

[Ucll beacons]

Project dossier Mobiele app

Dit project dossier bevat alle informatie van het UCLL BEACONS MOBIELE APP project.

Project omschrijving, opbouw van de app, documentatie.

Sven Liekens, Jo Claes en Jens Molenaers.

Project dossier Mobiele app

[Ucll beacons]

1. Inhoud

[2 Project omschrijving 5](#_Toc484000080)

[2.1 iBeacons 5](#_Toc484000081)

[2.2 Onze opdracht 6](#_Toc484000082)

[3 Opbouw app 7](#_Toc484000083)

[3.1 Klassendiagrammen 7](#_Toc484000084)

[3.1.1 Models 7](#_Toc484000085)

[3.1.2 ViewModels 8](#_Toc484000086)

[3.1.3 Interface 9](#_Toc484000087)

[3.1.4 Data 9](#_Toc484000088)

[3.2 Toelichting code 10](#_Toc484000089)

[3.2.1 App.xaml.cs 10](#_Toc484000090)

[3.2.2 IRestService 11](#_Toc484000091)

[3.2.3 RestService 11](#_Toc484000092)

[3.2.4 Viewmodels 13](#_Toc484000093)

[3.2.4.1 MainPageModel 13](#_Toc484000094)

[4 Screen mockups app (Werking) 17](#_Toc484000095)

[4.1 Splash screen 17](#_Toc484000096)

[4.2 Front screen 17](#_Toc484000097)

[4.3 Route screen 18](#_Toc484000098)

[4.4 Content screens 19](#_Toc484000099)

[4.4.1 Html 19](#_Toc484000100)

[4.4.2 YouTube 19](#_Toc484000101)

[4.4.3 Foto 19](#_Toc484000102)

[4.4.4 Audio 19](#_Toc484000103)

[4.4.5 Video 19](#_Toc484000104)

[5 Linken 20](#_Toc484000105)

Afbeelding 1: IBeacons Apple 5

Afbeelding 2: Klassendiagrammen models 7

Afbeelding 3: Klassendiagrammen viewmodels 8

Afbeelding 4: Klassendiagrammen interface 9

Afbeelding 5: Klassendiagrammen data 9

Afbeelding 6: Toelichting code App.xaml.cs 10

Afbeelding 7: Toelichting code IRestService 11

Afbeelding 8: Toelichting code GetRoutesAsync 12

Afbeelding 9: Toelichting code GetBeaconInRouteAsync 12

Afbeelding 10: Toelichting code GetContentForBeaconInRoute 12

Afbeelding 11: Screen mockup geen internet 17

Afbeelding 12: Screen mockup front screen 17

Afbeelding 13: Screen mockup route screen 18

Afbeelding 14: Screen mockup geen bluetooth 19

# Project omschrijving

## iBeacons

IBeacons zijn kleine zendertjes (bluetooth), die gericht informatie naar mobiele telefoons kunnen sturen als je in de buurt bent.

Apple introduceerde deze iBeacon-technologie in juni 2013 tijdens de ontwikkelaarsconferentie WWDC 2013. Sinds iOS 7 is ondersteuning voor iBeacons aanwezig, waarbij gebruikt wordt gemaakt van Bluetooth 4.0 voor de draadloze verbinding. IBeacons hebben afhankelijk van het type een bereik tot wel 50 meter. iBeacons werken ook met smartphones van andere merken.

[](http://iculture.textopus.nl/wp-content/uploads/2014/07/ibeacons-schoenen.jpg)Via Bluetooth kan een app de nabijheid van een iBeacon inschatten, bijvoorbeeld wanneer deze in een winkel is geplaatst. Apple gebruikt zelf iBeacons in de Apple Store: als je bij een bepaald product staat kun je via de Apple Store-app meer informatie opvragen. De iBeacon maakt een schatting van jouw locatie en stuurt informatie over het product waar je dichtbij staat.

Afbeelding 1: IBeacons Apple

IBeacons vormen een alternatief voor NFC, de draadloze technologie die ook in de iPhone voorkomt. Het verschil is dat NFC alleen op korte afstand werkt, namelijk enkele centimeters. Bij iBeacons mag de afstand veel groter zijn, tot wel 50 meter. iBeacons worden daarom vooral gebruikt om informatie te verstrekken en niet zoals NFC om betalingen te doen.

**Toepassingen van iBeacons**

iBeacons bieden eigenlijk tal van mogelijkheden. De toepassingen zijn in het algemeen onder te brengen in twee grote groepen.

* Apps die jou, op basis van je locatie, informatie geven betreffende deze locatie (musea, winkels, openbare plaatsen, muziekfestivals…).
* Apps die op basis van je locatie jou navigeert naar het punt waar je moet zijn. (Musea, hospitaal, luchthaven, muziekfestivals…)

## Onze opdracht

Onze opdracht is om een mobiele app te maken om een bepaalde navigatieroute te volgen en onderweg op je route gerichte informatie te krijgen op bepaalde locaties.

Deze opdracht is op te splitsen in drie deelopdrachten.

* Een REST api om informatie te leveren aan de mobiele client (json formaat)
* Een webapplicatie om de informatie en de routes op te slaan.
* De mobiele applicatie voor de toepassing.

Binnen het opleidingsonderdeel Mobiele apps 1 maken we de mobiele applicatie. Deze staat in dit dossier beschreven met de opbouw van de app en documentatie.

# Opbouw app

Bij de opbouw van de app komt er meer info over de applicatie, bijgestaan door enkele afbeeldingen.

## Klassendiagrammen

### Models

Deze modellen zorgen voor de data, via de viewmodels, aan de views door binding.

Het ontwerp van het model is geoptimaliseerd voor de logische relaties en transacties tussen het geheel, ongeacht de wijze waarop de gegevens worden gepresenteerd in de gebruikersinterface.

1. Beacon:
   * Beacon\_id: Verkrijgen van het Id van de beacon;
   * Description\_Txt: De beschrijving die aan de beacon vast hangt verkrijgen we;
   * Location\_Ln: De locatie waar de beacon zich bevindt wordt mee aangevraagd.
2. Content:
   * Beacon\_Content\_Id: Het id van de content per beacon;
   * Content\_Txt: De inhoud die bij de beacon hoort;
   * Metatype\_Sn: Het soort metatype (Html, link, YouTube, enz..) die we meegeven om na te kijken welke soort content er wordt meegestuurd.
3. Route:
   * Name\_Ln: De naam van de route verkrijgen;
   * Route\_Id: Het Id van de route.

Afbeelding 2: Klassendiagrammen models

### ViewModels

Het viewmodel haalt gegevens uit het model en manipuleert deze naar de indeling die vereist is door de view. Het geeft aan de view door of de onderliggende gegevens in het model zijn gewijzigd, en de data wordt in het model bijgewerkt naar aanleiding van UI-gebeurtenissen in de view.

Afbeelding 3: Klassendiagrammen viewmodels

### Interface

Een interface bevat alleen de ondertekening van methoden, eigenschappen, gebeurtenissen of indexen. Een klasse die de interface implementeert, moet de leden van de interface implementeren die gespecificeerd zijn in de interface-definitie. Wanneer een klasse een interface implementeert, biedt de klasse een implementatie voor alle door de interface gedefinieerde leden. De interface zelf biedt geen functionaliteit die een klasse kan erven.

Afbeelding 4: Klassendiagrammen interface

### Data

De Restservice gaat voor een verbinding zorgen met de database, dit door behulp van een URL.

Deze heeft vier methodes:

* RestService: De constructor van deze klasse, hierin wordt een HttpClient aangemaakt;
* GetRoutesAsync: Door de methode GetRoutesAsync op te roepen verkrijgen we hier al de routes die dan getoond worden in een lijst op de view;
* GetBeaconsInRouteAsync: Deze methode krijgt een route\_id meegestuurd waardoor deze van een route de beacons kan opvragen;
* GetContentForBeaconInRoute: Als we deze oproepen gaat die met behulp van een route\_id en een beacon\_id, de content opvragen van een beacon in die route.

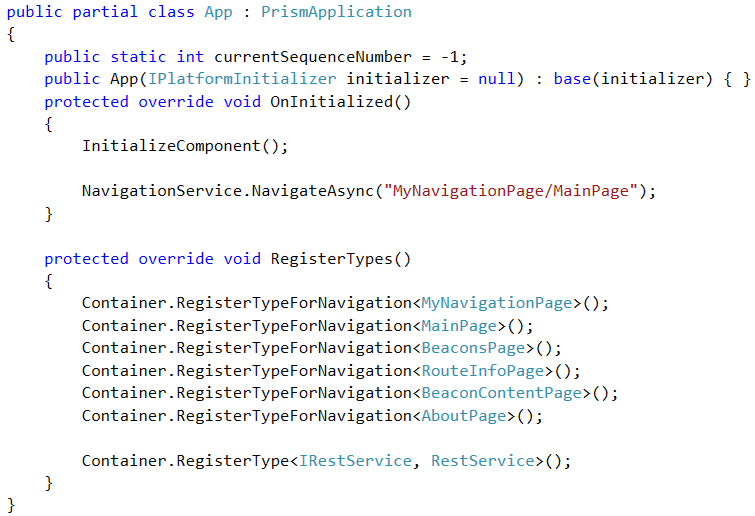
Afbeelding 5: Klassendiagrammen data

## Toelichting code

### App.xaml.cs

Bij de App.xaml.cs gaan we gebruik maken van dependency injection. Bij het gedeelte van de RegisterTypes (zie afbeelding 6) kunt u zien dat we hier verschillende klassen gaan koppelen.

Dit wil zeggen dat deze verschillende klassen data met elkaar kunnen uitwisselen zonder dat er een relatie is vastgelegd in de broncode.

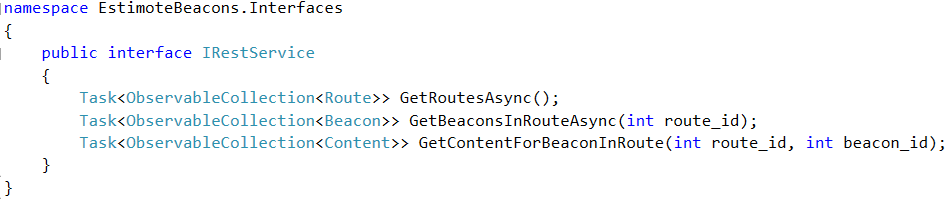
Bij OnInitialized vindt u NavigationService.NavigateAsync. Deze gaat navigeren naar de hoofdpagina eenmaal als de applicatie opstart.

Afbeelding 6: Toelichting code App.xaml.cs

### IRestService

IRestService is een interface die dient als template voor de RestService klasse. Hierin staan de methodes gedefinieerd die in een RestService moeten zitten. Doordat we deze interface aangemaakt hebben kunnen we gebruik maken dependency injection en zorgen dat onze app gebruik maakt van onze klasse RestService die gebaseerd is op de interface IRestService.

Als we de rest service zouden willen herschrijven kunnen we bijvoorbeeld een klasse bijmaken die RestService2 noemt en ook gebaseerd is op IRestService. Als we dan gebruik willen maken van RestService2 hoeven we maar 1 ding te veranderen in onze andere code. Namelijk in de App.xaml.cs Container.RegisterType<IRestService, RestService2>() schrijven i.p.v. RestService..



Afbeelding 7: Toelichting code IRestService

### RestService

Met de klasse RestService gaan we de URL ophalen met de eventueel meegestuurde data.

De basis URL die we meegeven verwijst naar de URL van de API die in verbinding staat met de database.

GetRoutesAsync (zie afbeelding 8) gaat al de routes gaan opvragen uit de database. De link die gevorm wordt zal uiteindelijk deze worden: <http://api.beacons.ucll.be/v1/route>.

Als de data correct is aangekomen gaan we deze omvormen naar json formaat zodat we deze kunnen terugsturen en hierop bewerkingen kunnen doen.

De data bevat het route\_id en de naam van de route. Deze staan gedefinieerd in het Route model.

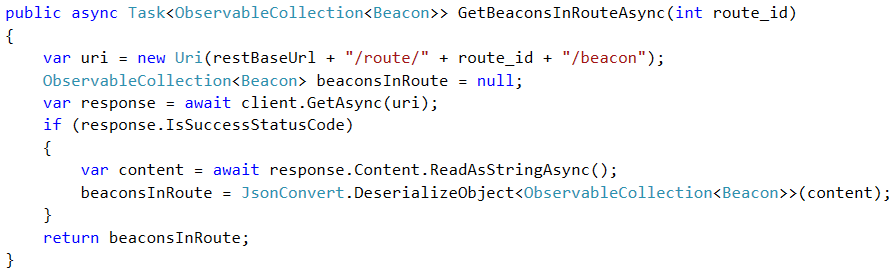
Met GetBeaconsInRouteAsync (zie afbeelding 9) verkrijgen we de data van al de beacons die zich bevinden in de route. Deze data wordt opgeroepen vanuit de database met het route id dat is meegestuurd. De link die gevorm wordt zal uiteindelijk deze worden: [http://api.beacons.ucll.be/v1/route/"beacon\_id"/beacon](http://api.beacons.ucll.be/v1/route/1/beacon).

Met behulp van GetContentForBeaconInRoute (zie afbeelding 10) kunnen we de data verkrijgen van een bepaalde beacon in een bepaalde route. Hierbij gaan we uit de database de content ophalen van de beacon (Html, YouTube, Audio, enz..).

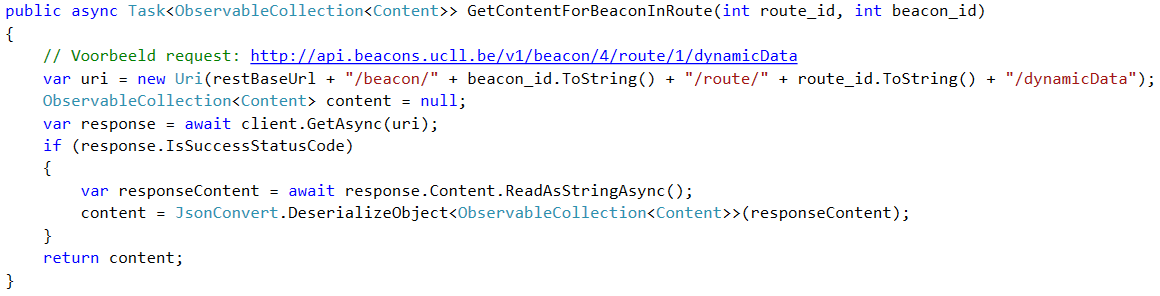
De link die gevormd wordt zal uiteindelijk deze worden: [http://api.beacons.ucll.be/v1/beacon/"beacon\_id"/route/"route\_id"/dynamicData](http://api.beacons.ucll.be/v1/beacon/6/route/2/dynamicData).



Afbeelding 8: Toelichting code GetRoutesAsync



Afbeelding 9: Toelichting code GetBeaconInRouteAsync



Afbeelding 10: Toelichting code GetContentForBeaconInRoute

### Viewmodels

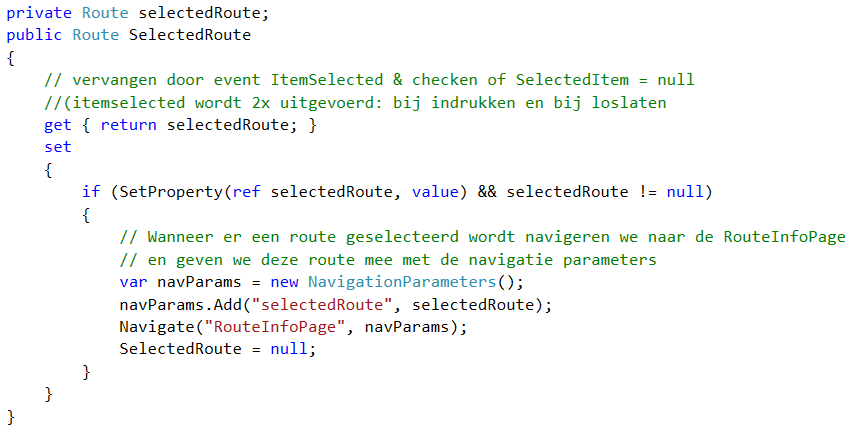
#### MainPageViewModel

De twee belangrijkste elementen in het MainPageModel is de selectedRoute en de Task LoadAndDisplayRoutes.

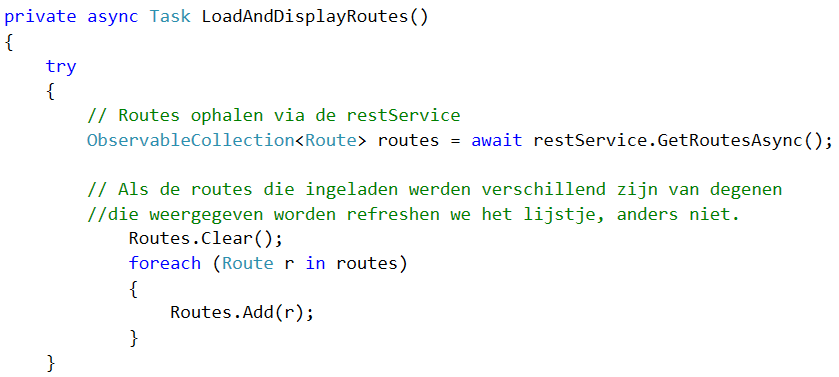
De selectedRoute wordt opgeroepen eenmaal als een gebruiker op een route heeft geklikt. We verkrijgen de parameters van deze route (Route\_id en route naam) om hier dan aan de parameters het id van de geselecteerde route extra mee te geven. Vervolgens roepen we Navigate op, deze gaat al de parameters meegeven inclusief de naam van de view die hij moet openen.

( await navigationService.NavigateAsync(page, navParams); ).

De gegevens worden in dit geval doorgestuurd naar het RouteInfoPageViewModel.



Afbeelding 11: Toelichting code SelectedRoute

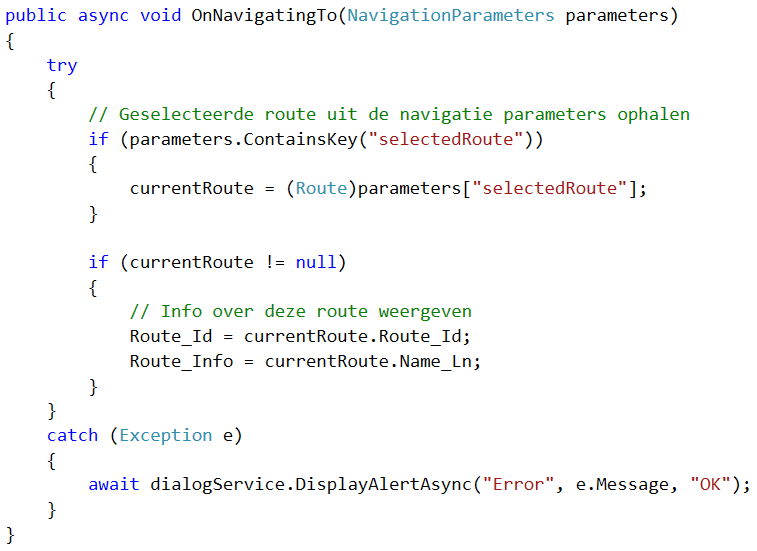
Deze methode gaat al de routes verkrijgen van de RestService. Deze wordt ook opgeroepen als je de refresh knop gebruikt.

Afbeelding 12: Toelichting code LoadAndDisplayRoutes

#### RouteInfoPageViewModel

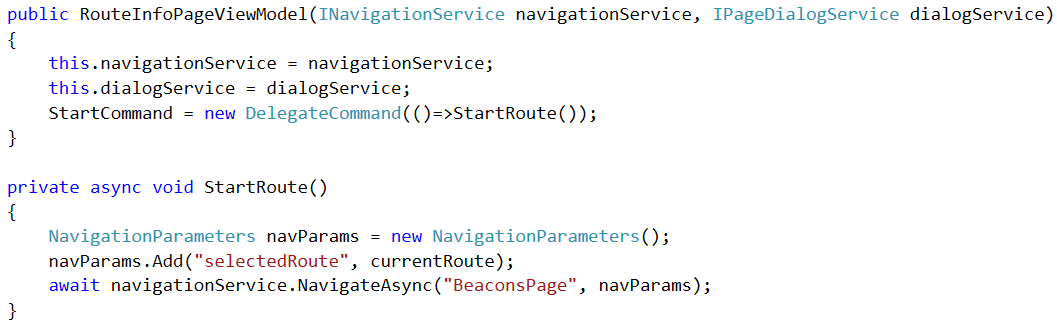
Als een gebruiker navigeert van de hoofdpagina, waar al de routes staan, naar een route die men heeft gekozen roept men via navigationService de pagina RouteInfoPage op. Door deze oproeping wordt de OnNavigatingTo opgeroepen.

De gegevens die worden meegegeven via de parameters van selectedRoute worden hier in Route\_id en Route\_info geplaatst.



Afbeelding 13: Toelichting code RouteInfoPageViewModel

Wil men een route starten drukt een gebruiker op de start knop. Hierbij wordt de methode StartRoute opgeroepen. Deze gaat de parameters ophalen van de route en deze doorgeven aan het BeaconsPageViewModel.



Afbeelding 14: Toelichting code StartRoute

#### BeaconsPageViewModel

Eenmaal als een gebruiker op de start knop heeft gedrukt worden de parameters doorgegeven en in currentRoute (zie afbeelding 16) gestoken.

Deze gaat erna het route\_id opvragen. Vervolgens kijken we na als de gebruiker toestemmingen heeft gegeven als wij zijn Bluetooth mogen gebruiken.

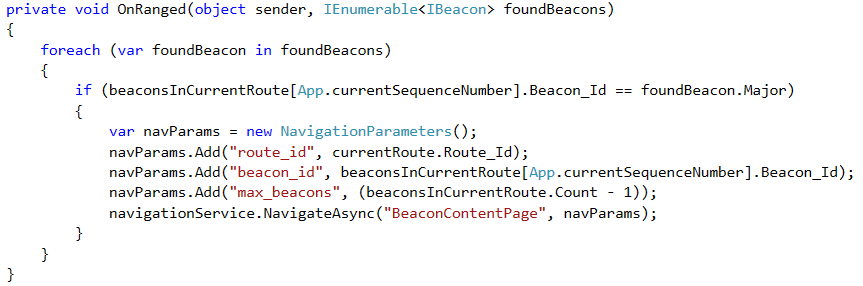
Als deze toestemming heeft gegeven kijken we na als bluetooth aanstaat, indien niet gooien we een error met de vraag om zijn bluetooth aan te zetten en navigeren we terug naar de vorige pagina.

Eenmaal als de route gestart is krijgt de gebruiker op het scherm te zien waar hij naar toe moet gaan en start de app met het scannen van beacons.

De methode Onranged (zie afbeelding 15) wordt gekoppeld aan het event Ranged van de estimote API.

De methode OnRanged wordt doorlopen zolang er beacons zijn en als de beacon gevonden is die in de route zit gaat hij de gegevens gaan meegeven via de parameters.

Deze parameters worden meegegeven aan de BeaconContentPageViewModel.



Afbeelding 15: Toelichting code OnRanged



Afbeelding 16: Toelichting code OnNavigatedTo

#### BeaconContentPageViewModel

# Screen mockups app (Werking)

## Splash screen

Als men de app opstart krijgt men eerst een splash screen dat het logo is van de organisatie.

## Front screen

Op het front screen krijgt men een opsomming van de verschillende routes die zich in de database bevinden.

Per route krijgt men een afbeelding en het Id van de route te zien, ook bevindt zich hier de naam.

Rechts in de bovenhoek bevindt zich een refresh knop. Deze knop gaat de pagina refreshen. Is er een update geweest in de database gaat deze de info opvragen en de pagina updaten als men de refresh knop heeft ingedrukt.

Als men een route selecteert gaat men verder naar het volgende scherm waar een knop getoond wordt.

Extra functie die er in zit is als men vergeet de wifi op te zetten van zijn apparaat. Hierbij krijgt de gebruiker een melding om de app af te sluiten en de wifi op te zetten.

Afbeelding 17: Screen mockup geen internet

Afbeelding 18: Screen mockup front screen

## Route screen

Op het route scherm krijgt de gebruiker in de titel te zien in welke route hij zit.

In het midden van het scherm bevindt zich een knop waar de gebruiker de route kan starten die hij heeft aangeduid vanuit het vorige menu.

Boven de knop komt nog te staan: “Welkom bij route-naam”.

Als de gebruiker de route wilt starten klikt hij op de knop en gaat de applicatie verder naar het volgende scherm.

Voordat hij volledig verder gaat krijgt de gebruiker een wacht icoon op het scherm.

Afbeelding 19: Screen mockup route screen

## C:\Users\jensie\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20170527-150324.pngContent screens

Als een gebruiker een bepaalde route gestart heeft, gaat de applicatie opzoek naar de beacon die het kortst bij de gebruiker bevindt. Eenmaal als de gebruiker kort bij een beacon komt wordt dit opgevangen door de applicatie en geeft deze bepaalde content naar verwant het soort media.

Als de gebruiker zijn bluetooth niet aanheeft krijgt men een melding dat deze de bluetooth moet aanzetten en gaat terug naar de vorige pagina.

Afbeelding 20: Screen mockup geen bluetooth

### Html

Als de media van het soort html is wordt er een html pagina getoond.

### YouTube

Is de media van het YouTube formaat (YouTube link) openen we een video van YouTube.

### Foto

Als we een foto aankrijgen als media tonen we de foto op het apparaat.

### Audio

Met het formaat audio spelen we de muziek af op het apparaat.

### Video

Als we van media een video hebben spelen we de video af op het apparaat van de gebruiker.

# Linken

* Developer android : <https://developer.android.com/index.html>
* Material design: <https://material.io/guidelines/style/color.html>
* Developer Xamarin: <https://developer.xamarin.com/>
* Material icons: <https://material.io/icons/>
* Icon & splash screen generator: <http://www.zelf-een-app-maken.nl/app-icon-splashscreen-generator/>
* Github: <https://github.com/jensMole/MobieleApp>
* GithubGist voor info over bepaalde code: <https://gist.github.com/> <https://gist.github.com/keannan5390/863c2072d1244fc3fde1>