Placez

Albert Hildenberg, Mark Kastner

Hochschule Osnabrück

Fakultät Ingenieurwissenschaften und Informatik

Barbarastr. 16, D-49076 Osnabrück

albert.hildenberg@hs-osnabrueck.de

mark.kastner@outlook.com

24. August 2016

Zusammenfassung

Die App Placez ermöglicht dem Nutzer, Orte auf einer Karte zu markieren und zu speichern. Diese können in verschiedene Kategorien eingeteilt werden.

Inhaltsverzeichnis

1 Einleitung

„Placez“ ist eine App mit der sich verschiedene Orte oder Gegenstände speichern lassen. Diese Orte können ganz einfach mit einem Namen, der Adresse, einer Kategorie, einem Bild und einer Beschreibung versehen werden. Bei der Kategorie gibt es vorgefertigte Auswahlmöglichkeiten wie Bank, Restaurant, Parkplatz, Freizeit oder Gegenstand. Außerdem können sämtliche gespeicherten Orte und Gegenstände auf einer Google Maps Karte angezeigt werden.

Geeignet ist diese App also, wenn man sich beispielsweise den aktuellen Parkplatz von seinem Auto oder Fahrrad speichern und später dorthin navigieren möchte. Dies ist gerade in fremden Städten praktisch. Außerdem ist es möglich den Ort von wichtigen Gegenständen zu speichern, denn dank des Fotos und der Beschreibung lassen sich auch Gegenstände leicht wiederfinden.

2 Konzept

In Kapitel soll auf den Aufbau der Anwendung eingegangen werden.

2.1 Placez

Bei dem Start der Anwendung sieht der Nutzer in der Placez Übersicht alle bisher gespeicherten Placez. Neben dem Bild, werden Name und Beschreibung angezeigt.

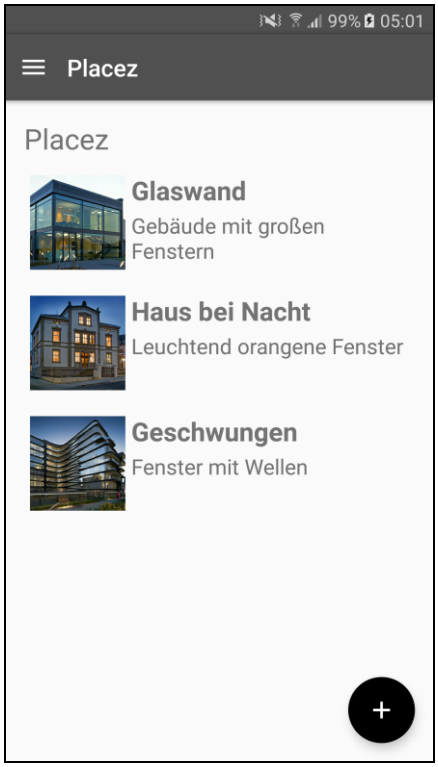

Abbildung 1 – Placez Übersicht

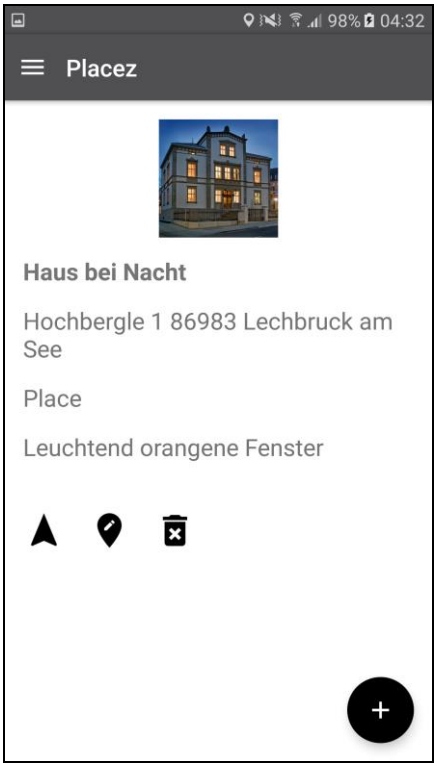

Abbildung 2 - Detailansicht

Wird auf einen der Einträge geklickt, öffnet sich eine Detailansicht des jeweiligen Place, in der Bild, Name, Adresse, Kategorie und Beschreibung angezeigt werden. Außerdem befinden sich, unterhalb der Informationen, drei Buttons. Der erste Button dient der Navigation und öffnet die Navigations-Anwendung des Gerätes, welche direkt den Standort des Place übergeben bekommt. Dadurch kann die Navigation direkt starten. Der zweite Button öffnet ein Fenster, in dem die vorhandenen Informationen bearbeitet werden können. Der Löschen-Button bietet die Möglichkeit, diesen Eintrag zu löschen. Zur Sicherheit, dass nicht versehentlich der Button gedrückt wurde, erscheint eine Sicherheitsabfrage, in der man bestätigen muss, den Eintrag zu löschen.

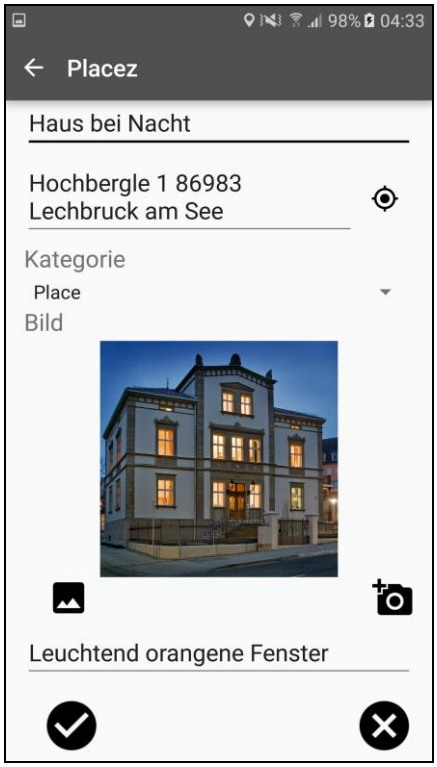

Abbildung 3 – Place Bearbeiten

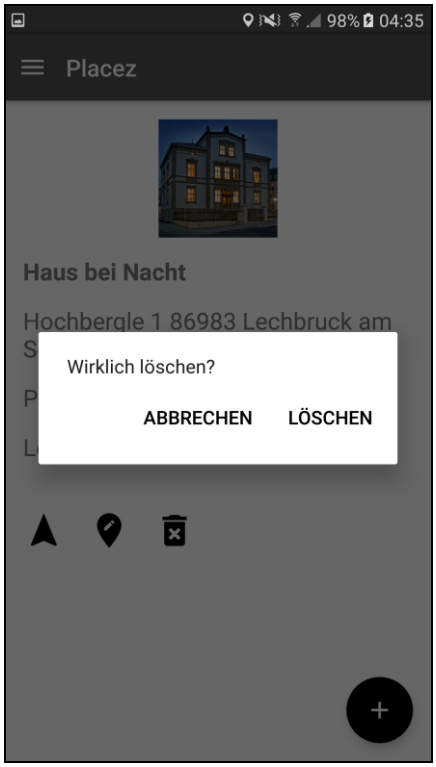

Abbildung 4 – Place löschen

In der rechten unteren Ecke kann mit einem *FloatingActionButton* ein neuer Ort hinzugefügt werden. In der Erstellung eines Place sind der Name und die Adresse zwingend nötig, wobei der Nutzer die Adresse entweder im Adress-Format angeben kann oder in Längen- und Breitengraden. Optional zu den Pflichtangaben kann eine Kategorie gewählt, ein Bild aus der Galerie geöffnet oder per Kamera aufgenommen und eine Beschreibung des Place hinzugefügt werden. Der Nutzer hat die Wahl zwischen sechs verschiedenen Kategorien: Place, Bank, Restaurant, Parkplatz, Freizeit und Gegenstand. Bei der Wahl der Bilder wird das Bild immer in ein 1x1-Format gesetzt, bei dem der Nutzer selber den Ausschnitt wählt.

Abbildung 5 – Place Erstellung

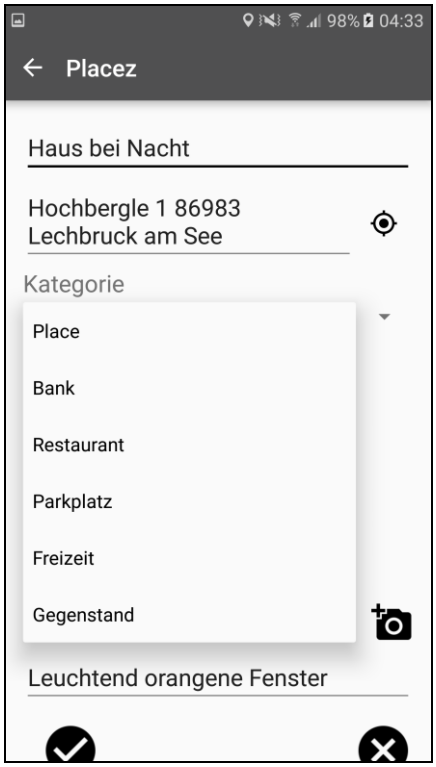

Abbildung 6 – Kategorie auswählen

2.2 Karte

Auf der Karte der Anwendung werden dem Nutzer die eigene Position in Rot angezeigt und die Placez in seiner Nähe in Blau. Klickt der Nutzer einen blauen Marker an, öffnet sich das zugehörige *InfoWindow*, in dem das Bild, der Name, die Adresse und die Beschreibung des Place angezeigt werden. Wenn ein Marker angeklickt wurde, erscheinen unten rechts auf der Karte zwei Symbole. Das eine Symbol startet eine Navigation zum ausgewählten Punkt und das andere zeigt den Ort in der Maps-App an.

Abbildung 7 – Karte

Abbildung 8 – Marker InfoWindow

2.3 Einstellungen

In den Einstellungen der App gibt es die Möglichkeit, sämtliche Einträge zu löschen, falls es zu viele nicht benötigte Placez gibt. Auch hier muss zunächst eine Sicherheitsabfrage bestätigt werden, damit nicht versehentlich alle Einträge gelöscht werden.

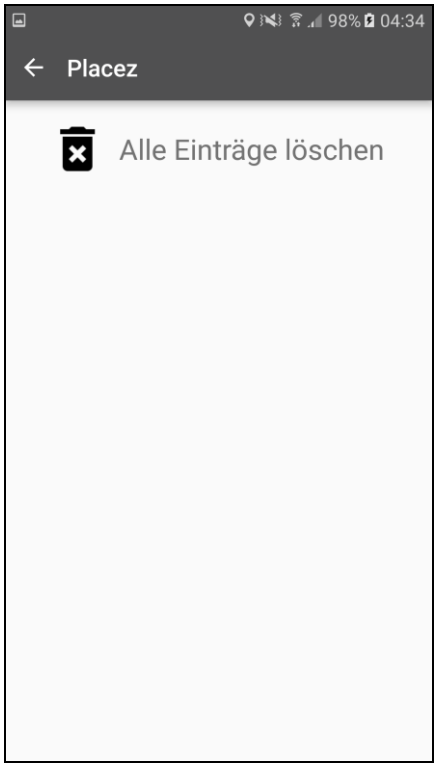

Abbildung 9 – Einstellungen

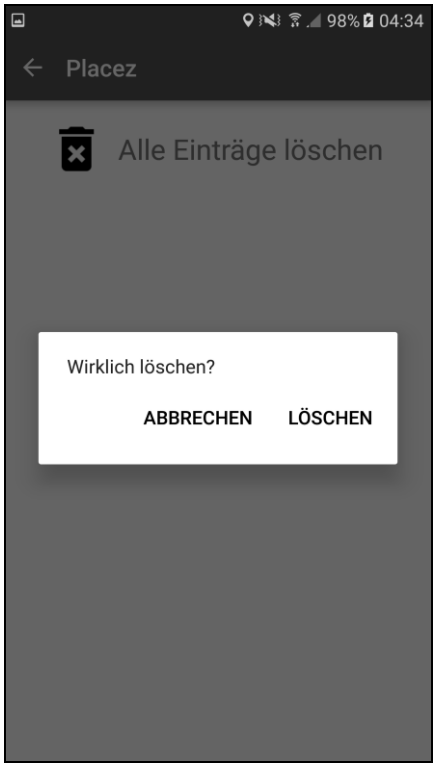

Abbildung 10 – Sicherheitsabfrage

2.4 Info

Die Info-Ansicht gibt Auskunft, von wem und in welchem Rahmen die App entwickelt wurde.

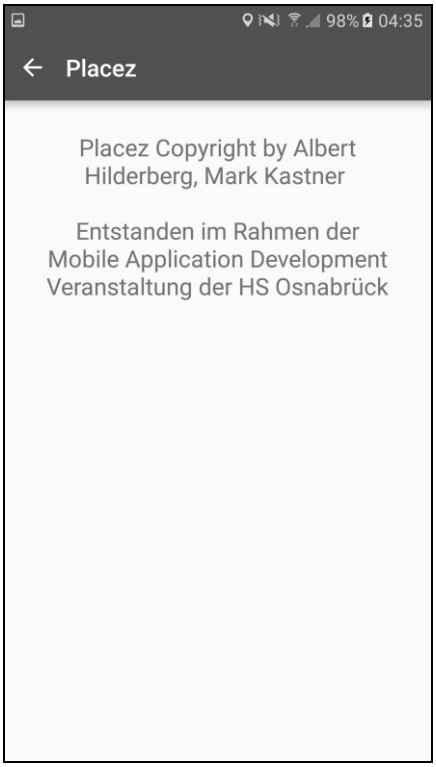

Abbildung 11 - Info

2.5 Menü

In dem Menü kann der Nutzer die Unterpunkte Placez, Karte, Einstellungen und Info öffnen.

3 Realisierung

Dieses Kapitel beschreibt die Implementierung der App.

3.1 Klasse Home

Für die *Home-View* wurde zum Anzeigen aller gespeicherten Placez eine *RecyclerView* verwendet.

Die *RecyclerView* löste mit API 8 die damalige *ListView* ab. Vorteil der *RecyclerView* ist das ,,Recyclen" von Listenelementen. In einem speziell dafür vorgesehenen Cache werden Listenelemente zwischengelagert, um Ressourcen zu sparen. Die Cachegröße ist in diesem Fall auf fünf festgelegt worden.

Ein speziell implementierter Adapter, der *CustomAdapter*, erbt vom *RecyclerView.Adapter* und übernimmt die Aufgabe, ein *PlaceObject* mit den Elementen eines Listenelementes zu verbinden. Dies geschieht mithilfe der Klasse *MyViewHolder*, welche von *RecyclerView.ViewHolder* erbt. In dem *customelement.xml* ist ein spezielles Layout für jedes Listenelement verwendet worden, bestehend aus einem Bild, dem Namen des Place und dessen Beschreibung. Auf diese drei Elemente hält der *Viewholder* eine Referenz. In der *onBind*-Methode des Adapters werden nun mithilfe von erstellten Placez die Layout-Elemente mit Daten gefüllt und angezeigt. Die *RecyclerView* stellt somit eine Live-Ansicht der aktuellen Placez-Liste dar. Gefüllt wird die Placez-Liste mit im *LocalStorage* abgelegten Placez im JSON-Format. Beim Erstellen der *HomeActivity* wird die gespeicherte Datei mithilfe der *Controller*-Methode *readFromJson()* ausgelesen. Wurde noch keine Datei angelegt, wird eine dementsprechende *Exception* geworfen und abgefangen.

Beim Tappen auf eines der Listenelemente wird die Position des angetappten Elementes zurückgegeben, mit der ohne durchlaufen von Schleifen oder Ähnlichem auf das entsprechende Element mit der *Liste.get(position)*-Methode sofort zugegriffen werden kann. Zudem wird ein Fragment erzeugt, welches mit einer Animation von rechts in den Screen slidet und sich über die eigentliche *View* legt. Diesem Fragment wurde die vorher gespeicherte Position des angefragten Listenelementes übergeben.

In der unteren rechten Ecke der *View* befindet sich ein *FloatingButton*, der dem Benutzer die Möglichkeit gibt, neue Orte zu erstellen, indem dieser die *AddPlaceActivity* öffnet. Mit der Hilfe von *SharedPreferences* wird die ID der Orte gespeichert, damit auch nach dem Neustart der App die aktuelle ID erhalten bleibt. Dadurch werden Fehler beim Löschen von Orten vermieden. So können nicht mehrere Orte gelöscht werden, die beispielsweise alle mit „Bank“ benannt wurden.

Ab Android 6.0 muss der Anwender der Nutzung von Standort und Speicher zustimmen, da dies nicht mehr automatisch bei der Installation der App geschieht. Um nun den Zugriff auf den Standort und den Speicher des Gerätes zu erhalten, wird überprüft, ob eine neuere Android Version als *Lollipop* installiert ist. Wenn dies der Fall ist, wird mit *checkIfAllreadyhaveLocationPremission()* und *checkIfAllreadyhaveStoragePermission()* überprüft, ob die Zugriffe bereits gewährt wurden, ansonsten wird nach den fehlenden Berechtigungen gefragt. Sollten diese Anfragen abgelehnt werden, erscheint ein Text mit einer Benachrichtigung an den Nutzer, dass diese Berechtigungen notwendig sind, damit die Anwendung korrekt funktioniert.

3.2 Klasse AddPlaceActivity

Die *AddPlaceActivity* dient einerseits zum Erstellen, andererseits zum Editieren von Placez. Nach dem Start der *Activity* wird zunächst geprüft, ob beim Starten des *Intents* der Übergabeparameter *position* mitgeliefert wurde. Ist dies der Fall, handelt es sich um das Bearbeiten eines Ortes und die Eingabefelder werden mit den Informationen des zu bearbeitenden Elementes gefüllt.

Nach den Eingaben des Nutzers und Tappen des Bestätigen-Buttons wird zunächst die eingegebene Adresse untersucht. Es wird unterschieden zwischen realen Adressen und Ortskoordinaten. Mithilfe einer *regularExpression* wird geprüft, ob es sich bei der eingegebenen Adresse um Zahlen handelt, die von einem Komma getrennt werden. Handelt es sich hierbei um Längen- und Breitengrade, wird mithilfe der Android *Geocoder*-Klasse, der *getFromLocation(lattitude, longitude)*-Methode und der Anzahl der Adressen versucht, den Koordinaten eine Adresse zuzuordnen. Mithilfe der restlichen Angaben des Benutzers wird nun ein Place-Objekt erstellt und der Liste hinzugefügt.

Handelt es sich um keine Koordinaten im Adressfeld, wird mit der *Geocoder*-Klasse sogenanntes *ReverseGeocoding* versucht. Es wird geprüft, ob für die eingegebene Adresse Längen- und Breitengrade zuzuordnen sind. Ist dies erfolgreich, wird auch hier ein Place-Objekt erzeugt und hinten an die Liste angehängt.

Beim Editieren eines vorhandenen Objekts wird nicht an die Liste angehängt, sondern das entsprechende Objekt in der Liste angepasst.

Falscheingaben werden abgefangen und mit sogenannten Toasts dem Benutzer zu erkennen gegeben.

Klickt der Nutzer in dem Abschnitt „Bild“ auf den linken Button, der die Galerie des Gerätes darstellt, wird ein *Intent* gestartet, das dem Nutzer die Möglichkeit gibt, ein Bild aus der Galerie auszuwählen. Durch das Anfügen von *putExtra(„crop“, „true“);*, *putExtra(„aspectX“, 1);* und *putExtra(„aspectY“, 1);* kann der Nutzer einen Bildausschnitt des gewählten Bildes im Verhältnis 1x1 bestimmen. Damit es zu keinem Qualitätsverlust des Bildes kommt, wird der gewählte Ausschnitt temporär gespeichert. Wird das Bild nun weiterverwendet und in einer *ImageView* angezeigt, wird das gespeicherte Bild wieder vom Gerät entfernt.

Die andere Option ist, über den Kamera-Button per *Intent* die Kamera zu starten und ein Bild aufzunehmen. Dieses wird, wie das Bild der Galerie, zugeschnitten und bis zur Nutzung gespeichert.

Um die Bilder für die spätere Verwendung zu speichern, ohne dass diese doppelt in der Galerie des Nutzers auftauchen, werden die Bilder in einen *Base64 String* kodiert und mit den anderen Informationen gespeichert. Sollte nun ein Bild benötigt werden, kann es mit der Funktion *decodeBase64* dekodiert und angezeigt werden.

3.3 Klasse DetailsActivity

Nach der Erstellung des Fragmentes durch Antippen eines Elementes in der Placez-Liste wird zunächst mithilfe der übergebenen Position das entsprechende Place-Objekt aus der Liste geholt. Mit diesem Objekt werden nun alle für den Benutzer relevanten Informationen über den ausgewählten Ort visualisiert. Zu sehen sind: Name, Adresse, Beschreibung, Kategorie und Bild des Place. Weiterhin hat der Benutzer die Möglichkeit, das ausgewählte Objekt zu löschen, zu bearbeiten oder mithilfe von GoogleMaps eine Navigation zum jeweiligen Standort zu starten.

Wird der Button für die Navigation zu dem aktuellen Ort betätigt, wird zunächst geprüft, ob die Standortdienste aktiv sind. Falls die Standortdienste deaktiviert sind, wird eine Meldung ausgegeben, die dazu auffordert, zuerst die Standortdienste zu aktivieren. Bei aktiviertem Standort wird ein *Intent* erstellt, um zur Karte zu gelangen und per Extra wird die ID des aktuellen Ortes beigefügt.

3.4 Klasse LocationService

Diese Klasse implementiert den *LocationListener* und ist für sämtliche Standortzugriffe zuständig. Zu Beginn wird in der statischen Variable *MIN_DISTANCE_CHANGE_FOR_UPDATES* die Distanz festgelegt, um die die aktuelle Position abweichen muss, bevor die Funktion *onLocationChanged* aufgerufen wird. Dies verhindert, dass bei geringen Standortabweichungen nicht jedes Mal die Karte aktualisiert wird. Bei der Initialisierung des *LocationService* wird geprüft, ob alle Berechtigungen vorhanden sind und ob der Standortdienst aktiviert ist. Kann nun ein Standort ermittelt werden, werden die Klassenvariablen *longitude* und *latitude* gesetzt, welche per *getLongitude()* und *getLatitude()* abgerufen werden können. Die bereits angesprochenen Funktion *onLocationChanged(Location location)* aktualisiert bei Änderung des Standortes die aktuelle Position auf der Karte. Um zu verhindern, dass der *LocationService* versucht, den Standort auf der Karte zu ändern, während noch keine Karte geöffnet wurde, überprüft dieser einen *boolean* in den *SharedPreferences*, welcher angibt, ob die Karte momentan aktiv ist.

3.5 Klasse MapsActivity

In der Klasse *MapsActivity* wird zu Beginn eine Instanz des *LocationService* angefordert, um die aktuellen Werte der Längen- und Breitengrade zu erhalten. Wenn die Karte geladen wurde, wird die Funktion *onMapsReady(GoogleMap googleMap)* ausgelöst. In dieser Funktion wird zunächst ein Marker an der aktuellen Position gesetzt und die Kamera dorthin bewegt. Des Weiteren wird, mit Hilfe des *Controllers*, die JSON-Datei mit den gespeicherten Orten ausgelesen und die Orte werden für die spätere Verwendung gespeichert. Für jeden dieser Orte wird nun mit der Funktion *addGoogleMapsMarker(Place place)* ein Marker auf der Karte erstellt. Wird einer dieser Marker angeklickt, öffnet sich ein *CustomInfoWindowAdapter*, welcher eine *Imageview* für das Bild des jeweiligen Markers und drei *TextViews* anzeigt, in denen der Name, die Beschreibung und die Adresse gespeichert sind. In der Liste der gespeicherten Marker wird der passende Marker herausgesucht und dessen *InfoWindow* angezeigt. Damit der *LocationService* nicht versucht, auf die Karte zuzugreifen, während diese nicht aktiv ist, werden die Funktionen *onStart()* und *onStop()* überschrieben, um einen *boolean* in den *SharedPreferences* zu speichern, welche im *LocationService* überprüft wird.

3.6 Controllers

Der Controller kümmert sich um die Verwaltung der Placez-Liste, sowie das Schreiben und Lesen der JSON-Datei und das Erstellen, Bearbeiten und Löschen von Place-Objekten.

Nach jedem Bearbeiten, Erstellen oder Entfernen eines Objekts wird die *writeToJSON()*-Methode aufgerufen, die die Place-Liste mithilfe von *Gson*, einer speziellen Bibliothek, in das benötigte Format bringt und von einem *FileOutputStream* in eine Datei im *LocalStorage* geschrieben wird.

Implementiert ist der Controller nach dem Singleton-Pattern. Es existiert durch Überschreiben des Standard-Konstruktors nur eine Instanz dieser Klasse zur Laufzeit, welche mit *getInstance(context)* abgerufen werden kann.

Zusätzlich besitzt der Controller die *decodeBase64*-Methode, die zum Umwandeln eines String in ein Bitmap dient.

4 Test

Im Rahmen der Tests wurde die Anwendung auf mehreren Geräten installiert und von dritten Personen getestet. Dabei wurden alle Anwendungsfälle durchlaufen.

Bei den Tests gab es Probleme, wenn der Nutzer einen Ort hinzufügen oder bearbeiten wollte und keine korrekte Adresse eingetragen hat. Durch die falsche Eingabe konnte der *geocoder* keine Adresse bestimmen. Gelöst wurde das Problem, indem eine Abfrage prüft, ob der *geocoder* erfolgreich war. Sollte dies nicht der Fall sein, wird dem Nutzer eine entsprechende Nachricht angezeigt und er hat die Möglichkeit seine Eingabe zu ändern.

Nachdem die Tests abgeschlossen waren und die Verbesserung eingefügt wurd, kann das Programm ohne Fehlermeldungen oder Abstürze bedient werden.

5 Fazit & Ausblick

Die entwickelte Anwendung kann mit vielen gut funktionierenden und übersichtlichen Features punkten, wie einem einfachen Design und einer simplen Menüführung. Bei allen Buttons ist die Funktion durch die gewählte Grafik eindeutig erkennbar. Die Kernfunktionen der Anwendung, wie das Erstellen, Bearbeiten und Anzeigen von Placez, wurden implementiert. Die Navigation zu einem Ort gelingt mit nur einem Klick. Die Liste der Placez gibt eine übersichtliche Ansicht auf seine gespeicherten Orte und macht es einfach, diese zu verwalten. Das Speichern und Lesen der Daten aus dem *LocalStorage* erfolgt automatisch und ohne Verzögerung des Anwendungsverlaufs. Ein weiterer gelungener Aspekt ist, dass die Bilder in einem speichersparenden *Base64* Format gespeichert werden.

In einer späteren Version der App soll es möglich sein, den nächsten Ort zu einer bestimmten Kategorie anzeigen zu lassen und Placez nach Kategorien zu filtern.