

# Stay connected!

## Het bouwen van een gebruikersprofiel op basis van locatiegegevens

---

*Door Hidde Hensel, Freddy de Greef en Diederik Beker*



## **Inleiding**

Voor het vak netcentric computing werd ons gevraagd een android applicatie te maken met netcentric onderdelen. Uiteindelijk is besloten een applicatie te maken met als doel locatiegegevens te gebruiken om het gedrag van een gebruiker vast te leggen. Dit gebruikersprofiel kan door vele toepassingen gebruikt worden, wij hebben gekozen om hier een dating app omheen te bouwen.

Bij deze dating app is het de bedoeling te kijken naar het echte gedrag van een gebruiker, niet een schets die iemand van zichzelf maakt. Op de achtergrond houdt de app meta-informatie bij over de gebruiker in kwestie. Er wordt bijgehouden waar de persoon is en waar deze naar toe gaat, hoelang die op deze locatie is, en er wordt door middel van geofencing bepaald wat er op die locatie gedaan wordt. Bovendien wordt er door middel van peer-to-peer communicatie tussen de verschillende, in de buurt van elkaar te bevinden, telefoons bijgehouden met wie deze activiteit wordt uitgevoerd. Het belpatroon en app gebruik kan worden bijgehouden en opgeslagen.

Neem al deze informatie bij elkaar over een bepaalde tijd en er ontstaat een vrij duidelijk patroon van wat een persoon doet, welke interesses hij/zij heeft en waar en wanneer deze persoon zich ergens bevindt. Door middel van vergelijking van dit patroon met het patroon van andere personen kan een match worden gevonden op basis van feitelijke informatie i.p.v. een zelf ingevuld profiel, en kan een grote 'match succes rate' worden behaald.

Als een match is gevonden kan er door middel van eerder verkregen informatie een plaats en tijd worden bepaald (voorspeld) waar deze personen kunnen ontmoeten. Zonder verdere informatie krijgen zij de tijd en plaats van de ontmoeting. De eerste ontmoeting zal dus de eerste keer zijn dat deze personen informatie over elkaar krijgen. Als het goed is hebben ze veel te bespreken aangezien ze dezelfde interesses, gedragspatronen, koffiedrinktijden, locaties, etc. delen.

Er wordt bij deze verschillende data opgeslagen. Deze data bestaat onder andere uit: locaties, dit wordt opgehaald m.b.v. Google Location Services, activiteiten en contact met andere personen. Dit wordt centraal bijgehouden in een database op de telefoon. Door middel van bluetooth kan er gecommuniceerd worden met andere telefoons en door middel van wifi en mobiel netwerk kunnen er gegevens worden gestuurd van en naar de server. Op de server wordt een patroon gemaakt van de data en dit wordt vergeleken met het patroon van andere gebruikers.

Het systeem kan worden versneld, voor de gebruiker, door rekening te houden met de waarschijnlijkheid van een ontmoeting tussen mensen. Als er een patroon wordt gevonden waarin een bepaald persoon vaak een ander bepaald persoon tegenkomt, kan de ene persoon data van de server meenemen voor de andere persoon. Op deze manier blijft iedereen (via-via) up-to-date en zijn de updates sneller. Met genoeg mensen die gebruik maken van de applicatie zijn deze updates zelfs bijna real-time en werkt het kosten besparend, doordat minder mensen met het netwerk verbinding hoeven te maken, en meer communicatie over het gratis bluetooth plaatsvindt.

## **Probleemstelling**

Het uitgangspunt is dus een applicatie die, op de achtergrond, informatie verzamelt over het gedrag van een persoon en deze, door middel van patroon herkenning, koppelt aan een ander persoon. Er wordt een ontmoeting gepland door middel van een voorspelling van tijd en plaats. Het enige wat de gebruiker hoeft te doen om een match te vinden is zijn of haar telefoon bij zich houden (wat toch al gebeurt). De vraag is echter of er, met behulp van locatiegegevens, een realistisch en nauwkeurig gebruikersprofiel kan worden aangemaakt. De onderzoeksvraag is dus: Kan er een realistisch en betrouwbaar patroon van het gedrag van een gebruiker ontwikkeld worden door middel van locatiegegevens?

## Implementatie

### SQLite Database (Diederik Beker)

De manier waarop data wordt opgeslagen en verzonden is natuurlijk erg belangrijk. Ten eerste moet er genoeg data worden opgeslagen om een reeel en logisch patroon te kunnen creëren. Ten tweede moet de data makkelijk uit te lezen zijn op zowel de server als het android toestel. Ten derde moet het niet te veel ruimte innemen. En als laatste moet het makkelijk kunnen worden verzonden tussen de server en het android toestel, en tussen twee android toestellen. Een eventuele toevoeging kan zijn: encryptie. De data kan gevoelige informatie bevatten, zoals locatie data, die voor andere personen moet worden afgeschermd.

Het lijkt ons het handigst om de data in een *string* te verzenden. Een string is makkelijk op te slaan in de SQLite database van een android toestel, het is namelijk makkelijk te verzenden via bluetooth en via wifi van en naar de server. We kunnen een string zo indelen dat er in één string alle data verpakt en verzonden kan worden door middel van scheidingstekens.

#### Voorbeeld:

We willen graag de volgende data verzenden:

Location: Long:53.2352349, Lat:4.23482092

Address: Sciencepark 500, Amsterdam, The Netherlands

App usage: phone:54min20secs, WhatsApp:21mins08secs, SnapChat:5mins12secs

We kunnen deze data in één string zetten met het scheidingsteken ';'.

“Location: Long:53.2352349, Lat:4.23482092;Address: Sciencepark 500, Amsterdam, The Netherlands;App usage: phone:54min20secs, WhatsApp:21mins08secs, SnapChat:5mins12secs”

Deze string kunnen we vervolgens op een ander toestel of op de server weer logisch opdelen in de individuele strings 'Location', 'Address' en 'App usage'.

### MySQL Database (Hidde Hensel)

In de MySQL database op de server zitten vier tabellen. De eerste tabel, testdatabase (bijlage 1), bevat alle informatie die van de telefoon naar de server is gestuurd. Er zijn vier kolommen: User, location, inputid en sex. Er wordt periodiek data van de app naar de server gestuurd, daardoor kunnen er meerdere rijen per gebruiker voorkomen. De location koom bevat nu nog een string met locatie informatie. Om met deze locatie informatie verder te werken wordt de data in de testdatabase verder verwerkt en opgeslagen in tabel twee, sortedDatabase (bijlage 2). In deze database staat de, voor locatie informatie, zeer belangrijke Latitude en Longitude. Met deze coördinaten kunnen namelijk steden worden opgevraagd en zo kan tabel 3 (bijlage 3) worden gemaakt. In deze tabel staat, per gebruiker, welke steden deze hebben bezocht (cities) en in hoeveel verschillende steden deze zijn geweest (activity). Per gebruiker is er dus één rij met gegevens. De gebruikers kunnen dan vergeleken worden en, indien er een match is gevonden, opgeslagen worden in tabel 4 (bijlage 4).

### User accounts (Diederik Beker)

Het is belangrijk dat de server weet welke data bij wie hoort. Iedere keer als er data wordt verzonden moet die data bij de juiste gebruiker worden opgeslagen om het juiste patroon te creëren. Binnen android is er een AccountManager beschikbaar. Deze kan de gebruiker van de telefoon achterhalen door middel van verschillende accounts die door de gebruiker op dat toestel worden gebruikt. De meest gebruikelijke is het google account, hiermee wordt standaard ingelogd op een

android toestel. Door het google account op te vragen en mee te sturen naar de database kan je de gebruiker identificeren. Het bijkomend voordeel hiervan is dat het niet toestel afhankelijk, maar gebruiker afhankelijk is. Als een gebruiker een ander toestel krijgt kan de data nog steeds bij zijn persoonlijke data worden gevoegd door middel van herkenning van het google account.

### **Services (Diederik Beker)**

Om deze applicatie goed te laten werken, moet deze continu informatie verzamelen. Om dit mogelijk te maken moet de applicatie constant runnen. Als je de applicatie constant laat runnen op de I/O-thread zou je de telefoon niet meer kunnen gebruiken voor andere dingen, dit willen wij vermijden. Om deze reden zal onze applicatie geïmplementeerd worden als een service. Een service is een applicatie die runt in de achtergrond op een andere thread dan de I/O-thread. Op deze manier kan er continu data verzameld worden zonder dat de gebruiker er last van heeft. Een probleem zou kunnen ontstaan als de telefoon wordt gerestart. De applicatie moet met de hand dan weer worden opgestart om informatie te verzamelen. Hiervoor hebben we een *BroadcastReceiver* geïmplementeerd die luistert naar *BOOT\_COMPLETE*. Omdat android de mogelijkheid heeft om de applicatie te laten stoppen als hij niet genoeg geheugen heeft, hebben we het *onStartCommand* "START\_STICKY" laten returnen. Bij *onCreate* zal de applicatie restarten zo gauw er weer geheugen vrij is.

### **Data visualisatie (Hidde Hensel)**

Data visualisatie wordt mede mogelijk gemaakt door Google Fusion Tables API[1]. Met deze API kan er zeer gemakkelijk de activiteit van de gebruikers in kaart worden gebracht. De locatiedatabase, de database waar alle locaties van alle gebruikers in opgeslagen zitten, wordt hiervoor gebruikt. Om de data in google fusion tables te zetten moet de database eerst geëxporteerd worden als een .csv. Door deze .csv file op Google Drive te zetten kan het zo worden ingeladen in google fusion tables. Daarna kunnen er automatisch van de Latitude en de Longitude kolommen locaties worden aangemaakt in google maps. Afhankelijk van de gebruiker kunnen deze punten verschillende kleuren krijgen, een voorbeeld is te zien in afbeelding 1.

### **Android-Server connectie (Hidde Hensel)**

Een Android applicatie kan niet rechtstreeks connectie maken met een database op een server. De app kan echter wel via een bestand op de server een connectie met de database maken, hiervoor gebruiken wij een bestand met een php script. Het belangrijkste van deze connectie is om de entries in de SQLite database naar de MySQL database te sturen. Dit is gerealiseerd met een HTTP POST van de android app.[2] Dit stuurt een string naar het php script, de benodigde data wordt uit deze string gehaald en in de MySQL database op de server gezet.

Deze methode wordt ook gebruikt bij het vinden van een match. De gebruiker kan, nu nog via een button, kijken of er een match is gevonden. De applicatie stuurt dan, *onClick*, de gebruikersnaam naar een php script op de server, dit script kijkt met een query of de gebruiker in de matchtabel voorkomt. Indien dit zo is stuurt het php script een string terug naar de applicatie van de gebruiker met de match.

### **Data sorteren op de server(Hidde Hensel)**

Om de data die van de android applicatie naar de mysql database tabel is gezonden te sorteren, is gebruik gemaakt van een java programma. Dit java programma, nu nog op onze laptops, kan in tegenstelling tot een android applicatie wel rechtstreeks connectie maken met de database op de server. Uiteindelijk is het de bedoeling om dit java programma periodiek op de server uit te voeren.

Het java programma haalt alle data op uit de mysql tabel met de data van de telefoon. Deze tabel heeft twee belangrijke kolommen, gebruiker en locatie. De gesorteerde database heeft meerdere kolommen voor locatie, het is dus nodig om de string van de locatiekolom te uit elkaar te halen. Het uit elkaar halen van deze string wordt gedaan m.b.v. .split. Uiteindelijk wordt de tijd, latitude, longitude en adres uit de locatiestring gehaald en opgeslagen in de nieuwe, gesorteerde, database tabel.

### **Matchen van gebruikers (Hidde Hensel)**

Voor het matchen van gebruikers is het nodig om van alle gesorteerde data van een gebruiker een gebruikersprofiel aan te maken. Dit hebben wij gerealiseerd door een aparte tabel(bijlage 3) te maken op de serverdatabase voor alle gebruikersprofielen, waarbij er één rij is per gebruiker. Gebruik makend van google api's, door middel van de longitude en latitude van de gesorteerde tabel, is het mogelijk om locatie informatie op te zoeken. Hier kan onder andere de stadsnaam makkelijk worden uitgehaald en zo opgeslagen worden in een andere databasetabel. In deze profielentabel worden gebruikers opgeslagen en wordt langzamerhand hun profiel gebouwd. Op dit ogenblik wordt er vooral gebruik gemaakt van bezochte steden. Er wordt per gebruiker opgezocht welke steden zijn bezocht en alle bezochte steden worden in één lijst opgeslagen. Achter de steden wordt opgeslagen hoe vaak, door middel van een counter, de gebruiker in deze steden is verbleven. Aan de hand van de aantal bezochte steden wordt er uitgerekend wat de activity van een persoon is (in principe dus hoeveel die persoon reist).

Het is zeer handig om elke dag minstens één keer een bespreking te houden. Zo kan er snel worden gekeken wat er gedaan is, wat er nog gedaan moet worden en of een idee of methode niet werkt en anders moet worden geïmplementeerd. Dit is door ons eigenlijk pas in de laatste week gedaan en daardoor hebben we belangrijke beslissingen, bijvoorbeeld over de Bluetooth-verbinding implementatie, te laat gemaakt. Het is ook niet handig om vooraf een vaste planning te maken, maar deze planning elke dag bij te stellen.

### **Bluetooth (Freddy de Greef)**

#### **Gebruik**

De app gebruikt de bluetooth momenteel alleen voor het vinden van de mac adressen van de apparaten in de buurt. Zo kunnen wij als ontwikkelaars van de app zien welke app gebruikers met elkaar omgaan. Zo kunnen we bijvoorbeeld groepen maken en kijken of de blind date daadwerkelijk heeft plaats gevonden.

De gevonden apparaten worden in een string list gezet en deze moet worden verstuurd naar de database. [3]

#### **Thread**

Om de bluetooth functie bij de app toe te voegen hebben we gekozen voor een aparte thread die de bluetoothactivity class zal aanroepen. Er is ook geprobeerd de bluetoothactivity aan te roepen met een asynchrone main thread. Dit werkte niet omdat de andere activiteiten al veel gebruik maken van de mainthread.

#### **De gebruiker**

De bluetooth wordt aanzet als de user op het knopje bluetooth drukt. Dit gebeurt zonder te vragen omdat de gebruiker van bluetooth gebruik wil maken en het niet nodig zal zijn om te vragen of hij bluetooth wil inschakelen.

Er wordt echter wel gevraagd of je zelf waarneembaar wilt worden op bluetooth. Dit door privacy overwegingen.

## Peer-to-peer (Freddy de Greef)

Zoals de bluetooth service nu is geïmplementeerd is er geen echte connectie tussen twee apparaten. Er wordt alleen gekeken wie er allemaal in de buurt is. Het oorspronkelijke idee was om een bluetooth peer-to-peer netwerk op te zetten. Waarbij bij de apparaten die in de buurt waren gekeken wordt of zij dezelfde app gebruiken. Als dit het geval is wordt er data uitgewisseld met behulp van een server client model. Dit model maakt gebruik van een server en client socket die in aparte threads worden gerund. De server socket zal luisteren met een bepaald UUID die gehardcode is in de app. De client socket zal kijken of hij een server socket kan vinden, hiervoor heb je het MAC adres nodig van de server socket en het UUID. Het MAC adres wordt verkregen door de lijst van gevonden apparaten in de buurt door te lopen en een voor een met de server sockets proberen te connecten. Als er een connectie is tussen twee devices wordt de server socket gesloten en zal stoppen met luisteren.

Bij het zoeken naar de server client connectie is er in de broadcast receiver de functie `discovery_finished` en `action_found` gebruikt. Bij het beëindigen van de discovery wordt de lijst met apparaten doorlopen en gekeken of er een connectie kan worden gemaakt. Is dit niet het geval dan wordt de lijst met apparaten gewist en de discovery weer aangezet. Bij het vinden van het apparaat wordt deze in de string list van apparaten gezet.

De file transfer zal nu starten met een input en outputstream. De reden dat we voor dit server client model hebben gekozen is omdat we bij de file transfer goed kunnen bijhouden wat voor data we kunnen sturen. Helaas kwamen er later pas achter dat deze files zo groot zijn dat dit bijna niet te gebruiken was. En hebben we ons plan moeten inperken tot het slechts vinden van de apparaten in de buurt,

Als de benodigde bluetooth activiteiten hebben plaatsgevonden wordt de bluetooth uitgezet en de thread beëindigt.

Met het vorige model zouden ook de sockets nog moeten worden opgeruimd anders kan er slechts een keer in een bepaalde tijd connectie gemaakt worden.

## Conclusie

Aan het einde van het project is de applicatie nog niet helemaal afgekomen. We hebben nu één manier om te kijken naar het echte gedrag van een gebruiker. Hierin gebruiken we nu alleen nog steden, dit zouden we kunnen uitbreiden met bijvoorbeeld matching op basis van adressen of matching op basis van de mensen waarmee de gebruiker omgaat. Het op de achtergrond houden van onze processen is goed gelukt en onze applicatie kan constant informatie bijhouden en de informatie wordt in gewenst formaat opgeslagen en gebruikt bij het maken van het profiel. De layout van de applicatie is nu nog niet af. Er zijn nu buttons die elke functie van de applicatie uit kunnen voeren, dit omdat het zeer handig is voor debuggen. Deze buttons horen te verdwijnen en de gebruiker zal slechts op “match” hoeven drukken voor resultaat.

De succes rate is nog niet geïmplementeerd. Dit kan door het kijken of de gekoppelde gebruikers bij elkaar zijn geweest door middel van de Bluetooth Mac adressen die een gebruiker heeft opgeslagen. De match functie is wel succesvol geïmplementeerd. Er kan dus in principe al gebruik worden gemaakt van onze applicatie, maar er zijn zeker dingen die verbeterd of veranderd kunnen worden en hier gaan wij, ook na dit project, aan werken.

## Discussie

Het project verliep niet helemaal perfect, wij hadden bijvoorbeeld niet goed nagedacht in welke volgorde wij welke stap van de applicatiebouw gingen zetten. Dit heeft ons, vooral bij het bluetooth-connectie gedeelte, de das omgedaan. Het originele plan was om eerst de connecties werkend te krijgen en daarna verder te werken. Alleen heeft het veel te lang geduurd om de bluetooth socket connectie goed werkend te krijgen om het vervolgens niet te gebruiken. De les hierin zou zijn om stapsgewijs steeds te kijken wat het volgende nodige is om te implementeren en niet al te veel vooruit te werken in een nog hypothetische richting. Omdat je niet zeker weet of je geleverde product nog in het plan zal vallen. Ook was de nodige communicatie over het niet gebruiken van de sockets in de Bluetoothservice er pas een dag later dat betekent een volle werkdag verspilde tijd voor het project.

Verder zijn we er achter gekomen dat het zeer goed mogelijk is om uit de locatiegegevens van een gebruiker een realistisch profiel te creëren. We hebben echter niet intensief kunnen testen met echte gebruikers, daarom hebben we zelf gebruikers toegevoegd aan de database. Wij zouden dit natuurlijk graag nog testen en kijken waar nog verfijning in mogelijk is.

Al met al zou er tijd bespaard kunnen worden door steeds te kijken hoever je met bepaalde delen van het project bent en wat de volgende logische stap is.

## Log

### Maandag 10-Juni:

- Bespreking project → wat gaan we doen?
- Conclusie: Stay Connected, Diederik verantwoordelijk voor git
- Taakverdeling (zie Planning hierboven)
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie: Iets meer tijd nodig voor alles, verslag voor groot deel opgesteld

### Dinsdag 11-Juni:

- Hidde is ziek
- Wat gaan we doen vandaag?
- Conclusie: Verder volgens planning
- Hidde werkt wanneer die kan, haalt in het weekend in wanneer beter
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie: Location gevonden, morgen verder met database

### Woensdag 12-Juni:

- Hidde is ziek
- Wat gaan we doen vandaag?
- Conclusie: Verder met taken
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie: Bluetooth is stuk lastiger dan gedacht, database opslaan gelukt, maar morgen maken met Service zodat die concurrent kan, brief naar Technology Transfer Office gestuurd

### Vrijdag 14-Juni:

- Hidde is nog steeds ziek
- Wat gaan we doen vandaag?
- Conclusie: Bluetooth afmaken, duurt nog wel even, anders in het weekend afmaken.

Database schrijven en ophalen concurrent maken d.m.v. Services

- Eind van de dag → wie heeft wat afgekregen?
- Conclusie: Verslag geschreven en concurrency van database schrijven en lezen afgemaakt. Bluetooth is nog steeds lastig, maar voortgang, dit weekend afmaken.

### Maandag 17-Juni:

- Hidde weer beter!
- Wat gaan we doen vandaag?
- Conclusie: Hidde gaat hard aan het werk met de server-android connectie, Diederik gaat werken aan data verzending (Wat? Hoe?), Freddy gaat weer verder met bluetooth want dat was nog niet gelukt.
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie: Verslag uitgebreid en data verzending plan afgerond. Adres wordt opgezocht en ook opgeslagen op toestel. Data verpakking klaar op android toestel, klaar om te verzenden. Bluetooth is nog altijd lastig, wordt hard aan gewerkt, nu ook in communicatie met andere groep. Server naar android connectie nog niet af, vanavond doorwerken.



### Dinsdag 18-Juni:

- Wat gaan we doen vandaag?
- Conclusie: Hidde gaat verder met de server, Freddy gaat verder met bluetooth en Diederik gaat een sendActivity maken en een AccountManager inschakelen zodat de user herkend kan worden.
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie: Verslag uitgebreid en data verpakking en verzending klaargezet voor gebruik. (Activity gemaakt die kan verzenden, alleen de connectie moet nog). Bluetooth bezig geweest, maar is nog steeds lastig, morgen afspraak met Ilja om ons te helpen. Server-android connectie bezig, maar lopen tegen enkele problemen aan. Uiteindelijk in de avond afgekregen AccountManager in gebruik genomen voor het herkennen van gebruikers.

### Woensdag 19-Juni:

- Wat gaan we doen vandaag?
  - Hidde: Server-android connectie app afmaken en mergen met hoofdapp.
  - Diederik: Helpen met server-android connectie, sorteren van data op server
  - Freddy: Bluetooth verbinding, zelf nog sockets werkend proberen te krijgen
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie:
- De server-android werkt nu in de hoofdapp, visualisatie van de locatiegegevens is uitgezocht en kan met Google fusion tables API, Bluetooth verbinding is nog steeds niet werkend.

### Vrijdag 21-Juni:

- Wat gaan we doen vandaag?
  - Hidde: Sorteren van data op server afmaken, visualisatie van data bekijken, logo maken, kijken naar het sorteren van data.
  - Diederik: afwezig
  - Freddy: Bluetooth verbinding
- Wie heeft wat afgekregen?
  - Hidde: Logo gemaakt, tutorial gedaan voor fusion tables, csv van database met longitude en latitude → automatisch een map voor visualisatie.
  - Diederik: afwezig
  - Freddy: Werkende code van sockets gebruikt, en gezorgd dat het met onze code werkt

### Maandag 24-Juni:

- Wat gaan we doen vandaag?
  - Hidde: Visualisatie van data, bedenken hoe we gaan vergelijken van users
  - Diederik: vergelijken van users
  - Freddy: Bluetooth verbinding, zorgen dat de verbinding vaker dan eens wordt gezocht totdat er connectie is
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie:
  - Hidde: De database ingeladen in google fusion tables. Het sorteren van de database beter gemaakt (verwijdert nu gebruikte data en geeft een kortere string mee). App kijkt nu of locatiegegevens verzamelen aan staat, zo niet dan gaat de app naar de instellingen van de telefoon. Gekeken naar of het google fusion tables inladen automatisch kan, maar is te veel werk voor nu. Gewerkt aan het vergelijken van gebruikers, uiteindelijk niets concreets uit gekomen.
  - Diederik: app aangepast zodat er meer informatie van de android naar de server wordt gestuurd. Java file gemaakt die de data van de database tabel sorteerd en verwerkt en in een andere tabel stopt.
  - Freddy: Werkende code in onze code krijgen. Morgen een loop in de discovery bouwen omdat hij vaak niet de eerste keer connectie maakt tussen twee devices.

### Dinsdag 25-Juni:

- Wat gaan we doen vandaag?
  - Hidde: Beetje aan vergelijken van users, poster maken
  - Diederik: Vergelijken van users
  - Freddy: Bluetooth verbinding
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie:
  - Hidde: Poster gemaakt, nieuwe database aangemaakt voor vergelijken user.
  - Diederik: Locatie informatie kan nu snel worden opgezocht door gebruik te maken van google api's, vooral stadsnaam wordt nu opgezocht. Begonnen met het creëren van de profielen database. Begonnen met het opslaan van steden informatie in profielen. Er wordt opgezocht welke steden zijn bezocht, alle bezochte steden worden in één lijst opgeslagen. Achter de steden wordt opgeslagen hoeveel procent van de totale tijd in deze steden is verbleven.
  - Freddy: De connectie tussen server en client kan nu gemaakt worden. Alleen nadat er connectie is gemaakt lukt het niet meer om dit te herhalen. Het is nu gelukt om connectie te maken en goed op te ruimen.

### Woensdag 26-Juni:

- Wat gaan we doen vandaag?
  - Hidde: Poster afmaken, GUI van de app, gender button maken
  - Diederik: Userdatabase maken met profiel per gebruiker,
  - Freddy: Bluetooth verbinding
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie:
  - Hidde: Poster afgemaakt en verzonden, gender button gemaakt, splashscreen gemaakt.
  - Diederik: De gebruikerstabel van de database wordt nu met de juiste informatie gevuld, het bouwen van een gebruikersprofiel werkt nu vooral met bezochte steden. Er wordt niet meer gekeken naar hoeveel procent van alle tijd de gebruikers in een stad zijn, dit gaat nu d.m.v. een counter.

-Freddy: Helaas zal het niet lukken om peer-to-peer metadata uit te wisselen onder de app gebruikers, omdat deze metadata te groot is. We hebben besloten om via bluetooth slechts Mac adressen uit te wisselen. Hiervoor zijn geen sockets nodig, slechts de bluetooth functies en de broadcast receiver. De sockets worden er weer uitgehaald.

Donderdag 27 Juni:

- Wat gaan we doen vandaag?
  - Hidde: Poster afmaken, Popup als er een match is gevonden.
  - Diederik: Userdatabase afmaken met profiel per gebruiker.
  - Freddy: Bluetooth-verbinding afmaken en implementeren.
- Eind van de dag → wie heeft wat afgekregen?
- Conclusie:
  - Hidde: popup gemaakt indien match, splashscreen mooier gemaakt.
  - Diederik: userdatabase is nu nauwkeuriger. Freddy geholpen met Bluetooth
  - Freddy: De bluetooth functie werkt nu in een aparte thread. Plaatje voor splashscreen gemaakt.

## Bronnen

- [1] <http://fusion-tables-api-samples.googlecode.com/svn/trunk/FusionTablesLayerWizard/src/index.html>
- [2] <http://forum.xda-developers.com/showthread.php?p=42638475>
- [3] <http://developer.android.com/guide/topics/connectivity/bluetooth.html>  
<https://developer.android.com/google/play-services/location.html>  
<https://developers.google.com/maps/documentation/geocoding/>

## Bijlage 1.

► Tabel: testdatabase

Verkennen Structuur SQL Zoeken Invoegen Exporteer Import Handelingen Legen Verwijderen

	Veld	Type	Collatie	Attributen	Null	Standaardwaarde	Extra	Actie
<input type="checkbox"/>	user	text	latin1_swedish_ci		Nee			[icon] [icon] [icon] [icon] [icon] [icon]
<input type="checkbox"/>	location	text	latin1_swedish_ci		Nee			[icon] [icon] [icon] [icon] [icon] [icon]
<input type="checkbox"/>	inputid	int(11)			Nee		auto_increment	[icon] [icon] [icon] [icon] [icon] [icon]
<input type="checkbox"/>	sex	text	latin1_swedish_ci		Nee			[icon] [icon] [icon] [icon] [icon] [icon]

↑ Selecteer alles / Deselecteer alles Met geselecteerd: [icon] [icon] [icon] [icon] [icon] [icon]

Printopmaak Tabel structuur voorstellen

Voeg 1 veld(en) toe ☐ Aan het eind van de tabel ☐ Aan het begin van de tabel ☐ Na user Start

Indexen: ⓘ					Ruimte gebruik		Rij statistiek	
Sleutel naam	Type	Kardinaliteit	Actie	Veld	Type	Gebruik	Opdrachten	Waarde
PRIMARY	PRIMARY	4	[icon] [icon]	inputid	Data	19,924 Bytes	Formatteren	dynamisch
Creëer een index op kolommen 1 Start					Index	2,048 Bytes	Collatie	latin1_swedish_ci
					Overhead	18,136 Bytes	Rijen	4
					Effectief	3,836 Bytes	Lengte van de rij ø	447
					Totaal	21,972 Bytes	Grootte van de rij ø	5,493 Bytes
					[icon] Optimaliseer tabel		Volgende Autoindex	50
							Gecreëerd	25 Jun 2013 om 11:44
							Laatst bijgewerkt	01 Jul 2013 om 18:50

## Bijlage 2.

Tabel: sortedDatabase

Verkennen Structuur SQL Zoeken Invoegen Exporteer Import Handelingen Legen Verwijderen

	Veld	Type	Collatie	Attributen	Null	Standaardwaarde	Extra	Actie
<input type="checkbox"/>	accountName	text	latin1_swedish_ci		Nee			      
<input type="checkbox"/>	time	float			Nee			      
<input type="checkbox"/>	lat	float			Nee			      
<input type="checkbox"/>	lon	float			Nee			      
<input type="checkbox"/>	address	text	latin1_swedish_ci		Nee			      
<input type="checkbox"/>	sex	text	latin1_swedish_ci		Nee			      
<input type="checkbox"/>	inputid	int(11)			Nee		auto_increment	      

Selecteer alles / Deselecteer alles Met geselecteerd: 

Printopmaak Tabel structuur voorstellen





























Voeg 1 veld(en) toe Aan het eind van de tabel Aan het begin van de tabel Na accountName Start








Indexen: 0					Ruimte gebruik		Rij statistiek	
Sleutel naam	Type	Kardinaliteit	Actie	Veld	Type	Gebruik	Opdrachten	Waarde
PRIMARY	PRIMARY	0	 	inputid	Data	13,340 Bytes	Formatteren	dynamisch
Creëer een index op kolommen 1 Start					Index	4,096 Bytes	Collatie	latin1_swedish_ci
					Overhead	13,340 Bytes	Rijen	0
					Effectief	4,096 Bytes	Volgende Autoindex	223
					Totaal	17,436 Bytes	Gecreëerd	26 Jun 2013 om 20:56
							Laatst bijgewerkt	28 Jun 2013 om 14:34

## Bijlage 3.

Tabel: profiles

Verkennen Structuur SQL Zoeken Invoegen Exporteer Import Handelingen Legen Verwijderen

	Veld	Type	Collatie	Attributen	Null	Standaardwaarde	Extra	Actie
<input type="checkbox"/>	accountName	text	latin1_swedish_ci		Nee			      
<input type="checkbox"/>	sex	text	latin1_swedish_ci		Nee			      
<input type="checkbox"/>	activity	int(11)			Nee			      
<input type="checkbox"/>	cities	text	latin1_swedish_ci		Nee			      

Selecteer alles / Deselecteer alles Met geselecteerd: 

Printopmaak Tabel structuur voorstellen

Voeg 1 veld(en) toe Aan het eind van de tabel Aan het begin van de tabel Na accountName Start

Indexen: 0					Ruimte gebruik		Rij statistiek	
Geen index gedefinieerd!					Type	Gebruik	Opdrachten	Waarde
Creëer een index op kolommen 1 Start					Data	540 Bytes	Formatteren	dynamisch
					Index	1,024 Bytes	Collatie	latin1_swedish_ci
					Totaal	1,564 Bytes	Rijen	6
							Lengte van de rij ø	90
							Grootte van de rij ø	261 Bytes
							Gecreëerd	26 Jun 2013 om 21:13
							Laatst bijgewerkt	28 Jun 2013 om 14:34

Bijlage 4.

Tabel: matches

VerkennenStructuurSQLZoekenInvoegenExporteerImportHandelingenLegenVerwijderen

	Veld	Type	Collatie	Attributen	Null	Standaardwaarde	Extra	Actie
<input type="checkbox"/>	accountName	text	latin1_swedish_ci		Nee			
<input type="checkbox"/>	matches	text	latin1_swedish_ci		Nee			
<input type="checkbox"/>	inputid	int(11)			Nee		auto_increment	

Selecteer alles / Deselecteer alles Met geselecteerd:

Printopmaak Tabel structuur voorstellen

Voeg  veld(en) toe Aan het eind van de tabel ☐ Aan het begin van de tabel ☐ Na

Indexen:

Sleutel naam	Type	Kardinaliteit	Actie	Veld
PRIMARY	PRIMARY	8		inputid

Creëer een index op kolommen

Ruimte gebruik

Type	Gebruik
Data	524 Bytes
Index	2,048 Bytes
Totaal	2,572 Bytes

Rij statistiek

Opdrachten	Waarde
Formatteren	dynamisch
Collatie	latin1_swedish_ci
Rijen	8
Lengte van de rij ø	65
Grootte van de rij ø	322 Bytes
Volgende Autoindex	19
Gecreëerd	27 Jun 2013 om 19:55
Laatst bijgewerkt	28 Jun 2013 om 14:34