IMPLEMENTATIE VERSLAG

EEN TU DELFT BACHELORPROJECT

Bachelorproject (IN3405), faculteit Elektrotechniek, Wiskunde en Informatica

Technische Informatica, studiejaar 2011-2012

Opdrachtgever (SSCM B.V.): Dr.Ir. S.I. Suddle

Begeleider (TU Delft): Drs. P.R. van Nieuwenhuizen

Auteurs:

V.J. Koeman (4008790) en E.C. Buckers (4009118)

SAMENVATTING

Via dr.ir. S.I. Suddle, een docent aan de TU Delft, en dr.ir. E.G.J. Blokhuis raakten wij betrokken bij een project genaamd 'CHAIN*els*'. Binnen dit project zijn wij verantwoordelijk voor de gehele ICT, met uitzondering van de (Graphische User) Interface, waar een apart designteam voor verantwoordelijk is. Ons systeem moet dus door hun gemakkelijk te gebruiken zijn.

De snelste manier om CHAIN*els* te beschrijven is als een 'Facebook voor bedrijven'. Dit doet echter te kort aan het concept; CHAIN*els* is veel meer dan dat: het is een professionele netwerktool die bedrijven met andere bedrijven verbindt. Het is dus puur op bedrijven gefocust, en bedoeld om zakelijk contact te kunnen houden met elkaar. Door dit hoge professionaliteitsgehalte is het onderscheidbaar van al bestaande sociale netwerken.

Op een iets dieper niveau begint alles met een Company (bedrijf). Dit is de 'main actor' op het netwerk, die aangestuurd wordt door een Account. Elk account kan bepaalde rechten hebben voor het aansturen van de Company op CHAIN*els*. Dit aansturen kan door publieke berichten op de zogenaamde wall/desk van het bedrijf in de vorm van een nieuwsbericht, een aanbodbericht, of een vraagbericht. Deze berichten kunnen bij andere bedrijven op hun journal verschijnen of bij door ons systeem aangeraden vraag en aanbod. Ook reacties op deze berichten zijn mogelijk, en hiernaast bestaat er nog een systeem om 'privé' berichten tussen bedrijven te kunnen versturen. Bedrijven kunnen ook met elkaar 'chainen', oftewel aangeven dat bedrijven interesse in elkaar hebben. Hiernaast kan er nog aangegeven worden dat een bedrijf een department of holding is van een ander bedrijf, wat een nog sterkere band aangeeft. Verder heeft elk bedrijf een eigen 'profielpagina' met daarop verschillende (statische) informatie over dat bedrijf. Om verder gebruik van het netwerk te vereenvoudigen zijn er logischerwijs allerlei andere gerelateerde functionaliteiten als zoekfuncties en dergelijke nodig. Ook op een lager niveau als serverbeveiliging waren wij verantwoordelijk. U kunt over dit alles in hoofdstuk 1 meer lezen.

Na deze eisen te hebben vastgesteld en uitgebreid onderzoek te hebben gedaan, onder andere naar de bestaande sociale netwerken, hebben wij voor een aantal 'mechanismen' gekozen om het bovenstaande te implementeren (zie Appendix A). Heel kort gezegd betekent dit dat wij met behulp van de programmeertaal PHP en het vrij nieuwe en onconventionele databasesysteem Redis hebben gewerkt, ondersteund door de Scrum ontwikkelingsmethode. We hebben dus het 'standaard' MySQL losgelaten om verschillende redenen, vooral gebaseerd op de drastisch hogere snelheid. Na deze vaststellingen hebben wij een planning uitgewerkt, onderverdeeld in vijf 'milestones' of fases, conform de Scrum ontwikkelmethode. In hoofdstuk 2 kunt u daar meer over lezen, samen met de requirements per fase en dergelijke.

Aan de hand van de gemaakte keuzes zijn we begonnen aan het ontwerpen van het daadwerkelijke systeem. Dit alles is logischerwijs in samenwerking met de rest van het CHAIN*els* team gebeurd, en was sterk onderhevig aan veranderingen in de loop van de tijd.

Uiteindelijk hebben we echter de volgende basisobjecten geïdentificeerd: Account, Company (met Address, About en Contact erbij), BaseMessage, WallMessage, PrivateMessage, Reply, en Result. Deze objecten komen overeen met het bovenstaande. BaseMessage is de generale (abstracte) klasse die functionaliteiten bevat die in alle drie de 'andere soort berichten' voorkomen. Een Result object is nog iets speciaals: dit object bevat naast een simpele boolean (geslaagd of niet geslaagd) ook nog een mogelijke foutmelding. Deze foutmelding is, net zoals de rest van ons systeem, taalafhankelijk. Op dit moment is bijvoorbeeld alles in het Nederlands en het Engels uitgevoerd, maar ons hele systeem is er op gebouwd om dit eenvoudig met meerdere talen uit te kunnen breiden.

Bovendien hebben we ook een uitgebreid (gestandaardiseerd) systeem van Validatie om te zorgen dat data altijd zuiver en uniform is; zo kunnen bijvoorbeeld postcodes nooit in een verkeerd formaat zijn.

Naast de basisobjecten, oftewel de dataklassen, zijn er logischerwijs ook functionele klassen. Zo zijn de systemen voor chainen, department/holdings, generatie van overzichten, notificaties, enzovoorts, in hun eigen klassen verwerkt. Ook is er een uitgebreid databasemodel samengesteld met bijbehorende keuzes voor verschillende datatypen en dergelijke. Redis, een NoSQL systeem, kent namelijk verschillende datatypen met bijbehorende eigenschappen. Bovendien is het geen (traditioneel) relationeel systeem, waar een heleboel gevolgen mee samengaan. Over dit alles kan in hoofdstuk 3 meer lezen.

Na het ontwerp volgt logischerwijs de implementatie. Voor de ontwikkeling hiervan hebben we de eerder genoemde Scrum methode met de fases (milestones) gebruikt, uitgewerkt in Appendix B. Zo zijn we begonnen met het implementeren van de basisobjecten, wat neerkomt op een één op één overzetting met het ontwerp, en daarna verder gegaan met de basis als de validatie en het multi-language systeem. Hierna kwamen de geavanceerdere functionaliteiten behorende bij de verschillende objecten.

Zo heeft een Account object naast simpele 'datavelden' ook te maken met geavanceerdere velden als een tijdzone en een status. Voor onder andere de bepaling van de tijdzone van een gebruiker gebruiken wij een externe PHP module (GeoIP), waarmee met behulp van het IP-adres van de gebruiker zijn geografische gegevens kunnen worden bepaald. Het statusveld geeft verder aan 'hoe compleet' het object is, wat vrijwel op dezelfde manier bij een Company object is geïmplementeerd. Naast de data-opsla zijn er ook aantal speciale acties die met een Account te maken hebben. Zo is er een activatiemail, waarmee het e-mailadres van de gebruiker op echtheid gecontroleerd wordt. Hiervoor wordt een door ons gegenereerde willekeurige code gebruikt die slechts in de mail zelf aan de gebruiker kenbaar wordt gemaakt. Ook de rechten van een account zijn van belang, maar worden meer op het niveau van de User Interface geïmplementeerd.

Een Account heeft als laatst, net als vrijwel elk ander data object, een unieke identificatie nodig. Nu is bij een Account toevallig ook het e-mailadres een identificatie, maar bij de andere objecten bestaat een dergelijk uniek veld niet. Daarom genereren wij 'Identification Numbers (IDs)' aan de hand van zogenoemde 'timestamps'. Een timestamp is simpelweg de verstreken tijd sinds nu en een bepaalde datum, standaard 1 januari 1970: de 'Unix Epoch'. Als we deze tijd maar precies genoeg meten zodat opeenvolgende identificaties altijd uniek zijn is dit voldoende om een object uniek mee te identificeren. Zo is het dan ook bij ons geïmplementeerd, op dit moment met een nauwkeurigheid van 10^-6 (microseconden). Door het bestaan van meerdere servers vormen er echter problemen, die op hun beurt weer op te lossen zijn door het eindcijfer (of de eindreeks) van elke timestamp per server hetzelfde te laten zijn.

Voor een Company geldt vrijwel hetzelfde als voor een Account , maar doordat de Company 's de hoofdelementen van ons netwerk vormen hebben ze een stuk meer datavelden. Onder andere hierdoor hebben wij een aantal opsplitsingen hierin gemaakt, namelijk About, Address, en Contact. Deze samengestelde objecten vormen elk één veld binnen het Company object, en kunnen indien gewenst nog hergebruikt worden in andere klassen. Een andere afwijking is dat bij een Company een logo kan worden geüpload. Deze wordt ook in onze database opgeslagen in een standaardcodering (base64).

Daarna volgde het systeem van chains, departments, en holdings. Deze drie eigenschappen zijn te generaliseren als 'koppelingen', en zijn dan ook vrijwel gelijk in hun werking. Zo bestaat er ook bij alle drie het concept van verzoeken: een bedrijf stuurt bijvoorbeeld een verzoek naar een ander bedrijf om daarmee te chainen.

Dit verzoek kan dan door dat andere bedrijf geaccepteerd of geweigerd worden. Verder kan een bedrijf een ander bedrijf ook blokkeren, iets wat ook als een 'koppeling' kan worden beschouwd, maar hier is logischerwijs geen acceptatie of iets dergelijks voor vereist. Een laatste opmerking hierover is dat een bedrijf slechts één holding kan hebben, en dat alle drie de genoemde koppelingen complementair zijn: er kan er maar één tegelijk bestaan. Ook kan een bedrijf niet een holding worden van haar eigen holding.

Aanvullend op het systeem van chains bestaan er zogenoemde 'Invites'. Dit zijn uitnodigingen naar derden om lid te worden van CHAIN*els* en een chain te worden van het uitnodigende bedrijf. Deze uitnodigingen kunnen met een simpel mailtje worden verstuurd, maar ook via verschillende sociale netwerken (op dit moment Twitter en LinkedIn). Een extra functie hierbij is dat opeenvolgende invites naar dezelfde persoon een additief effect hebben: de uitgenodigde persoon zal te zijn krijgen dat bijvoorbeeld ondertussen al vijf bedrijven hem vragen om in hun netwerk te komen. Ook zorgt dit ervoor dat er niet meerdere uitnodigingen van hetzelfde bedrijf naar dezelfde persoon verstuurd kunnen worden.

De basis wordt afgerond met het zeer belangrijke berichtverkeer. Zoals hiervoor genoemd bestaan er verschillende berichten: WallMessages, Replies, PrivateMessages, en ook InfoMessages. Deze zijn tevens allemaal een BaseMessage, waardin standaardvelden als de afzender, de inhoud, enzovoorts zijn vastgelegd. Een WallMessage is hierin nog speciaal, omdat deze weer in drie soorten voorkomt: Nieuws, Vraag, en Aanbod. Vraag en Aanbod berichten worden in verschillende nog te bespreken algoritmen gebruikt om vraag en aanbod van bedrijven op elkaar af te kunnen stemmen met behulp van zogenoemde labels: steekwoorden die de inhoud van het vraag of aanbod omschrijven. Nog niet eerder genoemd is de InfoMessage: dit is een soort 'secundair' bericht wat een bedrijf niet direct zelf aanmaakt, maar wat wel volgt uit haar acties. Zo wordt er bijvoorbeeld een InfoMessage genereerd als een bedrijf met een ander bedrijf chaint, of een bericht van een ander bedrijf aanraadt.

Na het afronden van deze basis hebben wij zaken geïmplementeerd als het in-en uitloggen van gebruikers, het terugvragen van wachtwoorden, enzovoorts. Hierbij moest grote zorg gedragen worden voor de veiligheid en de privacy van de gebruiker. Zo worden wachtwoorden om verschillende redenen niet met het standaard ('te snelle') MD5 algoritme gecodeerd, maar met het Bcrypt (Blowfish) algoritme. Verder gebruiken we conform standaarden willekeurig gegenereerde sessiesleutels die weer om de vijftien minuten worden geregenereerd, HTTPS verbindingen, beveiligde cookies, enzovoorts.

Een kleine tussenstap is het 'Statistiekenverhaal'. Het is namelijk voor ons, maar ook voor bedrijven zelf, nuttig om statistieken over het gebruik van ons (online) netwerk te verzamelen. Hiervoor wordt de eerder genoemde GeoIP module gebruikt om paginabezoeken bij te houden, gebonden aan de locatie van de bezoeker. Door het bestaan van meerdere webservers en beveiligde verbindingen waren hier enkele problemen mee die opgelost moesten worden, maar uiteindelijk wordt nu per pagina bijgehouden wie deze pagina heeft bezocht en waarvandaan.

Als laatste implementeerden we de zogenaamde 'geavanceerde algoritmen'. Het eerste voorbeeld hiervan is het algemene zoekalgoritme, waarmee bedrijven gevonden moeten kunnen worden op naam, provincie, industrie, enzovoorts. Om dit te realiseren wordt de invoer per woord gescheiden en dan per woord met behulp van een standaardalgoritme (similar tekst) vergeleken met de velden waar op gezocht kan worden. Zo zijn zoekopdrachten als 'Delft ICT' mogelijk.

Daarnaast is er een algoritme voor de aangeraden chains, oftewel suggesties voor bedrijven om met andere bedrijven te verbinden. Hierbij wordt op dit moment gebruik gemaakt van een soort 'vrienden van vrienden systeem'. Alle chains van de bestaande chains van een bedrijf worden met elkaar vergeleken, en gescoord op het aantal voorkomens.

Zo komt een bedrijf waarmee een heleboel van jouw chains gechaint zijn dus hoog in de resultaten terecht, mits er niet al mee gechaint is natuurlijk.

Verder zijn er ook voor de vraag- en aanbodfunctionaliteiten een aantal algoritmes nodig. Deze werken namelijk, zoals eerder gezegd, met labels: korte steekwoorden die een bericht identificeren. Om goed gebruik hiervan te stimuleren bestaat er een algoritme dat bestaande steekwoorden zoekt bij een bepaalde (incomplete) invoer, waarbij het meest gebruikte steekwoord 'bovenaan' komt te staan. Verder worden vraag en aanbod bij CHAIN*els* op elkaar afgestemd; dit door zogenaamde 'vraaggmogelijkheden' en 'aanbodmogelijkheden' voor de gebruiker te genereren. Ook hier is een geavanceerd algoritme voor nodig.

Ook voor het genereren van de homepage (Dashboard/Journal) van een bedrijf is een algoritme nodig. Hierbij worden alle WallMessages van de chains, departementen en mogelijke holding van een bedrijf tot een bepaald tijdstip of een bepaald aantal opgehaald, waarna alle mogelijke InfoMessages tussen het eerste en laatste opgehaalde bericht er tussen worden gezet. Hieruit volgen ook de notificaties: dit zijn eigenlijk ook InfoMessages, maar een bedrijf krijgt een melding wanneer het deze nog niet gezien heeft. Om dit te faciliteren worden alle ongelezen notificaties weergeven; als dit er echter minder dan vijf zijn worden deze aangevuld tot vijf, zodat er ook nog oude notificaties bekeken kunnen worden.

Als laatste wordt binnen CHAIN*els* de bestaande social media niet genegeerd, maar juist gebruikt. Zo was er al de mogelijkheid om uitnodigingen via Twitter en LinkedIn te sturen, maar WallMessages kunnen ook naar deze sociale netwerken 'doorgestuurd' worden. Dit laatste kan ook met Facebook. Hiervoor worden de verschillende zogenoemde 'APIs' van deze sociale netwerken gebruikt, die allemaal op een wel vergelijkbare maar toch net iets andere manier werken. Ze gebruiken namelijk allemaal hetzelfde zogenoemde 'OAuth' protocol. Hierbij kan een applicatie (zoals CHAIN*els*) aan een gebruiker toestemming vragen om berichten namens hem of haar te plaatsen. Dit gebeurd met behulp van een zogenaamde 'secret key' van de gebruiker in combinatie met de secret key van de eigen applicatie. Dit is een waardevolle mogelijkheid, aangezien bestaande netwerken nu benut kunnen worden binnen ons netwerk.

Over alle algoritmes kunt u meer lezen in hoofdstuk 4, waar w ook over bijvoorbeeld op pseudo-code en complexiteit ingaan.

Tijdens het implementeren van al deze functionaliteiten is één ding erg belangrijk geweest: testen. Zo hebben we met zogenoemde 'unit tests', tests die puur op het niveau van de code werken, gegarandeerd dat alles werkt zoals we willen. Dit was extra handig gezien het feit dat onze programmeertaal PHP pas 'on run' wordt gecompileerd, en dus fouten er niet van te voren uit te halen zijn. Ook is het erg handig om bij elke aanpassing in het systeem te kunnen controleren of de rest van het systeem nog steeds goed werkt. Deze unit tests kosten dus wel veel tijd om te schrijven, maar waren zeker heel erg nuttig. Zogenaamde user tests, tests van het systeem met gebruik van een (user) interface, waren voor lange tijd helaas niet mogelijk, aangezien de interface pas zeer laat gereed was. Uiteindelijk hebben we echter wel een aantal user tests opgesteld, waarvan de resultaten in onder andere Appendix F zijn opgenomen. Over deze tests kunt u in hoofdstuk 5 meer informatie vinden.
Het laatste hoofdstuk uit dit verslag, hoofdstuk 6, is een reflectie op ons ontwikkelproces. Zo was er bijvoorbeeld initieel een duidelijke tweedeling tussen business/marketing en ICT, en binnen ICT weer een duidelijke tweedeling tussen ons gedeelte en de user interface. Hierbij is nog op te merken dat in het begin het business/marketing team naast onze begeleider Shahid Suddle ook uit Erik Blokhuis bestond, de schrijver van het originele concept. Hij is echter helaas door persoonlijke omstandigheden uit dit concept gestart; voor hem wordt tot op moment van schrijven nog vervanging gezocht. Door dit vertrek is de tweedeling tussen marketing en ICT echter flink verwaterd, waardoor wij een stuk meer inspraak in het project verkregen.

In principe hebben we altijd met het hele team minstens wekelijks vergaderd over de actiepunten van elke voorgaande week. Beide teams kregen namelijk elke week actiepunten mee. De User Interface was in het begin echter een heikel punt: deze telkens achter op schema. Dit kwam onder andere omdat de user interface in het begin slechts door één persoon (Remi Baar) werd geïmplementeerd. Uit dit oogpunt hebben we uiteindelijk ondersteuning gevonden in de vorm van Willem Buijs, student Industrieel Ontwerpen aan de TU Delft. Bovendien zijn we intensief gebruik van maken van een beheersysteem, 'Flyspray', waarmee individuele taken en deadlines in een duidelijk overzicht gezet werden. Hierdoor werd het voor ons (het ICT-team) een stuk duidelijker wat er wanneer af moest zijn. Ook was het mogelijk om prioriteiten aan taken te geven, er commentaar aan toe te voegen, enzovoorts.

Toen alle problemen eenmaal opgelost waren werden we een steeds hechter team, waarin iedereen heel goed met kritiek kan omgaan, en kwesties altijd bespreekbaar zijn. We zijn allemaal enorm gemotiveerd om dit concept te laten slagen, en kunnen dat ook doen door dat onderhand duidelijk is waar de expertise van iedereen ligt. Een hoogtepunt voor ons was de officiële lancering van ons concept op 11-11-11 in Delft, welke erg succesvol was. Hieraan voorafgaand was er overigens door Shahid Suddle een 'pre-lancering' in New York op 1-11-11, waar we ook erg trots op zijn.

Afsluitend zullen we nog zeker door gaan met dit concept; op het moment van schrijven zijn we al bezig met 'CHAIN*els* 2.0', waarin we alle feedback die we hebben gekregen en zelf hebben vergaard sinds de lancering van ons netwerk zullen verwerken. Wij vinden dit zelf eg uitdagend en ook zeker vermakelijk, en denken er natuurlijk ook wat mee te kunnen verdienen; we hebben namelijk erg veel vertrouwen in dit concept, net als de rest van het team, waardoor onze doelstelling is om in 2012 ten minste tienduizend bedrijven op ons netwerk te hebben.

"One and one makes eleven possible" (Shahid Suddle)

VOORWOORD

Dit implementatieverslag is ter afsluiting van onze Bachelor Technische Informatica aan de Technische Universiteit Delft. Het bachelorproject kost normaal twaalf weken full time, maar in ons geval gaan wij langer met het project door. Dit is omdat wij ons project, CHAIN*els*, een innovatief en uitdagend concept vinden. Om toch de periode af te bakenen voor ons bachelorproject hebben we de periode van 18 juli 2011 tot en met 11 november 2011 gekozen. Dit is een periode van vier maanden, meer dan gewoonlijk, omdat wij vanaf september tevens met onze minor zijn begonnen: we hebben toen minder tijd aan het bachelorproject per week besteedt; dit ging van circa veertig uur per week naar circa dertig uur per week.

Het project in opdracht van Dr. Ir. S. I. Suddle, genaamd "CHAIN*els* – The business network", heeft ons de mogelijkheid geboden om onze ervaring en kennis in praktijk te gebruiken. Het interessante van het project is dat het dicht op sociale media zit, wat op het moment één van de populairste (ICT)toepassingen op het internet is. Daarnaast was het voor ons een kans om met meerdere disciplines samen te werken; tot nu toe hebben we immers in voorgaande projecten voornamelijk met andere Technische Informatica studenten gewerkt.

Zowel Dr. Ir. S. I. Suddle als Drs. P. R. van Nieuwenhuizen, respectievelijk onze begeleider bij CHAIN*els* en onze begeleidende docent namens de Technische Universiteit Delft, willen we bedanken voor de steun en de kans om CHAIN*els* te combineren met onze studie. Daarnaast willen we nog opmerken dat de startdatum van dit project midden in de zomervakantie was, en dat het dankzij Drs. P. R. van Nieuwenhuizen mogelijk was om het toch onder zijn begeleiding te starten; ook hartelijk dank hiervoor.

Wij hopen dat we ons enthousiasme over dit project in ons verslag hebben kunnen weerspiegelen.

INHOUDSOPGAVE

INLEIDING

Dit implementatieverslag gaat over het Bachelorproject genaamd "CHAIN*els* – The business network". Hieronder zullen we kort toelichten wat CHAIN*els* is.

CHAIN*els* is een 'business netwerk' waar vraag en aanbod op elkaar wordt afgestemd. Het bedrijf staat hier centraal, in tegenstelling tot de huidige sociale media waar de persoon centraal staat. Bedrijven kunnen een account aanmaken en zaken doen via het netwerk. Naast de acties die bedrijven zelf kunnen uitvoeren worden er ook automatisch kansen getoond buiten het netwerk van een bedrijf. Het is dus mogelijk om zowel een bedrijfsnetwerk te onderhouden als kansen buiten een dergelijk netwerk te benutten.

Wij waren vanaf het begin al bij dit concept betrokken. Gezien het feit dat het project op nul begon hebben wij gebruikt gemaakt van externe werkplekken, voornamelijk op de Technische Universiteit Delft. De grote lijnen waren uitgestippeld door dr.ir. S.I. Suddle in samenwerking met dr.ir. E.G.J. Blokhuis, maar de daadwerkelijke implementatie van het product is in teamverband tot stand gekomen. Door middel van vergaderingen bleef iedereen in het team op de hoogte van de ontwikkelingen op alle gebieden.

Het CHAIN*els*-team bestaat op dit moment uit de volgende personen:

- Dr. Ir. S. I. Suddle – Business en Marketing
- R. Baar – User Interface
- W. Buijs – Design
- V. J. Koeman – ICT
- E. C. Buckers – ICT

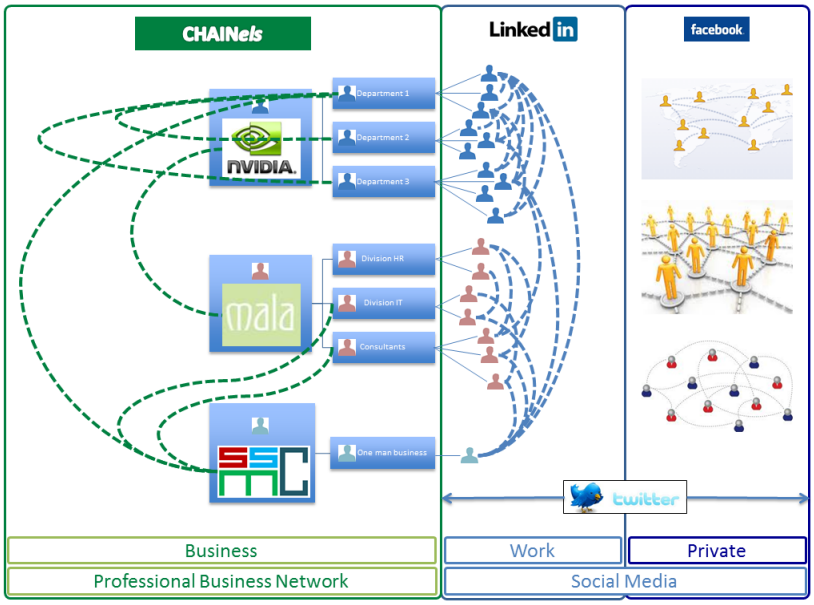
In hoofdstuk 1 is het project verder toegelicht en gedefinieerd. Het is aan te raden om na hoofdstuk 1 eerst Appendix A te lezen, aangezien het onderzoek uitgezet in deze appendix de basis voor de hoofdstukken 2 en 3 is geweest.

1. HET PROJECT

Naar opdracht van dr.ir. S.I. (Shahid) Suddle, assistent professor aan de TU Delft bij Civiele Techniek, en dr.ir. E.G.J. (Erik) Blokhuis, docent en onderzoeker aan de TU Eindhoven bij Bouwkunde, zijn wij betrokken begonnen met het project waar dit verslag betrekking op heeft: CHAIN*els*. Onze verantwoordelijkheid ligt in beginsel bij het ICT-gedeelte van dit project, uitgezonderd de grafische (user)interface. Deze interface wordt op dit moment verzorgd door een Rotterdamse student Economische Informatica in samenwerking met een student Industrieel Ontwerpen aan de TU Delft. De heren Suddle en Blokhuis waren in beginsel de 'producenten' en verzorgden ook de gehele marketing. De heer Blokhuis is echter door persoonlijke redenen uit dit project gestapt; naar vervanging voor hem wordt op dit moment nog gezocht. Naast onze ICT verantwoordelijkheden zijn wij ook onderdeel van het 'oprichtersteam', en beslissen in die hoedanigheid ook mee over allerlei gerelateerde zaken binnen CHAIN*els*.

1.1 INHOUD

Uiterst kort is CHAIN*els* te beschrijven als 'de Facebook voor bedrijven'. In meer detail is het een professionele netwerktool die bedrijven en organisaties met andere bedrijven en organisaties verbindt (chains). CHAIN*els* kan gebruikt worden om zakelijk contact te houden met relevante relaties. Het 'key-concept' is dat het puur op bedrijven gefocust is; het geheel moet dus een hoog professionaliteitsgehalte hebben. Dit is dan ook de onderscheiding tussen CHAIN*els* en al bestaande netwerken zoals Facebook, Twitter, LinkedIn, enzovoorts: het is bedoeld voor pure bussiness-to-business networking. Dit onderscheid is in figuur 1 gevisualiseerd.



Figuur 1 - Een visualisatie van CHAIN*els*

Met dit oogpunt is de naam CHAIN*els* ook te verklaren: het belangrijkste voor een bedrijf is de 'chain' en de 'channels'. Bedrijven kunnen via CHAIN*els* professioneel contact met hun relaties houden en vraag en aanbod met betrekking tot diensten en producten onbeperkt 'uploaden', waardoor ze meer kunnen leren over bedrijven waarmee ze zaken doen.

Onze verantwoordelijkheid binnen dit project is zoals eerder gezegd de 'interne' ICT. Hier valt logischerwijs een heleboel onder binnen een dergelijk groot (internet)project. Het basisidee is om bedrijven een 'pagina' te laten hebben waarop zij informatie over zichzelf kunnen verspreiden, waarbij de prioriteit vooral ligt bij vraag en aanbod. Bedrijven kunnen ook weer pagina's maken voor verschillende afdelingen binnen dat bedrijf, holdings, enzovoorts, waardoor ze hun verschillende 'bezigheden' kunnen scheiden.

Er wordt op dit moment ook aan functionaliteit als een market place, een nieuwscenter, een event kalender, enzovoorts gedacht of al gewerkt.

1.2 VERANTWOORDELIJKHEDEN

Over het algemeen hebben we de (sub)systemen waarvoor wij verantwoordelijk zijn als volgt benoemd en ingedeeld:

- **Website Systeem (front-end)**
  - Bedrijven
  - Bedrijfsafdelingen
  - Persoonlijke accounts
  - Rechtenbeheer
  - 'Wall' systeem (Desk)
  - Homepage-overzicht (Dashboard/Journal)
  - (Direct/Private) Messaging systeem
  - Chain systeem (connecties)
  - Customization (userinstellingen)
  - Overige zaken mbt. een profiel (events, marketplace, enz.)
- **Website Beheer (back-end)**
  - Schaalbaarheid, optimalisatie en controle (failsafe)
  - Server(s)
  - Beveiliging (ook mbt. privacy)
  - Databasebeheer
  - Statistieken
  - (Content) Management
  - Multi-langual support

Hier volgen nog een paar toelichtingen bij de bovenstaande items. De bedrijfs- en bedrijfsafdelingssystemen omvatten de zaken die nodig zijn om een bedrijf(safdeling) te registreren, en dus informatie erover op te slaan, er een pagina voor te creëren, enzovoorts. Daarna zijn er de persoonlijke accounts: een 'bedrijf' kan natuurlijk zelf niets doen. Daarom zijn er personen nodig die alles beheren, waarvan er weer bepaalde informatie moet worden opgevraagd en opgeslagen.

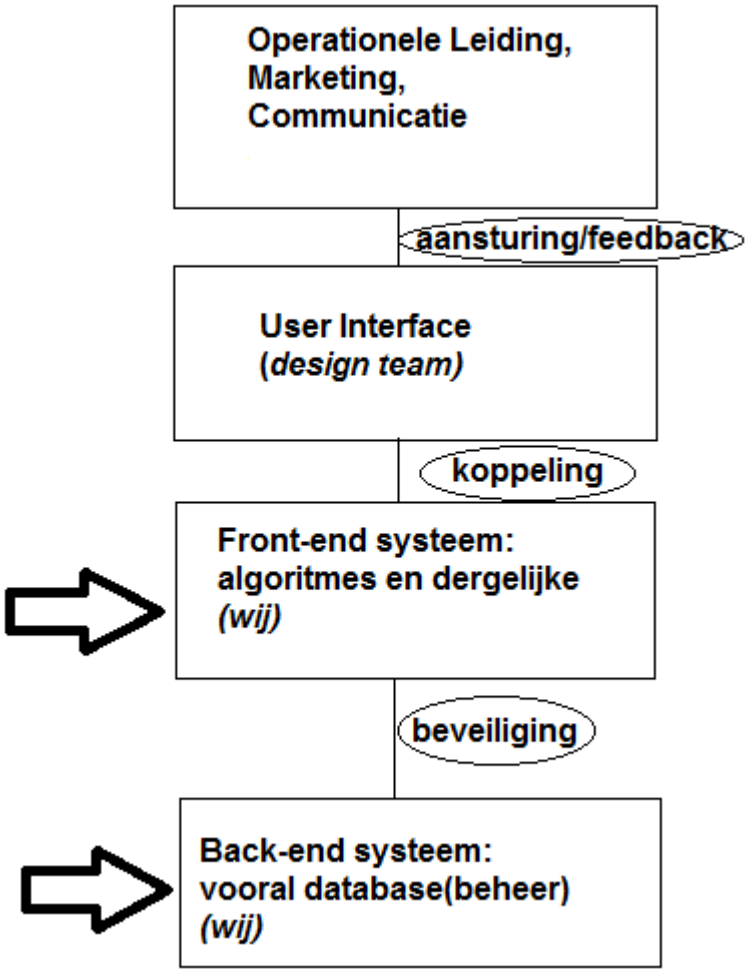
Het rechtenbeheer systeem moet ervoor zorgen dat de juiste personen het juiste kunnen doen binnen een bedrijf(safdeling). Verder heeft elk bedrijf zijn eigen 'wall' (desk) waarop alle vraag, aanbod of nieuws informatie zal verschijnen.

Deze informatie kan dan weer op de homepage (Journal) van een persoon verschijnen: wat voor die persoon relevant is zal hij dan zien. Hier zijn uitgebreide algoritmes voor nodig. Ook is er een systeem nodig waarmee direct berichten naar een bedrijf, bedrijfsafdeling of persoon kunnen worden gestuurd, dus zonder gebruik van een publieke wall. De volgende vereiste is het chain systeem: connecties met andere bedrijven maken, onderhouden, en weer nieuwe connecties vinden. Dit moet in eenvoudige overzichten komen, en suggesties voor nieuwe chains moeten gepast zijn. Verder moeten users ook zelf invloed kunnen hebben op dit alles: vandaar dat er een bepaalde 'customization' van de kant van de user gewenst is.

Op een nog lager niveau moet er ook flink wat werk verzet worden. Zo is er een uitgebreid databasesysteem nodig. Dit systeem moet schaalbaar blijven, net als de rest van de dienst. Er zijn dus veel optimalisaties, controles, enzovoorts nodig.

Ook beveiliging en daarmee privacy is belangrijk, en dient dus op een adequate manier verzorgd te worden, met weer de nodige inspraak van de gebruikers zelf.

In figuur 2 is een beknopt model weergegeven van de structuur binnen CHAIN*els*, om duidelijker te maken waarvoor wij verantwoordelijk zijn, is dit met pijlen in de figuur aangegeven.



Figuur 2 - De verantwoordelijkheden binnen CHAIN*els*

Als laatste willen we nog even het belang van een goed design en een hoogst geoptimaliseerde implementatie toelichten. Dit concept is gemaakt met het oog op miljoenen gebruikers! Een netwerk voor enkele miljoenen gebruikers met daarmee ook miljoenen walls, berichten, overzichten, enzovoorts maak je niet zomaar. Hier moet duidelijk een goed proces voor gebruikt worden, zoals we onder andere in het vak Software Engineering hebben geleerd. We zullen dus al onze kennis en ervaring in gebruik moeten stellen voor dit project: algoritmiek, databases, netwerken, en nog veel meer.

2 Oriëntatie

Aan het begin van dit project zijn wij gestart met wekelijkse vergaderingen om ons concept van 'business' media in kaart te brengen. Na enkele vergaderingen zijn dan ook de Unique Selling Points (USPs) van CHAIN*els* opgesteld, en daaruit volgden al snel de informele requirements die in het systeem verwerkt moesten zijn. Deze hebben uiteindelijk het Product Backlog gevormd. Het oorspronkelijke business plan van Erik Blokhuis (zie Appendix G) speelde hierbij ook een zeer belangrijke rol.

2.1 Opzet

In het oriëntatieverslag in Appendix A is beschreven waarom wij voor het framework Scrum hebben gekozen. De requirements zijn opgesteld in een zogenaamde backlog, zoals gebruikelijk bij deze software ontwikkelmethode. Zie hiervoor Appendix B. Een ander belangrijk document bevat enkele aandachtspunten die binnen ons concept zeer belangrijk zijn, en is dus iets om altijd in het oog te houden. Zie voor dit document Appendix C.

2.2 Product backlog

Het product backlog is in overleg opgesteld, maar daarnaast hebben wij zelf ook heel veel invloed gehad op functionaliteiten die in het product backlog zijn gedefinieerd. Ons ontwikkelproces is in vijf fases verdeeld:

1) Alpha
2) Closed Beta
3) Open Beta
4) Release Candidate
5) Post Release

De bovenstaande vijf fases zijn tevens milestones in het ontwikkelproces. In het hoofdstuk 'Proces en evaluatie' zullen de milestones nog nader worden toegelicht.

2.3 Planning

Naast de requirements hebben we ook een planning besproken met het team. De planning is meerdere malen aangepast, met verschillende redenen. De gevisualiseerde planning is de uiteindelijke planning geworden. De planning voor relatief kleinere stukjes hebben we naar het Scrum model ingedeeld. Dit houdt in dat we door middel van sprints doelen bereikten. De sprints legden we vast in Flyspray, een tool om actiepunten op prioriteit en deadlines in te delen.

Data richtlijn	**Milestone**
Begin september 2011	Alpha
Half september 2011	Closed Beta
Half oktober 2011	Open Beta
Half november 2011	Release Candidate
Begin 2012	Post Release

3. Ontwerp

Om het product te verwezenlijken is er een plan van aanpak nodig. Het plan is intern besproken en heeft volgens het Scrum model iteratief plaatsgevonden. Dit houdt in dat er veel sprints hebben plaatsgevonden om bepaalde functionaliteiten te verwezenlijken op prioriteit.

3.1 LOGBOEK

Samen werken kan alleen als er duidelijk is wie wat doet en wanneer aan het product werkt, daarom is er op de minuut een logboek bijgehouden. Zowel revisions als de activiteiten zijn bijgehouden. Het geeft tevens de mogelijkheid om tussentijds te evalueren en een extra controle of we op schema aan het ontwikkelen. In Appendix E staat een voorbeeld van onze werkweek.

Datum	Tijd	Uren	Beschrijving	SVN Revision(s)
24-8-2011	14.00 - 17.00	3	In de code bezig geweest aan de hand van de gister in de vergadering besproken….	Rev. 235 t/m 239

3.2 USE CASE DIAGRAM

Tevens is er een Use Case Diagram gemaakt om de requirements in kaart te brengen. (fig. 3)

Figuur 3 - Ons use case diagram

Op het netwerk zijn bedrijven actief en deze worden aangestuurd door minimaal één persoon binnen het desbetreffende bedrijf. De persoon en het bedrijf zijn beide actoren in het diagram. Er is onderscheid gemaakt tussen het persoons- of bedrijfsafhankelijk gedrag.

Bedrijfsafhankelijk
Een bedrijf maakt bijvoorbeeld het profiel aan met alle bedrijfsinformatie (Create Profile)

Persoonsafhankelijk
Een persoon kan bijvoorbeeld haar gebruikersinstellingen aanpassen (Customize Usersettings).

"De persoon handelt namens het bedrijf op het netwerk, want in tegenstelling tot andere netwerken is CHAIN*els* bedrijfsgebonden en niet persoonsgebonden"

3.3 Algemene structuur

In het netwerk is er een duidelijke afscheiding gemaakt tussen de User Interface (UI) en het achterliggende systeem. Het achterliggende systeem is opgedeeld in twee delen: de front-end en de back-end. Onderhoudbaarheid is de voornaamste reden, maar het maakt het ook iets eenvoudiger om los van elkaar te werken. Ook onze werktijden en de vakantieperiode speelden hierbij een rol. Wij waren dus verantwoordelijk voor de gehele back-end en front-end.

De User Interface omvat alle schermen die men ziet in het netwerk.
De front-end omvat alle processen op de voorgrond (client-side).
De back-end omvat alle achterliggende processen, zoals: database, algoritmes, objecten, enzovoorts.

De back-end is in de volgende componenten op te delen:

1) Validation:
   De objecten bestaan uit attributen die vrijwel allemaal gevalideerd moeten worden voordat het in ons database wordt opgeslagen. Daarom heeft het netwerk een groot validatie gedeelte.

2) Multi-Language:
   Alle tekst in het netwerk moet in meerdere talen mogelijk zijn. Daarom is er een apart Multi-Language gedeelte. De gebruiker kan dus zelf een taal selecteren voor de website.

3) Tests:
   Een gedeelte met test klassen, zodat we kunnen valideren dat het systeem correct en consistent werkt. Het zal voornamelijk uit Unit tests bestaan. Daarnaast zullen er ook user tests worden afgenomen. Dit zal verder in het Hoofdstuk 'Testen' ter sprake komen.

4) Database:
   De data wordt opgeslagen in een database. Dit zal in de implementatie ook afgescheiden worden. Voor verdere informatie, zie de paragraaf Database.

3.4 ENTITEITEN

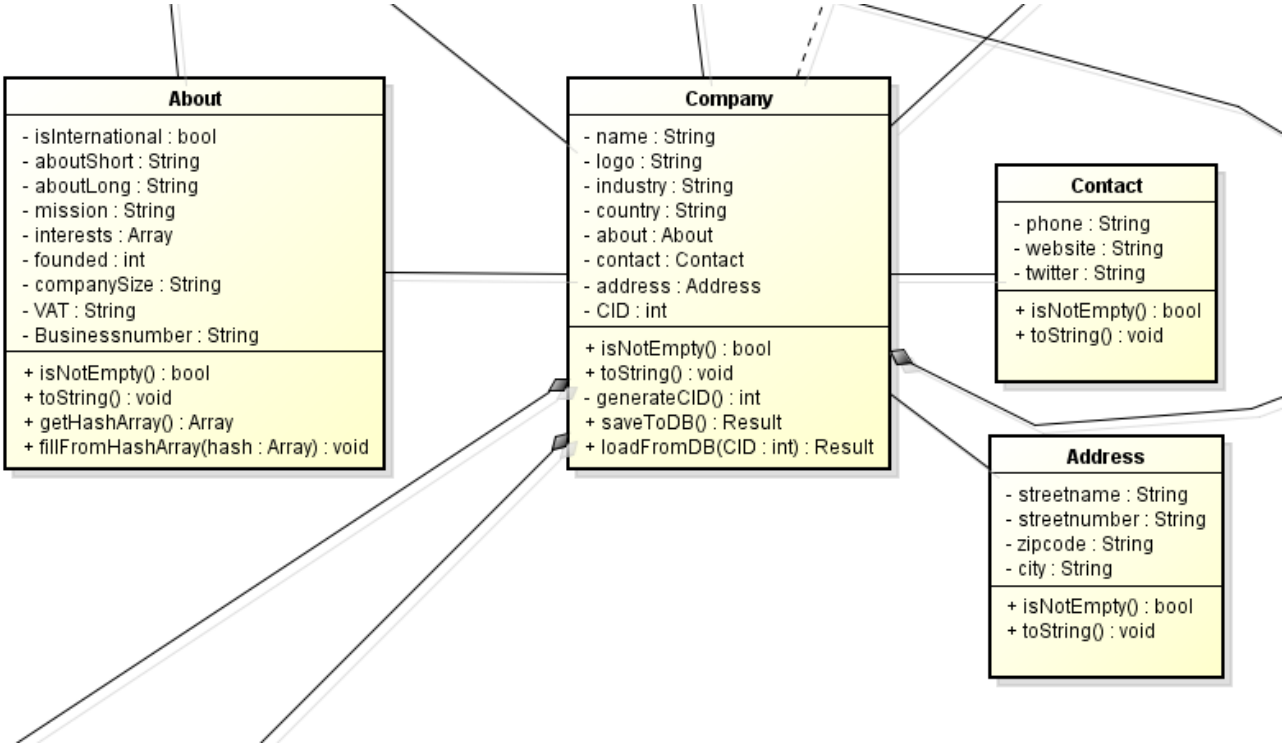
Binnen het netwerk erkennen wij de volgende entiteiten:

Entiteit	Korte uitleg
Company	Het bedrijf dat acteert op het netwerk
Account	De persoon die namens het bedrijf acteert op het netwerk
Address	De adresgegevens van een bedrijf
About	Algemene informatie over een bedrijf
Contact	Contactinformatie van een bedrijf
BaseMessage	De standaard message, waaruit alle verschillende berichten worden gemaakt (abstracte klasse)
WallMessage	Een bericht dat door een bedrijf op haar eigen pagin kan worden geplaatst
PrivateMessage	Een direct bericht tussen twee bedrijven
Reply	Een reactie op een WallMessage
Result	Een afgeleide van een boolean type; er zit indien nodig nog een foutmelding in verwerkt

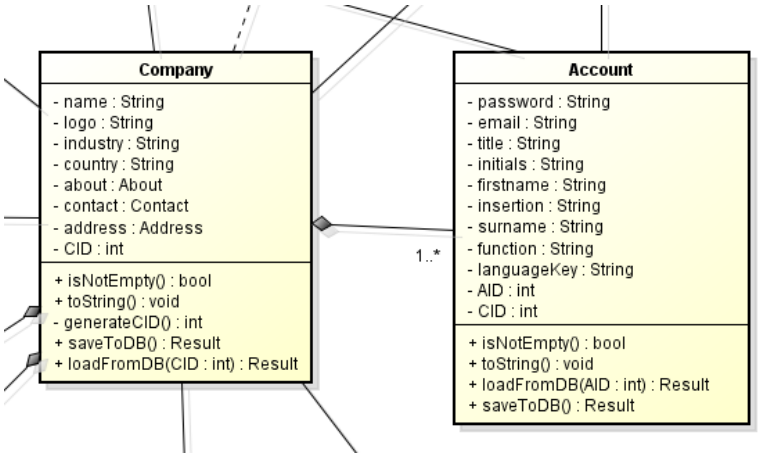
De entiteiten zijn afgeleid uit het Use Case Diagram.

Aan de hand van een klasse diagram worden de relaties tussen de verschillende entiteiten weergegeven en de functionaliteit van deze entiteiten vastgelegd.

De basis is de Company entiteit die verbonden is met een Address, About en Contact. De 'Company' bevat de informatie van het desbetreffende bedrijf. Om alles zo goed mogelijk onderhoudbaar te houden hebben we een aantal attributen gegroepeerd in: About, Address en Contact. Hierdoor blijft een relatief grote klasse klein en overzichtelijk. Dit is weergeven in figuur 4.

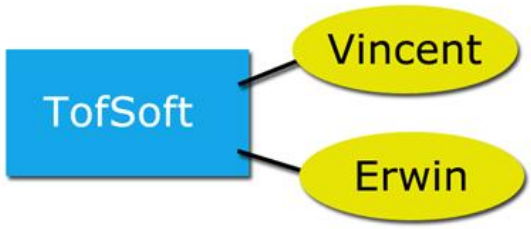


Figuur 4 - Een gedeelte van ons klassendiagram



Figuur 5 - Een gedeelte van ons klassendiagram

Het bedrijf wordt op het netwerk aangestuurd door een gebruiker die een Account bezit. Een account bevat de gepersonifieerde instellingen. Bedrijven bestaan vaak uit meer dan één persoon, en daarom moet het ook mogelijk zijn om met meerdere personen op één bedrijf actief te zijn. Zie hiervoor figuur 5.



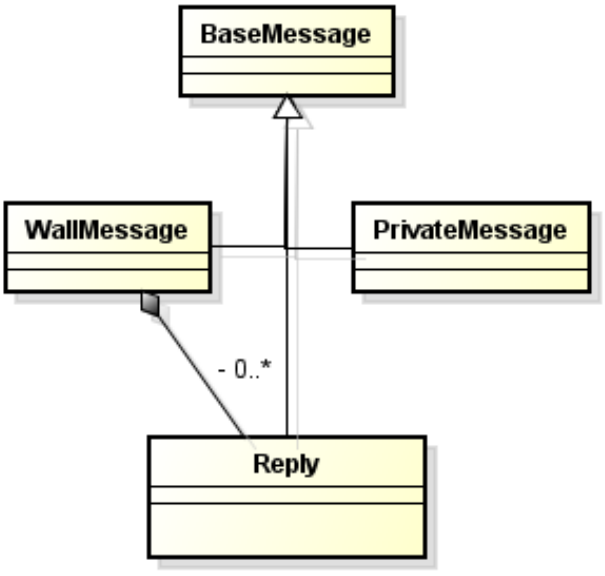
Figuur 6 - Een visualisatie van een bedrijf

In praktijk zal het als volgt schematisch gebruikt moeten worden: je hebt een bedrijf met daaraan een X aantal gebruikers gekoppeld. In figuur 6 is te zien dat Vincent en Erwin personen zijn, en dus Accounts hebben die gekoppeld zijn aan de bijbehorende Company met de naam TofSoft.

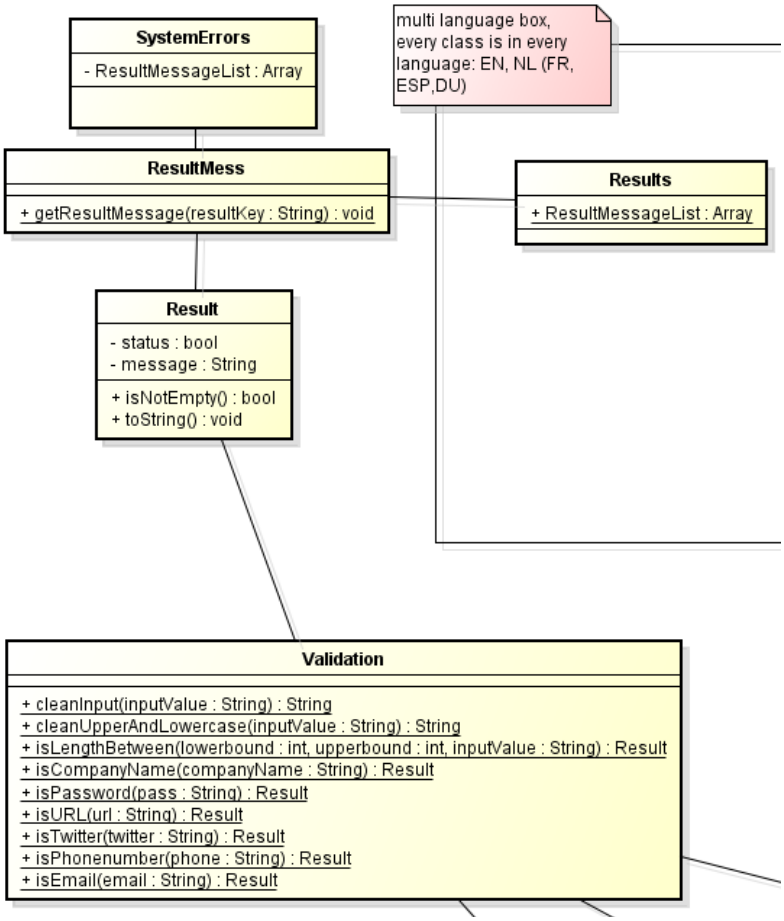
Op het netwerk is het mogelijk om berichten te versturen en op bepaalde berichten te reageren. We maken onderscheid tussen een PrivateMessage en een WallMessage

PrivateMessage:
Een bericht dat tussen twee bedrijven gestuurd kan worden: een direct message systeem. Het lijkt op het e-mailen tussen twee bedrijven.

WallMessage:
Een bericht dat zichtbaar is aan je Chains (Connecties) en/of de rest van de wereld. Dit is vergelijkbaar met tweets die je op Twitter plaatst, alleen niet met een beperkte lengte. Het is mogelijk om een reactie te plaatsen op een WallMessage: een Reply. Dit alles is gevisualiseerd in figuur 7.



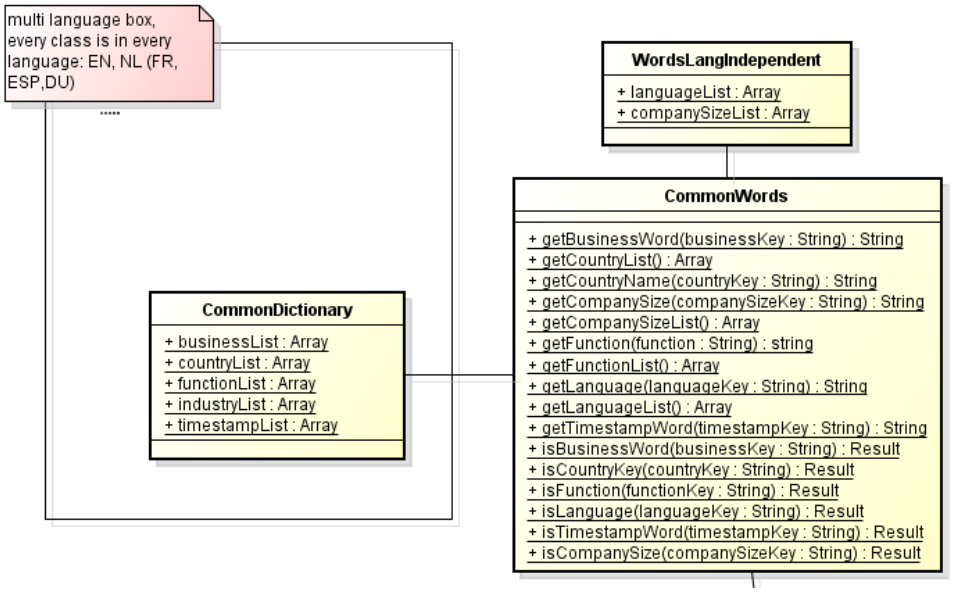
Figuur 7 - Een gedeelte van ons klassendiagram



Figuur 8 - Een gedeelte van ons klassendiagram

Alle data wordt gevalideerd in het systeem; data is dus per definitie in een goed of fout formaat. Het probleem is dat bij een fout formaat dit taalafhankelijk moet worden getoond aan de gebruiker met een melding. Het Result object biedt de uitkomst: het bevat naast een boolean een taalafhankelijke melding. Dit kan een System Error zijn of een indicatie voor de gebruiker wat er fout is aan de ingevoerde data.

Dit is gevisualiseerd in figuur 8.



Figuur 9 - Een gedeelte van ons klassendiagram

Het multi-language systeem is in het voorgaande voorbeeld al kort aan de orde geweest en is zowel voor foutmeldingen als andere tekst op dezelfde wijze ontworpen. Het basis principe is in figuur 9 weergeven.

1) Een functie wordt aangeroepen en op dat moment bepaalt de functie welk stuk tekst er nodig is en in welke taal. (klasse CommonWords)
2) Het gewenste stuk tekst word daarna opgevraagd uit de juiste taalafhankelijke klasse (klasse CommonDicitionary).
3) Het gewenste stuk tekst wordt dan teruggegeven (klasse CommonWords)

3.5 DATABASE

Zoals in Appendix A in het Oriëntatieverslag beschreven is, is er gekozen voor een Redis database. Het is in ons systeem namelijk van groot belang dat sommige data heel snel opgevraagd kan worden. De gekozen databasestructuur met de bijbehorende datatypen en dergelijke is daarbij van groot belang.

Omdat Redis geen conventioneel (relationeel) databasesysteem is, is er geen 'standaardschema' voor te maken. Evengoed is een overzicht wel mogelijk. In het hierna volgende overzicht gelden de volgende punten:

- Er wordt gebruik gemaakt van een zogenoemde 'seperator' (hier een dubbele punt) om 'subkeys' aan te geven. Een voorbeeld hiervan is comp:12345 of zelfs comp:12345:chains.
- De punten in de onderstaande lijst staan voor de keys; na de -> wordt de bijbehorende bedoelde waarde aangegeven.
- Een cursief woord (bijvoorbeeld *CID*) geeft aan dat dit vervangen moet worden door de daadwerkelijke waarde (bijvoorbeeld 12345).
- Een vetgedrukt woord geeft het datatype van het element aan.
- Bijvoorbeeld 'uit aStatus' geeft aan dat de enumeratie[1] 'aStatus' de bron is voor de waarde.
- Hashkeys geïdentificeerd met de kop *Basis* bestaan altijd, leeg of niet; andere keys bestaan mogelijk niet. Als er geen identificatie bijstaat wordt *Basis* aangenomen.
- Een array in een hash is geserialiseerd, oftewel gerepresenteerd met een string. Booleans worden met een integer (0 of 1) weergeven, maar deze integers zijn eigenlijk ook strings: in Redis is namelijk alles in beginsel een string.
- Alle waarden die in de Database staan zijn waar mogelijk altijd in een standaardformaat (door de validatie). Dit geldt voor bijvoorbeeld postcodes, telefoonnummers, enzovoorts.
- De keuze voor bijvoorbeeld een aparte set of een array binnen een hash hangt van het gebruik in de praktijk af. Zo is er bij bijvoorbeeld een set meteen op te vragen hoeveel elementen erin zitten; bij een array binnen een hash moet eerst de hele array worden binnengehaald, en vervolgens alle elementen geteld. De tweede methode kost echter wel minder ruimte: het scheelt weer een key, en hoe meer keys er zijn, hoe langzamer het databasesysteem is. Dit is overigens ook waarom de keys vaak zo kort mogelijk zijn: look-ups worden zo versneld.

[1] Wikipedia, *Enumerated type*, http://en.wikipedia.org/wiki/Enumerated_type (20-12-2011)

Het overzicht is dan als volgt (bekijk voor meer informatie het hoofdstuk 'Implementatie'):

- acc: -> Account
  - *AID* (**hash**)
    basis
    - CID (**integer**) -> CID van het bedrijf waar deze gebruiker bij hoort
    - lang (**string**) -> key (lengte 2) met de voorkeurstaal van de gebruiker
    - mail (**string**) -> e-mailadres van de gebruiker (unieke inlogkey ook!)
    - name (**string**) -> interne naam van de gebruiker
    - pass (**string**) -> wachtwoord van de gebruiker (Bcrypt hash)
    - right (**integer**) -> rechtenniveau van de gebruiker (1 t/m 5, uit Astatus)
    - status (**integer**)-> status van de gebruiker (1 t/m 3, uit Aright)
    - tz (**string**) -> de tijdzone van de gebruiker in standaardformaat
    - wants (**integer**)-> 0 of 1: wel of geen mails van CHAIN*els* ontvangen

    authenticatie
    - act (**string**) -> gegenereerde token voor een e-mailvalidatie
    - forgot (**string**) -> gegenereerde token voor een wachtwoordreset
    - prevtk (**string**) -> vorige sessieID van de gebruiker
    - token (**string**) -> gegenereerde sessieID van de gebruiker
  - *mail* (**string**) -> AID van de gebruiker behorende bij *mail*
  - token:*token* (**string**) -> AID van de gebruiker behorende bij *token*
- comp: -> Company (inc. About/Address/Contact)
  - *CID* (**hash**)
    basis (company)
    - country (**string**)-> key (lengte 2) met het vestigingsland van dit bedrijf
    - indus (**string**) -> key (lengte 2) met de industrie van dit bedrijf
    - name (**string**) -> de (niet-unieke) naam van dit bedrijf
    - status (**integer**)-> status van dit bedrijf (1 t/m 6, uit Cstatus)

    basis (about)
    - aLong (**string**) -> volledige informatie over dit bedrijf
    - aShort (**string**) -> beknopte informatie over dit bedrijf
    - COC (**string**) -> het KvK (o.i.d.) nummer van dit bedrijf
    - demand (**array**)-> alle demand-keys van dit bedrijf
    - inter (**integer**) -> 0 of 1: wel of niet internationaal opererend
    - miss (**string**) -> de missie van dit bedrijf
    - size (**string**) -> key (lengte 2) met het aantal bedrijfswerknemers
    - supply (**array**) -> alle supply-keys van dit bedrijf
    - VAT (**string**) -> het BTW (o.i.d.) nummer van dit bedrijf
    - year (**integer**) -> het oprichtingsjaar van dit bedrijf (4 cijfers)

    basis (address)
    - city (**string**) -> de vestigingsplaats van dit bedrijf
    - sName (**string**) -> de straatnaam van de vestiging van dit bedrijf
    - sNum (**string**) -> het huisnummer van de vestiging van dit bedrijf
    - state (**string**) -> de staat of provincie van de vestiging van dit bedrijf
    - zip (**string**) -> de postcode van de vestiging van dit bedrijf

    basis (contact)
    - phone (**string**) -> het telefoonnummer van dit bedrijf
    - twit (**string**) -> de naam van het Twitter account van dit bedrijf
    - web (**string**) -> de URL van de website van dit bedrijf

    social media
    - li (**array**) -> benodigde gegevens voor de LinkedIn koppeling
    - liT (**array**) -> tijdelijke dataopslag tijdens opzet LinkedIn koppeling

- tw (**array**) -> benodigde gegevens voor de Twitter koppeling
- twT (**array**) -> tijdelijke dataopslag tijdens opzet Twitter koppeling

overige

- hold (**integer**) -> het CID van de mogelijke holding van dit bedrijf
- logo (**string**) -> mogelijk geüpload bedrijfslogo in Base64 codering
- read (**array**) -> al gelezen notificaties (IMIDs)

- *CID:*AIDs (**set**) -> AIDs (accounts) die bij dit bedrijf horen
- *CID:*block (**set**) -> CIDs (bedrijven) die door dit bedrijf geblokkeerd zijn
- *CID:*chains (**set**) -> CIDs (bedrijven) waarmee dit bedrijf gechaint is
- *CID:*cReq (**set**) -> CIDs van bedrijven die aan dit bedrijf een chainverzoek hebben verstuurd
- *CID:*dReq (**set**) -> CIDs van bedrijven die aan dit bedrijf een departementsverzoek hebben verstuurd
- *CID:*fb (**hash**) -> benodigde gegevens voor de Facebook koppeling, tijdens opzet en gebruik; wordt door hun API beheerd
- *CID:*hReq (**set**) -> CIDs van bedrijven die aan dit bedrijf een holdingsverzoek hebben verstuurd
- *CID:*kids (**set**) -> CIDs van bedrijven die een departement van dit bedrijf zijn
- *CID:*MIDs (**list**) -> MIDs van berichten die dit bedrijf op haar Wall heeft geplaatst, met het laatst geplaatste bericht vooraan
- *CID:*notify (**list**) -> IIDs van notificaties (gelezen of ongelezen), met de nieuwste notificatie vooraan
- *CID:*PMs (**list**) -> PMIDs van privéberichten die dit bedrijf ontvangen heeft, met het nieuwste privébericht vooraan
- *CID:*sentPMs (**list**) -> PMIDs van privéberichten die dit bedrijf verstuurd heeft, met het laatst verstuurde bericht vooraan

- imess: -> <u>InfoMessage</u>
  - *IMID* (**hash**)
    - CID (**integer**) -> De 'veroorzaker' van dit infobericht
    - CIDto (**integer**) -> De 'tweede partij' bij dit infobericht
    - cont (**array**) -> Elementen voor de berichtsinhoud (taalonafhankelijk)
    - key (**string**) -> Key voor de berichtsinhoud (1 t/m 5, uit Ikey)
    - point (**string**) -> Verwijzing naar het object waar dit bericht bij hoort
- inv: -> <u>Invite</u>
  - *IID* (**hash**)
    - at (**integer**) -> Wanneer deze invite is gedaan (Unix timestamp)
    - mail (**string**) óf -> Bij reguliere invites: het ontvangende e-mailadres
    - acc (**string**) -> Bij social media invites: het ontvangende account
    - name (**string**) -> De naam van de ontvanger
- pmess: -> <u>PrivateMessage</u>
  - *PMID* (**hash**)

    Basis
    - attach (**string**) -> De URL van een mogelijke bijlage bij dit bericht
    - cont (**string**) -> De inhoud van dit privébericht
    - fromAID (**integer**)-> Het account dat dit bericht verzonden heeft
    - fromCID (**integer**)-> Het bedrijf dat dit bericht verzonden heeft
    - read (**integer**) -> 0 of 1: wel of niet gelezen (door de ontvanger)
    - toCID (**integer**) -> Het bedrijf waarnaar dit bericht verzonden is

    Gelezen
    - del (**string**) -> 'Y' indien dit bericht door één van de twee partijen verwijderd is; key bestaat anders helemaal niet

- reply: -> Reply
  - *RID* (**hash**)
    - AID (**integer**) -> Het account dat deze reactie geplaatst heeft
    - attach (**string**) -> De URL van een mogelijke bijlage bij deze reactie
    - CID (**integer**) -> Het bedrijf dat deze reactie geplaatst heeft
    - cont (**string**) -> De inhoud van deze reactie
    - prive (**integer**) -> 0 of 1: wel of niet privé
    - recom (**array**) -> CIDs van bedrijven die dit bericht aangeraden hebben

Opmerking: is altijd gekoppeld aan een WallMessage, maar deze koppeling bestaat alleen daarbij

- stat: -> Stats
  - *unieke paginaidentificatie* (**hash**)
    - at (**integer**) -> Wanneer dit paginabezoek is geweest (timestamp)
    - cc (**string**) -> Landscode van de bezoeker (2 letters)
    - cid (**integer**) -> Mogelijk CID van de bezoeker (indien ingelogd)
    - ct (**string**) -> Volledige stadsnaam van de bezoeker
    - ip (**string**) -> IP-adres van de bezoeker
    - rg (**string**) -> Regiocode van de bezoeker (standaardformaat)
- wmess: -> WallMessage
  - *MID* (**hash**)
    - AID (**integer**) -> Het account dat dit bericht geplaatst heeft
    - attach (**string**) -> De URL van een mogelijke bijlage bij dit bericht
    - cat (**integer**) -> De berichtsoort (1 t/m 3, uit Category)
    - CID (**integer**) -> Het bedrijf dat dit bericht geplaatst heeft
    - cont (**string**) -> De inhoud van dit bericht
    - keys (**array**) -> Labels die bij dit bericht horen (vraag/aanbod)
    - prive (**integer**) -> 0 of 1: privé of niet
    - recom (**array**) -> Bedrijven (CIDs) die dit bericht aangeraden hebben
    - replies (**array**) -> Reacties (RIDs) op dit bericht
  - *MID*:subs (**set**) -> Bedrijven (CIDs) die op dit bericht gereageerd hebben, en dus 'subscribed' zijn: ontvangen dan mails van verdere reacties

Afwijkend: de 'general sets'. Deze bevatten algemene data voor gebruik in verschillende algoritmes

- gen:
  - AIDs (**set**) -> Alle bestaande AIDs (Accounts)
  - CIDs (**set**) -> Alle bestaande CIDs (Bedrijven)
  - keys:*key* (**set**) -> Berichten (MIDs) waarbij deze key gebruikt is

3.6 Interactie

Onze 'ICT-module' moet eenvoudig te gebruiken zijn in de User Interface. Dit is onder andere gerealiseerd door in de ontwerpfase extra aandacht te besteden aan het formaliseren van goede functionele namen. Het is de bedoeling dat de UI zoveel mogelijk onthouden wordt van complexe algoritmes.

Vandaar dat het ook zo ontworpen is dat een UI programmeur minimale kennis van back-end zaken als de Database hoeft te hebben. Fouttolerantie speelt hierbij ook een grote rol, omdat vrijwel alle data gevalideerd wordt in ons systeem. Fouten krijgen een negatief Result object terug, waarmee mogelijk fout gebruik van de back-end zoveel mogelijk voorkomen wordt.

3.7 Problemen en proces

Na het eerste ontwerp bleek tijdens het implementeren dat het ontwerp niet compleet was. Ontwerp fase en Implementeer fase werd dan ook een alternerend proces. Een van de oorzaken was dat de requirements meerdere malen zijn bijgesteld. Het concept zelf was wel duidelijk, maar de daadwerkelijke invulling ervan op ICT gebied was afhankelijk van de ontvangen feedback tijdens het proces.

Het grootste probleem was dat de User Interface nog niet was ontwikkeld tijdens ons ontwikkelproces, waardoor er in een latere fase functionaliteiten veranderd moesten worden.

Bovendien krijgen wij in het proces bij het ontwerp zelf een steeds belangrijkere rol, omdat het team uit elkaar viel. De bedenker van het oorspronkelijke concept, Erik Blokhuis, stapte namelijk door privéredenen uit het team, en gaf ons het recht verder te gaan met dit Business media concept. We kregen hierdoor onverwachts veel meer verantwoordelijkheid, en dit heeft dan ook veel invloed gehad op het huidige ontwerp.

4 IMPLEMENTATIE

Na de ontwerpfase zijn we begonnen met het implementeren van de ontworpen UML. Zoals eerder beschreven hebben we meerdere malen van Ontwerpfase en Implementatiefase geruild. Bij het implementeren hanteerden wij het Scrum model, zie hiervoor Appendices A (onderdeel Ontwikkelmethode) en B (geheel). Aan de hand van de onderstaande hoofdstukken behandelen wij de implementatie van de back-end.

4.1 BASIS

In het ontwerp zijn de verschillende componenten in de back-end aan de orde gekomen. In het product backlog (Appendix B) staat gedefinieerd welke requirements als eerste zijn gerealiseerd. De basis bestaat uit alle Entiteiten die essentieel zijn voor het netwerk: Company, Account, Address, Contact, About, Result.

Om een consequente basis te schrijven hebben wij een soort interface ontwikkeld voor de object klasse. Zo bevat elk object bijvoorbeeld een isNotEmpty en __toString() functie. Daarnaast worden de get en set methoden altijd op dezelfde wijze opgebouwd. Iedere eerder genoemde klasse is getest met een Unit test, om te controleren of deze op de juiste wijze functioneert.

Met behulp van de __toString() methode waren we instaat alle objecten netjes te visualiseren. Zie voor een voorbeeld van deze visualisatie figuur 10.


company: TofSofts
logoURL: ico/logo.png
industry: Facility services and ICT **key**
country: The Netherlands **key**
inter/national: International
about (short)
TofSoft is een sterk klantgericht bedrijf met de doelstelling om programmaâ€™s en websites zo simpel en duidelijk mogelijk te maken, en dat tegen lage kosten.
about (long)
TofSoft ontwikkelt zelf programma's en applicaties die te downloaden zijn. UurBeheer is bijvoorbeeld een gratis programma om eenvoudig uw gewerkte uren bij te houden. TofSoft ontwikkelt software "on request". U kunt bij ons verzoeken indienen om bepaalde software

Figuur 10 – Resultaat van de toString methode

Het systeem is zo geïmplementeerd dat er in theorie nooit verkeerde data in de database kan worden opgeslagen, omdat een set methode alleen slaagt indien de data voldoet aan de validatie. De validatie zal in de volgende paragraaf worden toegelicht. In het systeem wordt er onderscheid gemaakt tussen vooraf gedefinieerde lijsten en zelf in te voeren waarden. Vooraf gedefinieerde waarden zijn doormiddel van een associatieve key-array geïmplementeerd. Het is hierbij belangrijk dat er rekening gehouden wordt met het ondersteunen van verschillende talen.

4.2 VALIDATIE

Ingevoerde data wordt altijd via een set methode ingesteld. Indien de invoer data niet aan de eisen voldoet van de desbetreffende set methode wordt de waarde niet ingesteld en geeft de functie een Result object terug met een foutmelding. Om redundantie in de validatie te voorkomen hebben wij een aparte Validation klasse gemaakt en daarin basis-validatiefuncties gedefinieerd doormiddel van statische methodes. Denk hierbij aan een isLengthBetween(lowerbound, upperbound, input) methode, waarbij gecontroleerd wordt of de lengte van de invoer voldoet aan de gestelde onder- en bovengrens. Alle vergelijkbare valideermethoden geven een Result object terug, zodat bij foute invoer altijd een waarschuwing of foutmelding wordt meegegeven.

Naast de al eerder relatief eenvoudige valideermethoden zijn er ook attributen die landafhankelijk zijn. Een voorbeeld hiervan is de postcodes: waar Nederland een 1234XX formaat heeft, heeft België een 1234 formaat, enzovoorts. Dit is bij de implementatie opgelost door een basisvalidatie voor postcodes te definiëren, en dan afhankelijk van de landkeuze een reguliere expressie in te laden om het formaat te controleren. De volgende reguliere expressie[2] wordt bijvoorbeeld gebruikt voor de Nederlandse postcodes:

$$(0 \dots 9)^4 (space)^? (A \dots Z)^2$$

Door de lage fouttolerantie is de data in theorie altijd zuiver in het systeem. De waarde van een netwerk ligt in de data, en daarbij is het van belang dat het correcte data is; dit is de voornaamste reden om de fouttolerantie zo laag mogelijk te houden. Bij het bovenstaande is overigens op te merken dat de postcode uiteindelijk zonder mogelijke spaties in het systeem wordt opgeslagen, om zo de uniformiteit van de data ook nog eens te verhogen.

4.3 MULTI-LANGUAGE SYSTEEM

Al enkele keren is beschreven dat het systeem taal afhankelijk is. In deze paragraaf zal verder worden toegelicht hoe dit geïmplementeerd is in het systeem.

Per klasse is er in iedere taal een aparte versie. Indien een taal element nodig is zal alleen de klasse worden ingeladen van de op dat moment geselecteerde taal. Dit voorkomt dat er onnodig veel data wordt ingeladen. Aangezien de tekst niet snel verandert zijn het statische lijsten met woorden.

In het onderstaande voorbeeld wordt indien Nederlandse taal is ingeladen de Nederlandse klasse gebruikt:

Nederlandse klasse	**Engelse klasse**

Class Example
{
exampleList =
['NL' => 'Nederland',....]
}
	Class Example
	{
	exampleList =
	['NL' => 'The Netherlands',....]
	}

[2] PHP Manual, *PCRE*, http://php.net/manual/en/book.pcre.php (20-12-11)

Dit is zo geïmplementeerd dat het in de toekomst mogelijk is om een programma te ontwikkelen om eenvoudig talen toe te kunnen voegen aan dit systeem. Met een dergelijk programma kan een tolk dit zonder veel benodigde systeemkennis zeer snel bereiken.

4.4 Bedrijven en gebruikers

Aan de basis van het hele netwerk staan de 'objecten' die de gebruikers en de bedrijven representeren.

4.4.1 Accounts

Om bij de gebruiker te beginnen bestaat er de Account klasse, waarin enkele basisbenodigdheden zijn gedefinieerd. Uit het ontwerp volgt dat een account enkele belangrijke attributen heeft die vooral als 'instelling' te classificeren zijn; er is niet veel informatie. Dit is logisch gezien het feit dat we op bedrijven gericht zijn, en niet op personen. Het belangrijkste is de e-mail-wachtwoord(hash) combinatie: de gegevens waarmee een gebruiker kan inloggen. Het e-mailadres wordt, zoals eerder bij validatie besproken, gevalideerd voordat het ingesteld kan worden. Ook wordt gecontroleerd of het e-mailadres nog niet bestaat: elk e-mailadres moet namelijk verplicht per gebruiker uniek zijn. Hoe het wachtwoord wordt opgeslagen en dergelijke wordt in de komende paragraaf over beveiliging besproken.

Ook zeer belangrijk zijn de velden voor taal en tijdzone. De taal wordt voor een nieuw account automatisch 'gegokt': door geografische locatie van het IP-adres van de gebruiker kan de taal van de gebruiker benaderd worden. Op het moment is CHAIN*els* alleen nog in het Engels en Nederlands beschikbaar, dus wordt de taal op Nederlands ingesteld indien de gebruiker binnen Nederland is (code NL), en anders op Engels. Door deze geografische locatie kan ook meteen de tijdzone van de gebruiker worden bepaald. Dit is van belang voor bijvoorbeeld het weergeven van de plaatsingstijd van een bericht.

Deze geografische bepalingen gebeuren overigens door een externe PHP module: GeoIP. Als deze module voor een bepaald IP-adres geen gegevens kan bepalen, wat kan gebeuren, dan worden de standaard waarden van Engels en tijdzone GMT+1 (Amsterdam) gebruikt. Bovendien kunnen de taal en tijdzone van de gebruiker altijd door de gebruiker zelf aangepast worden.

De zojuist besproken waarden worden ook in zogenaamde cookies opgeslagen. Dit zijn bestanden die bij de gebruiker zelf worden opgeslagen. Zodoende hoeft niet bij elk bezoek van de pagina alles opnieuw bepaald of opgehaald te worden; één keer ophalen en bij de gebruiker opslaan is genoeg! Als de gebruiker zelf een waarde aanpast dan wordt dit ook direct zowel in de cookies als in de database bijgewerkt. Om opslagruimte te besparen wordt slechts de 'key' van een taal opgeslagen: de unieke identificatie (zoals 'NL' voor 'Nederland'). Dit zelfde principe geldt ook voor andere velden die niet 'geheel willekeurig' zijn zoals een gebruikersnaam of een wachtwoord, en bovendien op vrijwel elke pagina nodig zijn.

Verder heeft een account nog een tweetal belangrijke velden die de gebruiker zelf kan beïnvloeden: interne naam en 'wantsMails'. Met de interne naam wordt voor intern gebruik (alleen binnen het bedrijf dus) de naam van de gebruiker duidelijk. Met het 'wantsMails' veld kan de gebruiker aangeven of hij/zij simpelweg wel of geen mails van CHAIN*els* wil ontvangen.

Velden die niet door de gebruiker kunnen worden ingesteld zijn onder andere de status en de rechten. Met het statusveld wordt dynamisch bepaald 'hoe ver' de gebruiker is in het compleet maken van de gegevens van zijn account. Een account kan namelijk 'new', 'completedStep1', 'completedReg', 'validated', of 'completedInfo' zijn. Deze statussen zijn in een enumeratie (vaste verzameling) vastgelegd en zijn allen incrementeel: een account is nieuw als er nog geen gegevens bekend zijn, een account is 'bij stap 1' als de basisvelden e-mailadres, wachtwoord, taal en rechten bekend zijn, een account is compleet als het vorige geldt er een compleet bedrijf bij hoort (dit wordt zo dadelijk besproken), het is gevalideerd als het vorige geldt en het e-mailadres met een activatiemail gevalideerd is, en uiteindelijk compleet als alle mogelijke velden door de gebruiker ingesteld zijn. Voor dit laatste wordt dynamisch de 'compleetheid' van een account in procenten uitgedrukt. Dit gebeurt simpelweg door het delen van het aantal velden dat de gebruiker kan instellen door het aantal velden dat daadwerkelijk ingesteld is ('niet leeg'). Bij een account is dat uiteindelijk alleen nog maar de interne naam, gezien het feit dat de rest automatisch bepaald wordt (en dan alleen nog maar naar iets anders gewijzigd kan worden) of al ingesteld is. Elke keer als een account opgeslagen wordt, wordt opnieuw de status van het Account geëvalueerd. Zo kan er in de website zelf meteen daarop actie worden opgenomen, zoals het doorgaan naar de volgende stap van de registratie als een account 'bij stap 1' is. Op basis hiervan worden er ook suggesties aan de gebruiker gegeven over hoe zijn of haar account en/of bedrijf 'completer' kan worden.

De activatiemail is nog een apart onderwerp; we willen namelijk dat het e-mailadres van de gebruiker ook echt zijn of haar e-mailadres is. Zodoende wordt er naar de gebruiker een mail gestuurd met daarin een unieke link waarmee het e-mailadres gevalideerd kan worden. Dit gebeurt door een willekeurige code van twaalf tekens te genereren, deze in de database op te slaan, en daarna in de betreffende link te verwerken. Zo is de e-mailadres/code combinatie voor elke gebruiker uniek, niet zelf 'te gokken', gezien het feit dat de link slechts naar het opgegeven e-mailadres wordt verstuurd.

Verder bepaalt het rechtenveld 'wat de gebruiker mag doen' voor zijn of haar bedrijf op CHAIN*els*. Voor de gebruiker die als eerste een bedrijf aanmaakt is dit automatisch alles ('Manager'), maar een bedrijf kan ook meerdere gebruikers hebben (wordt later besproken) op uitnodiging van een Manager gebruiker. Bij deze uitnodiging kan een Manager instellen welke rechten het nieuwe account krijgt: Manager, Editor, of Spectator. Aan de hand van deze rechten wordt in de website zelf bepaald wat de gebruiker wel of niet kan doen; dit ligt dus niet in ons systeem, maar in het UI gedeelte: acties die een gebruiker niet mag doen worden daar simpelweg verborgen. In het kort kan een Editor geen bedrijfsgegevens aanpassen of nieuwe mensen toevoegen, maar wel berichten plaatsen en dergelijke; een Spectator kan alleen maar de site bekijken en helemaal niets aanpassen of plaatsen.

4.4.2 Identification Numbers (IDs)

Als laatste zijn er nog de belangrijke velden AccountID (AID) en CompanyID (CID). Het AID is een gegenereerde unieke code die een gebruiker identificeert; een CID doet hetzelfde voor een bedrijf. Deze unieke code wordt bij de eerste opslag van een Account bepaald door de statische hulpfunctie 'getTimeStamp'.

In beginsel is de unieke code namelijk puur een 'timestamp': de huidige tijd in microseconden uitgedrukt. Dit kan door de verstreken tijd sinds de zogenaamde 'Unix Epoch' te tellen: 1 januari 1970[3], wat een PHP server standaard kan[4].

Als we deze verstreken tijd maar precies genoeg meten, zodat er nooit twee opeenvolgende tijden hetzelfde kunnen zijn, hebben we iets waarmee we elke gebruiker uniek kunnen identificeren, en bovendien kunnen we ook nog weten sinds wanneer de gebruiker lid is, of bijvoorbeeld wanneer een bericht geplaatst is. Hoeveel decimalen we nodig hebben hangt af van de snelheid van de server: er moet gezorgd worden dat zelfs direct opeenvolgende gegenereerde timestamps anders zijn. Een factor 10^-6 (oftewel microseconden) bleek hiervoor proefondervindelijk genoeg te zijn. Als servers sneller worden kan deze factor verhoogd worden, om zo aan de Wet van Moore te blijven voldoen[5]. 'Tijd' is immers het enige dat echt variabel is in een serveromgeving! Er is echter nog een groot probleem: dit werkt prima op één server, waar dus opeenvolgende 'requests' altijd een unieke timestamp opleveren, maar op meerdere servers is daar geen garantie voor! Nu kun je de kans dat er op precies dezelfde tijd een zelfde timestamp wordt gegenereerd verkleinen door nog meer decimalen te nemen, maar er is dan alsnog altijd 'een' kans. Daarom zou je misschien wel elke keer moeten controleren of 'jouw timestamp' niet al in gebruik is; een tijdrovend proces. De oplossing is echter veel eenvoudiger: zorg dat één server nooit een timestamp kan genereren dat gelijk is aan een timestamp van een andere server. Met twee servers is dit vrij simpel: de ene server genereert altijd even timestamps (een timestamp is immers gewoon een getal), en de andere server genereert altijd oneven timestamps. Zo kan een 'collision' nooit meer voorkomen! Met meerdere servers zal dit uitgebreid moeten worden tot misschien zelfs een aparte 'eindreeks' van de timestamp per server. Deze kleine verandering (of toevoeging) aan de timestamp heeft verder geen effect, aangezien voor de gebruiker de tijd op een dergelijke nauwkeurigheid allang al niet meer van belang is. Dit systeem wordt dus niet alleen voor een AccountID gebruikt, maar ook voor alle andere IDs.

4.4.3 DATABASEKOPPELING

Als we uiteindelijk al deze velden hebben, moeten deze wel in het PHP object naar de database bewaard kunnen worden, en er ook weer uit geladen. Bij het bewaren van een Account wordt er eerst gecontroleerd of er wel een e-mailadres is ingesteld: zonder deze unieke identificatie valt er niets op te slaan. Ook enkele andere basisvelden worden "niet leeg" vereist. Als er nog geen AID is, dan wordt deze voor de gebruiker gegenereerd in de opslagfunctie. Als er wel een AID is betekend dat dat we een bestaand account aanpassen, en dit vereist enige zorgvuldigheid. De gebruiker kan namelijk mogelijk zijn of haar e-mailadres hebben veranderd, waardoor er extra acties moeten worden ondernomen. Onder andere hierdoor is een opslag ook altijd 'volledig': alle data wordt altijd opnieuw geheel opgeslagen. Je zou namelijk ook alleen de gewijzigde velden kunnen opslaan bij elke wijziging; echter geeft een check op wijziging meer 'overhead' en veiligheidsrisico's dan alles geheel naar de database te sturen. De omvang van al deze data is namelijk zeer gering, en dat maakt extra checks inefficiënt.

---

[3] BJ Sintay, *Unix Time Stamp*, http://www.unixtimestamp.com/index.php (26-11-11)

[4] PHP Manual, *microtime*, http://php.net/manual/en/function.microtime.php (26-11-11)

[5] Wikipedia, *Moore's law*, http://en.wikipedia.org/wiki/Moore's_law (20-12-11)

Vlak voordat het account dan daadwerkelijk wordt opgeslagen wordt zoals eerder genoemd de status van het account gecontroleerd. Daarna worden de velden conform ontwerp in de database opgeslagen.

Hiervoor zijn verschillende commando's nodig, waardoor er nog een klein probleem ontstaat. Stel bijvoorbeeld dat we eerst het account opslaan, en daarna het ID van dat account aan de set met gebruikte Ids willen toevoegen. Als de server dan na het opslaan om welke reden dan ook wegvalt hebben we een probleem: we hebben wel een account, maar die staat niet in de set waarin hij wel hoort te staan. Zo krijg je dus zeer vervelende inconsistentie! We hebben dus een concept van 'alles of niets' nodig, wat zeker bestaat.

Zo zijn er namelijk transacties mogelijk in Redis: een blok van commando's dat of in zijn geheel wordt uitgevoerd door de server, of anders helemaal niet[6]. Bovendien is er ook nog de garantie dat tijdens de uitvoering van een transactie er niets anders wordt uitgevoerd, wat weer iets meer veiligheid oplevert. Naast deze dataveiligheid zijn transacties ook efficiënter: in plaats van bij elk los commando een antwoord te genereren en terug te sturen gebeurt dit nu in één keer bij het uitvoeren van de transactie. Dit levert code-technisch echter wel problemen op: een zogenaamd 'multi-exec' block is namelijk een constructie die impliciet wordt aangegeven met functieaanroepen. Met andere woorden: aanroepen binnen een dergelijk blok kunnen bijvoorbeeld niet naar een functie worden verplaatst, omdat binnen die ene functie dan niet gegarandeerd kan worden dat er in een transactie gewerkt wordt, wat dan wel vereist is. Zo worden wij gedwongen om dergelijke code 'bij elkaar te houden', in plaats van voor eenvoudig onderhoud op te splitsen. Dit nadeel hebben we helaas simpelweg moeten accepteren.

Voor het inladen van een account worden simpelweg alle opgeslagen velden in één keer opgehaald en ingeladen. Ook het verwijderen van een account is mogelijk, en gebeurt simpelweg door de opgeslagen velden uit de database te verwijderen. Ook dat gebeurt in één transactie zodat een account altijd gegarandeerd helemaal weg is, en niet gedeeltelijk. Overigens wordt het 'antwoord' van de database op commando's altijd gecontroleerd, zodat de beoogde actie gegarandeerd gebeurd is. Er kunnen namelijk mogelijk zeldzame fouten optreden waardoor commando's niet goed worden uitgevoerd. Er wordt dan bijvoorbeeld in plaats van waar (true) onwaar (false) door de functie geretourneerd, wat door ons gecontroleerd en afgevangen wordt in een foutmelding (in het systeem zoals eerder besproken).

4.4.4 Bedrijven

Bedrijven werken ongeveer hetzelfde als accounts, alleen dan uitgebreider. Dit omdat een bedrijf veel meer informatie met zich meebrengt dan een gebruiker. Ons netwerk is immers gericht op bedrijven! Zo heeft een 'Company' object de standaard velden naam, industrie, land, status en CID, maar ook de samengestelde velden 'about, 'address en 'contact'. De standaard velden zijn ongeveer gelijk als bij een account, en behoeven dan ook geen uitgebreide uitleg. Belangrijk is echter dat, in tegenstelling tot het e-mailadres van een account, een naam hier niet uniek hoeft te zijn. Het CID is dus het enige wat een bedrijf uniek identificeert. Dit is ook geen probleem, omdat deze CIDs altijd hetgene is wat op andere plekken het bedrijf identificeert. Een e-mailadres moet uniek zijn, omdat de gebruiker daarmee moet kunnen inloggen (in plaats van een lang getal).

---

[6] Redis, *Transactions*, http://redis.io/topics/transactions (26-11-11)

De samengestelde velden zijn echter afwijkend; deze samenstellingen representeren namelijk aparte objecten binnen de code. Zo bevat het aboutveld alle relevante velden met bedrijfsinformatie (zie ontwerp), en hetzelfde geldt voor contact en adres. Deze drie objecten bevatten dus puur dynamische informatie die door de gebruiker zelf ingesteld wordt.

Ze zijn voornamelijk voor optimalisatie en overzicht apart getrokken: als er geen contactgegevens nodig zijn hoeven deze immers ook niet ingeladen te worden. Bovendien zijn de objecten op zich herbruikbaar binnen andere klassen. Overigens valt het misschien op dat de velden industrie en land mogelijk ook binnen een van de samengestelde klassen zouden kunnen vallen. Omdat deze twee eigenschappen echter zo belangrijk zijn en zo vaak worden gebruikt hebben we besloten deze in de 'hoofdklasse' te houden voor snellere werking.

Voor het opslaan van samengestelde objecten is iets meer nodig dan het opslaan bij een account; verder werkt het proces vrijwel hetzelfde. Er wordt per samengesteld object gevraagd 'geef me je waarden'. Dit kan, omdat opslag gebeurt met een 'keyed array': een array waar in plaats van simpele indexen (0,1, enzovoorts) namen worden gebruikt. Eigenlijk een hashtable (dictionary) dus. De functie about geeft dus bijvoorbeeld een dergelijke keyed array terug met alle velden en waarden die daar van belang zijn, waarna deze array samengevoegd wordt met de 'basisarray' van het bedrijfsobject. Zo hoeft de opslagmethode daar niets te weten over hoe de about klasse in elkaar zit: scheiding van verantwoordelijkheden! Hetzelfde geldt ook voor de berekening van de 'volledigheid' van het bedrijfsobject, wat op een gelijke manier als bij een account gebeurt: er wordt apart aan het about object de volledigheid van dat object gevraagd, waarbij het aantal velden en het aantal ingevulde velden apart wordt opgevraagd, om zo de totale berekening in het basisobject kloppend te maken.

Als laatste opmerking binnen dit onderwerp zijn er zogenoemde interfaces voor deze klassen. Account en Company voldoen namelijk bijvoorbeeld beide aan een standaard profiel, waarin in een aantal vereiste functies worden gedefinieerd. Zo is er een standaard BaseObject interface die de laad- en opslagmethoden vereist, samen met de standaard 'isNotEmpty' en 'toString' functies die wij altijd gebruiken. De uitbreiding hierop is de BaseData interface die de vereiste functies voor de compleetheid van een object definieert. De BaseExtraData interface is voor de hierboven genoemde 'samengestelde objecten'. Met behulp van deze interfaces wordt uniformiteit gegarandeerd, wat het proces voor ons makkelijker maakt.

Overigens kan een bedrijf ook nog een logo uploaden; deze wordt gecomprimeerd met het standaard PNG formaat en volgens de standaardcodering (Base64) in de database opgeslagen.

4.5 Chains en invites

Na het registratieproces, waarin de gebruiker de basisgegevens van zijn of haar gebruikersaccount en bedrijfsprofiel heeft ingesteld, is één van de eerste mogelijkheden het 'digitaliseren van je bedrijfsnetwerk', oftewel het aanmaken van zogenoemde chains. Een chain is puur een indicatie van 'ik doe zaken met dit bedrijf', en zorgt er bijvoorbeeld voor dat berichten van dat bedrijf in jouw overzicht terechtkomen. Chains worden intern opgeslagen door simpelweg het ID van een bedrijf aan de set van chains van jouw bedrijf toe te voegen. Echter moet een bedrijf niet zomaar iedereen als 'chain' kunnen aangeven: er moet een systeem van verzoeken inzitten. Zo is het dan ook bij ons geïmplementeerd!

4.5.1 CHAINS

In de aparte (functionele) klasse Chains zijn alle benodigdheden hiervoor geïmplementeerd. Een bedrijf initieert het proces door een zogenoemd 'chainverzoek' naar een ander bedrijf te sturen. Dit wordt gerealiseerd door het ID van het verzoekende bedrijf in de set van verzoeken van het ontvangende bedrijf toe te voegen. Tegelijkertijd wordt er een e-mailtje (indien gewenst) met dit verzoek naar alle gebruikers bij de ontvangende company verstuurd.

De ontvanger kan vervolgens op een aparte pagina alle verzoeken die naar hem of haar zijn verstuurd bekijken. Deze set is heel simpel op te vragen, aangezien die bij het ontvangende bedrijf zelf bestaat. Bovendien kunnen de gewenste gegevens van een bedrijf aan de hand van het ID gemakkelijk worden opgevraagd. Het ontvangende bedrijf kan vervolgens het verzoek accepteren of weigeren. Bij weigeren van het verzoek wordt het ID simpelweg uit de set verwijderd, en anders wordt het ID vanuit de set doorgezet naar de set met chains van beiden bedrijven. Beide bedrijven krijgen bij acceptatie bovendien een mailtje van deze actie, en er wordt een zogenaamde 'InfoMessage' gegenereerd, die bij de volgende paragraaf Berichtverkeer nog besproken zal worden.

Chains kunnen later ook weer eenzijdig verbroken worden; de IDs van beide bedrijven worden dan simpelweg uit elkaars chainsets verwijderd. Ook kan een bedrijf zorgen dat er geen chainverzoeken meer van een ander bedrijf binnenkomen door dat bedrijf simpelweg te blokkeren. Bij een blokkeeractie wordt het ID van het te blokkeren bedrijf bij het blokkerende bedrijf in een set gezet. Bij acties als chainen, privé berichten versturen (zie kop Berichtverkeer) en meer wordt deze set van te voren gecontroleerd, en de acties worden niet uitgevoerd als er een blokkade bestaat. Het geblokkeerde bedrijf krijgt daar dan een melding van.

De chainsets met de IDs van bedrijven bieden nog een aantal 'extra opties' die voor de gebruiker van belang kunnen zijn. Zo kan bijvoorbeeld de intersectie tussen twee sets (verzamelingen) worden berekend om te kijken welke chains twee bedrijven gemeen hebben. Dit hele systeem van verzoeken wordt niet slechts bij chainverzoeken gebruikt, maar ook bij zogenoemde 'holding- en departementverzoeken'. Het is in CHAIN*els* namelijk mogelijk om aan te geven dat een bedrijf de holding is van jouw bedrijf, of, andersom, dat een bedrijf een van jouw departementen is. Dit 'aangeven' werkt op vrijwel dezelfde manier als bij chainverzoeken, en is in de 'Holdings' klasse verwerkt (zie Ontwerp). Ook hierbij wordt met blokkades en dergelijke rekening gehouden. Aangeven dat een bedrijf een holding of een departement is, is eigenlijk een speciaal geval van chainen. Het is namelijk niet alleen een 'aanduiding van vriendschap', maar geeft ook nog eens een hiërarchische relatie aan. Daarom kan bijvoorbeeld een bedrijf niet tegelijk een chain en een departement zijn van een ander bedrijf. Bovendien kan één bedrijf slechts één holding hebben, en een bedrijf kan ook niet de holding van zijn of haar holding worden. Hiervoor zijn dus extra controles aangebracht. Het verdere systeem werkt echter zoals gezegd op vrijwel dezelfde manier.

4.5.2 INVITES

Als laatste vallen onder dit onderwerp nog de zogenoemde 'Invites'. Dit zijn uitnodigingen naar bedrijven die nog niet op CHAIN*els* zitten om ook in het netwerk van het uitnodigende bedrijf te komen door te registreren op CHAIN*els*. Een gebruiker kan per e-mail of via sociale netwerken worden uitgenodigd. In de paragraaf 'Social Media' zullen we de uitnodigingen via sociale netwerken verder toelichten, maar het principe is hetzelfde als bij een uitnodiging via de mail.

Een bedrijf op CHAIN*els* kan namelijk een ander bedrijf uitnodigingen door simpelweg het e-mailadres en de naam van dat bedrijf op te geven, samen met de taal waarin de uitnodiging verstuurd dient te worden. Het bericht zelf wordt namelijk volledig door ons (multi-language)systeem gegenereerd. Voor elke uitnodiging wordt er een apart object in de database opgeslagen, geïdentificeerd door de MD5-hash van het uitgenodigde e-mailadres. Dit hashen is wegens veiligheidsredenen: een kwaadwillende gebruiker kan zo minder eenvoudig 'invites gokken'.

In een dergelijk Invite object wordt op een zelfde manier als bij bedrijven en accounts wat data opgeslagen, zoals een set van bedrijven die een uitnodiging naar het betreffende e-mailadres hebben verstuurd. Dit is om bedrijven niet steeds opnieuw uitnodigingen naar hetzelfde e-mailadres te laten sturen én om in opeenvolgende uitnodigingen alle bedrijven te kunnen noemen die hetzelfde bedrijf uitnodigingen. Zo kan een gebruiker nog meer aangespoord worden om lid te worden van CHAIN*els*. Het is bovendien zo dat als een uitnodiging geaccepteerd wordt, wat gebeurd door een in de uitnodigingsmail verstuurde link met daarin het 'InviteID' verwerkt te bezoeken, het uitgenodigde bedrijf automatisch gechaint wordt met alle bedrijven die ooit een uitnodiging hebben verstuurd naar het uitgenodigde bedrijf. Zo heeft het nieuwe bedrijf haar netwerk direct al een stuk in kaart gebracht zonder er zelf moeite voor te hoeven doen. Bedrijven kunnen hierna nog zeer eenvoudig 'ontchainen' met bedrijven waarmee ze toch geen contact willen hebben.

Bij het opslaan van een invite wordt er bijgehouden op welk tijdstip de uitnodiging is verstuurd. Dit is voor uitnodigingen die nooit geaccepteerd worden: deze kunnen dan na een bepaalde tijd verwijderd worden om zo databaseruimte vrij te maken. Dit wordt door een extern systeem periodiek gedaan. Als laatste wordt ook de taal van de uitnodiging bewaard om die taal weer te gebruiken op de website wanneer de gebruiker de uitnodiging accepteert.

4.6 BERICHTVERKEER

Naast het in kaart brengen van je bedrijfsnetwerk met behulp van chains en dergelijke draait CHAIN*els* verder vrijwel alleen maar om berichten! De berichten zijn in zes categorieën te onderscheiden: nieuwsbericht, aanbodbericht, vraagbericht, antwoordbericht, privébericht en infobericht. Code-technisch hebben al deze zes categorieën dezelfde basis. Vandaar dat er ook een basis klasse bestaat, een zogenoemde abstracte klasse: BaseMessage.

4.6.1 BASEMESSAGE

In deze klasse zijn een aantal standaard velden: fromCID, fromAID, attachment, en content. Het CID identificeert het bedrijf waarvan het bericht is, het AID de specifieke gebruiker die het bericht heeft geplaatst (voor intern gebruik), en de content de inhoud van het bericht. Het attachmentveld is op dit moment nog niet ondersteund, maar voor toekomstig gebruik. Het doel is namelijk om bij elk bericht de gebruiker de mogelijkheid te geven een bijlage toe te voegen, oftewel een plaatje, een portfolio, enzovoorts. Dit is echter nog niet gerealiseerd. Elk bericht heeft bovendien een eigen ID wat op een zelfde manier als eerder besproken wordt gegenereerd, waarmee meteen de tijd van de plaatsing van het bericht bepaald kan worden. Bovendien worden berichten, onafhankelijk van welk type het is, in lijsten opgeslagen. Bij lijsten kunnen gegevens slechts aan de voorkant (links) of aan het einde (rechts) worden toegevoegd, en in tegenstelling tot sets is de volgorde van belang. Zo kunnen dus de berichten altijd op volgorde worden gehouden, van nieuw (links) naar oud (rechts), waardoor het zeer eenvoudig is om 'de laatste zoveel' berichten op te vragen.

Ook een actie als 'de berichten sinds' is eenvoudig: een lijst met berichten kan dan in volgorde worden afgelopen, berichten toevoegend aan het resultaat, totdat er een bericht ouder is dan de opgevraagde tijd.

4.6.2 WallMessage

Per berichtsoort zijn er echter een aantal afwijkende opties. Zo zijn er eerst de nieuws-, aanbod- en vraagberichten. Deze zijn allemaal een vorm van een 'WallMessage': een bericht dat een bedrijf op haar profiel kan plaatsen. Een nieuwsbericht is hierbij de basis, en heeft in principe dezelfde velden als een BaseMessage.

Echter kan er ook worden aangegeven of een bericht 'privé' is of niet; niet te verwarren met de privéberichten die dadelijk besproken worden geeft deze instelling aan of het te plaatsen bericht door iedereen mag worden bekeken of alleen door de chains van het bedrijf. Ook kan een WallMessage 'aangeraden' worden. Een bedrijf geeft daarmee aan dat hij of zij een bericht met zijn of haar chains wil delen; een aangeraden bericht komt dan ook in het overzicht van de chains van de aanrader. Ook wordt er bij het nieuwsbericht zelf een teller gegeven van het aantal bedrijven dat het bericht aangeraden heeft, waarbij ook zichtbaar is welke bedrijven dit dan zijn. Hiervoor is er een 'recommendList' bij een WallMessage: een simpele set met CompanyIDs van de bedrijven die dit bericht hebben aangeraden. Omdat dit veld nooit apart opgevraagd hoeft te worden, en dus puur bij een bericht hoort, wordt het niet in een aparte set op databaseniveau opgeslagen, maar als een geserialiseerde array. Dit houdt in dat er een string wordt gemaakt van een array, welke opgeslagen kan worden, en weer omgezet naar een array bij het inladen. Hiervoor zijn vaste hulpfuncties gedefinieerd. Als een bericht ingeladen wordt, wordt dus ook deze array altijd ingeladen, wat ook gewenst is omdat de data die deze array met zich meebrengt altijd zichtbaar is. Manipulatie vindt zo ook in eerste instantie op PHP niveau plaats, met een 'doorsluismogelijkheid' naar de database: het opslaan van het bericht.

Ook kan er op een WallMessage gereageerd worden. Deze aparte 'Replies' zijn weer een categorie op zich. Net als bij een WallMessage kunnen deze, naast de standaard BaseMessage functionaliteiten, ook weer privé zijn en aangeraden worden. Privé betekent echter hier weer dat alleen de plaatser van het originele bericht de reactie kan lezen. De aanraadfunctie is vrijwel hetzelfde als bij een WallMessage. Een reply heeft een eigen ID en wordt apart opgeslagen, maar is wel altijd gelinkt aan een specifieke WallMessage. Dit is onder andere om de later te bespreken InfoMessages te faciliteren.

Naast een 'standaard' nieuwsbericht kan een WallMessage zoals eerder genoemd ook een vraag- of aanbodbericht zijn. Dit is van groot belang voor een aantal algoritmes (zie de paragraaf Geavanceerde algoritmen). Een vraag- of aanbodbericht heeft een extra veld: de keyArray. Deze array wordt op een vergelijkbare manier als de zojuist besproken recommendList opgeslagen. Echter bevat het in plaats van IDs sleutelwoorden ('labels') die door de gebruiker ingegeven zijn. Deze sleutelwoorden vormen de basis voor de algoritmes die bijvoorbeeld vraag en aanbod op elkaar afstemmen. Overigens zijn er voor de verschillende categorieën WallMessages geen aparte klassen; de categorie van een bericht wordt met één simpel veld (een enumeratie) aangegeven. Dit voor gebruiksgemak naar het websitesysteem (UI); er hoeft daar dan geen aparte afhandeling per berichtsoort te worden gemaakt. Dit is mogelijk omdat de berichten, afgezien van de keyArray, vrijwel compleet hetzelfde zijn opgebouwd.

4.6.3 PRIVATEMESSAGE

Daarnaast zijn er nog de privéberichten. Zoals het woord al zegt worden deze berichten nergens publiek 'geplaatst', maar puur van bedrijf naar bedrijf gestuurd. Het extra veld wat dus bij een zogenoemde PrivateMessage hoort is het 'toCID'. Bovendien is er nog een veld wat aangeeft of het privébericht gelezen is of niet (isRead). Dit veld is bij versturen 0 (ongelezen), en wordt automatisch op 1 (gelezen) gezet de eerste keer dat het bericht wordt opgehaald (en het veld dus nog 0 is). Ook privéberichten zijn weer objecten die apart worden opgeslagen, en dus een eigen ID hebben. Dit ID wordt bij het versturende bedrijf in de (geordende) lijst van verstuurde berichten gezet, en bij het ontvangende bedrijf in de lijst van ontvangen berichten.

Bij het versturen van een bericht wordt bovendien een mailtje naar de ontvanger verstuurd, vergelijkbaar met andere, eerder besproken notificatiemails. Kortom, via privéberichten kunnen bedrijven simpel gezegd traditionele conversaties met elkaar voeren, vergelijkbaar met een simpele chat.

4.6.4 INFOMESSAGE

Als laatste zijn er de InfoMessages. Dit zijn berichten die niet direct door de bedrijven zelf worden aangemaakt, maar wel volgen uit hun acties. Deze berichten zijn vergelijkbaar met de WallMessages in het feit dat ze op de overzichtspagina (homepage) van een bedrijf verschijnen.

Ze worden echter dus automatisch gegenereerd: zo komt er een InfoMessage op het profiel van een bedrijf wanneer dat bedrijf met iemand gechained heeft, op iemands bericht gereageerd heeft, een bericht aangeraden heeft, of een ander bedrijf als department of holding aangegeven heeft. Een InfoMessage heeft een aantal extra velden: toCID, hashPointer, en contentKey. Het toCID geeft 'de andere partij' aan in een InfoMessage (die er altijd is) en de contentKey geeft aan om wat voor soort bericht het gaat, zodat de opbouwing van het bericht via het multi-language systeem kan worden gedaan: het bericht wordt dus elke keer dat het opgevraagd wordt opnieuw 'opgebouwd' met behulp van de key en de beide CIDs, en dan in de taal van de gebruiker! De hashPointer heeft een speciaal doel: het wijst naar 'het object' waar de InfoMessage over gaat (indien van toepassing). Als het ene bedrijf bijvoorbeeld op het bericht van een ander bedrijf reageert, dan wijst de hashPointer naar het bericht waar dat om gaat. Dit is voor visuele maar ook voor interne doeleinden: bij het object waar de hashPointer naar wijst wordt namelijk een set bijgehouden met InfoMessages die bij dat object horen. Dit is om verwijderen van dat object te faciliteren: alle InfoMessages kunnen dan ook meteen verwijderd worden.

Verder wordt bij het aanmaken van een InfoMessage ook weer een e-mailnotificatie naar alle gebruikers van het met toCID geïdentificeerde bedrijf gestuurd, vergelijkbaar met het eerder besprokene.

Een laatste opmerking is dat alle berichten in principe 'oneindig' opgeslagen worden. Alles is wel door de gebruiker te verwijderen, maar als dat niet gebeurt, dan wordt het bericht gewoon opgeslagen. Hieruit volgen een aantal problemen, welke in Appendix A, onderdeel Database, uitgebreid besproken worden. Al deze berichten zullen namelijk het grootste aandeel in het databasegebruik van ons netwerk vormen, en zijn dus ook in dat opzicht van groot belang.

4.7 Beveiliging

De basisprincipes van beveiliging die we binnen ons netwerk hebben toegepast worden besproken in Appendix A, onderdeel Encryptie en andere beveiliging. Er zijn daarnaast echter nog een aantal specifieke zaken met betrekking tot inloggen en andere beveiliging die van belang zijn.

Zo kan het voorkomen dat een gebruiker zijn wachtwoord is vergeten: wat dan? In ons systeem (en in vele andere) kan de gebruiker aangeven dat hij of zij het wachtwoord behorende bij zijn of haar e-mailadres kwijt is. Als de gebruiker dit aangeeft zal hij of zij een mailtje krijgen met daarin een link om het wachtwoord terug te krijgen. In deze link is, net als bij de eerder besproken activatiemails, een unieke, willekeurig gegenereerde code verwerkt. Dit is om te voorkomen dat anderen dan de gebruiker zelf het wachtwoord van de gebruiker kunnen wijzigen.

Immers, de enige manier waarop iemand achter de code kan komen is door het mailtje met de code erin te krijgen, en alleen de gebruiker krijgt dat mailtje. Als de gebruiker via dat mailtje de link bezoekt zal er voor hem een nieuw wachtwoord worden gegenereerd en hem of haar via de mail toegestuurd worden. Zodoende kan de gebruiker met dat wachtwoord weer inloggen, en indien gewenst het wachtwoord wijzigen naar een persoonlijk wachtwoord.

Verder hebben wij Bcrypt (zie Appendix A, onderdeel Encryptie) op het moment op acht ronden ingesteld, waardoor de encryptie van een wachtwoord in gemiddeld een derde seconde plaats vindt. Dit vonden wij een optimaal gemiddelde tussen veiligheid en snelheid op dit moment. Bcrypt gebruiken we alleen voor de wachtwoorden; andere encryptie wordt toegepast op e-mailadressen (bij bijvoorbeeld invites) en de willekeurige codes zoals eerder besproken (bij bijvoorbeeld activatiemails), waarbij veiligheid door hashing van minder zwaar belang is.

De encryptie die we hierbij gebruiken is het zeer efficiënte MD5 algoritme. Dit is weliswaar iets minder veilig, maar wel een heel stuk efficiënter. Bovendien heeft dit algoritme als grote voordeel dat het 'veilige' links genereert: in URLs (weblinks) mogen namelijk niet zomaar alle tekens voorkomen[7]. MD5 geeft gegarandeerd altijd een output die 'URL safe' is, en kan daarom gebruikt worden voor de 'encryptie' van willekeurige codes in links. Bovendien voegt het nog een extra laag van willekeurigheid aan de al willekeurig gegenereerde codes toe.

Verder is het nog van belang dat de communicatie tussen de webserver en de database server volledig intern gebeurt, en dus niet afgeluisterd kan worden. Ook communicatie van de database servers met elkaar gebeurt intern in het datacentrum met behulp van zogenaamd 'statische netwerken[8], en kan dus niet zomaar worden afgetapt of iets dergelijks. Verder wordt er voor de client-server communicatie indien nodig van een HTTPS verbinding gebruik gemaakt, zoals verder besproken in Appendix A.

---

[7] Brian Wilson, *URL Encoding*, http://blooberry.com/indexdot/html/topics/urlencoding.htm (26-11-11)

[8] LinodeWiki, *Configure Static IPs*, http://www.linode.com/wiki/index.php/Configure_Static_IPs#Static_IP_Configuration (26-11-11)

Als laatste zijn de cookies nog een beveiligingsrisico, in het bijzonder de cookie die het in de Appendix A besproken sessieID bevat. Om de risico's te minimaliseren wordt deze cookie alleen over een HTTPS verbinding ingesteld, en het gebruik ervan beperkt tot het chainels.com domein.

Dit is echter wel afhankelijk van het systeem van de gebruiker, en is dus niet volledig veilig. Vandaar ook dat de in Appendix A besproken regeneratie gebruikt wordt.

4.8 STATISTIEKEN

Op een netwerk als CHAIN*els* zijn ook statistieken van groot belang. Niet alleen voor onszelf qua ontwikkeling, maar ook mogelijk voor de bedrijven zelf, die deze statistieken bijvoorbeeld zouden kunnen kopen. Om deze statistieken te verzamelen maken wij gebruik van de in Appendix A, onderdeel Database, genoemde GeoIP module, om zo het IP-adres van de gebruiker aan een locatie te kunnen koppelen. Het IP-adres van de gebruiker verkrijgen is echter niet altijd eenvoudig! Dit komt bijvoorbeeld door het gebruik van meerdere servers met behulp van de in Appendix A, onderdeel Server, besproken loadbalancer. De gebruikers 'bezoeken' namelijk in feite deze load balancer, die hun verzoek doorstuurt naar één van de (op dit moment twee) servers.

Het IP-adres dat deze servers zien is dan echter dat van de loadbalancer in plaats van dat van de gebruiker! Bij standaard HTTP verbindingen geeft de loadbalancer weliswaar een extra veld in de HTTP header mee met het IP-adres van de client erin[9], maar bij een HTTPS verbinding kan dat niet! Vandaar dat er een speciale procedure moet worden gevolgd om het IP-adres van de client te verkrijgen in sommige gevallen: als de client de site eerst in HTTPS benadert wordt hij of zij eerst naar HTTP doorgestuurd, waarna het IP adres van de client in een cookie wordt opgeslagen, en de client daarna weer naar HTTPS wordt teruggestuurd. Zo is het IP-adres voor verder gebruik in deze gebruikerssessie vastgelegd, en hoeft het niet steeds opnieuw te worden opgevraagd. Als een gebruiker eerst via HTTP binnenkomt, is het nog makkelijker, omdat er dan niet heen- en weer gestuurd hoeft te worden (van HTTPS naar HTTP en weer terug: we zijn immers al in 'HTTP modus').

De locatiegegevens, het tijdstip, en het mogelijke CID van de gebruiker worden bij elk pagina bezoek aan de set van bezoeken aan die pagina toegevoegd. Elke pagina op de website heeft een eigen, unieke key, soms afhankelijk van extra variabelen (zoals het CID bij een info-pagina). Aan de hand van deze gegevens kan dus per pagina worden bekeken hoeveel bezoekers er zijn geweest op welk tijdstip, waarvandaan, en mogelijk van welk bedrijf. Dit biedt ons en bedrijven zelf de mogelijkheid om gedetailleerd inzicht te krijgen in het gebruik van het netwerk, wat zeer belangrijk is voor de algemene ontwikkeling ervan.

4.9 GEAVANCEERDE ALGORITMEN

Met alleen informatiepagina's en berichten van bedrijven is het netwerk nog lang niet af. Deze gegevens moeten immers voor een gebruiker gemakkelijk te vinden en te bekijken zijn. Een eerste hulpmiddel hierbij is 'de zoekfunctie'. In het websitesysteem is dit een algemene zoekbalk bovenaan de pagina. Op het moment is deze balk puur bedoeld om bedrijven te kunnen zoeken. Dit is echter niet alleen maar op naam gebaseerd; een vereiste is ook dat een query als 'Delft Accountants' of zelfs 'Websites Noord-Holland' de gewenste resultaten opleveren.

---

[9] Wikipedia, *X-Forwarded-For*, http://en.wikipedia.org/wiki/X-Forwarded-For (26-11-11)

4.9.1 ZOEKEN

In de klasse Search zijn de algoritmen hiervoor geïmplementeerd. Op een bepaalde input worden op dit moment alle bedrijven die in het systeem staan afgegaan. Dit lijkt misschien inefficiënt, maar per bedrijf is slechts een klein beetje informatie nodig, en de database is afgestemd op dit systeem. Als de database echter explosief groeit, zal hier mogelijk een betere oplossing voor gevonden moeten worden. Op dit moment wordt echter per bedrijf opgevraagd wat de naam ervan is, de vestigingslocatie (plaats-provincie-land), de industrie (sector) en de status. De status is van belang om 'niet complete' bedrijven uit de zoekresultaten te filteren (zie de paragraaf Bedrijven en gebruikers); de andere zojuist genoemde factoren zijn de daadwerkelijke parameters.

Voordat elk bedrijf in de database wordt afgegaan, wordt de input van de gebruiker 'gesplitst op spaties'. In andere woorden: elk woord dat de gebruiker invoert wordt apart genomen, dus bijvoorbeeld 'Delft ICT' wordt array(Delft,ICT). Dit is van belang om op de zojuist genoemde manier te kunnen zoeken, namelijk apart op industrie en vestigingsplaats en dergelijke. Voor namen met spaties, vooral als het er veel zijn, is dit echter weer iets nadeliger.

Voor dit probleem hebben we echter op dit moment nog geen oplossing gevonden; het probleem speelt namelijk ook niet bij een groot deel van de bedrijven, omdat bedrijfsnamen vaak uit één woord bestaan. Het zal echter wel opgelost moeten worden, aangezien een bijzondere naam als 'Verseput Bouw- en Milieutechniek BV' nu niet goed op te zoeken is. Een mogelijke snelle oplossing als namen met de gehele input vergelijken in plaats van per woord is echter ook niet goed, aangezien queries als 'TofSoft Delft' er dan weer doorheen vallen. Het is dus afwegen, en op het moment hebben we voor de oplossing gekozen die er op dit moment is, met doorontwikkeling in het verschiet.

Om met het algoritme verder te gaan wordt er vervolgens per bedrijf in het systeem de zojuist genoemde parameters opgevraagd, en per apart woord in de invoer van de gebruiker vergeleken. Voor deze vergelijking maken we gebruik van een standaard PHP functie, namelijk 'similar text'[10]. Dit algoritme is gebaseerd op werk van J.J. Oliver in het boek 'Decision Graphs - An Extension of Decision Trees' uit 1993[11]. De precieze werking van dit algoritme is niet van groot belang; het gaat er voor ons om dat het werkt. Dit algoritme geeft namelijk voor twee invoerstrings een percentage als output dat uitdrukt hoeveel deze strings op elkaar lijken. Het is officieel van $O(N^3)$, waarbij N de lengte is van het langste invoerwoord, maar binnen PHP is het flink geoptimaliseerd voor kleine invoerwoorden, waardoor er veel snelheidswinst wordt geboekt[12].

---

[10] PHP Manual, *similar_text*, http://php.net/manual/en/function.similar-text.php (28-11-11)

[11] Jonathan J. Oliver, *Decision Graphs - An Extension of Decision Trees*, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.1476 (28-11-11)

[12] The Drawingboard, *PHP's similar text function*, http://blog.procontentanddesign.com/articles/single/215_PHP_s_similar_text_function.htm (28-11-11)

Proefondervindelijk is gebleken dat dit algoritme ook echt het gewenste resultaat geeft, in tegenstelling tot het eerder door ons gebruikte Levenshtein algoritme[13]. Ook het Levenshtein algoritme is namelijk bedoeld om een vergelijking tussen twee invoerwoorden te maken. Echter is dit algoritme meer gebaseerd op het aantal acties als toevoegen, verwijderen en vervangen dat nodig is om van het ene woord het andere te maken.

Het grote voordeel van dit algoritme is dat het slechts van O(N*M) is, met N en M de lengte van de invoerwoorden, maar de output is slechts de zogenaamde 'Levenshtein Distance'[14], een absoluut getal. De output van dit algoritme bleek proefondervindelijk minder geschikt voor onze doeleinden. Bovendien blijkt uit een aantal tests dat voor kortere invoer het similar text algoritme niet alleen beter is maar ook nog eens sneller! Gezien deze theoretische en praktische factoren is er dus uiteindelijk voor gebruik van het 'similar text' algoritme gekozen, vooral gebaseerd op de proefondervindelijke resultaten daarvan. Overigens worden de woorden die dit algoritme ingaan in onze implementatie altijd in kleine letters doorgegeven om verschillen in hoofdlettergebruik niet te laten meetellen.

Met het zojuist besproken 'similar text' algoritme wordt het volgende in pseudo-code beschreven algoritme door ons losgelaten op de invoer per bedrijf in het systeem:

```
totaalscore = 0
foreach( woord in input )
        score1 = similar_text( woord, bedrijf[naam] )
        score2 = similar_text( woord, bedrijf[industrie] )
        score3 = similar_text( woord, bedrijf[stad] )
        score4 = similar_text( woord, bedrijf[provincie] )
        score5 = similar_text( woord, bedrijf[staat] )
        subscore = max( score1, score2, score3, score4, score5 )
        if( subscore >= 20 ) then totaalscore += subscore
end
resultaat = (integer)(totaalscore / aantal woorden in input)
if( resultaat >= 30 ) then accepteer bedrijf, else verwerp bedrijf
```

De resultaten worden hierna nog gesorteerd: de hoogste scores komen bovenaan de uitvoer. De complexiteit van het algoritme is hetzelfde als die van similar_text, keer een factor vijf per invoerwoord, plus de complexiteit van het achteraf sorteren; dit gebeurt met een quicksort algoritme en is dus gemiddeld van O(n*log n)[15]. Zo komt de totale complexiteit op O((5M*N^3)B + r*log r), waarbij M het aantal woorden in de invoer is, N de lengte van het langste invoerwoord, B het totale aantal bedrijven in de database, en r het aantal geaccepteerde bedrijven in het uiteindelijke resultaat. Praktisch blijkt dat vrijwel alleen B een rol speelt in de uiteindelijke duur van het algoritme: de similar_text functie en de sortering op zichzelf werken proefondervindelijk in vrijwel verwaarloosbare tijd.

---

[13] PHP Manual, *levenshtein*, http://www.php.net/manual/en/function.levenshtein.php (28-11-11)

[14] Michael Gilleland, *Levenshtein Distance*, http://www.merriampark.com/ld.htm (28-11-11)

[15] H.W. Lang, *Quicksort*, http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/quick/quicken.htm (28-11-11)

B kan echter stijgen tot miljoenen bedrijven, en miljoen keer een klein beetje tijd is alsnog veel tijd. Vandaar dat er nog naar een oplossing moet worden gezocht waarbij niet elke keer alles voor elk bedrijf hoeft te worden berekend.

Dit is echter op dit moment niet geïmplementeerd noch uitgedacht, omdat het op de huidige manier proefondervindelijk nog steeds zeer snel werkt tot een databaseomvang van minimaal duizend bedrijven. Als laatste opmerking blijkt inderdaad dat queries als 'Delft', 'Consultancy Amsterdam', et cetera inderdaad het gewenste resultaat opleveren, met als enige 'drawback' de zojuist besproken issue van bedrijfsnamen met veel losse woorden erin.

4.9.1 RECOMMENDED CHAINS

Naast dit zoekalgoritme 'voor en door de gebruiker' zijn er meer algoritmes die principieel 'zoeken'. Zo is er ook een algoritme dat 'chains voorstelt' aan de gebruiker. In andere woorden: het geeft suggesties aan bedrijven om een verbinding aan te gaan met een ander bedrijf. Op dit moment werkt dit algoritme als volgt: elke huidige chain van het bedrijf wordt afgegaan, en alle chains daarvan worden in een set bewaard. Per uniek bedrijf in de set wordt dan een 'score' bijgehouden van hoeveel van jouw chains met dat bedrijf zijn gechaint. De redenering is namelijk dat als heel veel van jouw verbindingen met een bepaald bedrijf verbonden zijn deze verbinding hoogstwaarschijnlijk ook voor jou interessant is.

De pseudocode is als volgt:

```
subresultaat = lege array
foreach( chain in eigenChains)
        subresultaat += getCompanyChainsIDs( chain )
end
resultaat = array_diff( subresultaat, eigenChainsEnMezelf )
resultaat = array_diff( subresultaat, alDoorMijUitgenodigdeBedrijven )
eindresultaat = lege array
foreach( element in resultaat )
        index = vindIDinResultaat( element, eindresultaat )
        if( index == -1 ) then eindresultaat += element   //met score=1
        else eindresultaat[index].score++
end
```

De resultaten worden hierna nog op dezelfde manier als bij het zoekalgoritme gesorteerd: de hoogste scores komen bovenaan de uitvoer. Daarnaast zal de gebruiker standaard alleen 'de vier beste' resultaten zien, maar ook alle resultaten kunnen bekeken worden indien gewenst. De complexiteit van dit algoritme is van O(N*M), waarbij N het aantal eigen chains is en M daarbij het aantal chains per eigen chain, plus weer de complexiteit van de sortering met quick sort.

Zo komen we uit op O(R+r*log r), met r het aantal elementen in het resultaat, wat overigens niet gelijk is aan N*M omdat bedrijven in de te sorteren array allemaal uniek zijn (in andere woorden: slechts één keer in de lijst voorkomen). Zo worden er dus kortom op basis van de chains van je eigen chains suggesties voor de uitbreiding van jouw netwerk aangedragen. Later willen we hier meer factoren inbouwen in de vorm van een uitgebreidere 'score' per suggestie: niet meer alleen simpelweg 'hoeveel vrienden zijn met hem bevriend', maar ook onder andere de geografische afstand en sector laten meewegen.

4.9.2 VRAAG EN AANBOD

Vervolgens zijn er de algoritmes die met vraag en aanbod te maken hebben. Zoals besproken in de oriëntatie in Appendix A werken deze met 'labels' of 'keys' om te identificeren 'waar het over gaat'. Een eerste vereiste hierbij is om suggesties te krijgen voor deze labels.

Dit gebeurt simpelweg door de set van alle bestaande keys op te vragen, waarbij al is opgeslagen hoe vaak elke key gebruikt is. Aangezien het bij deze suggesties slechts om een mogelijkheid tot aanvullen gaat en niet 'zoeken', in andere woorden bij 'web' het label 'websites' suggereren en dergelijke, zijn geavanceerde vergelijkingsalgoritmen niet nodig. Het algoritme is dus al snel af:

```
labels = getKeysWithTheirValue( 'gen:keys:' + input + '*')
resultaat = lege array
foreach( label in labels )
        resultaat += (label, label.waarde)
end
```

Hierna wordt weer een quicksort algoritme toegepast om de labels met de hoogste waardes bovenaan te krijgen. Verder is op te merken dat 'gen:keys' de interne identificatie is van de labels (prefix), en dat '*' aangeeft dat er alles mag staan: oftewel, achter de input mag nog van alles komen; auto-complete dus.

Naast deze aanvullingen voor alle labels moeten vraag en aanbod op elkaar afgestemd kunnen worden. Een van onze doelstellingen is immers om bedrijven nieuwe mogelijkheden op dit gebied te laten ontdekken; vandaar dat wij hier algoritmes voor hebben geïmplementeerd. Gezien het feit dat het algoritme om 'vraagmogelijkheden' te ontdekken vrijwel hetzelfde is als het algoritme om 'aanbodmogelijkheden' te ontdekken zullen wij hier alleen het eerstgenoemde bespreken. Verder is het van belang dat er 'twee soorten' vraag en aanbod zijn: statisch en dynamisch. Statisch vraag en aanbod hoort bij een bedrijfsprofiel, en wordt zoals de naam zegt 'vast' ingesteld. Dynamisch vraag en aanbod hoort bij een bericht (WallMessage) dat een bedrijf plaatst (zie de paragraaf Berichtverkeer), welke ook weer verwijderd kan worden en dergelijke.

De pseudocode is als volgt:

```
mijnStatischeVraag = verkrijgStatischeVraagLabels( myCID )
mijnBedrijfsBerichten = verkrijgWallMessages( myCID )
mijnDynamischeVraag = lege array
foreach( bericht in mijnBedrijfsBerichten )
        if( bericht.categorie = vraag ) then mijnDynamischeVraag += bericht.label
end
labels = mijnStatischeVraag + mijnDynamischeVraag
resultaat = lege array
foreach( label in labels )
        berichten = verkrijgAlleBerichtenMetLabel( label )
        foreach( bericht in berichten )
                if( bericht.van != myCID && bericht.categorie == aanbod &&
                        !resultaat.bevat(label,bericht) ) then resultaat += (label,bericht)
        end
end
```

Hierna wordt de array met gevonden resultaten nog willekeurig geordend ('shuffle') en naar wens de eerste vier of alle resultaten teruggegeven. Het sorteren gebeurt achteraf zodat alle berichten per key evenveel 'kans krijgen' om in het resultaat terug te komen, en er dus geen 'bias' is per key of iets dergelijks.

De enige wijziging in het algoritme voor de aanbodmogelijkheden is dat in plaats van de eigen vraagberichten en de statische vraag de eigen aanbodberichten en het statische aanbod wordt opgehaald, en dan met een vraagbericht wordt gematcht in plaats van een aanbodbericht. Als laatste valt op te merken dat er alleen met dynamisch vraag en aanbod wordt gematcht in het tweede deel van de functie, en niet met statisch vraag en aanbod. Dit is bewust gedaan om te zorgen dat het statische aanbod van twee bedrijven niet direct met elkaar linkt en dus 'altijd' in de resultaten voorkomt. Het is echter wel zo dat het statische vraag en aanbod wordt meegenomen in het algoritme door het met al het andere dynamische vraag en aanbod te vergelijken, zodat er altijd ergens een match mee zal zijn, en geen enkel vraag of aanbod 'verloren zal gaan'. Een bedrijf dat namelijk bijvoorbeeld een bepaalde vraag plaatst dat matcht met een statisch aanbod van een ander bedrijf zal wel geen vraagsuggestie hiervoor krijgen, maar het andere bedrijf wel een aanbodsuggestie. Of deze manier van matchen optimaal is zal nog moeten blijken, maar op het moment is het dus zo geïmplementeerd.

4.9.3 DASHBOARD

Een ander 'geavanceerd algoritme' is de generatie van het zogenaamde 'Dashboard' van een bedrijf. Het Dashboard is een overzicht waar alle relevante WallMessages en InfoMessages (zie de paragraaf Berichtverkeer) op te bekijken zijn. Op dit moment houdt 'relevant' nog in dat het van óf een chain, óf een departement/holding van je bedrijf af komt. Later willen wij deze definitie van relevant gaan uitbreiden; echter is het aangeven van chains al een grote indicatie hiervoor: dit is immers al een indicatie dat het bedrijf voor jou interessant is, en dus zullen de berichten van dat bedrijf waarschijnlijk ook voor jou interessant zijn. Op dit moment werkt het zo dat per mogelijke chain, holding en departement 'alle' berichten opgehaald worden met de daarvoor relevantie functies. Dit kan altijd op twee manieren: vanaf een bepaald moment, of met een bepaald aantal. Op de eerste manier worden bij elk bedrijf alle berichten vanaf dat aangegeven moment opgehaald; op de tweede manier worden bij elk bedrijf naar ratio het aantal berichten opgehaald. Met andere woorden: een overzicht van tien berichten met tien bedrijven betekent één bericht per bedrijf. Als er te veel bedrijven voor berichten zijn wordt er alsnog minimaal één bericht per bedrijf opgehaald; andersom is meer natuurlijk geen probleem, waarbij er ook altijd naar boven wordt afgerond. Dit wordt gerealiseerd door per bedrijf een *ceiling(wantedAmount / totalAmount)* aantal berichten te pakken[16].

Behalve deze WallMessages spelen echter ook de InfoMessages een rol. Hierbij is gekozen om dit altijd 'per tijd' te doen. Met andere woorden: eerst worden alle WallMessages op de gekozen manier opgehaald, en daarna alle InfoMessages tot de tijd van het oudste WallMessage in de zojuist opgehaalde set. Daarna worden alle berichten, onafhankelijk van het type, logischerwijs op tijd gesorteerd, met de nieuwste bovenaan.

---

[16] Wikipedia, *Floor and ceiling functions*, http://en.wikipedia.org/wiki/Floor_and_ceiling_functions (28-11-11)

Een verbetering waar we nog op dit moment mee bezig zijn is om de InfoMessages te 'groeperen'. Immers is het niet optimaal als tussen twee berichten bijvoorbeeld tien individuele berichten staan in de trant van 'bedrijf X is gechaint met bedrijf Y'. Deze berichten willen we groeperen in één InfoMessage in de trant van 'N bedrijven hebben nieuwe chains aangemaakt', waarbij er dan in een tooltip of pop-up of iets dergelijks meer informatie kan worden bekeken (bijvoorbeeld dan wel de individuele berichten). Een dergelijke groepering zal in de generatie van het Dashboard overzicht moeten gebeuren.

4.9.4 NOTIFICATIES

Een laatste functie die nog als 'geavanceerd algoritme' kan worden aangeduid is het algoritme met betrekking tot de zogenoemde 'Notificaties'. Notificaties zijn eigenlijk niets meer dan InfoMessages die belangrijk zijn voor het bedrijf, en dus in een extra lijst worden bijgehouden. Een notificatie kan bovendien ongelezen/gelezen zijn, waarbij ongelezen notificaties tot extra meldingen leiden binnen de website (net als bij ongelezen privéberichten, zie hiervoor weer de paragraaf Berichtverkeer). Een notificatie is dus niets meer dan een verwijzing naar een InfoMessage en een veld dat aangeeft of de notificatie gelezen is of niet. Om dit gelezen/ongelezen automatisch te laten plaatsvinden is er een wat uitgebreidere code nodig. Bovendien willen we de notificaties niet een oneindige lijst laten zijn, maar alleen een overzicht van de laatste zoveel gelezen berichten, 'gecombineerd' met de nog ongelezen berichten indien van toepassing.

De pseudocode voor het verkrijgen van de notificaties wordt dan als volgt:

```
notificaties = verkrijgNotificatiesVoorMijnBedrijf( CID )
resultaat = lege array
teLezen = 0
foreach( notificatie in notificaties )
        if( !notificatie.gelezen ) then teLezen++
        notificatie.gelezen = true
        resultaat += notificatie.infoMessage
end
if( teLezen < 5 ) then teLezen = 5
trimBedrijfsNotificatiesTotGrootte( CID, teLezen )
```

Zo krijgt een gebruiker altijd minimaal vijf notificaties, en meer indien nodig (meer ongelezen notificaties).

4.10 SOCIAL MEDIA

Een 'grote feature' binnen dit netwerk is de mogelijkheid om berichten die je op CHAIN*els* plaatst door te linken naar je bestaande sociale media, waar wij op dit moment Twitter, LinkedIn en Facebook onder verstaan. Behalve het doorlinken van berichten kunnen ook, alleen via Twitter en LinkedIn, uitnodigingen via deze media worden verstuurd naar bedrijven om lid te worden van CHAIN*els* (zie paragraaf Chains en Invites). Deze functionaliteit is echter niet vanzelfsprekend! Er zijn immers zogenoemde APIs ('programming interfaces') nodig om onze software met de software van Twitter en anderen te laten werken. Over het algemeen werkt deze uitwisseling per genoemd netwerk vrijwel hetzelfde, en we zullen het in deze paragraaf dan ook generaliseren.

Voor elk netwerk is namelijk een API beschikbaar, vaak in allerlei programmeertalen, waaronder voornamelijk PHP, wat overigens achteraf nog een extra stimulans is om voor PHP te kiezen, maar dat terzijde. Per API dient de gebruiker ervan een soort account aan te maken waar dan een unieke key bij te verkrijgen is die gelinkt wordt aan de benodigde instellingen.

Dit gaat volgens het gestandaardiseerde OAuth protocol[17]. Dit werkt heel basaal als volgt: de gebruiker vraagt ons systeem om verbinding te maken met bijvoorbeeld Twitter. Vervolgens zal ons systeem de gebruiker naar Twitter doorlinken met als referentie de unieke key van onze site zoals bekend bij Twitter, en bovendien een tijdelijke key ter identificatie van de gebruiker. De gebruiker kan vervolgens op Twitter inloggen om onze site toegang tot zijn of haar account toe te staan. Wanneer dit gebeurd zal Twitter naar ons systeem teruglinken met als identicatie voor de gebruiker (immers weten we anders niet wie er teruggelinkt wordt) dezelfde tijdelijke key, en als resultaat de unieke key die bij de gebruiker in combinatie met onze applicatie hoort. Met deze unieke key kunnen wij als CHAIN*els* vervolgens zonder verdere vereisten acties verrichten in naam van de gebruiker op Twitter. Dit systeem werkt dus vrijwel hetzelfde op LinkedIn en Facebook.

Per bedrijf kan zo per sociaal netwerk de unieke key worden opgeslagen, waardoor we in combinatie met onze eigen 'applicatiekey' allerlei acties voor de gebruiker kunnen uitvoeren.

Zo kunnen we een bericht plaatsen voor de gebruiker (oftewel het 'doorsturen' van geplaatste berichten op CHAIN*els*) en privéberichten via het betreffende sociale netwerk aan door de gebruiker aan te geven contacten op dat sociale netwerk versturen (oftewel het uitnodigen van nieuwe bedrijven). Dit is een waardevolle feature, aangezien er zo gebruik gemaakt wordt van de al bestaande netwerken om CHAIN*els* uit te breiden, en dus te 'verbeteren'. Als laatste opmerking doet dit gegeneraliseerde verhaal te kort aan de moeite van de implementatie van de bovenstaande functionaliteiten, gezien de APIs per sociaal netwerk toch ook veel verschillen, en er dus voor elk netwerk een aparte, bijzondere implementatie voor de link met ons systeem moet worden gemaakt.

[17] OAuth, *Introduction*, http://oauth.net/about/ (28-11-11)

5. TESTEN

Om het systeem te controleren hebben we zowel Unit tests als User tests afgenomen. In de eerste fase waren het voornamelijk Unit tests. In een later stadium was de User Interface ontwikkeld waardoor User tests mogelijk werden, en dus ook zijn uitgevoerd.

5.1 UNIT TESTS

In het begin stadium zijn er veel Unit tests opgesteld. Iedere aparte klasse kreeg ook een Unit test klasse. In de hoofdklasse werden alle Unit tests achter elkaar doorlopen, zodat alle tests werden uitgevoerd. In figuur 11 is een voorbeeld van het resultaat van een dergelijke 'Test Suite' te zien.

Figuur 11 - Een voorbeeld unit-test resultaat

In eerste instantie leken de Unit tests onnodig veel werk, maar in een latere fase werd de meerwaarde van de tests duidelijk. Indien we iets aanpasten in het systeem zagen we door middel van de Unit tests waar we iets vergeten waren mee te veranderen. Kleine bugs die in eerste instantie weinig effect zouden hebben werden er bijtijds uitgehaald, wat ons tijd bespaarde in het vervolgproces. De Unit tests waren ook een goede oplossing voor de in het begin nog ontbrekende User Interface.

5.2 USER TESTS

Er hebben verschillende user tests plaatsgevonden. Het doel van deze testen was om zoveel mogelijk feedback te verzamelen waardoor we het netwerk snel konden verbeteren.

Ten eerste hebben we met het team zelf getest. Hierbij gebruikten we zelf alle functionaliteiten. Er was een test netwerk opgezet van circa vijf bedrijven om een real life situatie te simuleren. Echter is het nadeel van zelf testen dat gebruiksvriendelijkheid lastig waar te nemen is: voor je gevoel werkt bijna alles logisch, wat vooral te wijten is aan het feit dat je al iedere dag met het concept bezig bent en op den duur het systeem als `normaal` beschouwt. Kritisch zijn is daardoor vaak lastig. Een mogelijke oplossing hiervoor is het testen van andere functionaliteiten dan die je zelf hebt geïmplementeerd. Daarom werd waar mogelijk elkaars werk gecontroleerd.

Ten tweede is er een Closed Beta opgezet. Met een geselecteerde set bedrijven (circa 25) is er een klein netwerk opgezet. Op het netwerk zelf was een "Beta feedback" account aangemaakt, zodat feedback naar ons gestuurd kon worden. Bedrijven werden hierdoor gedwongen CHAIN*els* extra te gebruiken. Er was heel veel feedback verzameld en iedere dag verbeterden we het systeem, zodat dubbele feedback en irritatie met betrekking tot bugs geminimaliseerd werd. Er zijn interviews afgenomen met gebruikers en er is meegekeken met gebruikers. Dit laatste punt leverde nog veel informatie op, omdat bleek dat acties soms onvindbaar waren. Ter afsluiting is er een enquête opgesteld, welke inclusief resultaten in Appendix F is opgenomen.

Ten derde hebben we een Open Beta opgezet, waarbij alle functionaliteiten voor de Release waren toegevoegd. Op aanvraag ontving je een uitnodiging om aan de Beta deel te nemen. Een groot verschil was dat CHAIN*els* nu ook naar buiten werd gepresenteerd als "The business network" en dat het netwerk niet echt afgeschermd hoefde te worden.

Na de officiële lancering in Delft zijn we verder gegaan met testen, en zolang het systeem nog niet als 'af' wordt beschouwd zal er dan ook een Open Beta blijven draaien.

6. PROCES EN EVALUATIE

In een groot project als CHAIN*els* ga je een leerproces door waar een hoger niveau van de ICT wordt verwacht dan bij voorgaande projecten. In het proces blijkt samenwerken de sleutel naar succes.

6.1 HET TEAM

De samenwerking was initieel in tweeën opgedeeld: business/marketing en ICT. Wekelijks was er een vergadering om de actiepunten van de voorgaande week te bespreken. Ook het concept CHAIN*els* werd iedere week verder uitgewerkt.

Het ICT-team kreeg iedere week actiepunten mee om te kijken naar de haalbaarheid van verschillende functionaliteiten. Het business/marketing-team deed verder onderzoek naar het concept in het algemeen.

Met het volgende team zijn wij in mei 2011 officieel gestart:

Naam	**Werk en/of opleiding**	**Taak**
Dr. Ir. Erik Blokhuis	• Onderzoeker aan de TU Eindhoven	Business Marketing
Dr. Ir. Shahid Suddle	• Eigenaar SSCM BV • Oprichter Suddle Holding • Oprichter Inosar • Part-time Assistent Professor Safety Integrated Design aan de TU Delft	Business Marketing
Vincent Koeman	• Student Technische Informatica aan de TU Delft • Mede-oprichter TofSoft	ICT
Erwin Buckers	• Student Technische Informatica aan de TU Delft • Mede-oprichter TofSoft	ICT
Remi Baar	• Student Informatica en Economie aan de Universiteit Leiden • Oprichter Indextra	User Interface

Start team CHAINels met verleden en taak (mei 2011)

Na een start met een al relatief klein team voor de grootte van het project werd het team helaas nog kleiner toen Erik eruit stapte vanwege persoonlijke redenen. Dit was een grote teleurstelling voor het team, maar na goed overleg zijn wij met ons vieren doorgegaan, nog zoekende naar versterking aan de kant van business en marketing.

In de zomervakantie (juli en augustus) was het lastiger samenwerken, omdat Remi een aantal weken op vakantie was. We vergaderden nog wel een keer per week zonder Remi. Dit was heel belangrijk om de stemming goed te houden en het persoonlijk contact te behouden.

Tegen het einde van de vakantie was er een werkend systeem, alleen nog geen User Interface. Dit zorgde voor de nodigde frustratie.: het was onmogelijk om de geplande User tests uit te voeren. Na een goed gesprek met het team werden er planningen gemaakt door ons voor het Design. In eerste instantie werden de deadlines nog niet gehaald, waardoor we nogmaals het punt deadlines en planning aan de orde hebben gebracht. Wij gaven aan: “Zeg liever dat je het niet kan of haalt dan dat je niks zegt en het niet op tijd af is”.

Sinds eind oktober 2011 gaat het al een stuk beter, en is er ook een hechtere samenwerking ontstaan binnen de ICT, wat het resultaat ten goede komt.

Dit is ook dankzij de versterking van Willem Buijs in het team. De toevoeging van Willem in het team heeft ervoor gezorgd dat het balans weer beter was en dat ook een functie werd opgepakt die ontbrak: "Design en Marketing". De media aandacht zorgde tevens voor een extra motivatie binnen het team; zie Appendix D voor meer details.

Naam	**Werk en/of opleiding**	**Taak**
Dr. Ir. Shahid Suddle	(zie voorgaande tabel)	Business Marketing
Vincent Koeman	(zie voorgaande tabel)	ICT
Erwin Buckers	(zie voorgaande tabel)	ICT
Remi Baar	(zie voorgaande tabel)	User Interface
Willem Buijs	Student Industrieel Ontwerpen (Master)	Design

Huidige CHAINels team (november 2011)

Het sterke van het huidige team is dat iedereen heel goed met kritiek kan omgaan en dat kwesties altijd bespreekbaar zijn. De reden dat dit lukt is dat iedereen enorm gemotiveerd is om het concept CHAIN*els* te laten slagen. In het team is nu ook duidelijk waar iedereen zijn expertise ligt en deze vaardigheden worden nu ook optimaal benut. Figuur 12 bevat een van de weinigte foto's van ons huidige team.

Ondanks een aantal dieptepunten is het team hierdoor hechter geworden en hebben we allemaal één visie en één doel: CHAIN*els* een succesvol Business network maken.

"One and one makes eleven possible" (Shahid Suddle)

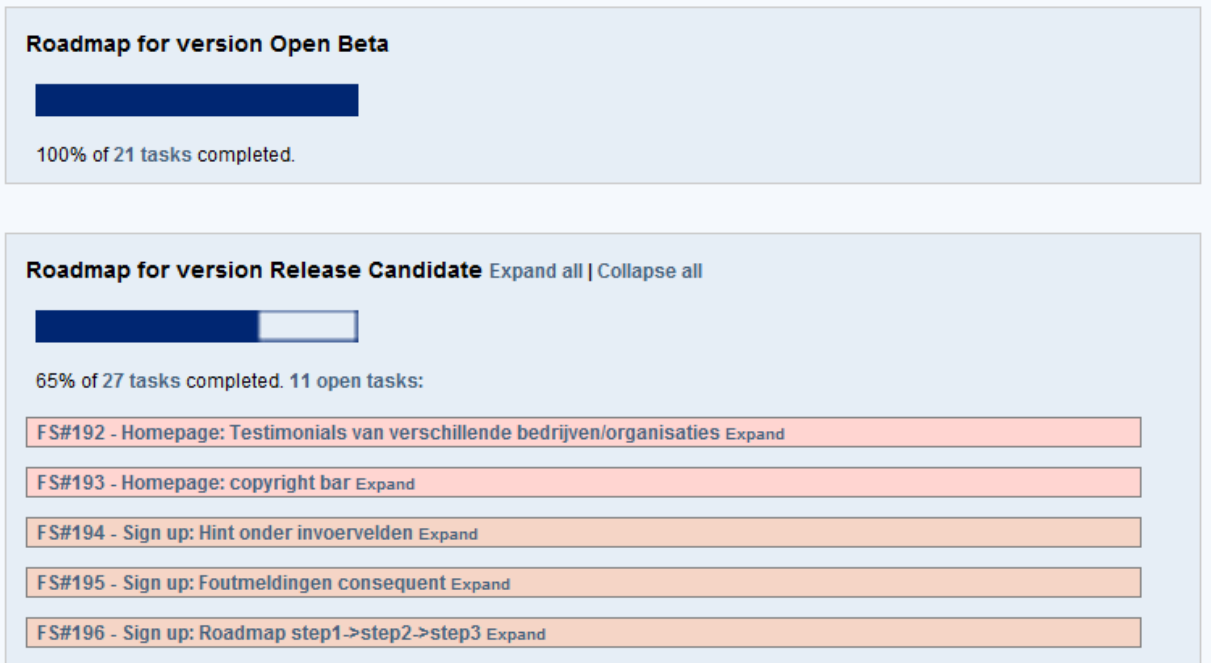
Figuur 12: Het team van links naar rechts: Vincent, Erwin, Willem, Remi en Shahid

6.2 ICT

De ontwikkelmethode die wij hanteerden was Scrum. Dit hield in dat we met sprints te werk zijn gegaan om taken te voltooien. Ter ondersteuning hebben we hierbij Flyspray gebruikt. Dit is een tool die we op onze server hadden draaien waar eenvoudig actiepunten kunnen worden aangemaakt en worden ingedeeld op prioriteit. Het is mogelijk om aan te geven wat voor taak het is, wie verantwoordelijk is, en wanneer het af zou moeten zijn. Zie hiervoor figuur 13. Doordat we in Flyspray ook aan kunnen geven bij welke milestone de taak hoort (zie paragraaf 2.2), is er ook een Roadmap met de voortgang, getoond in figuur 14. Het voordeel hiervan is dat iedereen altijd eenvoudig kan zien hoeveel taken er nog uitgevoerd moeten worden om een milestone te bereiken. De taken waren vooraf gedefinieerd in de Product Backlog (Appendix B en Hoofdstuk2).

ID	Category	Task Type	Severity	Summary	Status	Progress	Assigned To	Due Date
214	Back-end	Bug Report	Critical	Show en hide chains	Assigned		Erwin Buckers, +2	
208	Front-end	Feature Request	High	Accountrechten	Assigned		Remi Baar	
203	Front-end	Possible Improvement	High	Homepage: Klein schermen!!!!	Assigned		Remi Baar	
193	Front-end	Feature Request	High	Homepage: copyright bar	Assigned		Remi Baar	
192	Front-end	Feature Request	High	Homepage: Testimonials van verschillende bedrijven/orga...	Assigned		Remi Baar	
217	Back-end	Bug Report	Medium	Zoekalgoritme	Assigned		Erwin Buckers, +1	
215	Front-end	Feature Request	Medium	chainels.com/sscmbv	Assigned		Remi Baar	

Figuur 13: Flyspray actiepunten geordend op prioriteit



Figuur 14: Roadmap Flyspray

De belangrijkste communicatie was tijdens de vergaderingen, maar daarnaast streven we er sinds november naar zoveel mogelijk samen te werken. Indien dit niet mogelijk is gebruiken we vaak Skype.

Tijdens het ontwikkelen van het product, zijn de rollen wat gewisseld. Er is momenteel niet te zeggen wat nu precies al je taken zijn, want iedereen neemt indien nodig taken van anderen op zich. De hoofdrollen liggen wel vast, zie hiervoor de tabel in de vorige paragraaf.

6.3 CODE-EVALUATIE

Tijdens dit project is onze code tweemaal geëvalueerd door een extern bedrijf. De volledige resultaten hiervan zijn te vinden in Appendix H.

Samengevat scoort ons systeem best goed op de 'onderhoudbaarheidsschaal'. In de eerste evaluatie was de Database klasse het grootste knelpunt. Deze bevatte te veel en te lange functie, oftewel slechte Module Coupling en Unit Size/Complexity. Hierop hebben wij onder andere de Database klasse in meerdere klassen opgesplitst, namelijk Chains, GeneralActions, Holdings, waardoor een flink aantal functies op 'logischere plekken' terecht kwamen. Een slechte Unit Size en Complexity is echter lastig weg te werken met het oog op ons databasesysteem. Databasequeries zitten vaak binnen een multi/exec 'loop', wat betekent dat alle commando's daartussenin ($redis->multi() ... $redis->exec()) tegelijkertijd worden uitgevoerd voor atomiciteit (zie Appendix A, onderdeel Database). Dat soort dingen opsplitsen in functies vinden wij niet wenselijk, omdat er dan in die (sub)functies niet gegarandeerd kan worden dat Redis in de 'multi-mode' zit, wat wel vereist is.

Dit in acht nemende vond de evaluator bij de tweede evaluatie dat we zeker naar de adviezen geluisterd hadden. Overall was onze score echter iets omlaag gegaan door de verdubbelde omvang van het systeem. Wij zijn blij met deze resultaten, aangezien we zelf veel belang aan de onderhoudbaarheid van de code hechten. We ontwikkelen namelijk ook via de 'Clean Code'[18] methode, om onze code 'schoon' en uniform te houden.

6.4 DE LANCERING EN ANDERE LEUKE ASPECTEN

Naast een aantal dieptepunten zijn er een aantal hele leuke dingen gebeurd met betrekking tot CHAIN*els.* Zo is ons product twee keer op de radio geweest bij BNR, veel tweets over CHAIN*els* en er zijn circa tien artikelen gepubliceerd op Blogs over CHAIN*els.*

Het concept is gelanceerd in New York op 1-11-'11 (zie Appendix I voor foto's) door Shahid Suddle op het Social Media World Forum, wat ons een aantal leuke en bruikbare contacten op heeft geleverd.

De gesprekken met externen waren ook heel leerzaam en leuk. Zo is er een zakenman uit New York in Nederland geweest om onder andere een ochtend met ons te praten.

De officiële lancering in Delft op 11-11-'11 (zie Appendix I voor foto's) was ondanks een magere opkomst een bijzondere ervaring. Met microfoon, gefilmd worden en voor een zaal met voornamelijk directeuren (zestig aanwezigen) spreken was een nieuwe aangename ervaring. Ter afsluiting een borrel om na te praten maakte het plaatje compleet.

---

[18] Robert C. Martin, *Clean Code: A handbook of Agile Software Craftsmanship*, Prentice Hall 2008

6.5 WAT NOG KOMEN GAAT

Ondanks dat ons Bachelorproject ten einde is, gaan we door met CHAIN*els*. Er zijn verschillende redenen. Allereerst de uitdaging: het is spannend en moeilijk om dit concept daadwerkelijk werkend op de markt te krijgen. Ten tweede vermaak: de samenwerking is nu goed en er zijn ook veel leuke activiteiten rondom CHAIN*els*. Ten derde eigen bedrijf: naast dat we in januari 2011 met TofSoft (VOF) zijn begonnen vinden wij het leuk en uitdagend om je eigen bedrijf te hebben. Ten vierde potentie: de reactie die wij uit de MKB in Nederland ontvangen is positief. Het product zelf moet nog verbeterd worden maar indien dit lukt dan zijn er mogelijkheden dat CHAIN*els* heel groot wordt in Nederland en misschien zelfs internationaal.

Verder zijn we bezig om het team te versterken met afstudeerders en ervaren mensen. De business en marketing kant moet namelijk versterkt worden. Momenteel lopen er veel gesprekken. Ook wordt er een soort club van ambassadeurs opgezet waar directeuren van bedrijven in zitten die kunnen helpen bij het concept. Begin 2012 zal CHAIN*els* een BV zijn. Daarnaast zijn er plannen voor een kantoor/werkruimte. Het doel is om in 2012 10.000 bedrijven op het netwerk te krijgen voornamelijk in de regio Haagenlanden. Aan ondernemerswedstrijden als LiveWIRE en New Venture zullen we aan deelnemen. Het product zelf krijgt ook een nieuw design, zodat het concept nog innovatiever en unieker is. Hierbij krijgen we veel hulp van externen en zijn we bezig met het opzetten van een Pilot in een lokale industrie.

CONCLUSIE

Terugkijkend op dit project is het een geweldige ervaring geweest. Wat is er immers meer 'ICT-actueel' dan het ontwikkelen van een nieuw soort sociaal netwerk? Ook was het zeker uitdagend. We hopen dan ook nog lang met dit project door te kunnen gaan, en denken er zeker profijt van te hebben.

Maar zijn de in het oriëntatieverslag besproken keuzes wel goed uitgepakt? Wij denken van wel! Het resultaat spreekt immers voor zich. PHP was bijvoorbeeld over het algemeen een zeer fijne programmeertaal om mee te werken. Ondanks de zeer vervelende 'weak typing' (PHP kent geen type indicatoren als integer en String) en de perikelen met het testen (PHP compiled pas 'on run' waardoor je laat achter problemen komt) waren we erg tevreden met de functionaliteiten en snelheid ervan. Ook zeer handige modules als GeoIP, GD en phpRedis en de vele standaardfuncties (bijvoorbeeld 'similar_text') maakten het werk een stuk makkelijker. In combinatie met de eAccelerator module zijn er ook geen problemen met de snelheid. Bovendien hielden wij zelf de onderhoudbaarheid van het systeem sterk in de gaten, gestimuleerd door de vereiste koppeling met de User Interface en de evaluaties door de Software Improvement Group.

Onze grootste 'gok' was verder het gebruiken van Redis als databasesysteem. Na veel discussies binnen het ICT-team besloten we van het standaard MySQL af te stappen en voor een onconventioneel NoSQL systeem te gaan. Redis is niet wijdverspreid en zeker niet volledig doorontwikkeld, maar wel snel, heel snel. Bovendien passen de datatypen van Redis erg goed binnen ons concept. Veel functies zijn zo erg makkelijk en ook erg snel geïmplementeerd; wij zijn er zeker van dat ons systeem met MySQL een stuk langzamer was geweest. Van de relatief jonge leeftijd van dit product hebben wij tot nu toe niet veel gemerkt; de documentatie was bijvoorbeeld van een uitzonderlijk hoog niveau. Ook het geheugengebruik van Redis is tot nu toe nog geen probleem gebleken; honderden bedrijven passen in slechts een paar megabytes.

Qua server en beveiliging en dergelijke zijn wij ook erg tevreden. Sinds de lancering hebben we geen down-time gehad, en bovendien geen security issues. Gebruikers reageren erg positief op de snelheid van ons systeem.

Onze ontwikkelmethode hadden we echter beter kunnen gebruiken. Zeker in het begin waren we niet altijd heel georganiseerd bezig, vooral qua communicatie tussen ons en de User Interface programmeur. Dit hebben we echter later met vooral de introductie van Flyspray goed opgepakt, waarmee de taken en deadlines per persoon meteen duidelijk werden. Verder hebben wij persoonlijk erg veel met moderne middelen als Skype gewerkt, tot grote tevredenheid. Ook van de faciliteiten van de Technische Universiteit Delft hebben we dankbaar gebruik gemaakt voor programmeersessies. Het resultaat spreekt hopelijk voor zich. De evaluaties van de Software Improvement Group waren erg nuttig, en een welkome aanvulling op onze eigen visie van 'onderhoudbare code'. Dit laatste aspect is echter door tijdsgebrek voor de lancering licht ondergeschoven geraakt, maar op dit moment doen wij ons best om dit in te halen.

Op het gebied van testen hebben we echter veel geleerd. Voor al de unit tests bleken erg nuttig. Door dergelijke tests te schrijven weet je direct of je code werkt zoals je het wilt, wat extra nuttig is bij een niet vooraf gecompileerde taal als PHP. Door tijdsgebrek is de dekking van deze tests in de eindfase wat achteruit gegaan, maar zeker in het begin was deze erg hoog en zat er dan ook veel tijd in het maken van deze tests, maar dus met veel rendement. Vrijwel alle bugs in onze code zijn in dergelijke tests gevonden en verholpen; user tests droegen vooral op UI gebied veel bij.

Op een iets hoger niveau zijn we ook met het product zeer tevreden. Vooral ons gedeelte werkt in principe erg goed. Met de User Interface zijn er echter meer problemen, maar dit is niet geheel aan ons te wijten. De functionaliteiten zijn namelijk wel goed; de bruikbaarheid voor de user soms niet. Ook met dit aspect zijn wij op dit moment hard aan het werk gegaan.

Al met al was dit project zeker leerzaam. We hebben veel geleerd op allerlei gebieden: PHP, communicatie, Redis, serverbeheer, bedrijfsvoering, en nog veel meer. Wij hebben er zeker veel plezier in gehad, en hopen dat dit terug is te zien in de resultaten.

AANBEVELINGEN

Binnen CHAIN*els* valt er nog veel te verbeteren. Al tijdens het schrijven van dit verslag hebben we vele verbeteringen doorgevoerd. Zo hebben we op een zeer laag niveau de code 'gerefactored', wat puur verbeteren van de bruikbaarheid en onderhoudbaarheid van onze code inhoudt.

Er zijn echter nog een heleboel verbeterpunten over. Zo bleek bijvoorbeeld ons zoekalgoritme niet optimaal te zijn. Onze doelstelling van het zoeken naar industrieën en provincies en dergelijke heeft namelijk een serieus nadeel: het simpelweg zoeken op namen wordt lastiger. Zo hebben we al het voorbeeld van 'Verseput Bouw-en Mileutechniek B.V. ' genoemd, wat niet bovenaan komt als iemand op 'Verseput' zoekt. Een oplossing hiervoor is lastig, maar hier moet zeker aan gewerkt worden.

Verder zijn ook onze algoritmes niet optimaal. Zo gaat hetzelfde zoekalgoritme op dit moment alle bedrijven in het systeem af om tot het uiteindelijke resultaat te komen. Hier zit een mogelijk verbeterpunt in; immers, als het aantal bedrijven drastisch stijgt, is benadeling van de performance gegarandeerd. Dit probleem speelt in meer algoritmes een rol en moet dus nog aangepakt worden.

Ook ons multi-language systeem is niet zo efficiënt als het volgens ons zou moeten zijn. Door de opdeling in meerdere secties is er redundantie in geslopen. Ook staan er soms HTML-tags in de teksten; zeer onhandig is voor mogelijke vertalers. Dit gehele systeem moet gerevalueerd worden om het te verbeteren.

Een functie die nog helemaal niet is geïmplementeerd is de 'graaf'. Hiermee wordt een ongerichte graaf bedoeld waarmee een netwerk gevisualiseerd kan worden. Dit was wel in de originele plannen opgenomen, maar bleek te tijdrovend om te implementeren.

Ook het beheer van de twee servers bleek lastiger dan gedacht. Al is er op Redis niveau een koppeling, de bronbestanden (PHP) moeten ook gelijk zijn. Naar twee (en straks zelfs meer) servers uploaden bleek lastiger dan gedacht.

Ook Redis is helaas nog niet perfect geschikt voor een verdeling over een groot aantal servers. Daar wordt op dit moment echter binnen dat team aan gewerkt met de zogenaamde 'Redis Cluster'. Deze ontwikkelingen zullen wij op de voet volgen. Voor de serverkwestie zijn wij op dit moment bezig met een eigen programma om dit beheer te vereenvoudigen. Zo moet het uploaden van bestanden in één klik kunnen, en wordt het bovendien eenvoudiger om upgrades en dergelijke over meerdere servers uit te voeren.

Als laatste is het op te merken dat we veel eerder met een systeem als Flyspray hadden moeten beginnen. Vooral de samenwerking met de UI-programmeur liep in het begin stroef. Toen we begonnen met het aanmaken van kleine taken met een concrete deadline verliep dit een stuk beter. Ook dit is iets om in het vervolg door te zetten.

Met al deze punten zijn we zeker al aan het werk; op het moment van schrijven zijn we namelijk al bezig met 'CHAIN*els* 2.0', waarbij vooral de user interface volledig omgegooid zal worden aan de hand van de verkregen feedback sinds de lancering op 11 november 2011. Deze 'volgende versie' zal begin 2012 voltooid zijn, en we hopen dan alle bovenstaande punten verwerkt te hebben.

LITERATUURLIJST

1. Wikipedia, *Enumerated type*, http://en.wikipedia.org/wiki/Enumerated_type (20-12-11)
2. PHP Manual, *PCRE*, http://php.net/manual/en/book.pcre.php (20-12-11)
3. BJ Sintay, *Unix Time Stamp*, http://www.unixtimestamp.com/index.php (26-11-11)
4. PHP Manual, *microtime*, http://php.net/manual/en/function.microtime.php (26-11-11)
5. Wikipedia, *Moore's law*, http://en.wikipedia.org/wiki/Moore's_law (20-12-11)
6. Redis, *Transactions*, http://redis.io/topics/transactions (26-11-11)
7. Brian Wilson, *URL Encoding*, http://blooberry.com/indexdot/html/topics/urlencoding.htm (26-11-11)
8. LinodeWiki, *Configure Static IPs*, http://www.linode.com/wiki/index.php/Configure_Static_IPs#Static_IP_Configuration (26-11-11)
9. Wikipedia, *X-Forwarded-For*, http://en.wikipedia.org/wiki/X-Forwarded-For (26-11-11)
10. PHP Manual, *similar_text*, http://php.net/manual/en/function.similar-text.php (28-11-11)
11. Jonathan J. Oliver, *Decision Graphs - An Extension of Decision Trees*, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.1476 (28-11-11)
12. The Drawingboard, *PHP's similar text function*, http://blog.procontentanddesign.com/articles/single/215_PHP_s_similar_text_function.htm (28-11-11)
13. PHP Manual, *levenshtein*, http://www.php.net/manual/en/function.levenshtein.php (28-11-11)
14. Michael Gilleland, *Levenshtein Distance*, http://www.merriampark.com/ld.htm (28-11-11)
15. H.W. Lang, *Quicksort*, http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/quick/quicken.htm (28-11-11)
16. Wikipedia, *Floor and ceiling functions*, http://en.wikipedia.org/wiki/Floor_and_ceiling_functions (28-11-11)
17. OAuth, *Introduction*, http://oauth.net/about/ (28-11-11)
18. Robert C. Martin, *Clean Code: A handbook of Agile Software Craftsmanship*, Prentice Hall 2008

APPENDIX A - ORIËNTATIEVERSLAG

INLEIDING

In dit oriëntatieverslag zullen we een aantal onderwerpen bespreken die van belang waren bij de start van dit project. Basale keuzes als de keuze voor een programmeertaal, databasesysteem en meer worden besproken. Voor alle betreffende keuzes hebben we uiteraard uitgebreid onderzoek gedaan, en er zal dan ook een focus zijn op de literaire onderbouwing, ondersteund door onze eigen ervaring. Let wel dat er in het proces logischerwijs een flinke ontwikkeling is geweest op al deze gebieden, en sommige dingen hier mogelijk niet meer kloppen met het implementatieverslag zelf.

1. PROGRAMMEERTAAL

In een informatica project is een van de allereerste keuzes die voor een programmeertaal. Gezien het feit dat CHAIN*els* een webapplicatie is wordt de keuze al iets beperkt; er zijn echter alsnog vele opties met veel voor- en nadelen over! Om het wiel niet opnieuw te hoeven uitvinden hebben we vooral de systemen van bestaande (sociale) netwerken goed bekeken, en dan voornamelijk de systemen van de populairste netwerken: Facebook, Twitter en LinkedIn.

Allen hebben uitgebreide pagina's waarin hun systeem en dergelijke wordt uitgelegd, aangevuld door een flink aantal artikelen van derden.

1.1 ONDERZOEK

LinkedIn, om te beginnen, gebruikt vooral Java met de Spring MVC technologie[19]. Dit is vooral gericht op het schaalbaar en onderhoudbaar maken van projecten[20]. Er is echter aardig wat kritiek op een aantal fundamenten van dit systeem. Vooral de complexiteit, het hoge instapniveau, en het hoge percentage aan XML code, wat vele nadelen met zich meebrengt, krijgen kritiek[21]. Verder wordt er geklaagd over het ontbreken van documentatie, vooral op het gebied van beveiliging, en het overdadige aanbod van verschillende manieren om hetzelfde te doen. Als laatste wordt er ook getwijfeld aan de snelheid van dit systeem[22].

---

[19] LinkedIn, *LinkedIn Technology*, http://engineering.linkedin.com/technology (25-11-11)

[20] SpringSource, *Modern Web Applications*, http://www.springsource.org/features/modern-web (25-11-11)

[21] Web4J, *Criticisms of Spring, Struts, PHP, and Rails*, http://www.web4j.com/Criticisms_Drawbacks_Pitfalls_Spring_Rails_PHP.jsp (25-11-11)

[22] Peter Thomas, *Moving from Spring MVC / Webflow*, http://ptrthomas.wordpress.com/2007/03/02/wicket-impressions-moving-from-spring-mvc-webflow/ (25-11-11)

Twitter, vervolgens, gebruikt weer een heel ander systeem als basis: Ruby on Rails[23]. Dit is vooral gericht op het bieden van een volledig Model View Controller (MVC) framework om een zo eenvoudig mogelijke programmeertaal te bieden.

Dit is dus zonder het gebruik van XML documenten en dergelijke[24]. Net als bij de Spring technologie ligt de focus vooral op het faciliteren van de ontwikkeling. Echter zijn er ook weer een aantal nadelen: het databaseschema wordt door Rails zelf vastgelegd, meerdere databases gebruiken is moeilijk, en heeft net als Spring MVC een flink aantal beveiligingslekken[17].

Facebook, als laatste, gebruikt als basis het alom bekende PHP[25]. PHP is met meer dan 20.000.000 actieve websites verreweg de meeste gebruikte web-programmeertaal[26]. De grote voordelen van PHP volgen ook gedeeltelijk hieruit: er zijn duizenden modules voor specifieke taken, ontzettend veel documentatie, de beveiliging is door de lange ontwikkeling prima in orde, en het is vooral simpel: PHP is er op gericht om alles simpel te houden[27]. Een ander groot voordeel is onze eigen ervaring met PHP: we hebben hier beiden al veel in ontwikkeld. Met deze voordelen komen ook een aantal nadelen: zo is het in PHP, vooral zonder programmeerervaring en goed design, erg eenvoudig om de code zeer slecht onderhoudbaar te maken[17]. Ook kan het grote aanbod aan modules en standaardfuncties juist verwarrend werken. Qua snelheid is PHP vergelijkbaar met de andere talen, zeker in combinatie met Facebook's HipHop for PHP[28].

1.2 Keuze

Wat wij gezien de doelstellingen in ons systeem vooral in een programmeertaal zoeken is snelheid, veiligheid, en een brede basis. Bovendien eisen we dat er object-georiënteerd geprogrammeerd kan worden. Gezien onze eigen ervaring (en die van het designteam) lag onze eigen voorkeur al snel bij PHP, dat zeker aan de gestelde eisen voldoet[29]. Voor PHP gebruiken we de laatste stabiele versie (5.3.8), waarin op een ons zeer bekende manier object-georiënteerd geprogrammeerd kan worden[30].

---

[23] High Scalability, *Scaling Twitter*, http://highscalability.com/scaling-twitter-making-twitter-10000-percent-faster (25-11-11)

[24] ONlamp.com, *What is Rails,* http://onlamp.com/onlamp/2005/10/13/what_is_rails.html (25-11-11)

[25] Steve Campbell, *How Does Facebook Work?*, http://www.makeuseof.com/tag/facebook-work-nuts-bolts-technology-explained/ (25-11-11)

[26] PHP, *PHP Usage*, http://www.php.net/usage.php (25-11-11)

[27] Eric Ries, *Why PHP won*, http://www.startuplessonslearned.com/2009/01/why-php-won.html (25-11-11)

[28] Haiping Zhao, *HipHop for PHP*, http://developers.facebook.com/blog/post/358/ (25-11-11)

[29] Jeff Moore, *Comparing PHP with other languages*, http://www.procata.com/blog/archives/2006/02/09/comparing-php-with-other-languages/ (25-11-11)

[30] PHP Manual, *Classes and Objects*, http://php.net/manual/en/language.oop5.php (25-11-11)

Sinds de eerste versie van PHP in 1995 is de beveiliging van PHP op een zeer hoog niveau[31], en de zeer uitgebreide keuze uit modules en standaardfuncties van PHP die we zeker kunnen gebruiken trokken ons over de streep. Een ander pluspunt is dat de site waar we (zeker in het begin) het meeste op wilden lijken, Facebook, dit systeem ook gebruikt. Facebook is naar onze mening altijd snel en efficiënt geweest, in tegenstelling tot vooral Twitter, maar ook LinkedIn. Nu ligt dit niet alleen aan de programmeertaal, maar is dat zeker wel een factor.

Als laatste was de keuze voor PHP ook bijna praktisch noodzakelijk, gezien het feit dat wij en het designteam eerder alleen met deze taal hebben gewerkt, en het project snel van de grond moest komen. Het is dus echter zeker niet zo dat PHP ongeschikt is voor dit project; in tegendeel juist: we hebben er erg veel plezier van!

Overigens hebben we ook onderzoek gedaan naar Apache Thrift, een systeem (mede)ontwikkeld door Facebook. Dit is een systeem dat ontwikkeling in meerdere programmeertalen mogelijk maakt[32]. Door de grote complexiteit die bij een dergelijk systeem komt kijken hebben we hier echter niet voor gekozen: onze doelstelling is om het systeem zo uniform mogelijk te houden, vooral voor een zo simpel mogelijke koppeling aan het design.

2. DATABASE

Na PHP was de volgende stap de keuze voor de database. Gezien het feit dat onze keuze voor PHP vooral vanuit Facebook kwam, bekeken we als eerste het databasesysteem van Facebook: MySQL[33]. MySQL is net als PHP een zeer veel gebruikt systeem met een jarenlange geschiedenis[34]. Wij hebben beiden al aardig wat ervaring met MySQL, en voor standaardsystemen is het ook zeker een goede oplossing.

2.1 MySQL

Er is echter een groot probleem met MySQL: snelheid (en in mindere mate: schaalbaarheid). Een Google opdracht naar 'MySQL Performance' levert bijna alleen maar pagina's op met hoe de performance van MySQL te tweaken. Dit zegt natuurlijk al een heleboel over de basislijn van MySQL.

Uit eigen ervaring weten we ook dat MySQL langzaam is, en daarom zijn we op zoek gegaan naar een sneller alternatief, onder andere gebaseerd op het flinke aantal snelheidsvergelijkingen dat online te vinden is[35]. De alternatieven voor (My)SQL worden onder één algemene noemer NoSQL genoemd[36].

---

[31] Wikipedia, *PHP*, http://en.wikipedia.org/wiki/PHP#Security (25-11-11)

[32] Apache, *Thrift*, http://wiki.apache.org/thrift/ (25-11-11)

[33] Pingdom, *Exploring the software behind Facebook,* http://royal.pingdom.com/2010/06/18/the-software-behind-facebook/ (25-11-11)

[34] MySQL, *Why MySQL?*, http://www.mysql.com/why-mysql/ (25-11-11)

[35] Ruturaj.net, *MySQL comparison*, http://www.ruturaj.net/redis-memcached-tokyo-tyrant-mysql-comparison (25-11-11)

[36] Wikipedia, *NoSQL*, http://en.wikipedia.org/wiki/NoSQL (25-11-11)

2.2 NoSQL

NoSQL wijkt af van het 'standaard MySQL' in het opzicht dat er van het principe van een 'relationele database' wordt afgestapt. Relationele databases zijn namelijk bewezen niet geschikt voor data-intensieve applicaties, waarbij er veel gelezen én geschreven wordt[37]. Veel van de NoSQL use-cases komen overeen met onze beoogde doeleinden.

NoSQL databases bestaan in een aantal categorieën: document-georiënteerd, graaf-georiënteerd, en zogenoemde key-value opslag.

Gezien onze beoogde doelstellingen zijn de eerste twee categorieën niet van toepassing. Key-value opslag heeft fundamenteel geen schema, en er is dus geen vaststaand opslagmodel. Een dergelijk systeem leek ons zeer geschikt voor onze applicatie, vooral gezien de zeer hoge snelheid die met een dergelijk design gepaard gaat. De keuze voor een NoSQL key-value systeem was dan ook snel gemaakt, gezien het feit dat voor ons de belangrijkste aspecten snelheid, betrouwbaarheid en schaalbaarheid zijn. De keuze voor de daadwerkelijke tool was echter lastiger.

Onze eerste kandidaat was Memcached, een tool die meer en meer door Facebook gebruikt wordt[38]. Memcached is een heel simpel en basaal systeem waarmee objecten vanuit een al bestaande database in het geheugen van een server gecached kunnen worden[39]. Dit biedt zeer grote snelheidswinsten, en ook schaalbaarheid is eenvoudig haalbaar. Het is echter zo dat de beperking tot alleen het geheugen nadelen met zich meebrengt met betrekking tot de omvang van de dataset en gegevensverlies. Een mogelijk antwoord hierop is Membase, een opslagsysteem dat gekoppeld kan worden aan een Memcached systeem[40]. De ondersteuning en documentatie hiervoor is echter zo gering dat we al snel verder zochten, en wat vonden: Redis!

2.3 Redis

Wat ons meteen al opviel was de uitgebreide documentatie en ondersteuning hiervoor[41]. Er is zelfs een 'namaak Twitter' beschikbaar gebaseerd op Redis en PHP[42].

Ook uit verschillende NoSQL vergelijkingssites kwam Redis als een zeer goede optie naar voren, vooral door de zeer hoge snelheid van het systeem en de 'uitbreiding' op het standaard key-value systeem door extra datastructuren als sets, lijsten, en meer[43].

---

[37] Highscalability.com, *What the heck are you actually using NoSQL for?*, http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html (25-11-11)

[38] Paul Saab, *Scaling memcached at Facebook*, http://www.facebook.com/note.php?note_id=39391378919 (25-11-11)

[39] Memcached, *A distributed memory object caching system*, http://memcached.org/about/ (25-11-11)

[40] Couchbase, *Membase Server*, http://www.couchbase.com/products-and-services/membase-server (25-11-11)

[41] Redis, *Redis Documentation*, http://redis.io/documentation (25-11-11)

[42] Redis, *Retwis*, http://redis.io/topics/twitter-clone (25-11-11)

Al is Redis vrij nieuw (uit 2009), de ontwikkeling ervan is in een zeer hoog tempo gegaan, waardoor er nu al een Redis versie 2.4.2 is. Redis is bovendien geëvalueerd uit Memcached, met hetzelfde 'in memory' principe. Er is echter zonder extra systemen ook aan veilige opslag gedacht: er bestaan meerdere opties voor het opslaan van data, indien nodig met de benodigde redundantie. De gebruiker kan zelf kiezen hoe veilig dit moet zijn: 'snapshots' om de zoveel minuten, of zelfs een synchronisatie per seconde[44]. Ook het probleem van een te grote dataset die niet meer in het geheugen past is opgelost door een eigen, weer zelf instelbare 'virtual memory' oplossing in te bouwen.

Hierdoor kunnen zogenoemde 'hot spots' in het geheugen gehouden worden voor extreme snelheid, terwijl vrijwel niet (meer) gebruikte data naar de harde schijf weggeschreven kan worden[45]. Het is dus duidelijk te zien dat er veel verbeteringen in verhouding met het 'simpele' Memcached zijn doorgevoerd, zonder verlies aan snelheid; er wordt zelfs snelheidswinst geclaimd. Verder is Redis ook gebouwd op een systeem van meerdere servers: replicatie is zeer eenvoudig en duurzaam op te stellen, waardoor er gemakkelijk en snel winst in snelheid en datazekerheid geboekt kan worden[46]. In verhouding met een aantal andere NoSQL systemen was ook de 'instapdrempel' voor Redis heel laag, en het systeem zeer overzichtelijk, vooral door de extra datatypen! Zo zijn er dus sets, lijsten, hashes, en zelfs gesorteerde sets, die allemaal een zeer adequate ondersteuning bieden voor een flink aantal use-cases, waaronder ook veel van de onze. Uiteindelijk hebben we na overleg in het team besloten om de 'sprong in het diepe te wagen' en voor Redis te kiezen in plaats van MySQL.

Daarna was er nog één database-technische keuze over: binnen het Redis systeem bestaan er meerdere 'clients' per programmeertaal. Zo zijn er al voor alleen PHP ongeveer vijf. Gebaseerd op ons persoonlijk gevoel bij de API van de verschillende clients en vooral een aantal online benchmarks[47] kozen we uiteindelijk voor 'PhpRedis'. Deze client is in tests veruit de snelste doordat het een pure PHP module is, wat ook erg eenvoudig is qua installatie en gebruik. Het bestaat al sinds het begin van Redis zelf, en is in de tussentijd zeer doorontwikkeld. We vinden zelf bovendien dat de 'stijl' van PhpRedis paste binnen ons systeem: eenvoudig gebruik in duidelijke object-georiënteerde functieaanroepen. Ook de documentatie ervoor is erg overzichtelijk en uitgebreid[48].

---

[43] Kristóf Kovács, *NoSQL Comparsion*, http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis (25-11-11)

[44] Redis, *Persistence*, http://redis.io/topics/persistence (25-11-11)

[45] Redis, *Virtual Memory*, http://redis.io/topics/virtual-memory (25-11-11)

[46] Redis, *Replication*, http://redis.io/topics/replication (25-11-11)

[47] Ori Pekelman, *Which PHP Library to use with Redis?*, http://dev.af83.com/af83-open-source-projects/which-php-library-use-redis-benchmark/2011/01/01 (25-11-11)

[48] Nicolas Favre-Felix, *PhpRedis*, https://github.com/nicolasff/phpredis (25-11-11)

2.4 RESULTAAT

Zo kozen we dus uiteindelijk voor een combinatie van Redis en PHP, gekoppeld met PhpRedis. Uit deze keuze voor programmeertaal en database volgde nog het gebruik van een aantal andere tools, die we hier kort zullen toelichten. Ten eerste hebben we om server-technische redenen van het gebruik van het eerder genoemde HipHop for PHP afgezien en uiteindelijk gekozen voor eAccelerator: een optimalisatie- en cachingsoplossing voor PHP code.

PHP code wordt namelijk normaliter pas 'on run' gecompileerd naar uitvoerbare bytecode, wat voor veel tijdsverlies kan zorgen. eAccelerator cached de al gecompileerde code en optimaliseert deze ook nog eens, wat een flinke snelheidswinst tot gevolg heeft[49]. Er bestaan hier wat meer oplossingen (zoals APC) voor, maar eAccelerator is volgens verschillende tests de snelste[50].

Ook de eenvoudige installatie (weer als een pure PHP module) en configuratie is een voordeel. Verder gebruiken we voor de internationalisering de PHP 'mbstring' module. 'mb' staat hier voor multi-byte, wat inhoudt dat niet alleen de standaard ASCII karakters worden ondersteund, maar karakters uit alle talen[51]. PHP ondersteunt dit standaard niet; deze module bouwt echter deze ondersteuning zonder verdere vereisten in. Naast aparte 'multibyte functies' is het namelijk ook mogelijk om de standaard string functies te overschrijven. De karakterset die wij hierbij gebruiken is Unicode (UTF-8): een karakterset/coderingsstandaard die standaard álle mogelijke tekens ondersteunt[52]. Hieronder vallen letters, leestekens, letters met leestekens, symbolen, enzovoorts, in alle gebruikte schriften op de wereld. Verder gebruiken we voor statistische doeleinden de GeoIP PHP module van MaxMind, waarmee IP-adressen van bezoekende gebruikers gelinkt worden aan geografische locaties[53]. Een aantal standaard PHP uitbreidingen zoals een module voor het bewerken van afbeeldingen (gd), internationalisering van datum/tijd (date), en meer worden ook gebruikt. Tevens gebruiken we de SimpleTest module voor PHP[54], een van de weinige object-georiënteerde unit testing modules voor PHP.

3. SERVER

Naast de programmeertaal en het databasesysteem is ook de keuze voor de server waarop dit alles gedraaid wordt belangrijk. Eerst was er de keuze voor het besturingssysteem van de server. Dit kwam al heel snel uit op CentOS: een Linux variant gericht op professionele omgevingen.

---

[49] Wikipedia, *eAccelerator*, http://en.wikipedia.org/wiki/EAccelerator (25-11-11)

[50] 2bits.com, *Benchmarking Drupal with PHP op-code caches*, http://2bits.com/articles/benchmarking-drupal-with-php-op-code-caches-apc-eaccelerator-and-xcache-compared.html (25-11-11)

[51] PHP Manual, *Multibyte String*, http://www.php.net/manual/en/intro.mbstring.php (25-11-11)

[52] Markus Kuhn, *UTF-8 and Unicode*, , http://www.cl.cam.ac.uk/~mgk25/unicode.html (25-11-11)

[53] MaxMind, *GeoIP PHP API*, http://www.maxmind.com/app/php (25-11-11)

[54] SimpleTest, *Unit Testing for PHP*, http://www.simpletest.org/ (25-11-11)

Linux is het meest gebruikte systeem voor servers; er bestaan Windows servers, maar deze zijn ver in de minderheid[55]. Veel systemen, zoals Redis, draaien alleen op Linux. Ook is Linux kostenvrij, en simpelweg zeer snel en efficiënt. De keuze voor specifiek CentOS is op basis van persoonlijke voorkeur gemaakt; veel maakt dit dan ook verder niet uit.

3.1 HOSTING

De volgende keuze is die voor de host: waar moet de server draaien? Deze keuze was minder snel gemaakt. Voor een applicatie als de onze volstaat een standaard webhost niet: een eigen systeem dat geheel onder onze eigen controle staat is vereist, net als hoge snelheid, geschikte uitbreidingsmogelijkheden, en snelle ondersteuning.

Ook de niet geringe kosten van een dergelijke host speelden een grote rol in deze keuze. De eerste keuze die gemaakt moet worden is tussen zogenoemde 'virtual' en 'dedicated' hosting. Bij een dedicated hosting heb je geheel eigen hardware tot je beschikking. Dit kan iets beter zijn voor de snelheid, maar is zeer nadelig voor onderhoud, fouttolerantie, en ook zeker de kosten. Bij virtual hosting heb je wel een eigen server maar geen eigen hardware. De server draait dus 'ergens' in het datacentrum van de provider, wat ook wel 'cloud hosting' wordt genoemd.

Dit kan iets langzamer zijn, maar is veel gebruiksvriendelijker en vooral goedkoper, en bied ook voordelen op het gebied van fouttolerantie. Uiteindelijk kozen we er dus voor om een virtuele server te nemen, ook met het oog op mogelijk geheel eigen servers in de toekomst.

Na uitgebreid onderzoek naar verschillende grote hostingproviders in virtual (cloud) hosting als Amazon EC2, XLS Hosting, Rackspace, Slicehost, en meer kwamen we uit op onze uiteindelijke keuze: Linode. Wat ons bij Linode aantrok was het grote scala aan mogelijkheden: het uitbreiden of zelfs bijzetten van servers kan in slechts enkele minuten, er is een uitgebreide administratieinterface, high-tech datacentrums door de hele wereld (ook in Londen, een ideale locatie voor ons), ingebouwde back-ups, ingebouwde balanceringsmechanismen, en meer[56]! De kosten voor een Linode (oftewel: ene Linode server) waren bovendien zeer gering, zeker in vergelijking met andere gelijkwaardige producten. Bovendien biedt Linode een enorm scala aan nuttige documentatie voor allerlei server-gerelateerde zaken in een 'Linode Library'[57]. Ook blijkt uit verschillende tests dat de diensten van Linode van zeer hoge kwaliteit zijn[58]. Bovendien is er geen vast contract: je betaalt voor wat je krijgt, en verder niks.

---

[55] Ellen Messmer, *Windows Server vs. Linux*, http://www.networkworld.com/news/2010/060710-tech-argument-windows-server-linux.html (25-11-11)

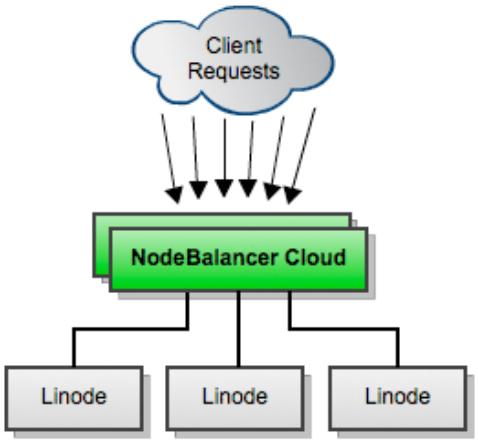
[56] Linode, *Features*, http://www.linode.com/features.cfm (25-11-11)

[57] Linode, *VPS Guides and Technical Articles*, http://library.linode.com/ (25-11-11)

[58] Eivind Uggedal, *VPS Performance Comparison*, http://journal.uggedal.com/vps-performance-comparison/ (25-11-11)

3.2 LINODE

We hebben dus uiteindelijk voor Linode gekozen, en draaien nu in London twee servers die automatisch gebalanceerd worden door een zogenoemde 'Load Balancer' (zie figuur 15). Ook is er een uitgebreid back-up systeem in werking, en zijn de servers volledig redundant: er is slechts op Redis niveau een koppeling voor de uitwisseling van nieuwe databasegegevens, en alle servers hebben daarbij alle (zelfde) gegevens Op elk willekeurig moment zouden we nog precies een zelfde server erbij kunnen zetten, of het geheugen van de bestaande servers kunnen uitbreiden. Ook onderhoud is eenvoudig: door het automatische balanceren heeft één server tijdelijk uitschakelen geen enkel effect voor de gebruiker, behalve mogelijk iets langere laadtijden. Dit komt omdat de balancer automatisch detecteert of een server werkt of niet, en het verkeer alleen naar werkende servers doorstuurt.



Figuur 15 - Visualisatie van de NodeBalancer

4. ENCRYPTIE EN ANDERE BEVEILIGING

Zoals eerder benoemd hebben wij beveiliging in een hoog vaandel staan, in combinatie met de privacy van de gebruiker. Dit uit persoonlijke standpunten, maar ook vanwege de vele kritiek die er recent is op sites als Facebook[59] of zelfs de hele overheid[60]. Veiligheid en privacy zijn dus hard nodig, en dit is in verschillende gebieden toepasbaar binnen ons concept.

De crux zit echter vooral in het inloggen van een gebruiker op onze website met een door hun gekozen wachtwoord. Aan dit concept zitten namelijk veel haken en ogen.

4.1 TRANSPORTLAAG

Ten eerste wordt het wachtwoord over 'Het Internet' vanaf de gebruiker naar onze server toegezonden. Onder normale omstandigheden is een dergelijke transactie af te vangen door een zogenaamde 'man-in-the-middle attack', waarbij een derde tussen de verbinding tussen onze gebruiker en onze server gaat inzitten om zo alle data uit te lezen[61]. Om dit te voorkomen kan er in plaats van een HTTP verbinding, het standaard internet protocol, voor een HTTPS verbinding gekozen worden, waarbij de S voor 'Secure' staat. Hierbij wordt er van een certificatenstructuur gebruik gemaakt om te zorgen dat de gebruiker direct met de server communiceert, en met niemand anders. Bij een certificeringsinstantie kan namelijk een dergelijk certificaat worden gekocht, waarbij er een unieke zware code wordt gegenereerd om de server te identificeren.

Dit certificaat is door de betreffende certificeringsinstantie 'ondertekend' met de code van die instantie.

---

[59] nu.nl, *Facebook dicht bij akkoord rond privacy*, http://www.nu.nl/internet/2665433/facebook-dicht-bij-akkoord-rond-privacy.html (25-11-11)

[60] nu.nl, *De lekkende overheid*, http://www.nu.nl/column-zaterdag/2642121/lekkende-overheid.html (25-11-11)

[61] Wikipedia, *Man-in-the-midle attack*, http://en.wikipedia.org/wiki/Man-in-the-middle_attack (25-11-11)

Door de code van deze instantie en de code van de server te controleren weet de gebruiker vrijwel zeker (als de instantie betrouwbaar is) dat er met de juiste persoon gecommuniceerd wordt. Eén servercertificaat is slechts op één specifieke domeinnaam bruikbaar. Op de CHAIN*els* servers wordt een dergelijk certificaat dan ook altijd gebruikt wanneer er gevoelige data moet worden verzonden of ontvangen.

4.2 OPSLAG

Als het wachtwoord eenmaal veilig over en weer kan worden gezonden is het natuurlijk ook niet de bedoeling dat het bij ons opgeslagen wachtwoord achterhaald kan worden. Hiervoor is encryptie nodig: het versleutelen van een wachtwoord zodat in opgeslagen vorm het origineel niet meer te achter halen is. Het moet dan zo zijn dat een zelfde wachtwoord tot een zelfde codering leidt, maar deze codering andersom niet te breken is, zodat slechts de codering opgeslagen hoeft te worden op onze server. Een hash moet uiteraard wel ook altijd uniek zijn. Hiervoor bestaan verschillende geaccepteerde technologieën, waarvan de meest gangbare MD5 en SHA-1 zijn. Dit zijn zeer efficiënte algoritmes die zeer snel een zogenoemde 'hash' voor een bepaalde input kunnen berekenen. Deze efficiëntie heeft echter ook een nadeel: het omgekeerde is ook sneller mogelijk. Gezien de moderne technologie zijn zogenaamde 'brute force attacks' of zelfs 'rainbow table attacks' namelijk steeds sneller te doen! Bij dit soort aanvallen wordt geprobeerd de hash te reconstrueren, oftewel bronwoorden uit te proberen. 'Rainbow tables' maken hierbij gebruik van veel voorkomende woorden om dit proces te versnellen. Om dit soort aanvallen te voorkomen kan ten eerste gebruik gemaakt worden van zogenoemde 'salts'. Hierbij wordt een willekeurige tekenreeks aan elk wachtwoord toegevoegd, waardoor checken op veelvoorkomende wachtwoorden niet meer kan. Hoe willekeuriger deze salt weer is, hoe beter, gezien het feit dat gebruikerswachtwoorden meestal niet zo willekeurig zijn. Ten tweede kan de encryptie 'zwaarder' worden gemaakt. Er moeten immers bij een aanval steeds hashes berekend worden om die te kunnen vergelijken met de opgeslagen hash. Als het dus simpelweg meer tijd kost om een hash te berekenen worden aanvallen hierop steeds onaantrekkelijker.

Hét aangeraden/geaccepteerde algoritme hiervoor is 'Bcrypt', gebaseerd op 'Blowfish' encryptie[62]. Bcrypt gebruikt namelijk een variabel aantal 'ronden', waarmee de tijd die de encryptie nodig heeft door de gebruiker kan worden aangepast, om zo een optimale oplossing tussen snelheid en veiligheid te vinden. Zo werken MD5 en SHA-1 principieel in minder dan een microseconde; Blowfish werkt normaliter in tienden van seconden: een zeer significant verschil! Dit algoritme is nog veel effectiever dan de zojuist besproken salts, die namelijk alsnog benaderd kunnen worden[63]. Ook houdt dit algoritme rekening met de Wet van Moore, waarbij er algemeen gesteld wordt dat de capaciteit van computers elke twee jaar verdubbelt; bij Bcrypt kan hiervoor gecompenseerd worden door simpelweg het aantal ronden van de encryptie te verhogen, waardoor het algoritme 'even zwaar' blijft bij ontwikkelingen in de technologie. Wij gebruiken dan ook Bcrypt voor het opslaan van al onze wachtwoorden, met behulp van de standaard ondersteuning die hiervoor is in PHP.

---

[62] Thomas Pornin, *Do any security experts recommend bcrypt for password storage (top rated answer)*, http://security.stackexchange.com/questions/4781/do-any-security-experts-recommend-bcrypt-for-password-storage (25-11-11)

[63] Coda Hale, *How To Safely Store A Password*, http://codahale.com/how-to-safely-store-a-password/ (25-11-11)

4.3 Sessies

Naast HTTPS en Bcrypt voor het inloggen, moet de gebruiker ook ingelogd kunnen blijven op onze site. Immers moet een gebruiker niet voor elke pagina het wachtwoord opnieuw in te hoeven vullen. Hiervoor is het concept van 'sessies' nodig. Nu heeft PHP hier een standaard ondersteuning voor, maar is deze niet geschikt om over meerdere servers tegelijk te werken, doordat er een bestand per server wordt opgeslagen bij deze methode. Daarom moet dit systeem in Redis komen: het enige systeem dat tussen servers gekoppeld is. Overigens geeft dit ook een niet geringe snelheidswinst.

Sessies werken doordat elke user bij het inloggen een door de server gegenereerde unieke code toegewezen krijgt die bij hem of haar opgeslagen wordt, waarmee de ingelogde gebruiker in het vervolg geïdentificeerd word zonder opnieuw een wachtwoord in te voeren. Deze code moet logischerwijs willekeurig zijn en lang genoeg om niet gegokt te kunnen worden[64]. Ook deze code moet logischerwijs over een HTTPS verbinding verstuurd worden om niet afgeluisterd te kunnen worden. Een derde kan immers met de sessiecode die een gebruiker toebedeeld krijgt alsnog voor die gebruiker inloggen[65].

Ook is het van belang dat een sessiecode niet altijd hetzelfde is, om de kans op 'gokken' nog meer te verkleinen, en daarom wordt deze bij ons voor elke gebruiker om het kwartier opnieuw gegenereerd. Dit is een afweging tussen snelheid en veiligheid: het genereren en opslaan van een code kost namelijk gemiddeld gezien veel tijd, en het is dus niet gewenst om dit te vaak te doen. Te lang wachten met regeneratie gaat echter weer ten koste van de veiligheid, omdat slechts in het betreffende tijdvak door een kwaadwillende derde geprobeerd kan worden de code te 'brute-forcen'.

Als laatste opmerking is natuurlijk ook het door de gebruiker gekozen wachtwoord van belang. Een te simpel (voorspelbaar) wachtwoord is ondanks alle encryptie nog steeds niet veilig. Vandaar dat wij ook eisen stellen aan het wachtwoord van de gebruiker. Een wachtwoord moet minimaal 'Sterk' zijn volgens ons systeem. Dit oordeel wordt geveld aan de hand van de lengte van het wachtwoord en het gebruik van cijfers en leestekens[66]. Zo wordt een groot gedeelte van 'onveilige wachtwoorden' uitgezonderd, wat het systeem nog een stap veiliger maakt. Doordat een gebruiker echter alsnog zelf onveilig kan zijn met de geheimhouding van zijn of haar wachtwoord (die wij nooit zullen weten) is er nog steeds een risico. Van onze kant hebben we er echter alles aan gedaan om het algemene risico zo veel mogelijk in te perken.

---

[64] Andrew Johnson, *Generating Session IDs*, http://www.itnewb.com/tutorial/Generating-Session-IDs-and-Random-Passwords-with-PHP (25-11-11)

[65] Chris Shiflett, *Essential PHP Security (Chapter 4)*, http://phpsecurity.org/ch04.pdf (25-11-11)

[66] Password Advisor, *Check password strength using PHP*, http://passwordadvisor.com/CodePhp.aspx (25-11-11)

5. ONTWIKKELMETHODE

Bij de keuze van een ontwikkelmethode is het belangrijk dat het systeem eenvoudig onderhoudbaar is en dat het nog uitgebreid kan worden, want een netwerk is in praktijk bijna nooit af. Daarnaast is het belangrijk dat we met kleine intervallen werken om veel functionaliteiten te kunnen ontwikkelen. Feedback van de gebruiker is ook heel belangrijk, omdat het op de bedrijven afgestemd moet worden. Tot slot is het aspect "teamwork" ook belangrijk, want vanaf het begin zijn we met ze drieën en in de toekomst waarschijnlijk nog met meer.

We hebben ons in de volgende ontwikkelmethode verdiept:

- Agile-software-ontwikkeling
    - Extreme Programming (XP)
    - Scrum
- Traditionele ontwikkelmethoden
    - System Development Methodology (SDM)
    - Bottom-up design
    - V-model

De eerste analyse was dat in het geval van een netwerk de traditionele ontwikkelmethode niet geheel tot zijn recht kan komen. De reden hiervoor is dat het eindproduct niet vastligt, want we zijn heel erg afhankelijk van feedback. Automatisch volgt hieruit dat ook de planning niet van te voren helemaal vastgelegd kan worden.

Bij het V-model moet elke fase afgerond zijn om aan de nieuwe te beginnen, dit principe is bij CHAIN*els* erg onpraktisch, omdat door feedback van gebruikers prioriteiten steeds kunnen veranderen. Het is ook lastig om aan de faseringen te voldoen want in de business case kan lastig worden vastgesteld wat een gebruiker precies wil. Dat is beter uit feedback van de gebruiker te halen.

SDM komt deels overeen met het V-model maar daar komt nog bij dat het principe dat een fase goedgekeurd moet worden door een opdrachtgever niet relevant is. Het aspect dat de systeemeisen hierdoor niet meer veranderd kunnen worden is in dit opzicht juist niet de bedoeling, want feedback en een iteratief proces is nodig om een netwerk functioneel werkend te krijgen.

Bottom-up design heeft een andere insteek, maar het los ontwikkelen van componenten en later toevoegen is in veel gevallen niet slim bij een netwerk, omdat veel functionaliteiten los niks voorstellen maar indien ze gebruikt worden allemaal in elkaar verweven zijn. In sommige gevallen is het wel slim om los te ontwerpen (ook bij CHAIN*els).* Zo kan het systeem relatief simpel gehouden worden.

Na de analyse van de traditionele systemen zijn wij ons gaan verdiepen in Agile-software-ontwikkeling. XP en Scrum liggen dicht bij elkaar en daarom hebben wij ons verdiept in de verschillen en deze tegen elkaar uitgezet (zie onderstaande tabel).

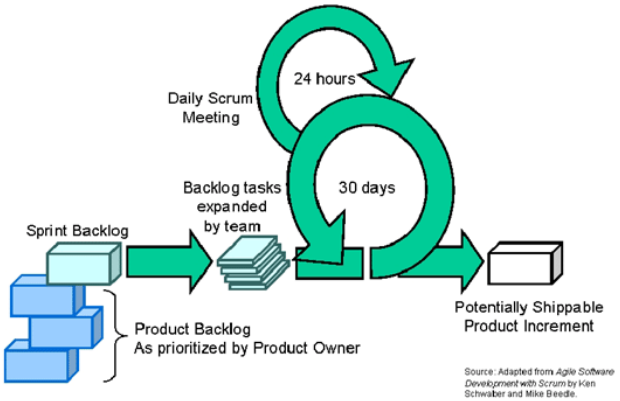
	Extreme Programming (XP)	Scrum	Voorkeur
1	Iteraties 2 weken lang	Iteraties 2 tot 4 weken	geen
2	In de iteratie meer flexibiliteit	Iteratie(sprint) wordt niet meer aangepast	Scrum
3	Hoogst prioriteit gaat voor, team beslist hier minder over	Wel een prioriteit lijst maar de keuze volgorde is aan het team	Scrum
4	Test driven developement	Geen echte engineering practise (peer programming)	XP

Het is belangrijk te beseffen dat wij gaan ontwikkelen in een team van drie. De meeste modellen zijn gericht op grote teams. Dus in een klein team is het niet mogelijk iedereen één rol toe te bedelen. Indien wij een rationeel besluit zouden nemen zeggen wij Scrum twee voor en maar één tegen. Onze keuze is wel Scrum geworden, maar is beter gefundeerd.

Wij communiceren in het team tijdens vergaderingen en verder voornamelijk via Skype. In vergaderingen worden actiepunten opgesteld en beslissingen genomen. Daarom is punt 2 in het voordeel van Scrum. Indien wij ze wel gaan aanpassen nemen wij beslissingen zonder dat we het met het hele team hebben overlegd. Hoelang een iteratie precies moet zijn is lastig te zeggen en daarom konden wij aan de hand van alleen documentatie niet vaststellen wat de voorkeur geniet.

XP heeft als voordeel het Test driven developement, maar omdat ons team klein is komt dit minder tot zijn recht. Er is ook maar een beperkt tijdsbestek dus peer programming is niet echt mogelijk. Dit alles is gevisualiseerd in figuur 16.

Kortom, wij kiezen voor Scrum omdat wij verwachten dat dit het beste werkt voor een concept als CHAIN*els* en daarnaast hebben wij ook al een klein beetje ervaring met deze methode, dankzij een lezing over Scrum bij het vak Software Engingeering en het toepassen van Scrum in een voorgaand project.



Figuur 16: Scrum gevisualiseerd

SAMENVATTING

In dit oriëntatieverslag hebben we op allerlei relevante gebieden de voor- en nadelen van verschillende designkeuzes uiteengezet en onze uiteindelijke keuzes gefundeerd. Zo programmeren we object-georiënteerd in PHP met daarachter een Redis database, ondersteund door verschillende beveiligingsmechanismen en een aantal ondersteunende modules. Dit alles draait op dit moment op een tweetal servers in de 'Linode cloud' met gebruik van load balancing. Verder ontwikkelen we met behulp van de Scrum methode, waarbij het Flyspray systeem ons daarin ondersteunt. Al achtten wij alle keuzes op het moment oprecht juist, slechts uit het resultaat zal blijken of dit de juiste keuzes waren. Immers: "Als we wisten wat we deden, heette het geen onderzoek." (Albert Einstein)

LITERATUURLIJST

19. LinkedIn, *LinkedIn Technology*, http://engineering.linkedin.com/technology (25-11-11)
20. SpringSource, *Modern Web Applications*, http://www.springsource.org/features/modern-web (25-11-11)
21. Web4J, *Criticisms of Spring, Struts, PHP, and Rails*, http://www.web4j.com/Criticisms_Drawbacks_Pitfalls_Spring_Rails_PHP.jsp (25-11-11)
22. Peter Thomas, *Moving from Spring MVC / Webflow*, http://ptrthomas.wordpress.com/2007/03/02/wicket-impressions-moving-from-spring-mvc-webflow/ (25-11-11)
23. High Scalability, *Scaling Twitter*, http://highscalability.com/scaling-twitter-making-twitter-10000-percent-faster (25-11-11)
24. ONlamp.com, *What is Rails*, http://onlamp.com/onlamp/2005/10/13/what_is_rails.html (25-11-11)
25. Steve Campbell, *How Does Facebook Work?*, http://www.makeuseof.com/tag/facebook-work-nuts-bolts-technology-explained/ (25-11-11)
26. PHP, *PHP Usage*, http://www.php.net/usage.php (25-11-11)
27. Eric Ries, *Why PHP won*, http://www.startuplessonslearned.com/2009/01/why-php-won.html (25-11-11)
28. Haiping Zhao, *HipHop for PHP*, http://developers.facebook.com/blog/post/358/ (25-11-11)
29. Jeff Moore, *Comparing PHP with other languages*, http://www.procata.com/blog/archives/2006/02/09/comparing-php-with-other-languages/ (25-11-11)
30. PHP Manual, *Classes and Objects*, http://php.net/manual/en/language.oop5.php (25-11-11)
31. Wikipedia, *PHP*, http://en.wikipedia.org/wiki/PHP#Security (25-11-11)
32. Apache, *Thrift*, http://wiki.apache.org/thrift/ (25-11-11)
33. Pingdom, *Exploring the software behind Facebook*, http://royal.pingdom.com/2010/06/18/the-software-behind-facebook/ (25-11-11)
34. MySQL, *Why MySQL?*, http://www.mysql.com/why-mysql/ (25-11-11)
35. Ruturaj.net, *MySQL comparison*, http://www.ruturaj.net/redis-memcached-tokyo-tyrant-mysql-comparison (25-11-11)
36. Wikipedia, *NoSQL*, http://en.wikipedia.org/wiki/NoSQL (25-11-11)
37. Highscalability.com, *What the heck are you actually using NoSQL for?*, http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html (25-11-11)

38. Paul Saab, *Scaling memcached at Facebook*, http://www.facebook.com/note.php?note_id=39391378919 (25-11-11)
39. Memcached, *A distributed memory object caching system*, http://memcached.org/about/ (25-11-11)
40. Couchbase, *Membase Server*, http://www.couchbase.com/products-and-services/membase-server (25-11-11)
41. Redis, *Redis Documentation*, http://redis.io/documentation (25-11-11)
42. Redis, *Retwis*, http://redis.io/topics/twitter-clone (25-11-11)
43. Kristóf Kovács, *NoSQL Comparsion*, http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis (25-11-11)
44. Redis, *Persistence*, http://redis.io/topics/persistence (25-11-11)
45. Redis, *Virtual Memory*, http://redis.io/topics/virtual-memory (25-11-11)
46. Redis, *Replication*, http://redis.io/topics/replication (25-11-11)
47. Ori Pekelman, *Which PHP Library to use with Redis?*, http://dev.af83.com/af83-open-source-projects/which-php-library-use-redis-benchmark/2011/01/01 (25-11-11)
48. Nicolas Favre-Felix, *PhpRedis*, https://github.com/nicolasff/phpredis (25-11-11)
49. Wikipedia, *eAccelerator*, http://en.wikipedia.org/wiki/EAccelerator (25-11-11)
50. 2bits.com, *Benchmarking Drupal with PHP op-code caches*, http://2bits.com/articles/benchmarking-drupal-with-php-op-code-caches-apc-eaccelerator-and-xcache-compared.html (25-11-11)
51. PHP Manual, *Multibyte String,* http://www.php.net/manual/en/intro.mbstring.php (25-11-11)
52. Markus Kuhn, *UTF-8 and Unicode*, , http://www.cl.cam.ac.uk/~mgk25/unicode.html (25-11-11)
53. MaxMind, *GeoIP PHP API*, http://www.maxmind.com/app/php (25-11-11)
54. SimpleTest, *Unit Testing for PHP*, http://www.simpletest.org/ (25-11-11)
55. Ellen Messmer, *Windows Server vs. Linux*, http://www.networkworld.com/news/2010/060710-tech-argument-windows-server-linux.html (25-11-11)
56. Linode, *Features*, http://www.linode.com/features.cfm (25-11-11)
57. Linode, *VPS Guides and Technical Articles*, http://library.linode.com/ (25-11-11)
58. Eivind Uggedal, *VPS Performance Comparison*, http://journal.uggedal.com/vps-performance-comparison/ (25-11-11)
59. nu.nl, *Facebook dicht bij akkoord rond privacy*, http://www.nu.nl/internet/2665433/facebook-dicht-bij-akkoord-rond-privacy.html (25-11-11)

60. nu.nl, *De lekkende overheid*, http://www.nu.nl/column-zaterdag/2642121/lekkende-overheid.html (25-11-11)

61. Wikipedia, *Man-in-the-midle attack*, http://en.wikipedia.org/wiki/Man-in-the-middle_attack (25-11-11)

62. Thomas Pornin, *Do any security experts recommend bcrypt for password storage (top rated answer)*, http://security.stackexchange.com/questions/4781/do-any-security-experts-recommend-bcrypt-for-password-storage (25-11-11)

63. Coda Hale, *How To Safely Store A Password*, http://codahale.com/how-to-safely-store-a-password/ (25-11-11)

64. Andrew Johnson, *Generating Session IDs*, http://www.itnewb.com/tutorial/Generating-Session-IDs-and-Random-Passwords-with-PHP (25-11-11)

65. Chris Shiflett, *Essential PHP Security (Chapter 4)*, http://phpsecurity.org/ch04.pdf (25-11-11)

66. Password Advisor, *Check password strength using PHP*, http://passwordadvisor.com/CodePhp.aspx (25-11-11)

APPENDIX B – PRODUCT BACKLOG

1. BASISFUNCTIONALITEIT (ALPHA*)

1. Functionaliteit om een bedrijf(saccount) aan te maken. Dus info (bedrijfsgevens, logo, enzovoorts) omzetten naar database. Er bestaat dan al een account voor het bedrijf.
2. Een bedrijf moet meteen al minimaal 3 connecties aangeven: chains aanmaken dus (al bestaand of niet: liever nieuwe in het begin). Chains dus in database kunnen stoppen. Daarna moet er ook (handmatig) gechaint kunnen worden (altijd request-accept: closed-loop opt-in).
3. Het volgende is de wall: in eerste instantie gewoon het plaatsen van tekstberichten op de wall van een bedrijf. Andere users moeten dit kunnen zien e.d. Public/private berichten moeten al ondersteund zijn.
4. Direct messages: berichten kunnen pushen naar een bedrijf (private).
5. Opzet home page: (relevante) berichten van de walls van je chain zien en de gepushte berichten.
6. Mogelijkheid tot aanmaken bedrijfsafdelingen: krijgen een eigen wall e.d., en moeten goed gekoppeld worden aan het 'moederbedrijf'.

Uitgangspunt: alleen bedrijven schrijven op hun eigen wall, waarop anderen dan kunnen reageren en liken, ook zijn directe (prive) messages mogelijk, maar dan wel puur naar bedrijven (niet personen). De homepage verzorgt dan het overzicht voor een bedrijf met behulp van alle andere walls (uit de chain) en ook de (private) messages.

**Alleen intern testen met het team. We betrekken hierbij geen externen.*

2. GEAVANCEERDE FUNCTIONALITEIT (CLOSED BETA*)

1. Uitbreiding wall functionaliteit: mogelijkheid tot public/private commentaar op berichten en like functie (mogelijk 'report spam' ofzo ipv. unlike, of recommended ofzo?). Bedrijven moeten bij elk bericht op hun wall kunnen aangeven of public comments wel toegestaan zijn: controle of je eigen wall! Ook moeten bedrijven de berichten kunnen verwijderen e.d.
2. Systeem van accountrechten. Wie mag wat doen in een bedrijf en/of afdeling(en). Dus nu ook support voor meerdere users binnen 1 bedrijf. +soort spectator functie? Niks mogen posten maar wel alles kunnen bekijken. Dit kan misschien ook 1 account zijn. Over nadenken! Misschien max 5 intern wel zien wie wat post op een wall, extern niet.
3. Uitbreiding home page: alleen relevante zaken en dergelijke tonen in een duidelijk overzicht.
4. Suggesties voor mogelijke chains krijgen. Algoritmes dus! Verder ook chains kunnen uitbreiden met behulp van externe media (facebook, linkedin, lokaal adresboek, enzovoorts).
5. Uitgebreide feedback aan de user weergeven, bijvoorbeeld wanneer een chain van jou met een nieuw bedrijf chaint.
6. Ondersteuning voor bedrijfslogo' en zaken als attachments bij berichten.

**Een aantal speciaal geselecteerde bedrijven mag de gesloten beta gebruiken, dit zal voornamelijk om interne investeerders gaan.*

3. Afronding (open beta*)

Zaken die voor de user customizable moeten zijn usercustomizable maken (instellingen).

1. Posten van foto's/video's, enzovoorts op een wall.
2. Mogelijk: algemeen iets van 'chain news' ofzo (de 'chain wall'?).
3. Chains op meerdere manieren visualiseren (Undirected Graph, Geografisch etc.)

**Alleen bedrijven die een account hebben kunnen andere bedrijven uitnodigen voor het netwerk, het is niet mogelijk als extern bedrijf op CHAINels een account aan te maken.*

4. Release (release candidate)

1. Controleer alle functionaliteiten: werkt alles zoals bedoeld.
2. Extra mogelijkheden naar betaling zoals: gesponsorde Demand & Supplies.
3. Future-proofing van het systeem: indien nodig MySQL erbij betrekken als storage van oude berichten, de server structuur goed vaststellen, enzovoorts! (optimalisatie)
4. Indien er geen grote bugs meer worden ontdekt zal CHAINels officieel uitgebracht worden.

5. Post release

1. App ontwikkelen (Apple, Android, Blackberry)
2. Auto Complete, meer invoervelden ophalen en daardoor beter valideren door middel van bijvoorbeeld het KVK nummer.
3. Talen uitbreiden (Spaans, Frans, Duits etc.)
4. Geavanceerde algoritmes verbeteren en/of uitbreiden.

6. Aandachtspunten

- Schaalbaarheid (dataverkeer)
- Snelheid
- Foutafhandeling/preventie
- Beveiliging
- Privacy
- Modulariteit
- Multi-language
- Aanpasbaarheid (content management later)
- Geoptimaliseerd voor Google

APPENDIX C – CONCEPT SYSTEEMONTWERP

1. SUBSYSTEMEN

Alle benodigdheden dienen zoveel mogelijk in (sub)systemen opgedeeld te worden om een overzicht te krijgen van wat er moet gaan gebeuren. Een voorstel:

- Website GUI (voor designteam)
  - User Interface (design)
  - User Interface (implementatie: koppeling aan systeem)
- Website Systeem (front-end)
  - Bedrijven
  - Bedrijfsafdelingen (+subcommunities?)
  - Persoonlijke accounts
  - Rechtenbeheer
  - 'Wall' systeem (berichten, uploaden, enzovoorts)
  - (Direct) Messaging systeem
  - Homepage (overzicht)
  - Chain systeem (connecties)
  - Customization (userinstellingen)
  - Overige zaken mbt. een profiel (events, marketplace, enz.)
- Website Beheer (back-end)
  - Schaalbaarheid, optimalisatie en controle (failsafe)
  - Server(s)
  - Beveiliging (ook mbt. privacy)
  - Google optimalisaties
  - Databasebeheer
  - Statistieken
  - (Content) Management
  - Multi-langual support

2. AANDACHTSPUNTEN PER ONDERDEEL

Dit is slechts een basis, en zou nog flink aangepast en/of uitgebreid kunnen worden

- Bedrijven: hieronder valt het aanmeldingsproces, de verschillende opties die een bedrijf later nog dynamisch kan veranderen, een zoekmogelijkheid binnen bedrijven, mooie overzichten hiervan, en meer.
- Bedrijfsafdelingen: ook weer het aanmaakproces en de verschillende opties die hier weer voor kunnen gelden. Ook moeten de afdelingen bij een bedrijf op een goede manier kunnen weergeven: het linken moet op een goede manier gebeuren.
- Persoonlijke accounts: het aanmeldingsproces, het inlogsysteem, de koppeling met bedrijven, enzovoorts.
- Rechtenbeheer: wie mag wat doen binnen een bedrijf(safdeling), en hoe is dit eenvoudig in te stellen.
- Wall systeem: berichten plaatsen, foto's/video's uploaden/linken, hoe een overzicht wordt gecreëerd (recent/belangrijk voor jou), onderscheid tussen bedrijfs/afdelingswall, enzovoorts. Ook een systeem van public/private reply's en mogelijk een soort like functie moet hierbij komen!
- Homepage: een overzicht van wat voor jou relevant is, ook een soort 'wall' dus. Algoritmes!
- Messaging systeem: gerichte, specifieke berichten naar andere personen sturen, naar bedrijven/bedrijfsafdelingen, verschillende berichttypen, enzovoorts.

- Chain systeem: connecties goed kunnen zien, suggesties voor nieuwe connecties met allerlei factoren in acht nemend, connecties op een goede manier kunnen maken (wie met wie?), en meer. Misschien later?
- Customization: alles wat een user kan aanpassen om zijn zicht op de site te veranderen. Hieronder vallen dus geen bedrijfsinstellingen o.i.d.
- Overige zaken: naast een wall en direct messaging willen we waarschijnlijk meer systemen hebben zoals een event kalender, een marketplace, en mogelijk nog meer. Kan later.
- Schaalbaarheid en dergelijke: alles moet snel draaien, schaalbaar zijn, niet teveel dataverkeer opslurpen, enzovoorts. Optimalisaties!
- Server(s): serverbeheer wordt een 'big issue' zodra het dataverkeer gaat toenemen. De server(s) moeten goed ingesteld en beheerd worden, mogelijk zelfs met behulp van cloud diensten (als Amazon EC2) en dergelijke.
- Beveiliging: gegevens moeten natuurlijk goed beveiligd worden. Ook de privacy van gebruikers (bedrijven) moet gewaarborgd worden. Hieronder valt dus ook een systeem waarmee de privacy goed beheerd kan worden door users zelf, maar ook simpele zorg dat SQL queries niet gehackt kunnen worden o.i.d.
- Google; de belangrijke zoekmachines. Bedrijven zullen het vast fijn vinden als ze via Chain ook goed in Google kunnen komen. Chain zelf (als homepage) wil natuurlijk ook goed in Google komen.
- Databasebeheer: belangrijk om apart te nemen. De databasestructuur, en daarna het beheer ervan zijn erg belangrijk. Welke databasestructuur gebruik je waarvoor, welke data kan je misschien weggooien, hoe wordt toegang tot data op een efficiënte manier geregeld, enzovoorts.
- Statistieken: misschien niet zo belangrijk, maar een soort makkelijke toegang tot allerlei statistieken zou handig zijn.
- Management: als alles eenmaal opgezet is wil je zo min mogelijk inhoudelijk veranderen. Met van te voren een soort CMS in gedachten moet er voor gezorgd worden dat aanpassingen later zo eenvoudig als mogelijk is kunnen worden gedaan, zonder downtime voor de user. Modulariteit is dus een belangrijk punt hier!
- Multi-langual: Engels is de basis, maar veel gebruikers vinden het erg lekker om ook de site in hun eigen taal te kunnen gebruiken. Een goed systeem voor meertaligheid van de site is dus vereist
- Website GUI: een groot onderdeel. Alle schermpjes, knopjes, logo's, enzovoorts moeten natuurlijk ontworpen worden, en daarna gekoppeld aan het gebouwde systeem. Het systeem moet hiervoor dan ook eenvoudig toegankelijk zijn.

APPENDIX D: MEDIA-AANDACHT

1. BNR

Datum: 1-11-2011

Radio interview van circa twee minuten.

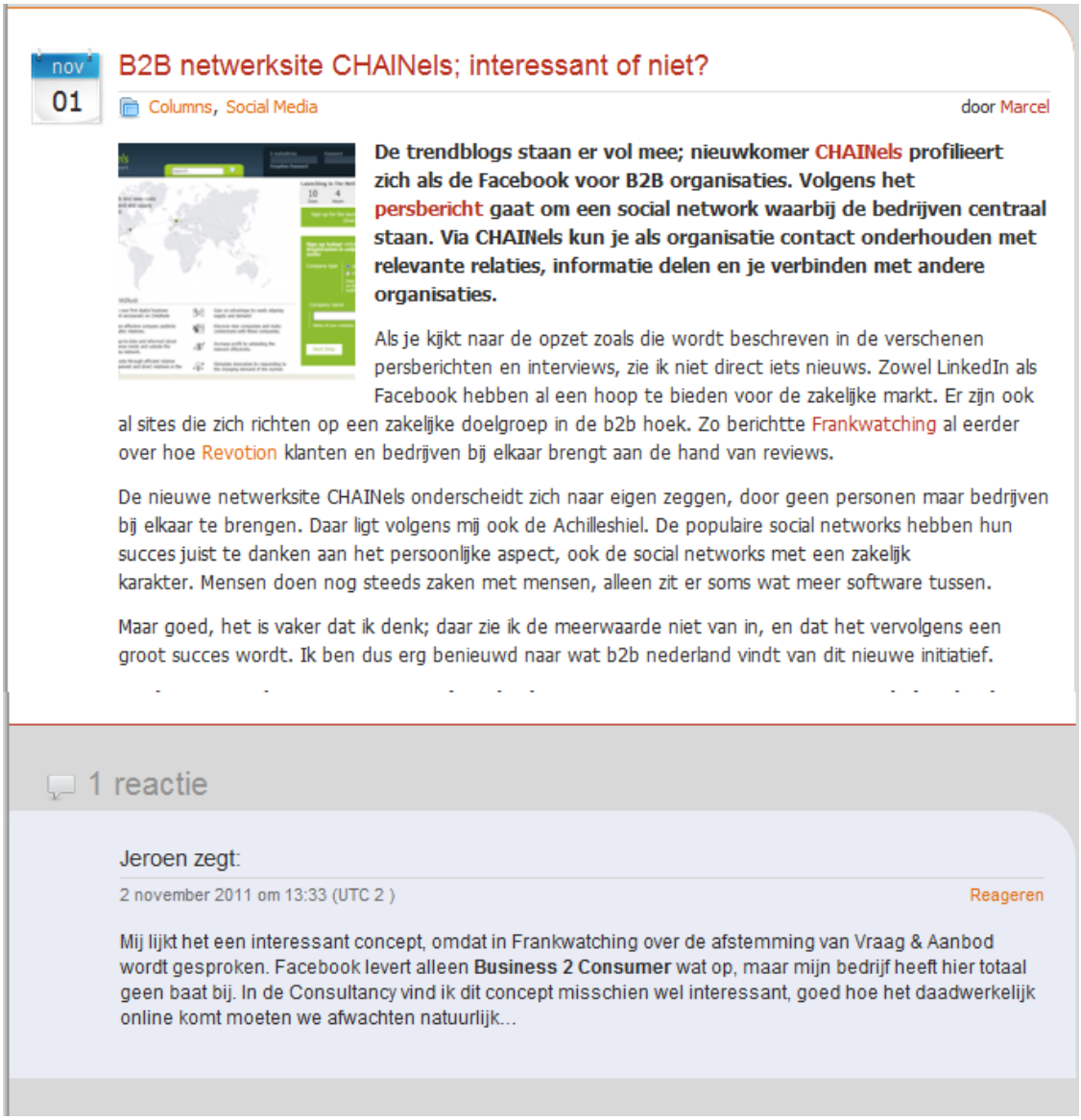
2. B2BMARKETEERS

Bron: http://www.b2bmarketeers.nl/2011/11/b2b-netwerksite-chainels-interessant-of-niet/

Auteur/contactpersoon: Marcel Nanning

E-mail: marcel@b2bmarketeers.nl

Datum: 1-11-2011



nov 01

B2B netwerksite CHAINels; interessant of niet?

Columns, Social Media door Marcel

De trendblogs staan er vol mee; nieuwkomer CHAINels profilieert zich als de Facebook voor B2B organisaties. Volgens het persbericht gaat om een social network waarbij de bedrijven centraal staan. Via CHAINels kun je als organisatie contact onderhouden met relevante relaties, informatie delen en je verbinden met andere organisaties.

Als je kijkt naar de opzet zoals die wordt beschreven in de verschenen persberichten en interviews, zie ik niet direct iets nieuws. Zowel LinkedIn als Facebook hebben al een hoop te bieden voor de zakelijke markt. Er zijn ook al sites die zich richten op een zakelijke doelgroep in de b2b hoek. Zo berichtte Frankwatching al eerder over hoe Revotion klanten en bedrijven bij elkaar brengt aan de hand van reviews.

De nieuwe netwerksite CHAINels onderscheidt zich naar eigen zeggen, door geen personen maar bedrijven bij elkaar te brengen. Daar ligt volgens mij ook de Achilleshiel. De populaire social networks hebben hun succes juist te danken aan het persoonlijke aspect, ook de social networks met een zakelijk karakter. Mensen doen nog steeds zaken met mensen, alleen zit er soms wat meer software tussen.

Maar goed, het is vaker dat ik denk; daar zie ik de meerwaarde niet van in, en dat het vervolgens een groot succes wordt. Ik ben dus erg benieuwd naar wat b2b nederland vindt van dit nieuwe initiatief.

1 reactie

Jeroen zegt:

2 november 2011 om 13:33 (UTC 2) Reageren

Mij lijkt het een interessant concept, omdat in Frankwatching over de afstemming van Vraag & Aanbod wordt gesproken. Facebook levert alleen **Business 2 Consumer** wat op, maar mijn bedrijf heeft hier totaal geen baat bij. In de Consultancy vind ik dit concept misschien wel interessant, goed hoe het daadwerkelijk online komt moeten we afwachten natuurlijk...

3. FRANKWATCHING

Bron: http://www.frankwatching.com/archive/2011/11/01/chainels-het-eerste-sociale-netwerk-voor-b2b/

Auteur/contactpersoon:

E-mail: redactie@frankwatching.com

Datum: 1-11-2011

CHAINels: Het eerste sociale netwerk voor B2B

Door Redactie Gespot van Frankwatching

op dinsdag 1 november 2011 om 10:00 uur

20 Print

CHAINels is een nieuw social networking platform voor business-to-business. Via CHAINels kun je als organisatie contact onderhouden met relevante relaties, informatie delen en je verbinden met andere organisaties. Shahid Suddle vertelt over de gedachte achter deze nieuwe netwerksite.

Wat is CHAINels?

"CHAINels is de eerste internationale business-to-business netwerksite en wordt geïntroduceerd in New York op 1 november 2011. De Nederlandse lancering vindt plaats op de TU Delft op 11 november 2011. In CHAINels is het concept businessmedia verwerkt, de tegenhanger en professionele opvolger van social media. CHAINels heeft unieke eigenschappen ten opzichte van social networks als LinkedIn, Twitter en Facebook. Waar bij deze social networks de persoon centraal staat, staat bij CHAINels het bedrijf, de organisatie of het merk centraal in alle communicatie. Bedrijven, bedrijfsonderdelen en organisaties vormen op deze manier een digitaal en professioneel bedrijfsnetwerk. Een andere uniek voordeel van CHAINels is dat bedrijven vraag en aanbod op elkaar kunnen afstemmen. Dit wordt bereikt met geavanceerde algoritmes. Bedrijven kunnen hun bedrijfsnetwerk hiermee eenvoudig en effectief uitbreiden. Op lange termijn moet dat leiden tot meer winst voor een bedrijf. Oftewel: CHAINels is de Facebook voor bedrijven."

4. COMBRON

Bron: http://www.combron.nl/2011/11/business-network-chainels-mag-je-niet-missen/

Auteur/contactpersoon: Edo Dijkgraaf

E-mail: combron@combron.nl

Datum: 14-11-2011

Business network CHAINels mag je niet missen

Gepubliceerd op: maandag, 14 november 2011 00:45 uur
Auteur: Edo Dijkgraaf
Herzien op: maandag, 14 november 2011 00:45 uur

Afgelopen vrijdag was de kick off. CHAINels het business network is live. De weg voor Shahid Suddle en consorten is nog lang, maar deze hub heeft zeker potentieel.

CHAINels wint het voorlopig nog niet van concurrenten Facebook en LinkedIn, maar wat niet is kan nog komen. Het netwerk onderscheidt zich door de business to business-opzet.

Het idee is eenvoudig. Zowel bij LinkedIn als bij Facebook draait het om persoonlijke contacten. Google+ is net begonnen met uitrollen, maar lijkt dezelfde kant op te gaan. LinkedIn biedt de mogelijkheid aan bedrijven zich te profileren, maar de medewerkers spelen de hoofdrol. Facebookpagina's laten organisaties wel in contact treden met klanten, maar deze klanten bevinden zich bij voorkeur aan het einde van de keten. Dit wordt ook wel b2c, business to consumer, genoemd.

Open beta

CHAINels is live, maar de komende tijd zullen er nog veel zaken veranderen. Net als bij Google+ dienen de gebruikers als testers. Bij CHAINels noemen ze deze hoedanigheid een open beta. Het is niet de enige overeenkomst met Google+. Net als bij de livegang van Google+ dienen de gebruikers andere gebruikers uit te nodigen. Organisaties die zichzelf willen registreren zonder te zijn uitgenodigd krijgen voorlopig niet de mogelijkheid.

Een van de co-founders vertelde dat CHAINels nog zoekt naar iemand die met de communicatie aan de slag gaat. In de open betafase zijn de heren geëxcuseerd, maar het is duidelijk dat de ontwikkeling is gedaan door technici. Zo was er tot vrijdagavond nog niet nagedacht over zaken als een api of een widget die organisaties op hun corporatewebsite kunnen plaatsen, is de hele opzet op dit moment uitsluitend gericht op zendende communicatie, is de grammatica op de website nog niet optimaal en is het nog niet mogelijk profielpagina's in verschillende talen aan te maken. Er is al wel nagedacht over een app voor de telefoon, maar deze is nog niet gereed.

Een voordeel van de dominantie van de technici is terug te zien in het feit dat er niet gekozen is voor een MySQL-database. Het systeem gebruikt door CHAINels is opmerkelijk veel sneller.

Waarom je business network CHAINels niet mag missen

CHAINels bevat een aanzienlijk aantal issues die moeten worden verbeterd. Er is één reden waarom bedrijven

5. BNETTV (NEW YORK)

Bron: http://www.youtube.com/watch?v=7PqWLZJgXXg

Auteur/contactperso on: Edo Dijkgraaf

Datum: 14-11-2011

6. OPHAKKEN

Bron: http://www.ophakken.nl/chainels/

Auteur/contactpersoon: Claudia Angenent

E-mail: redactie@ophakken.nl

Datum: 1-11-2011

Chainels

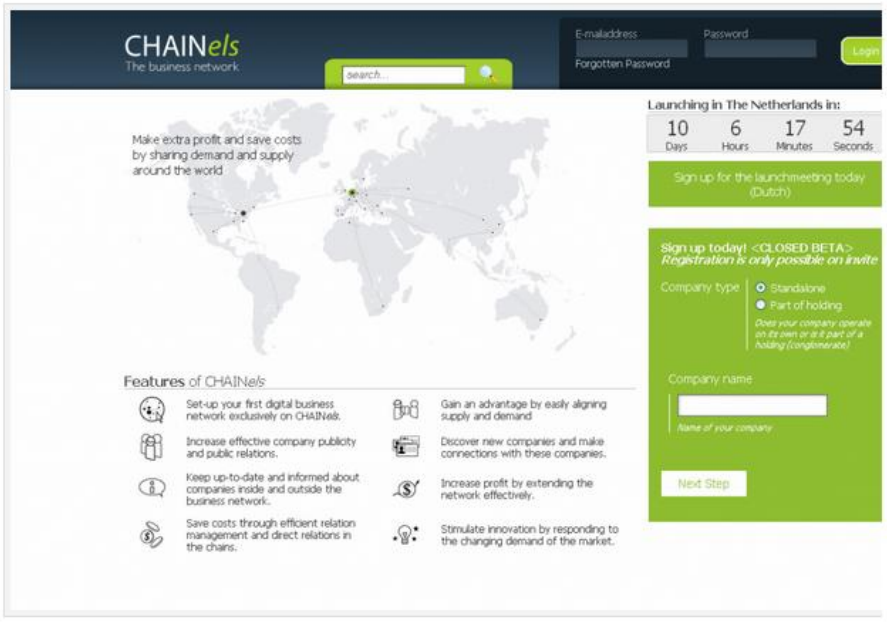
BY CLAUDIA ANGENENT – 1 NOVEMBER 2011

POSTED IN: SOCIAL MEDIA

Like 4 Tweet 1 Share 1

Veel grote bedrijven en merken zijn wel te vinden op de standaard social media platformen die we vandaag de dag kennen. Toch zijn Twitter, Facebook & Linkedin helemaal niet zo geschikt voor Business 2 business contact. Deze kanalen zijn meer bedoeld voor de business 2 consumer communicatie. Een fan-base creëren, de klant informeren en vooral je merk hypen.

Toch was er kennelijk ook behoefte aan een zakelijk netwerk. Vandaag word in New York het platform Chainles bekend gemaakt. Dit netwerk is bedoeld voor business 2 business contact.



APPENDIX E: LOGBOEK

Wij hebben hier twee gemiddelde weken gepakt uit het logboek dat wij hebben bijgehouden, om aan te geven hoe onze gemiddelde werklast is geweest. Sommige onderdelen zijn ingekort om een overdaad aan tekst te voorkomen. (…) staat dan ook voor enzovoorts.

ERWIN

Datum	Tijd	Uren	Beschrijving	SVN Revision(s)
8-8-2011	13:00 - 14:00	1	tel. Shahid, ivm designer en werk achterstand designteam, overleg vincent ivm contract/voortgang en implementatie	-
9-8-2011	15:00 - 17:00	2	TODOS afmaken, unit test fixe! (door aanpassingen) lists implementeren!	-
10-8-2011	07:30 - 13:30	6	lists, fixes in entity classes, unit tests, update PHP en XAMPP (nu PHP5,3)	Rev. 119, 126
10-8-2011	14:00 - 17:00	3	bugfixes.. en php optimalisation, overleg met Vincent over performance en verder implementatie issues	-
10-8-2011	20:00 - 22:00	2	validation country dependent! (design..)	Rev. 128, 129
11-8-2011	11:00 - 12:00	1	validation implementeren	Rev. 138
11-8-2011	15:00 - 17:00	2	validation fixen require_once bug.., overleg vincent om redundantie uit de code te krijgen	-
11-8-2011	20:00 - 22:00	2	implementatie op besproken manier	-
12-8-2011	08:30 - 16:00	7,5	unit tests, bugfixes, overleg vincent: hoe consequent is ons systeem?, document geupdate data klopt nu weer, …	Rev. 143 t/m 147, 149
13-8-2011	13:00 - 18:00	5	bugfixes, overleg met Vincent attribuut country verplaatst, document geupdate, ook modellen gemaakt voor de vergadering maandag, TODO functies geschreven, errors erbij	Rev. 161, 166, 168, 170
13-8-2011	19:00 - 22:00	2,5	Documentatie + modellen, nalopen set methods aan de hand van het model (alleen verplichte attributen empty error genereren)	Rev. 173, 178
14-8-2011	11:30 - 18:00	6,5	Documentatie, language probleem oplossen!, Redis documentatie lz (zie literatuur)	-
	Totaal:	**40,5**		

Logboek Erwin Buckers (1 week)

VINCENT

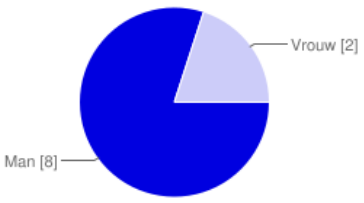
Datum	Tijd	Uren	Beschrijving	SVN Revision(s)
3-10-2011	14.30 - 17.30	3	Bekijken werk Remi. Wat bugs fixen op zijn commentaar. Flyspray doorgekeken en flink wat bijgewerkt…	Rev. 583 t/m 587, 592, 599
3-10-2011	19.00 - 03.00	8	Samen met (bij) Erwin en Remi (via Skye) gewerkt. Veel getest en veel bugfixes gedaan. Aantal functies geschreven die Remi nog nodig had. Authenticatie wat omgeschreven (….)	Rev. 604, 607, 608, 611, 613 …
4-10-2011	11.00 - 18.00	7	Nog wat meer bugfixjes/verbeteringen. Vooral veel getest en vele feedback/contact met Remi gehad over de UI…	Rev. 644 t/m 649
5-10-2011	08.00 - 18.00	10	En weer veel bugfixes/verbeteringen! Standaard logo, Engels helemaal nagelopen, activatie gefixt, enzvoorts…	Rev. 679 t/m 685, 687 t/m 689, 691 t/m 718
6-10-2011	09.30 - 14.30	5,5	Dingen van Remi uit de trunk naar de beta gemerged. Omdat er nog meer in het multi-language systeem staat weer nog meer Engels verbeterd…	Rev. 734, 736, 737, 739 t/m 741
6-10-2011	16.45 - 17.45	1	Nog een aantal bugs in de beta gefixt na weer een merge van Remi's werk	Rev. 753 t/m 757
6-10-2011	22.15 - 23.15	1	Weer een merge, en weer een aantal bugs gefixt na de beta nog even helemaal nagelopen te hebben aangezien die morgen online gaat! Is gelukt :)	Rev. 768 t/m 772
7-10-2011	10.00 - 16.00	6	Goed online zetten closed beta. Een aantal laatste dingetjes veranderd/gefixt na overleg met Erwin/Remi. Invite mail aan selecte testgroep de deur uit gedaan….	Rev. 774 t/m 779, 781 t/m 783, 785, 786,
	Totaal	**41,5**		

Logboek Vincent Koeman (1 week)

APPENDIX F: ENQUÊTE

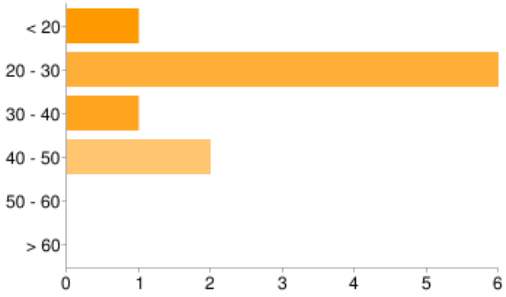
Hieronder staan enkele uitslagen weergeven die volgden uit een enquête die we tijdens de Closed Beta fase hebben gehouden (zie appendix B). Vraag én antwoord zijn hierin verwerkt.

Wat is uw geslacht?



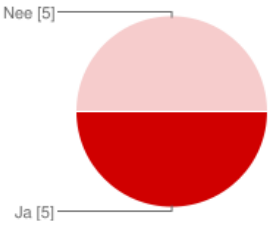
Man	8	80%
Vrouw	2	20%

Wat is uw leeftijdscategorie?



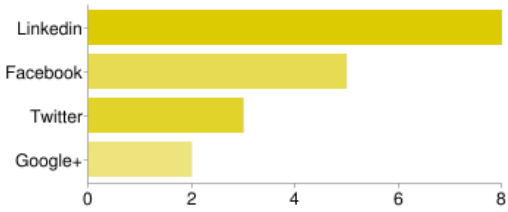
< 20	1	10%
20 - 30	6	60%
30 - 40	1	10%
40 - 50	2	20%
50 - 60	0	0%
> 60	0	0%

Heeft u een eigen bedrijf?



Ja	5	50%
Nee	5	50%

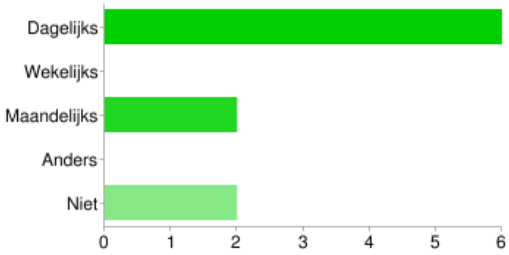
Welke sociale netwerken gebruikt u?



Linkedin	8	100%
Facebook	5	63%
Twitter	3	38%
Google+	2	25%

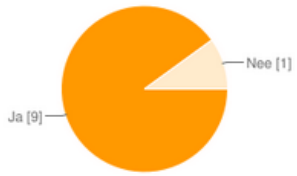
People may select more than one checkbox, so percentages may add up to more than 100%.

Hoe vaak gebruikt u gemiddeld de bovenstaande netwerk(en)?



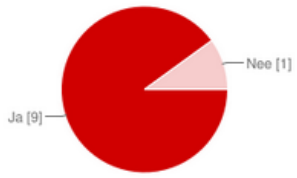
Dagelijks	6	60%
Wekelijks	0	0%
Maandelijks	2	20%
Anders	0	0%
Niet	2	20%

Het is duidelijk hoe ik nieuwe connecties kan maken (Chains)



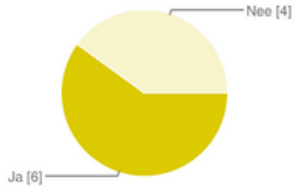
Ja	9	90%
Nee	1	10%

Het is duidelijk hoe ik een reactie op een bericht kan plaatsen



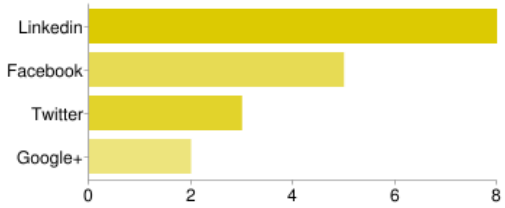
Ja	9	90%
Nee	1	10%

Het navigeren op de website is/lijkt ingewikkeld



Ja	6	60%
Nee	4	40%

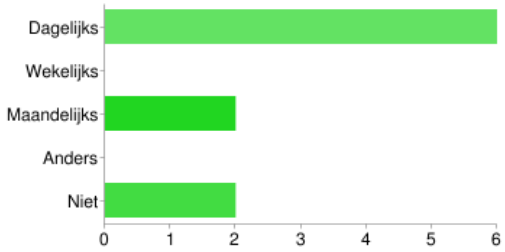
Welke sociale netwerken gebruikt u?



Linkedin	8	100%
Facebook	5	63%
Twitter	3	38%
Google+	2	25%

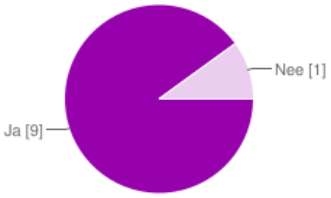
People may select more than one checkbox, so percentages may add up to more than 100%.

Hoe vaak gebruikt u gemiddeld de bovenstaande netwerk(en)?



Dagelijks	6	60%
Wekelijks	0	0%
Maandelijks	2	20%
Anders	0	0%
Niet	2	20%

Heeft u (actief) deelgenomen aan de Closed Beta van CHAINels?

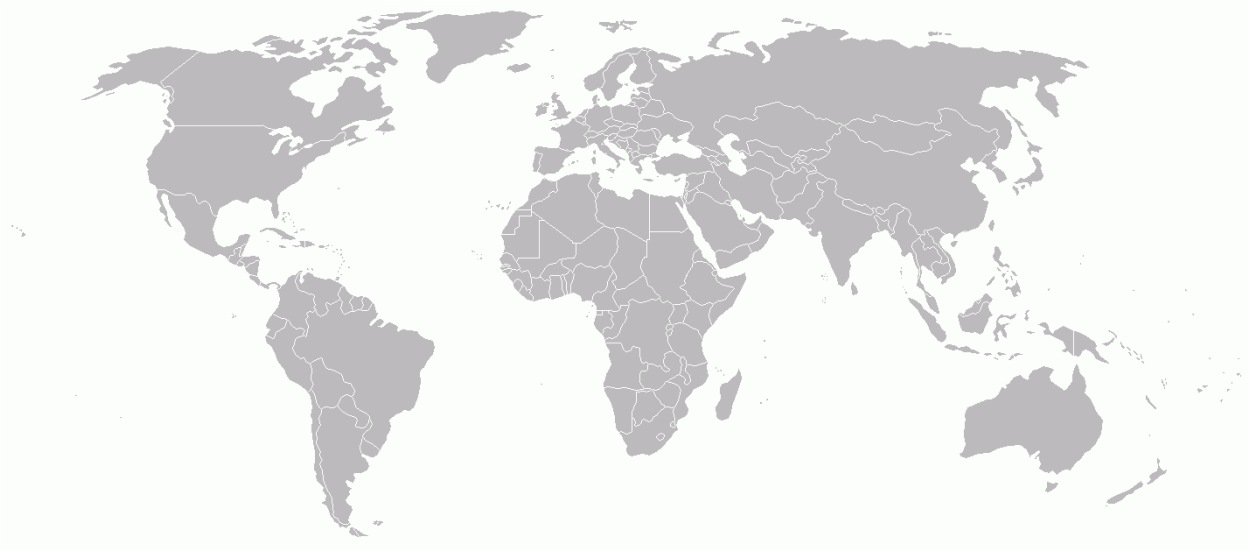


Ja	9	90%
Nee	1	10%

APPENDIX G: BUSINESSPLAN

Het eerst ontwikkelde document waarin het concept van Chain (nu CHAIN*els*) nader wordt toegelicht. Dit document is in beginsel door dr.ir. E.G.J. Blokhuis opgesteld.

CHAIN - The Business Network



Erik Blokhuis

Januari 2011

INTRODUCTIE

De schrijver van dit business plan is Erik Blokhuis. Hij is in juni 2010 gepromoveerd aan de Technische Universiteit Eindhoven. Het onderwerp dat centraal staat in zijn proefschrift is de interactie tussen stakeholders tijdens de herontwikkeling van verouderde bedrijventerreinen. Zijn persoonlijke interesses liggen op het gebied van keuzegedrag, samenwerkingsprocessen, networking, new media en operations research.

Een belangrijke conclusie uit het proefschrift heeft de eerste aanleiding gegeven tot het schrijven van dit business plan. Dit betreft de noodzaak tot het aanleggen, beheren en optimaliseren van netwerken tussen stakeholders, enerzijds om samenwerkings- en bedrijfsprocessen stabieler en beheersbaarder te maken, en anderzijds om de productieketen te kunnen optimaliseren. We kunnen dit concept ook breder zien, en concluderen dat netwerken essentieel zijn voor alle bedrijven die een schakel vormen in een productie- of procesketen.

Echter, de mogelijkheden tot beheersing en optimalisatie van dergelijke 'professionele' netwerken worden nog niet ten volle benut. Wanneer de analogie wordt gelegd naar andersoortige netwerken, zoals sociale netwerken, zien we wel veel activiteiten. Een goed voorbeeld is Facebook, een virtuele gemeenschap waar (meer dan 500 miljoen) individuen hun persoonlijke social network onderhouden. Dit business plan omvat het voorstel tot het opstarten van een virtuele gemeenschap voor professionele netwerken: CHAIN, the business network.

1. PRODUCTANALYSE

Overal ter wereld, en in elke bedrijfstak, hebben bedrijven elkaar nodig. Samen vormen ze productie- en procesketens, waarin de output van het ene bedrijf de input voor een ander vormt, en waaruit eindproducten voortvloeien. Echter, netwerken tussen bedrijven worden tot op heden op zeer ongestructureerde wijze geconstrueerd, voornamelijk op basis van bestaande contacten en contracten. Slechts sporadisch worden goede structuren tussen bedrijven en bedrijfsprocessen gecreëerd op een onderbouwde wijze.

In de praktijk zijn een aantal succesvolle gestructureerde bedrijfsnetwerken te onderscheiden. Een voorbeeld hiervan is The High Tech Campus in Eindhoven. Dit fysieke netwerk van bedrijven heeft een positieve invloed op de prestaties van de netwerk-partners, omdat processen van verschillende bedrijven op elkaar worden afgestemd en geoptimaliseerd. Andere succesvolle gestructureerde bedrijfsnetwerken zijn Kalundborg in Zweden en Silicon Valley in de Verenigde Staten. De georganiseerde structuur van deze professionele netwerken is zeer belangrijk voor het succes van de individuele deelnemers.

Maar waarom moeten dit fysieke netwerken zijn? Waarom moeten bedrijven elkaars buurman zijn om de bedrijfsprocessen te optimaliseren? En waarom is dit slechts toegankelijk voor een zeer kleine groep van meestal technisch georiënteerde bedrijven? Het hier beschreven productidee is het oprichten van een "virtuele High Tech Campus"; een virtuele gemeenschap waarin professionele netwerken worden opgebouwd, beheerd, en geoptimaliseerd. Maar dan wel een virtuele campus voor álle soorten bedrijven!

Product

Het productidee kan worden samengevat als 'Facebook voor bedrijven', onder de merknaam CHAIN. De aanleiding hiervoor is het idee van 'the six degrees of separation', gecombineerd met de ontwikkeling richting global village.

Six degrees of separation is een theorie van Karinthy die stelt dat iedereen op onze planeet is met elkaar verbonden via een netwerk met zes niveaus; via een netwerk met maximaal vijf tussenschakels is iedereen met iedereen te verbinden. Daarnaast is the global village voorspeld door McLuhan in 1962. Hij beschreef een trend van massamedia die de tijds- en plaatsbarrières van menselijke communicatie wegneemt, waardoor mensen op een mondiale schaal kunnen communiceren. Ook in het bedrijfsleven zien we dit in toenemende mate toegepast worden; bedrijven besteden via internet hun boekhouding uit in India, en softwarebedrijven laten hun computerprogramma's in China ontwikkelen.

CHAIN biedt een virtuele omgeving waarin ieder bedrijf de mogelijkheid krijgt zijn eigen network-account op te richten. Dit network-account wordt gevuld door de bedrijven zelf; interne bedrijfsonderdelen zorgen voor de input. Het doel van het totale netwerk is de optimalisatie van netwerken van bedrijfsprocessen: optimizing and completing the business cycle. Hiervan profiteren de individuele netwerk-connecties vooral.

De virtuele omgeving is opgebouwd uit drie boxes:

1. In de eerste box, de CHAIN-position, positioneert het bedrijf zichzelf binnen een business chain. Twee onderdelen zijn hierin cruciaal: de demand en de supply van het bedrijf. Onder het kopje demand wordt verstaan de prestatie-eisen voor de input in het bedrijfsproces (personeel, materiaal, energie, huisvesting, producten, et cetera). Onder het kopje supply vallen beschrijvingen van eindproducten, afzetmarkten en restproducten.

2. In de tweede box, de CHAIN-connections, worden de huidig gerealiseerde netwerkconnecties van het bedrijf weergegeven. Dit kan worden beschouwd als het onderdeel "Friends" binnen de Facebook community. In deze box kunnen business updates door bedrijven worden geplaatst, waarna deze updates leesbaar zijn voor alle connections in de opgebouwde community. Ook kunnen bedrijven elkaar aanbevelen door middel van recommendations. De bestaande connecties worden enerzijds gegenereerd door de bedrijven zelf, en anderzijds worden suggesties gedaan door informatie van het internet te verzamelen. Daarvoor worden spiders gebruikt: bots die het internet op een methodische en geautomatiseerde manier doorbladeren op zoek naar relevante informatie (bijvoorbeeld het onderdeel 'Partners' op websites van bedrijven). (Zie o.a. Maltego)

3. De derde box is bestemd voor CHAIN-possibilities. Daarin worden voorstellen gedaan voor mogelijke netwerkuitbreidingen; er wordt op basis van de ingevoerde demand en supply business keten gezocht naar interessante nieuwe connecties. Deze box is gebaseerd op geavanceerde algoritmes, waarmee verbanden worden gevonden tussen bedrijven, aan de voor- en achterkant van de business keten.

Additionals

Er worden een aantal mogelijke services geboden aan CHAIN connections:

- Creëren van sub-communities, bijvoorbeeld voor langlopende projecten
- Private / public messages
- Blogging
- News feeds
- Events
- Marketplace
- Bedrijfsprofielen kunnen worden afgesloten voor (delen van) het netwerk
    - Profielen kunnen bereikbaar worden voor 1st, 2nd, and 3rd+ level connections
    - Bedrijven bepalen zelf welke input ze geven op gebied van supply and demand. Privacy-gevoelige informatie hoeft uiteraard niet geplaatst te worden; bedrijven bepalen zelf hun CHAIN-position en op basis daarvan worden bestaande en mogelijke netwerken gecreeerd
- Bedrijven kunnen zelf instellen op welke schaal nieuwe CHAIN connections worden voorgedragen
    - Afstand
    - Innovativiteit
    - Capaciteit, etc.
- Ondersteuning bij het ontwikkelen van bedrijfs-accounts

In de toekomst is het zeer interessant om uit te breiden naar ordering: Als bedrijven in een netwerk met elkaar verbonden zijn, kunnen er via XML berichten orders geplaatst worden. Daarbij kan worden ingesteld dat voor producten die via CHAIN worden besteld een bepaalde korting geldt (gebruik van hotelbon principe / www.groupon.nl)...

Technische details

Het achterliggende systeem moet worden gezien als een 3-dimensionaal netwerk. Op de eerste as van het netwerk staat de type-codering van het bedrijf (volgens ISIC: International Standard Industrial Classification of All Economic Activities). Op de tweede as staan de specialismen van het bedrijf. Op de derde as worden geografische afstanden weergegeven. Bedrijven worden gepositioneerd in dit 3-D netwerk. Connecties worden gelegd met bedrijven die zorgen voor input in het bedrijfsproces, of die de output uit het bedrijfsproces afnemen.

Op de uiteindelijke interface moet het netwerk van het bedrijf ook 3-D grafisch kunnen worden weergegeven. Dit is noodzakelijk om de bedrijven te ondersteunen om hun doel te bereiken: het optimaliseren van netwerken van bedrijfsprocessen. Een goed voorbeeld van een technische uitwerking van onze visie op netwerkoptimalisatie is te vinden bij het bedrijf Oblong; zij hebben het product G-speak ontwikkeld waarmee zeer duidelijk en snel 3-dimensionale netwerken gevisualiseerd kunnen worden: http://vimeo.com/2229299. Een mogelijke toekomstige uitbreiding voor CHAIN is het inrichten van lokale centra (CHAIN rooms) waar met deze technologie de mogelijkheid wordt geboden aan bedrijven om een goede marktverkenning te doen.

Onder het beschreven 3-dimensionale netwerk ligt de database: het belangrijkste onderdeel van CHAIN. De samenstelling van de database moet vanaf het begin 'smart' zijn ingericht en berekend zijn op grootschalige extensies; voor het ontwerp van de database wordt de kennis van een expert ICT programmeur ingezet. Tevens wordt gezocht naar mogelijke partners op het gebied van database opbouw en beheer (Oracle, MySQL). Dat zal na de launch van CHAIN geschieden.

Tot slot is het principe van Cloud Computing erg interessant voor CHAIN. Web services als Amazon Elastic Compute Cloud (Amazon EC2) leveren resizable capaciteit in de cloud, en is dus uitermate geschikt om maximale flexibiliteit op het gebied van servercapaciteit te realiseren, en om web-scale computing gemakkelijker te maken voor het systeem. Daarvoor wordt gekozen voor Cloud Computing, ondanks de relatief hogere kosten voor serverruimte.

Positionering

CHAIN voegt een nieuw genre toe aan de reeds bestaande network communities. Dit genre wordt gecreeerd door een verdere professionalisering van reeds bestaande omgevingen voor networking. De basis van networking is gelegd door individuele social networks, daarna zijn enkele omgevingen ontwikkeld voor individuele en meer professionele networks, en CHAIN is de eerste die deze professionalisering van networking uitbreidt naar business networking. De belangrijkste onderscheidende factor van CHAIN ten opzichte van bestaande network communities is het feit dat deze bestaande communities worden bevolkt door *personen*. Personen zijn echter niet onvoorwaardelijk verbonden aan bedrijven; de ontwikkeling van een community waarin bedrijven de entiteiten zijn is daarom van grote toegevoegde waarde.

Social (individual) networking	Facebook / MySpace / Friendster	↓ Professionaliteit
Professional (individual) networking	LinkedIn / XING / Ecadamy	
Business networking	CHAIN	

Het volledige product is niet gemakkelijk te beschermen. De onderliggende algoritmen worden onderdeel van een patentaanvraag, maar het algemene idee om een virtuele gemeenschap te ontwikkelen voor professionele entiteiten is kopieerbaar.

Daarom is het zeer belangrijk om gebruik te maken van het first mover advantage. Bedrijven moeten zich, wanneer ze gebruik gaan maken van een dergelijk product, het eerst aanmelden bij CHAIN. De tweede belangrijke stap is dat deze gebruikers nóóit ontevreden mogen raken over het product. Positionering van het product is kortom ingestoken vanuit twee aspecten: het first mover advantage, en het vasthouden van de voorsprong die daardoor ontwikkeld wordt. CHAIN blijft het beste product door continue verbeteringen op het gebied van interface en netwerkstructuur. Hoe CHAIN dit wil bereiken wordt beschreven onder het kopje 'verankering'.

Doelgroep

Het virtuele business netwerk CHAIN is voornamelijk – maar niet uitsluitend – bestemd voor B2B bedrijven. Belangrijke algemene doelstellingen voor toekomstige CHAIN members moeten liggen op het gebied van optimalisering van inkoop en vergroting van de afzetmarkt. Het CHAIN netwerk daagt bedrijven uit om zich écht te positioneren op de internationale markt van bedrijven. Door het kiezen van een goede onderscheidende positie, en daaraan gerelateerd het formuleren van de juiste demand en supply vragen, belanden bedrijven in de juiste netwerken. De voorstellen voor netwerkexpansie zullen kwalitatief en kwantitatief groeien in de tijd, wat uiteindelijk resulteert ineen goedkopere en betere input in het bedrijfsproces, en in een groeiende output. Een ander voordeel voor CHAIN members is de mogelijkheid om snel en gericht zelf gekozen communities te informeren over bepaalde ontwikkelingen, en de ontwikkelingen van andere members in deze communities snel en overzichtelijk te kunnen volgen. In de toekomst kunnen bedrijven ook profiteren van mogelijkheden om via CHAIN bestellingen te plaatsen, en om gebruik te maken van CHAIN rooms met als doel het op efficiënte en effectieve wijze in kaart brengen van de markt.

Distributie

Het CHAIN netwerk ontstaat niet ineens; het moet groeien. In Nederland bestaan al verschillende bedrijvenverenigingen, waarbinnen bedrijven gekoppeld worden op basis van hun locatie, niet op basis van de positie in de bedrijfsketen. De gegevens van deze verenigingen kunnen gekoppeld worden, waarna het aspect van business chains toegevoegd kan worden. CHAIN begint met het in kaart brengen van business chains op kleine schaal – de regio Brainport. Waar liggen bestaande verbanden, en waar liggen mogelijke verbanden? Dit netwerk wordt opgezet met als doel het op gang brengen van een vliegwieleffect. Uitgangspunt hiervoor vormen de LISA database met daarin alle bedrijvigheid in Noord-Brabant, en de contacten bij de Brabantse Ontwikkelingsmaatschappij (BOM), het Brabant Center for Entrepreneurship (BCE), en Brainport Development.

Binnen Brainport is onlangs een nieuwe community ontwikkeld: TU/e Friends of BCE. Rondkijkend in deze community zijn er legio partijen die op zoek zijn naar optimalisatie van productketens: business development groepen bij TNO, High Tech Automotive Campus, Holst, Philips, High Tech Campus Eindhoven, NXP, ASML, etc. Verder van deze Brainport regio hebben onder andere de TU Delft en de TU Twente gelijksoortige initiatieven, met hun eigen specialisaties. Nog verder van huis zijn er initiatieven vanuit diverse Scandinavische landen (o.a. rond Chalmers University Zweden), verschillende activiteiten in Engeland (oa. door de diverse universiteiten in Londen) en diverse clusters in Frankrijk (Parijs, Grenoble).

De al bestaande contacten van vooraanstaande members van deze communities kunnen CHAIN helpen een vliegende start te realiseren. Door eerst het netwerk op te zetten in de Brainport regio, de resultaten en mogelijkheden terug te koppelen aan de ingevoerde Brainport community, en daarna de CHAIN gemeenschap open te stellen voor andere initiatieven (Delft, Enschede, Scandinavie, Engeland, Frankrijk) kan het netwerk worden klaargestoomd voor het brede publiek. Door promotie van het product, eerst op kleine schaal (Brainport) en daarna naar een steeds grotere schaal (Regio Oost-Brabant, provincie Noord-Brabant, Nederland, Europa, Wereld) wordt het gebruik van CHAIN gestimuleerd en wordt het netwerk uitgebreid.

Verankering

De virtuele business community CHAIN moet bij de start een voorsprong hebben op de concurrentie, en moet deze voorsprong vasthouden door continue verbetering van het product. Deze verbetering wordt gewaarborgd door het toevoegen van expert wiskundig programmeur en beveiliger aan het management team. Door het kijken naar verbeteringen op onder andere bestaande social networks moet het product voortdurend vernieuwd en verbeterd worden.

Daarnaast moet gezocht worden naar een expert op het gebied van interface design. De CHAIN omgeving moet een zeer cleane omgeving worden, waarin niet teveel ruimte wordt gegeven voor bedrijven om hun eigen interface design door te voeren. Bedrijven krijgen ruimte voor hun logo in de box CHAIN-position; de andere boxes worden overzichtelijk en strak ontworpen. Dit is analoog aan de Facebook interface, waarin members slechts een zeer kleine profielfoto kunnen uploaden; het grootste gedeelte van een Facebook screen is hetzelfde voor alle members. Echter, de CHAIN interface wordt zakelijker vergeleken met Facebook; in een oogopslag moeten members op de hoogte komen van ontwikkeling in bestaande – en mogelijk toekomstige – netwerken.

2. MARKTANALYSE

In Eindhoven zijn 25.891 bedrijven gevestigd, in de provincie Noord-Brabant zijn dat er 306.471, en in geheel Nederland 1.999.232. Dit zijn individuele adressen; er zitten een aantal doublures in omdat sommige bedrijven meerdere vestigingen hebben. In de database van Dun & Bradstreet, een leverancier van wereldwijde zakelijke informatie en eigenaar van de grootste B2B database, zijn wereldwijd 181.500.000 bedrijven opgenomen, verspreid over ongeveer 200 landen. Dit toont de enorme omvang aan van de professionele markt in de wereld.

In deze professionele markt zijn een aantal belangrijke trends waar te nemen. De meest belangrijkste trends hierin zijn internationalisatie, specialisatie (focussing op core business), marktsegmentatie, verbreding van distributiekanalen, en uitbreiding van concurrentie. CHAIN speelt in op deze belangrijke trends, door bedrijven een mogelijkheid te bieden zichzelf te profileren, marktontwikkelingen te volgen en bedrijfsketens te optimaliseren.

Marktanalyse

Internet en communicatie zijn in toenemende mate van belang voor bedrijven. De mogelijkheden van ICT, internet, en new social media zijn enorm. De meeste bedrijven beschikken inmiddels over een eigen website en in toenemende mate worden ook new social media ingezet ter bevordering van de bedrijfsactiviteiten. Echter, de mogelijkheden op het gebied van networking worden nu slechts beperkt tot het benaderen van potentiele new employees.

Daarom zijn verschillende bedrijven lid van social networks als Facebook of Hyves, en hebben ze company pages op bijvoorbeeld LinkedIn of XING. In de praktijk blijken deze sociale omgevingen helemaal niet of niet helemaal geschikt te zijn voor bedrijven; de potentie van networking wordt zeer beperkt benut. Hier ligt een grote markt voor CHAIN – the business network.

Daarnaast is ook een globalisering waarneembaar in verscheidene bedrijfstakken. Bedrijven zijn niet meer afhankelijk van lokale of regionale markten voor inkoop en afzet van producten. Mede door de ontwikkeling van het internet kunnen bedrijven producten kopen en verkopen over de hele wereld. Daarom moet CHAIN niet lokaal, regionaal, of landelijk worden ingestoken, maar internationaal. Dit biedt de meeste mogelijkheden voor aangesloten CHAIN connections.

Marktpositie

De beschreven markt is vooralsnog door geen enkel bedrijf ontgonnen. De perspectieven zijn daarom goed voor CHAIN. Het streven is om op zeer korte termijn – voor eind december 2010 – de eerste ontwikkelingsstappen te zetten, om daarmee een marktleiderschap te verwerven die daarna niet meer afgestaan wordt.

In de praktijk is gebleken dat de first mover op het gebied van social networks een marktaandeel van 7% van de totale wereldbevolking heeft verworven, en dat binnen 7 jaar tijd. De verwachting bestaat dat dit zal uitbreiden tot ongeveer 10% van de wereldbevolking. Het concept van CHAIN heeft de potentie om hetzelfde te bereiken, maar dan op het gebied van bedrijven. Dit betekent dat er een marktpotentie bestaat van ongeveer 20 miljoen CHAIN connections. De zwaartepunten van CHAIN zullen waarschijnlijk komen te liggen in geindustrialiseerde landen en continenten, zoals Europa, de Noord-Amerika, en Azie.

Verkoopinspanningen

Eigenlijk past de term verkoop niet goed in dit business plan; overtuiging en verleiding zijn termen die de lading beter dekken. Bedrijven moeten worden overtuigd van de toegevoegde waarde van het CHAIN business network, en worden verleid om een CHAIN connection te worden. De aftrap van CHAIN wordt gegeven in Brainport. Met behulp van de aanwezige netwerkorganisaties (BOM, Brainport Development, BCE) wordt het netwerk in deze regio in kaart gebracht. Bedrijven worden geïnformeerd over de mogelijkheden en potenties van het netwerk, en worden actief betrokken bij de ontwikkeling wanneer ze dat willen. Het succes hiervan is bepalend voor de toekomst van CHAIN: hier moet de basis liggen voor de toekomst, bedrijven moeten de meerwaarde van CHAIN inzien, en het woord moet worden verspreid. Dit is het moment waarop het netwerk openbaar wordt gemaakt.

Concurrentieanalyse

Naam	**Beschrijving**	**Omvang**
Biznik	A community of entrepreneurs and small businesses dedicated to helping each other succeed	
Cofoundr	A community for entrepreneurs, programmers, designers, investors, and other individuals involved with starting new ventures	5.540

E.Factor	An online community and virtual marketplace designed for entrepreneurs, by entrepreneurs	810.000
Ecademy	A business network for creating contacts and sharing knowledge	6.414
Entrepreneur Connect	A community by Entrepreneur.com where professionals can network, communicate, and collaborate with others	
Facebook Visa Business Network	The Visa Business Network is a networking site that brings together small business owners to move their businesses forward through today's challenges	100.000
Fast Pitch	A business network where professionals can market their business and make connections	
Focus	A community focused on helping business decision makers and IT professionals make decisions	850.000
InCompany	InCompany is a business social network, business directory, and online marketplace rolled into one	
JASEzone	A professional community where you can find potential clients and business partners	
LinkedIn	A professional network that allows you to be introduced to and collaborate with other professionals.	80.000.000
Loopthing	social networking platform which enables businesses to create and maintain win-win relationships with other businesses and individuals around the world	19.770
Manta	Largest free source of information on small companies, helping business professionals promote their business, sell faster, and make business connections	
Networking for Professionals	A business network that combines online business networking and real-life events	
PartnerUp	A community connecting small business owners and entrepreneurs	150.000
PerfectBusiness	A network of entrepreneurs, investors and business experts that encourages entrepreneurship and mutual success	
Plaxo	An enhanced address book tool for networking and staying in contact	15.000.000
Ryze	A business networking community that allows users to organize themselves by interests, location, and current and past employers	500.000

StartupNation	A community focused on the exchange of ideas between entrepreneurs and aspiring business owners.	
Upspring	A social networking site for promotion and social networking	
Viadeo	Manage your network of professional contacts, identify useful contacts through your extended network, share information and knowledge	30.000.000
Weebiz	Free online B2B center where companies all over the world connect and profit through networking	
XING	A European business network with more than 7 million members	8.000.000
Young Entrepreneur	A forum-based site for entrepreneurs and small business owners who are passionate about promoting business for themselves and others.	
Ziggs	professional connection portal founded on the principles of professionalism and respect	

Het gros van de hierboven vermelde communities omvatten netwerken tussen personen. Daarnaast zijn veel sites gericht op kleine ondernemers, ter ondersteuning van startende entrepreneurs en ter verbetering van business ideas. Betalen is bijna standaard bij deze sites, en de interface is veel te druk en onoverzichtelijk. Op het gebied van netwerken tussen bedrijven zijn er 5 sites actief: Ecadamy, Loopthing, Manta, Viadeo en Weebiz. Echter, deze sites zijn ingericht als een telefoonboek; er worden vaak bedrijfscontacten vanaf het internet gebruikt zonder dat het bedrijf daadwerkelijk member is, en er worden geen 'smart' connecties gelegd. Op dat vlak gaat CHAIN een nichefunctie vervullen; CHAIN is een slim en goed werkende network community waarbij bedrijven daadwerkelijk lid worden om bij te dragen aan succes van het individu en de community.

Er zijn nog geen bedrijven actief in het aanbieden van een slimme virtuele netwerk community voor bedrijven. Omdat het product een webapplicatie omvat, waarbinnen bedrijven zich gratis kunnen inschrijven in de netwerk community, is de onderhandelingsmacht van kopers gering. Door een strategische samenwerking aan te gaan met experts op het gebied van mathematical programming, security, en interface design, worden mogelijke problemen met leveranciers voorkomen.

De belangrijkste bedreigingen voor CHAIN bevinden zich op het gebied van potentiele nieuwkomers en vervangende producten. Om het hoofd te kunnen bieden aan nieuwkomers moet CHAIN gebruik maken van de first mover advantage, en een voorsprong opbouwen op concurrenten die hetzelfde product aan willen bieden. Uit ervaringen van andere networking communities is gebleken dat wanneer users eenmaal aangemeld zijn voor een netwerk, en wanneer ze tevreden zijn over de geleverde services, ze niet snel overstappen naar een ander netwerk. Dit gegeven moet worden benut door CHAIN. Daarnaast bestaat er een dreiging van vervangende producten.

Zo kunnen er netwerken worden ontwikkeld voor specifieke groepen van bedrijven. Daarom moet CHAIN een kwalitatief zeer goed netwerk te ontwikkelen, en kunnen aantonen dat een brede benadering veel meer mogelijkheden biedt dan een benadering op basis van bestaande structuren. Hierop wordt ingespeeld door de hoge kwaliteit en de continue verbetering van het onderdeel CHAIN-possibilities.

3. Visie en Missie

Visie

Het visualiseren en operationaliseren van bestaande bedrijfsnetwerken en het doen van 'smart' aanbevelingen over geschikte uitbreidingen van deze bedrijfsnetwerken, met als doel het optimaliseren van de bedrijfsketens.

Missie

Het bieden van een slimme virtuele omgeving waarbinnen bedrijven zich kunnen positioneren op basis van hun bedrijfsketen, waarna deze bedrijfsketen wordt geplaatst binnen het grotere CHAIN network.

Doelstelling

Binnen 10 jaar moet een netwerk zijn ontstaan waarbij minimaal 10% van de totale bedrijfs-bevolking in de wereld is aangesloten, teneinde voldoende ruimte te hebben voor optimalisatie van bedrijfsketens.

Waarden

De "clean-heid" van het systeem staat centraal. Interface design is strak, alle overbodige informatie en toepassingen worden achterwege gelaten. Daarnaast mag CHAIN geen reclamefolder worden waarin pop-ups en banner ads het gebruiksgemak verminderen.

Value drivers

Secundaire activiteiten					
Infra-structuur		Interface design and linking	Webbased virtuele omgeving voor netwerken	Cooperatie met netwerk-organisaties in Brainport regio	
Personeel & organisatie		Strategische samenwerking met expert o.g.v. mathematical programming, security and interface design			Uitbreiding van personeel op basis van uitbreiding van het netwerk

Technologie		Algoritmes voor bepalen van mogelijk interessante uitbreidingen van de CHAIN connections		Zorgen dat CHAIN hoog eindigt bij zoekresultaten	Continue optimalisatie van interface, algoritmes, beveiliging, inspelend op wensen van de gebruikers	
Inkoop-management		Aankoop domeinnamen	Aankopen van voldoende server capaciteit / kiezen voor Cloud Computing		Uitbreiding van server capaciteit moet tijdig gebeuren; er mogen nooit problemen ontstaan door het bereiken van de grenzen van het systeem	
	Opslag / ontvangst	Productie / uitvoering	Fysieke distributie	Marketing / verkoop	(After) sales service	Primaire activiteiten

4. ORGANISATIE

CHAIN start zeer low profile. Bij de start van het bedrijf staan twee aspecten centraal. Het eerste aspect is de personele organisatie. CHAIN start met drie partners: Erik Blokhuis is verantwoordelijk voor het 'real-world' netwerk, Pascal Spoek is verantwoordelijk voor het ICT netwerk en beveiliging, en voor de opbouw van de database en de 3-D visualisatie wordt nog een derde partner gezocht. Het tweede aspect is de IT infrastructuur. CHAIN behoeft slechts voldoende server capaciteit, en enkele computers om het netwerk te operationaliseren en te optimaliseren.

De investeringen in de IT infrastructuur worden gedragen door de partners. De connecties met de netwerkorganisaties die benodigd zijn om de structuur in Brainport uiteen te leggen bevinden zich in het netwerk van Erik Blokhuis. Hij heeft een interfacultaire opleiding gevolgd waarin delen van de masters Bouwkunde en Bedrijfskunde van de Technische Universiteit Eindhoven zijn gecombineerd. Daarna is hij gepromoveerd aan de TU/e, bij de leerstoel Construction Management and Engineering. De benodigde kennis op het gebied van IT bevindt zich bij Pascal Spoek, afgestudeerd aan de faculteit Technische Natuurkunde van de TU/e. De partners zijn tegelijkertijd ook de enige eigenaren van CHAIN.

Een vestigingsplaats is in eerste instantie niet noodzakelijk; we werken vanuit huis. Op den duur zullen strategische samenwerkingen met andere netwerken, databasebeheerders en websites op de agenda komen te staan. Ook worden in de tijd mogelijkheden verkend om gerichte advertising toe te passen. Dit mag echter geen negatieve gevolgen hebben voor de bruikbaarheid van het netwerk.

Tenslotte zal het bedrijf intensief worden ondersteund door professionals die zijn verbonden aan het Brabant Center for Entrepreneurship (BCE). Deze organisatie heeft tot doel het stimuleren van bedrijvigheid in het algemeen, en startups in het bijzonder, in de provincie Noord-Brabant. Het BCE verstrekt ondersteuning voor starters op verschillende aspecten van ondernemen.

5. ROADMAP

Nog uit te werken!

1. Brainport
2. Oost-Brabant
3. Provincie Noord-Brabant
4. Nederland
5. Europa
6. Wereld

Aantonen van de meerwaarde van een CHAIN membership

Door het zelf ontwikkelen van een netwerk voor Brainport, en daarna door het verspreiden van het nieuws dat CHAIN een nieuwe stap is richting een optimale bedrijfsketen.

6. FINANCIËN

De initiële investeringsomvang voor IT gerelateerde zaken bedraagt 5.000 euro. Daarbij komen maandelijkse kosten voor het gebruik van de Cloud serverruimte. Bij een stevige groei van CHAIN zullen deze maandelijkse kosten sterk stijgen, en zal extra personeel moeten worden aangetrokken om de database te beheren en de website up to date te houden. Echter, in het begin zijn de investeringen zeer laag.

Daartegenover staan de mogelijke revenues door advertising. Daarbij lijken Banned ads en Referral marketing het meest geschikt voor CHAIN. In tegenstelling tot social networking sites bestaat de verwachting dat CHAIN een hoge click through rate (CTR) kan behalen omdat de gebruikers professionele organisaties zijn, omdat zij geïnteresseerd zijn in het optimaliseren van hun bedrijfsketens door middel van het zoeken van nieuwe connecties, en omdat de advertisements heel specifiek kunnen worden afgestemd op het type organisatie / bedrijf. Deze CTR bepaalt voor een groot deel de inkomsten uit de advertising. Daarom verwachten wij een groei die exponentieel gerelateerd is aan het aantal CHAIN members:

- Eerste jaar – 10.000 members, 25.000 euro revenues
- Tweede jaar – 50.000 members, 100.000 euro revenues
- Derde jaar – 100.000 members, 300.000 euro revenues
- Vierde jaar – 250.000 members, 1.000.000 euro revenues
- Vijfde jaar – 1.000.000 members, 10.000.000 euro revenues

Naast jaarlijkse expenditures en revenues is de waarde van een dergelijk netwerk van doorslaggevend belang. De verwachting bestaat dat CHAIN een geweldige waardesprong gaat maken in de eerste vijf jaar, omdat professionele netwerken – vanuit een business perspectief – veel interessanter zijn dan sociale netwerken. Deze sociale netwerken vertolken reeds zeer grote waarde.

Vergelijking

Social Networking Sites als Facebook zijn in de eerste 3 jaar niet winstgevend. De revenues van Facebook lagen in 2008 bijvoorbeeld op 350 miljoen dollar (voornamelijk uit advertisement), de EBITDA (earnings before interest, taxes, depreciation and amortization) bedroeg 50 miljoen dollar, capital expenditures 200 miljoen dollar, negatieve cashflow van 150 miljoen dollar. Echter, 2009 was het eerste jaar dat Facebook een positieve cashflow had. Voor 2010 wordt een revenue verwacht van 1 miljard dollar.

Daarnaast zijn de negatieve cashflows in de eerste jaren ook niet echt problematisch als er een waardevol product aan ten grondslag ligt. Facebook is nu 50 miljard dollar waard, en dat trekt investeerders aan.

Het doel van CHAIN:

- Aantrekken van advertisement – maar dan wel zeer gericht op bedrijven en niet hinderlijk voor interface
- Een waardevol product maken

Potentiële waarde binnen 10 jaar: ...

APPENDIX H: EVALUATIES SOFTWARE IMPROVEMENT GROUP

Tijdens dit project is er tweemaal een evaluatie van onze code uitgevoerd door de Software Improvement Group (SIG). Deze zijn verricht door Eric Bouwers, waarvoor dank.

EERSTE EVALUATIE (3-10-2011)

"De code van het systeem scoort bijna 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De score wordt naar beneden gehaald door de Module Coupling, de Unit Size en de Unit Complexity.

Voor Module Coupling wordt er gekeken naar het percentage van de code wat relatief vaak wordt aangeroepen. Normaal gesproken zorgt code die vaak aangeroepen wordt voor een minder stabiel systeem omdat veranderingen binnen dit type code kan leiden tot aanpassingen op veel verschillende plaatsen. Wat in dit systeem opvalt is dat 'Database.php' zowel vrij groot is als relatief vaak wordt aangeroepen. Een nadere inspectie laat zien dat er binnen dit bestand meerdere verantwoordelijkheden terug te vinden zijn. Zoals verwacht is er code om de database te benaderen, maar er zijn ook utility-methodes zoals 'arrayToString' en 'getBaseURL' te vinden. Het apart zetten van beide verantwoordelijkheden zorgt ervoor dat het later makkelijker is om een bepaalde verantwoordelijkheid terug te vinden.

Een ander punt waarop de verantwoordelijkheden niet duidelijk zijn is de toegang tot de database. In Database.php zijn verschillende functies terug te vinden welke objecten verwijderen (bijvoorbeeld 'deleteAccount'), maar het toevoegen van het account-object is geïmplementeerd in 'Account.php' ('saveToDb'). Dit kan in de toekomst leiden tot verwarring omdat het niet duidelijk is waar welke database manipulatie wordt uitgevoerd. Het is aan te raden kritisch te kijken waar welke verantwoordelijkheid (bijvoorbeeld het manipuleren van de database) hoort te liggen en deze verantwoordelijkheid centraal te regelen.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt er ook weer voor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Commentaar regels zoals '// Get message statusses first' en '// Remove all sent private messages received by this company' geven meestal al aan dat er aparte blokken van functionaliteit te onderscheiden zijn binnen methodes welke makkelijk los te trekken zijn. Het is sterk aan te raden kritisch te kijken naar de langere methodes binnen dit systeem en deze waar mogelijk op te splitsen.

Voor Unit Complexity wordt er gekeken naar het percentage code dat bovengemiddeld complex is. Ook hier geldt dat het opsplitsen van dit soort methodes in kleinere stukken ervoor zorgt dat elk onderdeel makkelijker te begrijpen, makkelijker te testen en daardoor eenvoudiger te onderhouden wordt. In dit geval komen de meest complexe methoden ook naar voren als de langste methoden, waardoor het oplossen van het eerste probleem ook dit probleem zal verhelpen.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden zodra het systeem verder groeit. In onze mailwisseling werd nog gesproken over unit-tests, dit klinkt veelbelovend. Hopelijk kunnen deze met de hermeting ook meegestuurd worden om zo een beeld te krijgen van de kwaliteit van deze testen."

TWEEDE EVALUATIE (7-11-2011)

“In de tweede upload zien we dat de omvang van het systeem is verdubbeld, maar dat daarbij de score voor onderhoudbaarheid is gedaald. Deze daling is met name toe te schrijven aan de introductie van relatief veel lange methodes. In de begeleidende e-mail is aangegeven dat het wegwerken van deze lange units moeilijk is in verband met het gekozen database-systeem, hieruit blijkt dat men deze keuze in ieder geval bewust heeft gemaakt. De daling in de score wordt enigszins gecompenseerd doordat er een forse verbetering te zien is op het gebied van Module Coupling, dit omdat met name 'Database.php' flink is opgeschoond.

Uit deze observaties en de begeleidende e-mail kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject. De verhouding tussen de hoeveelheid test-code en de hoeveelheid code voor productie valt op dit moment nog wel vrij laag uit, het is aan te raden om kritisch te bekijken of alle belangrijke delen van de functionaliteit getest worden.”

APPENDIX I: FOTO'S LANCERING NEW YORK EN DELFT

NEW YORK SOFT LAUNCH (1-11-'11)

LANCERING DELFT CONGRESCENTRUM TU DELFT (11-11-'11)