

[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](#_Toc483991426)

[2.1 iBeacons 5](#_Toc483991427)

[2.2 Onze opdracht 6](#_Toc483991428)

[3 Opbouw app 7](#_Toc483991429)

[3.1 Klassendiagrammen 7](#_Toc483991430)

[3.1.1 Models 7](#_Toc483991431)

[3.1.2 ViewModels 8](#_Toc483991432)

[3.1.3 Interface 9](#_Toc483991433)

[3.1.4 Data 9](#_Toc483991434)

[4 Screen mockups app (Werking) 14](#_Toc483991435)

[4.1 Splash screen 14](#_Toc483991436)

[4.2 Front screen 14](#_Toc483991437)

[4.3 Route screen 15](#_Toc483991438)

[4.4 Content screens 16](#_Toc483991439)

[4.4.1 Html 16](#_Toc483991440)

[4.4.2 YouTube 16](#_Toc483991441)

[4.4.3 Foto 16](#_Toc483991442)

[4.4.4 Audio 16](#_Toc483991443)

[4.4.5 Video 16](#_Toc483991444)

[5 Linken 17](#_Toc483991445)

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: Screen mockup geen internet 14

Afbeelding 7: Screen mockup front screen 14

Afbeelding 8: Screen mockup route screen 15

Afbeelding 9: Screen mockup geen bluetooth 16

# 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: Hierbij maken we een HttpClient aan;
* 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.

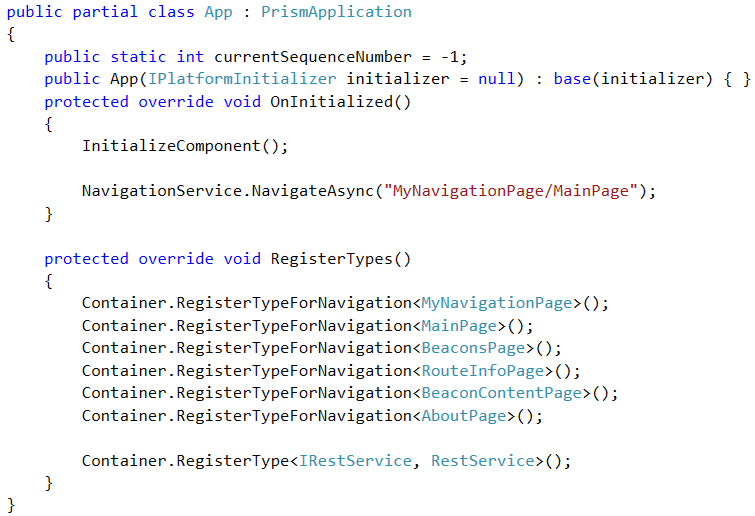
Afbeelding 5: Klassendiagrammen data

## 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.

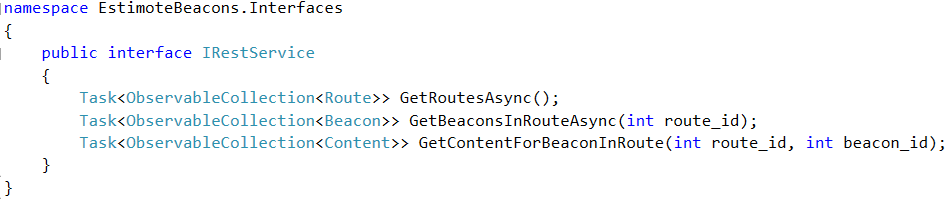
Afbeelding 6: Toelichting code App.xaml.cs

### IRestService

De IrestService is een interface die we gebruiken om een melding te geven aan een bepaalde functie. Deze wordt opgeroepen als een item wordt toegevoegd, verwijderd of als de hele lijst vernieuwd wordt.

De verschillende Tasks worden opgeroepen als er iets aanpast van wat er in de klassen staat.

Zoals bij de klasse Route, als hier een route\_id aangepast wordt gaat GetRoutesAsync() worden opgeroepen.



**in MainPageViewModel.cs**

private Route selectedRoute;

public Route SelectedRoute

{

// vervangen door event ItemSelected & checken of SelectedItem = null

//(itemselected wordt 2x uitgevoerd: bij indrukken en bij loslaten

get { return selectedRoute; }

set

{

if (SetProperty(ref selectedRoute, value) && selectedRoute != null)

{

// Wanneer er een route geselecteerd wordt navigeren we naar de RouteInfoPage

// en geven we deze route mee met de navigatie parameters

var navParams = new NavigationParameters();

navParams.Add("selectedRoute", selectedRoute);

Navigate("RouteInfoPage", navParams);

SelectedRoute = null;

}

}

De geselecteerde route in property steken.

Het inladen van alle routes uit de restservice als er internetconnectie is en tonen van de routes op de MainPage.

private async Task LoadAndDisplayRoutes()

{

try

{

// Routes ophalen via de restService

ObservableCollection<Route> routes = await restService.GetRoutesAsync();

//if (routes != Routes)

//{

// Als de routes die ingeladen werden verschillend zijn van degenen die   
 weergegeven worden refreshen we het lijstje, anders niet.

Routes.Clear();

foreach (Route r in routes)

{

Routes.Add(r);

}

//}

}

catch (HttpRequestException)

{

await dialogService.DisplayAlertAsync("Geen internetverbinding.", "Sluit de   
 app volledig af en zorg voor een werkende internetverbinding.", "OK");

}

catch (WebException)

{

await dialogService.DisplayAlertAsync("Geen internetverbinding.", "Sluit de  
 app volledig af en zorg voor een werkende internetverbinding.", "OK");

}

catch (Exception e)

{

await dialogService.DisplayAlertAsync("Error", e.Message, "OK");

}

}

**in BeaconsPageViewModel.cs**

private void NextBeacon()

{

Beacons = "";

if (App.currentSequenceNumber < beaconsInCurrentRoute.Count - 1)

{

App.currentSequenceNumber++;

Beacons = "Ga naar: " + beaconsInCurrentRoute[App.currentSequenceNumber].Location\_Ln;

}

else

{

Beacons = "last beacon reached";

}

}

private void OnRanged(object sender, IEnumerable<IBeacon> foundBeacons)

{

foreach (var foundBeacon in foundBeacons)

{

if (beaconsInCurrentRoute[App.currentSequenceNumber].Beacon\_Id == foundBeacon.Major)

{

var navParams = new NavigationParameters();

navParams.Add("route\_id", currentRoute.Route\_Id);

navParams.Add("beacon\_id",  
 beaconsInCurrentRoute[App.currentSequenceNumber].Beacon\_Id);

navParams.Add("max\_beacons", (beaconsInCurrentRoute.Count - 1));

navigationService.NavigateAsync("BeaconContentPage", navParams);

}

}

}

Het ordenen van de beacons van de geselecteerde route op hun volgnummer.

**in BeaconContentPageViewModel.cs**

public async void OnNavigatingTo(NavigationParameters parameters)

{

int route\_id = -1;

int beacon\_id = -1;

if (parameters.ContainsKey("route\_id")) route\_id = (int)parameters["route\_id"];

if (parameters.ContainsKey("beacon\_id")) beacon\_id = (int)parameters["beacon\_id"];

if (parameters.ContainsKey("max\_beacons")) max\_beacons = (int)parameters["max\_beacons"];

ObservableCollection<Content> content = await  
 \_restService.GetContentForBeaconInRoute(route\_id, beacon\_id);

SourceUrl = content[0].Content\_Txt;

}

De content van de beacon tonen.

**in** **RouteInfoPageViewModel.cs**

private async void StartRoute()

{

NavigationParameters navParams = new NavigationParameters();

navParams.Add("selectedRoute", currentRoute);

await navigationService.NavigateAsync("BeaconsPage", navParams);

}

De route starten door naar de BeaconsPage te navigeren en de geselcteerde route mee te sturen.

**in** **Restservice.cs**

public async Task<ObservableCollection<Beacon>> GetBeaconsInRouteAsync(int route\_id)

{

var uri = new Uri(restBaseUrl + "/route/" + route\_id + "/beacon");

ObservableCollection<Beacon> beaconsInRoute = null;

var response = await client.GetAsync(uri);

if (response.IsSuccessStatusCode)

{

var content = await response.Content.ReadAsStringAsync();

beaconsInRoute = JsonConvert.DeserializeObject<ObservableCollection<Beacon>>(content);

}

return beaconsInRoute;

}

Beacons van meegestuurde route binnenhalen van restservice.   
(URL: http://api.beacons.ucll.be/v1/route/”route\_id”/beacon)

public async Task<ObservableCollection<Route>> GetRoutesAsync()

{

var uri = new Uri(restBaseUrl + "/route");

ObservableCollection<Route> routes = null;

var response = await client.GetAsync(uri);

if (response.IsSuccessStatusCode)

{

var content = await response.Content.ReadAsStringAsync();

routes = JsonConvert.DeserializeObject<ObservableCollection<Route>>(content);

}

return routes;

}

Routes binnenhalen van restservice.   
(URL: http://api.beacons.ucll.be/v1/route)

public async Task<ObservableCollection<Content>> GetContentForBeaconInRoute(int route\_id,

int beacon\_id)

{

// Voorbeeld request: http://api.beacons.ucll.be/v1/beacon/4/route/1/dynamicData

var uri = new Uri(restBaseUrl + "/beacon/" + beacon\_id.ToString() + "/route/" +

route\_id.ToString() + "/dynamicData");

ObservableCollection<Content> content = null;

var response = await client.GetAsync(uri);

if (response.IsSuccessStatusCode)

{

var responseContent = await response.Content.ReadAsStringAsync();

content= JsonConvert.DeserializeObject<ObservableCollection<Content>>(responseContent);

}

return content;

}

Content van meegestuurde beacon binnen meegsestuurde route binnenhalen van restservice.   
(URL: http://api.beacons.ucll.be/v1/beacon/”beacon\_id”/route/”route\_id”/ dynamicData)

# 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 7: Screen mockup geen internet

Afbeelding 8: 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 9: 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 10: 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>