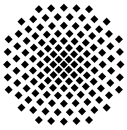


Universität Stuttgart

ifp

Development of Android based app for management of communal geoinformation

Deordic Dejan

supervised by

Dr.-Ing. Volker Walter

and

Dr.-Ing.habil. Wolfgang Bischoff

22.12.2015

Abstract

Communal geoinformation has to be maintained in an effective way in order to have up-to-date information about communal property. Collection and management of geoinformation is a very important component of any Geographical Information System (GIS), because Geographical Information System can visualize and make complete analysis if up-to-date and correct data is provided. We are living in the mobile world, so mobile solution for off-line and on-line management of communal geoinformation makes perfect sense.

The purpose of this project is to identify suitable data storage option which could be used for storing large amount of communal geoinformation on Android platform, i.e. hundreds of Megabyte (MB) of files, under specific conditions concerning query speed and generation of Graphical User Interface (GUI). The main role of dialogs is the representation of GUI for user interaction with localized database on a mobile device. Of course, Database Management System (DBMS) for Android platform has to be developed in order to manage data stored on this kind of a platform.

In order to find the best solution, two test apps were developed which shows that SQLite database is a better solution than Extensible Markup Language (XML) based storage option for this project.

This Master's thesis also investigated technologies for data transfer between local database on a mobile device and a database managed on the server. After few solutions are investigated and tested, Simple Object Access Protocol (SOAP) web service based on Hypertext Transfer Protocol Secure (HTTPS) is chosen for data transfer technology.

In the end, the developed app is presented and chief functional aspects are shown and explained. This Master's thesis is done in cooperation with CWSM Software Solutions GmbH.

Keywords: Mobile GIS, SAGis mobile app, web service, communal geoinformation, app development

1 Introduction

SAGis mobile app for management of communal geoinformation, which is apart from scientific research the main outcome of Master's thesis project, is mostly needed for government sectors in Germany. The outcome of this thesis is SAGis mobile app; that app is intended as a tool for municipals employees to collect and maintain geoinformation directly in the field, geoinformation such as tree cadastre and traffic signs. The app is developed for mobile devices running on Android OS. The main reason for choosing such a platform is the fact that Android dominated smartphone market with a share of 82,8 percent in 2015 comparing to iOS with a share of 13,9 percent as stated in International Data Corporation report[1].

Another reason for choosing Android platform is that the domination of Android OS is also expected in next few years, so product should target the majority of the smartphone and tablet market.

This Master's thesis is mostly dealing with GIS and Geoinformatics concepts, mainly databases and web service. Its area of study is Mobile GIS. The thesis showed that app for management of communal geoinformation can be developed meeting the most important requirements set by CWSM GmbH, i.e. search query speed must be under 2 seconds and opening time of the dialogs for user interaction with database must be less than 3 seconds.

Mobile GIS market is also investigated together with current technology for Web Service development and Data Base Management System suitable for Android devices to store large amount of data and web services, such as Simple Object Access Protocol.

1.1 Presentation of the topic and the problem

This Master's thesis does not concentrate only on doing research concerning Android development, databases and communal geoinformation. A very important aspect was to develop software for CWSM GmbH, which will meet their requirements. Because of that the topic is in a way special and this is the main reason why requirement engineering was done. Requirement engineering and analysis of the mobile GIS market is presented in the thesis itself.

[1] IDC(website). *IDC: Smartphone OS Market Share*. 2015. URL: `http://www.idc.com/prodserv/smartphone-os-market-share.jsp` (visited on 09/27/2015).

Several questions and problems were solved during research analysis and the software development phase. One of them was to define storage options and to make speed test in order to see which one of them is the best for Android platform. Currently, municipal workers are doing paper based field work unless there is some kind of Mobile GIS system. This has to be improved by developing an app which could replace paper-based management of geoinformation. Also, dynamical user interface is compared with the user interface created from XML and detailed results are given in chapter 5 of the thesis. If a mobile app is relying only on static UI it means that if one wants to manage some additional theme, the program has to be improved and new functionality added. The goal was to develop a fully dynamical UI. i.e. just one Java class for creation dialogs for each individual theme.

1.2 Current status and problems concerning management of communal geoinformation

Management of communal geoinformation is the task which has to be done very effectively in order to have up-to-date information concerning communal property, i.e. traffic signs, trees, points of interests, green surfaces, etc. Currently, three methods frequently used are collecting the geodata using Toughbooks, paper based geodata collection and Mobile GIS based data collection.

In some local authority districts in Germany mobile GIS is used, but many municipal offices are still relying on paper-based field work during management of spatial data and geoinformation. Some municipalities are using mini Notebooks (Toughbooks) with Windows OS. These systems are weighting 3-4 kilograms because the laptops have to be engineered with security to withstand drops, spills, dust, etc. This is why such a system costs 1500-3500 Euro. This price includes only hardware.

Another problem is the number of such Toughbooks available for municipal workers. For example, 5-6 people work on management of geoinformation. If only one Toughbook is available, that means that such field work is expensive and time-consuming. Additionally, it often happens that the work is doubled, because user has to update all the changes from the paper to the database where all communal spatial data is stored. This is of course worst case scenario, when no version of Mobile GIS is used. Buying 80-200 Euro smartphone or tablet running Mobile GIS would significantly reduce the costs and enable 5-6 workers to work at the same time. During the thesis project, above mentioned problems were solved by development of SAGis mobile app.

2 Methodology of the research

Very first task of this Master's thesis project was to make requirement engineering document together with CWSM GmbH, because all requirements have to be set and clearly stated. If this is done, developer will have clear plan what and how it should be done. Requirements set by CWSM GmbH:

- Load data
- Read, write and manage the data
- Communication with the web server
- Register the user (Device name, MAC address, IMEI)
- Reliability of the app
- Quick responsiveness (1-2 seconds)
- Simple GUI
- Store 10 000 - 500 000 records
- SAGis mobile dialog should be identical to Autodesk Map 3D dialog
- On-line and off-line mode possible

Logically, next step was to investigate and test solutions which are already on the market, e.g. ArcGIS Collector, GIS Cloud Mobile and QGIS. This is done and evaluation is given in the thesis document. Shortly, it is discovered that all three apps aren't suitable for solving above mentioned problems. Also, requirements like storage of several hundred MB files, import of the SQLite/XML files couldn't be met if above mentioned apps are used.

Before app development, the research has to be conducted in order to have clear idea regarding existing technologies for data storage option and web service development and their advantages and disadvantages. After detailed research it is possible to choose suitable technology for the project, in this case application for management of communal geoinformation. This is why test apps are developed before final app development, one based on XML as storage option and the other one based on SQLite database.

TestAppXML, which is based on XML as storage option, was developed first. Of course, just one test app isn't enough for sufficient research. In order to get better and more stable results with respect to query speed and parsing time another possibility for storing data was investigated, namely **SQLite** database. As stated in official Android documentation for developers, large data sets with several different data types should be stored in a structured way using SQLite database. It is also shortly discussed which other databases for mobile devices are available on the market (Berkeley DB, Couchbase Lite, LevelDB and UnQLite). The main advantages and disadvantages of each method are noted in the Master's thesis.

After test apps development, discussion is made and it is decided which technology is more suitable for problem solving. In this case SQLite is chosen and final app is developed based on this technology. It is also important to mention that UML diagrams (class, use case and sequence) are developed as part of software development.

2.1 Test app 1 - XML and ArrayList

First test app is based on XML file, which is actually large table exported from Oracle database. XML is than parsed into ArrayList of HashMaps, which can be seen as intermediate structure for storing data on Android device. HashMap is populated with three data types: String, Integer and DateTime.

Java Heap constraint

If there is a large XML (more than 1 GB), then RAM memory is extremely important. For example, Motorola Moto G has just 1 GB RAM so the app that uses 500 MB of total RAM for parsing would significantly reduce the speed of the mobile device.

In order to see how much RAM TestAppXML app is using, a small test can be done. Java Heap memory is tracked in Android Studio (Android Integrated Development Environment), and linear growth of the Java Heap can be seen with respect to time. This is shown in Figure 1 on the next page.

The figure shows that for 30 seconds of parsing 11 MB of RAM is needed (just to put it into perspective, this is only 4 MB XML file). It also displays clear linear behaviour so it can be conclude that for parsing a 40 MB file 110 MB RAM is needed, which is large amount of Java Heap memory allocated. CWSM GmbH handles several hundred MB database files, so this criteria should be taken into consideration.

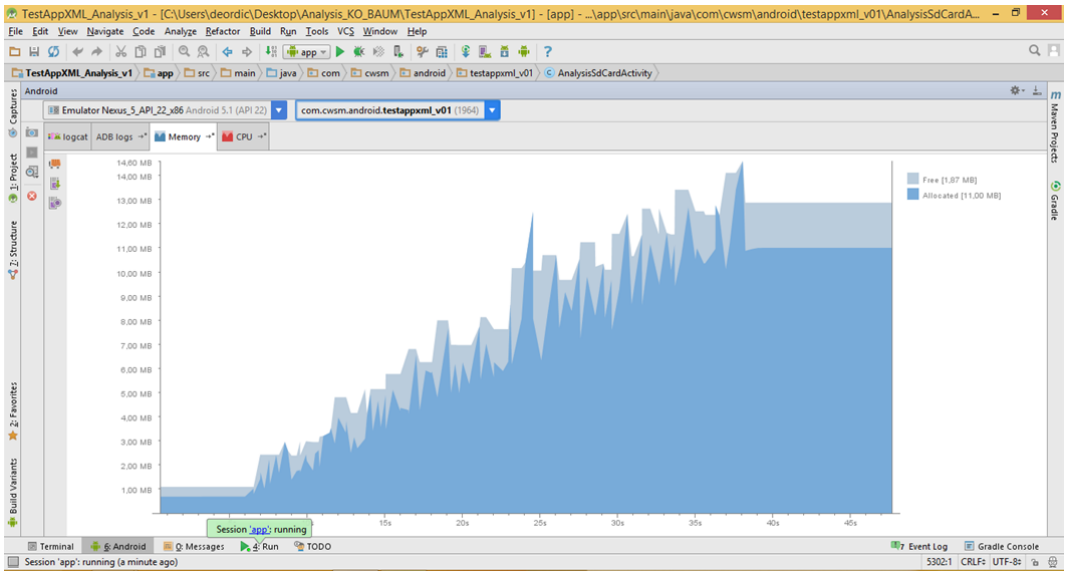


Figure 1: Java Heap constraint.

Parsing time

In this section parsing time will be discussed. The RAM analysis showed that parsing time also directly influenced the usage of RAM. For more parsing time, more RAM will be used and collected in "Java Garbage Collector". In Figure 2 parsing time is shown with respect to time.

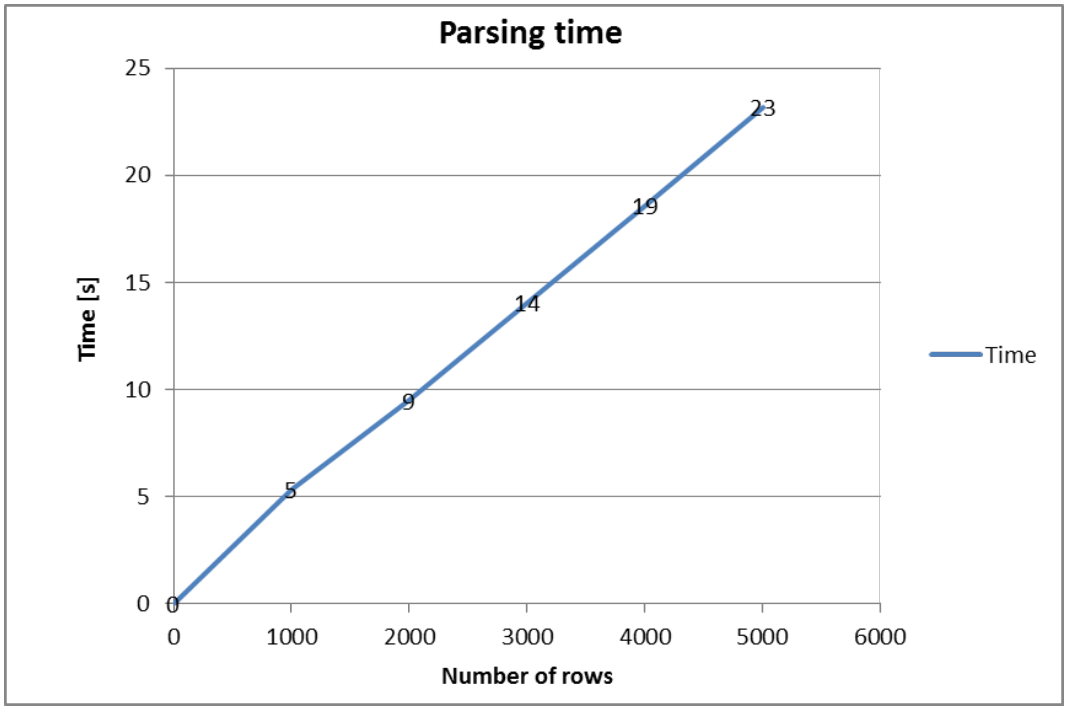


Figure 2: Linear growth of the parsing time with increasing the columns.

The graph above confirms that with every 1000th row parsing time increased by 5 seconds. The main reason for this is conversion to data types, actual RAM capacity of the device and processor speed. For a greater number of rows, e.g. 100 000 parsing time would use large amount of RAM and it would last longer than required by CWSM GmbH.

2.2 Test app 2 - SQLite database

When using SQLite, a very small amount of **Java Heap** memory is allocated. Also, it is important to mention that Java Heap memory is staying reasonably small even when parsing several hundred MB XML file to SQLite. On the following graph it can be seen that Java Heap allocated is in 4 MB range. This means that theoretically even larger – few GB XML files could be parsed into database if necessary without memory leakage.

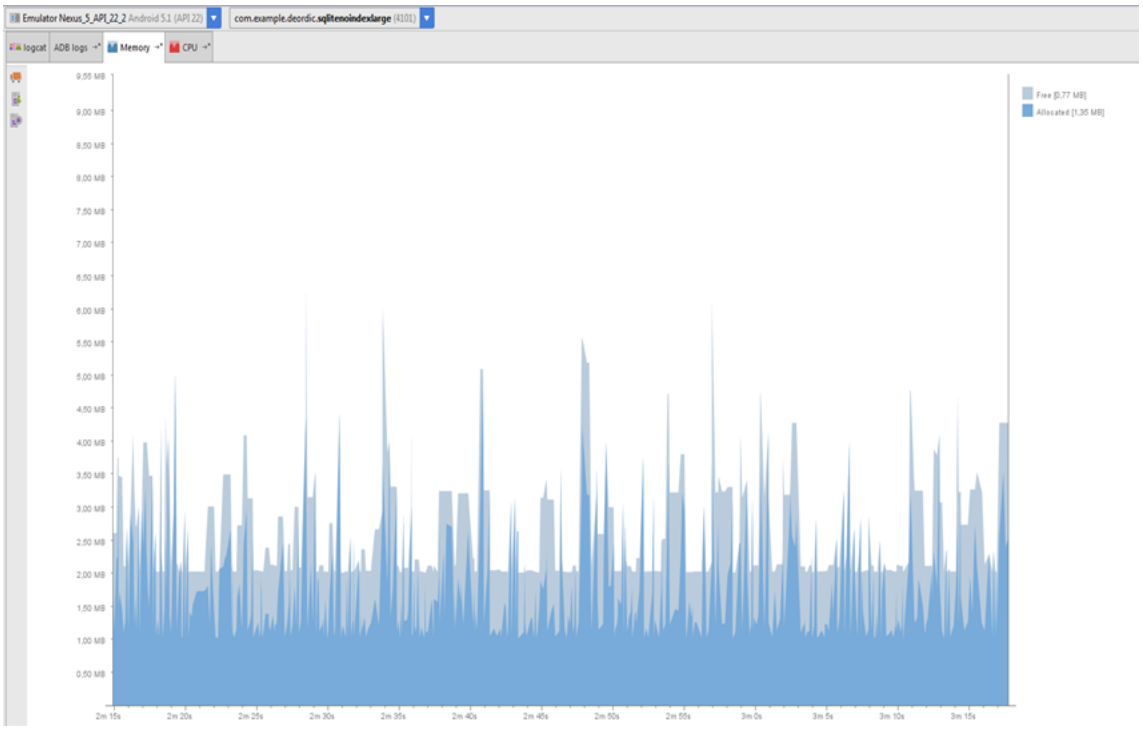


Figure 3: Java Heap memory usage – 540 MB XML file.

After **parsing** XML to SQLite database (table) it can be seen that parsing is significantly faster. Parsing in TestAppXML (5710 rows and 53 columns) lasts for 25 seconds but only 4,8 seconds in TestAppSQLite. The main reason for this is lower Java heap usage and internal SQLite conversion of Strings to appropriate data type. In order to have complete performance overview for the test app based on SQLite, performance tests with larger datasets are done. Query speeds using different test devices (Motorola MotoG

2nd Gen, Sony Xperia Z1 and Sony Xperia Z2) considering table with 100 000 rows and 150 columns is given in Table 1 below.

Table 1: Query speed - String, Integer and DateTime

100 000 rows	MotoG(ind)	MotoG(no ind)	Z1(ind)	Z1(no ind)	Z2(ind)	Z2(no ind)
Units	t[s]	t[s]	t[s]	t[s]	t[s]	t[s]
String	0,002	6,26	0,007	5,664	0,006	5,645
Integer	0,004	6,23	0,007	5,734	0,008	5,63
DateTime	0,003	6,147	0,004	5,602	0,007	5,72

After a discussion at CWSM GmbH it is decided that the requirement for parsing speed is no longer valid. Such a long parsing time can only be accepted under condition that parsing has to be preformed only once when application is started just to create the database. Later, queries can be executed without parsing every time. The results for the smaller dataset (5710 rows and 53 columns) are given in Figure 4.

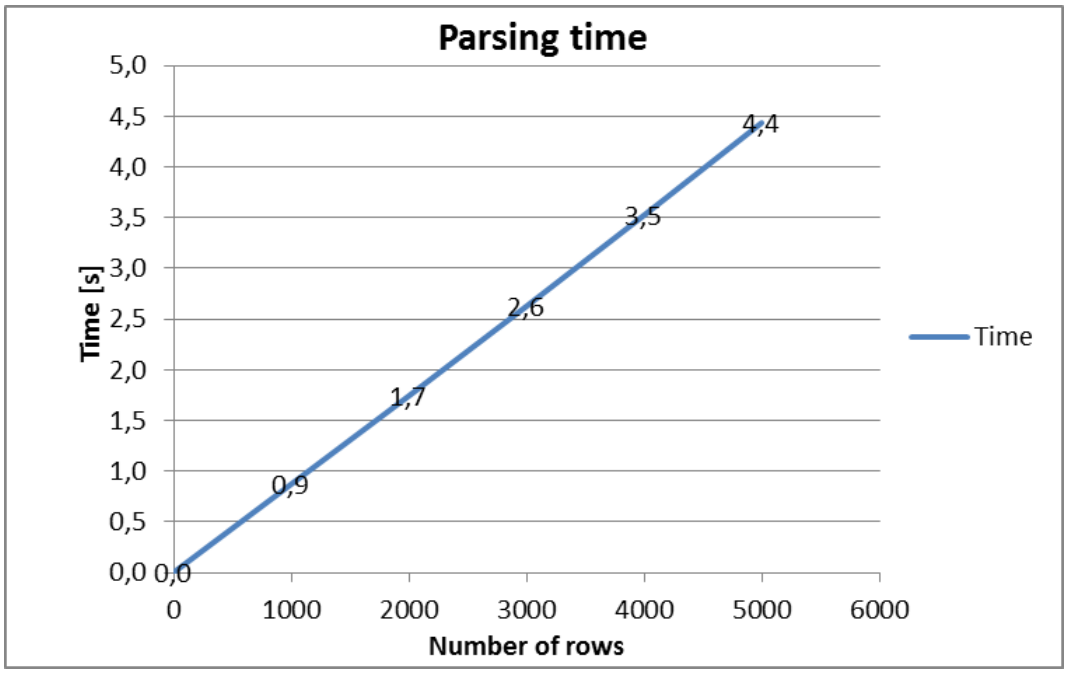


Figure 4: Linear growth of the parsing time with increasing the columns.

2.3 Evaluation of the test apps

In terms of speed query TestAppSQLite didn't show better performance than TestAppXML, both are in 20 ms range for table with 5710 rows and 53 columns. But, SQLite database has functionalities such as indexing which improves search query a lot. Also,

search queries are easier to preform using SQL language and therefore more practical to use. Further comparison is given in Table 2. As it can be seen in the Table 2, TestApp-SQLite is definitely a better solution. There are two main reasons for this statement-Java Heap usage and parsing time is acceptable when SQLite is used.

Table 2: Difference between two test apps

	Java Heap usage	Parsing 100 000 rows	Query Speed	SQL	Indexing
TestAppXML	extremely large	not possible	fast	no	no
TestAppSQLite	**very low**	**possible**	fast	**yes**	**yes**

3 SAGis mobile

System architecture

There are two main components of SAGis mobile - client and server component. Server component is consisted of Oracle database and other components which fill the database with data (AutoDesk MapGuide and SAGis web) and using the same data for visualisation (Geoportal).

The most important parts of client side are SAGis mobile app and localized SQLite database. Communication between client and server is done via Web Service (SOAP) using HTTPS protocol. Initial transfer of SQLite database can be done using USB, before going out into the field. This can be seen on Figure 5 below.

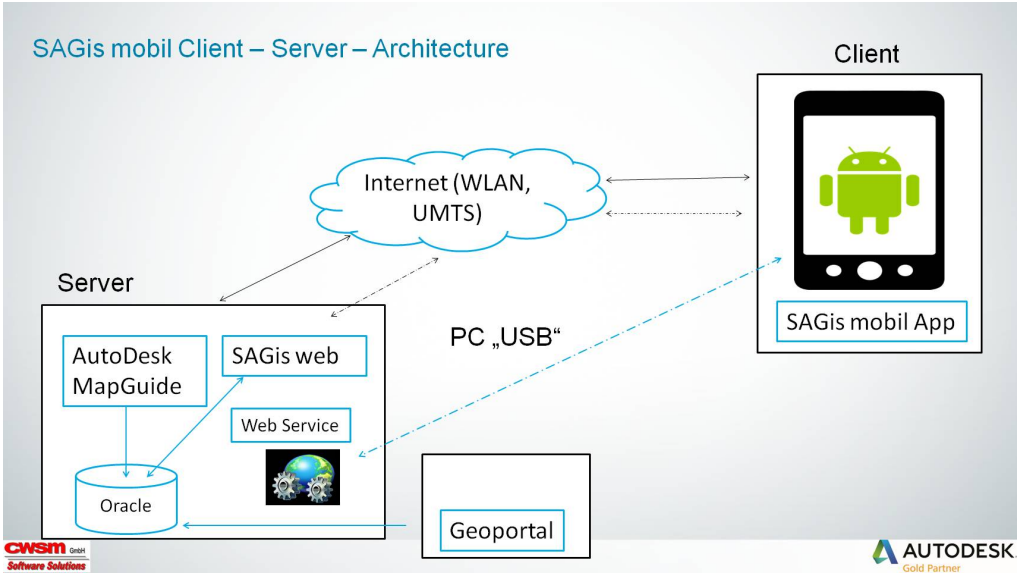


Figure 5: SAGis mobile architecture.

App functionality

SAGis mobile is, of course, only a prototype. But, some of the main functional requirements are already met. Some of the most important functionalities listed in requirement engineering document will be briefly discussed.

Firstly, the user can **load data** before going out to the field. There are several possibilities for data storage. Data can be stored either as XML and than parsed into database or it can be stored as SQLite database right away. Of course, only the data needed for one working day will be stored directly on device SD card or internal memory.

Secondly, **visualization** of data stored in SQLite database is done successfully. One of the main requirements was that dialogs in SAGis mobile should be exactly the same as those in AutoDesk Infrastructure Admin, which is the desktop solution for management of geoinformation together with AutoCAD Map 3D. Another important requirement was that GUI is dynamical. That means that GUI is generated from tables and developed completely programmatically, which was also a very demanding task.

Thirdly, one of the requirements was that user can **read**, **write** and **change** the records in the database. That basically means that DBMS has to be developed again in a way that is very similar to AutoDesk Infrastructure Administrator. GUI interface is intuitive because the controls change colors depending on whether user searches, edits or saves in database, which should help to use the app without user manual. Some components of web service are developed using kSOAP2 library which is actually third-party library for Web services on Android platform.

Moreover, the app can be used both on smartphones and tablets. For example, there is also zoom functionality which is very useful when the app is used on the smartphone, which has a relatively small screen. The app is developed and tested for devices running on Android Lollipop.

But, it is also tested on Android KitKat and almost all functionalities are working. User interface is slightly different and zoom functionality doesn't work properly. CWSM GmbH development team stated that target version should be Android Lollipop, which is actually not latest Android OS version. Android Marshmallow was already released in October 2015. These are just some of the app's main functionalities.

Figure 6 shows SAGis mobile search functionality with intuitive color change for UI items. A user can search using one or more fields including all types of UI items-TextBox, ComboBox and CheckBox.

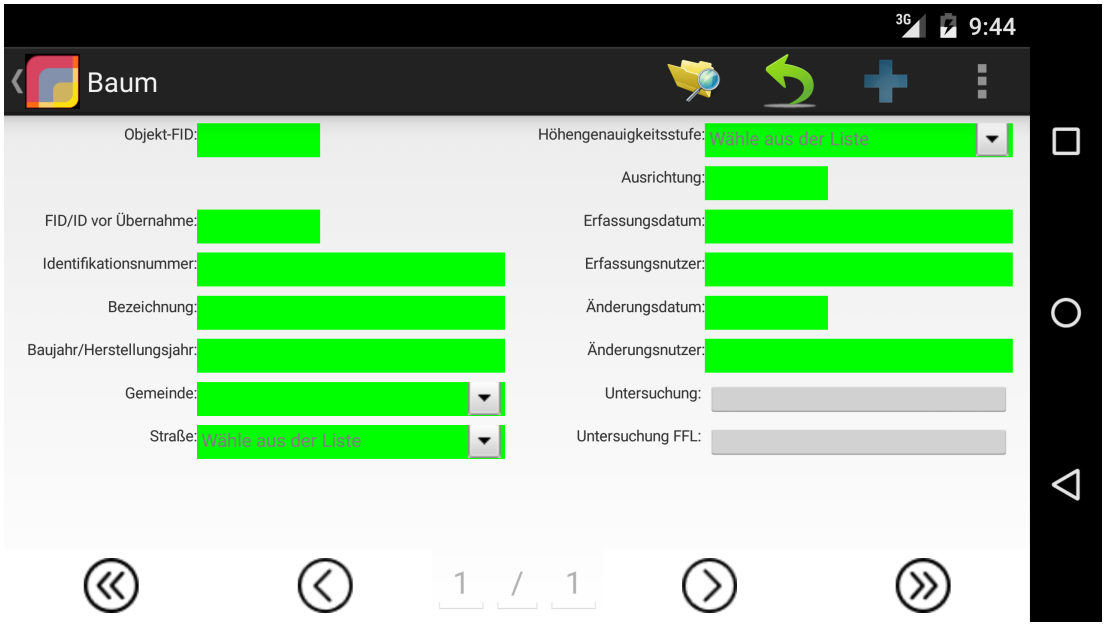


Figure 6: SAGis mobile GUI - search mode.

Figure 7 shows edit mode of the app. User interface buttons also have dynamical behaviour because action bar items change depending on the application mode. For example, when user is in edit or add mode, search button is disabled. This is done because of lack of space in action bar area in portrait mode if another button is added.

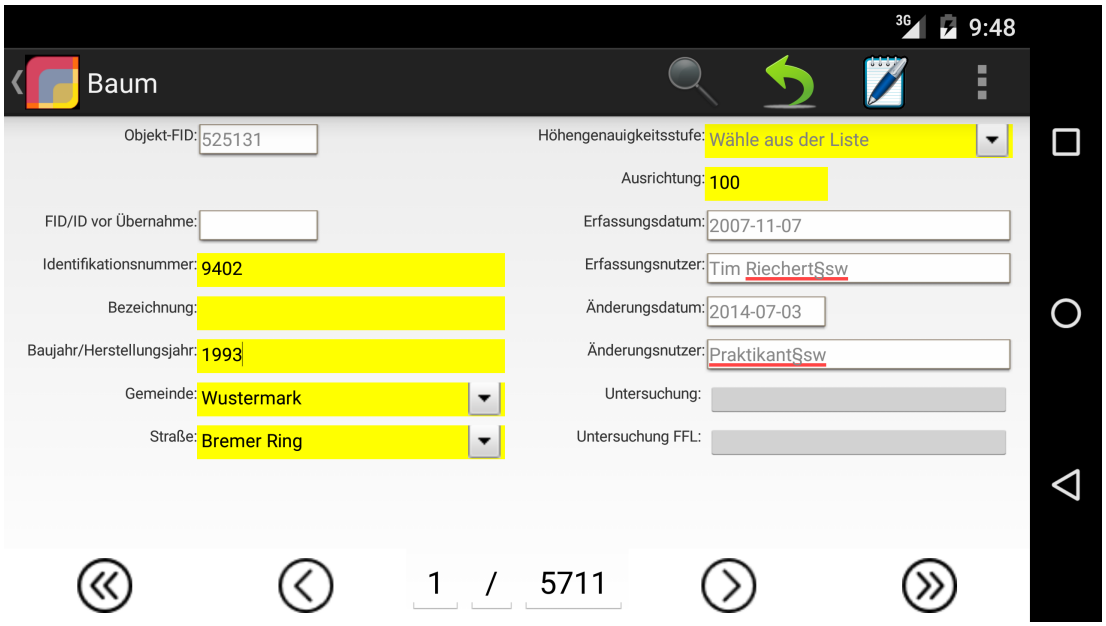
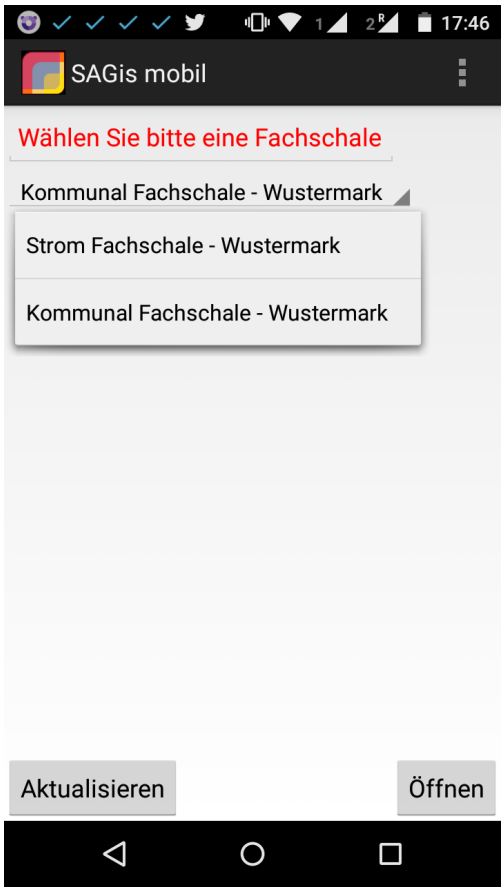


Figure 7: SAGis mobile GUI - edit mode.



On the figures above, two important dialogs of the app are shown. On the left side, start dialog of the app can be seen. It gives possibility to the user to choose between several application modules depending on the tasks which have to be executed for that particular working day. Also, it is possible to update current databases or simply open the application module. When application module is opened, dialog on the right side is displayed.

The dialog on the right side enables the user to choose one of the themes, e.g. traffic sign, info point, green surface, etc. Also, the statistics is shown on the left side of the dialog where user can check how many records he added, deleted or updated. Also, date of last update can be seen. When user clicks on one of the buttons, management of communal geoinformation can start.

4 Research results and discussion

Research showed that apps on the market already have large number of functionalities for management of spatial data, but they are not intended to store and manage 100,000 - 500,000 records with spatial context. Another problem is that existing databases, such as Oracle, used by local authority districts in Germany cannot be updated automatically during the field work if such apps are used. The reason for that is that these apps usually offer native server side technologies which have to be integrated together with the app in order to have updated state on client and server. In short, investigated apps have large number of functionalities but they couldn't meet all business and functional requirements set by CWSM GmbH.

As desirable solution on the app market is not found, app has been developed from scratch. Research is done in such a way that two test apps are developed which should show if main requirements from CWSM GmbH and local authorities districts can be met. Two test apps provided following findings:

- XML combined with HashMap and ArrayList as intermediate data structure is possible only for smaller datasets, because parsing 100 MB XML files into such structures lasted too long (23 seconds to parse 5000 rows) and Java Heap management caused runtime error for files around 300 MB
- SQLite is better solution for querying and storing tables containing 100,000 - 500,000 records compared to XML, because Java Heap management and time needed to parse XML to SQLite was not problematic
- Search queries preformed both on XML file and SQLite database lasted less than 20 ms (time needed to find last row in table with 5710 rows and 53 columns)
- Data storage option, namely external, internal storage or Assets folder, doesn't play a role when preforming the search query (in terms of query speed)
- Time needed to find last record in SQLite table with 100,000 rows and 150 columns (fully populated table) is around 5 seconds
- Overall responsiveness of the app when dealing with dataset with 5710 rows and 53 columns is under 2 seconds

These results are showing that storage and management of huge amount of spatial data on mobile device is possible. Management of cadastral data stored in 500 MB

SQLite databases (such as ALKIS databases) could be possible in the future to the certain extent. The main reason for that is the fact that processor speeds and amount of RAM on mobile devices tends to become bigger and bigger. Beside research analysis, SAGis mobile app that meet CWSM GmbH requirements is developed. App provides functionalities for adding, deleting, editing and searching some record in the database. Also, communication with web server is possible and it is realized using SOAP web service.

This Master's thesis showed that there is a huge potential in mobile devices, especially tablets, because of screen diagonal size and weight. Some components for building relatively complex Mobile GIS are available. At this point of time, it is not yet possible to compare Desktop GIS and Mobile GIS solution. Mainly, because of a complex spatial analysis aren't possible using Mobile GIS apps.

However, this shouldn't be task of Mobile GIS. Data should be acquired in the field using mobile app, some simple analysis, e.g. buffer analysis and management of geo-data should be possible. Acquired data should be transferred to server component which will do complex spatial analysis based on acquired field data.