App Quest 2019

Münzensammler

Neben den Schatzkisten, die auf dem HSR Gelände verteilt sind, gibt es auch noch kostbare Münzen zu entdecken. Diese sind nur sehr schlecht sichtbar und es wird deshalb eine App benötigt, die das schwache Bluetooth Signal dieser 15 Beacon-Münzen aufspüren kann.

Wie ist so ein Signal aufgebaut?

- UUID (universally unique identifier): Eine 16-Byte lange Zahl, die für unsere Apps bereits vorgegeben ist. Sie lautet:

  ```
  52495334-5696-4DAE-BEC7-98D44A30FFDA
  ```

  Benötigt wird sie, damit die Apps wissen, ob das Signal wirklich von einer zugehörigen Münze kommt.

- Major: Jede der 15 Münzen sendet auch eine Major Nummer aus. Der Einfachheit halber wurden hierbei die Nummern 1-15 gewählt.
- Minor: Neben der UUID und der Major Nummer senden die Münzen auch noch eine Minor Nummer aus. Diese Nummer gilt es für die 15 Münzen herauszufinden.

Eine detailliertere Einführung in iBeacons findet Ihr im PDF Getting Started with iBeacon (englisch, optional).

Um euch die Suche nach den Münzen etwas zu vereinfachen, haben wir sie verschiedenen Regionen zugeordnet, die Ihr aufgrund ihrer Beschaffenheit erkennen könnt:

- Region Seeufer: Major Nummern 1-5
- Region Inseli: Major Nummern 6-8
- Region Mensa: Major Nummern 9-11
- Region Veloständer: Major Nummern 12-13
- Region Forschungsgebäude: 14-15

App testen

Wir können euch leider keine Beacons mit nach Hause geben. Damit ihr Eure App aber trotzdem testen könnt, könnt ihr eure eigenen Geräte als iBeacon senden lassen. Es gibt verschiedene Tools die das erlauben:

- Windows: Beacon Simulator (Microsoft Store App)
- Android: Beacon Simulator (Google Play Store)
- Mac: BeaconEmitter (GitHub)
- iOS: My Beacon (App Store)

Eingabe des Lösungswortes

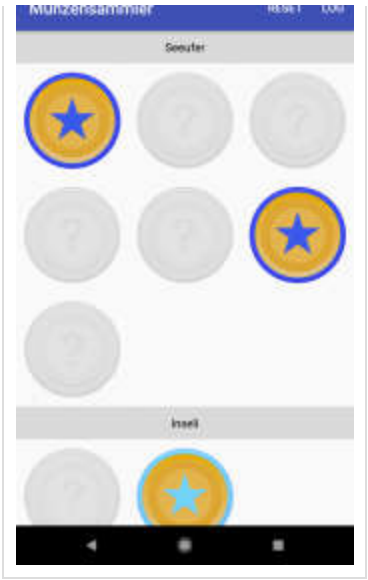
Um eure Lösung einzureichen, könnt ihr wie gewohnt die SolutionLogger Klasse verwenden. Das Format für eine gültige Lösung findet ihr unter Logbuch Format.

Android

Für diese App haben wir eine einfache Projektvorlage vorbereitet. Diese Vorlage könnt ihr hier herunterladen. Die Vorlage ist an den mit TODO gekennzeichneten Stellen zu ergänzen. Wenn ihr die Vorlage startet, sieht die App am Anfang so aus:

Die Vorlage enthält bereits eine MainActivity. Diese ist ausgestattet mit einer SectionedRecyclerView mit je einer Section für jede Region (Seeufer, Inseli, etc.). Allerdings wird in den Section-Headers nur der Text „TODO: Regions-Name einfügen“ angezeigt und die Zellen für die einzelnen Münzen haben noch gar keinen Inhalt.

Eure Aufgabe ist es jetzt, die App zu vervollständigen. Am Schluss soll sie wie folgt aussehen:



In den Section-Headers soll der jeweilige Regions-Name angezeigt werden. Je nachdem, ob eine bestimmte Münze schon gefunden wurde, wird in der jeweiligen SectionedRecyclerView-Zelle entweder das entsprechende Münzen-Bild oder die Fragezeichen-Münze angezeigt. So kann man auf einen Blick erkennen, wieviele Münzen man in einer Region schon gefunden hat.

Daten Management

Die Daten einer Münze (Major- und Minor-Nummer) müssen irgendwo abgespeichert werden, damit ihr sie am Schluss ans Logbuch übertragen könnt. Wenn die Daten nur in einer Variable (z.B. in einem Int-Array) abgelegt werden, dann gehen die Daten verloren, sobald die App geschlossen wird. Deshalb müssen wir die Daten abspeichern / persistieren. Dafür könnt ihr die CoinManager-Klasse verwenden, die wir für euch vorbereitet haben.

Nach Beacons suchen

Das Aufspüren von Beacons wird in Android am einfachsten mittels der Android Beacon Library gemacht.

1. Die MainActivity benötigt ein Property vom Typ BeaconManager.
2. Die MainActivity muss das Interface BeaconConsumer implementieren. Hinweis: Das Interface hat nur eine Methode, die für diese Aufgabe überschrieben werden muss.
3. Danach muss das Ranging der Beacons in einer sogenannten Region gestartet werden. Dazu gibt es eine passende Methode in der BeaconManager Klasse.
4. Damit der BeaconManager mit den gefundenen Beacons etwas anfangen kann, muss dem BeaconManager ein RangeNotifier hinzugefügt werden.

Danach sollte automatisch die Callback Methode des RangeNotifiers aufgerufen werden, sobald ein Beacon gefunden wurde.

App im Hintergrund laufen lassen

An der Treasure Hunt werdet ihr auch die anderen Apps verwenden, um die verschiedenen Aufgaben zu lösen.

Während dieser Zeit sollte die Münzensammler-App im Hintergrund weiterlaufen, damit trotzdem noch Münzen gefunden werden können. Ihr braucht dafür nichts weiter zu konfigurieren, denn das Ranging funktioniert in Android automatisch im Hintergrund. Im Gegensatz zu iOS kann aber das Ranging in Android nicht fortgesetzt werden, falls ihr die App komplett beendet habt.

Denkt also daran, die App nicht ganz zu schliessen, wenn ihr im Hintergrund Münzen finden wollt.

Optional: Notifications anzeigen

Wie bereits erwähnt kann eure App (sofern richtig umgesetzt) Beacons auch aus dem Hintergrund entdecken. Das ist für den Treasure Hunt ein grosser Vorteil, weil ihr so auch Münzen sammeln könnt, während ihr andere Aufgaben löst.

Trotzdem wäre es doch aber schön, wenn ihr eine Benachrichtigung erhalten würdet, sobald ihr eine Münze entdeckt habt. Und glücklicherweise geht das ganz einfach:

1. In der Vorlage findet ihr die Klasse NotificationUtil. Diese implementiert bereits einen Teil der nötigen Infrastruktur. Die Methode createNotificationChannel macht, was ihr Name verspricht. Diese Methode wird in der MainActivity im Call von setupNotificationUtil aufgerufen.
2. Um eine Notification über diesen Channel zu verschicken, muss die Methode sendNotificationToUser fertig implementiert werden. Um die Notification zu bauen, soll vorzugsweise der NotificationCompat.Builder verwendet werden.
3. Zuletzt braucht man sendNotificationToUser nur noch aufzurufen, wenn eine neue Münze entdeckt wurde.

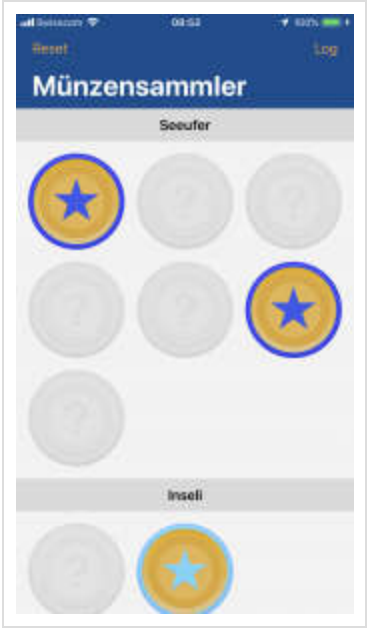
Das war’s schon. Ihr solltet nun bei jedem Fund einer Münze benachrichtigt werden.

iOS

Für diese App haben wir eine einfache Projektvorlage vorbereitet. Diese Vorlage könnt ihr hier herunterladen. Die Vorlage ist an den mit TODO gekennzeichneten Stellen zu ergänzen. Wenn ihr die Vorlage startet, sieht die App am Anfang so aus:

Die Vorlage enthält bereits eine Collection View mit je einer Section für jede Region (Seeufer, Inseli, etc.). Allerdings wird in den Section-Headers nur der Text „TODO: Regions-Name einfügen“ angezeigt und die Zellen für die einzelnen Münzen haben noch gar keinen Inhalt.

Eure Aufgabe ist es jetzt, die App zu vervollständigen. Am Schluss soll sie wie folgt aussehen:



In den Section-Headers soll der jeweilige Regions-Name angezeigt werden. Je nachdem, ob eine bestimmte Münze schon gefunden wurde, wird in der jeweiligen CollectionView-Zelle entweder das entsprechende Münzen-Bild oder die Fragezeichen-Münze angezeigt. So kann man auf einen Blick erkennen, wieviele Münzen man in einer Region schon gefunden hat.

Daten Management

Die Daten einer Münze (Major- und Minor-Nummer) müssen irgendwo abgespeichert werden, damit ihr sie am Schluss ans Logbuch übertragen könnt. Wenn die Daten nur in einer Variable (z.B. in einem Int-Array) abgelegt werden, dann gehen die Daten verloren, sobald die App geschlossen wird. Deshalb müssen wir die Daten abspeichern / persistieren. Dafür könnt ihr die CoinManager-Klasse verwenden, die wir für euch vorbereitet haben.

Nach Beacons suchen

Das Aufspüren von Beacons wird in iOS mittels des CLLocationManagers (CL = Core Location) gemacht:

1. Der MainViewController benötigt ein Property vom Typ CLLocationManager.
2. Der MainViewController muss das Protokoll CLLocationManagerDelegate implementieren. Hinweis: Für diese Aufgabe muss nur eine Methode dieses Protokolls implementiert werden.

3. Der Delegate des CLLocationManagers muss auf den MainViewController gesetzt werden.
4. Zum Schluss muss noch das Ranging der Beacons in der beaconRegion gestartet werden. Dazu gibt es eine passende Methode in der CLLocationManager Klasse.

Danach sollte automatisch die entsprechende CLLocationManagerDelegate Methode aufgerufen werden, sobald ein Beacon gefunden wurde.

App im Hintergrund laufen lassen

An der Treasure Hunt werdet ihr auch die anderen Apps verwenden, um die verschiedenen Aufgaben zu lösen. Während dieser Zeit sollte die Münzensammler-App im Hintergrund weiterlaufen, damit trotzdem noch Münzen gefunden werden können. In iOS werden Apps, die sich im Hintergrund befinden normalerweise schon nach kurzer Zeit suspendiert (bzw. in einen Schlafmodus gesetzt). Eine Ausnahme sind Apps, die im Hintergrund immer die aktuelle Position tracken müssen (z.B. Maps-Apps, etc). In der Münzensammler-App könnt ihr das ausnützen, indem ihr die folgenden 3 Schritte durchführt:

1. Rufe die Methode requestAlwaysAuthorization() auf dem CLLocationManager auf.
2. Setze das CLLocationManager-Property allowsBackgroundLocationUpdates auf true.
3. Rufe die Methode startUpdatingLocation() auf dem CLLocationManager auf.

Damit läuft die App im Hintergrund weiter und kann weiterhin Münzen detektieren.

Optional: Notifications anzeigen

Wie bereits erwähnt kann eure App (sofern richtig umgesetzt) Beacons auch aus dem Hintergrund entdecken. Das ist für den Treasure Hunt ein grosser Vorteil, weil ihr so auch Münzen sammeln könnt, während ihr andere Aufgaben löst.

Trotzdem wäre es doch aber schön, wenn ihr eine Benachrichtigung erhalten würdet, sobald ihr eine Münze entdeckt habt. Und glücklicherweise geht das ganz einfach:

1. Der MainViewController benötigt einen Member vom Typ UNUserNotificationCenter (UN = User Notifications).
2. Um eine Notification zu verschicken, muss ein UNNotificationRequest auf dem UNUserNotificationCenter abgesetzt werden. Wir wollen das immer dann tun, wenn wir eine Münze gefunden haben.
3. (Optional) Um die Notification zu personalisieren, könnt ihr den bereits vorhandenen coinSound verwenden.
4. Damit die Notifications auch im Vordergrund angezeigt werden können, muss der MainViewController als UNUserNotificationCenterDelegate agieren. Die Funktionen könnt ihr wie folgt überschreiben:

```
1  func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive response: UNNotificat
2    completionHandler()
3  }
4
5  func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification: UNNot
```

```
6     completionHandler([.alert, .sound, .badge])
7 }
```

MainViewController_UNUserNotificationCenterDelegate.swift hosted with ❤ by **GitHub** **view raw**

Hinweis: Normalerweise müsstet ihr jetzt noch die Erlaubnis zum Versenden von Notifications erfragen. Das macht man im AppDelegate, indem man den Callback didFinishLaunchingWithOptions mit einem Request um die Erlaubnis für Notifications erweitert. Der Einfachheit halber haben wir das für euch bereits übernommen.

Das war’s schon. Ihr solltet nun bei jedem Fund einer Münze benachrichtigt werden.