

App-Front-End

Funktionalität

Die Applikation bietet, mit Hilfe der Live Daten der Stadt Amberg, eine Übersicht der Parkhäuser mit Parkinformationen. In der Übersicht ist für jedes Parkhaus die Anzahl an freien Parkplätzen, der Trend, Preis und Öffnungszeiten für kostenpflichtiges Parken gegeben. Aktuell werden die Live-Daten von 8 Parkhäuser geliefert. Der Benutzer hat die Möglichkeit eine Favoriten Auswahl für die 8 Parkhäuser zu treffen, sowie sich auf der Karte durch die aktuelle Position orten und verfolgen zu lassen. Des Weiteren ist es möglich eine Sprachausgabe zur Unterstützung zu aktivieren. Hierbei werden die Anzahl der freien Parkplätze, der Name des Parkhauses, sowie eine Parkgebühren-Auskunft, abhängig von der Uhrzeit wiedergegeben.

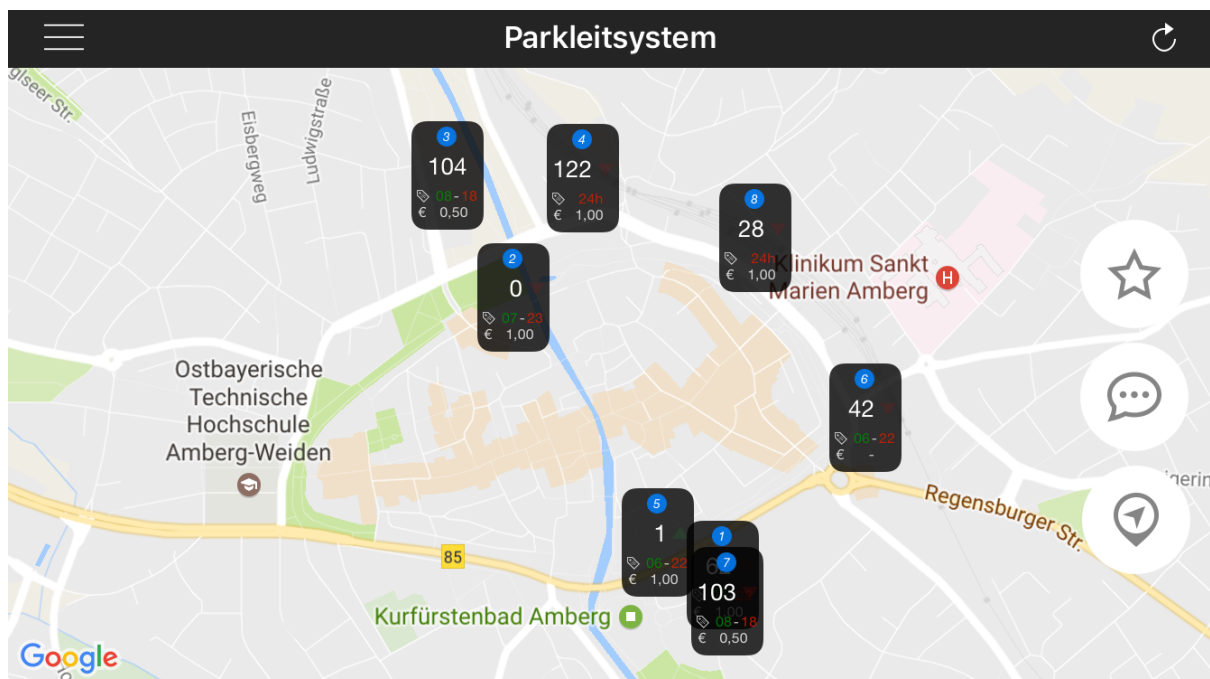



Abbildung 1: Kartenansicht des Parkleitsystems

Interaktionsmöglichkeiten

 Menü-Button: Durch einen Tap auf den Button erscheint ein Menü von rechts auf dem Bildschirm. Durch einen erneuten Tap schließt sich das Menü wieder.

 Refresh-Button: Durch einen Tap auf den Button werden die aktuellen Daten vom Server geladen und die Ansicht mit den aktuellen Daten erneuert. Falls keine Internetverbindung besteht, werden die zu letzt geladenen Daten verwendet. Des Weiteren wird ein Toast auf dem Bildschirm ausgegeben, dass keine Internetverbindung besteht.

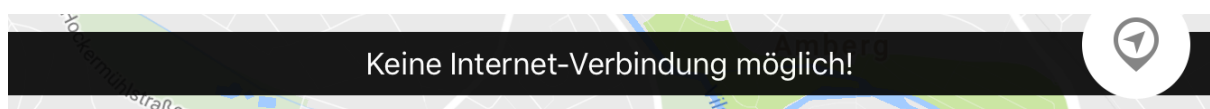
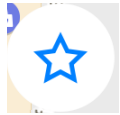


Abbildung 2: Toast bei fehlerhafter Internetverbindung



Favoriten: Ist der Button aktiv (blau), werden in der Kartenansicht nur Parkhäuser, welche zuvor in der Favoritenliste aktiviert wurden, dargestellt. Diese Einstellung wirkt sich auf die Sprachausgabe und dem Autozoom aus.



Sprachausgabe: Ist der Button aktiv (blau), werden bei Annäherung an ein Parkhaus (Radius 150 Meter) Informationen akustisch über den Lautsprecher wiedergegeben (siehe Beschreibung Funktionalität).



Aktuelle Position: Ist der Button aktiv (blau), aktualisiert sich die Kameraposition der Karten zur aktuellen Position. Bewegt sich der Benutzer, so wird bezüglich der veränderten Position die Kameraposition erneuert. Die Option wird automatisch deaktiviert sobald der User manuell mit der Karte interagiert (Tap, etc.)

App-Menü

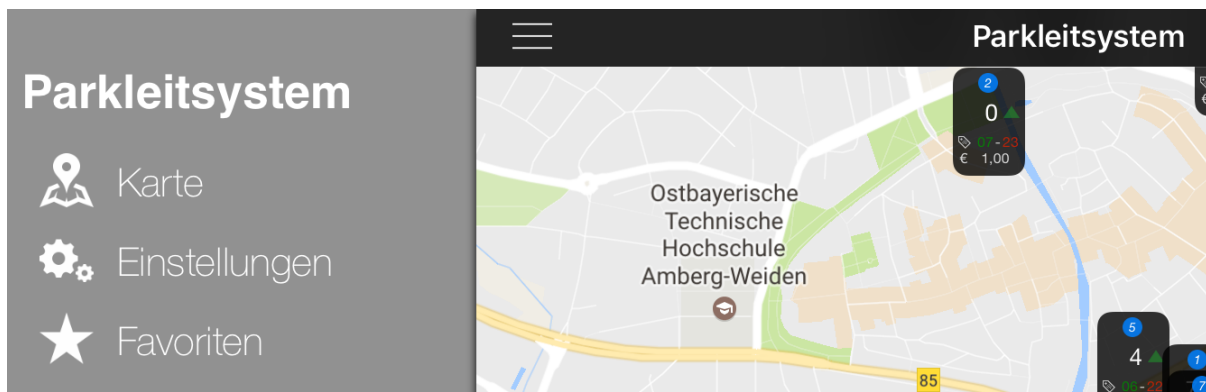


Abbildung 3: Menü-Ansicht des Parkleitsystems

Durch das öffnen des Menüs, besteht die Möglichkeit zur Karte, Einstellungen oder Favoriten zu springen. Unter "Karte" öffnet sich die bereits bekannte Kartenansicht des Parkleitsystems (siehe Abbildung 1) mit deren Interaktionsmöglichkeiten. Unter dem Menü-Punkt "Einstellungen" kann der Benutzer die bereits bekannten Optionen "Favoriten" und "Sprachausgabe" festlegen aber auch "Autozoom" und "Detailansicht". Beim der Annäherung eines Parkhauses, wird an die

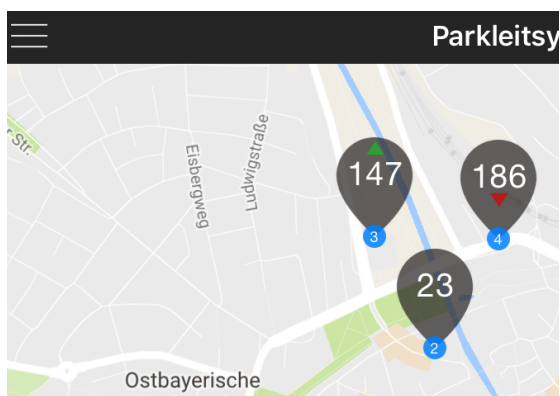


Abbildung 4: Detaillosen Marker

Position des Parkhauses automatisch heran gezoomt und beim verlassen wieder heraus gezoomt. Dies kann durch die Option "Autozoom" aktiviert bzw. deaktiviert werden. Die Detailansicht eines Parkhauses kann durch die Option "Detailansicht" deaktiviert werden, wodurch sich der Marker auf der Karte verändert und nur noch die aktuelle Anzahl der freien Parkplätze mit deren Trend sichtbar ist.

Im Menü-Punkt "Favoriten" kann der User bestimmen, welche der Parkhäuser ein- bzw. ausgeblendet werden sollen. Hierzu kann aus der Liste der 8 Parkhäuser gewählt werden. Aktiviert der User den "Favoriten" Button in der Kartenansicht so werden, basierend auf dieser Liste die Parkhäuser dargestellt.

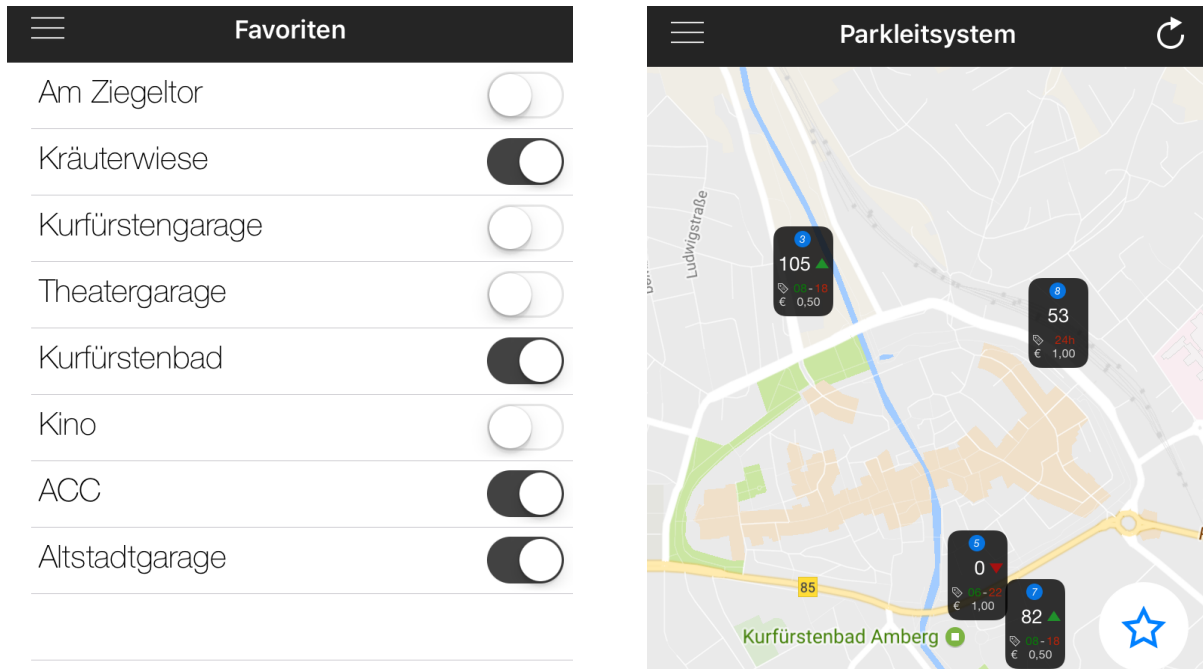


Abbildung 5: links: Auswahl der Parkhäuser in der Favoritenliste, rechts: Darstellung der ausgewählten Favoriten in der Kartenansicht

Parkhausdarstellung

Je nach Bedarf kann der Benutzer zwischen zwei Darstellungsarten im Einstellungs-Menü wählen. Eine die Detailansicht kann zur Orientierung vor der Fahrt benutzt werden. So kann sich der Benutzer über das Parkhaus informieren und auf Grund dessen eine Entscheidung treffen. Die Ansicht mit weniger Details kann während der Autofahrt benutzt werden um nur das wichtigste auf einen Blick zu sehen.



Abbildung 6: links: Detailansicht eines Parkhauses, rechts: Darstellung mit weniger Details eines Parkhauses

Betriebsmodus

Startet der Benutzer die App auf seinem Smartphone, befindet sich die Ansicht standardmäßig in der Kartenansicht mit Amberg als Mittelpunkt der Karte (vgl. Abbildung 1). Des Weiteren werden die vom User getroffenen Einstellungen geladen und auf Grund dessen die Parkhäuser dargestellt. Nähert sich der User dem Umkreis von 150 m eines Parkhauses, so wird ein Toast mit dem Namen des Parkhauses dargestellt, sowie bei aktivierter "Autozoom"-Einstellung auf das Parkhaus gezoomt und bei gewünschter Sprachausgabe akustisch wiedergegeben. Verlässt der Benutzer den Umkreis des Parkhauses, so wird wieder die Standardansicht von Amberg dargestellt.

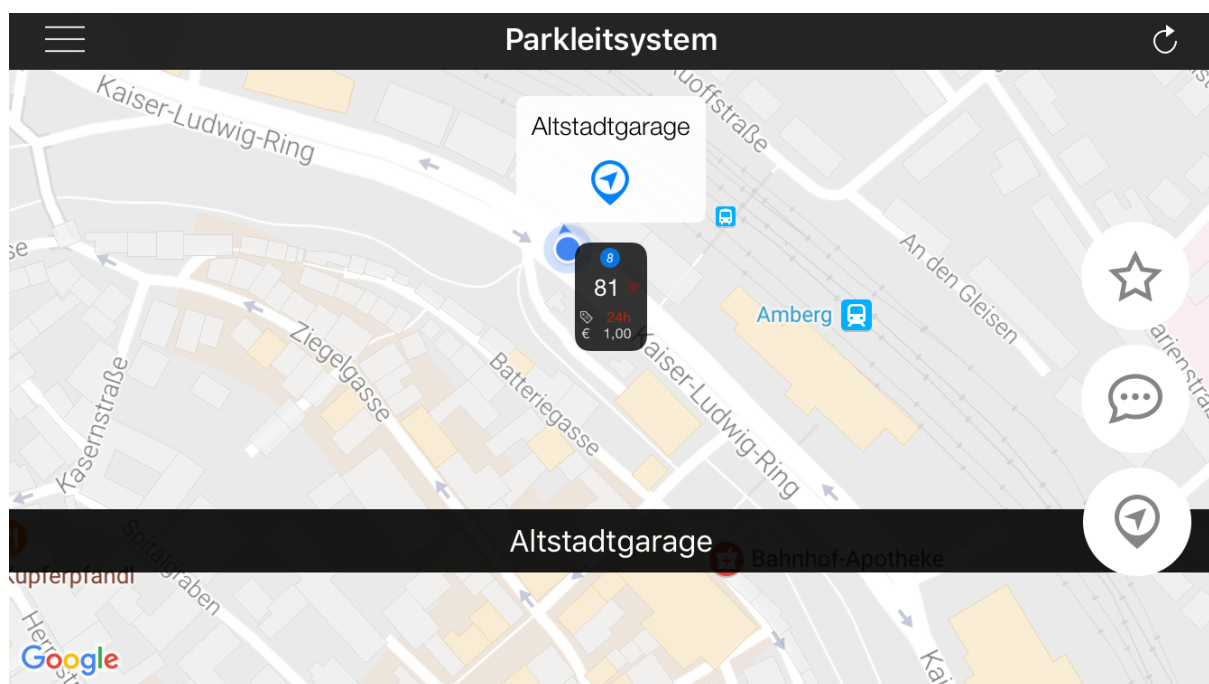


Abbildung 7: Zoom-Ansicht eines Parkhauses mit Navigations-PopUp-Menü

Möchte der User sich zu einem Parkhaus navigieren lassen, so kann dies über einen Tap auf ein Parkhaus ermöglicht werden. Hierbei öffnet sich ein PopUp-Menü überhalb des Parkhauses, wodurch die Navigation gestartet werden kann. Durch das Starten der Navigation öffnet sich Google Maps und übernimmt den weiteren Verlauf.

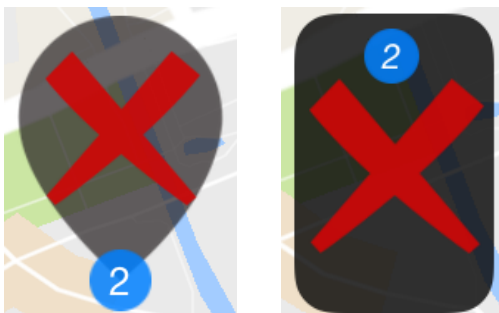


Abbildung 8: Parkhaus ist geschlossen

Sollte ein Parkhaus einen Status-Fehler aufweisen oder Geschlossen sein, so wird das Parkhaus auf der Karte durch ein rotes "X" kenntlich gemacht. Statusinformationen werden ausgeblendet um den Benutzer nicht zu verwirren. Sämtliche Regel wie "Autozoom", "Sprachwiedergabe", etc. werden für diese Parkhäuser nicht mehr berücksichtigt.

Realisierung

Die App wurde aufgrund meiner Vorliebe zu Apple für das iOS Betriebssystem entwickelt. Als IDE wurde Xcode mit der Programmiersprache Swift 3 verwendet. Zu den Standard Libraries wurde noch "GoogleMaps", "SWXMLHash" und "SWR" eingebunden.

Das "Google Maps" Framework wird verwendet für die Kartenansicht und um die Parkhaus-Marker zu setzen. Die Ortung des Nutzers folgt über das "CoreLocation" Framework von Apple. Zum Parsen des Parkleitsystem XML Files wird "SWXMLHash" verwendet. Für das Slide-Menü wird das von Joan Lluch "SWR" Framework verwendet. Bezogen und eingebunden wird das ganze über CocoaPods.

Positionierung der Parkhäuser

Da das Parkleitsystem XML der Stadt Amberg keine Geocode Informationen über die Parkhäuser liefert mussten diese manuell bezogen werden. Es ist möglich über die Google Maps API zu einer bestimmten Adresse Geocode Informationen zu beziehen, jedoch ist dies an den eigenen API Key gebunden, was wiederum bedeutet, dass nur eine bestimmte Anzahl an Anfragen pro Tag kostenlos sind. Würde diese Anzahl überschritten, könnten Benutzer der App beim Initialisieren der App die Parkhäuser nicht sehen. Aufgrund dessen wurden die Positionsdaten der 8 Parkhäuser fest codiert und hinterlegt.

```
// Creating parking lots with the missing data
parkingLots.append(ParkingLot(name: "Parkgarage Am Ziegeltor", location:
CLLocationCoordinate2D.init(latitude: 49.44852, longitude: 11.856722),
pricePerHour: "1,00", opening: "00:00:00", closing: "23:59:59",
costOpening: "00:00:00", costClosing: "23:59:59", xmlID : 4))
```

Fehlende Informationen wie die Öffnungszeiten, gebührenpflichtige Parkzeiten und der Preis wurden hier zusätzlich zu den Positionsdaten ergänzt. Das Feld "xmlID" bietet hier den Foren-Key, um eine Verbindung zwischen XML Daten und festcodierten Daten zu erstellen.

Zoom, Sprachausgabe und Toast der Parkhäuser

Wie bereits in der Funktionalität erklärt, wird in einem Umkreis von 150 m dem User Informationen über das Parkhaus bereitgestellt. Die Berechnung der Distanz vom User zum Parkhaus wäre wieder über die Google Maps API möglich gewesen, jedoch wieder API Key bezogen mit dem bekannten Problem. Aufgrund dessen wurde die Berechnung über den Location-Manager des "CoreLocation"-Frameworks realisiert.

```
func locationManager(_ manager:CLLocationManager, didUpdateLocations
locations: [CLLocation]) {
    if Int((locations.last?.distance(from: parkingLotCoordinate))) < 150
    {
        // User is less than 150 m close
    }
}
```

Da die Parkhäuser immer zugänglich sind und 24 Stunden geöffnet haben, konnte keine intelligente Sprachausgabe basierend der Öffnungszeiten realisiert werden. Es wurde jedoch auf die gebührenpflichtigen Parkzeiten angewandt. Befindet sich der User innerhalb der gebührenpflichtigen Parkzeit im Umkreis, so wird die Uhrzeit wiedergegeben, ab wann das Parkhaus wieder gebührenfrei wird. Außerhalb der Parkzeiten wird die gebührenpflichtigen Öffnungszeit angegeben. Falls es 24 Stunden gebührenpflichtig ist entfällt diese Information.

```
let now = Date()
let openToday = now.dateAt(date: pLot.costOpening)
let closeToday = now.dateAt(date: pLot.costClosing)
let dateFormatter = DateFormatter()
dateFormatter.dateFormat = "HH:mm"

if !pLot.is24hFeeBased {
    if now >= openToday &&
        now <= closeToday
    {
        //und wird ab 17.00 Uhr kostenlos
    }
    else{
        //und wird ab 9.00 Uhr kostenpflichtig
    }
}
```

Bei der "Autozoom" Einstellung wird auf das Parkhaus gezoomt sobald es sich im Umkreis des Users befindet. Es wird vermerkt auf welches Parkhaus und ob schon gezoomt wurde, um somit bei der nächsten Standortaktualisierung, welche jede Sekunde getriggert wird, nicht wieder die gleiche Aktion auszuführen. Im Fall das automatisch gezoomt wird und der User die Karte verschiebt, verhindert dies ein hin und her "springen" auf der Karte.

```
if !pLot.gotZoomed && zoomCount <= 1 {
    if self.defaults.object(forKey: "zoomSwitch") != nil{
        if self.defaults.bool(forKey: "zoomSwitch") == true {
            mapViewArea?.animate(toLocation:
                CLLocationCoordinate2D(latitude: pLot.location.latitude,
                    longitude: pLot.location.longitude))
            mapViewArea?.animate(toZoom: 17)
        }
    }
}
```

In dieser Routine wird auch die Sprachausgabe und der Toast mit verarbeitet um so nur einmal beim Betreten des 150 m Umkreises die Sprachausgabe auszugeben und die Information anzuzeigen. Beim Verlassen des Umkreises wird wieder aus der Karte heraus gezoomt.

Würde sich die Position eines Benutzers innerhalb mehrerer 150 m Parkhaus-Radien befinden, so würde auf das Parkhaus mit der kleineren Parkhausnummer gezoomt werden und erst beim Verlassen des Parkhauses wieder neu gezoomt werden.

Parkhauser-Marker

Da der Standard Marker von Google Maps nicht zur Informationsdarstellung ausreichte, wurde ein eigener Marker entworfen. Hierzu wurde eine eigene View mit Darstellungsoptionen implementiert und der View des Google Maps Markers zugewiesen. Insgesamt wurden zwei verschiedene "Custom" Marker entworfen, welche je nach Einstellung eingebunden werden. Auch das Info-Fenster der Marker wurde durch eine eigenes Fenster ausgetauscht, welches jetzt die Navigation-Möglichkeit bereitstellt.

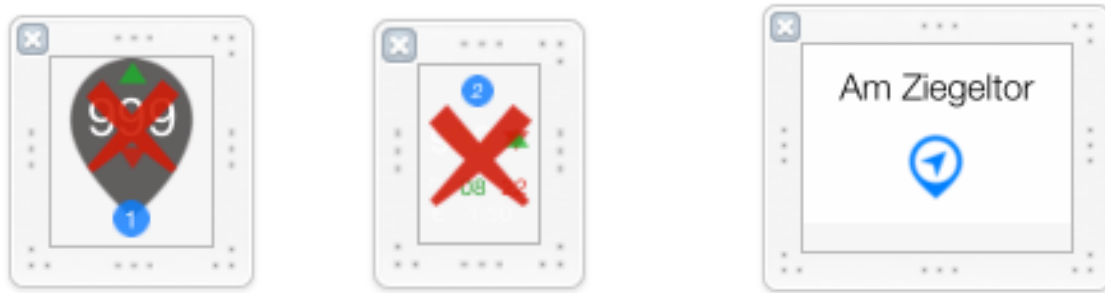


Abbildung 9: Die Implementierten Custom Views des Parkleitsystems

Slide-Menü

Als Menü wurde das von John Lluich entwickelte SWR-Slide-Menü verwendet. Das Menü basiert noch auf Objective C und dem Cococa Framework, was jedoch problemlos durch einen Swift Bridge Header eingebunden werden konnte. Das Menü wird Initialisiert über einen Root View Controller. Front- und Backview (Menüansicht) werden mit Hilfe der Identifier im Storyboard bestimmt.

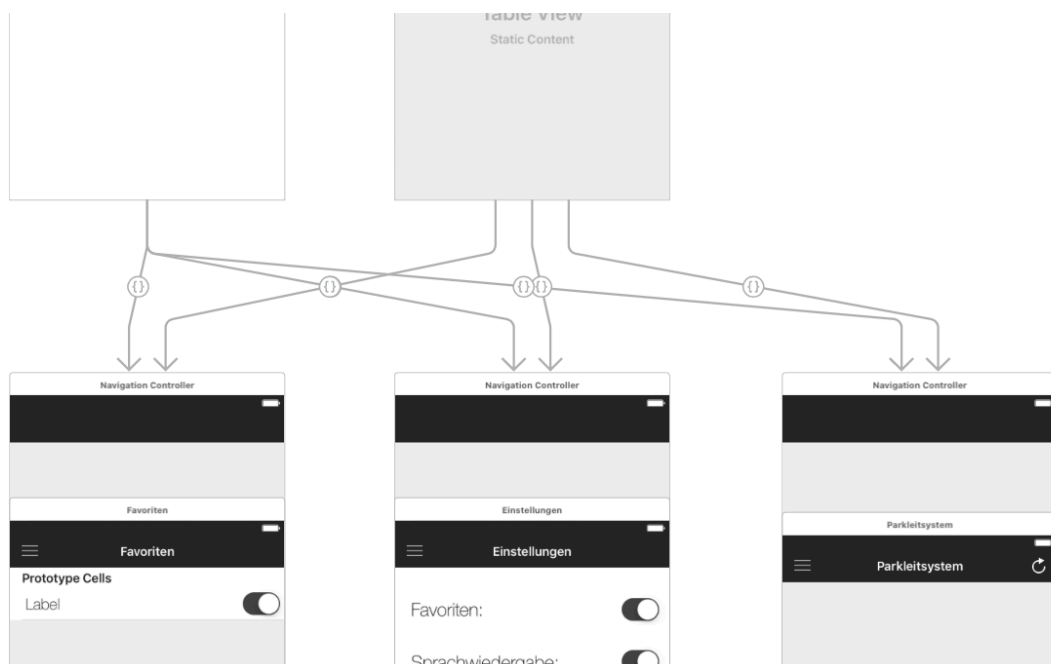


Abbildung 10: Ausschnitt aus der Storyboardansicht der Parkhausleitsystem App

Quellen:

<https://github.com/John-Lluch/SWRevealViewController>

<https://developers.google.com/maps/documentation/geocoding/intro>

<https://developer.apple.com/library/content/documentation/Swift/Conceptual/BuildingCocoaApps/MixandMatch.html>

<https://developers.google.com/maps/documentation/ios-sdk/map>

<https://developers.google.com/maps/documentation/ios-sdk/marker>

<http://stackoverflow.com/questions/20967496/google-maps-ios-sdk-getting-current-location-of-user>

<http://stackoverflow.com/questions/30743408/check-for-internet-connection-in-swift-2-ios-9>

<https://www.raywenderlich.com/129059/self-sizing-table-view-cells>

<https://www.ralfebert.de/tutorials/ios-swift-uitableviewController/custom-cells/>

<https://www.raywenderlich.com/97014/use-cocoapods-with-swift>

<http://www.appcoda.com/google-maps-api-tutorial/>

<https://developer.apple.com/swift/resources/>

<https://developer.apple.com/reference/>

<http://stackoverflow.com/questions/24857986/load-a-uiview-from-nib-in-swift>

<http://stackoverflow.com/questions/38507289/swift-how-to-read-coordinates-from-a-gpx-file>

<http://stackoverflow.com/questions/24370061/assign-xib-to-the-uiview-in-swift>

<http://stackoverflow.com/questions/29462187/creating-custom-info-window-in-swift-with-the-google-maps-ios-sdk>