Abstract

This report describes the Around(J2)ME midlet application that has been developed as a project for the Mobile Services course at the Free University of Bozen-Bolzano. The aim of the application is to provide information what is around the user (e.g. banks, bars, pubs, restaurants) using the mobile phone’s current location. In particular, the report presents the system functionalities, the architecture, the user interface, development strategies and technical problems that have been discovered during development.
Contents

1 System functionalities 3
   1.1 General description of the major user functions  . . . . . . . . 3
   1.2 Use case diagram . . . . . . . . . . . . . . . . . . . . . . . 3
   1.3 Detailed description of the use cases . . . . . . . . . . . . . . 4

2 Architecture 6
   2.1 Used libraries and software tools . . . . . . . . . . . . . . . . 6
   2.2 Internal logic . . . . . . . . . . . . . . . . . . . . . . . . . . 6
      2.2.1 Package “com.aroundj2me.main” . . . . . . . . . . . . 7
      2.2.2 Package “com.aroundj2me.location” . . . . . . . . . . . 8
      2.2.3 Package “com.aroundj2me.controller” . . . . . . . . . . . 9
      2.2.4 Package “com.aroundj2me.model” . . . . . . . . . . . . . 10
      2.2.5 Package “com.aroundj2me.view” . . . . . . . . . . . . . 12
      2.2.6 Package “com.aroundj2me.executableTasks” . . . . . . 14
      2.2.7 Package “com.aroundj2me.utility” . . . . . . . . . . . . 15
      2.2.8 Package “com.aroundj2me.xml” . . . . . . . . . . . . . . 16
      2.2.9 Package “com.aroundj2me.uicomponents” . . . . . . . . 16
   2.3 Client - Server communication . . . . . . . . . . . . . . . . . . 17
   2.4 MVC like structuring . . . . . . . . . . . . . . . . . . . . . . . 17

3 User interface 19

4 Development strategies 22
   4.1 Handling location data . . . . . . . . . . . . . . . . . . . . . . 22
   4.2 JUnit testing . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
   4.3 ExecutableTask “pattern” . . . . . . . . . . . . . . . . . . . . . 23
   4.4 The ScreenController - managing Displayables . . . . . . . . . 23

5 Technical problems 25
   5.1 Location API - Landmark, LandmarkStore and Categories . . . 25

6 Future development 26
1 System functionalities

1.1 General description of the major user functions

Around(J2)ME is a location-based J2ME application that uses the phone’s current location in order to provide all nearby places that are either belonging to a pre-defined category (e.g. banks, bars, pubs, restaurants, taxis, theaters, parking) or that match a certain search criteria (e.g. name, description, address). Similar to a car navigation system, the user is able to connect to the Around(J2)ME-server to download location-related data organized into continents (i.e. Africa, Asia, Europe, North America and South America) to the mobile device. The user can then browse the retrieved locations offline without having to rely on network connectivity. Furthermore the current position and the place of interest can be displayed on a Google Map for a better understanding of its location.

1.2 Use case diagram

![Use case diagram](image)

Figure 1: Use case diagram
1.3 Detailed description of the use cases

<table>
<thead>
<tr>
<th>Name</th>
<th>Find nearby places</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The user has the possibility to select a category from a list of possible ones (e.g. banks, bars, pubs, restaurants, taxis, theaters, parking) which are provided by the application. By choosing a category, the application will retrieve all places that are nearby the user’s current position.</td>
</tr>
</tbody>
</table>

Table 1: Find nearby places

<table>
<thead>
<tr>
<th>Name</th>
<th>Find nearby places by category</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The user has the possibility to select a category from a list of possible ones (e.g. banks, bars, pubs, restaurants, taxis, theaters, parking) which are provided by the application. By choosing a category, the application will retrieve all places that are nearby the user’s current position.</td>
</tr>
</tbody>
</table>

Table 2: Find nearby places by category

<table>
<thead>
<tr>
<th>Name</th>
<th>Find nearby places by search criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The user can open a search form on the mobile phone where he can enter a search query. By clicking on the search button, a search for places matching the entered search query and being around the current location of the user will be done and the result displayed to the user.</td>
</tr>
</tbody>
</table>

Table 3: Find nearby places by search criteria
### Name | View place
---|---
**Description** | By clicking on the different places on the list, found by one of the different possibilities (see use cases 1, 2 and 3), the user can view the details about that place. This detail view shows information such as the distance from the user’s current position, address information (i.e. street, city,...), contact information such as the phone number and a short textual description about the place of interest.

| Name | View place on map
---|---
**Description** | When viewing a place found by one of the different possibilities (see use cases 1, 2 and 3), the user can display this place on a map.

| Name | Manage location data
---|---
**Description** | All of the location data is managed locally on the device. The user can manage this data by retrieving/updating it from the server. The places can be downloaded for each continent independently.

Table 4: View place

Table 5: View place on map

Table 6: Manage location data
2 Architecture

2.1 Used libraries and software tools

The following table shows the tools and libraries that have been used for the development of the Around(J2)ME application.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>Main development IDE</td>
<td><a href="http://www.eclipse.org">http://www.eclipse.org</a></td>
</tr>
<tr>
<td>MTJ</td>
<td>Eclipse plugin Mobile Tools for Java (former EclipseME)</td>
<td><a href="http://www.eclipse.org/dsdp/mtj/">http://www.eclipse.org/dsdp/mtj/</a></td>
</tr>
<tr>
<td>Apache</td>
<td>Used webserver</td>
<td><a href="http://tomcat.apache.org/">http://tomcat.apache.org/</a></td>
</tr>
<tr>
<td>Tomcat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kXML2</td>
<td>XML pull parser library</td>
<td><a href="http://kxml.sourceforge.net/">http://kxml.sourceforge.net/</a></td>
</tr>
<tr>
<td>J2MEUnit</td>
<td>Unit testing library for J2ME</td>
<td><a href="http://j2meunit.sourceforge.net/">http://j2meunit.sourceforge.net/</a></td>
</tr>
</tbody>
</table>

Table 7: libraries and tools used for development

2.2 Internal logic

This section describes the internal logic of the application in terms of appropriate UML diagrams.
2.2.1 Package “com.aroundj2me.main”

Figure 2: Packages

Figure 3: Packages
Around(J2)ME

2.2.2 Package “com.aroundj2me.location”

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AroundJ2ME</td>
<td>This is the application MIDlet and is the entry point of the Around(J2)ME application. This class is responsible for starting up the application and for calling the appropriate application initialization methods.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocationService</td>
<td>The LocationService is the central point of interaction with the JSR-179 Location API. It provides methods for retrieving the current location as well as for storing and retrieving Landmark objects to and from the LandmarkStore.</td>
</tr>
</tbody>
</table>
2.2.3 Package “com.aroundj2me.controller”

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationController</td>
<td>This class contains the main domain logic. It is the main point of interaction between the user interface. Each user interface class contains a reference to the ApplicationController. For more details about this MVC-like structure refer to section 2.4.</td>
</tr>
<tr>
<td>ScreenController</td>
<td>The ScreenController class is responsible for managing the different kind of J2ME Displayable objects on the UI. All the displaying and removing of Displayable objects is done over this class. This guarantees to have a single point of handling UI windows and a clean structure throughout the application. For managing the different Displayable’s, this class uses a Stack (java.util.Stack) datastructure. Moreover it includes convenient functionalities such as to automatically add a “back” command to the Displayable that is being displayed. More in section 4.4</td>
</tr>
</tbody>
</table>
### Package “com.aroundj2me.model”

**Figure 6: Packages**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place</td>
<td>This is the class for holding data of a place retrieved from the local landmark store that is being displayed on the UI to the user.</td>
</tr>
</tbody>
</table>
### Architecture

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContinentPack</td>
<td>This class is used for transporting the data about installed continents.</td>
</tr>
<tr>
<td>ExecutableTask</td>
<td>This abstract class is being inherited by classes that encapsulate so-called “executable tasks”. This is a pattern that has been used in the Around(J2)ME application for handling threaded code in a uniform and convenient way. For more information please refer to section 4.3.</td>
</tr>
<tr>
<td>TaskProcessListener</td>
<td>This is an abstract class that is used for attaching to the execution of an ExecutableTask for getting notified about the successful termination or occurrence of errors.</td>
</tr>
<tr>
<td>Constants</td>
<td>This class contains commonly used constants throughout the application</td>
</tr>
<tr>
<td>CategoryName</td>
<td>This is a class containing all of the category names as constant strings in order to have a central point in the program for changing category names.</td>
</tr>
<tr>
<td>ContinentName</td>
<td>Similar to the CategoryName, this class holds constant strings for all of the available continents.</td>
</tr>
<tr>
<td>Messages</td>
<td>This class holds the messages that are being displayed to the user on the UI.</td>
</tr>
<tr>
<td>BackCommand</td>
<td>This class extends the javax.microedition.lcdui.Command class and is used by the ScreenController to automatically hook a “back” command on the Displayable objec that is being shown on the UI.</td>
</tr>
</tbody>
</table>
2.2.5 Package “com.aroundj2me.view”

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IView</td>
<td>This is the interface that is implemented by all of the view classes defined in the application. This construct facilitates the easy construction and reuse of view objects in the application.</td>
</tr>
<tr>
<td>CategoryListView</td>
<td>This is the UI list where all of the available categories are displayed to the user. The user can choose a category for getting all of the according places.</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ContinentListView</td>
<td>This is the list showing all the available continents. Here the user can download or update his places from the server. Appropriate icons show whether a certain continent with its data is already installed or whether it has still to be downloaded.</td>
</tr>
<tr>
<td>MapView</td>
<td>This view shows a Google Map with the current position of the user marked with a red point and the desired place with a blue point.</td>
</tr>
<tr>
<td>PlaceListView</td>
<td>This is the list displayed to the user with all of the retrieved nearby places.</td>
</tr>
<tr>
<td>PlaceView</td>
<td>On this view, the details about the selected place such as the title, description, address etc. . . are displayed to the user.</td>
</tr>
<tr>
<td>SearchView</td>
<td>This is the view where the user can enter search queries for retrieving nearby places.</td>
</tr>
<tr>
<td>SplashScreen</td>
<td>This is the initial splash screen that is shown during initialization at the application startup.</td>
</tr>
<tr>
<td>AboutForm</td>
<td>This is a view for showing information related to the Around(J2)ME application such as the version and the authors.</td>
</tr>
<tr>
<td>IProgressForm</td>
<td>This is the interface for views showing progress information to the user. It defines the methods that should be implemented.</td>
</tr>
<tr>
<td>ProgressForm</td>
<td>This class implements IProgressForm and displays a progress bar together with a status message for informing the user about the ongoing operations.</td>
</tr>
</tbody>
</table>
2.2.6 Package “com.aroundj2me.executableTasks”

All of the classes extend the abstract class “ExecutableTask”. For more information about this kind of pattern see section 4.3.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlaceFinder</td>
<td>This class is responsible for reading the places that match the current user’s location from the local landmark store.</td>
</tr>
<tr>
<td>ContinentRetriever</td>
<td>This class is responsible for establishing a connection to the server and for downloading the according data for the continent chosen by the user.</td>
</tr>
<tr>
<td>GoogleMapsRetriever</td>
<td>This class is responsible for contacting the Google Maps API for retrieving the map for displaying the user’s location and selected point of interest.</td>
</tr>
</tbody>
</table>
### 2.2.7 Package “com.aroundj2me.utility”

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageLoader</td>
<td>This class provides static constants with the uri to the corresponding images and a static utility method for loading them and creating the according in-memory image object.</td>
</tr>
<tr>
<td>StringUtils</td>
<td>This class contains utility methods for working with strings which are missing in the J2ME library such as replace and split methods.</td>
</tr>
<tr>
<td>HttpCommunicator</td>
<td>This class provides the logic for performing HTTP calls and for reading the returned data.</td>
</tr>
<tr>
<td>AlertCreator</td>
<td>This class contains utility methods for creating various Alert objects to be displayed on the screen.</td>
</tr>
</tbody>
</table>
2.2.8 Package “com.aroundj2me.xml”

![XMLDecoder class diagram]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLDecoder</td>
<td>This class is responsible for parsing XML documents by using the XML pull parser technology (kXML).</td>
</tr>
</tbody>
</table>

2.2.9 Package “com.aroundj2me.uicomponents”

![UIC model]

Figure 10: Packages

Figure 11: Packages
### 2.3 Client - Server communication

The main application logic is performed offline, without relying on a server for providing the data. The only communication with the server is for downloading the location data to the client where it is then stored for performing operations locally.

The application connects to the server by performing a HTTP GET operation passing the continent as parameter. The server processes the request and encodes the result as XML file which is sent back to the mobile client. There the result is parsed by using the kXML parser and stored locally inside a LandmarkStore.

![Client - Server communication diagram](image)

**Figure 12:** Free form diagram showing the client - server communication

### 2.4 MVC like structuring

For separating the user interface logic, the domain logic and the model classes, a very simplified MVC like structure has been used. Mostly the concept of the layering and decoupling has been taken from the MVC pattern.

- **Controller**
  The ApplicationController class plays the role of the controller. All of the calls to the actual application logic go through this class. It functions as a mediator that gets the UI actions initiated from the user’s
interaction, executes the appropriate actions on the different domain classes (model) and then displays the changed model data again on the user interface.

• **Model**
  The model part of the MVC is represented by the classes in the com.aroundj2me.model package. These are the classes that contain the data that is being displayed (by the ApplicationController) on the UI.

• **View**
  The view part is represented by the classes in the com.aroundj2me.view package. All of these classes contain mainly user interface related logic such as constructing the appropriate elements for showing the data and for handling user events.

It has to be stated that a full implementation of the MVC pattern has been avoided for minimizing the amount of objects that are involved in such a structure. The MVC\(^1\) or MVP\(^2\) model in a J2SE or J2EE environment involves a lot of objects and event handling which could pose a possible overhead on the mobile device.

\(^1\)[http://blog.js-development.com/2008/03/logical-separation-with-mvc.html]
3 User interface

This section shows some screenshots presenting the main functionalities of the Around(J2)ME application.

![Screenshots showing the main functionalities](image)

(a) Main menu  
(b) Manage places  
(c) Places search  
(d) Information message

Figure 13: Screenshots showing the main functionalities
Figure 14: Screenshots showing the main functionalities
Figure 15: Screenshots showing the main functionalities

(a) Map view normal
(b) Map view satellite
(c) Map view hybrid
4 Development strategies

4.1 Handling location data

Location data is managed by heavily using the JSR-179 Location API. Places are represented by the Location API’s Landmark class, which are stored inside a LandmarkStore together with appropriate categories. There is one landmark store for each continent.

When the user wants to retrieve places around his current position by one of the available functionalities (either through a search, by a certain category or just all nearby places), the local landmark stores are queried for locations of about 3 kilometers around the user’s current position.

4.2 JUnit testing

The J2MEUnit\(^3\) library has been used for writing JUnit tests for the mobile device. This ensured the proper working of the implemented features during the development lifecycle.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{JUnitTests.png}
\caption{JUnit tests running on the WTK simulator}
\end{figure}

\(^3\)http://j2meunit.sourceforge.net/
4.3 ExecutableTask “pattern”

When developing J2ME applications there are always again operations
that have to be launched inside a separate thread in order to first not block
the main user interface and second to be able to provide valuable information
at the same time to the user, e.g. a progress window about the ongoing
operation. In order to have the same reusable structure for handling these
threaded operations throughout the Around(J2)ME application, a so-called
“Executable task pattern” has been created.

The base classes involved in this construct are the ExecutableTask and
the TaskProcessListener. Both are abstract classes and get implemented by
the actual executable task classes (see section 2.2.6). The ExecutableTask
implements the runnable interface for being launched inside a thread and
it accepts a TaskProcessListener for being notified about the status of the
ongoing operation.

Executing an operation that extends the ExecutableTask class can then
be handled in this simple way:

```java
executableTask . setListener(new TaskProcessListener () {
  public void onError(String status) {
    //an error happened, notify the user
  }

  public void taskFinished(Object result) {
    //everything went fine..
    //do something with the ‘‘result’’
  }
});
```

```java
Thread t = new Thread (executableTask);
t . start ();
```

4.4 The ScreenController - managing Displayables

Mobile phones don’t dispose of a multi-window environment as in normal
desktop environments. Always just one window at a time can be shown
on the display where users can usually navigate forward and backward be-
tween different windows. This kind of behaviour matches perfectly a stack
datastructure: if the user navigates to a new screen, the current screen will
be pushed on the stack. When the user activates the back button, the last
screen will be popped from the stack and again displayed.
These kind of operations are managed by the ScreenController. It is a singleton class that is used throughout the Around(J2)ME application as a central point for handling display related operations.
5 Technical problems

5.1 Location API - Landmark, LandmarkStore and Categories

One of the major technical issues that have been encountered during development was a problem with the JSR-179 Location API. Around(J2)ME uses categories such as “Bars, Restaurants, Pubs, Hotels,...” for organizing places. These categories are associated to the landmarks when they are persisted in the LandmarkStore. When retrieving a certain landmark from the store it is however not possible to know which kind of category it has associated. A Landmark object has an array of associated categories but provides no public accessor for retrieving it.

```
landmarks:
int nCategories:
for(int i = 0; i < nCategories; i++)
```

Figure 17: Landmark’s categories array not publicly accessible

Such information is however needed by the Around(J2)ME application. When the user retrieves nearby places without any restriction, he would like to know immediately which kind of category a found place belongs to. For instance whether it is a bar, pub or restaurant. As a workaround, the description field of the Landmark class has been used to store the category information.
6 Future development

Future development of the application could go in the direction of extending the server logic for providing features like

- letting the user enter new interesting places locally and submit them to the server.
- letting the user rate/comment places (i.e. rate restaurants, pubs, bars . . .)

Another feature could be to enhance the navigation support, that is, enhancing the Google Map to continuously track and update the user’s position on the map by exploiting the LocationListener of the JSR-179 Location API.