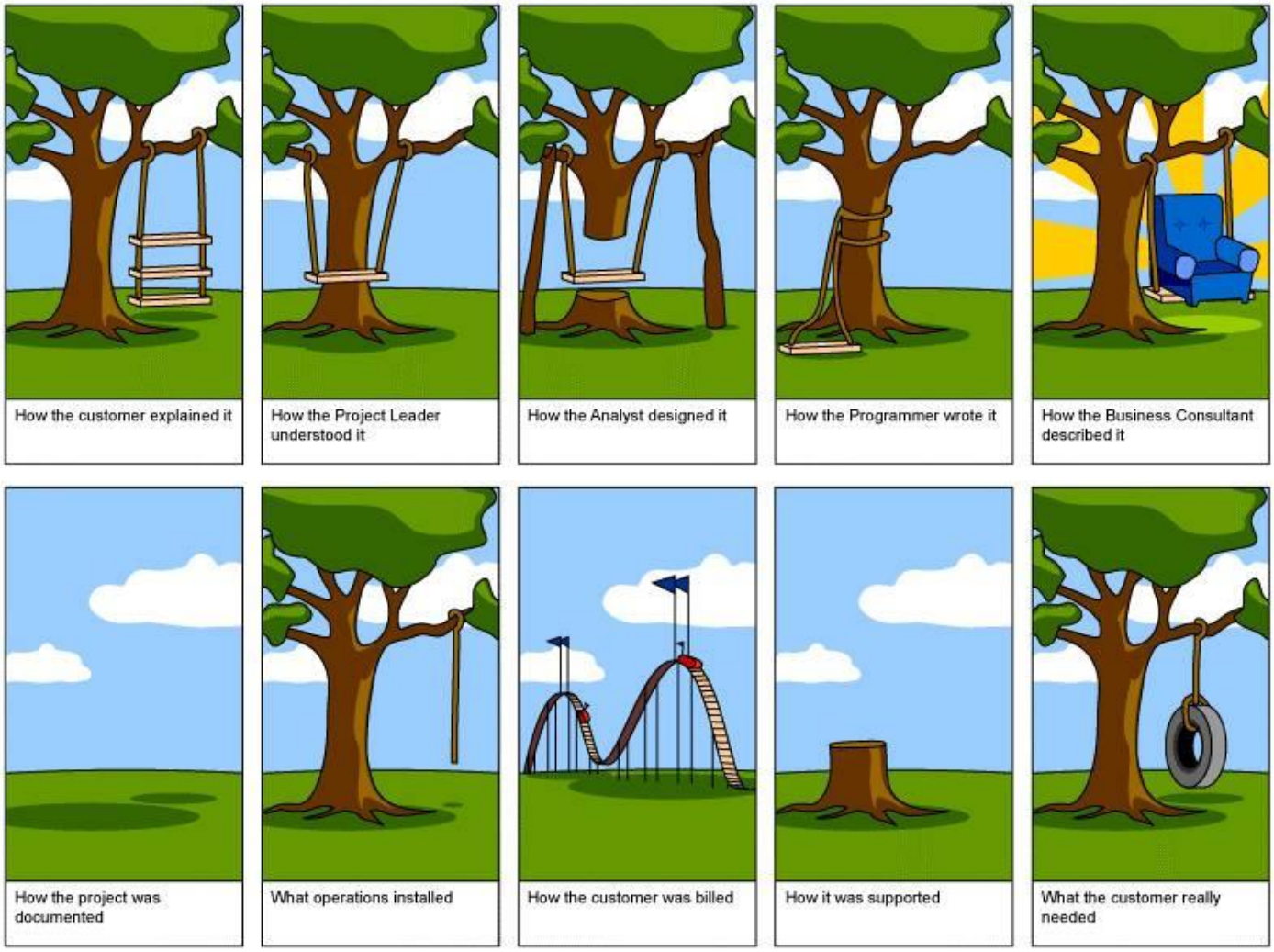


PROJECT DOSSIER MOBIELE APP

[UCLL BEACONS]

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

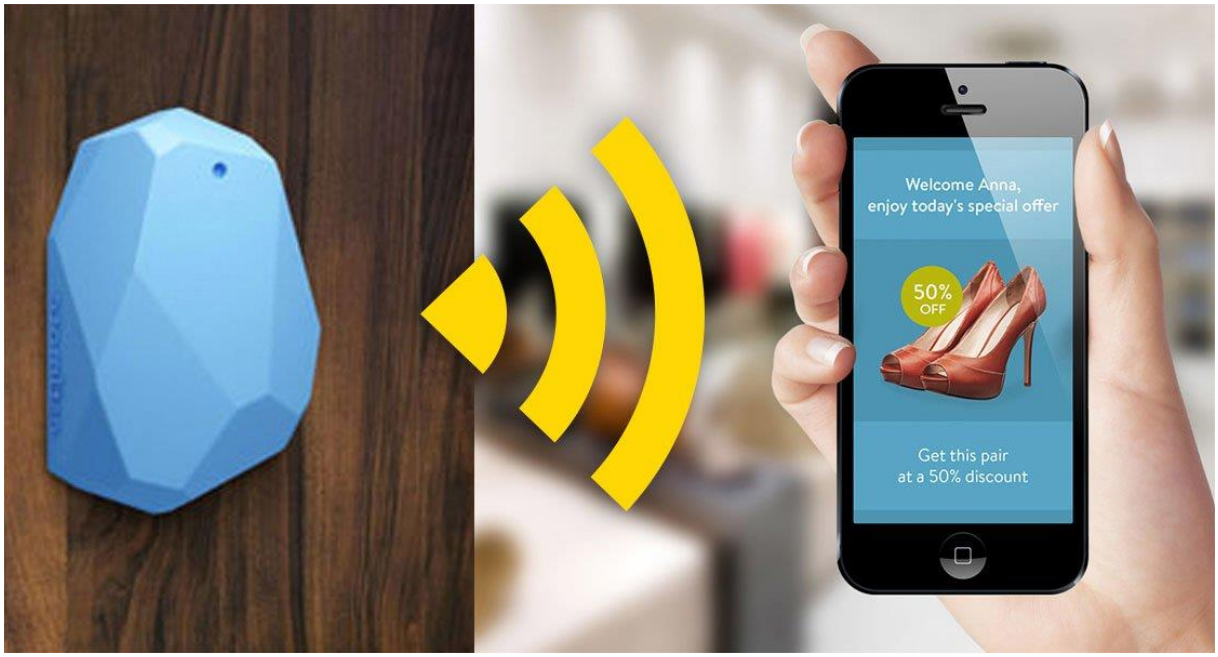
2 PROJECT OMSCHRIJVING

2.1 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.

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…)

2.2 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 samen met ICT Project.

- 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.

3 OPBOUW APP

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

3.1 Klassendiagrammen

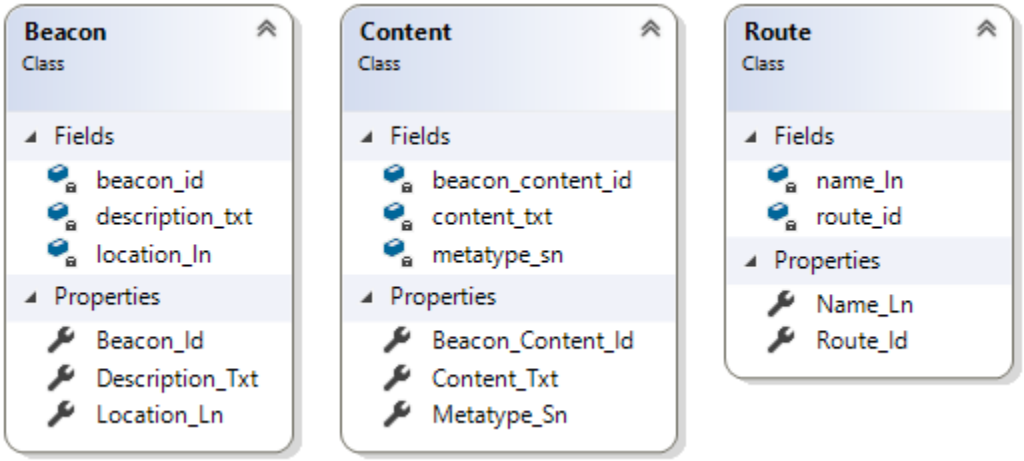
3.1.1 MODELS

Models zorgen voor de data aan de views door binding. Dit gebeurd via de viewmodels.
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.

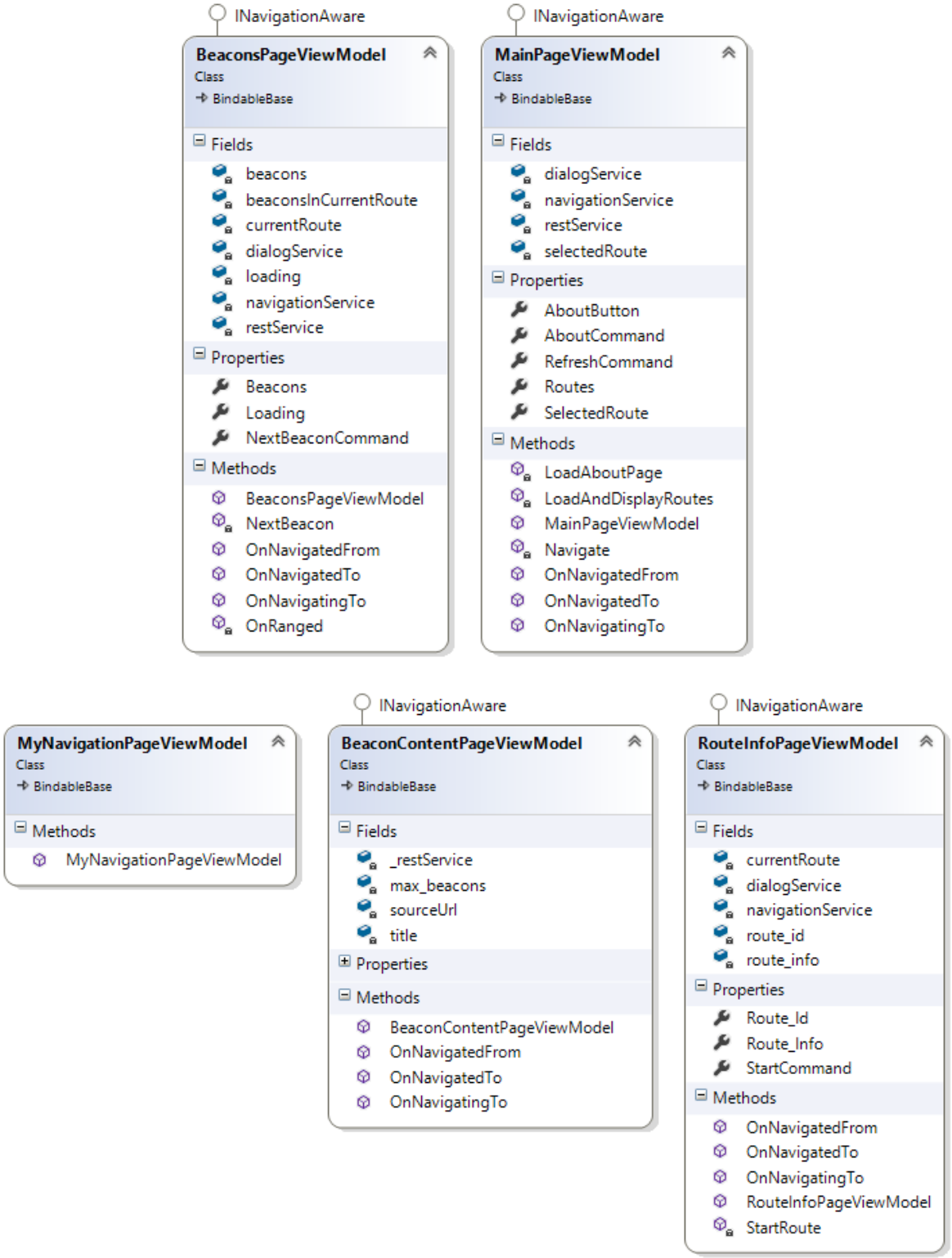
3. Route:
    - Name_Ln: De naam van de route verkrijgen;
    - Route_Id: Het Id van de route.



AFBEELDING 2: KLASSENDIAGRAMMEN MODELS

3.1.2 VIEWMODELS

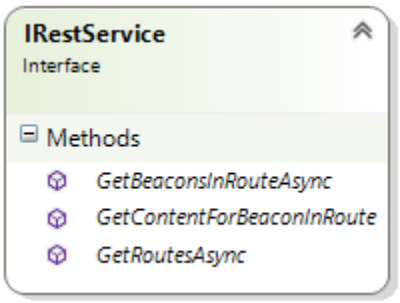
Een viewmodel haalt gegevens uit het model en manipuleert deze naar de indeling die vereist is voor de view. Het geeft aan de view door of de onderliggende gegevens in het model zijn gewijzigd. De data wordt in het model bijgewerkt naar aanleiding van UI-gebeurtenissen in de view.



AFBEELDING 3: KLASSENDIAGRAMMEN VIEWMODELS

3.1.3 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. Als een klasse een interface implementeert, biedt de klasse een implementatie voor alle door de interface gedefinieerde leden. De interface biedt geen functionaliteit die een klasse kan erven.

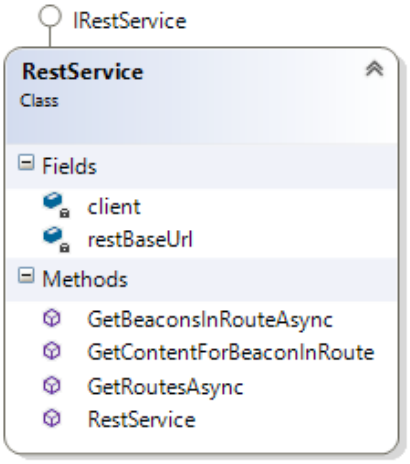


AFBEELDING 4: KLASSENDIAGRAMMEN INTERFACE

3.1.4 DATA

De Restservice gaat met behulp van een URL voor een verbinding met de database zorgen.
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 de beacons kan opvragen;
- GetContentForBeaconInRoute: Als we dit oproepen gaat deze met behulp van een route_id en een beacon_id, de content opvragen van een beacon in die route.



AFBEELDING 5: KLASSENDIAGRAMMEN DATA

3.2 Toelichting code

3.2.1 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. Deze verschillende klassen kunnen met elkaar data 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.

```
public partial class App : PrismApplication
{
    public static int currentSequenceNumber = -1;
    public App(IPlatformInitializer initializer = null) : base(initializer) { }
    protected override void OnInitialized()
    {
        InitializeComponent();

        NavigationService.NavigateAsync("MyNavigationPage/MainPage");
    }

    protected override void RegisterTypes()
    {
        Container.RegisterTypeForNavigation<MyNavigationPage>();
        Container.RegisterTypeForNavigation<MainPage>();
        Container.RegisterTypeForNavigation<BeaconsPage>();
        Container.RegisterTypeForNavigation<RouteInfoPage>();
        Container.RegisterTypeForNavigation<BeaconContentPage>();
        Container.RegisterTypeForNavigation<AboutPage>();

        Container.RegisterType<IRestService, RestService>();
    }
}
```

AFBEELDING 6: TOELICHTING CODE APP.XAML.CS

3.2.2 IRESTSERVICE

IRestService is een interface die dient als template voor de RestServiceklasse. Hierin staan de methodes gedefinieerd die in een RestService moeten zitten. Doordat we deze hebben aangemaakt kunnen we gebruik maken van dependency injection. Dit zorgt 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. Indien we gebruik willen maken van RestService2 moeten we maar één ding veranderen in onze andere code. Namelijk in de App.xaml.cs Container.RegisterType<IRestService, RestService2>() schrijven i.p.v. RestService.

```
namespace EstimoteBeacons.Interfaces
{
    public interface IRestService
    {
        Task<ObservableCollection<Route>> GetRoutesAsync();
        Task<ObservableCollection<Beacon>> GetBeaconsInRouteAsync(int route_id);
        Task<ObservableCollection<Content>> GetContentForBeaconInRoute(int route_id, int beacon_id);
    }
}
```

AFBEELDING 7: TOELICHTING CODE IRESTSERVICE

3.2.3 RESTSERVICE

Met de klasse RestService gaan we de URL ophalen met de eventuele 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 opvragen uit de database. De link die gevormd wordt zal uiteindelijk deze worden: http://api.beacons.ucll.be/v1/route.
Als de gegevens correct zijn aangekomen gaan we deze omvormen naar json formaat zodat we deze kunnen terugsturen en hiermee bewerkingen kunnen doen.
De gegevens bevatten 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 gegevens worden opgeroepen vanuit de database met het meegestuurde route id. De link die gevorm wordt zal uiteindelijk deze worden:
http://api.beacons.ucll.be/v1/route/"beacon_id"/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.

```
private static string restBaseUrl = "http://api.beacons.ucll.be/v1";
private HttpClient client;
public RestService()
{
    client = new HttpClient();
}

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;
}
```

AFBEELDING 8: TOELICHTING CODE GETROUTESASYNC

```
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;
}
```

AFBEELDING 9: TOELICHTING CODE GETBEACONINROUTEASYNC

```
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;
}
```

AFBEELDING 10: TOELICHTING CODE GETCONTENTFORBEACONINROUTE

3.2.4 VIEWMODELS

3.2.4.1 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). Aan de parameters wordt het id van de geselecteerde route extra meegegeven. 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.

```
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;
        }
    }
}
```

AFBEELDING 11: TOELICHTING CODE SELECTEDROUTE

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

```
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);
            }
        }
    }
```

AFBEELDING 12: TOELICHTING CODE LOADANDDISPLAYROUTES

3.2.4.2 RouteInfoPageViewModel

Als een gebruiker navigeert van de hoofdpagina 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.

```
public async void OnNavigatingTo(NavigationParameters parameters)
{
    try
    {
        // Geselecteerde route uit de navigatie parameters ophalen
        if (parameters.ContainsKey("selectedRoute"))
        {
            currentRoute = (Route)parameters["selectedRoute"];
        }

        if (currentRoute != null)
        {
            // Info over deze route weergeven
            Route_Id = currentRoute.Route_Id;
            Route_Info = currentRoute.Name_Ln;
        }
    }
    catch (Exception e)
    {
        await dialogService.DisplayAlertAsync("Error", e.Message, "OK");
    }
}
```

AFBEELDING 13: TOELICHTING CODE ROUTEINFOPAGEVIEWMODEL

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

```
public RouteInfoPageViewModel(INavigationService navigationService, IPageDialogService dialogService)
{
    this.navigationService = navigationService;
    this.dialogService = dialogService;
    StartCommand = new DelegateCommand(()=>StartRoute());
}

private async void StartRoute()
{
    NavigationParameters navParams = new NavigationParameters();
    navParams.Add("selectedRoute", currentRoute);
    await navigationService.NavigateAsync("BeaconsPage", navParams);
}
```

AFBEELDING 14: TOELICHTING CODE STARTROUTE

3.2.4.3 BeaconsPageViewModel

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

Deze gaat een route_id opvragen. Vervolgens kijken we na als de gebruiker toestemmingen heeft gegeven of de applicatie zijn Bluetooth mag gebruiken.
Heeft de gebruiker toestemming gegeven, kijken we na of bluetooth aanstaat. Indien niet verschijnt er een error met de vraag om 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 wordt doorlopen zolang er beacons zijn. Als de beacon gevonden is gaat de applicatie de gegevens meegeven via de parameters.
Deze parameters worden meegegeven aan de BeaconContentPageViewModel.

```
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);
        }
    }
}
```

AFBEELDING 15: TOELICHTING CODE ONRANGED

```
public async void OnNavigatedTo(NavigationParameters parameters)
{
    try
    {
        // Geselecteerde route ophalen uit navigatieparameters
        if (parameters.ContainsKey("selectedRoute"))
        {
            currentRoute = (Route)parameters["selectedRoute"];
        }

        if (currentRoute != null)
        {
            // Beacons die bij de geselecteerde route horen ophalen via de restService
            beaconsInCurrentRoute = await restService.GetBeaconsInRouteAsync(currentRoute.Route_Id);
        }

        // Android permissions check
        var status = await CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Location);
        if (status != PermissionStatus.Granted)
        {
            if (await CrossPermissions.Current.ShouldShowRequestPermissionRationaleAsync(Permission.Location))
            {
                await dialogService.DisplayAlertAsync("Need location", "This app needs location services", "OK");
            }

            var results = await CrossPermissions.Current.RequestPermissionsAsync(new[] { Permission.Location });
            status = results[Permission.Location];
        }
       // Checken of de gebruiker toestemming heeft gegeven
       if (status == PermissionStatus.Granted)
       {
           var status2 = await EstimoteManager.Instance.Initialize();

           // Checken of bluetooth aan staat (android)
           if (BluetoothAdapter.DefaultAdapter.IsEnabled)
           {
               if (status2 != BeaconInitStatus.Success)
               {
                   return;
               }
               Beacons = "";
               Beacons = "Ga naar: " + beaconsInCurrentRoute[App.currentSequenceNumber].Location_Ln;
               // Als bluetooth aan staat beginnen we met scannen naar beacons
               EstimoteManager.Instance.Ranged += OnRanged;
               EstimoteManager.Instance.StartRanging(new BeaconRegion("estimote", "B9407F30-F5F8-466E-AFF9-25556B57FE6D"));
           }else
           {
               // Als bluetooth uit staat krijgt de gebruiker hier een melding van en wordt er terug genavigeerd
               await dialogService.DisplayAlertAsync("Geen Bluetooth", "Bluetooth moet ingeschakeld zijn om een route te starten.", "OK");
               await navigationService.GoBackAsync();
           }
       }
       else if (status != PermissionStatus.Unknown)
       {
           await dialogService.DisplayAlertAsync("Location Denied", "Can not continue, try again.", "OK");
       }
    }
    catch (Exception ex)
    {
        Beacons = "Error: " + ex;
    }
}
```

AFBEELDING 16: TOELICHTING CODE ONNAVIGATEDTO

3.2.4.4 BeaconContentPageViewModel

Via het BeaconPageViewModel worden de parameters meegegeven bij het naderen van een beacon.

De methode OnNavigatedTo wordt opgeroepen en gaat met behulp van de parameters de gegevens ophalen. Als de gegevens opgehaald zijn gaan we nakijken wat voor soort metatype het gegeven is. Aan de hand van het soort type geven we mee welke zichtbaar mag gemaakt worden op de view (zie afbeelding 18). Ook word het sequence nummer verhoogt zodat de applicatie opzoek gaat naar de volgende beacon.

```
public async void OnNavigatedTo(NavigationParameters parameters)
{
    ObservableCollection<Content> contentCollection = await _restService.GetContentForBeaconInRoute(route_id, beacon_id);
    Content content = contentCollection[0];

    if (content.Metatype_Sn == "link" || content.Metatype_Sn == "audio")
    {
        WebViewVisible = true;
        SourceUrl = content.Content_Txt;
    }
    else if (content.Metatype_Sn == "image")
    {
        ImageVisible = true;
        ImageSource = content.Content_Txt;
    }
    else if (content.Metatype_Sn == "html")
    {
        HTMLWebViewVisible = true;
        HTMLSource = content.Content_Txt;
    }
}
```

AFBEELDING 17: TOELICHTING CODE BEACONCONTENTPAGEVIEWMODEL

```
<AbsoluteLayout>
    <Image IsVisible="{Binding ImageVisible}" Source="{Binding ImageSource}"
           Aspect="AspectFit" AbsoluteLayout.LayoutBounds="1,1,1,1" AbsoluteLayout.LayoutFlags="All"/>

    <WebView IsVisible="{Binding WebViewVisible}" Source="{Binding SourceUrl}"
             AbsoluteLayout.LayoutBounds="1,1,1,1" AbsoluteLayout.LayoutFlags="All"/>

    <WebView IsVisible="{Binding HTMLWebViewVisible}"
             AbsoluteLayout.LayoutBounds="1,1,1,1" AbsoluteLayout.LayoutFlags="All">

        <WebView.Source>
            <HtmlWebViewSource Html="{Binding HTMLSource}" />
        </WebView.Source>

    </WebView>
</AbsoluteLayout>
```

AFBEELDING 18: TOELICHTING CODE BEACONCONTENTPAGE

```
public void OnNavigatedFrom(NavigationParameters parameters)
{
    if (App.currentSequenceNumber < max_beacons)
    {
        App.currentSequenceNumber++;
    }
}
```

AFBEELDING 19: TOELICHTING CODE ONNAVIGATEDFROM

4 SCREEN MOCKUPS APP

4.1 Splash screen

Bij het opstarten van de app krijgt de gebruiker een splash screen dat het logo van de organisatie is.

4.2 Front screen

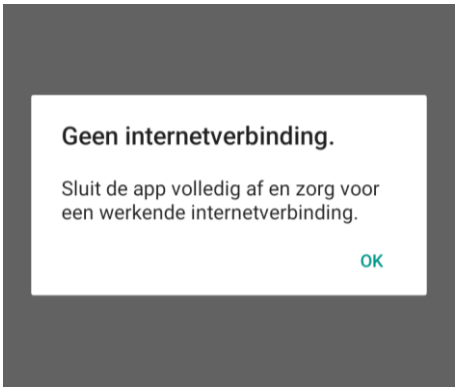
Op het front screen krijgt men een opsomming van de verschillende routes die zich in de database bevinden.

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 de app de info opvragen en de pagina updaten.

Als de gebruiker een route selecteert gaat zij/hij verder naar het volgende scherm waar een knop getoond wordt om de route te starten.

Een extra functie geeft een melding als men vergeet de wifi op te zetten. Hierbij krijgt de gebruiker een pop-up om de app af te sluiten en de wifi op te zetten.



AFBEELDING 20: SCREEN MOCKUP GEEN INTERNET

AFBEELDING 21: SCREEN MOCKUP FRONT SCREEN

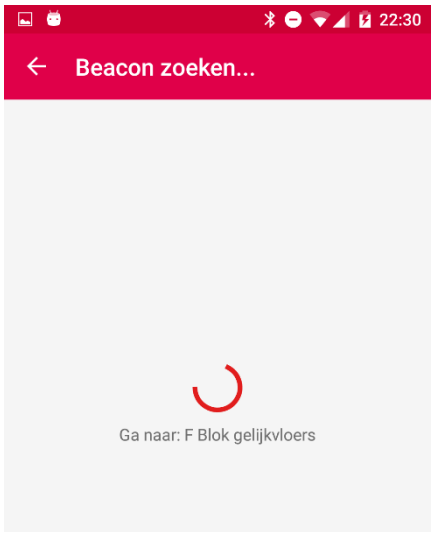
4.3 Route screen

Op het route screen krijgt de gebruiker in de titel balk de gekozen route te zien.

In het midden van het scherm bevindt zich een knop waar de gebruiker de route kan starten.

Boven de knop komt nog te staan: “Welkom bij de route: 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 met in de titel Beacon zoeken.

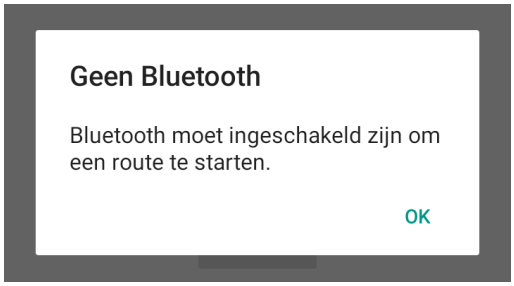


AFBEELDING 22: SCREEN MOCKUP WACHT SCHERM

AFBEELDING 23: SCREEN MOCKUP ROUTE SCREEN

4.4 Content screens

Als een gebruiker een bepaalde route gestart heeft, gaat de applicatie opzoek naar de beacon die zich het kortst bij de gebruiker bevindt. Eenmaal als de gebruiker kort bij een beacon komt wordt dit opgevangen door de applicatie en wordt de media getoond.
Als de gebruiker zijn bluetooth niet aanheeft krijgt deze een melding dat bluetooth aangezet moet worden en de applicatie gaat terug naar de vorige pagina.



AFBEELDING 24: SCREEN MOCKUP GEEN BLUETOOTH

4.4.1 HTML

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

4.4.2 YOUTUBE

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

4.4.3 FOTO

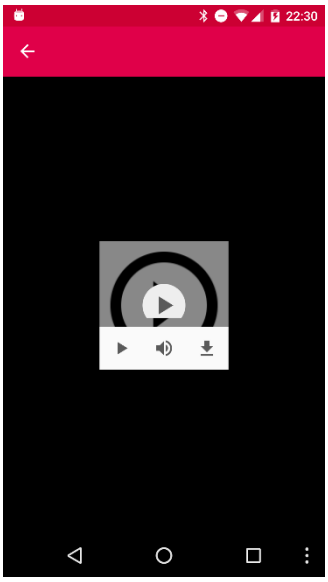
Als we een foto aankrijgen als media tonen we de foto op het apparaat.

4.4.4 AUDIO

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

4.4.5 VIDEO

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



AFBEELDING 25: SCREEN MOCKUP CONTENT

4.5 Over screen

Als de gebruiker info over de applicatie wilt hebben kan deze op het front screen op de 3 puntje drukken. Deze brengen je naar de over pagina waar meer info opstaat hoe je de app moet gebruiken.

AFBEELDING 26: SCREEN MOCKUP ABOUTSCREEN

5 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