HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

# Android App: LoationReminder

**Vladyslav Pekker,
Stephan Hagios,
Felix Schmidt**

**Konstanz, 26.01.2012**

Mobile Computing WS11/12

## Contents

## List of Figures

## 1 Introduction

Mobile computing is winning over more users with every day. Nowadays virtually everybody owns a smartphone or at least a similar mobile device. Some people even use more than one mobile device in their everyday life. This "addiction" makes people rely more and more on their daily mobile helpers. They get very busy during the day, which makes it easy to forget even simple tasks such as buying milk. The problem is that when the user does remember accomplishing something, he is prevented due to various factors like location differences.

Our goal is to provide a solution for a simple to-do list combined with a location aware service, which would remind the user of a planned activity when he or she gets near the provided geo point.

## 2 Requirements

The tool is an android application. An Android 4.0 smartphone or higher is required. A GPS connection has to be maintained through the complete life cycle of the application. An established Internet connection is also assumed to enable the creation, editing and showing of a to-do task.

## 3 Architecture

The following section provides an insight over the structure and architecture of the application.

### 3.1 Components

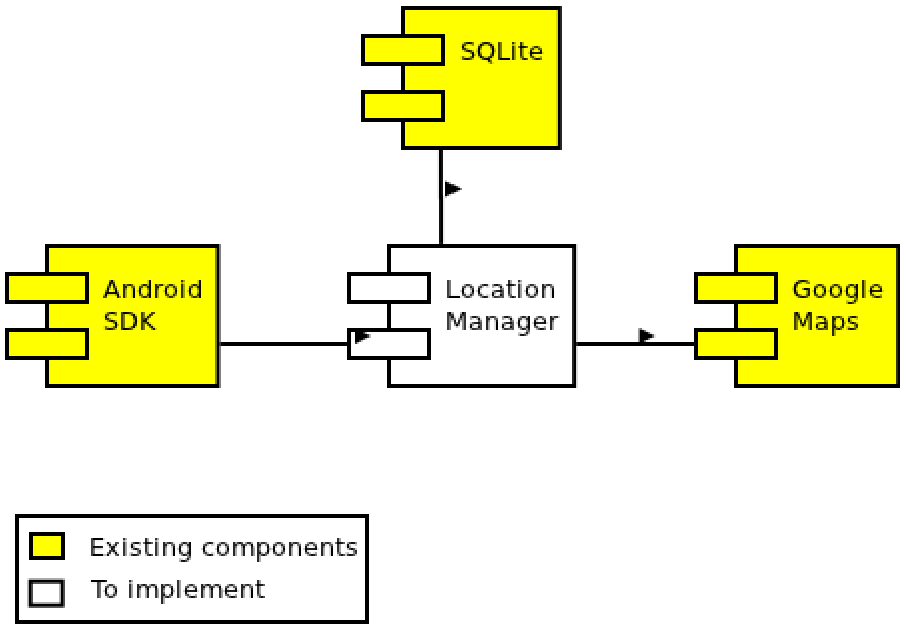
The components presented in the Figure 1 are divided into two categories:

**yellow** already existing components

**white** components, which have to be implemented.

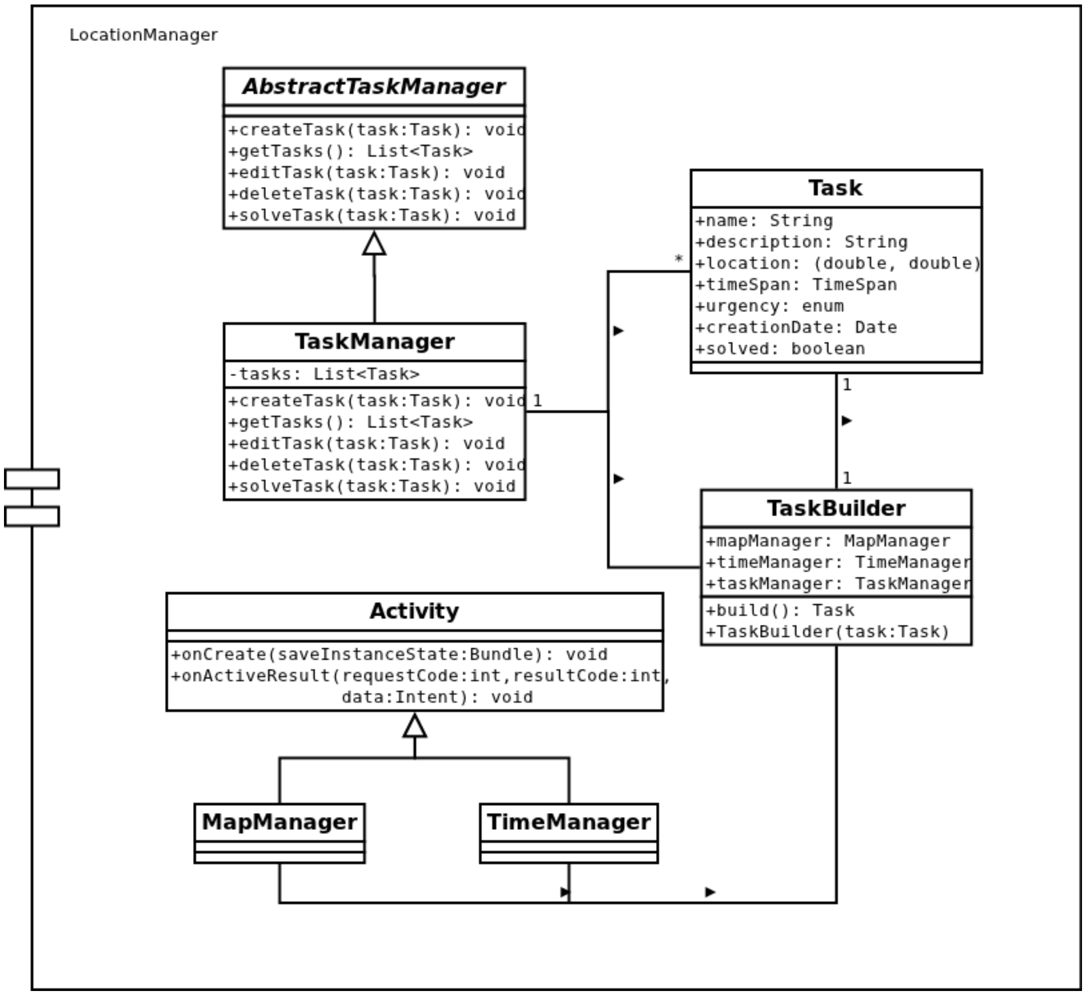
SQL Lite database was chosen, because it is directly supported by the Android SDK. The same reasoning applies to the selection of the Google Maps API.

Figure 1: LocationManager Component Diagram



The *TaskManager* is the core component of the application. It takes care of creating, editing, saving and deleting tasks. The *TaskBuilder* is the central component of the delivery system of the application. It controls the *MapManager*, the *TimeManager* and the *TaskManager* respectively. Both the *MapManager* and the *TimeManager* are derived from the *Activity*, which is the main part of the Android application life-cycle. The *MapManager* handles the view of the map, by using the Google Maps API. The *TimeManager* uses standard Android widgets to assist the user in selecting the time span.

Figure 2: LocationManager Class Diagram


LocationManager
AbstractTaskManager
+createTask(task:Task): void
+getTasks(): List<Task>
+editTask(task:Task): void
+deleteTask(task:Task): void
+solveTask(task:Task): void
TaskManager
-tasks: List<Task>
+createTask(task:Task): void
+getTasks(): List<Task>
+editTask(task:Task): void
+deleteTask(task:Task): void
+solveTask(task:Task): void
Task
+name: String
+description: String
+location: (double, double)
+timeSpan: TimeSpan
+urgency: enum
+creationDate: Date
+solved: boolean
TaskBuilder
+mapManager: MapManager
+timeManager: TimeManager
+taskManager: TaskManager
+build(): Task
+TaskBuilder(task:Task)
Activity
+onCreate(saveInstanceState:Bundle): void
+onActiveResult(requestCode:int,resultCode:int, data:Intent): void
MapManager
TimeManager
1
*
1
1

## 3.2 Used Technologies

The application is written in Java and built with Eclipse. The SQL Lite database was used to persist the tasks and further vital information. Android API were used to determine the current location. The Google Maps API was used to visually represent the coordinates.

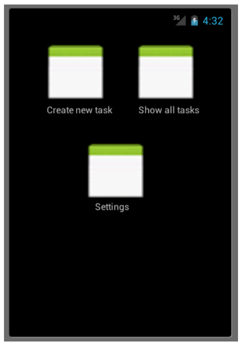
# 4 User Manual

## 4.1 Overview

When the application is started the user can choose between three features.

- Creating a new task
- Showing all tasks
- Modify the settings

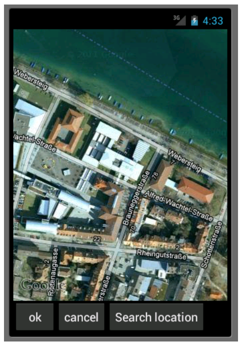
Figure 3: Main screen



## 4.2 Creating a task

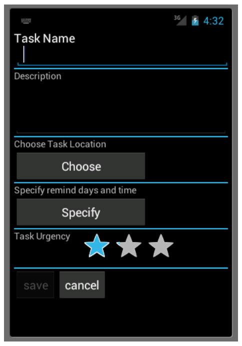
In order to create a new task the user has to assign the *task name*, a *task description* and link it to a *location*. Google Maps is used to assist the user with the search for a location. The user has the ability to link his task to a

Figure 4: Choosing a location



time span. This feature will ensure, that the user is not only reminded when he is in range of the given location point, but also when the current time matches the specified time span. In case the time span is not explicitly specified, the application defaults to always reminding the user. The user might also assign an urgency indicator to a task. The urgency will be represented by a color code. There are three urgency states: *high*, *middle* and *low*, represented by red, yellow and green colors respectively.

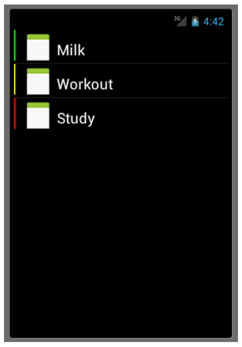
Figure 5: Creating a task



## 4.3 Showing all available tasks

All tasks are shown in a simple list. The urgency is represented by a vertical colored line on the left side of the icon. The icon indicates the status of a given to-do task. The user can press one of the items so a further menu is displayed. The menu contains features described in the following sections.

Figure 6: Showing all available tasks

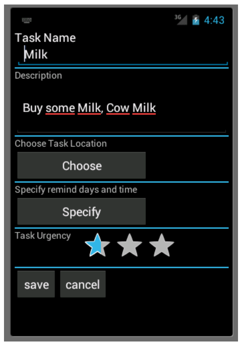


## 4.4 Editing a task

The *edit task* feature is available from the *show all tasks* screen. In order to edit a task the user has to select the desired task by touching it. The menu is inflated and the *edit task* item has to be selected.

The screen shown to the user is identical to the screen he already knows from the *create task* feature. This time all predefined fields are already filled in and their content may be altered as desired.

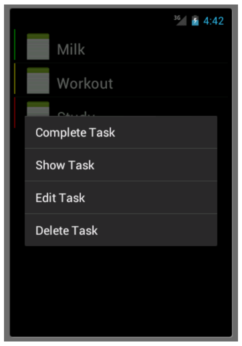
Figure 7: Editing a task



## 4.5 Deleting a task

The *delete task* feature is also available from the *show all tasks* screen. In order to delete a task the user has to select the desired task by touching it. The menu is inflated and the *delete task* item has to be selected. After the task has been deleted the user is redirected to the *show all tasks* screen.

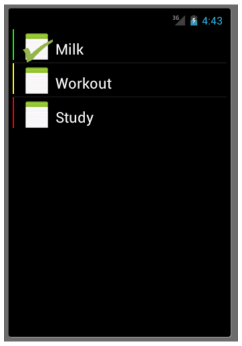
Figure 8: Item menu



## 4.6 Solving a task

The *solve task* feature is also available from the *show all tasks* screen. In order to complete a task the user has to select the desired task by touching it. The menu is inflated and the *complete task* item has to be selected. After the task has been completed the user is redirected to the *show all tasks* screen. This time the icon of the task is altered. The icon generally indicates the status of the task. The status is restored to *unsolved* after the task has been edited.

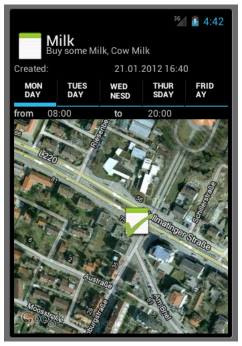
Figure 9: Completed task



## 4.7 Showing a task

The *show task* feature displays the details of the given tasks. It is also available from the *show all tasks* screen. In order to show a task the user has to select the desired task by touching it. The menu is inflated and the *show task* item has to be selected. The following figure shows the representation of the task.

Figure 10: Showing a task



Milk
Buy some Milk, Cow Milk
Created: 21.01.2012 16:40

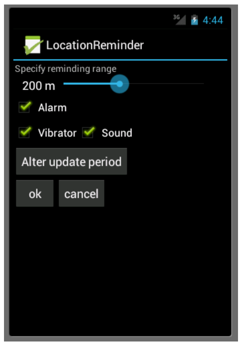
from 08:00 to 20:00

## 4.8 Configuring the application

The user has the ability to configure the following options:

- Setting the reminding radius
- Setting the update frequency
- Adjusting the reminding mode

Figure 11: Settings screen



Specify reminding range

200 m

✓ Alarm

✓ Vibrator ✓ Sound

### 4.8.1 Setting the reminding radius

The radius specifies the number of meters from the current position to the linked to-do item. The radius has to be in range from 0 up to 500 meters. A horizontal scrollbar makes sure that the user stays within this range.

### 4.8.2 Setting the update frequency

A service runs in the background of the application, which provides the current position of the smartphone. The user has the ability to specify the frequency of how often this service updates its data. The user may pick one of the many options, spanning from 1 minute up to 24 hours, represented by a drop-down list.

### 4.8.3 Adjusting the reminding mode

The reminder can be turned on and off. The user may select the way he wants to be reminded when the reminder is on. The options are *ring-alarm*, *vibration* or both.