# *Locator App*

Waar is Bambi?

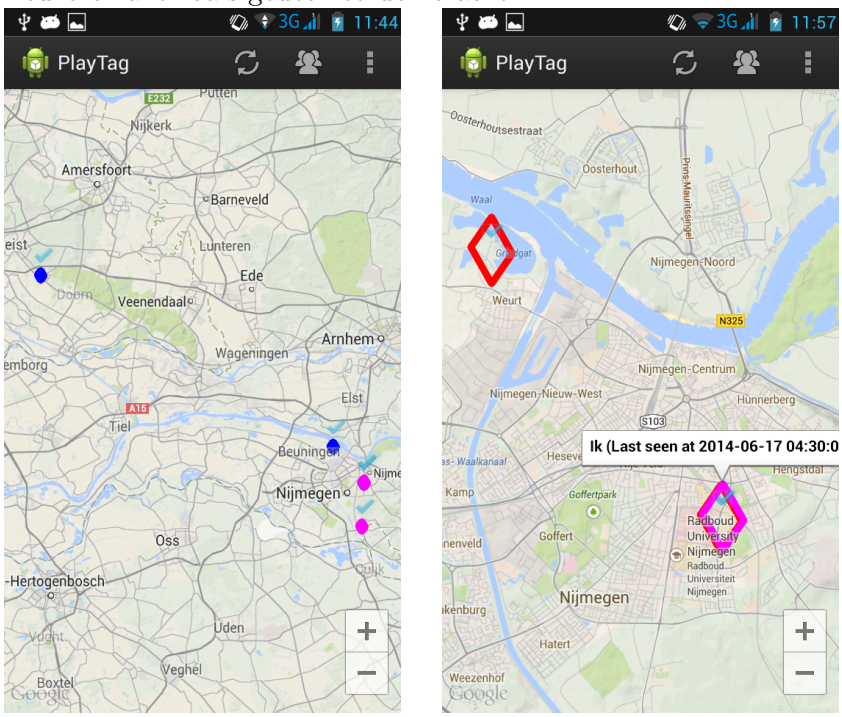
**Jordy Gennissen**
**Hessel Bongers**
**Brandaan Brouwers**

# 1 Voorwoord

- Dit document dient als uitleg over onze Locator App, die we tijdens het vak *Research and Development* hebben gemaakt. In dit document zullen we het hebben over wat onze app inhoud, wat je ermee kunt doen, maar we zullen ook op een wat dieper niveau op onze app ingaan. Verder zullen we uitleggen waarom we sommige keuzes hebben gemaakt, en uiteindelijk een reflectie geven over ons project, en de samenwerking hierbij.

# 2 Beschrijving

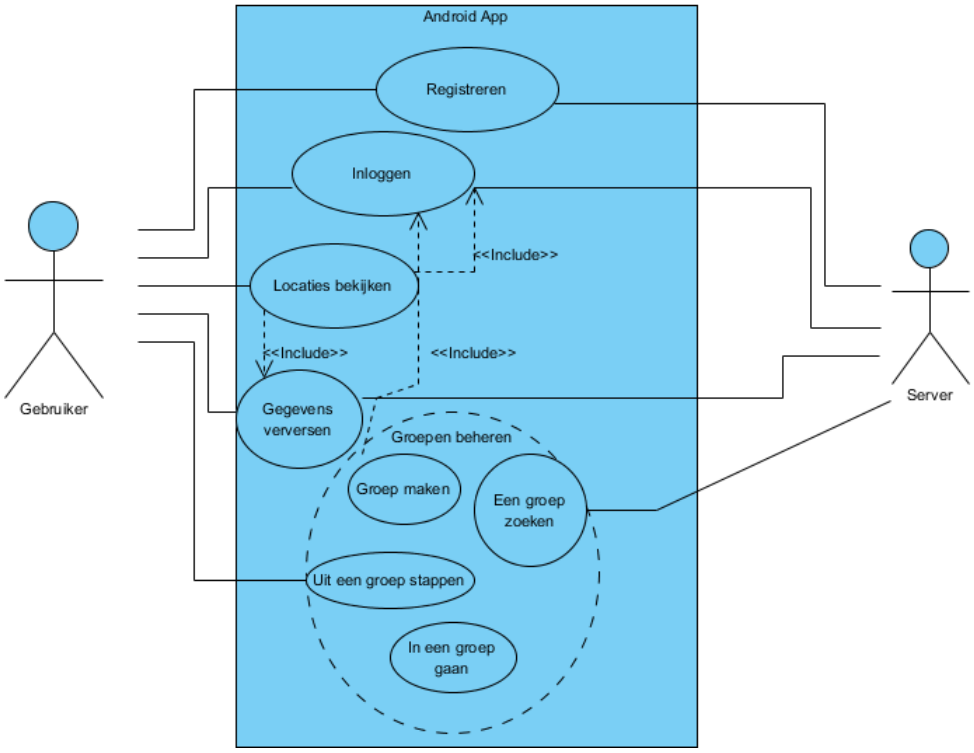
- **Inleiding**
  - De Locator app is een app waarmee je de locaties kunt zien van jezelf, en de mensen waarmee je in een groep zit. Je kan zelf een groep aanmaken, daarnaast kan je ook lid worden van een al eerder aangemaakte groep. Het idee is dat dit een basis wordt van meerdere apps, zoals een tikkertje app, of misschien een serieuzere kant zoals gedetineerden tracken.




Hierboven zie je hoe de locaties op de kaart worden aangegeven. Verschillende groepen hebben verschillende kleuren, maar kunnen dezelfde kleur hebben. Deze kleuren worden random bepaalt op het moment dat de kaart (opnieuw) geladen wordt. Verder is het

tweede plaatje een ingezoomd beeld van het eerste, waarbij de ruit groter wordt, naarmate de schaal kleiner wordt. Deze ruitjes zijn een schatting waar de persoon is, aangezien de ruitjes vast staan in grootte.

- **Productverantwoording**
  - Naar hoeverre wij weten, is er geen ander product zoals ons product op de markt. Er zijn natuurlijk wel apps waarbij je je eigen locatie op een kaart kan zien, denk hierbij aan *Google Maps*, maar er is naar ons weten geen app waarbij je van andere mensen de locatie kan zien, met uitzondering van *Facebook*, waar je de locatie kan zien waar iemand een status-update heeft gedaan. Aangezien onze app tot heel veel dingen kan worden uitgebreid, en wij niet op de hoogte zijn van andere apps die deze features hebben, denken we dat onze app wel iets gaat toevoegen op de markt. Denk hierbij aan spelletjes zoals tikkertje via de mobiel, of aan het meer security geven bij bijvoorbeeld een app zoals *CuraLoci*, waarbij je personen in een vertrouwensgroep elkaars locatie kan laten zien, zodat je weet of iemand een buurtgenoot is op het moment dat er iets verdachts gebeurt.
- **Specificaties: funcionele eigenschappen**
  - Bij onze app kun je inloggen, registreren, een groep maken, lid worden van een groep, een groep verlaten, een groep bekijken, een groep zoeken, locaties van groepsleden bekijken op de kaart en de kaart verversen. Om het duidelijker te maken hebben we hieronder een use-case model.



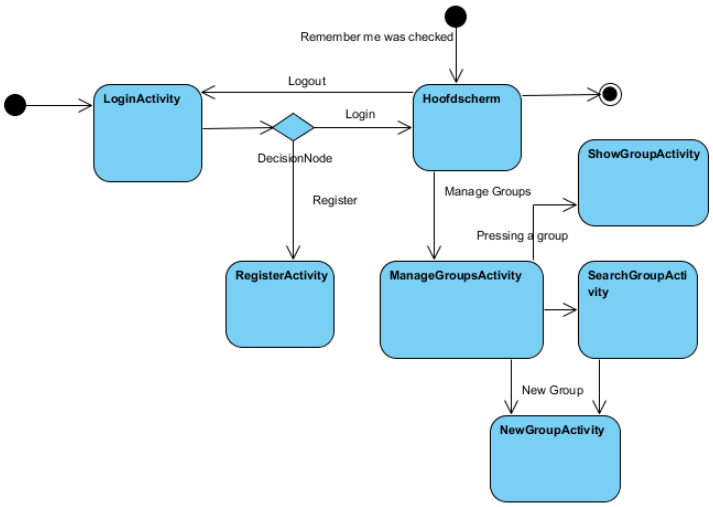

- **Specificaties: non-functionele eigenschappen**
  - De app gebruikt *Fine Location* ofwel *GPS* (om een bruikbare App te hebben moet je Coarse Location of Fine Location toestaan en gebruiken)
  - De app gebruikt *Coarse Location* ofwel locatiebepaling aan de hand van een 3G-verbinding of een WiFi-verbinding. (om een bruikbare app te hebben moet je Coarse Location of Fine Location toestaan en gebruiken)
  - De app gebruikt *Internet* (benodigd voor de app).

# 3 Ontwerp

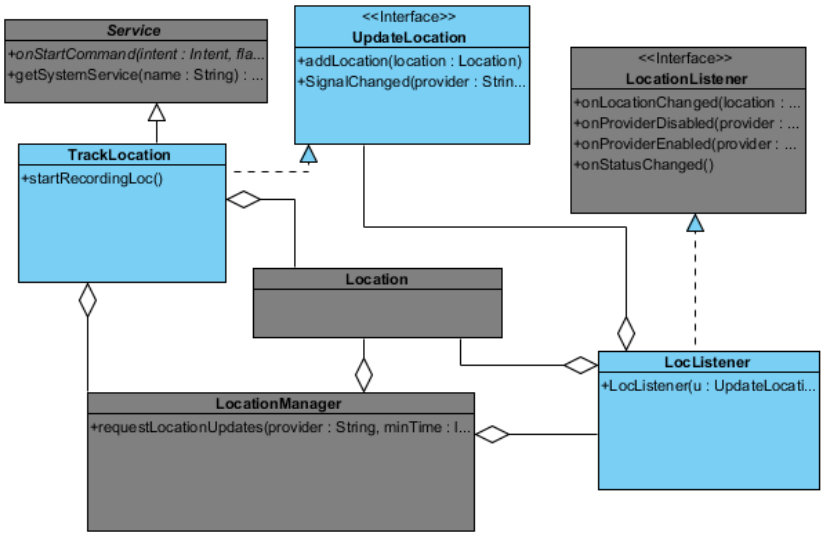
- **Globaal ontwerp**
  - Onze app kan in een paar onderdelen verdeeld worden: Inloggen, locaties bekijken, en groepen beheren. Hierbij moet je ingelogd zijn voordat je locaties kunt bekijken, of groepen kunt beheren. Je wordt automatisch ingelogd als je jezelf registreerd, of als je eerder hebt ingelogd, en niet handmatig hebt uitgelogd. Hierna kom je gelijk op de kaart met de locatie van jou, en je groepsgenoten. Elke groep krijgt een willekeurige kleur toegewezen tijdens

het laden van de kaart, hierbij kan het voorkomen dat twee groepen dezelfde kleur hebben. Verder kun je vanaf hier naar groepen beheren. Hier kun je een groep aanmaken, zoeken, bekijken of verlaten, of lid worden van een bestaande groep. Op het moment dat je een groep aanmaakt vraagt hij een naam en een maximum aantal mensen die in de groep mogen. Bij het lid worden van een groep ga je ermee akkoord dat je groepsgenoten je locatie kunnen zien, maar dan kun je ook hun locaties zien.

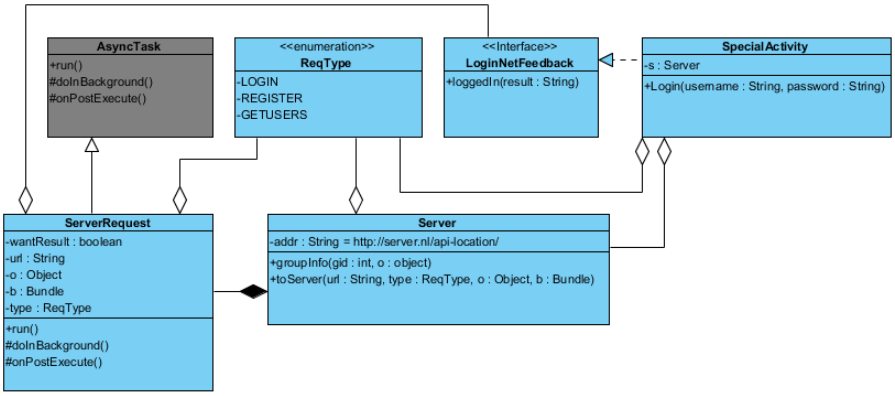
- **Detailontwerp**
  Een globaal overzicht van de activiteiten wordt hieronder in (ironisch genoeg) een activity-diagram laten zien:




  – Bij onze app is de locatie bepaling erg belangrijk, net zoals het contact met de server.




Hier zie je het schema van de locatie bepaling. De grijze blokken zijn al gegeven door Android, de blauwe zijn door ons gemaakt.




En hier het schema voor de feedback van het netwerk. Ook hier geldt dat de grijze blokken door Android zijn gemaakt, en de blauwe door ons.

- **Ontwerpverantwoording**
  **HTTP-Request**

  Een http-request is een typisch voorbeeld in android dat niet in de main thread kan. Als je dit toch doet (wat wij uiteraard geprobeerd hebben) dan loopt de GUI vast en krijg je een “http request in main thread” error. Je kunt dan zelf threads aanmaken en starten, een threadpool aanmaken, et cetera. Wij hebben er echter voor gekozen om het te doen met een AsyncTask object. Dit is een android klasse, typisch voor gebruik voor opdrachten die enkele seconden maximaal

duren.
Omdat wij maar één AsyncTask nodig hebben voor de app, hebben we de abstracte klasse geëxtend en hier een concrete klasse van gemaakt; ServerRequest.
Bij het maken van een ServerRequest geef je een url mee. Dit is de url waarnaar de (GET-)request gestuurd wordt. Ook kun je optioneel een object meegeven, een request type meegeven en een Bundle met extra informatie meegeven.
Bij elke ReqType (Request Type) hoort een interface. Als een object en een ReqType wordt doorgegeven, is dit genoeg om feedback te geven naar het object dat de request heeft aangeroepen. Hiervoor moet echter het object te casten zijn naar de interface die hoort bij de Request Type. Toch gebruiken we hier voor de leesbaarheid van de code zo veel mogelijk RequestTypes in plaats van directe interfaces.
Als alles goed gaat, wordt het resultaat verwerkt en daarna via de interface teruggestuurd.

### Objecten uit de app

Objecten uit de app weten allemaal niets van de ServerRequest af. Deze sturen simpelweg iets naar een Server-object. Dit Server-object (in de code vaak s) heeft verschillende functies, als een login, register, et cetera.
Mocht het object dat de request aanvraagt feedback willen, dan moet hij zichzelf als parameter meegeven. Dit gebeurt bijvoorbeeld bij het inloggen (je moet weten of de credentials juist zijn of niet). Dit hoeft echter niet bij het opslaan van een nieuwe locatie; En locatie missen in 7 minuten is geen ramp, en blijven versturen is dan vaak toch geen oplossing.
Alle objecten die een request sturen en feedback willen sturen dus een object mee. Dit object moet de InternetActivity extenden. De InternetActivity is een activity die één belangrijke functie bevat; isOnline(). Hiermee check je de verbinding van de telefoon. Als de persoon géén verbinding heeft, hoeft er ook geen request gestuurd te worden. Het meegegeven object casten we naar een InternetActivity.
isOnline() maakt gebruik van een context, maar niet alle objecten die requests moeten kunnen doen bevatten een context. Deze objecten overriden de methode met een return true;.Dit was geen ideale oplossing, maar wel de beste die er was. Eventueel kunnen we nog een interface “ConnectionChecker” maken met die methode, die InternetActivity implementeert. Dit was echter naar ons idee niet nodig, omdat er op het moment maar één klasse is die op deze manier geöverride moest worden.

### Totaalplaatje

Dus, zeg de activity “LoginActivity” wil een gebruiker inloggen. Deze stuurt de gegevens (in dit geval username en password) via de server door s.login(username, password, this); . Let hier op de this, dit is voor de terugkoppeling! De server zet de gegevens om in een geldige url voor de GET-request en zoekt hierbij de te gebruiken RequestType. Daarna checkt hij eerst (d.m.v. de cast naar Internet Activity) of er een internetverbinding beschikbaar is. Is er verbinding, dan maakt hij een nieuwe ServerRequest aan waar hij de url meegeeft, het object meegeeft en de RequestType meegeeft. Hierna start hij de ServerRequest. De ServerRequest slaat alle gegevens die binnenkomen op. Als het RequestType een type is dat feedback wil hebben, dan zet deze een boolean op true (anders op false). Hierna doet deze een http request. Als alle informatie binnen is, triggert dat een methode in de ServerRequest. Deze kijkt dan of er feedback nodig is (door middel van de boolean hierboven genoemd). Zo ja, dan cast hij het meegekregen object naar de interface die correspondeert bij de RequestType (in ons geval LoginNetFeedback). Hier triggert hij dan een methode met het resultaat van de HTTP-request als parameter. Zo ontvangt de main thread dan de gevraagde informatie en is het cirkeltje rond!

**LocationService**
Voor de bruikbaarheid van onze app, moesten we een service hebben die regelmatig de locatie van de gebruiker opvroeg en naar de server stuurde, ook als de app zich niet op de voorgrond bevindt. Hiervoor zijn Android Services gemaakt. Deze blijven op de achtergrond draaien.

### Integreren van de code tussen de android code

In bovenstaande class-diagram zie je grijze klassen en blauwe klassen. De blauwe zijn zelf geschreven/gemaakt, terwijl de grijze klassen puur vanuit android komen.
Om een locatie op te vragen en te verwerken, zijn een aantal stappen nodig:
Een locatie wordt opgevraagd door een LocationManager. Deze LocationManager krijgt een LocationListener mee bij het ‘maken’ (je maakt overigens geen nieuwe LocationManager door middel van “New LocationManager”, maar door er een op te vragen). LocationListener is echter een interface en moet gemplementeerd worden. Deze implementatie van LocationListener is dan de klasse die de nieuwe locatie moet verwerken. We wilden deze alleen in de hoofdklasse doen wegens leesbaarheid van de code en communicatie met de server. Om dit te doen, moet dus informatie teruggaan van de LocListener (de implementatie van LocationListener) naar de TrackLocation (de extensie van de Service). Hiervoor hebben we hetzelfde ontwerp gebruikt als

hierboven, namelijk de Service zichzelf laten meegeven als interface die hij implementeert. Deze nieuwe interface (UpdateLocation) bevat enkel methodes om de nieuwe locaties door te geven aan de TrackLocation.

# 4 Reflectie

- In onze groep was in het begin nogal een probleem met de groepsindeling, dit hebben we zo goed als mogelijk opgelost, door mensen op hun eigen tempo te laten werken, en te laten werken op de plaatsen waar ze beter in zijn, zodat zo min mogelijk tijd verloren gaat in het vast lopen of aanleren van onderdelen. Deze fase ging aardig goed, natuurlijk soms met wat hobbels en stoten, maar niet alles is *"A yellow brick road"*. Wat een positief punt was, was dat we niet altijd samen moesten zijn om iets te kunnen. Aangezien we twee mensen hebben die een dubbele bachelor doen, hadden we weinig tijd om samen ergens te gaan zitten, en te werken aan het project. Aangezien we niet samen hoefden te zitten, konden we onze eigen tijd vinden om aan het project te werken. Wat hierbij meekomt is dat je niet weet hoeveel tijd mensen besteden aan het werk dat ze hebben gedaan, wat soms verkeerde producten opleverde, en dat kost meer tijd dan geplanned. Verder vinden we het jammer dat de app niet is geworden wat we hoopten, maar vinden we het wel leuk dat we iets hebben wat we verder kunnen uitbreiden.