



Donaustadt Guide



DIPLOMARBEIT

Titel: Smartphone-basierter Bezirksführer

Nummer: 22IT1101

Jahrgang: 5BHITN

Schuljahr: 2010/11

Gruppe: Christoph GWIHS, Alexander GRAFL, Patrick STIEL

Betreuer: Dr. Erwin RYBIN

ERKLÄRUNG

Die nachstehenden Autoren erklären, dass sie die vorliegende Diplomarbeit aufgrund ausschließlich selbst durchgeführter theoretischer und praktischer Arbeiten eigenständig verfasst und keine unerlaubten Hilfsmittel verwendet haben.

Datum: 16.06.11

Christoph Gwihs

Patrick Stiel

Alexander Grafl

INHALT

1	PROBLEMSTELLUNG/OBJECTIVE.....	6
1.1	Deutsch.....	6
1.2	English.....	8
2	ZUSAMMENFASSUNG/SUMMARY.....	9
2.1	Deutsch.....	9
2.2	English.....	10
3	EINLEITUNG.....	11
3.1	Motivation.....	11
3.2	Danksagung.....	12
4	TECHNISCHE GRUNDLAGEN.....	13
4.1	Android.....	13
4.1.1	Allgemein (Stiel).....	13
4.1.2	Entwicklungen (Stiel).....	13
4.1.3	Lizenz (Stiel).....	15
4.1.4	Verbreitung (Stiel).....	16
4.1.5	Konkurrenz (Stiel).....	17
4.1.6	Architektur (Gwihs).....	18
4.1.7	Portierbarkeit (Gwihs).....	24
4.1.8	Entwicklung auf Android (Stiel).....	25
4.1.9	Einschätzung (Stiel).....	26
4.2	PHP – PHP: Hypertext Preprocessor (Gwihs).....	27
4.2.1	Allgemeine Funktionsweise.....	27
4.2.2	Geschwindigkeit.....	27
4.3	Java (Grafl).....	29
4.3.1	Allgemein.....	29
4.3.2	Grundkonzept.....	30
4.3.3	Objektorientierte Programmierung.....	30
4.3.4	Klassen.....	31

4.4	JSON (Grafl).....	33
	Warum ist JSON so wichtig für unser Projekt?.....	34
4.5	MySQL (Stiel).....	35
4.5.1	phpMyAdmin.....	36
4.6	OpenStreetMap-Projekt (Stiel).....	37
5	PRAKTISCHE REALISIERUNG.....	38
5.1	Grundlegende Hard- & Softwarefunktionen von Android (Grafl).....	38
5.1.1	MapView.....	38
5.1.2	Kamera.....	40
5.1.3	HTTP	41
5.1.4	ListView.....	42
5.1.5	Aufbereitung der Daten.....	43
5.2	Programmierung der Smartphone-Applikation (Gwihs).....	45
5.2.1	Menü.....	46
5.2.2	LiveView.....	49
5.2.3	Suchfunktion.....	52
5.2.4	MapView.....	55
5.2.5	Advisor.....	57
5.3	Datenbank-API (Gwihs).....	58
5.3.1	Positions-Modus.....	58
5.3.2	Such-Modus.....	59
5.3.3	Advisor-Modus.....	60
5.3.4	Wetter-Modus.....	61
5.4	Bereitstellen der Points of Interest (Stiel).....	62
5.4.1	OpenStreetMap-Projekt.....	62
5.4.2	MySQL-Datenbank erstellen.....	64
5.4.3	Programmieren des Interpreters.....	66
6	LITERATURVERZEICHNIS.....	69
6.1	Internetverweise.....	69
7	ABBILDUNGSVERZEICHNIS.....	72
8	ABBILDUNGSQUELLENANGABE.....	73

9	ANHANG.....	74
9.1	Bedienungsanleitung.....	74

1 PROBLEMSTELLUNG/OBJECTIVE

1.1 Deutsch

Seit einiger Zeit spielt der Tourismus auch jenseits der Donau, so zum Beispiel auch in der Donaustadt, eine immer größere Rolle. Viele Touristen, aber auch Einheimische sind jedoch mit den derzeit am Markt vorhandenen Stadt- bzw. Bezirksführern unzufrieden, da diese unübersichtlich, unvollständig und umständlich in der Handhabung sind. Ebenso zeigen diese Führer oft nur einen kleinen beschränkten Ausschnitt aus dem Kulturangebot, veraltete Daten, beziehungsweise unzureichend aufbereitete Informationen.

Um dieses Problem zu lösen, entwickeln wir einen softwarebasierenden Stadt- bzw. Bezirksführer für Smartphones aufbauend auf der Android-Plattform. Der Benutzer kann ein Objekt mittels der eingebauten Handykamera identifizieren, er kann aber auch nach Objekten in der Datenbank suchen.

Die Software, die unter dem Namen *Donaustadt Guide* erstellt wird, bietet einige einzigartige, innovative Funktionen und Eigenschaften:

- Erster digitaler Bezirksführer in Wien
- Neuartige und interaktive Führung des Nutzers durch den Bezirk
- Leichter und schneller aktualisierbare Datenbasis

Diese Anwendung revolutioniert das Angebot von Tourismusführern. Dem Benutzer wird die lästige Aufgabe abgenommen, das gewünschte Ziel auf einer Karte zu suchen. Ebenfalls unterstützt der digitale Bezirksführer den User dabei, ein potentielles Ziel zu finden.

Der erste Prototyp soll bis Anfang Jänner 2011 fertig sein und auf verschiedenen Android Smartphones getestet werden. Ende März 2011 soll die fertige Anwendung voll funktionsfähig und einsatzfähig sein.

Die Anwendung wird mit der Programmiersprache JAVA realisiert und soll auf Smartphones mit einem Android Betriebssystem lauffähig sein. Als Entwicklungsumgebung wird die freie Software, Eclipse verwendet. Die Sehenswürdigkeiten werden in einer MySQL-Datenbank gespeichert. Der Zugriff auf diese Datenbank wird mittels einer eigens erstellten Schnittstelle realisiert. Die Kartendaten werden durch das OpenStreet Map Projekt bereitgestellt.

Das Programm kann auf weitere Bezirke und ganz neue Städte ausgebaut werden. Durch das modular aufgebaute System kann das Programm einfach adaptiert und an andere Städte angepasst werden. Durch die Erstellung eines kompletten Paketes (Server, Anwendung, Webinterface) kann der Bezirksführer ohne viel Einarbeitungszeit angepasst werden.

1.2 English

Recently the role of tourism beyond the Danube, for example in Donaustadt, has been increasing.

However many tourists, as well as residents, are unhappy with the current city and district guides, because they are confusing, incomplete and difficult to handle. In addition they just show a small and limited part of the culture on offer, outdated and insufficient information.

To solve this problem, we want to develop a city and district guide based on software for Smartphones using the Android operating system. The user is able to identify an object by using his built-in phone camera, as well as looking up objects in the database.

The software, which will be created under the name *Donaustadt Guide*, will offer some unique, innovative functions and properties:

- First digital district guide in Vienna
- New and interactive tour around the district
- Simple and fast up-to-date information

This application will revolutionize the format of tourist guides. The user will be spared the annoying task of looking for the desired destination on a map. In addition the digital district guide will alert the user to other potential points of interest.

The first prototype should be finished by January 2011 and be tested on several different Smartphones. The final version with all features should be finished by the end of March 2011.

The application will be built with the programming language JAVA and will run on Smartphones using the Android operating system. We will be using the freeware software Eclipse as development environment. The points of interest are stored in a MySQL database which we will be accessing via a self-built interface. The maps will be provided by the OpenStreetMap project.

The application can be extended to other districts or whole new cities. Due to the modular system of the program it could be easily adapted. By creating a complete package (servers, application, web interface) the district guide can be used in other cities with little adjustment.

2 ZUSAMMENFASSUNG/SUMMARY

2.1 Deutsch

Die Aufgabe unseres Teams „Smartphone-basierter Bezirksführer“ war es, einen herkömmlichen Stadt- oder Bezirksführer überflüssig zu machen.

Es ist uns gelungen eine Smartphone-basierte Bezirksführungsapplikation auf Android-Basis zu erstellen, welche es möglich macht die Points-of-Interests in der Umgebung des Users zu lokalisieren und Informationen zu dem gewünschten Ziel zu erhalten.

Da in den heutigen Smartphones bereits ein moderner GPS-Empfänger integriert ist, verwendeten wir diesen für die Positionsbestimmung. Für die Bestimmung der Himmelsrichtung verwendeten wir den eingebauten Kompass und zur Lagebestimmung im Raum ein Gyroskop.

Das Kartenmaterial, welches wir zur Synchronisation mit dem GPS-System benutzten, bezogen wir aus dem OpenStreetMap-Projekt, welches das Material unter einer Open-Source-Lizenz in einem xml-Format bereit stellt.

Anschließend erstellten wir eine MySQL-Datenbank, die auf einem Server gehostet wird. Diese arbeitet im Hintergrund und liefert allgemeine Informationen, wie Öffnungszeiten und Adresse der gewünschten Ziele.

Abschließend fügten wir eine Map-View hinzu, welche mittels Symbolen Tankstellen, Einkaufsmöglichkeiten und weitere Kategorien anzeigt. Zudem wurde ein Ratgeber (Advisor) integriert, der dem Benutzer z.B.: bei der Auswahl eines Restaurants hilft.

2.2 English

The task of our team “smart-phone-based district guide” was to replace conventional city guides.

We created an smart-phone-based application based on the Android platform. This application allows the user to locate and get information on points of interest in the surrounding area.

Because today's smart-phones already have built-in GPS-recievers we used them to get the location of the user. To determinate the point of the compass we used the built-in compass and to determinate the horizontal and vertical position of the smart-phone in space we used a gyroscope.

We took the geographical data from the OpenStreetMap-Project. This data is provided under an open-source license and in xml-format.

Afterwards we created a MySQL-database which is hosted on a server. The database works in the background and provides general information, for example opening-hours or the address of the location.

Finally we added a map view which displays gas stations, shopping facilities and other categories with symbols. Additionally we integrated an advisor to help the user e.g. to choose a restaurant.

3 EINLEITUNG

3.1 Motivation

Jeder kennt das, wenn man in einer fremden Stadt ist, in der man noch nie war, mit einem Stadtplan, der nur auf umständlichste Weise entfaltet und „bedient“ werden kann, und hat keine Ahnung, wo man sich befindet. Mit einem konventionellen Stadtführer kann man sich ja leider nur sehr schlecht orientieren und was ist, wenn plötzlich eine Straße nicht mehr vorhanden ist oder man vor einem Gebäude steht, welches in dem Plan noch gar nicht eingezeichnet ist? Ein weiterer Kritikpunkt unserer Seite an herkömmlichen Stadtführern ist auch, dass man nicht einfach nach einem gewünschten Ziel suchen kann oder nach Kategorien einteilen und somit leichter eine passende Destination finden kann.

Um diese Problematik lösen zu können, haben wir uns entschlossen eine Smartphone-App zu programmieren, welche die Genauigkeit eines herkömmlichen Stadtführers mit der Aktualität und der universellen Einsetzbarkeit des Internets verbindet. Zudem wollten wir etwas Kreatives und Innovatives auf den Markt bringen, was auch nach unserer Schulzeit die Bestimmung hat, weiter zu existieren.

3.2 Danksagung

Wir möchten uns bei unseren vielen Unterstützern bedanken, die uns mit Rat und Tat zu Seite gestanden sind:

- Dr. Erwin Rybin, der uns immer geholfen hat, falls wir in unserem Projekt Schwierigkeiten hatten, der unser Projekt bei verschiedenen Wettbewerben eingereicht hat oder versucht hat das Projekt bestmöglich voranzutreiben, um es auch nach unserem Schulabgang weiter bestehen lassen zu können.
- Firma Plansinn mit welcher wir seit dem Beginn des Jahres eine Kooperation anstreben
- Firmenkonzern REWE hat uns mit diversem Bild- und Informationsmaterial unterstützt.

4 TECHNISCHE GRUNDLAGEN

4.1 Android

4.1.1 Allgemein (Stiel)

Android ist eine Entwicklung von Google. Es ist ein Betriebssystem für mobile Endgeräte und besteht neben dem Betriebssystem auch aus einer Middleware und Schlüsselanwendungen. Android ist komplett in Java programmiert und deshalb universell einsetzbar.

Außerdem ist Android in der Lage umfangreiche Funktionen für die verschiedensten Geräte zur Verfügung zu stellen und dem Entwickler sind hierbei keine Grenzen gesetzt. Es ist auch vollkommen gleich für welches mobile Endgerät mit Android entwickelt werden soll, denn es werden alle Geräte unterstützt, die in der Lage sind den Android Code auszuführen. Das können Mobiltelefone, GPS Empfänger, Smartphones, mobile Computer oder auch Navigationssysteme sein. [1]

4.1.2 Entwicklungen (Stiel)

Die erste Version, nämlich die Version 1.1 veröffentlichte Android am 10. Februar 2009, welche das Speichern von MMS-Anhängen ermöglichte. Multimedia Messaging Service oder kurz MMS ist einfach eine Erweiterung von SMS (Short Message Service) anzusehen und bietet die Möglichkeit multimediale Nachrichten an mobile Endgeräte zu versenden. Diese Version wird aber nicht mehr unterstützt.

Ab der zweiten Version (1.5), welche am 30. April 2009 erschien, erhielten die Versionen neben der Versionsnummer zusätzlich den Namen einer Süßspeise. So heißt die Version 1.5 auch „Cupcake“. Diese Version ermöglichte den automatischen Wechsel zwischen Hoch- und Querformat auch eine Bildschirm-Tastatur. Außerdem konnte der User Videos aufnehmen bzw. wiedergeben, sich automatisch mit Bluetooth Verbinden und Stereo für Bluetooth wurde ermöglicht. Zusätzlich wurden neben Englisch und Deutsch noch weitere Sprachen hinzugefügt.

Die Version 1.6 oder „Donut“ erschien am 15. September 2009 und ermöglichte den Benutzern Virtual Private Networks zu konfigurieren. Außerdem konnte der User mit der neuen, differenzierten Energieverbrauchssteuerung sehen, welche Applikationen zu viel Strom verbrauchen und diese gegebenenfalls stoppen oder löschen.

Weiters wurde eine quellenübergreifende und selbstoptimierende Suchfunktion mitgeliefert, sowie eine Gestenerkennung und eine Sprachsynthese (Text-to-Speech), welche die menschliche Stimme künstlich erzeugt. Zusätzlich gab es auch noch mehrere Bildschirmauflösungen zwischen der man wählen konnte.

Die Version 2.0 oder „Eclair“ (Liebesknochen) wurde am 26. Oktober 2009 veröffentlicht und beinhaltet neben dem Digitalzoom auch die Unterstützung von Blitzlicht in der Kamerafunktion. Zusätzlich wird nun auch der Microsoft Exchange Server unterstützt, welcher es erlaubt Intranets aufzubauen, sowie E-Mails zu verwalten bzw. zu filtern. Außerdem wurde die neuere Bluetooth Version 2.1 eingefügt.

Am 12. Januar 2010 wurde die Version 2.1, die ebenfalls „Eclair“ heißt, veröffentlicht. Diese ermöglichte es Benutzern animierte Hintergrundbilder zu verwenden. Weiters erhielt die Software eine Erweiterung des Webkits, welcher HTML5 unterstützt, sowie einen neuen WebStorage, welcher es erlaubt auf Datenbanken zuzugreifen bzw. etwas zu speichern. Zusätzlich wurden sogenannte Geolocation Permissions-Methoden hinzugefügt, um die Berechtigung über die Geolocations zu erhalten und diese dann auf den Web-View zu legen.

Die Version 2.2 oder „Froyo“ (Frozen Yogurt), welche am 20. Mai 2010 veröffentlicht wurde, erhielt den Linux-Kernel 2.6.32, da dieser weniger Arbeitsspeicher benötigt. Damit wurde es auch möglich Arbeitsspeicher zu verwenden, die größer als 256 MByte sind. Zusätzlich wurde ein Just-in-Time-Compiler zur neuen Version hinzugefügt, welcher es ermöglicht während der Laufzeit in Maschinen-Code zu übersetzen. Dies führt zu einer wesentlichen Performancesteigerung gegenüber herkömmlichen Interpretern. Zusätzlich wird auch die Flash-Version 10.1 unterstützt und OpenGL ES 2.0 wurde erweitert. Mittels Tethering wurde es auch möglich via Smartphone sich mit einem PC zu verbinden, um diesen eine Internetverbindung über GSM/UMTS zu ermöglichen. Weiters wurde es mit der SD-Karte App2SD möglich Apps auf der SD-Karte zu speichern. Zusätzlich wurde ein Android Cloud to Device Messaging (C2DM) Service entwickelt, welches es den Entwicklern ermöglicht Daten von Servern zu ihren Android Endgeräten zu senden.

Am 6. Dezember 2010 wurde die Version 2.3 „Gingerbread“ (Lebkuchen) veröffentlicht, dazu wurde wieder ein neuer Linux-Kernel hinzugefügt, um den Arbeitsspeicher zu verringern, und nun wird auch HTML5 Audio sowie Google TV unterstützt. Zusätzlich wird auch WebM unterstützt.

Dieses Projekt wurde von Google initiiert und wird von der Mozilla Foundation sowie von der Opera Software unterstützt. Weiters erhält der User ab dieser Version die Möglichkeit mittels NFC (Near Field Communication) Daten über kurze Strecken kontaktlos auszutauschen. Es wurde aber auch die Parallele Garbage Collection eingebaut, welche für eine ruckelfreie Animation sorgen soll sowie die verbesserte Integration von sozialen Netzwerken. Außerdem kann Android nun auf die Unterstützung von Gyroskopen, aber auch auf das Ext4-Dateisystem zurückgreifen. Es wurde aber auch ein SIP-Client für VoIP sowie ein Downloadmanager integriert.

Die Version 2.3.3 heißt ebenfalls „Gingerbread“, da die Änderungen nur minimal zur Vorgängerversion 2.3 sind. Hauptsächlich wird die Unterstützung von Dualcore-Prozessoren und NFC-Chips auch Apps zur Verfügung stehen und nicht nur dem Betriebssystem. Zusätzlich wurden noch kleinere Verbesserungsmaßnahmen vorgenommen wie zum Beispiel eine bessere Bluetooth-Unterstützung.

Kurz danach wurde die Version 3.0 „Honeycomb“ (Bienenwabe) ins Leben gerufen, welche aber nur für Tablets erschienen ist, abgesehen von einigen Hacks. Diese bietet eine benutzerfreundlichere Oberfläche und unterstützt Google Talk mit Videotelefonie.

Die beiden Versionen (2.3.3, 3.0) sollen dann in der demnächst erscheinenden Version 4.0 „Ice Cream“ zusammengeführt werden. [2]

4.1.3 Lizenz (Stiel)

Das Android Open Source Project setzt mit der Apache-2.0-Lizenz bevorzugt auf eine Lizenz, welche von der Free Software Foundation als kompatibel zur GNU General Public License gesehen wird. Google verspricht sich aus diesem Versuch des Open Source Projects klare Vorteile, indem sie eine breite Entwickler-Community ansprechen. Durch die Quelloffenheit soll auch die Weiterentwicklung des Systems beschleunigt werden. Google betont auch, dass die Open-Source-Plattform wesentliche Zeit- und Kostenersparnisse auf dem Weg zur Markteinführung von Mobilgeräten bedeutet. Dieser Schritt bedeutet also auch, schnell eine größere Vielfalt an Android-Geräten auf den Markt zu bringen. [3][4]

4.1.4 Verbreitung (Stiel)

US-amerikanische Marktforscher von der NPD Group stellten in einer Grafik fest, dass die Verbreitung im ersten Quartal 2010 von Android Geräten in den USA stark steigt. Laut Google werden zu diesem Zeitpunkt jeden Tag 65.000 mobile Geräte mit dem Android OS neu registriert.

Die Verkaufszahlen von Windows, Palm und RIM sind weiterhin sinkend. Apple hat wohl mit dem iPhone 3GS eine Marktsättigung erreicht und deshalb wird in der Zukunft sicherlich Android auf den meisten Telefonen der Welt zum Einsatz kommen.

Außerdem stellten Marktforscher im zweiten Quartal 2010 erstmals fest, dass mehr Smartphones mit dem Betriebssystem Android verkauft wurden, als jene Smartphones mit dem Betriebssystem iOS von Apple. Siehe Abbildung 1: Verbreitung von Android

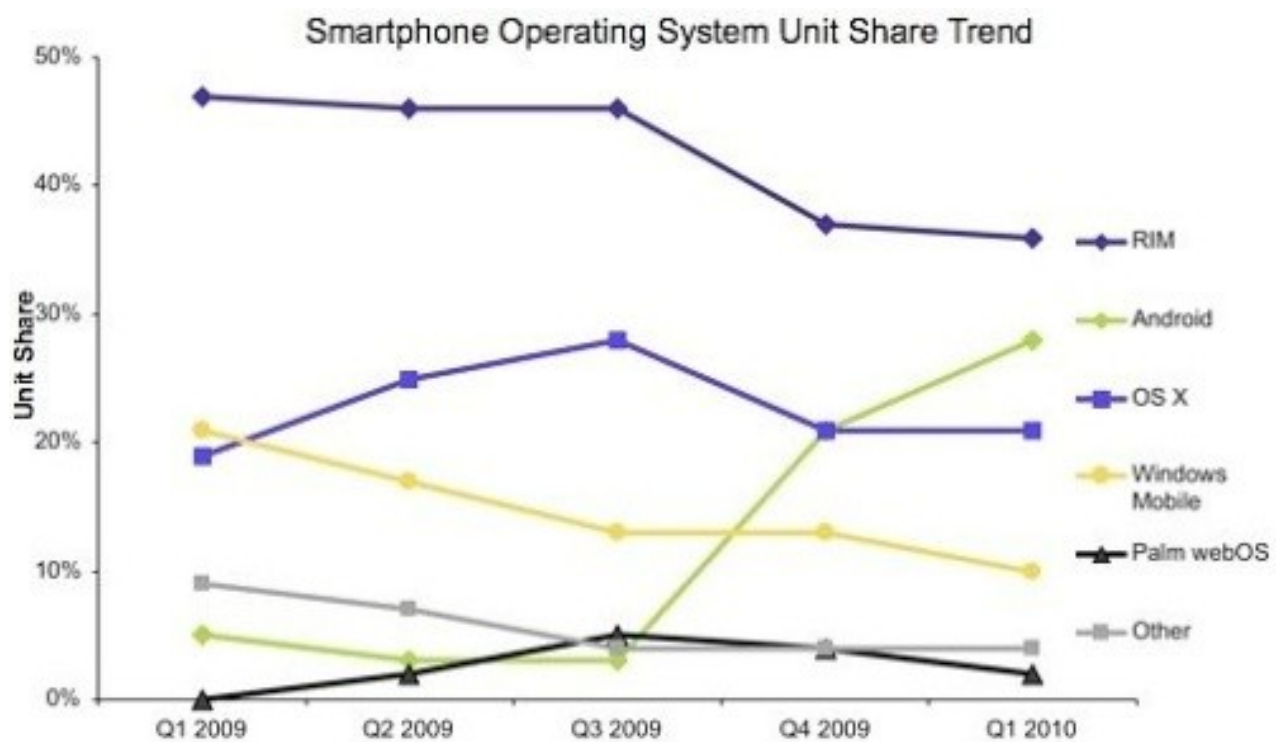


Abbildung 1: Verbreitung von Android

4.1.5 Konkurrenz (Stiel)

Die Konkurrenten des Android OS sind neben Apples iOS auch Microsofts Windows Phone 7 und Nokias Symbian OS. Apple hat schon Mitte Februar verkündet, dass sie das iPhone nano entwickeln, um den derzeitigen Marktanteil von 16 Prozent wieder in die Höhe zu schrauben und Android Konkurrenz zu machen.

Im Smartphone-Bereich hatte Microsoft bisher immer das Nachsehen, denn Apple und Google legten mit ihren Betriebssystem für Smartphones einiges vor. Jetzt will Microsoft wieder zu seinen Konkurrenten aufschließen und bringt das Windows Phone 7 auf den Markt. [6]

4.1.6 Architektur (Gwihs)

4.1.6.1 Aufbau

Im Grundlegenden besteht die Android Plattform aus 4 Schichten. Die Grundlage des Android Betriebssystems bildet der Android Kernel. Dieser baut in der derzeitigen Version auf einen abgespeckten, auf die Anforderungen eines Mobilgerätes reduzierten, Linux 2.6 Kernels auf.

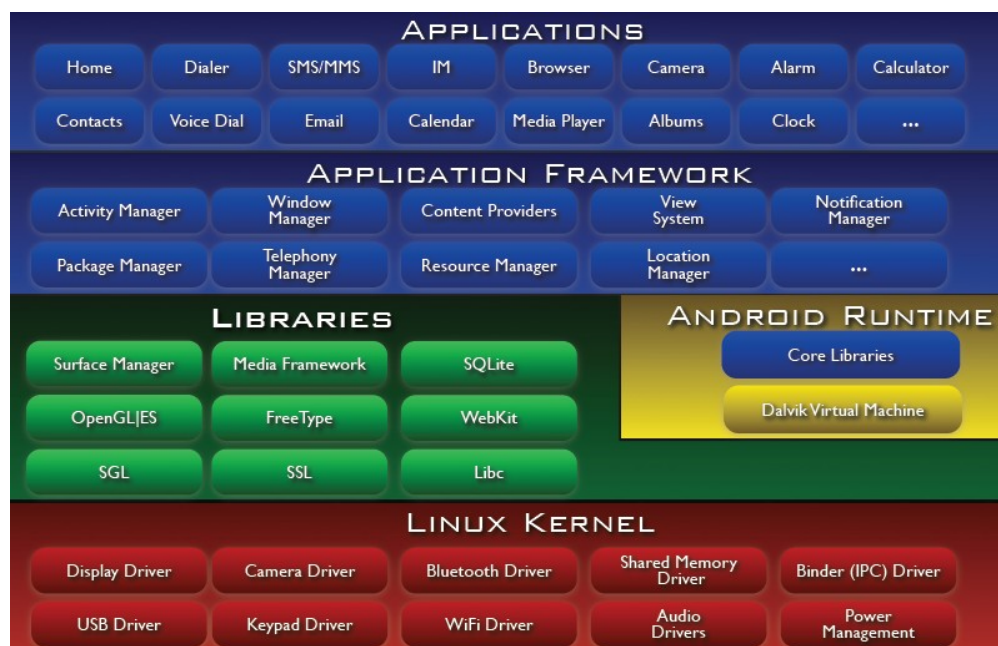


Abbildung 2: Architektur von Android

Auf diesen Kernel aufbauend existieren mehrere Bibliotheken. Diese stellen sowohl Standard-Java-Funktionen als auch verschiedene androidspezifische Bibliotheken zur Verfügung. In der selben Ebene befindet sich die sogenannte Android-Runtime. Diese stellt die Dalvik Virtual Maschine bereit, welche die in Java geschriebenen Anwendungen ausführt.

Auf der nächsten Ebene liegt, das Application Framework. Diese bildet den Rahmen in dem Anwendungen ausgeführt werden. Dieser steuert den sogenannten Lifecycle. Ebenso beinhaltet das Application Framework den Window Manager, den Content Provider und verschiedene Manager, die zum Beispiel den Ortungsdienst oder den Notificationdienst bereit stellen.

Auf der letzten Ebene laufen schließlich die eigentlichen Anwendungen ab. Diese werden normalerweise in Java geschrieben. In Android bilden sowohl die Telefonfunktion, die Nachrichtenfunktion als auch die Telefonbuchfunktion selbstständige Anwendungen. Durch diesen Umstand können diese Funktionen durch beliebige andere, auch von Drittherstellern programmierten, Applikationen wahrgenommen werden.[7]

4.1.6.2 Linux-Kernel

Der Kernel basiert auf einem Linux 2.6 Kernel. Dieser wurde durch verschiedene Anpassungen an der Speicher- und Prozessverwaltung an die Umgebung eines Smartphones angepasst. Teile die für die Verwendung auf einem Mobilendgerät unwichtig sind wurden entfernt. Eine weitere Einschränkung zu einem normalen Linux Kernel liegt in der fehlenden Unterstützung eines nativen Window-Systems und der nicht vollständig enthaltenden Standard Linux Tools.

Der Kernel übernimmt die Steuerung des Displays, die verschiedenen Eingabefunktionen, die Audioausgabe sowie die Audioeingabe. Ebenso steuert der Kernel die Netzwerkverbindung sowohl über Mobilfunk als auch über WLAN.

Ein weiteres Aufgabengebiet des Android Kernels liegt in der Energieverwaltung. Dies spielt auf einem Smartphone eine besondere große Rolle. Die Verwaltung baut zwar auf dem Standard Linux Power Management auf, wurde aber durch viel strengere Stromspareinstellungen ergänzt. Durch sogenannte „wake locks“ können unterschiedliche Komponenten den Kernel dazu bringen aus dem Stromsparmodus auf zu wachen. [8]

4.1.6.3 Libraries & Android-Runtime

Die Libraries der Android Plattform beinhalten neben den Standard Java Bibliotheken noch androidspezifische Bibliotheken. Insgesamt umfasst die Android Plattform derzeit 1448 unterschiedliche Klassen. Die Android Bibliotheken an sich umfassen circa 500 Klassen welche fast ausschließlich von Google entwickelt wurden. Weiters enthalten die Libraries noch Klassen von verschiedenen Apache Projekten und vielen weiteren kleineren Bibliotheken die zum Beispiel die Konvertierung in das JSON-Datenformat ermöglicht.

Ebenso stellen die Android Libraries, die Webkit Engine bereit. Dies ist eine HTML-Rendering-Engine, welche zur Darstellung sowohl im integrierten Browser als auch in beliebigen Anwendungen integriert werden kann. Die Webkit-Engine ist der Opensource Teil des von Apple entwickelten Safari Browser. Die Ursprünge der Webkit-Engine finden sich jedoch in dem KHTML-Code des KDE Linux Projektes. Die Webkit-Engine wird nicht nur zur Darstellung von Webseiten benutzt, sondern auch zur leichteren und formatierten Darstellung von Inhalten in Anwendungen. Diese Möglichkeit nimmt mit der fortschreitenden Entwicklung des HTML 5 – Standards immer mehr Reiz für Entwickler an.

Der andere Teil dieser Ebene ist die Android Runtime. Sie beinhaltet einen der wichtigsten Teile der Android Plattform, die Dalvik Virtual Maschine. [7]

4.1.6.4 Dalvik Virtual Maschine

Die Dalvik Virtual Maschine ist eine Art von einer Java Virtual Maschine. Diese virtuelle Maschine ermöglicht, dass Anwendungen portierbar sind und auf einer großen Anzahl von verschiedenen Geräten ausgeführt werden.

Die Dalvik Virtual Maschine ermöglicht das Ausführen von für Android optimierten Programmen im .dex Format und von Dalvik bytecode. Programme werden wie normale Standard-Java Programme zu erst in .jar Files konvertiert, aber dann anschließend in das .dex Format übersetzt.

Während des Entwicklungsprozesses entschied man sich, statt der Standard-Java-Virtual-Maschine, eine eigenen, an die Bedürfnisse und Erfordernisse einer Mobile Plattform angepasste Virtual Maschine zu entwickeln. Die Dalvik Virtual Maschine nutzt einen hoch CPU-optimierten bytecode Interpreter, ebenso verwaltete sie den Speicher während der Ausführung der Anwendungen sehr effizient. Die Virtual Maschine wurde weiters an die meist schwachen Smartphone Prozessoren angepasst.

Derzeit befindet sich Google in einem Rechtsstreit mit Oracle, Inhaber der Java-Patente (ehemals Sun Microsystem). Oracle wirft Google mit der Entwicklung von Android vor allem bei der Dalvik Virtual Maschine Lizenzverstöße vor. Dieser recht harte Schritt von Oracle wird von der Java-Community kritisch bewertet.[8]

4.1.6.5 Application Framework

Das Android Application Framework regelt den kompletten Life Cycle, vom Start der Applikation, bis zu deren Beendigung. Ebenso stellt das Application Framework verschiedenen APIs zum Zugriff auf tiefere Hardware-Ebenen über sogenannte Manager Objects bereit. Diese werden zum Beispiel für den Zugriff auf den Lage- beziehungsweise auf den GPS-Sensor verwendet.

Jede Applikation in Android besteht zumeist aus mehreren Activities. Jede Activity übernimmt die Verwaltung eines Layouts, also eine für den Benutzer sichtbare Oberfläche, diese wird zur Laufzeit an die Activity gebunden. [8][9]

Jede Activity durchläuft während ihrer Ausführung mehrere Phasen.

Jede gestartete Applikation startet mittels der onCreate() Funktion. Diese Funktion wird ausgeführt

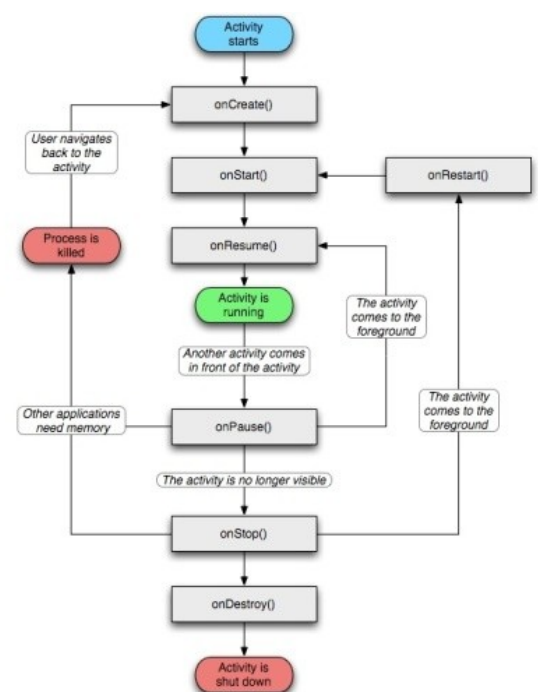


Abbildung 3: Activity Lifecycle

nachdem der Benutzer beziehungsweise das Betriebssystem eine Anwendung startet. In dieser Funktionen werden die verschiedenen Komponenten initialisiert, das im XML-Format vorliegende Design an die Activity gebunden und gegeben falls Variable initialisiert.

Anschließend nach dem die `onCreate()` Methode erfolgreich ausgeführt wurde, wird die `onStart()` Funktion ausgeführt. Diese Funktion wird aufgerufen, wenn die eigentliche Applikation für den Nutzer sichtbar wird. Nach dem Aufruf dieser Funktion wird die Applikation ausgeführt.

Doch die Anwendung kann durch verschiedenen Ereignisse, zum Beispiel durch Anruf und Benutzeraktionen unterbrochen werden. Durch diese Ereignisse werden verschiedene Methoden in der Activity aufgerufen.

Wenn die Activity durch eine andere Applikation unterbrochen wird, wird die Methode `onPause()` aufgerufen. Mittels dieser Funktion wird es der Applikation ermöglicht, bevor die andere Applikation die aktuelle Activity verdeckt, noch Abläufe, die CPU Zeit kosten, zu beenden. Dies können zum Beispiel Threads sein, oder die Verwendung von Location Listener oder sonstigen Sensoren.

Die `onPause()` Methode wird entweder von der `onStop()` oder von der `onResume()` Funktion gefolgt. Die `onResume()` Methode wird aufgerufen, wenn der User wieder mit der Activity interagieren will.

Im Gegensatz zu der `onStop()` Funktion, welche aktiviert wird, wenn die Activity für den Benutzer nicht mehr sichtbar ist. Nachdem die `onStop()` Methode erfolgreich ausgeführt wurde, kann diese entweder von der `onDestroy()`, wenn die Activity temporär, zum Beispiel wegen Arbeitsspeichermangel, automatisch vom System aus dem Speicher gelöscht wird, oder von der `onRestart()` Funktion gefolgt werden. Die `onResume` Funktion ermöglicht es der Applikation zum Beispiel vom GPS-Sensor Positions wieder Updates anzufordern. [10]

4.1.6.6 Applikationen

Applikationen bestehen im Grundlegenden aus zwei verschiedenen Arten von Dateien. Von den Ressource-Files und den Code-Files. Die Ressource-Files gliedern sich wiederum in verschiedenen Unterkategorien.

Der Programm-Code der Applikation wird in Android üblicherweise in Java geschrieben. Es gibt aber auch verschiedene Projekte die das Programmieren in C++ ermöglichen soll, diese befinden sich aber noch in einem ziemlich frühen Entwicklungsstadium.

Ein wichtiger Aspekt bei der Androide-Programmierung ist, dass der Code und das Layout der Applikation meistens getrennt sind. Man kann zwar das Layout in den Code verlagern, dies wird aber in der Android-Community nicht als „schöne“ Programmierung angesehen und führt auch zu einer unnötigen Komplizierung des Quellcodes. Anstatt das Layout in den Quellcode zu integrieren, verwendet man in der Androide-Programmierung sogenannte Layout-Files. In diesen Layout-Files, welche im XML-Format formatiert sind, werden verschiedene GUI-Elemente, wie Buttons oder Views eingebaut.

Eine Android-Eigenheit ist die komplette View basierte GUI-Gestaltung. Jedes grafische Element in Android ist eine View. Eine View muss zumindest eine ID besitzen welche im gesamten Programm einzigartig sein muss. Durch diese ID kann die View in der Applikation angesprochen werden und einem lokalen Objekt zugewiesen werden.

```
Button Button1 = (Button) findViewById(R.id.button1);
```

Da in Android alle grafischen Objekte aus Views bestehen, kann man sie verschachteln und dadurch sehr komplexe grafische Oberflächen erstellen. Zur einfachen Positionierung von Objekten innerhalb eines Layouts gibt es verschiedene Arten von Layouts.

- Relativ-Layout

Position der Objekte wird relativ zu den anderen Objekten angegeben. Dies ermöglicht die Erstellung von sehr dynamischen und flexiblen Layouts

- Absolut-Layout

Die Position von Objekten wird absolut angegeben. Dies bedeutet, dass der genaue Abstand vom oberen und unteren Layoutrand angegeben wird.

- Linear-Layout

Die Objekte werden je nach Einstellung untereinander oder nebeneinander eingereiht. Dies benötigt keine zusätzlichen Distanz- oder Positionsangaben.

- Table-Layout

Die Objekte werden in einer Art Tabelle angezeigt. Diese Art der Darstellung eignet sich besonders für die Anzeige von vielen verschiedenen Werten.

[11]

4.1.7 Portierbarkeit (Gwihs)

Da durch die Android Dalvik Maschine eine Abstraktionsschicht geschaffen wurde, können die Applikationen, die für ein Android Smartphone Modell geschrieben wurden, auch auf einer großen Anzahl an anderen Geräten betrieben werden.

Eingeschränkt wird die Portierbarkeit nur durch bestimmte Anforderungen, die die Applikation an die Hardware des Smartphones stellt. Dies könnten zum Beispiel erforderliche Sensoren sein zum Beispiel:

- Lagesensor
- Positionssensor
- Touchscreen
- Netzwerkverbindung
- Größe des Arbeitsspeicher

Durch eine hohe Anforderungen an die Hardware eines Smartphones wird die Anzahl an möglichen Kunden stark eingeschränkt. Eine weitere Einschränkung erfolgt durch die Begrenzung auf Plattformen. Durch die ständige Weiterentwicklung der Android Plattformen, werden gesammelte Änderungen in bestimmten Versionen zusammengefasst. Doch nicht alle Mobiltelefone erhalten diese Updates auf die aktuelle Version. Dadurch ergeben sich große Kompatibilitätsprobleme, da manche Applikationen die Vorteile von neueren Versionen nutzen können, diese auf älteren Versionen der Android Plattform nicht mehr verwendet werden.

4.1.8 Entwicklung auf Android (Stiel)

Auf die einfache Entwicklung von Applikationen für die Android Plattform wurde ganz besonders Rücksicht genommen. Durch verschiedene Tools und Programme wird die Entwicklung stark vereinfacht und ermöglicht eine schnelle Entwicklung. Eine SDK, die von der offiziellen Website heruntergeladen werden kann, bringt bereits einen Compiler, die verschiedenen Libraries und einen Emulator mit.

4.1.8.1 ADT

Die Android Development Tools(ADT) sind in einem Plugin für Eclipse, eine kostenlose Entwicklungsumgebung für eine große Zahl von Programmiersprachen und Plattformen, zusammengefasst. Mittels ADT können innerhalb von Eclipse Android Projekte erstellt werden, debuggt, compiliert und auf dem Emulator ausgeführt werden.

Ebenso ermöglicht das ADT das Exportieren von signierten, mittels einem eigens für jeden Entwickler erstellten Zertifikats, oder auch unsignierten Anwendungen, zum Testen der Applikation. Das ADT wird laufend weiter entwickelt und Updates automatisch über den integrierten Updatemechanismus von Eclipse installiert. [12]

4.1.8.2 ADB

Die Android Developer Bridge (ADB) ist ein Tool von Google, welches mit dem Software Developer Kit (SDK) mitgeliefert wird. Die ADB besteht aus drei Komponenten:

- Einem Client, welcher auf dem Smartphone oder auf dem Emulator läuft.
- Einem Server, welcher als Hintergrundprozess auf dem Smartphone oder auf dem Emulator läuft.
- Einem Daemon, der als Hintergrundprozess auf dem Smartphone oder auf dem Emulator läuft.

Mit der ADB kann man unter anderem Softwareinstallationen auf dem Smartphone direkt vom Computer aus abwickeln. Ebenfalls unterstützt es Debugging, mit dem man Programme testen kann, sowie die Erstellung von Screenshots auch ohne root-Zugriff. Außerdem ist es möglich, interne Werte auszulesen. [13]

4.1.8.3 Emulator

Das Android SDK enthält einen virtuellen Emulator, der so aussieht wie ein Handy. Auf diesem Emulator kann man Android-Anwendungen entwickeln und testen, ohne ein physikalisches Gerät.

Wenn der Emulator läuft, kann man mit dem emulierten Gerät so interagieren, wie mit einem normalen Smartphone. Dieser Emulator ermöglicht es die Android-Anwendungen schnell zu testen, ohne einen großen Aufwand betreiben zu müssen. Außerdem kann man zu den verschiedenen virtuellen devices die Hardware selbstständig konfigurieren und unter anderem auch darüber entscheiden, ob das device eine Kamera besitzen soll oder mit dem Touchscreen bedienbar sein soll. [14]

4.1.8.4 Vertriebsmöglichkeiten

Um Spiele oder andere Applikationen auf den mobilen Geräten zu vertreiben, wird die von Google entwickelte Software „Android Market“ schon mit dem Betriebssystem ausgeliefert. Auf diesem Android Market wird eine große Anzahl kostenloser als auch kostenpflichtiger Programme angeboten. Es überwiegt aber der Anteil an kostenlosen Applikationen. [15]

4.1.9 Einschätzung (Stiel)

Zu Beginn der Entwicklung wurde Android von vielen Leuten nur belächelt, doch in letzter Zeit spielt Android in der Mobilfunkbranche eine immer größere Rolle. Android hat sich von einem Nischenprodukt zu einem Globalplayer und Cash-Cow entwickelt. Immer mehr Mobiltelefone verwenden das Android Betriebssystem. Der Kampf zwischen Apple und Android um die Führungsposition bei mobilen Betriebssystem ist noch nicht entschieden. Die weitere Entwicklung wird sich in den nächsten Jahren zeigen. Eine Koexistenz zwischen diesen Betriebssystemen ist eine weiteres und ziemlich wahrscheinliches Szenario.

4.2 PHP – PHP: Hypertext Preprocessor (Gwihs)

PHP ist eine Skriptsprache, welche eine ähnliche Syntax wie die Programmiersprachen C oder Perl verwendet. Ursprünglich wurde PHP als Ersatz von einer Sammlung von Perl-Skripten verwendet. Doch bald darauf entwickelte dessen Erfinder, Rasmus Lerdorf daraus eine eigene Skriptsprache, welche auch noch heute, 2011, unter dem Namen PHP weiterentwickelt wird. PHP liegt derzeit in der stabilen Version 5.3.6 vor. [16]

4.2.1 Allgemeine Funktionsweise

PHP baut auf einem System auf welches den Code serverseitig ausführt, das bedeutet, dass die Codeausführung zwar von einem Client veranlasst wird, aber auf dem Server ausgeführt wird. Siehe Abbildung 4. Weiters wird der Code in den meisten Fällen nicht direkt von dem Webserver-Prozess

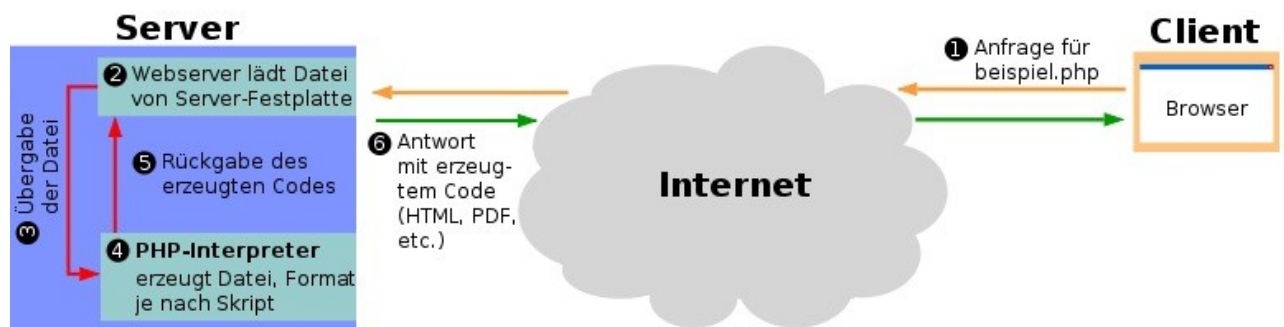


Abbildung 4: Ablauf nachdem Aufruf einer PHP-Datei

ausgeführt sondern der Webserver übergibt dem PHP-Interpreter eine Referenz auf die auszuführende PHP-Datei und die zugehörigen Parameter, zum Beispiel Start-Variablen. Der PHP Interpreter interpretiert die PHP – Datei und wandelt diese zumeist in ein für einen Browser lesbares Format, zum Beispiel HTML um. Diese HTML Datei kann der Client-Browser dann von dem Server abfragen und den Nutzer präsentieren.

4.2.2 Geschwindigkeit

Früher wurde der PHP-Interpreter mittels eines CGI, Common Gateway Interface, - Programmes realisiert. Dieses erforderte aber für jede Ausführung, also Anforderung, einer PHP-Datei durch den Webserver, dass eine eigene Instanz des PHP-Interpretern gestartet werden musste. Durch diesen Umstand wurden Server, welche mit vielen Anfragen belastet wurden, durch die große Anzahl von laufender PHP-Interpreter-Instanzen ausgebremst.

Um dieses Problem zu adressieren werden heutzutage die meisten PHP-Interpreter als Servermodule über das FastCG Interface betrieben. Dieses Interface ermöglicht, dass der einmal gestartete PHP-Interpreter von mehreren Anfragen verwendet werden kann. Dies erspart das umständliche Starten eines neuen PHP-Interpreters, welches einen großen Overload erzeugt und die Antwortzeit vergrößert.

Um die Ausführungsgeschwindigkeit von PHP Dateien weiter zu steigern gibt es noch weitere Möglichkeiten um den Code zu optimieren. Im nachfolgenden Teil möchte ich mich aber auf 2 beschränken.

4.2.2.1 OP-Code-Caching

Normalerweise muss vor jeder Ausführung einer PHP-Datei diese Datei in einen sogenannten OP-Code umgewandelt werden. Da normale PHP-Implementationen keinen OP-Code-Cache besitzen muss diese Umwandlung bei jeder Ausführung der PHP-Datei wiederholt werden. Um diese unnötig ausgeführte Mehrfachumwandlung zu vermeiden gibt es mehrere Plugins, welche die PHP Engine um einen OP-Code-Cache erweitern. Diese speichern den bei der ersten Ausführung erstellen OP-Code und führen diesen bei nochmaligen Aufruf der selben PHP-Datei direkt aus. Dies beschleunigt die Ausführung immens. Eine kostenlose Umsetzung dieser Funktion bildet der Eaccelerator. [17]

4.2.2.2 Kompilierung

Bei dieser Methode um die Geschwindigkeit von PHP-Dateien zu erhöhen wird der PHP-Code in die jeweils von der Plattform abhängigen Maschinensprache übersetzt. Durch diese Kompilierung wird die Ausführungsgeschwindigkeit erheblich gesteigert.

Eine Software welche auf dieser Idee funktioniert ist zum Beispiel die freie Software HipHop. HipHop wurde ursprünglich von einem Entwickler-Team von Facebook entwickelt, welches aber den Quell-Code unter einer OpenSource-Lizenz veröffentlicht. Im Gegensatz zu anderen Optimierungsmethoden, kann durch die Pre-Kompilierung die Notwendigkeit eines PHP-Interpreter komplett entfallen, da der Webserver in den meisten Fällen die erzeugten Programme direkt ausführen kann. Dadurch ergibt sich eine drastisch geringere Speicher- als auch Prozessorauslastung. Diese Steigerung kann laut Entwicklern bis zu 50% betragen.[18]

4.3 Java (Grafl)

4.3.1 Allgemein

Als weiteren großen Grundbaustein, neben dem Android Betriebssystem, für unser Projekt ist die Programmiersprache Java. Diese ist entwickelt worden, um Probleme bei modernen Programmieranwendungen lösen zu können.

Java wurde 1991 von James Gosling, Mike Sheridan und Patrick Naughton als internes Projekt bei Sun Microsystems entwickelt. Sie wollten bei der Entwicklung an die C/C++ Notation anschließen, da diese den Programmierern der damaligen Zeit bekannt war. Das Entwicklerteam stellte sich selbst die Anforderungen, dass die Sprache:

1. simpel, objektorientiert und vertraut
2. robust und sicher
3. Hardware- und Plattformunabhängig
4. hoch leistungsfähig
5. interpretierbar, parallelisierbar und dynamisch

sein muss.

Diese Ziele wurden alle eingehalten und sind das Grundkonzept der Programmiersprache, auf das im nächsten Kapitel näher eingegangen wird.

4.3.2 Grundkonzept

Wie schon erwähnt, wollten die Entwickler eine vertraute Sprache und haben sich so an die Schreibweise und Notation von C bzw. C++ angelehnt. Wie hier in diesem Beispiel zu sehen ist, sind die Syntaxen der beiden Sprachen sehr ähnlich.

C++

```
class Foo {                // Deklaration der Klasse
    public:
        int x;              // Variable
        Foo(): x(0) {}      // Konstruktor für Foo, initialisiert x
        int bar(int i) {    // Funktion bar()
            return 3*i + x;
        }
};
```

Java

```
class Foo {                // Definiert die Klasse Foo
    public int x;           // Eigenschaft
    public Foo() {         // Konstruktor für Foo
    }
    public int bar(int i) { // Methode
        return 3*i + x;
    }
}
```

4.3.3 Objektorientierte Programmierung

Was man in diesem Beispiel auch erkennen kann, ist die Eigenschaft der objektorientierten Programmierung (fortan OOP genannt). Die OOP ist ein Prinzip der Programmierung, in dem Daten (sogenannte Eigenschaften) und Funktionen (oder auch Methoden), die auf diese Daten angewandt werden können, möglichst eng in einem Objekt zusammenzufassen sind. Auf diese Objekte kann von außen nicht zugegriffen werden, um mögliche Manipulation der gesicherten Daten zu verhindern. [19]

4.3.4 Klassen

Um ein solches Objekt nutzen zu können, braucht es zuerst einen „Plan“ nachdem es zusammengesetzt wird. Dieser „Plan“ ist die Klasse. So muss, um ein Objekt erstellen zu können, zuerst die Klasse mit den Eigenschaften und Methoden gefüllt werden, die benötigt werden. Verallgemeinernd könnte man auch sagen, dass eine Klasse dem Datentyp eines Objekts entspricht. Formal gesehen belegt eine Klasse somit zur Programm-Ausführungszeit keinen Arbeitsspeicher, sondern immer nur die Objekte, die von ihr instanziiert wurden. [20]

Nehmen wir als Beispiel eine Lampe. Diese besitzt mehrere Eigenschaften, wie zum Beispiel Farbe, Größe, Gewicht, etc. Und Funktionen, nämlich ein- und ausschalten. Würde man dies als Klasse anschreiben, könnte es abstrakt so aussehen:

```
class Lampe {  
    // Eigenschaften  
    farbe;  
    gewicht;  
    lichtfarbe;  
    helligkeit;  
  
    // Methoden  
    einschalten();  
    ausschalten();  
}
```

Ein wichtiges Konzept der Objektorientierung ist die Vererbung, bei der eine neue Klasse von einer bereits vorhandenen Klasse abgeleitet wird. Die neue Klasse ist eine Abwandlung bzw. Erweiterung der ursprünglichen Klasse und wird Unterklasse, Kindklasse oder Subklasse genannt. Die ursprüngliche Klasse wird als Oberklasse, Elternklasse oder Superklasse bezeichnet.

In unserem Lampen-Beispiel hier könnte das eine Straßenlaterne oder eine Taschenlampe sein. Diese haben die Eigenschaften der Superklasse (Lampe) und zusätzlich noch spezifische Eigenschaften und Methoden, welche die Unterklassen erweitern.

```
class Taschenlampe extends Lampe {  
    //Eigenschaften  
    maximaleLeuchtdauer;  
    //Methoden  
    batterieLaden();  
}
```

4.3.4.1 Datenkapselung

Nachdem die Grundlagen der OOP erläutert wurden, stellt sich nun die Frage: Warum braucht man diese eigentlich beziehungsweise, wozu wird sie verwendet? Nun, in erster Linie zur Datenkapselung.

Als Datenkapselung versteht man in der Informatik das verbergen oder geheim halten von wichtigen Daten vor einem Zugriff von außen. Damit wird der Eingriff auf die internen Datenstrukturen unterbunden und erfolgt stattdessen nur noch über vordefinierte Schnittstellen.

In Java wird dies durch ein Attribut vor der Klassendefinition, einer Variablen- oder Methodendeklaration gekennzeichnet.

Es gibt drei beziehungsweise vier Arten der Zugriffseinstellung.

`public` – Zugreifbar für alle Instanzen (auch die anderer Klassen)

`private` – Nur für Instanzen der eigenen Klasse zugreifbar

`protected` – Nur für Instanzen der eigenen Klasse und Ableitungen davon zugreifbar

default (ohne Angabe eines Attributs) – erlaubt den Zugriff für alle Elemente innerhalb des eigenen Pakets

Beispiel: Konto-Klasse mit gekapselten Daten

```
class Konto {  
    private int kontoNr;  
    private double kontostand;  
    public Konto(int nr) {  
        kontoNr = nr;  
    }  
    public void einzahlen(double betrag) {...  
    }  
    public void abheben(double betrag) {...  
    }  
    public double kontostandAbfragen() {...  
    }  
}
```


4.4 JSON (Grafl)

Ein JSON, kurz für JavaScript Object Notation, ist ein leicht lesbares, kompaktes Datenformat zur Übertragung von Datenobjekten. Für Maschinen ist dieses Format besonders leicht zu lesen, aufzugliedern und zu generieren, da es komplett unabhängig von Programmiersprachen ist, jedoch eher der Notation der C Sprachen ähnelt.

Ein JSON baut auf zwei Strukturen, einerseits, den Name/Wert-Paaren, welche als *Objekt*, Struktur, Verzeichnis, Hash Tabelle oder als assoziatives Array realisiert werden kann, je nachdem in welcher Sprache es verwendet wird, und andererseits einer geordneten Liste von Werten, dies wird in den meisten Programmiersprachen als *Array*, Vektor, Liste oder Sequenz umgesetzt.

Als *Objekt* (oder auch object) versteht man in JSON eine ungeordnete Liste von Name/Wert-Paaren, welche mit Beistrichen (' , ') getrennt und vom geschwungenen Klammern am Anfang und Ende zusammengefasst werden (' { ' und ' } '). Um ein solches Name/Wert-Paar zu erstellen, muss zuerst ein Name angegeben werden, anschließend wird nach einem Doppelpunkt (' : ') der Wert zugewiesen.

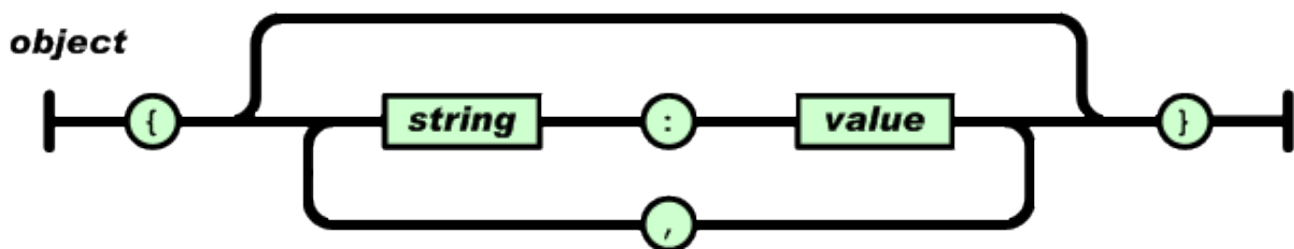


Abbildung 5: JSON-Object

Ein *Array* hingegen ist eine geordnete Liste von Werten (im Englischen „values“). Diese Werte können verschiedenste Formen annehmen (zum Beispiel Objekte, Strings, Nummern, Characters, Arrays, etc.), innerhalb des Arrays werden diese Werte mit Beistrichen getrennt (' , ') und mit eckigen Klammern (' [' und '] ') zusammengefasst.

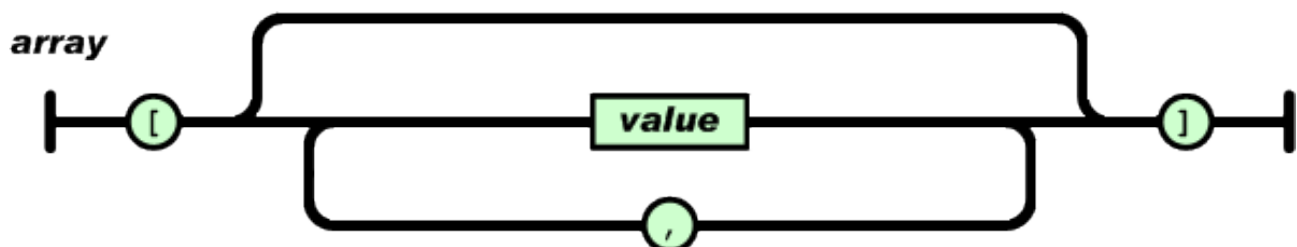
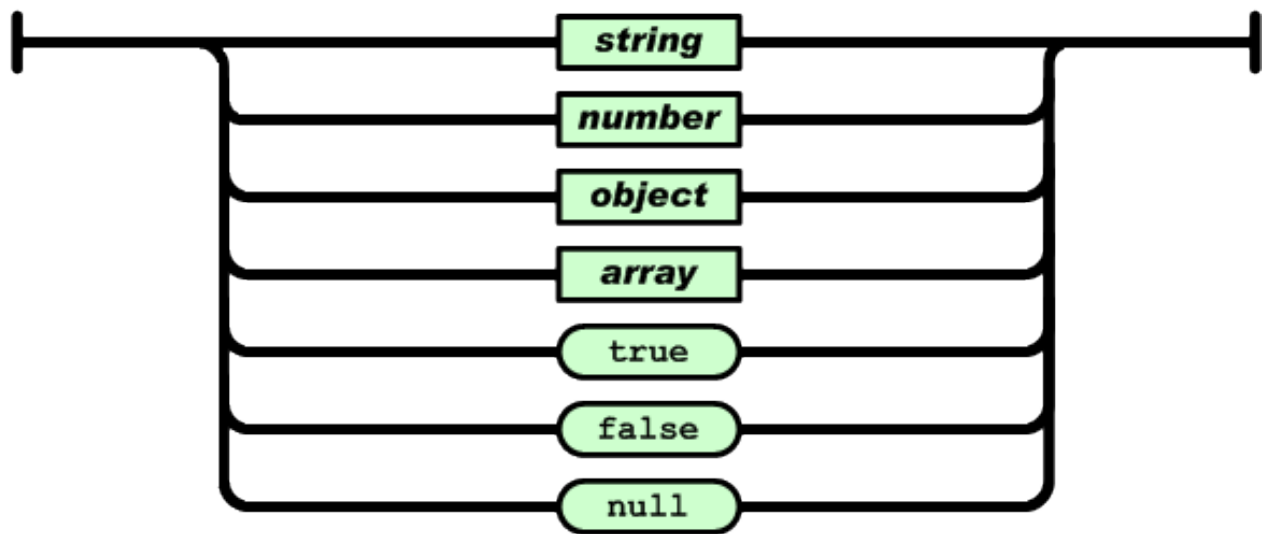


Abbildung 6: JSON-Array

value*Abbildung 7: JSON-Value*

Warum ist JSON so wichtig für unser Projekt?

Mit JSON ist uns eine reibungslose Datenübertragung von unserer Datenbank über eine PHP-API zu den Android Geräten möglich. Wir übergeben dieser Schnittstelle die Daten unserer Datenbank in SQL Format, diese überträgt sie dann in JSON und sendet sie per HTTP an das Gerät. Auf dem Gerät kann das JSON ohne Probleme gelesen und aufgegliedert werden.

4.5 MySQL (Stiel)

MySQL ist das populärste quelloffene SQL-Datenbankmanagementsystem der Welt und wird von der MySQL AB entwickelt, vertrieben und supportet. MySQL AB ist ein, von den MySQL-Entwicklern gegründetes, kommerzielles Unternehmen aber auch ein Open-Source-Unternehmen, der zweiten Generation. Das bedeutet, dass die Werte und Methodik von Open Source mit einem erfolgreichen Geschäftsmodell verbunden wird.

Eine Datenbank ist im allgemeinen nichts anderes als eine strukturierte Sammlung von Daten. Diese kann aus riesigen Informationsmengen bestehen, aber auch aus einer gewöhnlichen Einkaufsliste. Um diese Daten dann einer Computerdatenbank hinzufügen zu können, wird ein Datenbankmanagementsystem wie zum Beispiel der MySQL Server benötigt.

Damit man nicht alle Daten in einem einzigen Speicherraum speichern zu muss, ist MySQL auch ein relationales Datenbankmanagementsystem, welches dem User das Speichern von Daten in separaten Tabellen ermöglicht. SQL ist die gebräuchlichste Sprache, die für Datenbanken eingesetzt wird und ist seit 1986 in Verwendung.

Da MySQL eine quelloffene Software ist, bedeutet das für jeden, dass er diese Software benutzen aber auch verändern kann. Man kann sie ganz einfach vom Internet herunterladen ohne etwas zu bezahlen. MySQL verwendet die Lizenz GPL (GNU Gernerall Public License). Diese legt fest, was man mit der Software machen darf und was nicht. Wenn dem User der Gebrauch der GPL Lizenz nicht zusagt, kann dieser auch eine kommerzielle Version von MySQL erwerben.

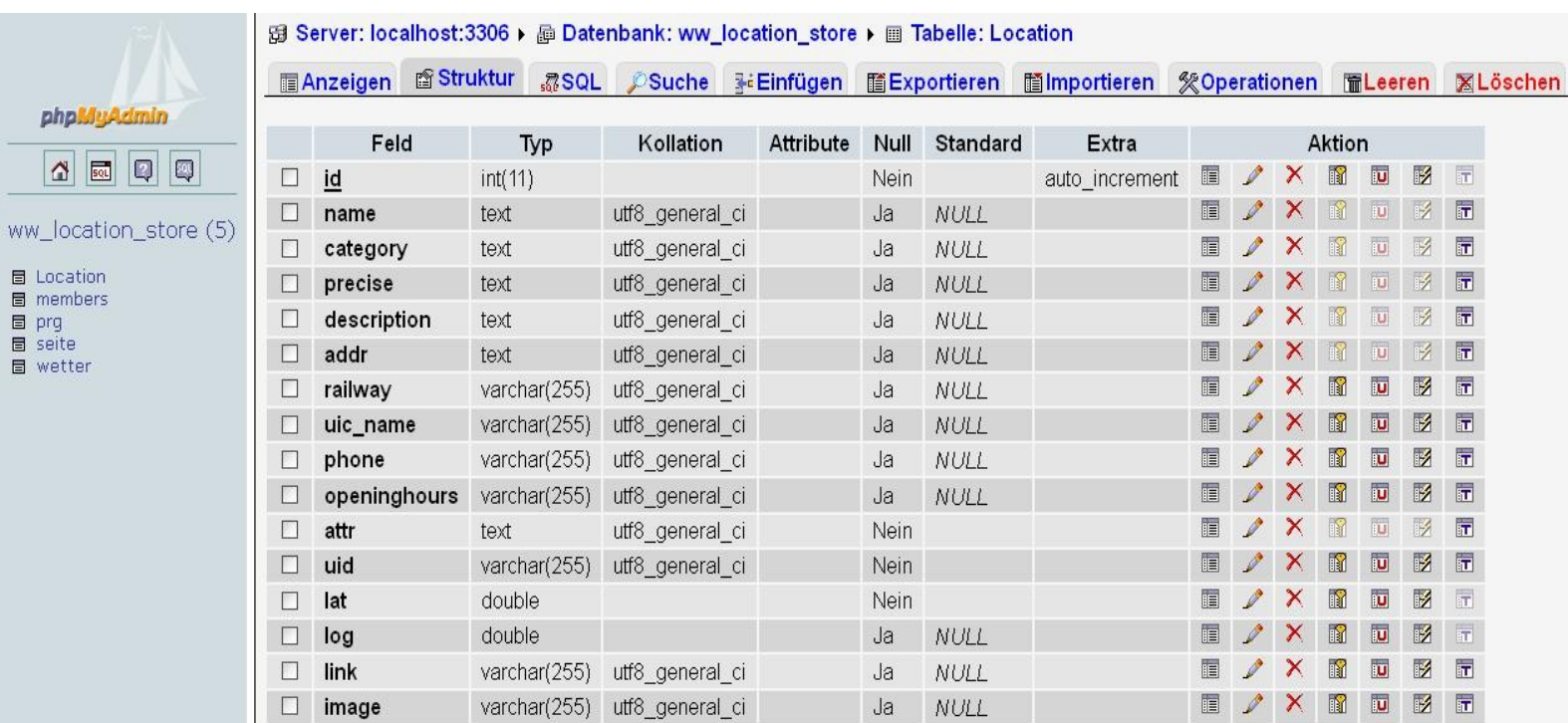
Der MySQL-Datenbankserver ist auch deshalb so beliebt, weil er sehr schnell, zuverlässig und einfach zu benutzen ist. Aus diesen Gründen befindet sich der Server auch schon seit Jahren in wichtigen Produktionsumgebungen im Einsatz und ist auch auf Grund der flexiblen Anbindungsmöglichkeiten, sowie Geschwindigkeit bzw. Sicherheit für den Zugriff auf Datenbanken im Internet geeignet.

Die Datenbanksoftware von MySQL ist ein Client-Server-System, das aus einem Mehr-Thread-SQL-Server besteht. Dieser Mehr-Thread-SQL-Server unterstützt verschiedene Backends sowie diverser Clientprogramme bzw. Clientbibliotheken und Verwaltungswerkzeuge. Außerdem kann dieser mittels vieler Programmschnittstellen angesprochen werden.[21]

4.5.1 phpMyAdmin

phpMyAdmin ist ein Tool für die Administration von MySQL-Datenbanken über eine Web-Oberfläche, welches vollständig in HTML und PHP programmiert ist. Da die Administration über HTTP erfolgt, ist es auch möglich, dass Datenbanken über das Internet administriert werden. Um phpMyAdmin zu verwenden benötigt man keinerlei SQL-Kenntnisse, da es nach dem WYSIWYG-Prinzip (What you see is what you get) arbeitet. Bei diesem Prinzip wird ein Dokument genauso am Bildschirm angezeigt, wie es danach auf dem Papier aussieht.

Mit MySQL kann man Datenbanken erstellen und löschen. Weiters kann man Tabellen erstellen, ändern und löschen, sowie Tabellenfelder hinzufügen, bearbeiten oder löschen. Außerdem können Schlüssel verwaltet und MySQL-Abfragen ausgeführt werden. Daher ist es auch möglich MySQL-User zu verwalten.[22]



The screenshot displays the phpMyAdmin web interface. At the top, the navigation bar shows the server 'localhost:3306', the database 'ww_location_store', and the selected table 'Location'. Below this, a row of tabs allows switching between 'Anzeigen' (Display), 'Struktur' (Structure), 'SQL', 'Suche' (Search), 'Einfügen' (Insert), 'Exportieren' (Export), 'Importieren' (Import), 'Operationen' (Operations), 'Leeren' (Empty), and 'Löschen' (Delete). The 'Struktur' tab is active, showing a table structure with columns: id, name, category, precise, description, addr, railway, uic_name, phone, openinghours, attr, uid, lat, log, link, and image. Each column has a checkbox for selection and a set of icons for actions like edit, delete, and insert. The left sidebar shows the database 'ww_location_store (5)' and a list of tables: Location, members, prg, seite, and wetter.

	Feld	Typ	Kollation	Attribute	Null	Standard	Extra	Aktion
<input type="checkbox"/>	id	int(11)			Nein		auto_increment	[Icons]
<input type="checkbox"/>	name	text	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	category	text	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	precise	text	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	description	text	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	addr	text	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	railway	varchar(255)	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	uic_name	varchar(255)	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	phone	varchar(255)	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	openinghours	varchar(255)	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	attr	text	utf8_general_ci		Nein			[Icons]
<input type="checkbox"/>	uid	varchar(255)	utf8_general_ci		Nein			[Icons]
<input type="checkbox"/>	lat	double			Nein			[Icons]
<input type="checkbox"/>	log	double			Ja	NULL		[Icons]
<input type="checkbox"/>	link	varchar(255)	utf8_general_ci		Ja	NULL		[Icons]
<input type="checkbox"/>	image	varchar(255)	utf8_general_ci		Ja	NULL		[Icons]

Abbildung 8: phpMyAdmin Interface

4.6 OpenStreetMap-Projekt (Stiel)

OpenStreetMap (OSM) ist eine kostenlose Landkarte, welche das Ziel hat freie geographische Daten, wie Straßen, Häuser, Flüsse und alles weitere das auf Karten zu sehen ist, zu erfassen. Finanziert wird das Projekt durch verschiedenen Aktivitäten wie zum Beispiel Spenden um die Infrastruktur aufrecht zu erhalten.

Das Kartenmaterial wird von freiwilligen Usern gesammelt, dadurch gewinnt es täglich an Umfang. Der Inhalt steht unter der Creative-Commons-Lizenz CC-by-sa. Das bedeutet, dass jeder mit den Daten machen kann was er möchte, solange diejenige Person oder Firma auf die Herkunft bzw. die Lizenz der Daten hinweist. Wenn dies geschieht, ist die freie Nutzung der kostenlosen Daten gewährleistet.

Die Kartendaten von OSM lassen außerdem offen, wofür sie verwendet werden. Sie können die Karten in eine einfache Webseite einbinden, aber auch als Routenplaner oder als Navigationssoftware verwenden. Die Daten selbst kann man vom OSM-Server downloaden und speichern, um sie dann auch im offline-Betrieb, zum Beispiel für ein Navigationsgerät, verwenden zu können. Der User kann wählen, in welchen Format er den Export der Karte haben möchte. Er muss den Bereich der Karte markieren, den er benötigt und bevor er die Karte exportieren kann das Format auswählen. Außerdem kann der User wählen in welchen Detaillierungsgrad die Daten angezeigt werden, oder ob die nur für Wanderer wichtigen Informationen angezeigt werden. Dies ist nur abhängig von den Anzeigeprogrammen bzw. der Auswahl der Daten. [23]

5 PRAKTISCHE REALISIERUNG

Die Realisierung des Smartphone-basierten Bezirksführers umfasst nicht nur die Programmierung der Smartphone – Applikation sondern noch weitere Arbeiten. Im Allgemeinen kann man das Projekt in 4 große Abschnitte unterteilen.

- Erarbeiten der grundlegenden Hardwarefunktionen in Android
- Programmierung der Smartphone-Applikation
- Bereitstellen der Points of Interest
- Administrative Oberfläche für Verwaltung und Verbreitung im Web erstellen

Auf den folgenden Seiten werden wir auf die praktische Umsetzung eingehen.

5.1 Grundlegende Hard- & Softwarefunktionen von Android (Graf1)

5.1.1 MapView

Um die MapView realisieren zu können brauchen wir in erster Linie Kartenmaterial. Wir haben Überlegungen angestellt, wie wir das am besten realisieren können, und sind dadurch auf die freie Google Maps Bibliothek gestoßen, welche es erlaubt die Karte samt Multitouch-Funktion zu verwenden. Wir haben die Kartenfunktion in unser Programm eingebaut, um alle beziehungsweise bestimmte Kategorien der Objekte aus unserer Datenbank auf der Karte anzeigen zu lassen.

Um die Karte darstellen zu können, muss die Hauptklasse von der Klasse MapActivity abgeleitet werden. Anschließend muss die Karte definiert und initialisiert werden. Dies geschieht in der onCreate-Methode und bindet die MapView aus dem XML-File an die Java-MapView. Um die Karte ordnungsgemäß steuern zu können, musste noch ein MapController angelegt werden. Dieser MapController wird per

```
map.getController();
```

an die Karte gebunden und beinhaltet Funktionen um sie zu steuern, wie zum Beispiel Zoomen oder zu einem bestimmten Punkt „hinfahren“.

```
public class main extends MapActivity{
```

```
private MapActivity map;

private MapController myMapController;

public void onCreate (Bundle savedInstanceState){

    map = (MapActivity) findViewById(R.id.map);

    MyMapController = map.getController();

}

}
```

Um die einzelnen Objekte auf der Karte anzeigen lassen zu können, haben wir eine eigene Klasse dafür geschrieben. Diese Klasse nennt sich ItemizedOverlay (grob übersetzt bedeutet das so viel wie „einzeln aufgeführte Einblendung“) und regelt die Erstellung, Verwaltung und Interaktion mit solchen „Einblendungen“.

Damit die Einblendungen an der richtigen Stelle angezeigt werden, benötigt man sogenannte GeoPoints. Diese GeoPoints können jeweils eine X- und eine Y-Koordinate zugewiesen und der ItemizedOverlay Klasse übergeben werden. Diese erstellt dann auf dem Punkt die Einblendung.

5.1.2 Kamera

Um die Kamerafunktion etablieren zu können, muss eine Erlaubnis im `AndroidManifest` hinzugefügt werden. Wir haben zudem eine eigene Klasse geschrieben, in der die Funktionen, die die Kamera besitzen soll definiert wurden. Zudem ist in dieser Klasse, genannt „Preview“, angegeben, was zu tun ist, wenn die Activity geschlossen oder verändert wird. Diese Preview wird in der Hauptklasse an ein `FrameLayout` gebunden und liefert somit ein Kamerabild über das gesamte Display. Ein `FrameLayout` kann nur ein Element auf dem ganzen Bildschirm anzeigen und wird gerne beim Programmieren von Spielen verwendet oder wie in unserem Fall um die Kamera anzeigen zu lassen.

Mit dem Befehl

```
camera = Camera.open();
```

wird das Kamerabild schlussendlich angezeigt.

Nachdem wir die Kamerafunktion eingebaut hatten, gab es jedoch Probleme mit der Auflösung und der Darstellung. Das Bild war nämlich verzerrt und um 90° verdreht. Nach etlichen Recherchen und ausreichendem Testen konnten wir das Problem lösen in dem wir die vorgegebene Bildschirmgröße auf 800x480 statisch eingestellt haben, da es mit einem automatischen Verfahren nicht funktioniert hat. Zudem haben wir das Bild um 90° nach rechts gedreht und in das Android-Manifest eingetragen, dass das Bild fixiert ist und es sich durch kippen nicht verstellt.

5.1.3 HTTP

Die Datenübertragung zwischen dem Android Gerät und der PHP-Schnittstelle findet per HTTP statt. Damit diese HTTP-Übertragung zustande kommt, wird eine URL benötigt, von der der Inhalt gelesen werden kann. Die PHP-API überträgt die Daten aus der Datenbank in ein JSON und stellt diese dem Smartphone zur Verfügung.

```
String url = "http://www.rybin.de/User/Guide/LocatorAPI/api.php?
mode=search&name=Asperner%20L%F6we";
String result = "";
DefaultHttpClient client = new DefaultHttpClient();
HttpGet method = new HttpGet(url);
HttpResponse res = null;
try {
    res = client.execute(method);
}
catch (IOException e1) {

    e1.printStackTrace();
}

InputStream is = res.getEntity().getContent();
BufferedReader reader = new BufferedReader(new InputStreamReader(is, "iso-
8859-1"));
StringBuilder sb = new StringBuilder();
String line = null;
while ((line = reader.readLine()) != null) {

    sb.append(line + "\n");
}

is.close();

result = sb.toString();
```

Das ist der Algorithmus mit dem die Daten von der URL auf das Gerät gelesen werden. Nachdem der DefaultHttpClient den Datensatz mit der Methode HttpGet geladen hat, kann das Ergebnis mit Hilfe eines BufferedReaders und eines StringBuilders in einen String umgewandelt werden.

5.1.4 ListView

Eine ListView ist eine scrollbare Liste von Objekten die über den ListAdapter automatisch hinzugefügt werden. Dem ListAdapter muss man lediglich die Information geben, die in der ListView angezeigt werden soll. Um eine ListView überhaupt implementieren zu können, müssen die Felder aus denen die Liste besteht erst einmal im XML-File definiert werden.

```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:textSize="16sp" >
</TextView>
```

Diese Codezeilen definieren das Aussehen dieser List-Items und kann danach im Programm zugewiesen werden. Die Hauptklasse des Programm muss zudem erst von der Klasse ListActivity abgeleitet werden. Anschließend wird der ListAdapter aufgerufen, welcher die Elemente in die Liste einträgt. Nachdem die ListView erstellt wurde, muss ein onItemClickListener aufgerufen werden, welcher die Elemente anklickbar macht.

```
public class HelloListView extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setListAdapter(new ArrayAdapter<String>(this, R.layout.list_item,
            COUNTRIES));

        ListView lv = getListView();

        lv.setTextFilterEnabled(true);

        lv.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View view,
                int position, long id) {
                // When clicked, show a toast with the TextView text
```

```
        Toast.makeText(getApplicationContext(), ((TextView) view).getText(),
        Toast.LENGTH_SHORT).show();

    }

});
}

static final String[] COUNTRIES = new String[] {
    "Afghanistan", "Albania", "Algeria", "American Samoa", "Andorra",
    "Angola", "Anguilla", "Antarctica", "Antigua and Barbuda", "Argentina"
}
}
```

Im String COUNTRIES befinden sich einige Testelemente, die in die Liste eingetragen werden.

5.1.5 Aufbereitung der Daten

Die Vielzahl von Daten, welche wir über den Interpreter von OpenStreetMapProject bekommen haben, war zum damaligen Zeitpunkt noch unsortiert und kaum mit Informationen bestückt. Also haben wir es uns Anfang Februar zur Aufgabe gemacht, sämtliche Daten per Hand auszusortieren, unnötige Daten zu löschen und das brauchbare Material eigenhändig aufzubereiten.

Um die Daten zugänglich zu machen, haben wir ein Interface entwickelt, das diese Datensätze, nach ID geordnet anzeigt. Mit der Option „Löschen“ kann man einen der Datensätze vollständig aus unserem System entfernen. Mit der Schaltfläche „Bearbeiten“ kann der Datensatz per Hand nachbearbeitet werden. Siehe Abbildung 9: Web-Interface

Für die meisten dieser Daten war eine ausgiebige Recherche im Internet notwendig, da wir dem Benutzer möglichst detaillierte Angaben liefern wollten. Da dies sehr aufwendig ist, werden wir uns auch noch in Zukunft damit beschäftigen müssen um alle 900 Datensätze mit korrekten Informationen, die der Anwender brauchen könnte, zu versehen.

Action	id	name	category	precise	description	addr
bearbeiten loeschen	35526358	OMV Brünner Straße 172	amenity	fuel	OMV 24h-Tankstelle Filiale Brünner Straße 172	Brünner Straße 172, 1210 Wien
bearbeiten loeschen	55411316	Hilton Vienna Danube, Handelskai 269	tourism	hotel	Hilton Hotel	Handelskai 269, 1020 Wien
bearbeiten loeschen	55411636	Marina Wien	amenity	restaurant	Die Sonnenterrasse mit Blick auf die Donau und Skyline von Wien lassen Sie alles andere vergessen. Erholen Sie sich vom Alltagsstress und lassen Sie sich vom Küchen- und Serviceteam verwöhnen.	Seitenhafenstraße 15
bearbeiten loeschen	61275091	Billa Castagnagasse	shop	supermarkt	Billa Filiale Castagnagasse	Castagnagasse 1220 Wien
bearbeiten loeschen	61275147	Cineplexx Reichsbrücke	amenity	cinema	Digital Cinema, 3D Digital	Wagramerstraße 79
bearbeiten loeschen	61675671	Surfclub Otto Wiener Nordrand Schnellstraße	sport	NULL	NULL	Wiener Nordrand Schnellstraße, 1220 Wien
bearbeiten loeschen	61972167	Donaufelder Kirche Zum hl. Leopold	religion	catholic	Gottesdienstordnung: Montag, Mittwoch und Freitag: 7.00 Uhr Dienstag und Donnerstag: 18.30 Uhr Samstag: 8.00 Uhr: Seelenmesse / bei Bedarf, 18.30 Uhr: Vorabendmesse Sonntag und Feiertag: 8.00, 9.30, 18.30 Uhr Beichtgelegenheit nach jeder	Kinzerplatz 19

Abbildung 9: Web-Interface

5.2 Programmierung der Smartphone-Applikation (Gwihs)

Von Beginn der Realisierung der Applikation war es uns sehr wichtig, dass das spätere Endprodukt so benutzerfreundlich wie möglich sein sollte. Deswegen befragten wir Freunde und Verwandte was sie an konventionellen Stadtführern stört.

Nach der Auswertung dieser Umfrage konnten wir die gewünschten Eigenschaften in 5 Grundfunktionen vereinen. Die sogenannte: LiveView, Such-Funktion, Karten-Ansicht und die Advisor Funktion und um die Navigation zwischen diesen Funktionen zu ermöglichen ein Menu. In den folgenden Kapiteln werde ich die Funktionsweise, den technischen Hintergrund und den Nutzen für den User beschreiben.

5.2.1 Menü

Um zwischen den verschiedenen Funktionen navigieren zu können implementierte ich ein Menü. Diese Menü besteht aus verschiedenen Buttons welche mit großen Symbolen bestückt sind. Die Symbole sind so gestaltet, dass sie ohne weiteren Text auskommen. Dieser Umstand ermöglicht eine schnelle Navigation ohne kleine Texte auf relativ kleinen Smartphone-Displays lesen zu müssen.



Abbildung 10: Menü

Hinter jedem Button arbeitet ein ImageButton welcher einen onClickListener besitzt, welcher Benutzerinteraktionen abfangen kann.

```
ImageButton live = (ImageButton)findViewById(R.id.live);  
live.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View arg0) {  
        Intent liveactivity = new Intent(menu.this, liveview.class);  
        startActivity(liveactivity);  
    }  
});
```

In der onClick Methode wird dann die ausgewählte Funktion mittels einem Intent gestartet.

Ebenso bietet das Menü eine Übersicht auf das Wetter der nächsten 3 Tage. Diese Wetterdatenbank wird lokal mittels einer SQLite-Datenbank realisiert. Dadurch wird unnötiger und redundanter Datentransfer vermieden und eine höhere Reaktionszeit gewährleistet.



Abbildung 11: Wetter

Wenn die Applikation zum ersten Mal ausgeführt wird, wird eine neue SQLite-Datenbank erstellt.

```
private void onCreateDBAndDBTabled()
{
    SQLiteDatabase myDB = null;
    try {myDB = this.openOrCreateDatabase(MY_DB_NAME, MODE_PRIVATE, null);
        myDB.execSQL("CREATE TABLE IF NOT EXISTS " + MY_DB_TABLE + " (_id
        integer primary key autoincrement, zeit varchar(255), date var
        char(100), temp integer, typ varchar(50))" + ";");
    } finally {
        if (myDB != null)
            myDB.close();
    }
}
```

Die Funktion `onCreateDBandTabled()` erstellt eine Tabelle, falls diese nicht bereits am Smartphone vorhanden ist, mit den Spalten `id`, `zeit`, `date`, `temp` und `typ`. Die Tabelle wird mittels einem von der API zur Verfügung gestellten Wetterfeed gefüllt.

```
int valid = Integer.valueOf(c.getString(c.getColumnIndex("zeit")));
long aktuell = System.currentTimeMillis()/1000;
if(valid>aktuell)
{ //Anzeigen der Daten
}
else
{ //Neu laden der Daten
}
```

Die Applikation überprüft jedes Mal, wenn das Menü angezeigt wird, ob die Datensätze noch aktuell sind, wenn nicht wird eine Verbindung mit der API aufgebaut und mittels dem Aufruf der URL <http://www.rybin.de/User/Guide/LocatorAPI/api.php?mode=wetter> neue Daten angefordert.

Diese Abfrage der Daten wird in einem Thread ausgeführt um den Benutzer weiterhin die Interaktion mit der Applikation zu ermöglichen. Wenn die Abfrage ausgeführt wurde werden die erhaltenen Daten mittels SQL Befehlen in die Datenbank geschrieben.

```
for(i=0;i<=wetter_forecast.size()-1;i++)
{
    wetter_data ins = (wetter_data)wetter_forecast.get(i);
    myDB.execSQL("INSERT INTO "+MY_DB_TABLE+" (_id, zeit, date, typ, temp) "
    +"VALUES ('"+String.valueOf(i)+"',"+ "'"+ins.timestamp.toString()+"',"+
    "'"+ins.date.toString()+"',"+ "'"+ins.type.toString()+"',"+
    "'"+String.valueOf(ins.temp)+" ');");
}
myDB.close();
```

Nachdem nun Wetterdaten vorhanden sind werden diese auf dem Menü angezeigt und die Applikation kann nun wieder bis zu dem Zeitpunkt, bis die Daten wieder ungültig geworden sind (üblicherweise wird jeden Tag ein Update durchgeführt) auf die lokale Datenbank zugreifen und benötigt keine Internetverbindung.

5.2.2 LiveView

Die LiveView ist die Hauptfunktion des Bezirksführers. Diese Funktion ermöglicht es dem Benutzer das einfache Erkennen und Identifizieren einer Sehenswürdigkeit. Der Benutzer muss um das Objekt zu erkennen einfach nur das Smartphone auf das Objekt richten. Siehe Abbildung 12: LiveView Demonstration. Der Bezirksführer gibt dann die entsprechenden Informationen auf dem Display aus. Abbildung 13: LiveView Demonstration Smartphone Screenshot



Abbildung 12: LiveView Demonstration



Abbildung 13: LiveView Demonstration Smartphone Screenshot

Um dem Benutzer das Gefühl zu geben, das gewünschte Objekt zu „scannen“, implementierte ich eine Live Ansicht der Kamera auf der Rückseite des Mobiltelefons. Diese Methode wurde von meinem Kollegen Alexander Grafl übernommen. Dieser hat diese Funktion bereits in einer Test-Applikation getestet und so konnte diese direkt von mir übernommen werden.

Um die Position und die Ausrichtung des Mobiltelefons im Raum feststellen zu können benötigte ich umfangreiche Daten von den verschiedensten Sensoren des Smartphones, neben der genauen geografischen Position auch die magnetischen Ausrichtung (vereinfacht Himmelsrichtung). Besonders die Errechnung der magnetischen Ausrichtung stellte sich als eine komplizierte Angelegenheit dar. Eine weitere Herausforderung war die korrekte Eindämmung der möglichen Objekte besonderes in urbanen Bereichen, da hier die Objektdichte am höchsten ist.

Um die konkrete Position des Users bestimmen zu können, griff ich auf den sogenannten Location Listener zurück. Ob ein Smartphone die Möglichkeiten besitzt den Standort des Nutzers festzustellen kann mittels

```
lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
```

ermittelt werden. Falls das Gerät diesen Service bietet liefert die `getSystemService` Funktion einen `LocationManager` zurück, über diesen können dann alle weitere Einstellungen, zum Beispiel die Rate der Positionsbestimmung (die Zeit, die vergeht bis ein neues Update durch das System bereitgestellt wird) eingestellt werden. Von welchem Sensor diese Daten stammen sind der Applikation egal, da das Android System den Hintergrund der Daten durch standardisierte Schnittstellen verschleiert. Dies erleichtert die Programmierung wesentlich, da hier dem Entwickler mögliche Fehlerbehandlungen abgenommen werden.

Um die bereitgestellten Updates zu behandeln benötigt man einen sogenannten Listener. Dieser kann an einen `LocationManager` gebunden werden. Innerhalb des Listeners behandelte ich die ankommende Updates und entschied dann ob neue Daten vom Server nachgeladen werden müssen, oder ob die bereits vorhandenen Daten noch aktuell sind. Dieser Datenradius, also der Radius in dem die Daten auf dem Mobiltelefon bereits zwischengespeichert sind, beschränkte ich auf 500m. Dies ermöglichte eine nicht allzu große Datenmenge pro Abruf, besonders im Hinblick auf die derzeit noch weit verbreiteten eingeschränkten Datenvolumen bei den verschiedenen Mobilfunkanbietern, ebenso konnte ich dadurch eine schnelle Reaktionszeit der Applikation garantieren.

Da ich die Distanz zwischen dem aktuellen und dem letzten Positionsupdate bestimmen musste, suchte ich eine Methode, die die Distanz zwischen 2 Punkten mit einer hohen Genauigkeit bestimmen kann. Da sich die Berechnung der Daten auf eine runde Oberfläche (Erde) bezieht, kann die einfache Bestimmung mittels dem Satz des Pythagoras große Ungenauigkeiten aufwerfen. Die genauere Bestimmung der Distanz mittels Winkelfunktionen, welche Rücksicht auf die Kugelform der Erde nimmt, benötigen jedoch wesentlich mehr CPU-Zyklen. Hier stellt sich die Frage ob mehr Wert auf Genauigkeit oder Geschwindigkeit gelegt wird.

Im Endeffekt entschied ich mich für die folgende Berechnung

```
dist = 111.324 * acos(sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(lon2 - lon1))
```

Falls das Ergebnis dieser Berechnung 500m übersteigt wird der Abruf neuer Daten mittels einem eigenen Thread, um die weitere Benutzerinteraktion mit der Applikation zu ermöglichen, veranlasst. Die Daten werden mittels einem einfachen HTTP-Request angefordert. Der Server liefert anschließend die angeforderten Daten im JSON-Format an das Mobiltelefon. Die Applikation verarbeitet diese Daten und speichert diese anschließend in eine lokale Datenstruktur für die weitere Verarbeitung.

Um dem Nutzer das entsprechende Objekt anzeigen zu können muss noch die konkrete Ausrichtung des Mobiltelefons im Raum errechnet werden. Um dies bewerkstelligen zu können, griff

ich auf einen Magnetsensor, im Grunde ein integrierter digitaler Kompass, zurück. Statt einem Locationmanger benötigte ich für den Zugriff auf den Magnet Sensor einen SensorManager. Dieser SensorManager ermöglicht einen Zugriff auf hardwarenahe Sensoren.

```
SensorManager.getRotationMatrix(rotationMatrix, null, lastMagFields, lastAccels)

SensorManager.getOrientation(rotationMatrix, orientation);
```

Mittels den beiden Funktionen `getRotationMatrix` und `getOrientation` wird zuerst eine Matrix von Positionsdaten des Mobiltelefon befüllt und anschließend in eine Matrix umgewandelt die unter anderem die Anzahl der Grad bis zum Nullmeridian beinhaltet.

Da die Applikation nun weiß was der genaue geografische Standpunkt und in welcher Lage sich das Mobiltelefon im Raum befindet kann nun berechnet werden welche Objekte im Sichtfeld des Benutzers liegen und welche auf dem Display angezeigt werden sollen. Die Überprüfung, ob sich ein Objekt (Points of Interests) im Sichtfeld befindet, habe ich in eine Funktion ausgelagert. An diese Funktion werden alle Objekte geliefert welche sich in einem Umfeld von 60m befinden. Im Gegensatz zu der vorherigen Distanzberechnung mittels Winkelfunktion, verwendete ich in diesem Fall die einfache Berechnung mittels dem Satz von Pythagoras, da hier die anfallende Ungenauigkeit zu vernachlässigen ist.

```
public boolean check(Double xu, Double yu, Double compass, Double x1, Double y1)
```

Diese `check` – Funktion benötigt als Angabe die x und y Koordinaten des Nutzers sowie die Koordinaten des zu überprüfenden Objektes und die magnetische Ausrichtung im Raum. Mittels dieser Parameter kann durch Berechnungen im Polarkoordinaten System der Differenzwinkel zu dem Objekt errechnet werden. In der `check`-Funktion wird anschließend überprüft ob dieser in einem gewissen Blickfeld liegt, welches ich bei $\pm 20^\circ$ festgelegt habe.

Wenn diese Funktion ein positives Ergebnis zurück liefert wird das Objekt am Display ausgegeben. Diese Überprüfung wird jedes Mal durchgeführt wenn sich der Wert des Kompass ändert. Hierbei ruft das System die `onSensorChanged()` Funktion auf. Ein Objekt wird solange auf dem Display angezeigt bis es außerhalb des Sichtbereiches liegt beziehungsweise ein anderes Objekt näher beim Benutzer liegt.

5.2.3 Suchfunktion

Die Such Funktion ist ein Feature welches erst durch die Verlagerung eines Stadtführers auf ein elektronisches Gerät umgesetzt werden konnte. Das Problem der meisten Stadtführer ist der unübersichtliche Aufbau und die schwer zu findenden Einträge. Bei unseren Umfragen war der Wunsch nach einer Suchfunktion dementsprechend sehr groß.

Die Suche durfte aber auf keinem Fall umständlich und schwer zu verwenden sein. Ebenfalls war es sehr wichtig, dass der Benutzer nicht den kompletten beziehungsweise richtigen Suchbegriff eingeben musste. Diese Anforderung an den Suchalgorithmus erfordert die Anwendung einer Fuzzy-Search.

Die relevanten Ergebnisse sollten in einer Liste untereinander dargestellt werden und nach einem Klick auf einem Eintrag sollte eine Detailansicht des gewünschten Eintrags angezeigt werden.

Zu Beginn der Programmierung stand ich vor dem Problem, dass die Einträge in einer Liste angezeigt werden sollten. Die Android Plattform stellt zwar eine sogenannte ListView zu Verfügung, doch die praktische Anwendung war komplizierter als gedacht. Hier half mir die von meinem Kollegen Alexander Grafl getestete Demo-Applikation. Diese stellte verschiedene Strings als Liste untereinander dar. Auf dieser Beispiel-Anwendung aufbauend programmierte ich die Suche.

Die eigentliche Suche findet auf einem Datenbank Server im WWW statt. Der Donaustadtführer schickt den von dem Benutzer eingegebene String an die Datenbank-API. Diese liefert entweder Ergebnisse im JSON-Format, beziehungsweise falls keine Ergebnisse gefunden werden, eine entsprechende Fehlermeldung. Diese Fehlermeldung wird in einem benutzerfreundlichen Format angezeigt. Im Fall, dass die Suche Ergebnisse liefert, wird das Ergebnis mittels einem JSONArray in einzelne Objekte zerlegt.

```
JSONArray json = new JSONArray(result);
```

Ein JSONArray ermöglicht es die einzelne Objekte welche im JSON Format vorliegen zu trennen. Mit der Funktion `getJSONObject(Integer)` können diese der Reihenfolge nach abgerufen werden, sodass die einzelne Objekte als lokale Objekte am Smartphone vorliegen.

Da die ListView einzelne Strings zum Befüllen der Liste benötigt mussten nun nur noch alle Namen der einzelnen Objekte mittels einer Schleife ausgelesen und in String-Array gespeichert werden. Diese String-Array muss nun dem ListAdapter übergeben werden. Dieser Adapter ermöglicht die Steuerung der ListView.

```
setListAdapter(new ArrayAdapter<String>(getBaseContext(), R.layout.  
list_item, erg));
```

Nachdem das String-Array erg dem ListAdapter übergeben wurde und mit der Funktion `setListAdapter` an die `ListView` gebunden wurde, kann die `ListView` angezeigt werden. Siehe Abbildung 14: Such-Funktion



Abbildung 14: Such-Funktion

Um nun auf Interaktionen des Benutzers einzugehen muss ein `OnItemClickListener` an die `ListView` gebunden werden. Dieser bearbeitet dann alle Nutzerinteraktionen.

```
lv.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View view, int position,
        long id) {if(Integer.valueOf(Long.toString(id))!=ende)
        {todisplay = (Daten)datensatz.get(Integer.valueOf(Long.to
            String(id)));
        Bundle bundle = new Bundle();
        bundle.putSerializable("Daten", todisplay);
        Intent detail = new Intent(search.this, detailview.class).putEx
            tra("Objekt", bundle);
        if(vonact==1)detail.putExtra("from", 2);
        if(vonact==2)detail.putExtra("from", 3);
        startActivity(detail);
        }
    }
});
```

Die `onItemClickListener` Funktion wird aufgerufen falls der Benutzer auf einen Eintrag klickt. Das Android Betriebssystem liefert nun die ID der Auswahl zurück. Mittels dieser ID kann nun der entsprechende Datensatz aus der Liste von Objekten ausgewählt werden, da die Reihenfolge der Liste die selbe ist wie die in der `ArrayList`. Dieser Datensatz wird anschließend der Detailview übergeben welche diese anschließend benutzerfreundlich ausgibt.

5.2.4 MapView

Um eine große Menge an Points-of-Interests benutzerfreundlich darzustellen, entschied ich mich dazu eine Karten-Activity, die sogenannte MapView zu implementieren. Diese Activity baut auf die von meinem Kollegen Alexander Grafl erarbeiteten Grundlagen auf. Er analysierte die verschiedenen Methoden um Karten auf mobilen Geräten anzuzeigen. Er entschied sich für die Google-Map View. Diese View stellt ohne großen Programmieraufwand eine ausgreifte Karte zu Verfügung. Auf dieser Karte können verschiedene Objekte und auch die aktuelle Position des Benutzer angezeigt werden.

Die MapView-Activity kann in verschiedenen Modi gestartet werden. Diese werden der Activity zuvor mit der putExtra-Methode mitgeteilt.

Die Activity kann entweder einen einzelnen Punkt mit Beschriftung anzeigen, oder mehrere Punkte mit angepassten Symbolen.

```
final ItemizedOverlay itemizedoverlay = new ItemizedOverlay(icon, this);

point = new GeoPoint(48221689, 16.445585);

OverlayItem overlayitem = new OverlayItem(point, "HTL Donaustadt", "HTL
    Donaustadt");

itemizedoverlay.addOverlay(overlayitem);

mapOverlays.add(itemizedoverlay);
```

Falls nur ein Punkt angezeigt werden soll wird der Name des anzuzeigenden Objekt der Funktion TexttoBitmap übergeben.

```
final HelloItemizedOverlay itemizedoverlay = new
HelloItemizedOverlay(TexttoBitmap(todisplay.name), this);
Drawable TexttoBitmap(String text)
{
    Bitmap icon= BitmapFactory.decodeRe
source(this.getResources(), R.drawable.flag).copy(Bitmap.Con
fig.ARGB_8888, true);
    Canvas canvas_icon = new Canvas(icon);
    Paint paint = new Paint();
    paint.setARGB(255, 0, 0, 0);
    paint.setStyle(Paint.Style.FILL);
    paint.setTextSize(17);
    paint.setTextAlign(Align.CENTER);
    canvas_icon.drawText(text.toString(), icon.getWidth()/2, 90, paint);
    Drawable d = new BitmapDrawable(icon);
    d.setBounds(-150, -150, 150, 100);
    return d;}
```

Die Methode `TexttoBitmap()` zeichnet auf ein davor in den Programmressourcen abgelegtes Bild eine Textunterschrift. Anschließend wird mittels Geopoint das zuvor gezeichnete Bild auf die Karte gezeichnet und die Kartenansicht auf das anzuzeigende Objekt fixiert.

Falls die Karte mehrere Punkte anzeigen soll wird ein loader-Thread gestartet. Dieser Thread lädt alle vorhandenen Daten aus der Datenbank. Anschließend werden die Objekte in verschiedene Kategorien wie zum Beispiel Restaurants, Tankstellen, Sehenswürdigkeiten eingeteilt. Jede dieser Kategorien besitzt ein eigenes Symbol.

Nach dieser Einteilung wird jedes Objekt einzeln mittels Geopoint auf der Karte dargestellt.

Wenn nun der Nutzer auf ein bestimmtes Symbol drückt, ruft die MapView die `onTap` Funktion des Punktes auf. Da das Objekt zuvor dem Punkt übergeben wurde kann nun das komplette Objekt der Detailview übergeben werden und dem Benutzer angezeigt werden.

5.2.5 Advisor

Ein weiteres Problem welches sich bei der Befragung herausstellte war, dass es oft ein Problem gibt, dass der Benutzer nicht genau weiß nach welchem Objekt er eigentlich sucht. Deswegen entschieden wir uns dazu einen sogenannten Advisor zu programmieren. Dieser soll dem Benutzer durch die Auswahl von verschiedenen Kategorien eine Auswahl an Ergebnissen liefern welche alle seine Kriterien erfüllen.

In der derzeitigen Entwicklungsphase stehen 4 Hauptkategorien zu Verfügung, „Sehenswürdigkeiten“, „Night-Life“, „Unterhaltung“, „Essen&Trinken“. Nachdem der Benutzer einer dieser Hauptkategorien ausgewählt hat stehen weitere Unterkategorien zu Verfügung. Wenn der Benutzer in der letzten Unterteilungs-Stufe angelangt ist wird eine Kategorie-Suche gestartet. Siehe Abbildung 15: Advisor Anwendungsbeispiel Diese Such-Funktion wird durch die Datenbank API zu Verfügung gestellt.

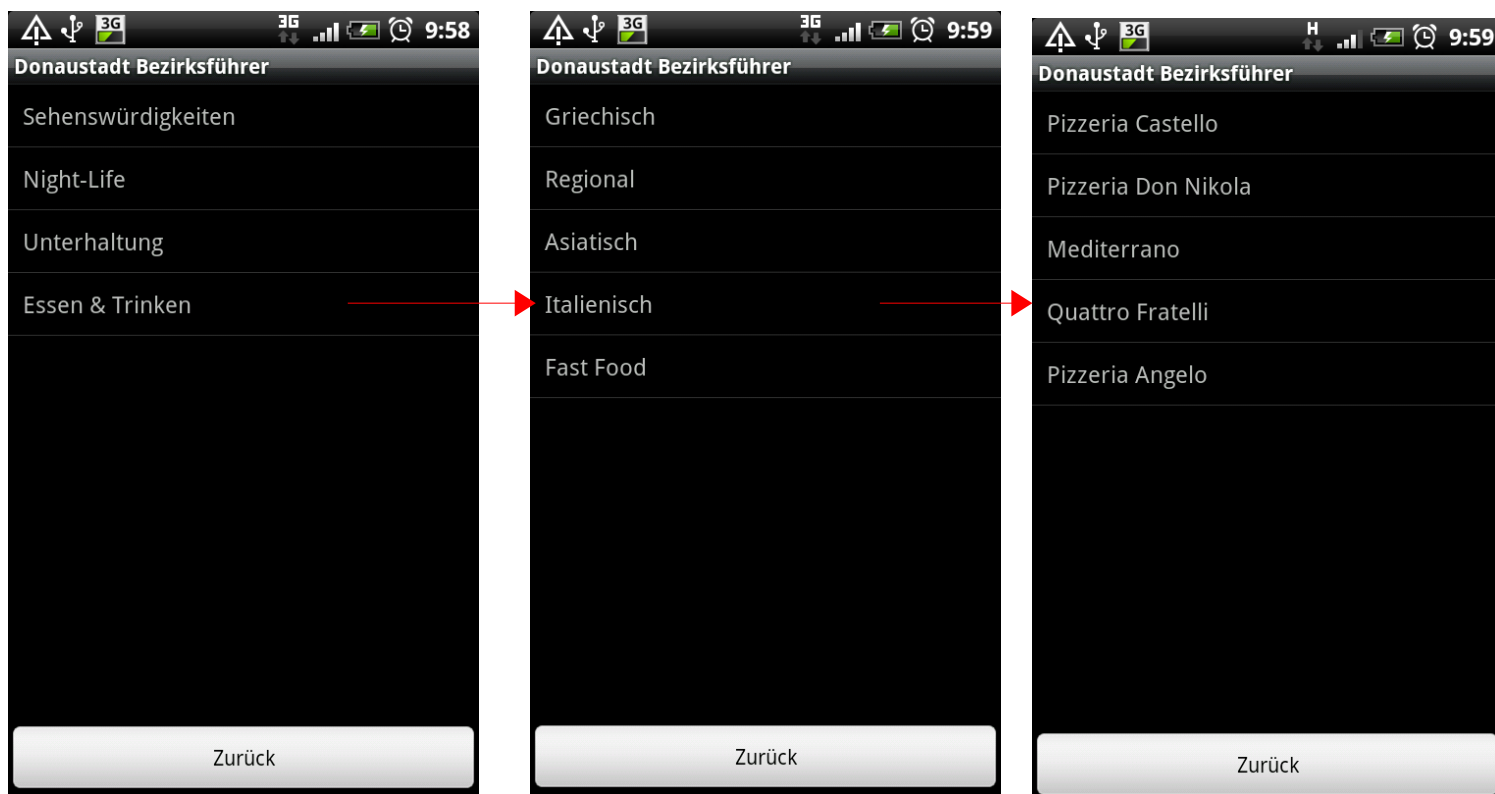


Abbildung 15: Advisor Anwendungsbeispiel

Die Datenbank liefert nun die Ergebnisse, wie bei der Such-Funktion, in einem JSON-Format. Ebenfalls wird zu der Anzeige die selbe ListView wie bei der Such-Funktion verwendet. Durch diesen Umstand ersparten wir uns viel Programmieraufwand.

5.3 Datenbank-API (Gwihs)

Zu Beginn unserer Arbeit an dem Donaustadtführer hatten wir 2 Optionen wie wir die Abfrage der Daten von der im Internet stehenden Datenbank durchführen konnten. Entweder über eine direkte SQL Abfrage an die Datenbank oder einer Schnittstelle zwischen Datenbank und Smartphone.

Da die direkte Verbindung mit der Datenbank ein gewisses Sicherheitsrisiko barg und Änderungen an der SQL-Abfrage große Eingriffe in die Programmlogik notwendig machen würden, entschieden wir uns für die Umsetzung einer Schnittstelle. Diese sollte verschiedene Modi zur Verfügung stellen und die Daten in einem leicht wieder in eine interne Datenstruktur umwandelbaren Format bringen lassen.

Um diese Kriterien zu erfüllen entschieden wir uns für die Datenübertragung das JSON Datenformat. Die Schnittstelle sollte mit der Skriptsprache PHP umgesetzt werden. Da diese Sprache

von den meisten Webhostern unterstützt wird und keine tiefe Eingriffe in die Serversoftware notwendig ist war PHP die Ideale Wahl für unseren Fall.

5.3.1 Positions-Modus

Der Positions-Modus ist der Standard Modus der API. Mittels diesen Modus werden die Objektdaten rund um eine Geo-Position ausgegeben. Dieser Modus wird zum Beispiel durch die LiveView verwendet.

Um die Datenbank-Abfrage zu starten muss die URL, <http://www.rybin.de/User/Guide/LocatorAPI/api.php>, mit den notwendigen Parameter, im Falle des Positions-Modus, den Längen- und Breitengrad der aktuellen Position, aufgerufen werden. Eine Anfrage die von einem Smartphone ausgeführt wird welches sich am Standpunkt der HTL-Donaustadt befindet sieht zum Beispiel so aus:

`http://www.rybin.de/User/Guide/LocatorAPI/api.php?lat=48.221465&log=16.445244`

Geographische Breite

Geographische Länge

Die Variablen lat und log werden als HTTP-GET-Argumente den Server übergeben. Diese beiden Argumente können dann im PHP Skript mittels `$_REQUEST['Argument-Name'];` abgerufen werden.

Anschließend wird die geografische Distanz zu jedem Objekt in der Datenbank überprüft.

```
while($e=mysql_fetch_assoc($q)){
    $lat2 = $e["lat"]; //Punkt aus der DB
    $log2 = $e["log"];
    $distanz =
111.324*acos(sin($lat1)*sin($lat2)+cos($lat1)*cos($lat2)*cos($log2-$log1));

    if($distanz < 0.5)$output[]=$e;
}
```

Alle Objekte die innerhalb eines 500m Radius um die angegebene Position befinden werden in ein Array gespeichert. Anschließend wird mit der Funktion `print(json_encode($output));` zu erst das Array mit den gefundenen Objekten in das JSON-Format konvertiert und danach mittels der `print`-Funktion ausgegeben. Das Smartphone kann nun die Daten in diesem Format weiter verarbeiten.

5.3.2 Such-Modus

Der Such-Modus erweitert die API mit der Funktion die Datenbank nach Namen zu durchsuchen. Um diesen Modus auszuwählen muss ein weitere Parameter hinzugefügt werden. Mittels dem Argument `mode = search` kann der Modus aktiviert werden, ein weiterer Parameter namens „name“ gibt das gesuchte Objekt an. Eine Anfrage nach Objekten welche die Zeichenkette „Donaustadt“ enthält sieht zum Beispiel so aus:

<http://www.rybin.de/User/Guide/LocatorAPI/api.php?mode=search&name=Donaustadt>

Das PHP-Skript versetzt die API durch den Parameter `mode = search` in den Suchmodus, dadurch ändert sich die weitere Abfragelogik.

```
$query = "SELECT * FROM Location where MATCH (name)
        AGAINST('+'.$name."'') OR name = '".$name."'";
$q=mysql_query($query);
    while($e=mysql_fetch_assoc($q)){ $output[]=$e;
    }
print(json_encode($output));
```

Anstatt einer normalen SQL-Vergleichs-Abfrage in dem Format `Select * From Location where name = $name` verwendete ich die komplexere Volltextsuche. Dies geschieht mittels den SQL-Operatoren `MATCH & AGAINST`. Durch diese beiden Operatoren werden nicht nur exakt gleiche Texte gefunden, sondern auch ähnliche. Dies ermöglicht mögliche Tippfehler auszugleichen und trotzdem korrekte Ergebnisse zu erzielen.

Die Verwendung dieser Funktion erforderte aber Änderungen an der bereits vorliegenden Datenbankstruktur. Um eine Volltextsuche durchführen zu können musste das Namen-Feld in der

Objekt-Datenbank als Volltext indiziert werden. Diese Konvertierung ist sehr rechenintensiv, muss aber nur einmal durchgeführt werden, sodass dieser Umstand kein Problem für uns darstellt.

5.3.3 Advisor-Modus

Der Advisor-Modus versetzt die API in die Lage Objekte gewisser Kategorien auszugeben. Da die Daten innerhalb der Datenbank eine normierte Einteilung in verschiedene Kategorien besitzt ist es sehr leicht mittels einiger SQL-Abfragen das gewünschte Ergebnis zu erreichen. Da die Datenbank auf normierte Kategorien setzt, kann bei diesem Modus auf die Volltextsuche verzichtet werden, sodass der Datenbankserver weniger stark belastet wird.

Der Advisor-Modus wird derzeit nur von der Advisor-Activity benutzt, da die Schritt-für-Schritt-Suche von Objekten auf Kategorien basiert. Wie bei dem Such-Modus muss der Advisor-Modus mit einem zusätzlichen Parameter, „mode=kategorie_search“ ausgewählt werden. Ein weiteres Argument mit dem Namen „kategorie“ gibt an nach welcher Kategorie gesucht werden soll.

Das PHP-Skript überprüft nun ob die gewünschte Kategorie vorhanden ist und führt dann die entsprechende SQL-Abfrage aus.

```
if($kategorie=="Italienisch")$query="SELECT * FROM `Location` WHERE `precise` = 'italian' || `precise` = 'pizza'";
if($kategorie=="Asiatisch")$query="SELECT * FROM `Location` WHERE `precise` = 'chinese'";
if($kategorie=="Griechisch")$query="SELECT * FROM `Location` WHERE `precise` = 'greek'";
if($kategorie=="Regional")$query="SELECT * FROM `Location` WHERE `precise` = 'regional'";
if($kategorie=="Fast Food")$query="SELECT * FROM `Location` WHERE `precise` = 'fast_food'";
if($kategorie=="Kino")$query="SELECT * FROM `Location` WHERE `precise` = 'cinema'";
if($kategorie=="Theater")$query="SELECT * FROM `Location` WHERE `precise` = 'theatre'";
if($kategorie=="Bars")$query="SELECT * FROM `Location` WHERE `precise` = 'bar' || `precise` = 'pub'";
if($kategorie=="Club")$query="SELECT * FROM `Location` WHERE `precise` = 'nightclub'";
if($kategorie=="Historisch")$query="SELECT * FROM `Location` WHERE `category` = 'historic'";
```

5.3.4 Wetter-Modus

Da der smartphone-basierte Bezirksführer eine 3 Tages – Wettervorschau besitzt musste die API um eine eigenen Wetter-Modus erweitert werden. Dieser Modus liefert die Wettervorschau der nächsten 3 Tage. Dieser Modus benötigt keine weiteren Parameter. Nach dem Aufruf der Wetter-API liefert die API ein JSON-Array mit dem Wetter für die nächsten 3 Tagen. Solch ein Array kann dann zum Beispiel so ausschauen:

```
[{"id": "16", "timestamp": "1306274399", "date": "Di", "temp": "28", "typ": "sonnig"},  
{ "id": "17", "timestamp": "1306360799", "date": "Mi", "temp": "30", "typ": "bewolkt"},  
{ "id": "20", "timestamp": "1306447199", "date": "Do", "temp": "29", "typ": "sonnig"}]
```

Jeder Datensatz besteht aus einem Timestamp, einem Wochentag (date), Temperatur(temp) und einem Typ. Die Typen bestimmen das anzuzeigende Symbol auf der Android Applikation. Es gibt insgesamt 7 verschiedene Arten: sonnig, leicht bewölkt, stark bewölkt, Regen, Sturm, Scheefall.

Im derzeitigen Entwicklungsstadium werden die Wetterdaten noch manuell über ein Webinterface in die Datenbank eingetragen. In Zukunft soll durch den Zugriff auf eine externe Wetter-API welche auf den Server gespiegelt wird ersetzt. Dadurch minimiert sich der Wartungsaufwand auf ein Minimum.

5.4 Bereitstellen der Points of Interest (Stiel)

5.4.1 OpenStreetMap-Projekt

5.4.1.1 Allgemein

Als Kartenmaterial haben wir uns für das des OpenStreetMaps-Projects (OSM) entschieden, da es, wie schon im Theoretischen Teil erwähnt, kostenlos ist und auf Grund der Creative-Commons-Lizenz CC-by-sa möglich ist, die Daten nach Belieben zu verwenden. Auf der Homepage von OSM kann man den Bereich von dem man das Kartenmaterial benötigt auswählen. In unserem Fall war das der 22. Wiener Gemeindebezirk. Von diesem Bezirk konnten wir die Daten als XML-File herunterladen um diese dann weiter zu verwenden.

5.4.1.2 Kartenmaterial analysieren

Als nächsten Schritt war es dann meine Aufgabe, das Kartenmaterial Zeile für Zeile zu durchsuchen bzw. zu analysieren. Diese Arbeit war für uns sehr wichtig, da wir herausfinden mussten, was diese Tags bedeuten, um später nur die für uns relevanten Tags mittels eines Interpreters automatisch in unsere Datenbank speichern zu können. Die für unser Projekt relevanten Daten sind Ämter, Restaurants, Supermärkte, historische Gebäude, Parkplätze, Pubs, Diskotheken, Einkaufszentren, Freizeitanlagen sowie deren Zusatzinformationen.

Diese Arbeit war sehr aufwendig, weil dieser OpenStreetMap-Ausschnitt des 22. Bezirks über 136.000 Objekte beinhaltet. Außerdem musste ich die Tags übersetzen, da diese nicht in deutscher Sprache verfügbar sind.

Dies kann man am folgenden Beispielen sehen:

- `<tag k="amenity" v="pub"/>`

Dieser „Tag“ bedeutet, dass an diesem Ort Bier und andere alkoholische Getränke ausgeschenkt werden. Darüber hinaus können ebenfalls Essen und Übernachtungen angeboten werden.

- `<tag k="leisure" v="sports_centre"/>`
`<tag k="sport" v="soccer"/>`

```
<tag k="sport" v="basketball"/>
```

Dieser bedeutet, dass an diesem Ort Leute ihre Freizeit verbringen und darunter, in Kombination mit „leisure“, was in diesem Sport-Center angeboten wird.

Aufgrund dieser Umständlichkeiten nahm diese Arbeit mehrere Tage in Anspruch.

5.4.2 MySQL-Datenbank erstellen

Zum Testen haben wir eine MySQL-Datenbank erstellt, da bei dieser keine Internetverbindung nötig ist und wir unsere Tests schneller durchführen konnten.

Um die Datenbank zu erstellen mussten wir noch einmal überlegen wie viele Felder oder Spalten wir für die Datenbank benötigen, um später den Interpreter, der die Daten in die Datenbank füllt, an die Datenbank anzupassen. Zusätzlich war es noch meine Aufgaben den Feldern deren richtigen Typ zuzuweisen, je nach dem wie viele Zeichen oder Ziffern die Felder darin maximal enthalten sollten. Schließlich und endlich wurden es 16 Spalten:

- id
In dieser Spalte wurde der primär-Schlüssel des jeweiligen Datensatzes gespeichert, um diesen eindeutig zu identifizieren.
- name
Hier wurde der Name des Objekts gespeichert.
- category
In dieser wurde der primäre-Tag der wichtige Einrichtungen wie Restaurants, Tankstellen und weitere Einrichtungen beinhaltet gespeichert, da wir diese in Kategorien unterteilt haben.
- precise
Die Spalte „precise“ gibt genau an um welche Art von Restaurants, Tankstellen oder sonstige relevanten Einrichtungen es sich handelt.
- description
Dieser Spalte wurde eine nähere Beschreibung zu den jeweiligen Einrichtung hinzugefügt.
- addr
Hier wurde die Adresse aber auch die Postleitzahl gespeichert.
- railway
In dieser Spalte wurde gespeichert ob sich in der jeweiligen Straße eine U-Bahn- oder S-Bahnhaltestelle befindet.

- uic_name
Diese Spalte verwenden wir derzeit noch nicht, gibt aber den genauen Namen an und benötigen würden wir diese erst bei weiteren Entwicklungen unseres Programms.
- phone
Wenn es zu den diversen Einrichtungen eine Telefonnummer gab, wurde diese hier gespeichert.
- openinghours
Hier wurden Öffnungszeiten von Restaurants, Ämter, Supermärkten & anderen Einrichtungen hinzugefügt.
- attr
In die Spalte wurden alle sonstigen Informationen gespeichert, wie zum Beispiel, ob in einem Restaurant das Rauchen erlaubt ist oder nicht.
- uid
Dies gibt die eindeutige ID in der OpenStreetMap an und wird auch erst bei weiteren Entwicklungen benötigt.
- Lat
In der Spalte „lat oder latitude“ wurde der Breitengrad der Koordinate gespeichert.
- Log
Da bekannter Weise eine Koordinate nicht nur aus dem Breitengrad besteht, wurde in der Spalte „log oder longitude“ der Längengrad gespeichert, um die Koordinate zu vervollständigen.
- link
Wenn die diversen Einrichtungen auch eine Homepage aufwiesen, wurde die Internetadresse in dieser Spalte festgehalten.
- image
Ich habe auch diverse Einrichtungen gebeten, ihr Firmenlogo als zusätzliche Information verwenden zu dürfen. Dieses wurde dann in der Spalte „image“ gespeichert.

5.4.3 Programmieren des Interpreters

Nach dem Erstellen der Datenbank war es meine Aufgabe, die Daten, welche sich noch in dem XML-File befanden, mittels eines Interpreters in die Datenbank zu schreiben. Dafür musste ich die Java-Entwicklungsumgebung Netbeans herunterladen. Außerdem musste ich mich mehr mit der Programmiersprache Java auseinander setzen, da meine Programmierkenntnisse zu diesem Zeitpunkt noch nicht ausreichten um dieses Aufgabenstellung zu lösen. Diese Kenntnisse eignete ich mir auf diversen Internet-Seiten bzw. Internet-Foren wie zum Beispiel <http://www.java-forum.org/> an.

Zuerst habe ich das XML-File, mit den OpenStreetMap-Daten, in die Entwicklungsumgebung übertragen. Dies sah in der Praxis folgender Maßen aus:

```
DocumentBuilder docBuilder=docBuilderFactory.newDocumentBuilder();
Document doc = docBuilder.parse(
newFile("C:\\Users\\Stieli\\Desktop\\5BHITN\\MaturaProjekt\\map9.osm"));
```

Hier wird eine Klasse „docBuilder“ erstellt, die ein XML-File parsed. Zusätzlich wird noch der Pfad mit der Stelle angegeben, wo sich das XML-File befindet.

Die nächste Aufgabe war es, dem Programm beizubringen, wie es die richtigen Daten einliest und in die Datenbank speichert. Daher mussten wir zuerst einmal angeben, wo sich die Datenbank befindet.

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

Hier wird ein Driver erstellt und dieser dient zur Verbindungsherstellung einer MySQL-Datenbank.

```
java.sql.Connection con = DriverManager.getConnection("
jdbc:mysql://127.0.0.1:3306/location3","root", "");
```

Hier baut der Driver die Verbindung zur Datenbank auf, dieser benötigt dazu den Pfad zur Datenbank.

Danach musste ich angeben, welche Felder sich in der Datenbank befinden. Das heißt es gibt 16 Felder und daher müssen auch alle in einem String gespeichert werden.

```
String query = "INSERT INTO Location (id, log, lat, uid, attr,
name, link, description, category, precise, phone,
openinghours, railway, uic_name, addr) " +
"VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

Außerdem wird für jedes Feld ein Platz freigehalten und dies geschieht mit VALUES (...). Jedes Fragezeichen hält einen Platz für ein Feld frei.

Als nächstes wurden mittels einer for-Schleife die Elemente von id, log, lat und uid aus dem XML-File geholt und dem Objekt stmt übergeben.

```
for (int i = 4; i < listOfRows.getLength(); i++) {
    Element element = (Element) listOfRows.item(i);
    stmt.setInt(1,Integer.parseInt(element.getAttribute("id")));
    stmt.setDouble(2,Double.parseDouble(element.getAttribute("lon")));
    stmt.setDouble(3,Double.parseDouble(element.getAttribute("lat")));
    stmt.setString(4, element.getAttribute("uid"));
```

Weiters habe ich mittels mehrerer if-Anweisungen die irrelevanten Daten, wie zum Beispiel Datensätze mit der Auskunft, dass dieses Objekt eine Autobahn ist, aus dem XML-File ausgeschlossen. Diese Aufgabe war zwar sehr aufwendig, da ich aber die relevanten bzw. die irrelevanten Datensätze schon beim manuellen Durchsuchen ausgewählt habe, konnten wir einiges an Zeit einsparen.

```
if(!valueattr.equals("highway"))
    {if(!valueattr.equals("created_by"))
        {//und viele mehr If-Anweisungen
        }
    }
```

Danach musste ich mittels einer weiteren if-Anweisung angeben, welche Attribute für uns relevant sind, damit diese auch in die Datenbank, in das Feld attr, gespeichert werden:

```
if(!valueattr.equals("name")&&!valueattr.equals("link")&&!
    Valueattr.equals("description")
    &&!valueattr.equals("amenity")
    &&!valueattr.equals("tourism")
    &&!valueattr.equals("religion")
    &&!valueattr.equals("historic")//und noch viele weiter Bedingungen
    {System.out.println(valueattr);
    attr = attr + test.getAttributes().getNamedItem("k") +
    test.getAttributes().getNamedItem("v") +
    attr_aviabile=true;
    }
```

Nach der Auswahl der benötigten Attribute werden diese gültig gesetzt damit diese in die Datenbank gespeichert werden können.

Da in unserer Datenbank auch keines der Felder leer sein bzw. ein Fragezeichen enthalten darf, schrieb ich in jedes leere Feld „NULL“. Dies war nötig um bei der nächsten Aufgabe zu wissen, in welchen Datensatz wir noch manuell Informationen hinzufügen müssen.

Dies sah in der Praxis so aus:

```
if(!name_aviable) stmt.setString(6, "NULL");
if(!link_aviable) stmt.setString(7, "NULL");
if(!description_aviable) stmt.setString(8, "NULL");
if(!category_aviable) stmt.setString(9, "NULL");
if(!precise_aviable) stmt.setString(10, "NULL");
if(!phone_aviable) stmt.setString(11, "NULL");
if(!openinghours_aviable) stmt.setString(12, "NULL");
if(!railway_aviable) stmt.setString(13, "NULL");
if(!uic_name_aviable) stmt.setString(14, "NULL");
if(!addr_aviable) stmt.setString(15, "NULL");
```

Abschließend fügten wir noch die Methode `executeUpdate()` ein um die Datenbank zu aktualisieren, aber nur wenn der Datensatz einen Namen besitzt. Da Datensätze ohne Namen keinen Wert für unser Applikation hätten.

```
if(name_aviable) stmt.executeUpdate();
```

6 LITERATURVERZEICHNIS

6.1 Internetverweise

- [1] o.V.: „Was ist Android?“
<http://www.go-android.de/was-ist-android>
(13.03.2011)
- [2] o.V.: „SDK“
<http://developer.android.com/sdk/index.html>
(13.03.2011)
- [3] o.V.: „Licenses“
<http://source.android.com/source/licenses.html>
(14.03.2011)
- [4] o.V.: „Android Open Source und im Handel“
<http://www.wortgefecht.net/open-source/android-open-source-und-im-handel/>
(13.03.2011)
- [5] Michael Held: „Verbreitung von Android Smartphones nimmt stark zu“
<http://mobiles-internet.buemo.net/2010/05/verbreitung-von-android-smartphones-nimmt-stark-zu/>
(12.03.2011)
- [6] o.V.: „Android-Konkurrenz: Neue Details zum Apple iPhone nano“
<http://www.mobilfunk-talk.de/news/32615-android-konkurrenz-neue-details-zum-appleiphone-nano/>
(14.03.2011)
- [7] Hans-Christian Dirscherl: „Die Android-Architektur“
<http://www.pcwelt.de/ratgeber/Die-Android-Architektur-Smartphone-Grundlagen-1005300.html>
(09.03.2011)
- [8] o.V.: „Physiology-of-an-Android“
<http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf>
(13.03.2011)

- [9] o.V.: „Inside the Android Application Framework“
http://stid.googlecode.com/files/Inside_the_Android_Application_Framework.pdf
(05.03.2011)
- [10] o.V.: „Activities“
<http://developer.android.com/guide/topics/fundamentals/activities.html#Lifecycle>
(11.03.2011)
- [11] o.V.: „Hello, Views“
<http://developer.android.com/guide/tutorials/views/index.html>
(10.03.2011)
- [12] o.V.: „ADT-Plugin for Eclipse“
<http://developer.android.com/sdk/eclipse-adt.html>
(14.03.2011)
- [13] o.V.: „Android Debug Bridge“
http://www.androidpit.de/de/android/wiki/view/Android_Debug_Bridge
(14.03.2011)
- [14] o.V.: „Android Emulator“
<http://developer.android.com/guide/developing/tools/emulator.html>
(15.03.2011)
- [15] o.V.: „Android Market“
<http://market.android.com/>
(15.03.2011)
- [16] o.V.: „PHP 5.3.5 and 5.2.17 Released!“
<http://www.php.net/archive/2011.php#id2011-03-17-1>
(20.04.2011)
- [17] Harry Slaughter
http://devbee.com/opcode_cache_for_dummies
(22.04.2011)
- [18] Scott MacVicar
http://www.facebook.com/note.php?note_id=416880943919
(22.04.2011)
- [19] o.V.: „Lesson: Object-Oriented Programming Concepts“
<http://download.oracle.com/javase/tutorial/java/concepts/index.html>
(20.04.2011)

[20]o.V.: „What is a Class?“

<http://download.oracle.com/javase/tutorial/java/concepts/class.html>

(21.04.2011)

[21]o.V.: „Was ist MySQL?“

<http://dev.mysql.com/doc/refman/5.1/de/what-is.html>

(20.04.2011)

[22]o.V.: „phpMyAdmin“

http://v.hdm-stuttgart.de/~riekert/lehre/php/mysql_php.html#pma_vorbemerkung

(17.04.2011)

[23]o.V.: „OpenStreetMap - Über“

<http://wiki.openstreetmap.org/wiki/%C3%9Cber>

(19.04.2011)

7 ABBILDUNGSVERZEICHNIS

Abbildung 1: Verbreitung von Android.....	16
Abbildung 2: Architektur von Android.....	18
Abbildung 3: Activity Lifecycle.....	20
Abbildung 4: Ablauf nachdem Aufruf einer PHP-Datei.....	27
Abbildung 5: JSON-Object.....	33
Abbildung 6: JSON-Array.....	33
Abbildung 7: JSON-Value.....	33
Abbildung 8: phpMyAdmin Interface.....	36
Abbildung 9: Web-Interface.....	44
Abbildung 10: Menü.....	46
Abbildung 11: Wetter.....	47
Abbildung 12: LiveView Demonstration.....	49
Abbildung 13: LiveView Demonstration Smartphone Screenshot.....	49
Abbildung 14: Such-Funktion.....	53
Abbildung 15: Advisor Anwendungsbeispiel.....	57

8 ABBILDUNGSQUELLENANGABE

Nr.	Quelle
Abbildung 1: Verbreitung von Android	http://mobiles-internet.buemo.net/wp-content/uploads/verteilung-smartphone-2010.jpg
Abbildung 2: Architektur von Android	http://www.pcwelt.de/images/5/7/4/7/1/8/def1e3e0a6c2d7e5.jpeg
Abbildung 3: Activity Lifecycle	http://developer.android.com/images/activity_lifecycle.png
Abbildung 4: Ablauf nachdem Aufruf einer PHP-Datei	http://de.wikipedia.org/w/index.php?title=Datei:PHP_funktionsweise.svg&filetimestamp=20081027202252
Abbildung 5: JSON-Object	http://www.json.org/object.gif
Abbildung 6: JSON-Array	http://www.json.org/value.gif
Abbildung 7: JSON-Value	http://www.json.org/value.gif
Abbildung 8: phpMyAdmin Interface	Eigene Darstellung
Abbildung 9: Web-Interface	Eigene Darstellung
Abbildung 10: Menü	Eigene Darstellung
Abbildung 11: Wetter	Eigene Darstellung
Abbildung 12: LiveView Demonstration	Eigene Darstellung
Abbildung 13: LiveView Demonstration Smartphone Screenshot	Eigene Darstellung
Abbildung 14: Such-Funktion	Eigene Darstellung
Abbildung 15: Advisor Anwendungsbeispiel	Eigene Darstellun

9 ANHANG

9.1 Bedienungsanleitung

Willkommen bei dem Donaustadt Guide, auf den folgenden Seiten möchten wir Ihnen die grundlegenden Funktionen des Guides erklären und näher bringen.

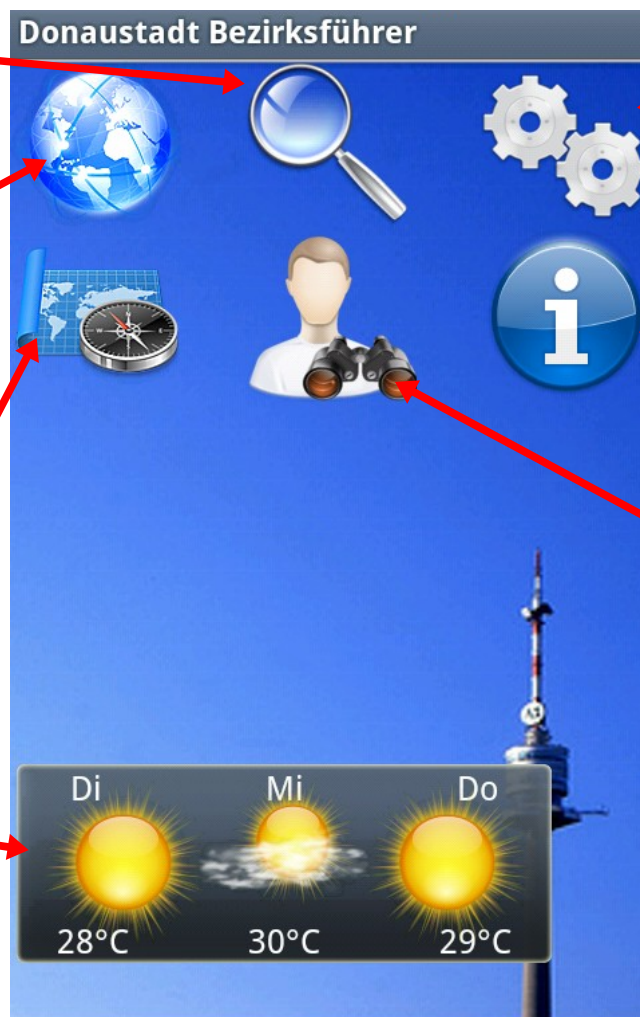
Wenn Sie den Donaustadt Guide starten, gelangen Sie zuerst in das Hauptmenü. Das Menü ermöglicht Ihnen eine leichte und übersichtliche Navigation innerhalb des Führers. Ebenso beherbergt es eine aktuelle 3-Tages-Wettervorschau.

Suche – mit dieser Funktion finden Sie mit einigen wenigen Schlagwörtern ihr gewünschtes Ziel

LiveView – Ihr persönlicher Stadtführer: einfach auf ein Objekt richten und sie gibt Ihnen weitere Informationen aus

MapView – Ihr Stadtplan in der Hosentasche

Wetter-Vorhersage – immer die Vorhersage für die nächsten 3 Tage dabei.



Einstellungen – hier können sie verschiedene Parameter ändern oder die korrekte Funktionsweise überprüfen

Info – hier erhalten Sie eine kurze Zusammenfassung der einzelnen Funktionen

Advisor – Ihr persönlicher Ratgeber

Auf den folgenden Seite erhalten Sie nähere Informationen zu den einzelnen Funktionen.



LiveView

Mittels der LiveView können Sie einfach den Namen und weitere Informationen von vielen Sehenswürdigkeiten und Gebäuden in der Donaustadt herausfinden.

Das Smartphone auf das gesuchte Objekt richten. Nun berechnet der Donaustadt Guide welches Objekt vor Ihnen befindet und gibt Ihnen auf dem oberen Bildschirmrand den Namen des Objektes aus.



Nun können Sie durch einen einfachen Klick auf den Namen, in diesem Fall Donauzentrum, weitere Informationen erhalten.



Suche

Mit der Suche haben Sie ein sehr mächtiges Werkzeug in der Hand. Durch die Eingabe von Teilwörtern des gesuchten Objektes sucht der Donaustadt Guide die passenden Einträge und zeigt sie übersichtlich in einer Liste an.

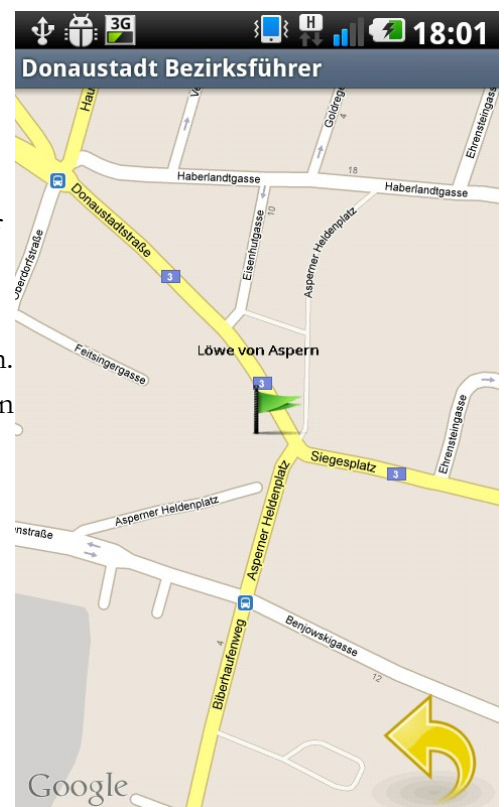
Zum Beispiel suchen wir nach dem bekannten Asperner Denkmal den Löwen von Apsern.



Nun können Sie auf den Eintrag mit dem Namen Löwen von Aspern klicken und Ihnen wird eine übersichtliche Information mit Bild zu dem Objekt geliefert. Falls Sie wissen wollen wo sich das gewünschte Objekt auf der Karte befindet reicht einfacher Klick auf das Kartensymbol in der rechten oberen Ecke.



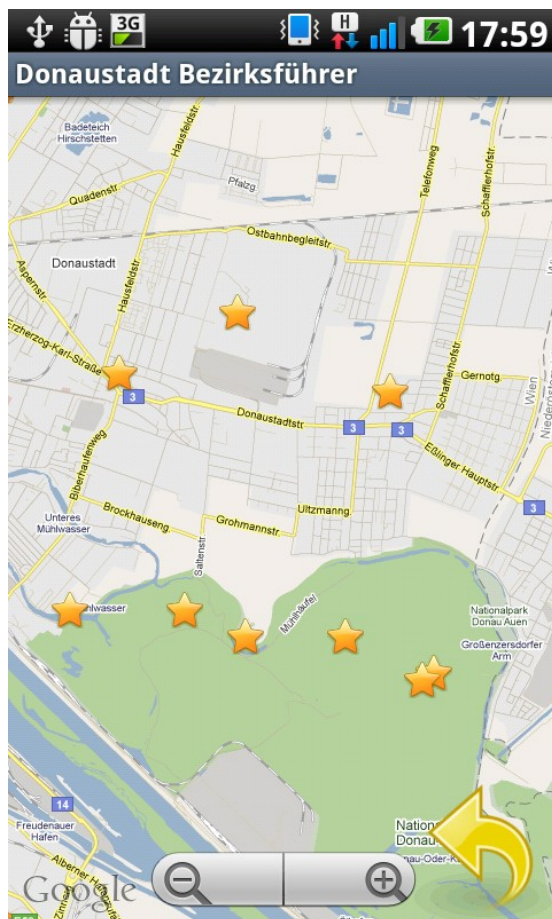
Nun sehen Sie eine Karte auf der Sie sich frei bewegen können. Die kleine grüne Flagge symbolisiert den Standort Ihres gesuchten Objektes.





MapView

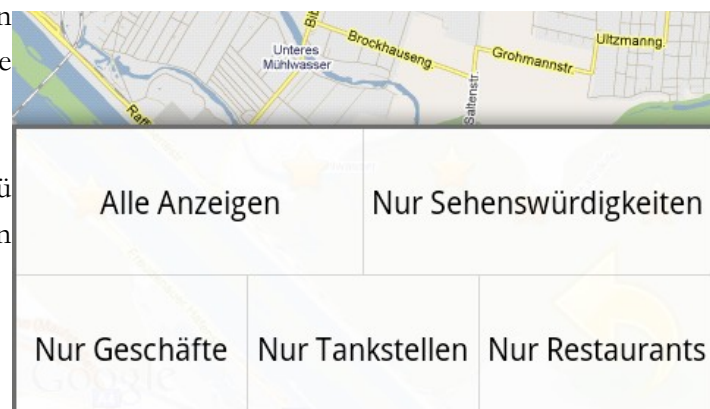
Wie jeder gute Stadtführer besitzt auch der DonaustadtGuide einen Stadtplan. Sie können sich auf den Stadtplan verschiedene Objektkategorien ein-/ausblenden lassen. Ebenso wird Ihre aktuelle Position auf der Karte angezeigt und sofort darauf zentriert.



Tipp! Mit einem einfachen Klick auf ein Objekt auf der Karte können Sie sich Informationen anzeigen lassen.

Die MapView unterstützt eine große Anzahl von verschiedenen Objekt Kategorien und kann diese getrennt aber auch alle auf einmal anzeigen lassen.

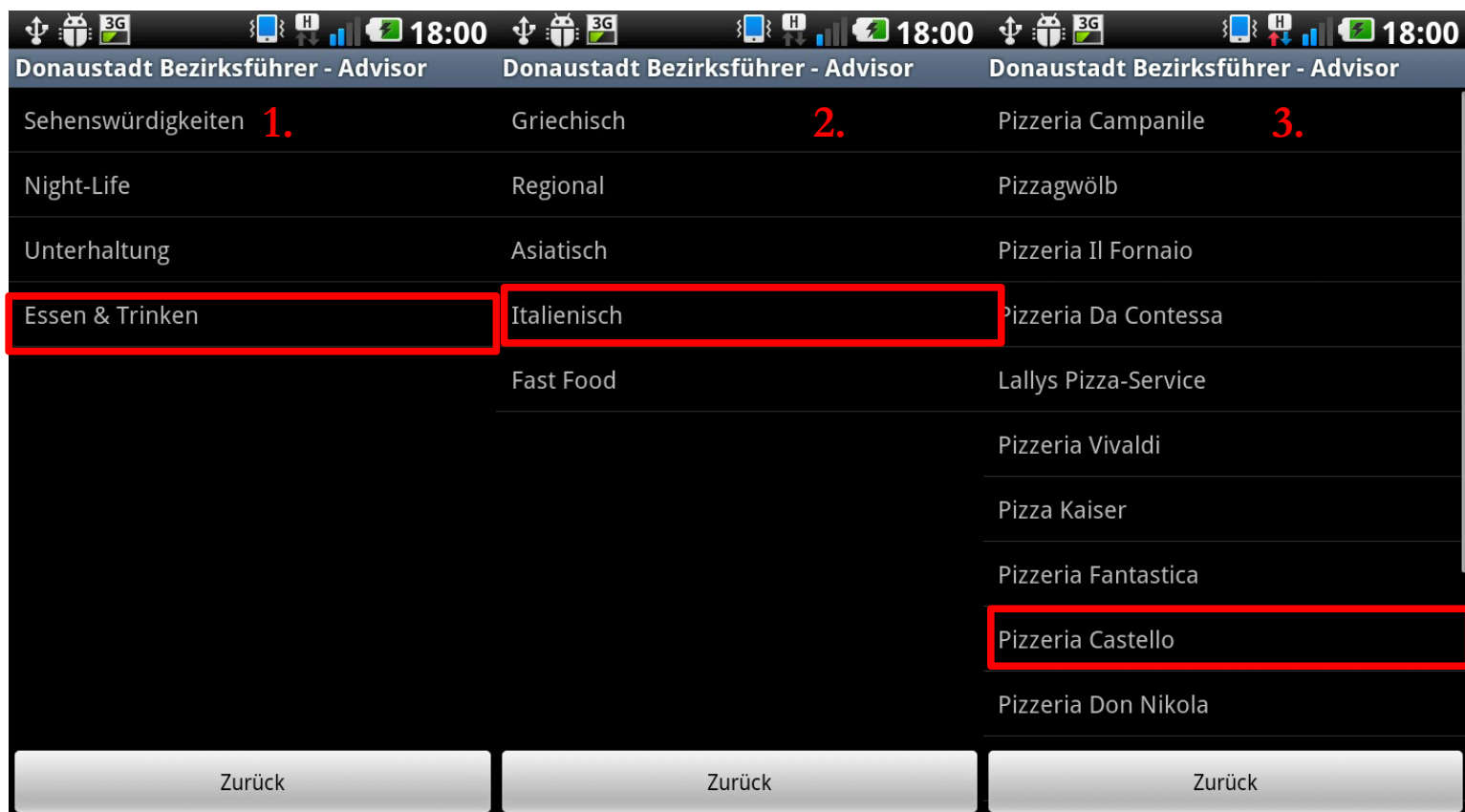
Durch einen einfachen Klick auf dem Menü Button des Mobiltelefons können Sie verschiedenen Optionen ein- und ausblenden.






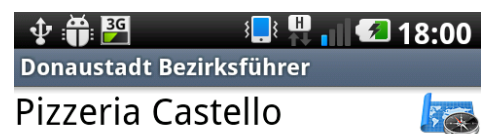
Advisor

Falls Sie den Namen des gesuchten Objektes nicht wissen oder nicht genau wissen nach was Sie suchen ist der Advisor genau das richtige für sie. Der Advisor führt Sie durch verschieden Schritte Ihr gewünschtes Objekt zu finden, dazu klicken Sie einfach auf die gewünschte Objekt-Kategorie.



Schnell einen guten Italiener finden, ganz einfach mit dem Advisor

Ebenso reicht auch hier, wie bei der Suche ein Klick auf den gewünschten Eintrag und eine detaillierte Information wird Ihnen präsentiert. Auch hier können Sie sich ganz schnell das Objekt auf der Karte anzeigen lassen mit einem Klick auf das Kartensymbol 



Beschreibung: Wir beliefern Sie nicht nur mit Pizzen sondern auch anderen frisch zubereiteten Fleischgerichten, Fischgerichten, Salaten, Nudel & Pasta - Gerichten, Suppen, und Dessert 's.

Adresse:

Esslinger Hauptstraße 81-87 A-1220 Wien
 Telefon: +43 1 774-88-75
 Öffnungszeiten: Dienstag - Samstag 11.00 - 14.30
 17.00 - 23.00
 Sonntag 11.00 - 23.00
 Montag Ruhetag
 Web: <http://www.castello.co.at/>

Zurück

Info

Sie haben sich jetzt nicht alles gemerkt, keine Angst. Informationen über die Funktionen erhalten Sie auch ganz bequem im Donaustadt Guide. Ein einfacher Klick auf das Informationssymbol und eine Hilfe öffnet sich und Sie bekommen die wichtigsten Informationen geliefert.

