



Universität Regensburg

Philosophische Fakultät III
Sprach-, Literatur- und Kulturwissenschaften
Institut für Information und Medien, Sprache und Kultur (I:IMSK)
Lehrstuhl für Medieninformatik

Einführung in die Anwendungsprogrammierung
Modul: MEI-M03.3
Sommersemester 2016

Eventastic

Never miss a date

Kristine Wolf
1735572
Medieninformatik und vergleichende
Kulturwissenschaft

4. Semester B.A.

Ägidiusstraße 17
85139 Wettstetten/ Echenzell
08406/1559
Kristine.Wolf@stud.uni-regensburg.de

Teresa Then
1720261
Medienwissenschaft und Medieninformatik

4. Semester B.A.

Fasanenweg 23
85356 Freising
08161/12287
Teresa.Then@stud.uni-regensburg.de

Abgegeben am 30.09.2016

Inhalt

1	Eventastic – Never miss a date	4
2	Technische Vorraussetzungen	5
2.1	Voraussetzungen	5
2.2	Externe Bibliotheken und Frameworks.....	5
2.2.1	Firebase Datenbank.....	5
2.2.2	Android Design Support Library	6
3	Das schwarze Brett für Veranstaltungen	7
4	Umsetzung	8
4.1	Design	11
4.2	Implementierung	18
5	Testing und auftauchende Probleme	21
6	Endzustand	23
6.1	Implementierte Features	23
6.2	Fehlende Features	23
6.3	Mögliche Erweiterungen	24
7	Projektmanagement.....	25

Abbildungen

Abb. 1: Startbildschirm Landscape	11
Abb. 2: Startbildschirm mit Menü-Dropdown	11
Abb. 3: AllEvents mit Standort-Permission	12
Abb. 4: AllEvents ohne Standort-Permission	12
Abb. 5: AllEventsInThisCity - München	13
Abb. 6: AllCities	13
Abb. 7: EventsInCity	13
Abb. 8: Event abgesagt.....	14
Abb. 9: AllInformationOfAnPart.Event	14
Abb. 10: AllInformationOfAnEvent zugesagt.....	14
Abb. 11: AllInformationOfAnEvent	14
Abb. 12: Tablet AddEvent mit Toast	15
Abb. 13 AddEvent mit Spinner	15
Abb. 14: CalendarView	16
Abb. 15: MyEvents.....	16
Abb. 16: Shared Message	16
Abb. 17: AboutTheApp	17
Abb. 18: Notification.....	17

1 Eventastic – Never miss a date

Immer über alle Veranstaltungen Bescheid wissen und das egal in welcher Stadt? Mit Eventastic bekommen Sie einen Überblick über alle veröffentlichten Events. Egal ob als Bewohner einer Stadt oder als Tourist lässt die App Sie nie wichtige Termine verpassen.

Sie erhalten einen Einblick in das tägliche Leben sowohl von Städten als auch von Orten und werden dadurch schnell Anschluss in einer fremden Region finden können.

Nehmen Sie an einer Veranstaltung teil und möchten dies Ihren Freunden mitteilen oder andere Menschen darauf hinweisen, so ermöglicht Eventastic das Teilen von Events auf verschiedensten Weisen wie beispielsweise über eine SMS oder über Ihre E-Mail. Das Teilnehmen an solchen Ereignissen hingegen geschieht ganz anonym. Sie sind nicht verpflichtet ein Account zu erstellen, sondern können ganz ohne Hindernisse die App nutzen und lieben lernen. Mit einem eingebauten Kalender haben Sie immer das nächste Event vor Augen und bekommen somit einen besseren Überblick. Sind Sie hin und wieder gestresst und vergessen somit möglicherweise, dass ein Termin bevorsteht, so kann die App Sie eine Stunde vor Beginn des Events daran erinnern. Möchten Sie an Veranstaltungen teilnehmen so liefert Eventastic Ihnen zunächst eine Liste an allen Terminen, die immer nach ihrem Veranstaltungstag und –zeitpunkt geordnet sind. Anschließend kann Eventastic, je nachdem ob Sie wollen, Ihnen Veranstaltungen in Ihrer Nähe zeigen. Sie können sich aber auch über Events in anderen Städten informieren. Möchten Sie eine eigene Veranstaltung veröffentlichen, so benötigen Sie nur gewisse Daten wie den Ort, das Datum, die Uhrzeit, einen Titel, den Typ des Events und zuletzt ein paar zusätzliche Informationen. Als Auswahl für den Typ bietet die App Ihnen einige Auswahlmöglichkeiten an, die die wichtigsten Bereiche abdecken. Zusammenfassend präsentiert sich Eventastic als schwarzes Brett für Veranstaltungen, das immer aktuelle Termine anzeigt. Mit Eventastic you will never miss a date.

2 Technische Voraussetzungen

2.1 Voraussetzungen

Eventastic ist eine reine Mobil-Anwendung. Ihre Daten bezieht die App von dem Cloud Service-Provider Firebase. Diese Firma ermöglicht eine schnelle und einfache Implementierung, um gute Apps zu entwickeln. Der Entwickler muss sich dabei keine Sorgen um die Infrastruktur machen, sondern konzentriert sich lediglich auf seine App. Damit es möglich ist eine Anwendung mit Firebase zu erschaffen, muss das Mobilgerät der Nutzer gewisse Faktoren erfüllen. Jedes Gerät muss eine Google Play Service Version ab 9.0 besitzen. In Bezug auf Eventastic müssen die Nutzer zusätzlich mindestens über die Version 4.1 des Android-Systems und einer aktiven Internetverbindung verfügen. Hinsichtlich der vollständigen Nutzung von allen Features der App, sollte das Handy zusätzlich GPS-Daten empfangen können. Ansonsten könnten keine Events in der Nähe des Nutzers angezeigt werden. Eventastic ist sowohl auf Smartphones als auch auf Tablets lauffähig, die das Android-Betriebssystem besitzen. Durch eine (manuelle) Installation der Anwendung auf einem Android-Gerät kann die App-Verwendung letztendlich gestartet werden.

2.2 Externe Bibliotheken und Frameworks

2.2.1 Firebase Datenbank

Um Daten zu speichern und zu beschaffen wurde das Feature „Realtime Database“ der Firma Firebase verwendet. Für die Entwicklung und das Testen einer Firebase-App sollte Android Studio über die neusten Versionen der beiden Tools, Google Play Services und Google Repository, verfügen. Nach Erstellen eines neuen Projektes auf der Internetseite „<https://firebase.google.com/>“ wird passend zur Android-App die Datei „google-services.json“ zum Herunterladen bereitgestellt, die die wichtigsten Konfigurationsinformationen zum eigenen Firebase-Account bereitstellt. Nachdem folgende JSON-Datei in den Modulwurzeln der App eingebunden wurde, mussten neben den Analytic- und Realtime-Database-Bibliotheken auch „google play service plugin“ im `build.gradle` des App-Levels ergänzt werden. Im Gradle des Projektes hingegen war nur das Einbeziehen von „google service“ nötig.

Die einbezogene, externe Datenbank selbst speichert ihre Daten in JSON-Format. Hierbei wird bei jedem Hinzufügen von Daten ein neuer Knotenpunkt erstellt. Somit entsteht ein Baumgerüst mit vielen Knoten. In Eventastic werden alle Event-Objekte in dem Oberknotenpunkt „events“ gespeichert. Jedes Objekt bekommt einen einzigartigen und individuellen Key.

Durch die Methode `push()` der `DatabaseReference`-Klasse werden die Objekte an die Datenbank geschickt und durch den `ChildEventListener` aus der Datenbank geholt. Dabei wird darauf geachtet, ob Daten geändert, gelöscht oder hinzugefügt werden. Mit Hilfe der Klasse `DataSnapshot` bekommt die App Daten zurück, die sie anschließend durch die `getValue()`-Methode und einer in Klammern definierten Klasse in ein Objekt umwandeln kann. Im Falle von Eventastic werden nur Events beachtet die noch bereits stattfinden werden oder am aktuellen Tag stattfinden. Das Löschen von Veranstaltungen geschieht nicht auf App-Seite. Diese Aufgabe bleibt denjenigen überlassen, die Zugriff auf den Firebase-Account haben. Mit der E-Mail-Adresse eventasticproject@gmail.com und dem Passwort eventasticproject2 ist ein Einloggen in den Account und somit eine Einsicht auf die Datenbank über der Internetseite <https://firebase.google.com/> möglich.

2.2.2 Android Design Support Library

Um nützliche UI Elemente zu verwenden, welche nicht im Framework enthalten sind, wurde die Android Design Support Library als externe Bibliothek eingebunden. Hierfür musste lediglich `compile 'com.android.support:design:23.4.0'` zu den dependencies im `build.gradle` ergänzt werden. Nun konnte in der `AboutThisApp`-Activity ein `TabLayout` verwendet werden. Mit Hilfe eines `ViewPager` kann der Nutzer zwischen verschiedenen Fragments über scrollable Tabs wechseln.

3 Das schwarze Brett für Veranstaltungen

Nach getaner Arbeit steigt oftmals das Bedürfnis mit Freunden nach Feierabend oder während des lang ersehnten Wochenendes etwas in der Umgebung zu unternehmen. Dabei werden von Stadt zu Stadt die unterschiedlichsten Veranstaltungen angeboten. Die Frage, die sich diesbezüglich allerdings stellt ist, wie findet man die geeignete Veranstaltung für sich und seine Freunde.



Soziale Netzwerke wie Facebook bieten in diesem Zusammenhang bereits die Möglichkeit für den User gezielt nach Feierlichkeiten zu suchen und die Liebsten einzuladen. Einen groben Überblick über sämtliche Veranstaltungen einer Stadt, die sowohl große Events als auch die kleineren eher gemütlichen Abende in einer Bar erfasst, wird aber auf derartigen Plattformen dagegen vergebens gesucht.



An dieser Stelle knüpft Eventastic mit seinem vielfältigen Angebot an Veranstaltungen an, das von Partys bis zu Comedy und Sportevents reicht. Egal ob jung oder alt, klein oder groß – jeder wird das Richtige für sich finden können. Der Nutzer dieser App kann sich einen Überblick über die aktuellsten Events in seiner Umgebung verschaffen und diese mit Freunden teilen.


Besonders auch für Touristen entpuppt sich Eventastic als Inspiration für die Suche nach dem nächsten Reiseziel. Anhand der großen Auswahl kann der Nutzer die passenden Veranstaltungen je nach Stadt einsehen und seine Entscheidung treffen. Hierbei ist vor allem die Möglichkeit, die App in vier verschiedenen Sprachen zu bedienen, zu erwähnen.

4 Umsetzung

Eventastic besteht aus vielen Komponenten, die ineinander verflochten sind. Neben *AsyncTasks*, *Adaptern*, *Datenbanken*, einem *Interface*, einem *Service*, *Fragments* und vielen weiteren Klassen stellen die Hauptpunkte der App die zwölf *Activites* dar. Durch das Einbeziehen einer *ActionBar* auf jeder *Activity* gelangt der Nutzer über diverse Menü-Icons zu den wichtigsten Funktionen.

- In der *MainActivity* hat der User über die Menü-Leiste in der *ActionBar* Zugriff auf den App-eigenen Kalender, die Einstellungen, eine *Activity* mit Informationen zur App, auf alle gepostete Events und auf seine eigenen Veranstaltungen, an denen er teilnehmen möchte. Über Buttons unterhalb des Logos gelangt er zu den zwei wichtigsten *Activities*: *AllEvents*  und *ParticipatingEvents* , die die beiden zentralen Seiten der App darstellen.
- Diese zwei *Activities* ermöglichen das Teilen und Verbreiten von Veranstaltungen, sowie das Teilnehmen und Speichern in einer eigenen Liste. Durch aussagekräftige Symbolen an der *ActionBar* ist ein Wechsel zwischen und zu diesen *Activities* von fast allen anderen *Activites* aus leicht möglich und für den User schnell verständlich. Die beiden Haupt-*Activities* besitzen überwiegend denselben Aufbau des User Interfaces, indem sie auf dieselbe Weise Events in einem *ListView* nach Eventbeginn anordnen. Hierbei werden zunächst nur die wichtigsten Daten zum Event aufgezeigt, die aus dem Titel, der Stadt, dem Eventtypen, das Datum und der Uhrzeit bestehen.
- Klickt man auf eines der Events, so gelangt man zur *AllInformationsOfAnEvent-Activity* bzw. zur *AllInformationsOfAParticipatingEvent-Activity*. Dort werden vollständigen Informationen über das jeweilige Event angezeigt. Über einen Button kann der Nutzer daran teilnehmen und das Event wird im *ListView* bei *ParticipatingEvents* aufgelistet. Ebenso besteht aber natürlich die Möglichkeit auf die gleiche Weise über einen Button wieder abzusagen. Der Nutzer erhält dabei durch entsprechende *Toasts* Feedback, was er durch das Klicken auf die jeweiligen Buttons bewirkt hat und fühlt sich schnell sicher im Umgang mit Eventastic. Beide Möglichkeiten kommunizieren dabei mit der internen Datenbank der App und fügen Einträge hinzu oder löschen welche.

- Durch das Setzen der Werte auf *true* in der `SettingsActivity`  erlaubt der Nutzer der App den Zugriff auf seine Position und das Senden von `Notifications`. Durch letztere wird er eine Stunde vor Eventbeginn an eine teilnehmende Veranstaltung durch eine entsprechende Benachrichtigung erinnert. Diese `Notifications` werden in der `AlertReceiver`-Klasse erzeugt und im `NotificationService` gescheduled. Die Positionsdaten hingegen werden erst in der `EventNearLocation`-Activity geholt und anschließend verarbeitet. Hierbei werden je nach Position die nächst gelegenen Veranstaltungen aus der externen Datenbank präsentiert. Genehmigt der Nutzer den Zugriff auf seine GPS-Daten allerdings nicht, so wird er beim Erscheinen der `AllEvents`-Activity jedes Mal um Erlaubnis gefragt. Verneint er dies so bleibt ihm der Zugriff auf die `EventNearLocation`-Activity verwehrt. Alle anderen Activities und somit User Interfaces bleiben ihm allerdings erhalten.
- In der `AllEvents`-Activity gelangt man über die `ActionBar` nicht nur zur `ParticipatingEvents`-Activity, sondern kann auch die Einstellungen ändern und Events mithilfe der `AddEvent`-Activity  hinzufügen. Hierbei werden gewisse Daten zum neuen Event verlangt. Erstens wird ein kurzer Titel der Veranstaltung benötigt. Außerdem kann der Nutzer mithilfe eines `AutoComplete` (welcher die größten Städte Deutschlands kennt) einen Ort hinzufügen oder einen eigenen wählen. Der Nutzer kann sich anschließend zwischen Konzert, Party, Festival, Theater, Comedy oder Sportevent entscheiden, welche über einen `Spinner` zur Verfügung gestellt werden. Fällt die neue Veranstaltung allerdings in keine dieser sechs Kategorien, so kann auch Sonstiges gewählt werden. Datum mit Uhrzeit werden über entsprechende `Picker` ausgewählt. Schließlich besteht die Möglichkeit zusätzliche Informationen, wie z.B. Eintrittspreise, kurze Beschreibung der Veranstaltung oder Ähnliches, hinzuzufügen. Nur dann, wenn alle Eingaben vollständig sind, kann der User durch Klicken auf den mittig platzierten Button den Event hinzufügen. Daraufhin wird es in die externe Datenbank aufgenommen und gespeichert, sodass jeder App-Nutzer Zugriff auf diese Veranstaltung hat.

- Durch einen Klick auf den Button „Zu allen verfügbaren Städten“ gelangt der Nutzer sowohl über die `AllEvents-Activity` als auch – wenn erlaubt - über die `EventNearLocation-Activity` auf in `AllPossibleCities-Activity`. Hier werden alle Städte alphabetisch aufgelistet, für welche bereits Veranstaltungen gepostet sind. Berührt man eine Stadt, leitet die App den User zur `EventsInCity-Activity` weiter, die wiederum nur diejenigen Events anzeigt, die in der ausgewählten Stadt stattfinden.
- Sowohl in `EventsInCity`, als auch in `EventNearLocation` gelangt man über die `ActionBar` entweder zu seinen eigenen teilnehmenden Events (`ParticipatingEvents-Activity`) oder kann neue Veranstaltungen hinzufügen (`AddEvent-Activity`).
- Möchte man einen Blick auf seine eigenen teilnehmenden Veranstaltungen werfen (`ParticipatingEvents-Activity`), gelangt der Nutzer über die `ActionBar` zurück zu seinen eigenen Events, kann aber auch durch das Kalender-Symbol  einen Blick auf den eigenen App-Kalender werfen (`CalendarActivity`). Hier wird das nächste stattfindende Event angezeigt, zu welchem der User zugesagt hat. Der jeweilige Termin wird im `CalendarView` markiert. Möchte man dieses Event mit seinen Freunden oder Mitmenschen teilen, so findet man hierfür den vertrauten `SharingIcon` in der `ActionBar`. Sollte der User kein anstehendes Event auf seiner Liste stehen haben, so hat er die Möglichkeit direkt aus der `CalendarActivity` heraus zur `AllEvents-Activity` zu gelangen.

4.1 Design

Eventastic unterstützt Portrait-, sowie Landscape-Format. Außerdem wurden Layouts für verschiedene Bildschirmgrößen realisiert, um die App auf Smartphones, als auch auf Tablets uneingeschränkt nutzen zu können.

Im Folgenden werden die einzelnen Activities der App bezüglich ihres Layouts kurz vorgestellt, die Beschreibung wird durch entsprechende Screenshots unterstützt.

▪ Startbildschirm: MainActivity

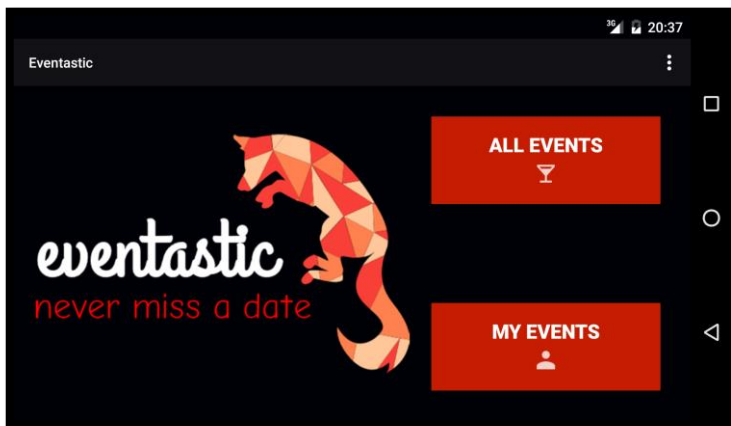


Abb. 1: Startbildschirm Landscape

Der Startbildschirm (MainActivity) gliedert sich in zwei große Bereiche. Im oberen (bzw. im Landscape-Format linken) Abschnitt des Bildes befindet sich das Logo der App mit dem Slogan „Eventastic – you will never miss a date“. Darunter (bzw. daneben) sind zwei

große Buttons platziert, einer leitet den User zu allen Events (AllEvents-Activity) weiter, der andere zu seinen gespeicherten Events (MyEvents-Activity). Auf den Buttons befindet sich zusätzlich zu einem kurzen Text das jeweilige Icon, welches in anderen Activities in der ActionBar als Menü-Icon dient, um zu eben zu jenen Activities zu gelangen. Die beiden Buttons sind gleich aufgebaut und die Buttonfarbe nimmt den Rotton des Logos auf. Zusammenfassend lässt sich sagen, dass der Startbildschirm ansprechend gestaltet, aber dennoch möglichst schlicht gehalten ist, um den User nicht mit allzu vielen Aktionsmöglichkeiten zu überfordern. Hier wurde auf Ästhetik und minimales Design gesetzt.

Rechts oben in der ActionBar kann das versteckte Menü geöffnet werden, die den User zu anderen wichtigen Activities der App führt, welche interessant sind, sie aus der MainActivity heraus zu öffnen.

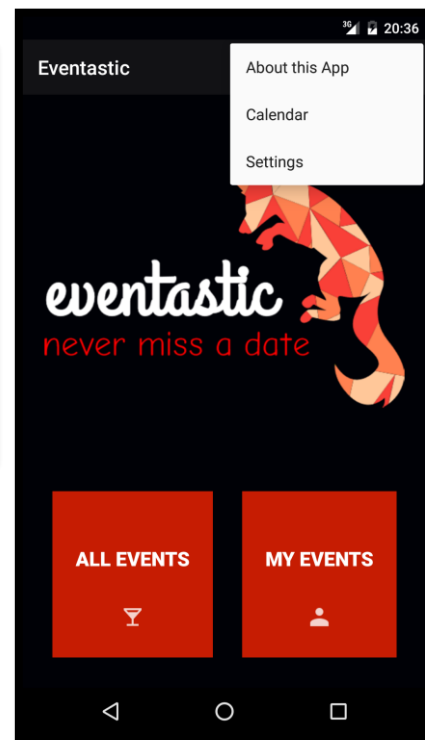


Abb. 2: Startbildschirm mit Menü-Dropdown

- AllEvents

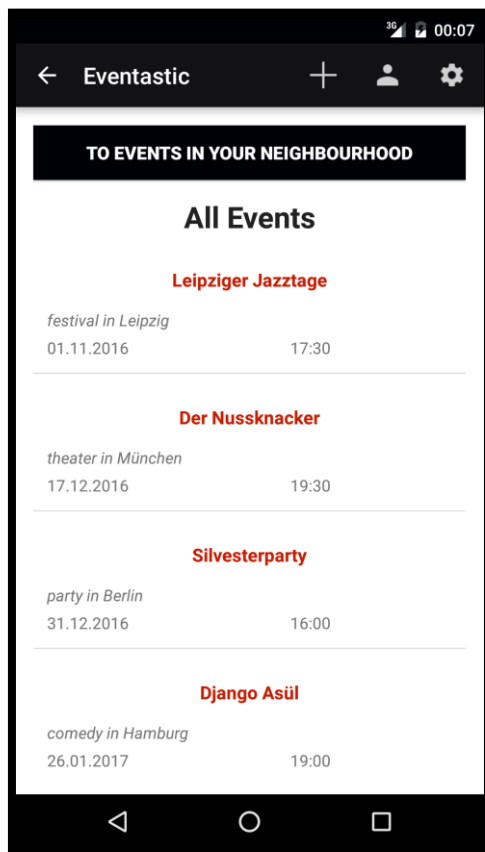


Abb. 3: AllEvents mit Standort-Permission

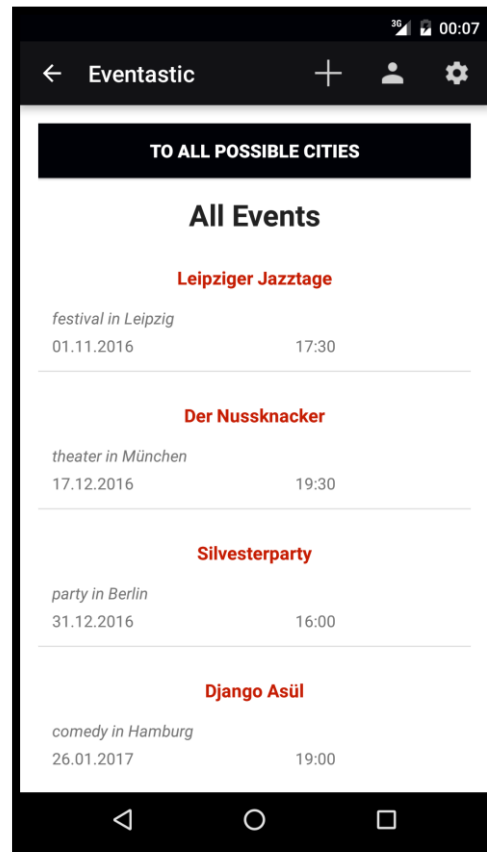


Abb. 4: AllEvents ohne Standort-Permission

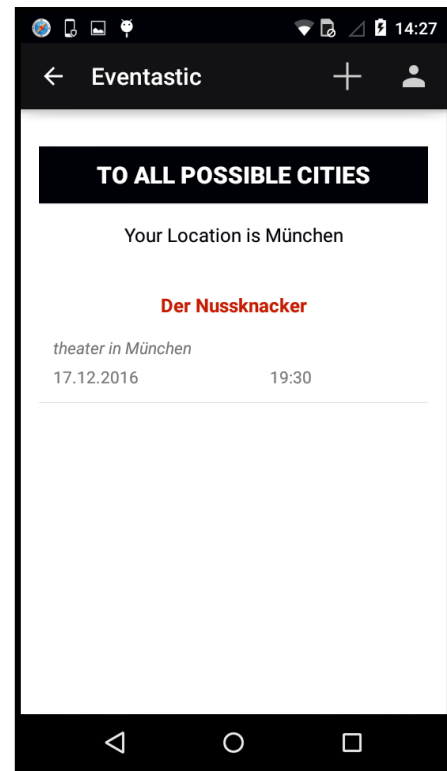
In dieser Activity werden alle Events aus der online-Datenbank in einem ListView angezeigt, wie der schwarze Listentitel im oberen Bereich des Bildschirms beschreibt. Falls der User der App die Standort-Permission erteilt hat, so hat er die Möglichkeit über den oben platzierten Button zur `EventsNearLocation`-Activity zu gelangen, wo ihm Veranstaltungen in seiner Umgebung angezeigt werden. Hat er den Zugriff jedoch verwehrt, wird der Buttontext verändert angezeigt und nach Berühren des Buttons wird man zur `AllCities`-Activity weitergeleitet. Für die `ListItems` des `ListView` ist für die Schriftfarbe der Eventtitel derselbe Rotton verwendet worden, wie bereits in der `MainActivity`. Die zusätzlichen Informationen werden in einem sanften Grau dargestellt. Dieses `ListView`-Layout findet sich in vielen weiteren Activities wieder.

In der `ActionBar` ist ein Plus-Icon platziert, über welchen der User zur `AddEvent`-Activity gelangt, wo er Einträge zu diesem `ListView` hinzufügen kann. Außerdem kommt er über das bereits bekannte User-Symbol zu seinen eigenen Events. Das Zahnrad-Symbol leitet zu den Settings weiter. Dies sind wohl die drei wichtigsten Activities, die der User aus der `AllEvents`-Activity heraus aufrufen will, und wurden deshalb in der `Actionbar` platziert.

▪ AllEventsInThisCity

Diese Activity zeigt dem Nutzer nur jene Events an, die in der Nähe seines aktuellen Standorts stattfinden werden. Wo er sich befindet, sagt ihm das Label oberhalb des ListViews. In dem Beispiel des Screenshots befindet sich der User in München. Für diese Stadt wird ihm ein Event *Der Nussknacker* angezeigt. Über einen Button kommt man zu AllCities. In der ActionBar werden Menü-Icons zu AddEvent und MyEvents zur Verfügung gestellt.

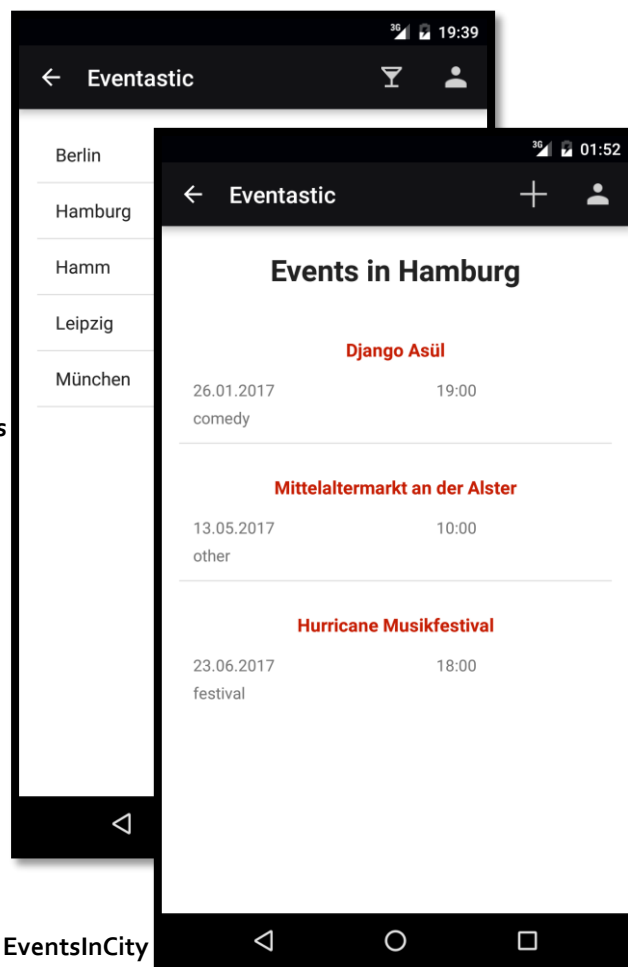
Abb. 5: AllEventsInThisCity - München



▪ AllCities

In dieser Activity wird eine Liste angezeigt, welche allen Städten auflistet, für welche bereits Veranstaltungen in der Datenbank eingetragen sind. Durch Auswählen einer Stadt gelangt man zur EventsInCity-Activity.

Abb. 6: AllCities



▪ EventsInCity

Der Titel zeigt dem User an, für welche Stadt er sich zuvor entschieden hat. Hier ist ein ListView mit allen Events in der zuvor ausgewählten Stadt implementiert. Über die ActionBar kann unter anderem ein neues Event hinzugefügt werden.

Abb. 7: EventsInCity

- AllInformationOfAnEvent u. AllInformationOfAnParticipatingEvent

Wählt der User eines der angezeigten Events aus, kommt der zu AllInformationsOfAnEvent. Dort werden ihm detaillierte Angaben zu dieser Veranstaltung gegeben. Mit dem Button fügt er das Event zu seinen Events hinzu und bekommt durch den Toast und durch Änderung in Buttontext und -farbe entsprechende Rückmeldung.

Wählt der User aus MyEvents eines seiner zugesagten Events aus, kommt er zu AllInformationOfAnParticipatingEvent. Diese Activity ist identisch aufgebaut, nur der Buttontext ist unterschiedlich. Sagt der User ab, wird das Event aus der internen Datenbank gelöscht und die Activity wird beendet. Wiederum erscheint ein entsprechender Toast.

Abb. 11: AllInformationOfAnEvent

Abb. 10: AllInformationOfAnEvent zugesagt

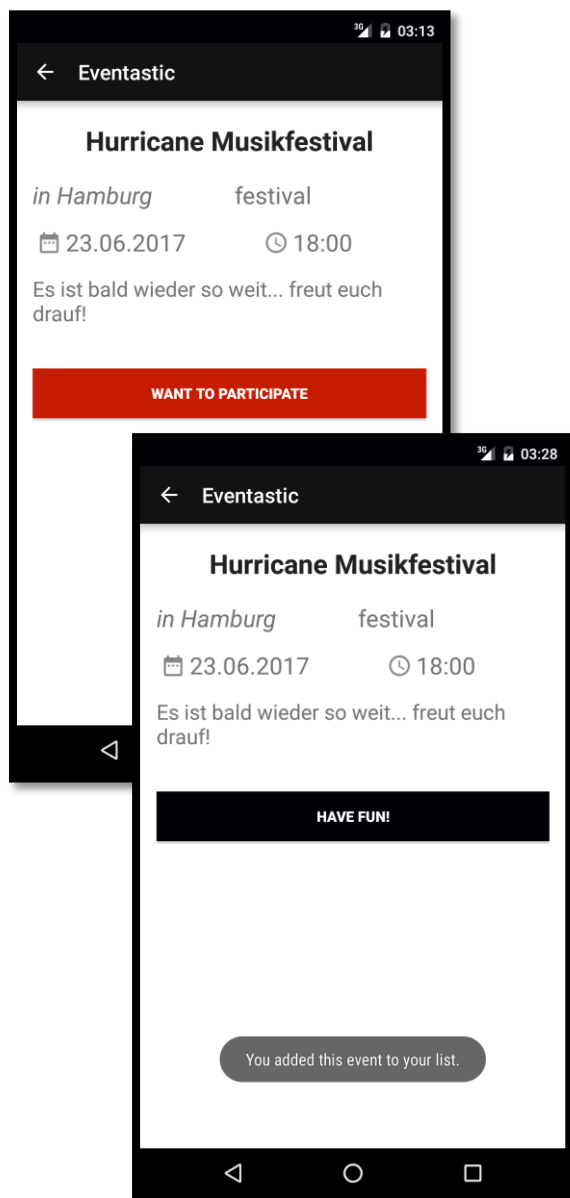
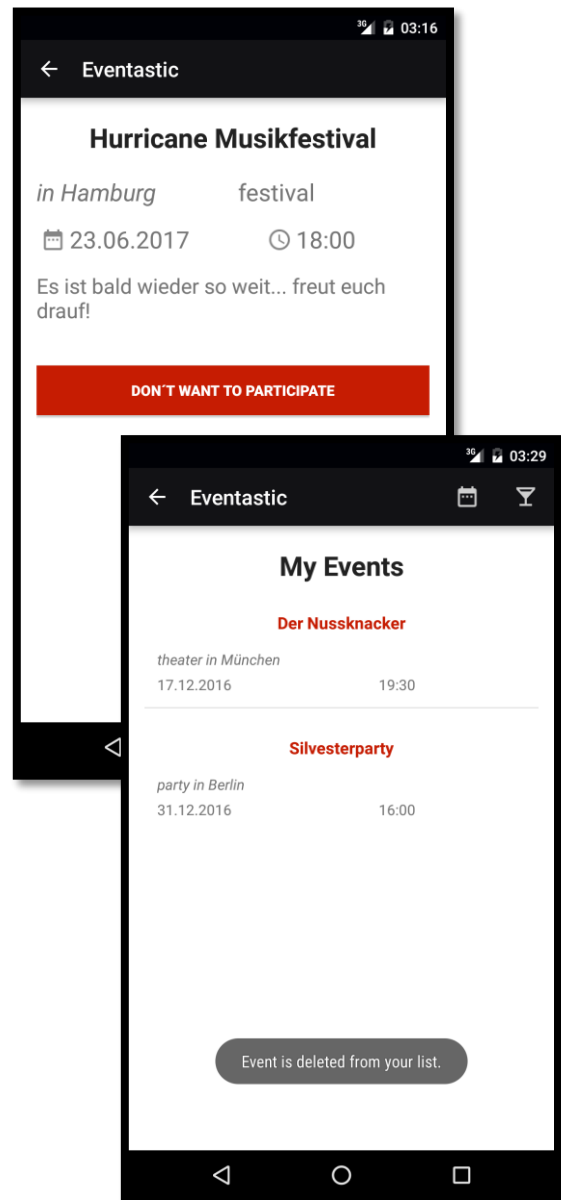


Abb. 9: AllInformationOfAnPart.Event

Abb. 8: Event abgesagt



▪ AddEvent

Über die AddEvent-Activity kann der User Events zur online Datenbank hinzufügen, sodass sie allen Eventastic-Usern zur Verfügung stehen und sie daran teilnehmen können. Es müssen einige Felder hierfür ausgefüllt werden. Für diese Aufgabe erhält der User ausreichend Unterstützung durch die App. Wie in Abb. zu sehen ist, wurde ein AutoComplete für die Eingabe der Stadt implementiert. Um den passenden Eventtyp auszuwählen, steht dem User ein Spinner zur Verfügung. Für das Festlegen von Datum und Uhrzeit sind Picker implementiert. Im Landscape-Layout wurde der Platz in die Breite genutzt, indem man die erste und zweite Zeile der EditText-Views zusammengefügt hat. Der Button unterhalb fügt das Event zur Datenbank hinzu und der User wird durch einen entsprechenden Toast auf diese Aktion hingewiesen.

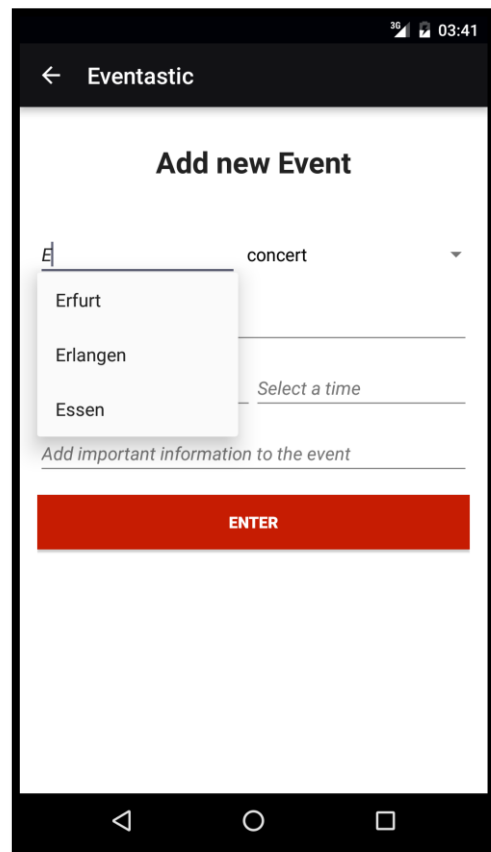
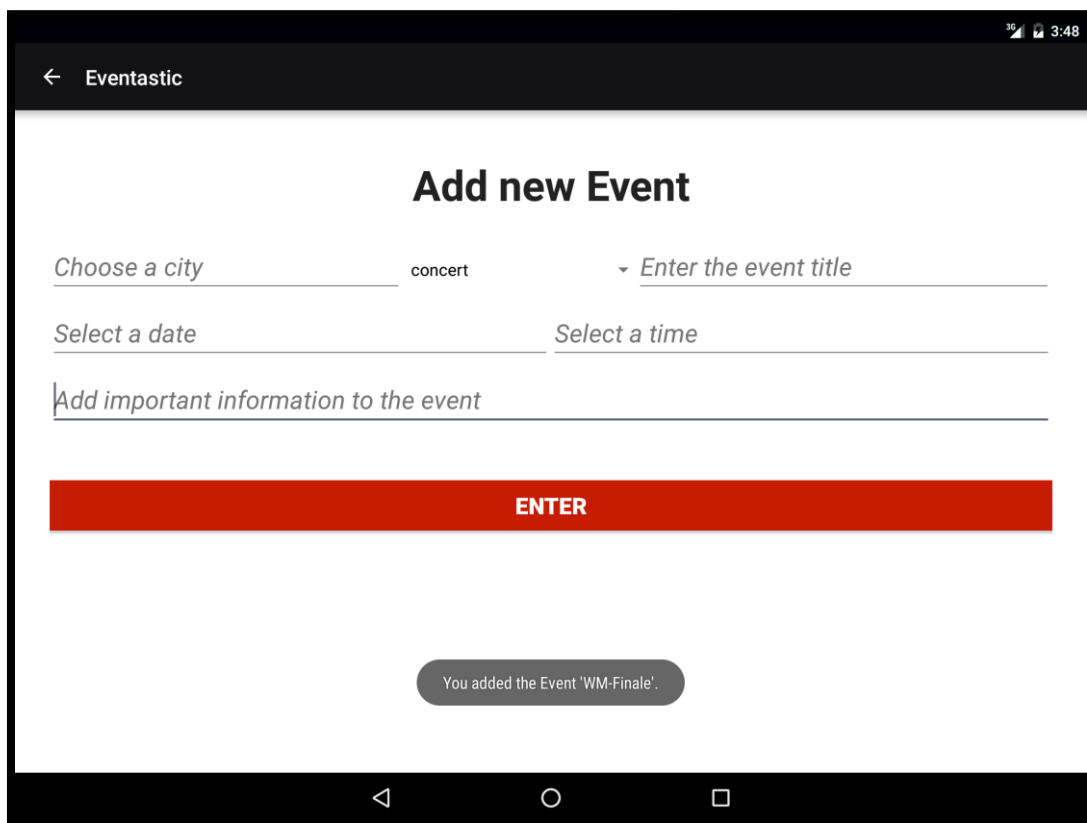


Abb. 13 AddEvent mit Spinner

Abb. 12: Tablet AddEvent mit Toast



▪ MyEvents

Diese Activity zeigt, wie der Titel im oberen Bereich sagt, alle Events an, die der User in der internen Datenbank gespeichert hat. Das Layout ist – bis auf den fehlenden Button – identisch mit dem Layout der AllEventsActivity, um Konsistenz zu bewahren. Über die ActionBar hat man Zugang zur CalendarActivity, sowie zu AllEvents.

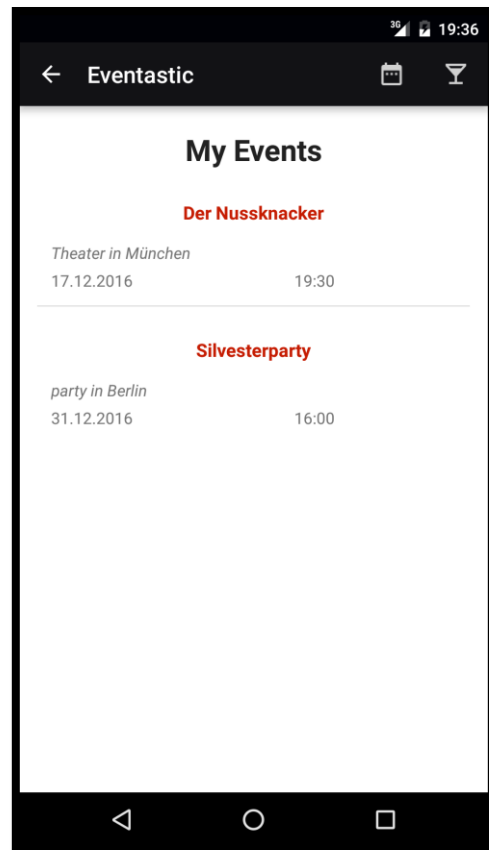


Abb. 15: MyEvents

Abb. 14: CalendarView

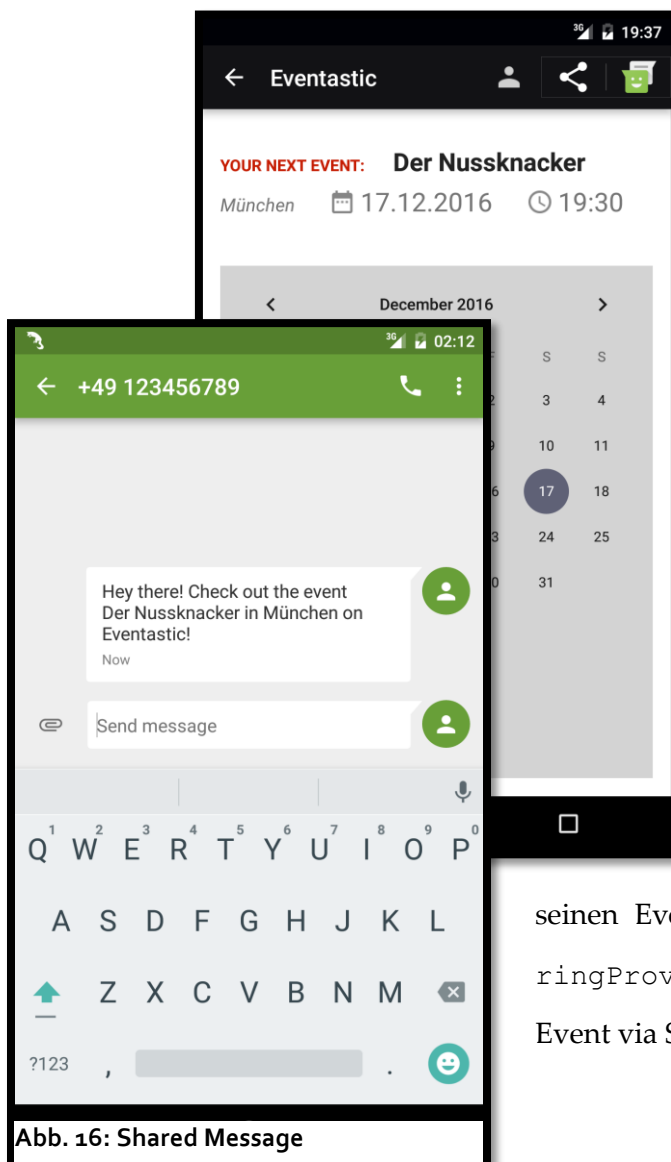


Abb. 16: Shared Message

▪ CalendarActivity

Hier findet der Nutzer Informationen über das nächste Event, zu welchem er zugesagt hat. In einem CalendarView wird der Tag auf Datum der Veranstaltung gesetzt.

Über die ActionBar kommt er zu all seinen Event. Außerdem wird hier mittels SharingProvider die Möglichkeit angeboten, dieses Event via SMS, WhatsApp, Mail, ... zu teilen.

▪ AboutTheApp

Hier wurde mithilfe der Android Design Support Library ein Tab-Layout erstellt. In 4 SlidingTabs werden Informationen über *Was kann die App?*, *Wie funktioniert sie?*, *Was kann ich tun?* Und *Kontakt* bereitgestellt. In diesem letzten Tab hat der Nutzer die Möglichkeit durch Klick auf den Button einen Email-Client auszuwählen und die Developer zu kontaktieren.

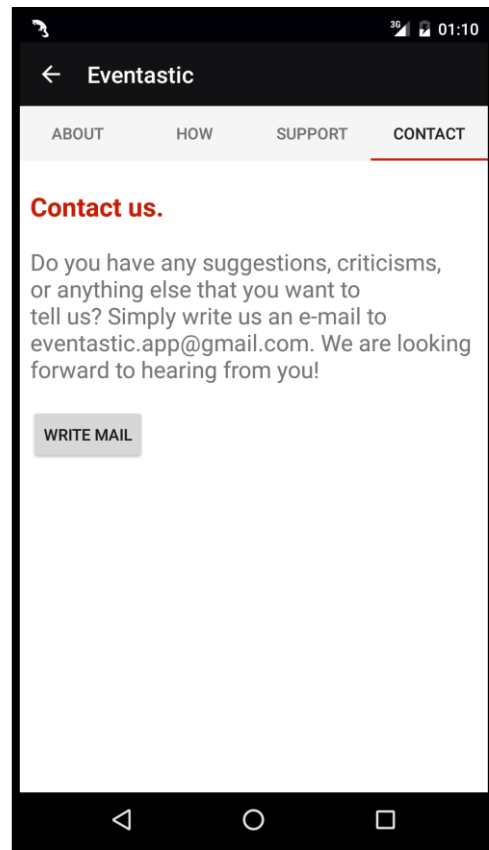


Abb. 17: AboutTheApp

▪ Notifications

In der Notification ist das Fuchs-Symbol aus dem Logo aufgegriffen, sodass der User sofort weiß, von welcher App diese Benachrichtigung kommt. Es wird ein kurzer Text angezeigt, der sich im BigView leicht verändert. Von dieser Notification aus bieten sich dem User zwei Aktionen: er kann entweder zur `CalendarActivity` oder zu `MyEvents` navigieren.

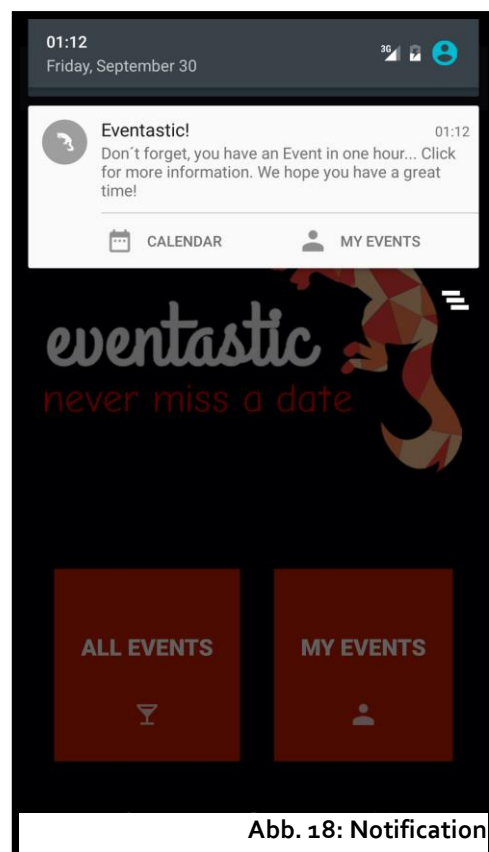


Abb. 18: Notification

4.2 Implementierung

Die Hauptaufgaben der App ist das Erstellen und Teilnehmen an Events. Hierfür sind vor allem sowohl eine externe als auch eine interne Datenbank nötig. Dementsprechend wurde zunächst daran gearbeitet eine externe Datenbank zu finden und diese mit der App zu verbinden. Nach langer Recherche stießen wir auf den Cloud-Provider Firebase, der mit seinem kostenlosen Angebot, den vielen Features und der simplen Handhabung überzeugen konnte. Hierbei mussten wir zunächst einen Google Account erstellen, um anschließend ein neues Projekt zu beginnen. Im Falle von Eventatsic wurde des Weiteren ein neues Android Projekt begonnen, wobei der Package-Name benötigt wurde. Um eine Verbindung zwischen diesen beiden Komponenten zu Erstellen und die Realtime Database des Anbieters zu verwenden, fügten wir die nötige JSON-Konfigurationsdatei, die Bibliotheken und das Google Service Plugin in unser Projekt ein. Dadurch konnten gewisse Bereiche wie das Erstellen einer Internetverbindung und das Konvertieren der Daten in das benötigte Format bereits erfüllt werden. Nachdem eine Verbindung zum Server bestand, wurde zunächst daran gearbeitet neue Events zum Server zu schicken und diese dort in JSON-Format zu speichern. Hierbei mussten wir die Zugangsberechtigung der Datenbank auf öffentlich ändern. Als dieses Vorhaben funktionierte, wurde daran gearbeitet, die Daten nun aus der Datenbank zu holen und aufgelistet nach Veranstaltungstermin in einem ListView anzuzeigen.

Anschließend verknüpften wir die Datenbanken auf eine bestimmte Weise miteinander. Beim Teilnehmen an einen Event werden die jeweiligen Daten in der relationalen, internen Datenbank gespeichert und aus dieser wieder gelöscht, falls der Nutzer sich um entscheidet oder die Veranstaltung bereits stattgefunden hat. Zu diesem Zeitpunkt war die `ParticipiatingEvents-Activity` die einzige Komponente, die Daten aus der internen Datenbank holte und auf dieselbe Weise auflistete. Nachdem diese komplette Kommunikation in der App vorhanden war und funktionierte, kümmerten wir uns um weitere Features wie das Auflisten der Städte und das Erstellen eines eigenen Kalenders.

Hinsichtlich der Städte erwies sich mit der Zeit das Erstellen einer statischen Klasse, in der alle Events aus dem Server gespeichert werden, als vorteilhafter und verkürzte das Warten auf diese Daten. Mit Hilfe von `AllEventsPuffer` werden alle vorhandenen Städte aus dem Array gefiltert und alphabetisch angeordnet. Berührt der Nutzer eine Stadt, so hat er Einsicht in alle Events dieser Stadt. Klickt er auf einen dieser Events gelangt er zu einer Activity, die alle

Daten über das Event aufzeigt. Dieser Vorgang wiederholt sich auch in folgenden Activities: `AllEvents`, `ParticipatingEvents`, `EventNearLocation`.

Geplant war eine `CalendarActivity`, welche in einem App-eigenen Kalender einen Eintrag für jedes Event erstellt, zu dem der User zusagt. Dafür sollte ein `CalendarView` verwendet werden. Es stellte sich jedoch heraus, dass man dem default `CalendarView` von Android keine Events hinzufügen kann. Schließlich haben wir uns darauf geeinigt, dennoch dieses View zu verwenden und lediglich das Datum des nächsten Events zu markieren. Es wurden Views für die Daten dieses Events zum Layout hinzugefügt. Somit fungiert diese Activity weniger als wirkliche Kalender-Activity, sondern sie stellt eine Möglichkeit für den User dar, genaue Informationen über das nächste Event zu erhalten. Außerdem wurde ein `ShareActionProvider` implementiert, welcher es dem User erlaubt, das Event via SMS, Mail o.ä. zu teilen.

Die Implementierung der Notifications war eine Entwicklungsphase, während welcher wir auf kleinere Bugs gestoßen sind. Mithilfe der offiziellen Android Developer Dokumentation konnte das Layout der Notification Schritt für Schritt erstellt werden. Die Notification bietet dem User zwei Aktionen an: er kann entweder zur `CalendarActivity` oder zur `ParticipatingEvents` navigieren. Das Auslösen zum richtigen Zeitpunkt und auch von außerhalb der App, stellte ein Problem dar, für das wir eine Weile brauchten. Schließlich wurde die Methode `scheduleNotification(int i)` in die `onStartCommand`-Methode der Klasse `NotificationService` ausgelagert und anstatt einem festen Wert die Variable `i` dem `PendingIntent pendingIntent = PendingIntent.getBroadcast(getApplicationContext(), i, intent, PendingIntent.FLAG_UPDATE_CURRENT);` übergeben. Für diese letzte Codeänderung bekamen wir den Tipp von einem der Tutoren.

Nächster Schritt war das Erstellen der `AboutTheApp`-Activity, in welcher mithilfe der Android Design Support Library ein scrollable `TabLayout` implementiert wurde. Hierzu wurden vier Fragments zu einem `ViewPagerAdapter` hinzugefügt. Das vierte Fragment verfügt über eine Möglichkeit per Buttonklick einen Email-Client zu öffnen und direkt mit den Developern in Kontakt zu treten.

Dann wurde am Auslesen und Verarbeiten der Locationdaten des Nutzers geschrieben. Hierbei werden die Daten durch den `LocationManager` und `LocationListener` geholt, die sich beide in der Klasse `LocationController` befinden. Durch den von außen übergebe-

nen `LocationUpdateListener` wird die Position übergeben. In der `EventNearLocation-Activity`, die dieses Interface implementiert, werden anschließend die Daten ausgelesen und mit Hilfe des `GetNearestCityAsyncTasks` die nächste Stadt ermittelt. Dabei werden die mitgegebenen Adressdaten der Städte mit den Latitude- und Longitude-Daten der Benutzerposition verglichen und der kleinste Abstand durch den Satz des Pythagoras berechnet. Die Werte, die vom `AsyncTask` zurückgeliefert werden, sind alle Events in dieser Stadt, die in einem Array zusammengefasst werden. Diese werden anschließend über den `CityAdapter` in einem `ListView` angezeigt. Die `Locationpositionen` aller Städte hingegen werden in der `onCreate-Methode` der Activity durch das Starten des `AsyncTasks` `GetAddressesOfCities` ermittelt. In diesem `AsyncTask` werden zunächst alle möglichen Städte aus `AllEventsPuffer` ermittelt, um im Anschluss die Address-Daten durch den Geocoder zu bekommen und zusammengefasst in eine `ArrayList` zurückzuliefern.

Nachdem dies alles funktionierte wurde die `SettingsActivity` vollendet, die dem Nutzer die Möglichkeit bietet, den Zugriff auf seine Position und das Schicken von Benachrichtigungen in Form von Erinnerungen zu verweigern. Erstere wird dabei immer beim Erscheinen des User Interfaces von `AllEvents` überprüft. Wurde keine Erlaubnis gegeben, so fragt die App den Nutzer nach diese. Letztere hingegen wird zu Beginn der `MainActivity` und somit beim Starten der App überprüft und nur bei Erlaubnis werden `Notifications` gesetzt. Die App selbst fragt allerdings hier nicht nach einer Genehmigung.

Des Weiteren wurde im Laufe der Entwicklungsphase immer wieder die Navigation in der `ActionBar` ergänzt, gewisse Klassen wie `ChangeDateFormat` und `ContemporaryDate` erstellt, die wiederholende Codepassagen zusammenfassen, das Design angepasst und einige zusätzliche Features erarbeitet.

Schließlich neben der `default strings.xml` wurden ebenfalls Strings auf Deutsch, Spanisch und Französisch erstellt und somit `multiple language support` erreicht. Zum Ende hin wurde hauptsächlich am Layout gearbeitet, um verschiedene Bildschirme zu unterstützen. Es wurden Layouts für `Portrait` und `Landscape` implementiert, sowie für Tablets mit einer Mindestbreite von `800dp`. Zusätzlich führten wir einige User-Tests durch, wodurch wir auf die ein oder andere Ungereimtheit gestoßen sind, die schnell behoben werden konnten. Letzter Schritt war das Überprüfen des Codes auf `Magic Numbers` bzw. -Strings. Außerdem haben wir die Kommentare überarbeitet und ergänzt.

5 Testing und auftauchende Probleme

Durch Usability-Tests während der Entwicklungsphase konnten kleinere Nutzungs-Probleme aufgedeckt werden. Erwähnenswert sind die Folgenden:

- Erstens wurde daran gearbeitet, die Navigation durch die App hindurch zu verbessern. Mithilfe von sprechenden, wiederkehrenden Menü-Icons sollte es nun leichtfallen, sich von einer Activity zur anderen zu klicken und einen schnellen Überblick über Eventastic und die Funktionen dieser App zu bekommen.
- Weiterhin wurden entsprechende Toasts, sowie Soundeffekte hinzugefügt, um den Nutzer sowohl visuell, als auch auditiv auf Vorgänge innerhalb der App aufmerksam zu machen, z.B., wenn er ein neues Event erstellt und zur Liste hinzufügt. In der `MainActivity` hat man Zugriff auf die `AboutThisApp-Activity`. Dort findet der Nutzer unter anderem Informationen zur Nutzung. Diese dient gewissermaßen als Bedienungsanleitung, falls die Anwendung an der einen oder anderen Stelle doch noch Fragen aufwirft.
- Außerdem stellte das Auslösen der `Notifications` zum korrekten Zeitpunkt ein Problem, an welchem wir eine Weile saßen, das aber schließlich gelöst werden konnte, indem die Methode `set (...)` ; durch `setExact (...)` ; ausgetauscht wurde.
- Des Weiteren stellte sich in diversen Tests heraus, dass Probleme beim Anzeigen des richtigen Events in der `CalendarActivity` bestanden. Falls der Nutzer zu einem Event für den heutigen Tag zugesagt hat, das aber bereits begonnen hat, wurde dennoch dieses Event als nächstes Event angezeigt. Durch Änderungen im Code konnte dieser Fehler schließlich behoben werden.
- Ein weiterer Mangel war das fehlerhafte Anzeigen des Eventtyps in der Sprache, welche auf dem Gerät eingestellt ist. Erstellte ein User, der Englisch als Sprache festgelegt hat, ein Event vom Typ *concert*, so wurde es in die Datenbank mit aufgenommen und für andere User, welche nicht die default strings.xml verwenden, ebenso als *concert* (nicht etwa als *Konzert* oder *concierto*) angezeigt. Durch entsprechende Anpassungen konnte dieser Fehler behoben werden.
- Zusätzlich stellten alle Activities ein Problem dar, die Daten aus der externen Datenbank holen sollten. Das Warten auf die gewünschten Daten dauerte zu lange und beanspruchte dabei viel Akku. Um dies zu umgehen wurde dementsprechend eine statische Klasse an-

gelegt (`AllEventsPuffer`), die alle möglichen Events speichert und für die übrigen Activities zur Verfügung stellt. Dadurch müssen die Daten nur einmal in der `AllEvents`-Activity aus der Firebase-Datenbank geholt werden und gewährleisten somit einen verantwortungsvollen Umgang mit der Internetverbindung.

- Zu allerletzt tauchten häufig Probleme in der `EventNearLocation`-Activity auf, die aufgrund der Überlastung im Main-Thread das User Interface für längere Zeit blockierten und somit zu einer „Activity not responding“-Meldung führte. Durch das Auslagern einiger Aufgaben in `AsyncTasks` konnte dieses Problem allerdings weitgehend gelöst werden.

6 Endzustand

Mit Blick auf den finalen Zustand wurden alle Hauptfeatures im Kern umgesetzt, wenngleich die Umsetzung selbst in einigen Punkten anders als zu anfangs vorgesehen stattgefunden hat.

6.1 Implementierte Features

Neben den sechs Kernfeatures 1) Daten aus dem Internet übertragen, anzeigen und auswerten, 2) Daten extern und 3) intern speichern, 4) Verwenden von Sensor-Daten, 5) Navigation durch eine ActionBar und 6) Adaptive Layout für verschiedene Bildschirmgrößen, sowie -orientierungen wurden einige weitere Features in dieser App eingebaut. Hierbei sind die Verwendung von Notifications, Toasts und Dialogen zu erwähnen, sowie das Implementieren eines TabLayouts mithilfe der Android Design Support Library. Des Weiteren wurden folgende Features verwendet: Fragments, Listener, ShareActionProvider, SharedPreferences, Spinner, AutoComplete, Time- & DatePicker. Außerdem werden neben der englischen default strings.xml drei weitere Sprachen bereitgestellt.

6.2 Fehlende Features

Für den Einbau von Google Maps haben wir uns bewusst dagegen entschieden. Ursprünglich sollte dem User eine derartige Nutzung im Rahmen von Eventastic die genaue Ortsangabe einer Veranstaltung ermöglichen und darüber hinaus als Navigation dienen. Hinzu kommt, dass das Aufzeigen des eigenen Standortes durch Google Maps die Auflistung der am nächstgelegenen Event erleichtert, indem diese ihrer Entfernung nach angeordnet werden können. Demgegenüber war es uns aber ein Anliegen private Informationen wie die Adresse nicht zu veröffentlichen, sondern auf Anonymität zu setzen. Mithin werden bewusst nur grobe Angaben zum Veranstaltungsort, insbesondere die Stadtangaben, sowie einige wichtige Informationen zu dem jeweiligen Event preisgegeben. Ist ein Veranstaltungsort besonders schwierig zu finden können präzisere Hinweise in der Beschreibung ergänzt werden. Wird dennoch eine private Adresse angegeben, stellt unser Zugriff auf die externe Datenbank sicher, dass derartige Informationen zügig wieder entfernt werden können. Insgesamt betrachtet soll der Fokus unseres Projekts darin liegen, Veranstaltungen aufzustellen gegebenenfalls mit Freunden zu teilen oder an besagten Events teilzunehmen.

6.3 Mögliche Erweiterungen

Eventastic könnte um verschiedene Features erweitert werden, wodurch der Nutzungskontext größer werden würde. Denkbar sind folgende Implementierungen:

- Das Anlegen einer privaten Event-Historie von vergangenen Veranstaltungen. Der Nutzer könnte eigene Fotos und einen kurzen Erinnerungs-Text zu einem Event hinzufügen, welches daraufhin in einer Art Tagebucheintrag gespeichert wird. So würde die App zusätzlich als digitales Event-Tagebuch dienen.
- Das Übertragen oder automatische Synchronisieren von teilnehmenden Events in den Geräte-eigenen Kalender. Somit können Überschneidungen mit anderen Terminen vermieden werden.
- Die Möglichkeit des Filterns nach bestimmten Veranstaltungskategorien oder Eingrenzen des Veranstaltungszeitraums. Plant man beispielsweise in einem Monat einen Berlin-Urlaub und möchte sich bereits heute über die dortigen Partys während eben diesem Aufenthaltszeitraum informieren, wäre dies sehr nützlich.
- Die Möglichkeit, über mehrere Tage andauernde und sich wiederholende Events zu erstellen. Ein Barbesitzer könnte somit schnell und einfach für jeden Samstag zum Bundesliga-Nachmittag in seiner Kölner Kneipe aufrufen.
- Eine Verbindung zwischen beiden Datenbanken erstellen, sodass auch die intern gespeicherten Events im Falle einer Änderung oder Löschung angepasst werden.
- Das freiwillige Hinzufügen von Fotos oder Videos beim Erstellen eines Events, um das Interesse an diesem Event zu stärken. Zumal Bilder und Videos mehr ausdrücken können als Texte.

7 Projektmanagement

Um gemeinsam an diesem Projekt zu arbeiten nutzten wir Github, wie auch in den Vorlesungsfolien empfohlen. Nach anfänglichen Schwierigkeiten haben wir uns mit diesem Online-Dienst gut zurechtgefunden und gerne damit gearbeitet. Durch regelmäßige Pushes und ausführliche Kommentare konnte man die Arbeit des anderen leicht nachvollziehen und schnell daran anknüpfen. Über soziale Netzwerke hielten wir Kontakt und tauschten uns über den Fortschritt, Probleme und Ideen bezüglich des Projekts aus.

Leider konnten wir es nicht mehr einrichten, uns vor Projektbeginn persönlich zu treffen, da Teresa seit Ende Juli im Auslandssemester in Südamerika ist. Hinzu kommt, dass wir das Projekt zu zweit, anstatt wie ursprünglich geplant zu dritt bearbeiten mussten. Trotz dieser Widrigkeiten haben wir es geschafft, *Eventastic* zu realisieren und sind nun mit dem Endergebnis zufrieden.

Kristine hat das Grundgerüst der App erstellt und war für das Erstellen sowohl der externen Firebase-Datenbank als auch der internen SQLite-Datenbank zuständig und sorgte dabei für das korrekte Auslesen und Anzeigen von Daten aus dem Internet. Sie verband beide Datenbanken miteinander und entwickelte hierfür die nötigen Activities. Außerdem kümmerte sie sich um das Ermitteln der GPS-Daten des Nutzers und passte dafür die `SettingsActivity` an.

Terasas Aufgabenbereich war das adaptive Layout, die Designvorgaben und die Notifications. Des Weiteren sorgte sie für einige Anpassungen und Implementierungen in verschiedenen Activities, wie z.B. korrekte Navigation, Benachrichtigungen, `TabLayout`, und weitere Features. Außerdem stellte sie die `strings.xml` in drei Fremdsprachen zur Verfügung und sorgte somit für multiple language support.

Die Erstellung dieser Dokumentation ist Gemeinschaftsarbeit.