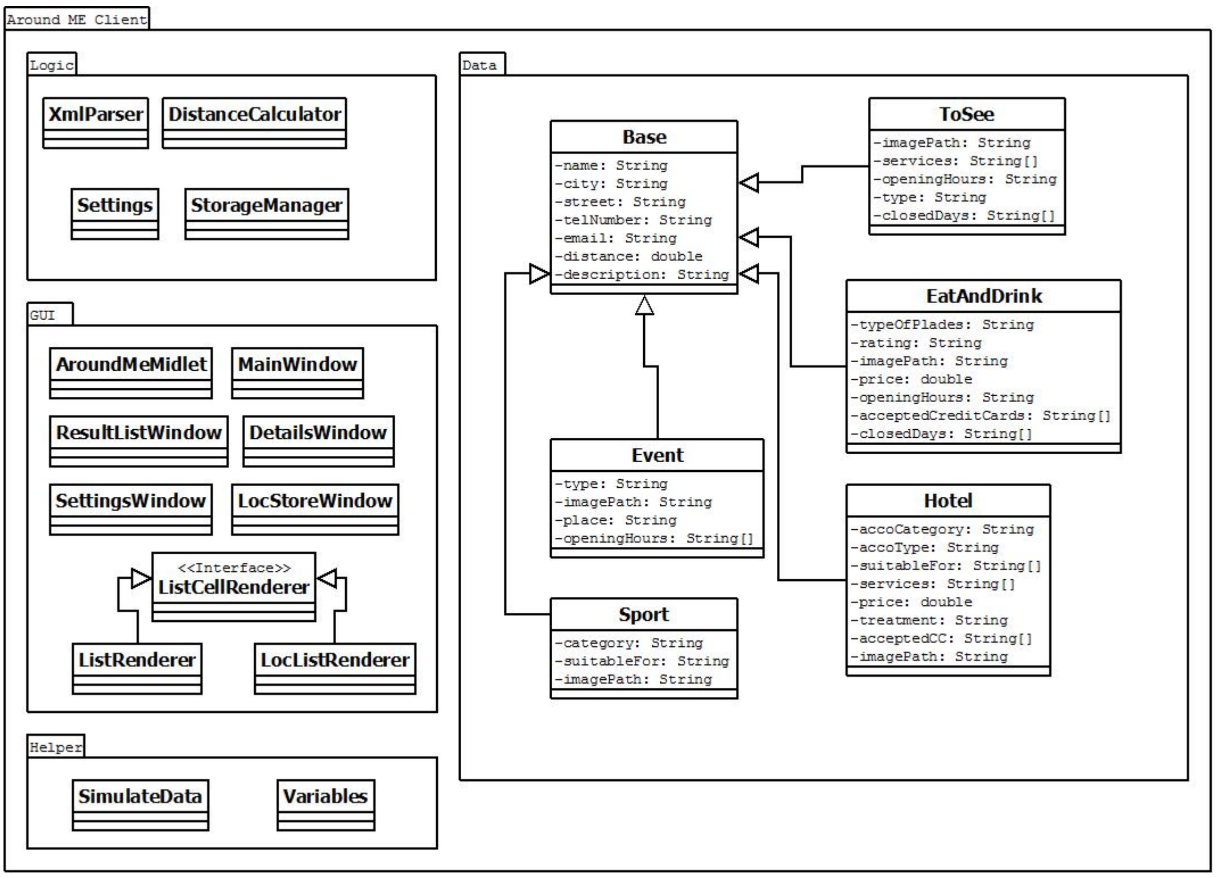
1. Client establishes an HttpConnection and sends a query-request
2. Server receives the query-request and computes an xml file with the resultset
3. Server sends the computed xml file back
4. Client receives the xml file, extracts the data and displays it

Client-Server communication

The user runs the client on his mobile device. The application firstly establishes an Http-Connection to the server by logging on and then it can submit query-requests. The result of the query is then sent back to the client and displayed. For user-friendly reasons the client sends at each query also the location-range (current or of *future plan*), so the user gets only interested information (in xml format -> parsed with kxml) respectively to the location. For this reason the MIDP location API is needed. Optionally we want to provide the user the possibility to see the location of the elements contained in the result query on a map (Google Map API -> *future work*).



Rough class diagram



Benefits

- Well defined data flow between client and server
- We designed our architecture using the MVC pattern, so we can better focus on the model performance
- The offline user cannot create any request to the web service. Neither a request is stored temporarily for the next time he comes online, because in most cases it is already out of date and therefore useless. Additionally, only with one click the user can make a new request.
- Each requested XML file is temporarily stored, only if the user wants detailed information about an element, the possible image of the element is loaded on demand. So only those images are that user really wants to see are loaded.
- For each query the returned resultset should contain at most x items. If there are more elements matching the query, the user can request them by clicking at the "More" button (*future work*)

4) Display / Screen

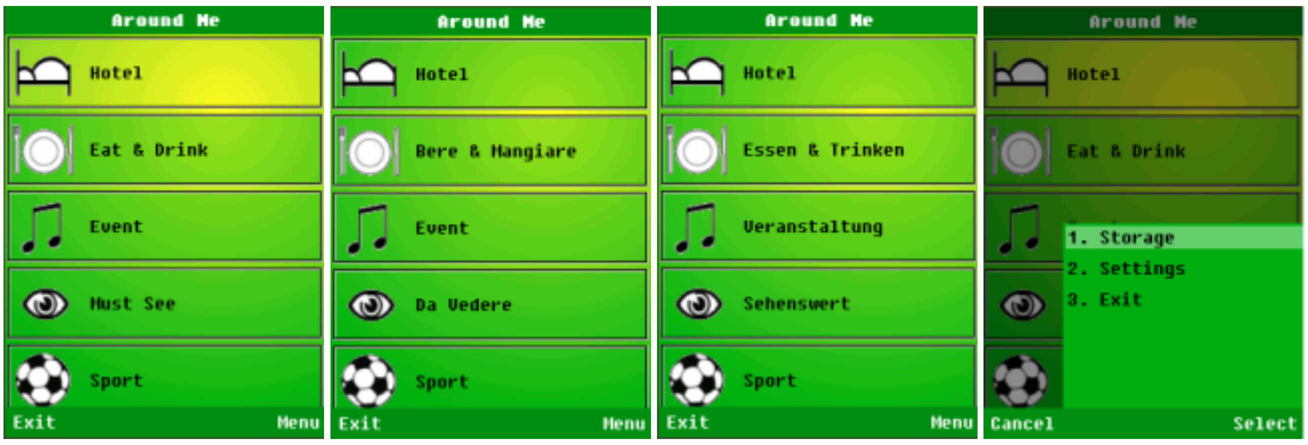
General

For our project the user does not have to insert much, so we decided that the minimal required resolution of the display must be 320x240 pixels. Therefore we use the "DefaultCldcMsaPhone2" emulator (display size 320x240px) during the development of our project. This phone is a normal cell phone with no touch screen. However, this app can also be used by mobile phones with touch screen.
The user can reach the detailed information about an element of interest within two button presses. For more specialized actions we provide a menu where he can plan future activities (*future work*), open "saved elements" or change the settings. On each user interface there exists a "back" button to return to the previous UI.

User face and interaction

Start up

As a first step before anything is displayed the application checks if in the local storage there exist two files, the settings-file and storage-file. In the settings file the previously stored settings done by the user are stored. If the file at the moment does not exist (can happen only if it was deleted or the application is started the first time) no action is performed and the default values of the parameters are loaded. The other file, the storage file, contains the items, previously stored by the user. To have a good overview over the stored items, the amount of elements was limited to ten elements. If the application is started the first time or if this file was deleted, no loading occurs and the construction of the main screen can finally start.



Around me supports serveral languages (eng, it, de)

Main screen

The main screen is the start-point for the user of this application and consists of five list items. Every list item represents a different search category. The user selects a category of interest by clicking on the corresponding button. If done so the user sees immediately the loading screen of the application. At this point a second thread is started, which establishes an internet connection and starts the login process to the web-service provided by "www.Visit-Trentino.it". The login process consists of three steps. Firstly a session ID is requested which is further used in the login process. Secondly the language of the query-result is determined. Finally a request containing the login data is send to the server. If in all three requests the web-service were successfully, the application sends a XML-query to the server which URL consists of information partially contained in the settings (location and max. displayed items) and partially influenced by the category (f. ex.

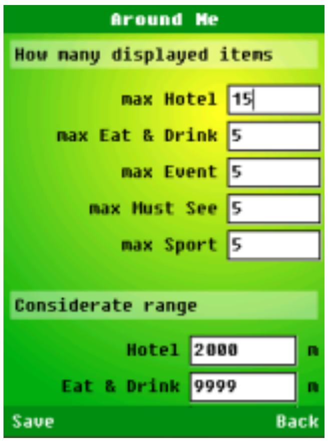
“hotel”). Otherwise, if the login fails, a second try to get a valid response from the server is performed. If another time the login to the web-service was not successful, the application displays to the user that there is a problem with the service.

After the successful login and the sending of the XML-query the application receives a response in XML format. This XML is parsed to extract all relevant data, then this data is used to construct the result-list shown to the user. However if in the XML response of the server there is no element (matching the query), a message is shown to the user which advises him to either increase the search range (can be done in the settings), because no element can be found with the actual parameters. One of the most important benefits of starting a second thread is that the user can abort at every time the establishment of the internet connection and also the parsing process. This can be done by simply lunching the back command displayed at the left corner of the application. Done so, the user returns immediately to the main screen of the application.

When the user makes a request to the web service, the loading screen is shown

Settings window

In this window the user can change different parameters, which influences, as just mentioned above, the creation of queries to get the matching items and also the amount of elements displayed in the result list. This amount is always added to the query send to the server, such that no more elements than specified in setting are downloaded. Another adjustment, which can be done by the user, is to change the language. The user can select between three languages (English, Italian and German). To store the modification of the settings permanently the user has to execute the save command. If the language was changed, the displayed strings and command-names are immediately adapted and the user can continue using the application with the desired language. All settings are stored immediately after the user exits the application and so kept for the future.



Around Me

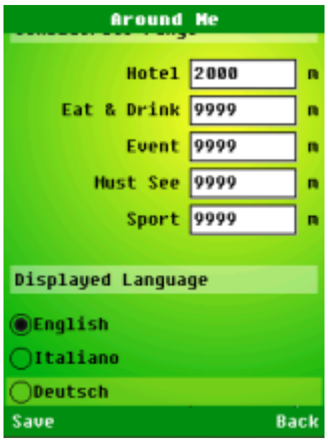
How many displayed items

max Hotel	15
max Eat & Drink	5
max Event	5
max Must See	5
max Sport	5

Considerate range

Hotel	2000	m
Eat & Drink	9999	m

Save Back



Around Me

Hotel	2000	m
Eat & Drink	9999	m
Event	9999	m
Must See	9999	m
Sport	9999	m

Displayed Language

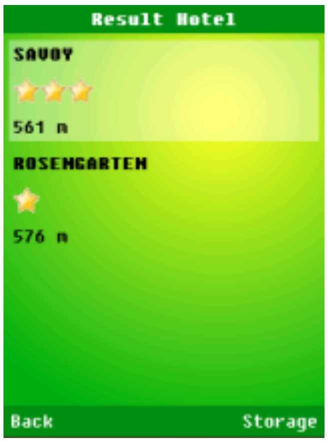
[✓ English]
Italiano
Deutsch

Save Back

In the settings window the actual language and the maximum elements for each category is specified

List of results

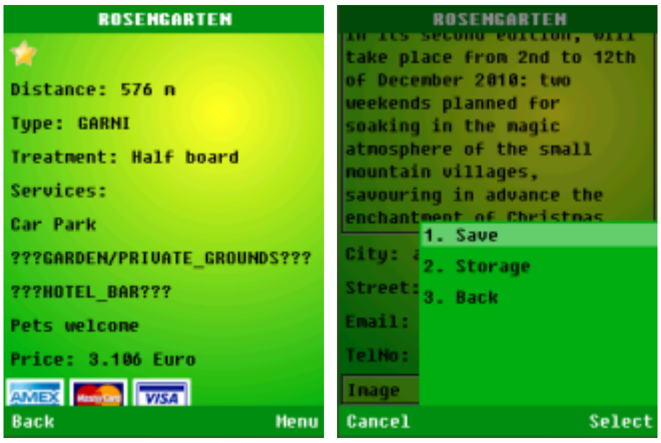
If the XML query was successfully sent to the web-service and the response contains at least one element, a list of the matching elements is displayed to the user. Now the user has the possibility to navigate through the list, which is ordered by the distance calculated (difference between actual GPS position of the user and the position of the corresponding item). Every item in the category consists of three lines in which the most important information are shown to the user. In the first line the user can see the name of the element and in the third line the distance from his position to the location of the item is displayed. The second line differs according to the category of the listed elements. So, for example, it contains the category (stars) of an hotel element and in the event category the event-type is displayed. If the user, during the navigation through the list, finds an element of particular interest, he has only to press the "center"-button to take a deeper look.
If the user returns to the main screen and decides to reenter in the same category, a check is performed if the change of the current GPS position is greater than a specific range (20m). If so, the list is requested again from the web-service. Otherwise if the change in the location is only small, the same list is shown to the user without downloading it again from the internet.



Request results of category hotel

Detail window

In this window the user can see all available information about the selected element. The type and amount of data is different from category to category and often also from element to element. At the bottom of the window the user can see a button which, if pressed, establishes an internet connection and downloads the picture from the involved element if available. For storage reasons the pictures are not saved locally on the phone, but must be downloaded every time the user wants to take a look at it. If the downloading of the picture is not possible or if it is currently not available, the user receives an error-message and the button disappears. During the reading of the detailed information the user can also decide to save the current element in the local storage of the phone to be able to look at the information also without internet or GPS connection. He can do this by lunching the save-command. Consequently the chosen element is copied into the local storage and a confirmation message is shown to the user. The user can save up to ten elements in his local store. If the limit is reached, no additional elements can be stored until some old elements are removed from the local storage. All locally stored elements are saved in a file when the user exits the application.
If the users returns to the list of results and reenters in the detail view of the same element the information is shown again to him without performing any rebuilding. This has the advantage, that if he loaded the picture form the web-service, it remains displayed without performing another download. Instead, if in the meantime the user looks at another element, the picture is dropped and must be reloaded.


ROSENGARTEN

Distance: 576 m

Type: GARNI

Treatment: Half board

Services:

Car Park

???GARDEN/PRIVATE_GROUNDS???

???HOTEL_BAR???

Pets welcome

Price: 3.106 Euro

Back Menu

ROSENGARTEN

take place from 2nd to 12th of December 2010: two weekends planned for soaking in the magic atmosphere of the small mountain villages, savouring in advance the enchantment of Christmas

City:

Street:

Email:

TelNo:

Image

1. Save

2. Storage

3. Back

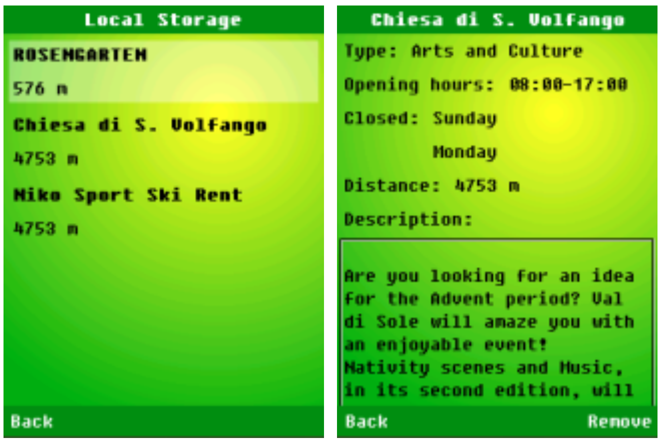
Cancel Select

Detailed view of a selected hotel, and the available commands

Local storage

From every point of the application the user can take a look in the local storage of the phone. If done so, a list appears on the screen which consists of the name of the stored element and the distance. If during the load process no GPS is available the distances are all set to 0.

Now the user can navigate through the displayed list and if he selects an element, all details of the involved element are shown. In the detail view there is also the possibility to remove the currently selected element from the local storage by lunching the remove command. Consequently the element is removed from the list and if the list was full, now it is again possible to save another element of interest.


Local Storage

ROSENGARTEN

576 m

Chiesa di S. Volfango

4753 m

Niko Sport Ski Rent

4753 m

Back

Chiesa di S. Volfango

Type: Arts and Culture

Opening hours: 08:00-17:00

Closed: Sunday

Monday

Distance: 4753 m

Description:

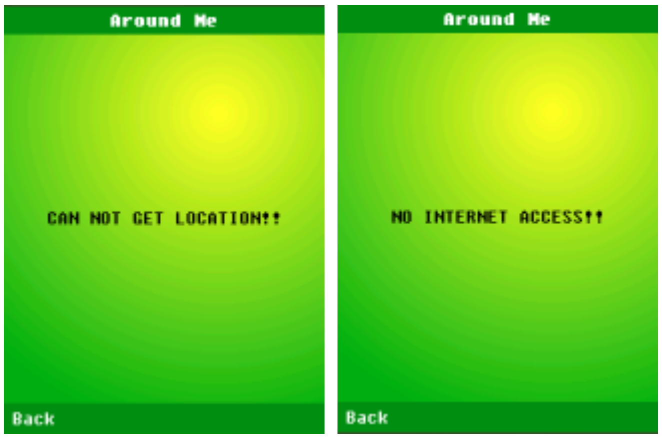
Are you looking for an idea for the Advent period? Val di Sole will amaze you with an enjoyable event! Nativity scenes and Music, in its second edition, will

Back Remove

On the left the local stored items are listed sequentially, on the right the detail view of an stored element

Error screens

There is always the possibility that the phone at the moment of use, for some reasons, cannot connect to the internet or that the GPS position cannot be determined. In such a case an error screen is shown to the user which informs him about the error occurred during the usage of the application.


Around Me

CAN NOT GET LOCATION!!

Back

Around Me

NO INTERNET ACCESS!!

Back

Error messages when location cannot be determined or internet is not available

5) Technologies used

kXML

kXML is a small XML pull parser, specially designed for constrained environments such as Applets, Personal Java or MIDP devices. Pull based XML parsing combines some advantages of SAX and DOM:

- In contrast to push parsers (SAX), pull parsers such as kXML make it possible to model the XML processing routines after the structure of the processed XML document. It is similar to an InputStream. If a part of the stream requires special handling, the parser can simply be delegated to a specialized method by handing over the parser.
- While the above is also possible with an explicit DOM, DOM usually requires that the whole document structure is present in main memory
- In contrast to DOM based parsing, the XML events are accessible immediately when they are available, it is not necessary to wait for the whole tree to build up.

LWUIT

The Lightweight User Interface Toolkit (LWUIT) is a versatile and compact API for creating attractive application user interfaces for mobile devices. LWUIT offers a basic set of components, flexible layouts, style and theming, animated screen transitions, and a simple and useful event-handling mechanism. The toolkit has been tested and debugged on a variety of mobile devices.

Location API

The Location API for J2ME specification defines an optional package, `javax.microedition.location`, that enables developers to write wireless location-based applications and services for resource-limited devices like mobile phones.

6) Technical problems

The parsing of the xml file we received from the web service was initially a challenge because it has to work dynamically, this means we don’t know in advance what tag follows the actual one and if it is empty (for example for one hotel there are services listed, for the other not).
Another problem was to store the elements such as hotel. We didn’t want to store the data in the RecordStore as key/value pair, because it is not so well structured and ordered. Therefore we tried to use the class

`Document` (org.kxml2.kdom.Document). After we have created for every item the elements, we found out that the kxml parser cannot handle the additional symbols added by J2ME for each line. So we decided to write the whole xml in one line and obmit the first additional symbol. It worked fine, until we have stored several elements. We discovered that there is a limit of characters per line. At the end we solved our problem by writing in a file ID's with their values for each variable on the `DataOutputStream`. During the search of a solution for our problem we discovered an external library named XStream(http://xstream.codehaus.org/index.html) where each java object can be transformed in xml and viceversa (within a few lines of code). Since in J2ME important classes in the util package (java.util.map) are missing, we could not use this library.

7) Future work

- Possibility to plan future trip in advance providing the location (using GoogleMaps API) and time
- Displaying the current position and the location of the matching elements on a map. If the user is interested in one of them, he can select the number displayed at the location of the desired element
- The distance between the specified location and the location of an element should be computed by the server; because of the limited CPU and memory capabilities of the mobile device. Goal: use server resources as most as possible. So we can maximize the performance and minimize waiting time of the user. Additionally the server should receive from the client its position and order the matching items according to their distances to the user
- Stored items can be stale after some time (for example "events"), so the program should manage them
- There is the possibility that the web service gets many requests, so the requests should be queued and executed one at a time (FIFO)
- The client requests only specific data of a type. To minimize the exchange of data the webservice should only return an xml file containing the data needed (and no other information)
- To optimize the search results, an additional feature is to consider the weather
- The distance should be pre-computed by the server, so the client does not have to compute it (save computing resources)
- Only those elements that are active at specified time should be displayed (returned by the server)