PROVINCIALE HOGESCHOOL LIMBURG

STAGERAPPORT

php webapplicatie

Project Information System

Naam: **Davy Rego**

Klas: **3 TIN E**

Academiejaar: **2006 - 2007**

Stageplaats: Sanmax

PROVINCIALE HOGESCHOOL LIMBURG (PHL)

HANDELSWETENSCHAPPEN EN BEDRIJFSKUNDE

Elfde-Liniestraat 26 (gebouw B)

3500 Hasselt

t | 011 23 87 77

f | 011 23 87 90

e | h&b@phl.be

WOORD VOORAF

In dit stagerapport vindt u een beschrijving terug van de activiteiten tijdens mijn stageperiode. Als derde en tevens laatstejaarsstudent Toegepaste Informatica aan de Provinciale Hogeschool Limburg stond er ook mij een stageperiode van drie maanden te wachten. Van 5 maart tot en met 8 juni 2007 bracht ik de werkweek door bij Sanmax B.V.B.A. te Genk.

Graag zou ik enkele personen willen bedanken die bijgedragen hebben tot een vruchtbare en leerrijke stage.

Mijn dank gaat allereerst uit naar mevr. Gabrielle Erlingen. Ze heeft haar taak als stagelector zeer goed uitgevoerd. Ze was steeds bereikbaar voor vragen en zorgde zelf voor de nodige feedback. Dit laatste is een meerwaarde geweest voor dit stagerapport, aangezien ik deze feedback steeds in het achterhoofd gehouden heb bij het samenstellen van dit rapport.

Ook wil ik dhr. Pascal D'Helft, zaakvoerder van Sanmax, en mijn stagebegeleider mevr. Lies Ceuppens bedanken voor de kans en het vertrouwen dat ik gekregen heb om een dergelijk project te mogen uitwerken en dat in een aangename werksfeer.

Tot slot bedank ik alle andere collega's van Sanmax voor de raad die ze mij van tijd tot tijd gaven, een figuurlijk schouderklopje na een drukke werkdag of de ontspannen sfeer tijdens de middagpauzes. Toch even mijn speciale dank aan webontwikkelaar dhr. Andries Seutens voor de technische ondersteuning tijdens het werken met het door hem ontwikkelde framework.

U kunt dit rapport ook digitaal terugvinden op mijn persoonlijke website www.davyrego.be of de bijgevoegde CD-ROM. Hier vindt u tevens de volledige broncode van de uitgewerkte applicatie terug. Deze CD-ROM vindt u terug op de kaft achteraan de bundel.

Ik hoop dat ik u als lezer van dit stagerapport evenveel kan bijbrengen als deze stage mij heeft bijgebracht. Veel leesplezier !

INHOUDSTAFEL

Sanmax

PHL

1 INLEIDING

Zoals al aangegeven in het voorwoord loopt iedere derdejaarsstudent Toegepaste Informatica tijdens zijn laatste jaar drie maanden stage. Al vroeg tijdens het academiejaar kregen we de nodige uitleg en mochten we op zoek naar een geschikte stageplaats.

Ik koos voor Sanmax, een jong en dynamisch webontwikkeling bedrijf gelegen in Genk. Sanmax was mij ondermeer bekend als ontwikkelaar en beheerder van de officiële website van voetbalclub KRC Genk. Na contact gehad te hebben met zaakvoerder Pascal D'Helft werd mijn sollicitatie voor een stage geaccepteerd.

De keuze voor een bedrijf dat actief is in de webontwikkeling was niet onbewust. Vanuit mijn vrije tijd was er al een zeer sterke interesse voor het ontwikkelen van webpagina's aanwezig. Ik heb in het verleden al enkele websites uitgewerkt, al ging het dan vooral over statische websites. Naast het ontwikkelen van een standaard gastenboek met PHP en MySQL was de wereld van dynamische websites mij nog volledig onbekend. Ook tijdens het vak webtechnologie werd enkel de basis van PHP behandeld. Ik was dus zeer nieuws- en leergierig naar de wereld van het dynamisch ontwikkelen van webpagina's.

Meer informatie over Sanmax en het project dat ik voor hun uitgewerkt heb, vindt u gestructureerd terug in het volgende hoofdstuk onder de noemer "Plan van Aanpak".

> *Practice is the best of all instructors.*
>
> *Publilius Syrus, Latijns schrijver*

2 PLAN VAN AANPAK

Met behulp van een "Plan van Aanpak" geef ik de nodige informatie weer over het project en de realisatie ervan. Een plan van aanpak is dan ook ideaal om te gebruiken als een eerste kennismaking met het uit te werken project. U kunt dit onderdeel dus gerust zien als een uitgebreide inleiding van deze stagebundel, weliswaar meer gestructureerd weergegeven.

2.1 Inleiding

2.1.1 Doelgroep

De lezers die ik met dit document zal proberen te bereiken en waarvoor deze informatie dus opgesteld is, zijn:

- de zaakvoerder en de medewerkers van Sanmax;
- de stagebegeleider en de stagelector;
- eventuele toekomstige medewerkers of stagiaires die aan het project zullen verder werken.

2.1.2 Aanleiding

Het huidige systeem, dat gebruikt wordt voor het interne klanten- en projectbeheer van het bedrijf Sanmax, is aan een aanpassing toe. De huidige webapplicatie zou geoptimaliseerd moeten worden en een aantal losstaande applicaties zouden geïmplementeerd moeten worden en dit met het oog op een gebruiksmakkelijkere en functionelere applicatie.

2.1.3 Achtergrond

Het huidige klanten- en projectbeheer gebeurt met behulp van de tool "PMtool". Pmtool is een op PHP en MySQL gebaseerde project management applicatie. Via deze tool is het mogelijk om al de klanten te beheren en hun projecten op te volgen. Per project kunnen er verscheidene subtaken toegevoegd worden door verschillende medewerkers van de firma. Verder is het bijvoorbeeld mogelijk om via rapportering op te vragen hoelang een bepaalde medewerker gewerkt heeft aan bepaalde onderdelen. Deze applicatie kan als freeware gedownload worden.

2.2 Projectdefinitie

2.2.1 Bedrijfsgegevens

Naam:	Sanmax B.V.B.A.
Adres:	Mieënbroekstraat 33
	B-3600 Genk
Tel.:	070 25 02 36
Fax:	089 65 66 39
Website:	http://www.sanmax.be
E-mail:	info@sanmax.be

2.2.2 Bedrijfsgegevens & opdrachtgever

Sanmax is een dynamisch internetbedrijf, dat creativiteit en enthousiasme combineert met technische kennis en ervaring. Sanmax levert aan zelfstandigen en KMO's professioneel en betaalbaar maatwerk voor alle toepassingen op het internet: webdesign, database management, webhosting, e-marketing, webapplicaties, enzovoort.

Sanmax luistert naar de klant en zet zijn wensen om naar de meest efficiënte van alle beschikbare oplossingen. Mogelijk is dit een zelf ontwikkeld maatproduct. Een professionele en volledig transparante aanpak: no nonsense, met regelmatig overleg en goede afspraken.

De Genkse bvba is in Limburg ondermeer bekend om zijn nauwe samenwerking met voetbalclub KRC Genk. Naast het designen, onderhouden en hosten van de website verzorgt men nog tal van andere services voor de Belgische topclub. Zo zorgt dit bedrijf voor de visuele beelden op het intern tv-circuit, de ledwalls en ledboaring in het stadion van KRC Genk. Ook heeft men tijdens de thuiswedstrijden een live-streaming met wedstrijdverslag uitgewerkt en verzendt men naar heel wat supporters maandelijks een “e-flash” met het laatste nieuws van hun favoriete voetbalclub. Andere grote projecten zijn de website voor wielerploeg Quickstep-Innergetic, Dagtoerisme Limburg, Oceade & Mini Europa, Een hart voor Limburg,…. Dit is slechts een kleine greep uit de vele projecten waar de Sanmax medewerkers wekelijks zoet mee zijn.

In onderstaand krantenartikel vindt u nog wat meer informatie terug. Dit werd op 29/03/2006 gepubliceerd in Het Belang van Limburg naar aanleiding van een special over de bedrijven op de Limburgse, en dus ook de Genkse industrieterreinen.

"Dankzij ons alles over Tom Boonen op website"

SANMAX

GENK - Sanmax is een snelgroeiend internetbedrijf in het Genkse. "Vijf jaar geleden zijn we begonnen als éénmansbedrijfje, nu werken we hier al met vijf mensen," vertelt Pascal D'Helft. Sanmax is op drie gebieden actief: "We ontwerpen websites voor het bedrijfsleven en we doen voor bedrijven ook aan *webhosting*, zeg maar dat we d'r voor zorgen dat websites op het internet komen." Een derde pijler zijn de webapplicaties of internettoepassingen. "Zo hebben we een agendabeheerspakket uitgewerkt voor artsen. Dankzij een dergelijk systeem kunnen patiënten via pc een afspraak maken met hun dokter. Momenteel zijn er een 300-tal artsen die dit systeem al gebruiken. Daarnaast hebben we ook een pakket voor e-mailmarketing (www.beflash.com) op de markt gebracht dat al door tientallen bedrijven gebruikt wordt om via e-mail relaties en klanten aan te schrijven. Voor de verschillende Vlaamse provincies zijn we nu ook de volledige geestelijke gezondheidszorg in kaart aan het brengen. Op die manier zullen de psychiatrische instellingen zelf hun info op internet up-to-date kunnen houden."
Sanmax is verder erg actief in de sportwereld. "De officiële website van KRC Genk hebben we ontworpen. Net zoals die van wielerploeg Quickstep (www.quickstepcycling.com), met onder meer spurtbom Tom Boonen. Naast het design zorgen wij ervoor dat de info constant bijgewerkt wordt."
En nog meer sport: Sanmax is 3 jaar geleden met de ontwikkeling van een voetbalmanagerspel gestart. "Daar wordt nu al door meer dan 5.000 surfers dagelijks mee gespeeld. Het spel is al ver over de landgrenzen verspreid en vertalers hebben er voor gezorgd dat het momenteel al in het Italiaans, Litouws, Portugees, Roemeens en Pools speelbaar is. Andere talen zitten in de pipeline. Het is een gratis spel, dus zeker en vast de moeite waard om eens een kijkje te gaan nemen op www.soccerproject.com."

Bij Sanmax zetten ze alle info over spurtbom Boonen op de website van wielerploeg Quickstep. Foto Tony VAN GALEN

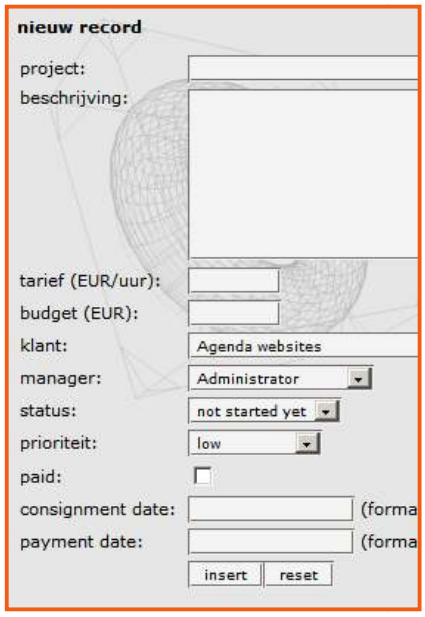
Het Belang van Limburg – dinsdag 29 maart 2006

2.2.3 Projectstatus en voortgang tot nu toe

Naast het gebruik van PMtool, werden er ook al een aantal losstaande applicaties ontwikkeld door de medewerkers van Sanmax zelf, om de tekortkomingen in PMtool aan te vullen. Bijvoorbeeld om het beheer van onderhoudscontracten op te vragen en te wijzigen voor bepaalde klanten, om de ftp gegevens van een klant op te vragen,….

2.3 Doel van het project

2.3.1 Probleemstelling



Zoals in de inleiding al vermeld werd, gebruikt men momenteel de webapplicatie PMtool voor het klanten- en projectenbeheer. PMtool is een gratis applicatie die Sanmax enige tijd terug gedownload heeft als een tijdelijk oplossing. Dit pakket voldoet echter niet volledig aan hun eisen. Een aantal functies zijn overbodig, een aantal functies zouden functioneler en gebruiksvriendelijker geraadpleegd moeten worden, en een aantal modules zijn gewoon niet beschikbaar via PMtool. De overzichten en rapporten die momenteel opgevraagd kunnen worden via de PMtool webapplicatie zijn eerder onoverzichtelijk en de opvolging van de verschillende projecten is daardoor niet altijd even vanzelfsprekend.

Ook werkt men op dit ogenblik met een aantal losstaande zelfgeschreven webapplicaties om het klanten- en projectbeheer toch zo goed mogelijk op te volgen. Dit betekent echter wel dat deze webapplicaties op verschillende plaatsen en dus via verschillende URL's geraadpleegd moeten worden.

2.3.2 Doelstelling informatiesysteem

Het informatiesysteem dat ontwikkeld zal worden heeft tot doel het beheer van de klanten en de projecten binnen Sanmax aanzienlijk te optimaliseren. De bestaande tool PMTool zal niet meer gebruikt worden. Een gelijkaardige webapplicatie zal volledig opnieuw geschreven worden met als doel deze functioneler en gebruiksvriendelijker te maken. Zo is het de bedoeling dat men op een overzichtelijke manier alle nodige informatie kan opvragen per medewerker, klant en/of project. Vragen als "waar is persoon x mee bezig?", "hoe zit het met project y?" en "wat moet er nog gebeuren voor klant z?", "hoelang heeft persoon a gewerkt aan project b ?" moeten zo snel en overzichtelijk mogelijk beantwoord kunnen worden met deze webapplicatie. Kortom men zal via deze applicatie de volledige geautomatiseerde administratie kunnen opvolgen.

Andere bestaande en wel goed functionerende applicaties zullen wat het principe betreft grotendeels behouden blijven, maar wel geïmplementeerd worden in de uit te werken beheerapplicatie.

2.3.3 Projectopdracht

- de huidige beheerapplicatie efficiënter en functioneler maken door deze volledig opnieuw uit te werken;
- de gegevens opslaan in een centrale database, deze is al aanwezig maar kan en zal aangepast worden;
- de nodige gegevens ophalen en rapporten genereren uit de centrale database wanneer nodig.

2.3.4 Eisen van de opdrachtgever

- de applicatie moet via het intern netwerk geraadpleegd kunnen worden;
- de gegevens moeten verstuurd worden over het netwerk;
- er moet gebruik gemaakt worden van een MySQL database (beschikbaar via het net);
- er zal geprogrammeerd worden in PHP en dit volgens een bepaalde werkwijze en structuur (framework, template engine);
- de zaakvoerder en medewerkers moeten de gegevens op elk moment kunnen raadplegen;
- de zaakvoerder en medewerkers kunnen gemakkelijk en snel rapporteringen opvragen;
- de zaakvoerder heeft de mogelijkheid om de werkzaamheden van zijn medewerkers op te volgen.

2.3.5 Op te leveren producten en diensten

Product	Omschrijving	Datum
Plan van Aanpak	Algemene omschrijving en planning van het project	2007-03-07
Informatieanalyse (FCO-IM)	IGD's, gegevensmodel, databasestructuur en E.R.D.	2007-03-12
OO-analyse (UML)	UseCase-diagrammen	2007-03-16
MySQL Scripts	Script voor het aanmaken van de MySQL database	2007-03-21
Presentatie analyse	Presenteren van gemaakte analyse + feedback	2007-03-26
Einde programmatie	Definitieve versie van het programma	2007-06-01
Eindpresentatie	Presenteren van de beheerapplicatie	2007-06-07
Test- & implementatiefase	Applicatie controleren, corrigeren en implementeren	2007-06-08

2.3.6 Projectafbakening

2.3.6.1 Begrenzingen

Buiten de scope van dit project valt:

- het onderhoud van de applicatie;
- het onderhoud van de hardware;
- het hosten van de applicatie;
- eventuele interne opleidingen.

2.3.6.2 Ingangscriteria, -documenten en –producten

- de al bestaande beheerapplicatie PMtool en de bijhorende structuur van de database;
- de al bestaande losstaande applicaties (onderhoudscontracten,…).

2.3.6.3 Projectrelaties en afhankelijkheden

Dit project- en klantenbeheersysteem vormt de basis voor een beheersysteem voor elke klant individueel. In een verdere fase is het de bedoeling dat de klanten hun bepaalde informatie via een externe website zal kunnen raadplegen en aanpassen. Dit project wordt al dan niet door een andere medewerker/stagiair verder uitgewerkt en staat los van dit project.

2.4 Projectaanpak

2.4.1 Inleiding en strategie

De eerste fase bestaat uit het onderzoeken en het ontleden van de huidige werkwijze, de bepaling van de doelgroep voor wie de applicatie bestemd is en het vastleggen van de verwachtingen om te komen tot een beter, functioneler en efficiënter beheerssysteem.

2.4.2 Methode en werkwijze

Het project bestaat uit vier fases: enerzijds onderzoek en ontledende analyse, anderzijds de programmatie, de implementatie van de software en de testperiode, tot slot de definitieve implementatie.

In de eerste fase wordt er onderzoek gedaan naar de al beschikbare gegevens (databases,…) en wordt het plan van aanpak opgesteld met alle verzamelde informatie van het huidige systeem.

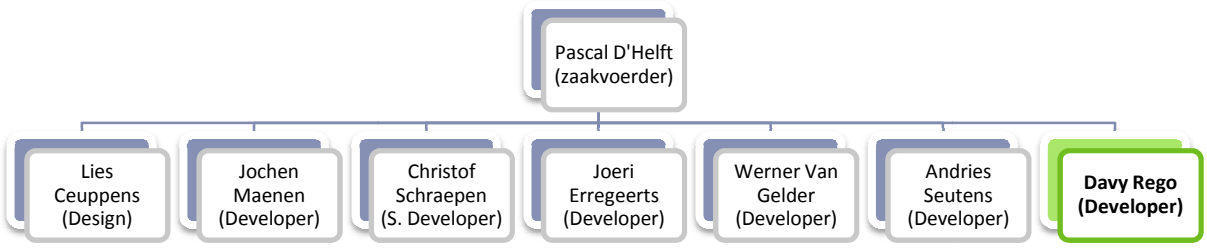
In de tweede fase wordt de applicatie ontwikkeld en geprogrammeerd. Vervolgens zal deze in de overige fases geïmplementeerd worden, wat ook betekent dat er een zo uitgebreid mogelijke testperiode plaatsvindt alvorens de applicatie definitief in gebruik genomen zal worden.

Er zal in deze verschillende fases regelmatig gecommuniceerd worden tussen enerzijds zaakvoerder, stagebegeleider en andere medewerkers en anderzijds mijzelf, de stagiair, om elkaar op de hoogte te houden. Via de nodige feedback zal bekeken worden of we op dezelfde lijn zitten wat het doel en de juiste uitwerking van de beheerapplicatie betreft.

2.5 Projectbeheersing

2.5.1 Organisatie

2.5.1.1 Structuur stagebedrijf



Samenstelling functieverdelingen binnen organisatie

Functie	Naam
Zaakvoerder	Pascal D'Helft
Design (stagebegeleider)	Lies Ceuppens
Senior Webdeveloper (PHP)	Christof Schraepen
Webdeveloper (PHP)	Andries Seutens
Webdeveloper (PHP)	Joeri Erregeerts
Webdeveloper (PHP)	Werner Van Gelder
Webdeveloper (Input & HTML)	Jochen Maenen
Webdeveloper (stagiair PHL)	Davy Rego
Webdeveloper (stagiair Xios)	Mindy Stukken

In- en externe taakverdeling project

Analyse, ontwikkeling & implementatie	Davy Rego
Opvolging (stagebegeleider)	Lies Ceuppens
Opvolging (algemeen)	Pascal D'Helft
Opvolging (technisch)	Andries Seutens & Christof Schraepen

3 ANALYSE

In dit onderdeel bespreek ik de analyse die ik uitgevoerd heb alvorens ik startte met het ontwikkelen van de applicatie. Let wel op: in deze bundel doorloop ik elk deel van de analyse maar dit slechts voor een specifiek onderdeel van de applicatie. Zo neem ik steeds het beheer van de projecten onder de loep. Dit om het overzicht te bewaren en duidelijk het verband te zien. De volledige analyse is terug te vinden in de verschillende documenten die als bijlage zijn toegevoegd.

3.1 Informatieanalyse

3.1.1 Inleiding

Een succesvol eindresultaat van een goed project is afhankelijk van een goede informatieanalyse. Dankzij deze analyse zal ondermeer een goed databaseontwerp bekomen kunnen worden. Voor er echter gegevens over tabellen kunnen verdeeld worden, moet eerst bepaald worden welke gegevens relevant zijn voor het systeem en dus in het systeem opgenomen zullen worden. Dit alles gebeurt tijdens de informatieanalyse. Het eventueel huidige systeem en de toekomstige behoeften worden geïnventariseerd en geanalyseerd.

Voor de informatieanalyse wordt gebruik gemaakt van de werkwijze volgens FCO-IM (Fully Communication Oriented – Information Modeling), een onderdeel van NIAM (Nijssen Information Analysis Method). De grote troef van deze methode is dat er vertrokken wordt vanuit de natuurlijke taal. De analyse zal in feite samen met de klant gemaakt worden, waarbij acties die het uit te werken systeem moet kunnen doen, verwoord zullen worden. Deze expressies zullen met behulp van inductie naar een conceptueel schema verwerkt worden. Dit schema zal weergeven welke gegevens uitgewisseld moeten worden en welke betekenis aan deze gegevens toegekend wordt. De schema's die gegenereerd zullen worden met een tool worden Informatie grammatica diagrammen genoemd.

Na het verwerken van de feittypes zal een gegevensmodel en een databasestructuur kunnen opgemaakt worden. Ook dit zijn resultaten van een informatieanalyse. Meer hierover leest u in de volgende onderdelen van dit hoofdstuk.

De gereedschapskist die in mijn geval gebruikt wordt, is CaseTalk. Dit is een uiterst krachtige CASE-tool die als download verkrijgbaar is via hun website. De tool is gebaseerd op een Windows-georiënteerde interface, heeft een volledig transparante repository en biedt volledige grafische ondersteuning. Deze tool werd ook gebruikt tijdens mijn opleiding bij het maken van informatieanalyses.

3.1.2 Informatie Grammatica Diagrammen

Als eerste worden de IGD's (Information Grammatica Diagrams) opgesteld. Deze diagrammen worden gegenereerd op basis van feittype-expressies. Deze feittypen zijn "gesproken" expressies uit het domein van de gebruiker. De analist zal dus in principe "samen met de klant" de analyse doen, wat een beter resultaat zal leveren. Dit gebeurt meestal via een interview met de klant. Tijdens dit gesprek wordt gekeken wat de huidige werkwijze is, wat de tekortkomingen zijn van het huidige systeem en waar men graag naar toe wil met het toekomstige systeem. De probleem- en doelstelling, zoals aangegeven in het plan van aanpak, zullen tijdens dit onderdeel dan ook goed in het achterhoofd gehouden moeten worden.

3.1.2.1 Beschrijving

Om een goede basis te hebben voor het opmaken van de feittype-expressies is het het makkelijkste en ook gebruikelijk om eerst in een doorlopende tekst het systeem of het onderdeel van het systeem duidelijk te omschrijven. Hieronder vindt u de beschrijving voor het beheer van de projecten.

> « Een van de andere belangrijke onderdelen is het beheer van de projecten. Een project heeft een uniek nummer en naam. Twee projecten met dezelfde naam zijn dus logischerwijze niet mogelijk. Naast een naam zal een project ook een duidelijke omschrijving bevatten. Een project zal aan een klant gekoppeld worden, het is mogelijk dat een klant één of meerdere projecten heeft. Daarnaast zal een project ook gekoppeld worden aan een gebruiker, een medewerker van Sanmax.
>
> Het financiële aspect van een project was niet noodzakelijk, maar wordt wel al voorzien. Zo kan een budget in uren opgegeven worden, en zal ook het aantal gepresteerde uren bijgehouden worden. U kunt stellen dat dit berekend kan worden aan de hand van onderliggende gegevens van de tijdsduur van taken, maar er wordt hier toch een apart veld voorzien. Zo kan men eventueel fictieve gepresteerde uren opgeven in plaats van enkel en alleen de mogelijkheid te hebben om ze te automatisch te laten berekenen. Ook een kostprijs per uur zal gedefinieerd worden, omdat dit per project kan verschillen of door de jaren heen kan veranderen.
>
> Verder heeft een project ook een prioriteit zodat men weet welk project voorrang moet krijgen op andere projecten. Een project kan ook in een bepaalde status zijn. Afhankelijk van de status zal softwarematig bepaald worden wat de gebruiker kan met die project. Wanneer het project nog niet gestart is zal men geen taken kunnen toevoegen, wanneer alle taken afgewerkt zijn zal het project kunnen voltooid worden,…. Wanneer een project de status afgewerkt krijgt, zal er een einddatum bijgehouden worden. »

3.1.2.2 Feittype-expressies

Op basis van de beschrijving worden de feittypes opgemaakt. Er wordt gebruik gemaakt van concrete voorbeelden, om de betekenis van elk feittype zo duidelijk mogelijk te maken. Het is ook belangrijk dat het hier gaat om elementaire zinnen. Dit wil zeggen dat de zinnen niet verder opsplitsbaar zijn zonder dat er relevante informatie verloren gaat.

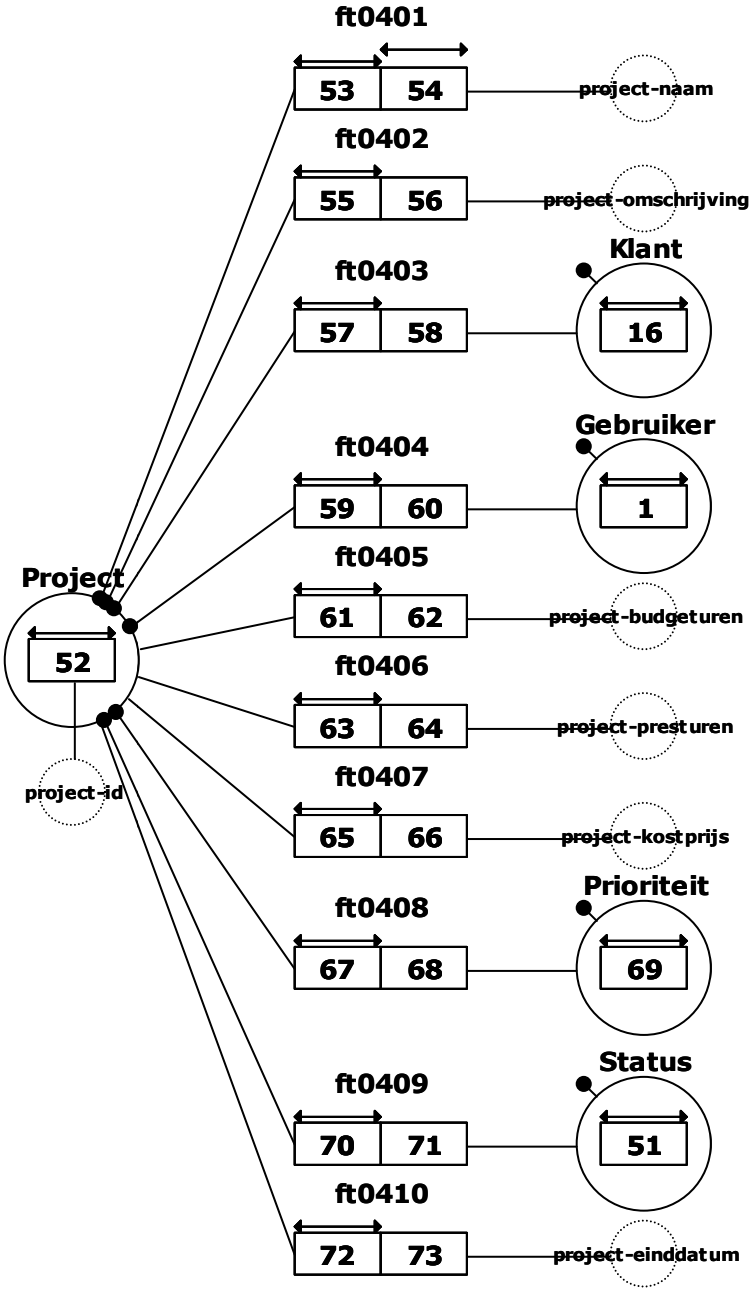
Project:
Project 0037
Project 0037 heeft als naam “Website Argipel”
Project 0037 heeft als omschrijving “Design en ontwikkeling van dynamische website”
Project 0037 wordt uitgewerkt voor Klant 00043
Project 0037 werd aangemaakt door Gebruiker 002
Project 0037 heeft als budget 80 uren
Project 0037 heeft als gepresteerde uren 75 uren
Project 0037 heeft als kostprijs per uur 17,5 EUR
Project 0037 heeft als Prioriteit 03
Project 0037 heeft als Status 04
Project 0037 werd afgewerkt op 2007-02-29 11:35:45

De tweede stap is het analyseren van bovenstaande feittypen. Dit gebeurt met behulp van een casetool, in mijn geval CaseTalk. Elke zin zal ingevoerd worden. Als voorbeeldzin neem ik ‘Project 0037 wordt uitgewerkt voor Klant 00043’. Deze zin wordt als nieuwe expressie ingevoerd en krijgt een naam toegewezen, in dit geval ‘ft0404’. Elk betekenisvol zinsdeel zal aangeduid en benoemd moeten worden. Onderstaand schema geeft de indeling van de voorbeeldzin duidelijk weer.

Bovenstaande expressie is van het binair type, hiermee wordt bedoeld dat er twee objecten in voorkomen, namelijk het object ‘Project’ en het object ‘Klant’. Het is ook mogelijk dat een zin van het primaire type is, waarbij enkel label aan een object gekoppeld zal worden. Een voorbeeld hiervan is ‘ft0402’. In de meeste gevallen zal deze vorm voorkomen: een objecttype dat gekoppeld is aan een label, dat een bepaalde eigenschap van dat object beschrijft.

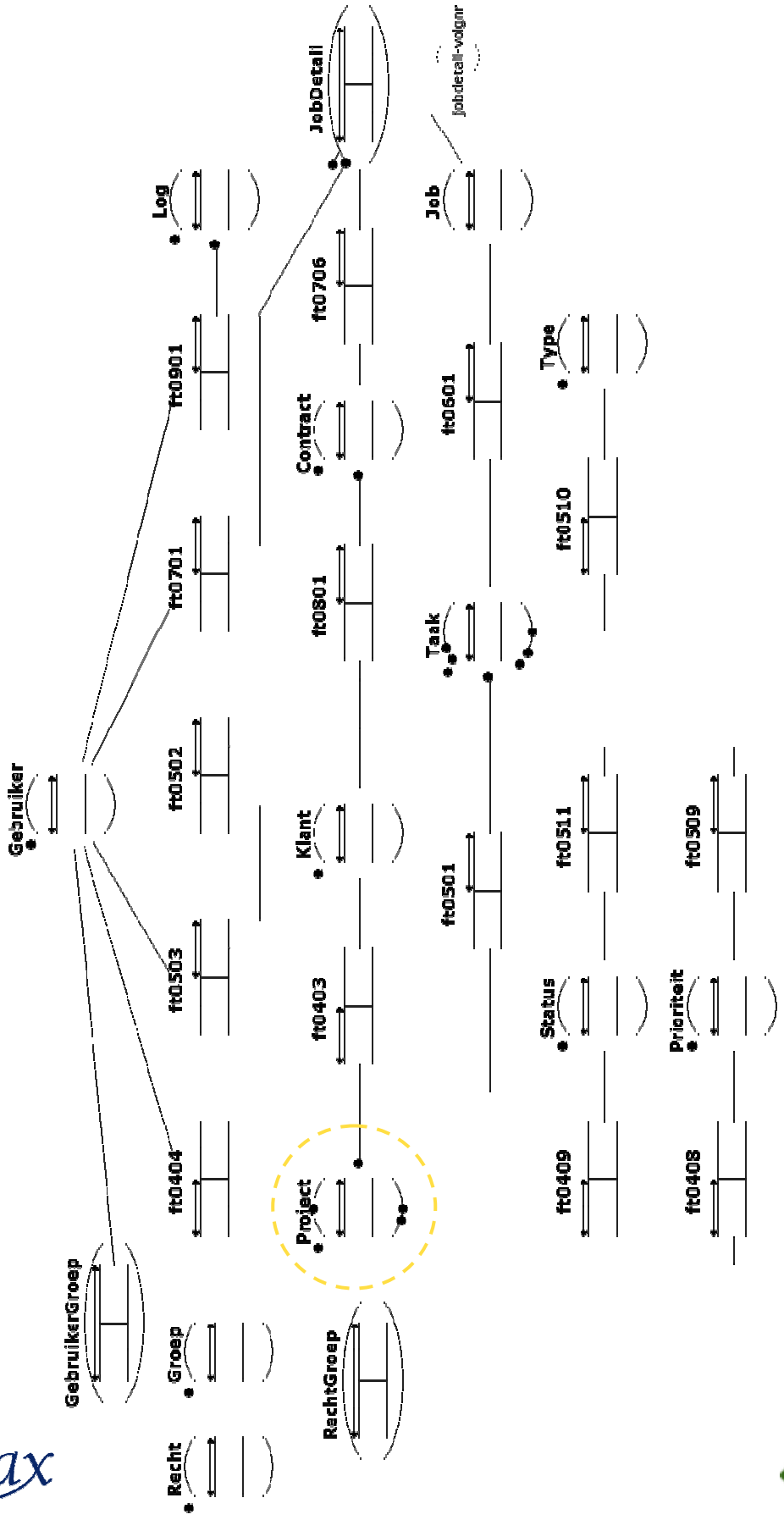
3.1.2.3 Informatie Grammatica Diagram

Na het ingeven van de feittype expressies kan het eigenlijke diagram opgemaakt worden. Dit gebeurt handmatig. We zien duidelijk de verschillende feittypenexpresies met hun object en/of labeltype terugkomen in dit diagram.



Na het grafisch weergeven van het diagram kunnen eventuele beperkingen aangebracht worden. Uniciteitbeperkingen geven bijvoorbeeld aan of een gegeven of combinatie van gegevens slechts eenmaal mag voorkomen of juist niet. Deze regels worden in het diagram weergegeven met de dubbele pijl. Totaliteitbeperkingen geven op hun beurt aan dat een gegeven moet voorkomen en dat het dus gaat om een verplicht gegeven. Dit wordt in het diagram aangeduid met een bolletje aan het object waaraan het feittype verbonden is, in dit geval dus het object 'Project'. Er zijn nog andere beperkingen mogelijk zoals een waardebeperking, waarbij het gegeven slechts een aantal waarden kan aannemen, maar dit is in dit diagram niet van toepassing.

Om u toch even een algemeen beeld te geven van het volledige uit te werken project, alsook de kadering van het bovenstaand onderdeel binnen het volledige project, vindt u hieronder het volledige Informatie Grammatica Diagram terug.



3.1.3 Gegevensmodel:

Wanneer de ingegeven feittypen worden gegroepeerd (feittypen samenvoegen zonder dat er redundantie optreedt), gelexicaliseerd (feittypen transformeren zodat alle rollen gespeeld worden door labeltypen) en gereduceerd (sommige feittypen schrappen) zal uiteindelijk een relationeel schema bekomen worden, ook wel GLR-IGD genoemd. Van dit schema kan een genormaliseerd gegevensmodel afgeleid worden. Dergelijk model is opgebouwd uit componenten zoals **entiteiten** met bijhorende attributen, Ł relaties, sleutels en integriteitregels (o-ptioneel, u-niek, b-erekenbaar).

Projecten

project-id		
project-naam	u	
project-omschrijving		
klant-id		Ł Klanten
gebruiker-id		Ł Gebruikers
project-budgeturen	o	
project-presturen	o b	
project-kostprijs	o	
prioriteit-id		Ł Prioriteiten
status-id		Ł Statussen
project-einddatum		

3.1.5 Databasestructuur:

Op basis van bovenstaand gegevensmodel zal dan het database ontwerp opgesteld worden. De structuur van de database zal normaliter gelijkaardig blijven aan die van het gegevensmodel. Eventuele veranderingen zijn afhankelijk van het type database waarmee gewerkt wordt, maar ook eventuele standaarden waar men zich binnen de programmeeromgeving of intern binnen de onderneming zal aan houden.

De structuur van het bovenstaande gegevensmodel zal in dit geval behouden blijven. Er zullen echter enkele aanpassingen gebeuren wat de naamgeving betreft, hierbij is rekening gehouden met enkele interne gewoontes die men handhaaft bij Sanmax. De tabelnamen blijven in het meervoud. Dit ook om problemen te voorkomen. Zo zouden de tabelnamen 'group' en 'right' fouten geven, omdat dit gereserveerde benamingen zijn binnen MySQL. 'Groups' en 'rights' zullen dus geen probleem vormen. Tabelnamen die bestaan uit samengevoegde namen zullen gescheiden worden door een underscore (liggend streepje, _). Er zal ook een underscore gebruikt worden bij de scheiding in veldnamen (in plaats van het koppelteken, -). Deze zullen ook naar het Engels omgezet worden. Ook elke prefix, die verwijst naar de tabel, zal weggelaten worden.

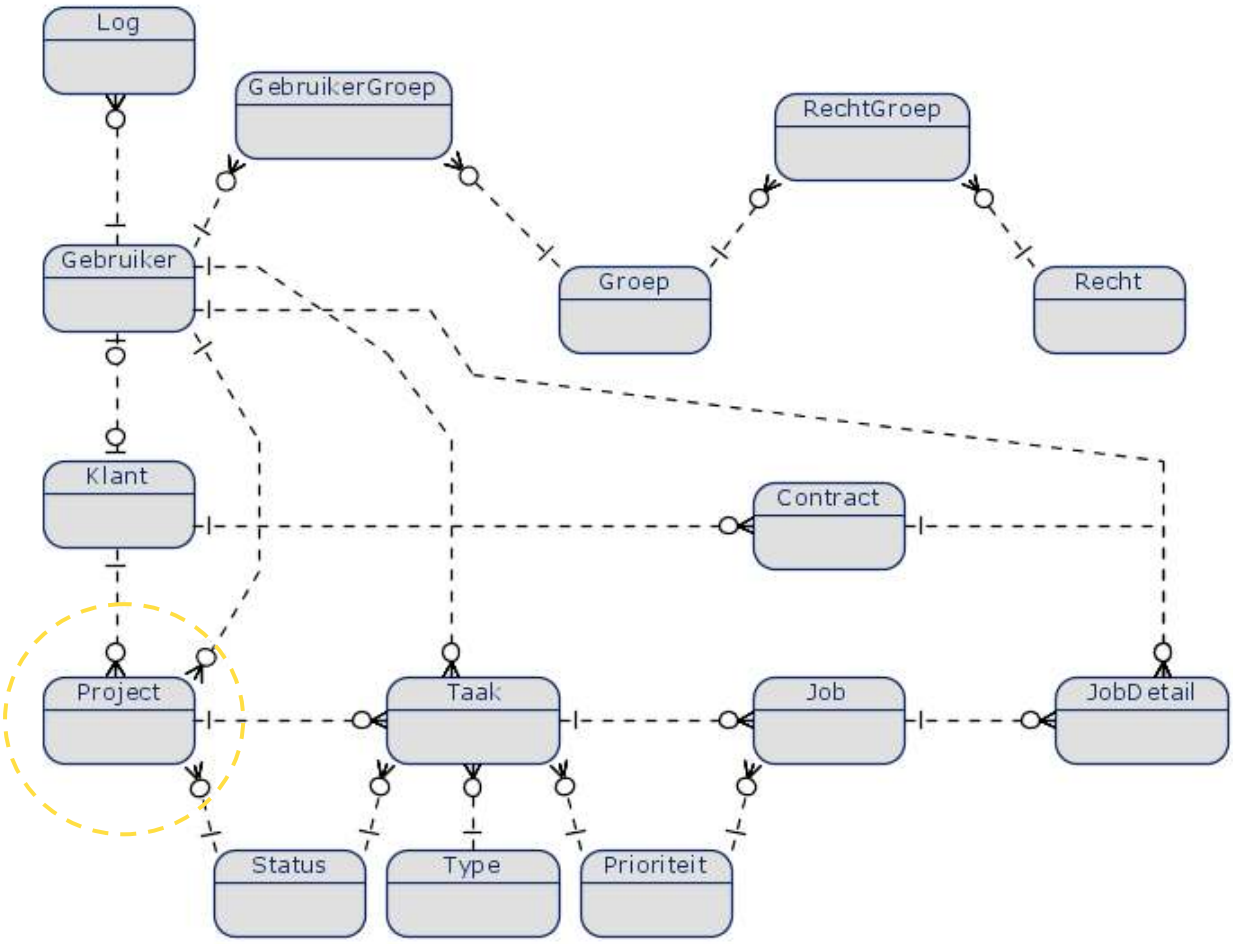
Bepaalde velden zullen ook een ander type krijgen dan verwacht, dit is namelijk afhankelijk van de mogelijkheden afhankelijk van het type databank. Tijdens dit project wordt er gewerkt met een MySQL databank.

projects

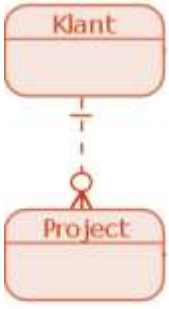
id		
name	u	
description		
client_id		customers
user_id		users
hours_budget	o	
hours_perform	o b	
cost	o	
priority_id		priorities
status_id		status
end_date	o	

3.1.6 Entity-Relationship Diagram

Tot slot zal er dan nog een ERD (Entity Relationship Diagram). Dit diagram zal het conceptueel datamodel grafisch representeren. Het is eigenlijk een visuele weergave van de entiteiten en hun relaties. Vooral het laatste is belangrijk. Het ERD zal ook gebruikt worden om met de opdrachtgever te controleren of dat het inzicht van de analist / ontwikkelaar correct is. Voor alle duidelijk: dit ERD is gebaseerd op het Nederlandstalig gegevensmodel.



Dit diagram werd aangemaakt in de tool Visual Paradigm. Meer informatie over deze tool vindt u terug in het volgende onderdeel. In het hoofdstuk van de objectgeoriënteerde analyse bespreek ik deze tool uitgebreid. Om het nut van bovenstaand diagram te verduidelijken interpreteren we even het hier naast afgebeeld onderdeel.



De stippellijn duidt op een relatie tussen beide entiteiten. Het uiteinde van de lijn bepaald het type relatie. Het voorbeeld hierlangs heeft een 1:N-relatie. Een project moet gekoppeld worden aan een en slechts een klant (). Een klant kan aan nul of meerdere projecten gelinkt worden.

Mogelijke lijneinden:

slechts 1 | 0 of meer | 0 of 1 | 1 of meer

Mogelijke relaties:

1:1-relatie (een op een) | 1:N-relatie (een op meer) | N:M-relatie (meer op meer)

3.2 Object Georiënteerde Analyse

3.2.1 Inleiding

Dat modellen belangrijk zijn in de huidige informaticawereld kan zonder discussie aangenomen worden. Een afbeelding zegt veel meer dan duizend woorden en kan of zal makkelijker begrijpbaar zijn. Zeker voor mensen die niet thuis zijn in de wereld van de analyses, of globaler, de informatica.

Een model is een uitgebreide beschrijving van iets. Dit 'iets' kan al bestaan, maar kan ook in ontwikkeling zijn of nog in de planningfase zitten. Het is belangrijk bij de ontwikkeling van een model de eisen en functionaliteit in het achterhoofd te houden en niet zo zeer de omgeving (programmeertaal) waarin het uitgewerkt zal worden. Vandaar ook de naam UML, Unified Modelling Language. UML is onafhankelijk van de programmeeromgeving, maar tevens ook accuraat, consistent, gemakkelijk aan derden uit te leggen door dat het begrijpelijk is (lees: eenvoudig maar niet simpel) en gemakkelijk te veranderen indien deze behoefte er is.

Ik zal in UML enkel de use case diagrammen uitwerken. Deze diagrammen geven de te leveren functionaliteit weer van het systeem zoals waargenomen door externe actors. Een volledige objectgeoriënteerde analyse uitwerken zou niet alleen tijdrovend zijn, maar ook minder nuttig. Use case diagrammen worden tegenwoordig veelal gebruikt voor het weergeven van de functionaliteit en structuur van de te ontwikkelen applicatie. Het vervangt de verouderde functionele analyse, waarbij DFD's (Data Flow Diagram) getekend werden. Use case diagrammen worden ook regelmatig in de latere testfase gebruikt als checklist, om te controleren of alle nodige onderdelen aanwezig zijn.

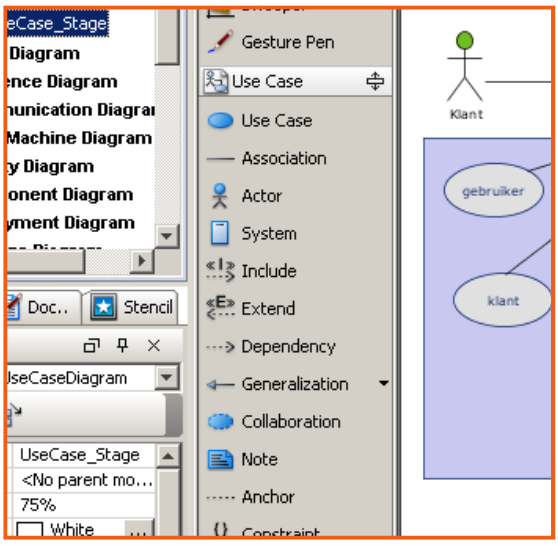
De tool waarin de diagrammen getekend zullen worden is Visual Paradigm. Met deze tool wordt er tegenwoordig ook gewerkt tijdens op het tweede jaar Toegepaste Informatica van de PHL. Tijdens de lessen op het derde jaar gebruikten we echter Rational Rose, maar Visual Paradigm zou meer mogelijkheden hebben. Dit check ik even in de volgende vergelijking tussen beide tools.

> *A picture speaks a thousand words.*

3.2.1.1 Vergelijking Visual Paradigm - Rational Rose

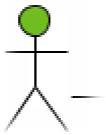
	Visual Paradigm for UML	IBM Rational Rose Modeller
versie	community edition	authorized edition
UML diagrammen	Use Case, Class, Sequence, Communication, State Machine, Activity, Component, Deployment, Package, Object, Composite Structure, Timing en Interaction Overview (13)	Class, Component, Deployment, Sequence, Statechart, Use Case en Collaboration (7)
ondersteuning Use Case	detail editor (use case beschrijvingen) ondersteuning voor gegevens stroom tekstuele analyse	-
database modellering	ERD	-
collaboratie	VP Teamwork Server	
code generatie	mogelijk	mogelijk
documentatie	PDF, MS Word, HTML printen documentatie & diagrammen	HTML, SoDA printen diagrammen
website	www.visual-paradigm.com	www.ibm.com/software/rational
prijs	gratis voor niet commercieel gebruik (professional edition : USD 1678,50)	USD 1850,00

Ook persoonlijk haalt Visual Paradigm het in deze tweestrijd. De mogelijkheden binnen VP liggen hoger dan bij de Rational Rose versie die we op school gebruikten. Er kunnen niet enkel meer UML-diagrammen aangemaakt worden, maar ook het aanmaken van een ERD-diagram (normaal via MS Visio) was geen probleem. Het opmaken van de diagrammen wat de lay-out betreft werkte ook makkelijker via het properties-window (vergelijkbaar met Visual Studio .NET), terwijl er bij Rational Rose heel wat meer acties nodig zijn om bijvoorbeeld simpelweg de kleurtjes van een use case te veranderen. Ook handig zijn de extras die VP aanbiedt ten opzichte van RR, zoals de mogelijkheid om Use Case Detail schema's aan te maken. Bij Rational Rose is deze mogelijkheid niet en bent u genoodzaakt deze gewoon aan te maken via een tekstverwerker. Een laatste en geen onbelangrijk voordeel is dat Visual Paradigm gratis een community (educatieve) versie ter beschikking stelt voor niet commercieel gebruik.



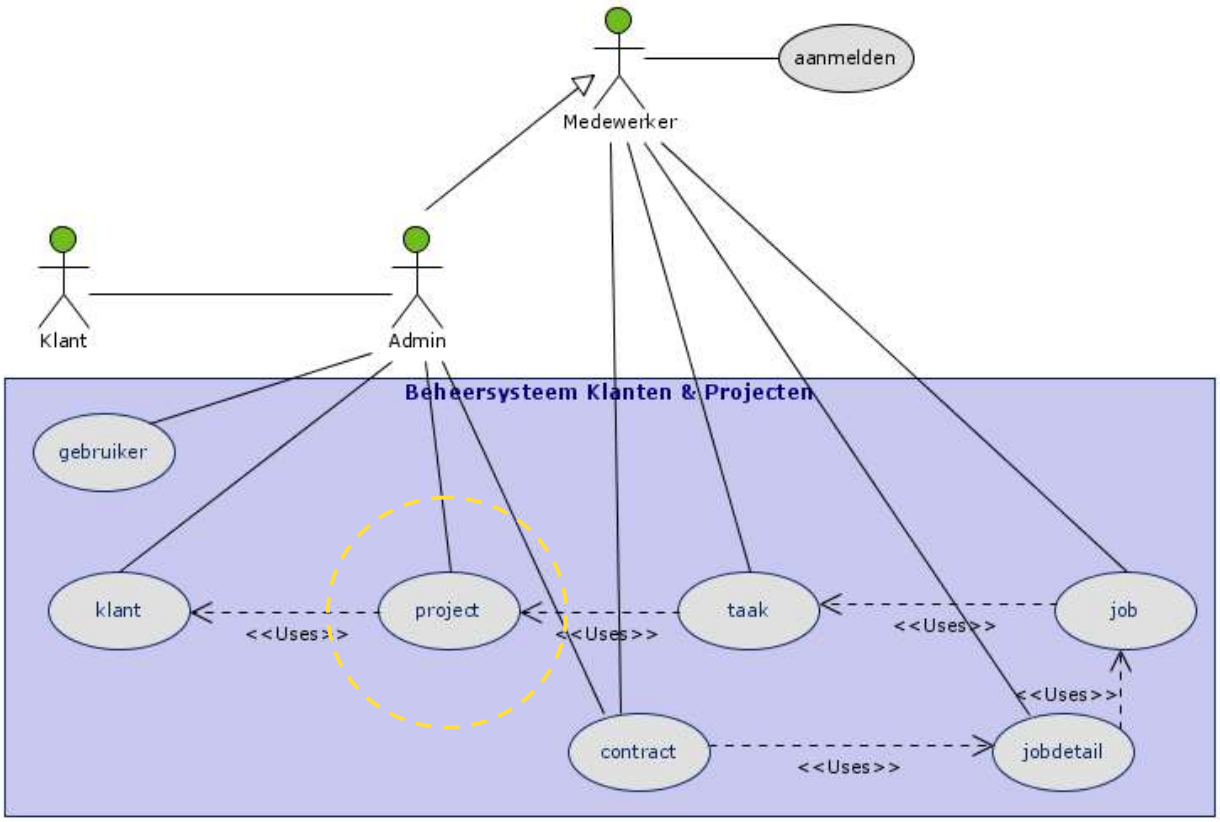
3.2.2 Use case-diagrammen

Het onderstaand diagram geeft een globaal overzicht van het uit te werken systeem. Het systeem dat uitgewerkt zal worden is een klanten- en projectenbeheersysteem. Het systeem is duidelijk afgebakend en alle mogelijke taken bevinden zich binnen dit systeem. Met uitzondering van het aanmelden. De redenering die hier achter zit is het feit dat men zich eerst zal moeten aanmelden vooraleer men toegang krijgt tot het systeem. U kunt ook stellen dat deze use case ook tot het systeem zou moeten behoren, maar ik koos er voor om deze er voor de duidelijkheid niet bij te steken.



Dan de actors, dit is iets of iemand die van buitenaf (extern) in interactie is het met het systeem. Er zijn in ons systeem drie actors die alle drie een persoon in kwestie zullen vertegenwoordigen. De eerste actor is de klant bij Sanmax. De klant zal moeten communiceren met een van de medewerkers van Sanmax. Dat is dan ook onmiddellijk onze tweede actor, een medewerker. En tot slot de administrator. De administrator is een generalisatie van een medewerker. In feite is een administrator gewoon een medewerker, die enkel verschilt doordat hij andere en zelfs meer rechten zal hebben.

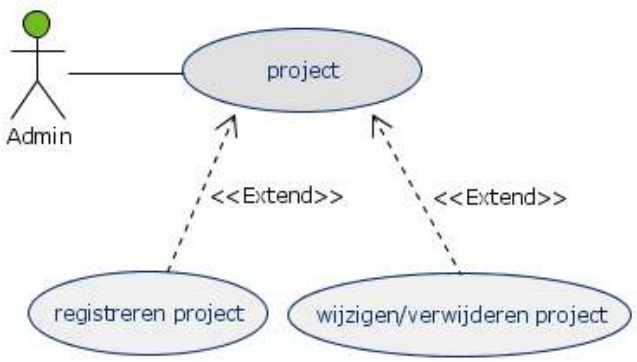
Tot slot bevat het systeem ook heel wat use cases. Elke use case binnen het systeem wordt benoemd met een zelfstandig naamwoord. Het gaat telkens over het beheer van dit onderwerp. Vb.: de use case 'project', staat in voor het beheer van de projecten binnen Sanmax. Het is ook deze use case waarvan ik de extended use cases zal uitwerken tot op het dieper niveau op de volgende pagina's.



Project:

Naam	**Beheer van de projecten**	
Samenvatting	Een administrator zal de projecten beheren. Projecten kunnen toegevoegd, gewijzigd (zowel inhoud als status) of verwijderd worden uit het systeem.	
Actors	Admin	
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren. Voor het wijzigen en verwijderen moeten er bestaande projecten in het systeem aanwezig zijn.	
Beschrijving	***Actor ingave***	***Systeem bewerking***
	1 Actie selecteren: - nieuwe project toevoegen - project wijzigen of verwijderen	
	2	Geselecteerde actie uitvoeren (herhaling: vanaf 1)
Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen. De medewerker (= geen admin) heeft niet genoeg rechten om deze functie op te roepen	
Resultaat	De gebruiker kan alle projecten binnen het systeem beheren.	

Specifieker:



Naam	**Registratie van een project**	
Samenvatting	Een nieuwe klant zal op het systeem geregistreerd worden.	
Actors	Admin	
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren.	
Beschrijving	***Actor ingave***	***Systeem bewerking***
	1 Klant selecteren	
	2 Gegevens project ingeven	
	3	Gegevens valideren (ok: melding + wegschrijven DB) (fout: melding + 2) (herhaling: 2)
	4 Eventueel mogelijkheid om al een taak aan te maken voor dit project. (use case 'taak)	
	5	Doorverwijzen naar de pagina voor het aanmaken van een taak. (vb. door het meegeven van een parameter met het project-id).
Uitzondering	(+ zie uitzonderingen bovenliggende use case)	
Resultaat	Een nieuwe klant is aan het systeem toegevoegd.	

Naam	**Wijzigen van een project**	
Samenvatting	De gegevens van een bestaand project zullen gewijzigd worden.	
Actors	Admin	
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren. Er moet een of meerdere projecten aanwezig zijn in het systeem.	
Beschrijving	***Actor ingave***	***Systeem bewerking***
	1 Project selecteren en de nodige gegevens aanpassen.	
	2	Gegevens valideren (ok: melding + aanpassing DB) (fout: melding + 1) (herhaling: 1)
Uitzondering	(+ zie uitzonderingen bovenliggende use case)	
Resultaat	De gegevens van een bestaand project zijn gewijzigd in het systeem.	

Naam	**Verwijderen van een project**	
Samenvatting	De gegevens van een bestaand project zullen definitief verwijderd worden.	
Actors	Admin	
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren. Er moet een of meerdere projecten aanwezig zijn in het systeem.	
Beschrijving	***Actor ingave***	***Systeem bewerking***
	1 Project selecteren en aangeven dat men deze wil verwijderen.	
	2	Bevestiging vragen
	3 Bevestiging geven (ok) of annuleren (fout)	
	4	Gegevens valideren (ok: melding + aanpassing DB) (fout: melding + 1) (herhaling: 1)
Uitzondering	Het project heeft nog onderliggende taken en zal men dus niet kunnen verwijderen. (+ zie uitzonderingen bovenliggende use case)	
Resultaat	De gegevens van een bestaand project zijn verwijderd uit het systeem.	

3.3 Website Architectuur

3.3.1 Inleiding

De twee vorige uitgewerkte onderdelen van de analyse zouden moeten volstaan voor een goede uitwerking van een applicatie. Ik werk echter aan een webapplicatie, wat dus ook als website bekeken kan worden. Speciaal voor websites bestaat er ook een analyse die de structuur van de website en meer bepaald van de verschillende pagina's op een overzichtelijke manier zal weergeven. Het leek mij dus een goed idee om dit, weliswaar beperkt, uit te werken.

Tijdens mijn opleiding heb ik echter nooit gewerkt aan een dergelijke analyse dus zocht ik zelf de nodige informatie. Deze vond ik in eerste instantie via het boek 'Information Architecture for the World Wide Web, Second Edition'. Dit boek handelt vooral over het theoretische gedeelte en bevat niet echt een gids over het uitwerken van een structuur met behulp van diagrammen. Na wat zoeken kwam ik dan terecht op een online gids genaamd 'Site Diagrams: Mapping and Information Space' geschreven door een zekere Jason Withrow. In deze tutorial wordt de praktische uitwerking, zowel wat documentatie als diagrammen betreft, specifiek uitgewerkt.

> *To successfully communicate the characteristics of an information space, I needed an approach for creating easily understood diagrams. To be useful to my audience, the diagrams must communicate the "big picture" of the website to stakeholders, while providing enough detail to be useful for the development team.*

Jason Withrow, faculty member Internet Professional department, Washtenaw Community College.

3.3.2 Levels

Allereerst zal de structuur van de website uitgeschreven worden. Het uitschrijven van de structuur betekent concreet dat men zal werken met verschillende niveaus. De startpagina zal op het eerste niveau gelegen zijn. De globale pagina's die ook gelinkt zullen zijn op de homepagina zijn van het tweede level. Binnen deze pagina's kan dan nog eens locale navigatie aanwezig zijn die dan van het derde level zullen zijn, enzovoorts. Het aantal levels is in principe onbeperkt, toch is het aan te raden de website niet te diep uit te werken. De kans wordt dan groter dat de structuur en navigatie voor de gebruiker niet meer duidelijk en gebruikersvriendelijk zal zijn.

Ook bij het opstellen van de structuur zal gewerkt worden met een specifieke nummering. De reden hiervoor is dat het mogelijk is dat bepaalde pagina's dezelfde naam zullen dragen, daarom geef ik elke pagina een uniek nummer. Aan de hand van deze nummering zal ook het level afgeleid kunnen worden.

De structuur van de door mij uitgewerkte website zal er als volgt uitzien. Let wel op, dit schema is vereenvoudigd en beperkt gehouden met als doel u een beeld van de werkwijze te scheppen. Het onderdeel 'Toevoegen' moet hier in de ruime zin bekeken worden. Onder deze pagina wordt alles verstaan wat aanpassingen teweeg zal brengen in de gegevens. Naast het eigenlijk toevoegen van gegevens wordt ook het aanpassen en verwijderen van gegevens hier geplaatst, omdat dit toch binnen hetzelfde level geplaatst zal worden.

1 Home
- 1.1 Klant
  - 1.1.1 Toevoegen
  - 1.1.2 Overzicht
- 1.2 Project
  - 1.2.1 Toevoegen
  - 1.2.2 Overzicht
- 1.3 Taken
  - 1.3.1 Toevoegen
  - 1.3.2 Overzicht
- 1.4 Jobs
  - 1.4.1 …

Afhankelijk van de rechten van de aangemelde gebruiker zal deze structuur verschillen. Ook is het mogelijk dat de eigenlijke indeling van de pagina's en mappen zal verschillen. Aangezien er gewerkt zal worden met een framework zal er waarschijnlijk een bepaalde structuur aangehouden moeten worden. Dit zal echter in het onderdeel van de ontwikkeling opgeklaard worden.

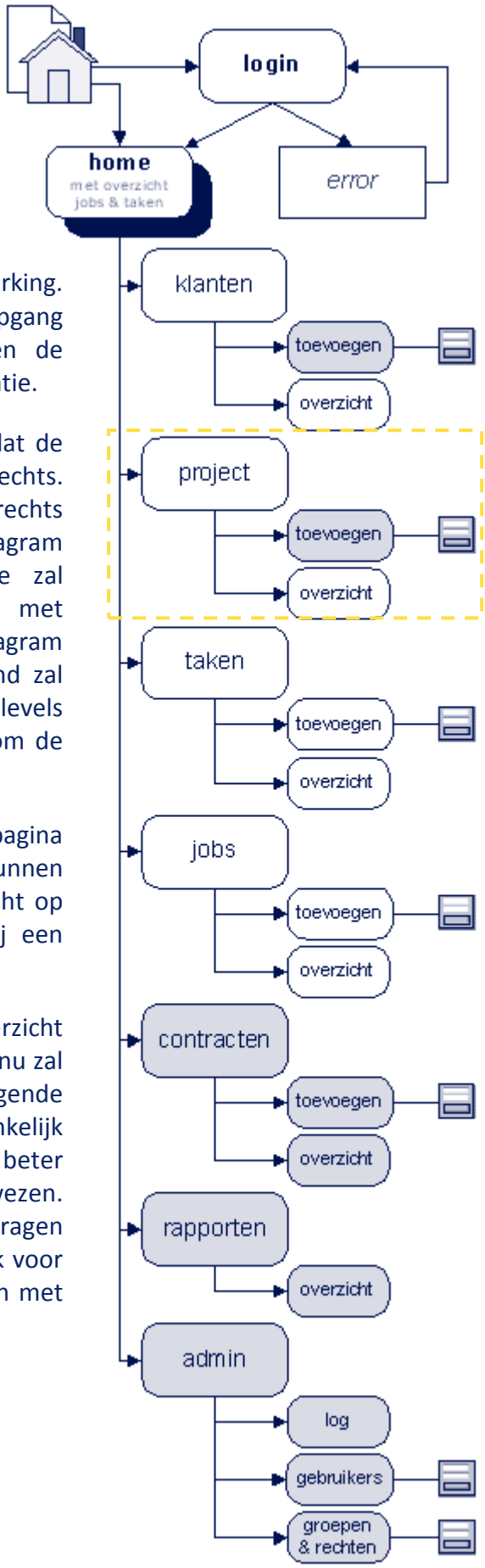
3.3.3 Diagram

Naast het uitschrijven zal ook hier weer een diagram gemaakt worden. In dit diagram zal bovenstaande structuur (de levels) logischerwijze terugkomen. Er zijn twee soorten diagrammen mogelijk, een horizontaal of verticaal diagram. Deze keuze is voor de analist zelf en maakt geen verschil voor de uitwerking. De keuze zal meestal afhankelijk zijn van de diepgang van de structuur en de manier waarop men de diagrammen zal willen afprinten in de documentatie.

Ik koos een horizontaal diagram. Dit betekent dat de levels uitgewerkt zullen worden van links naar rechts. Met level 1 uiterst links en de volgende levels rechts ervan. De volledige website wordt hier in één diagram verwerkt. Wanneer men een grote website zal uitwerken kan men het best werken met subdiagrammen. Men zal dan één globaal diagram maken met de level 1 en 2 pagina's. Bijkomend zal voor elke level in 2 pagina's alle onderliggende levels uitgewerkt worden in een nieuw diagram. Dit om de structuur zo overzichtelijk mogelijk te houden.

Mijn uitgewerkt diagram vindt u rechts op deze pagina terug. Vanop de startpagina zal de gebruiker kunnen inloggen. Bij het juist aanmelden komt hij terecht op de startpagina, bij het fout aanmelden zal hij een foutmelding krijgen en opnieuw kunnen inloggen.

Op de startpagina zal de gebruiker een overzicht krijgen van zijn huidige taken en jobs. Via het menu zal hij ook kunnen navigeren naar de onderliggende pagina's. De mogelijkheden zullen echter afhankelijk zijn van de rechten die aan deze gebruiker, of beter gezegd de groep waartoe hij behoort, zijn toegewezen. Een administrator zal meer pagina's kunnen opvragen dan een gewone gebruiker. De pagina's specifiek voor een administrator zijn in dit diagram aangegeven met een donkerdere achtergrond.



4 DATABASE

4.1 Inleiding

Als laatste belangrijk deel van de voorbereiding zal de eigenlijke database opgezet worden. De database die bij dit project gebruikt zal worden is een MySQL database. MySQL is een open source relationeel database management systeem (RDBMS). Voor het communiceren met een MySQL database wordt, zoals de naam al verklapt, gebruik gemaakt van SQL (Structured Query Language). De syntaxis voor het opstellen van MySQL database tabellen verschilt, weliswaar zeer licht, van de werkwijze die gebruikt wordt naar pakweg een Oracle databank toe.

Een relationele database is een database die werkt volgens een relationeel model. Dit betekent dat de gegevens worden opgeslagen in tabellen waarbij een rij een record vormt met alle samenhorende informatie. Verschillende tabellen kunnen met elkaar verbonden worden door een kolom toe te voegen waarin een verwijzing naar een record in een andere tabel wordt opgenomen. Deze verwijzing zal meestal gebeuren met een numerieke verwijzing naar het uniek id van het record waar men naar verwijst. Het zijn deze databases die gebruikt worden wanneer men gebruikt maakt van een relationeel database management systeem. Deze werkwijze wordt momenteel zeer actief toegepast, andere bekende relationele databases zijn DB2 (IBM), Access (Microsoft), Oracle, SQL Server (Microsoft),….

In de wereld van webapplicaties is MySQL echter niet het enige gebruikte RDBM-systeem. Een andere bekend systeem is PostgreSQL. Beide systemen zijn open source en dus gratis te gebruiken, het verschil moet zich dus in de functionaliteit zitten. Afhankelijk van uw behoeften zal u dus een keuze moeten maken tussen MySQL en PostgreSQL. Op de volgende pagina bespreek ik in het kort de verschillen tussen beide systemen.

4.1.1 Vergelijking MySQL - PostgreSQL

In het algemeen zijn de mogelijkheden (qua complexiteit) met PostgreSQL groter. MySQL is minder complex, waardoor het makkelijker maar ook sneller werkt. MySQL wordt over het algemeen ook beter ondersteund door hosting providers.

	MySQL	PostgreSQL
laatste versie	5.1	8.2.4
mogelijkheden	Beperkt, al begint MySQL de extra mogelijkheden die PostgreSQL al lang aanbiedt ook te integreren. Zo worden stored procedures sinds versie 5.0 ook ondersteund (weliswaar nog steeds beperkter).	Groot: werken met subqueries, stored procedures, triggers, cursors, views,…
snelheid	Snel (vooral bij tabeltype MyIsam)	Relatief langzamer.
algemene ondersteuning	Door zijn grotere bekendheid is er meer ondersteuning voor MySQL, zo wel wat support door internet providers betreft als wat informatie betreft (cursussen, online communities, boeken, …)	Ondersteuning is beperkter, als zijn er wel enkele goede mailingslists te vinden en zijn er tal van commerciële bedrijven die ondersteuning voorzien.
transacties	Kunnen enkel gebruikt worden bij een bepaald type tabellen (zoals InnoDB)	Alle queries zijn standaard transactioneel (zonder verdere instellingen).
relationeel	Ook hier wordt het relationeel werken bepaald door het tabeltype.	De referentiële integriteit is, mits een goede tabeldefinitie, altijd gegarandeerd.
Tools (GUI)	MySQL Query Browser	PgAdmin
Tools (web)	phpMyAdmin	phpPgAdmin
website	www.mysql.com	www.postgresql.org

Voor mij was de keuze echter snel gemaakt. Bij Sanmax werkt men zowel intern als extern met MySQL databases. Ik werk wel met tabellen van het type InnoDB en niet het standaardtype MyIsam. Waarom ik hiervoor gekozen heb, komt verder in dit hoofdstuk aan bod.

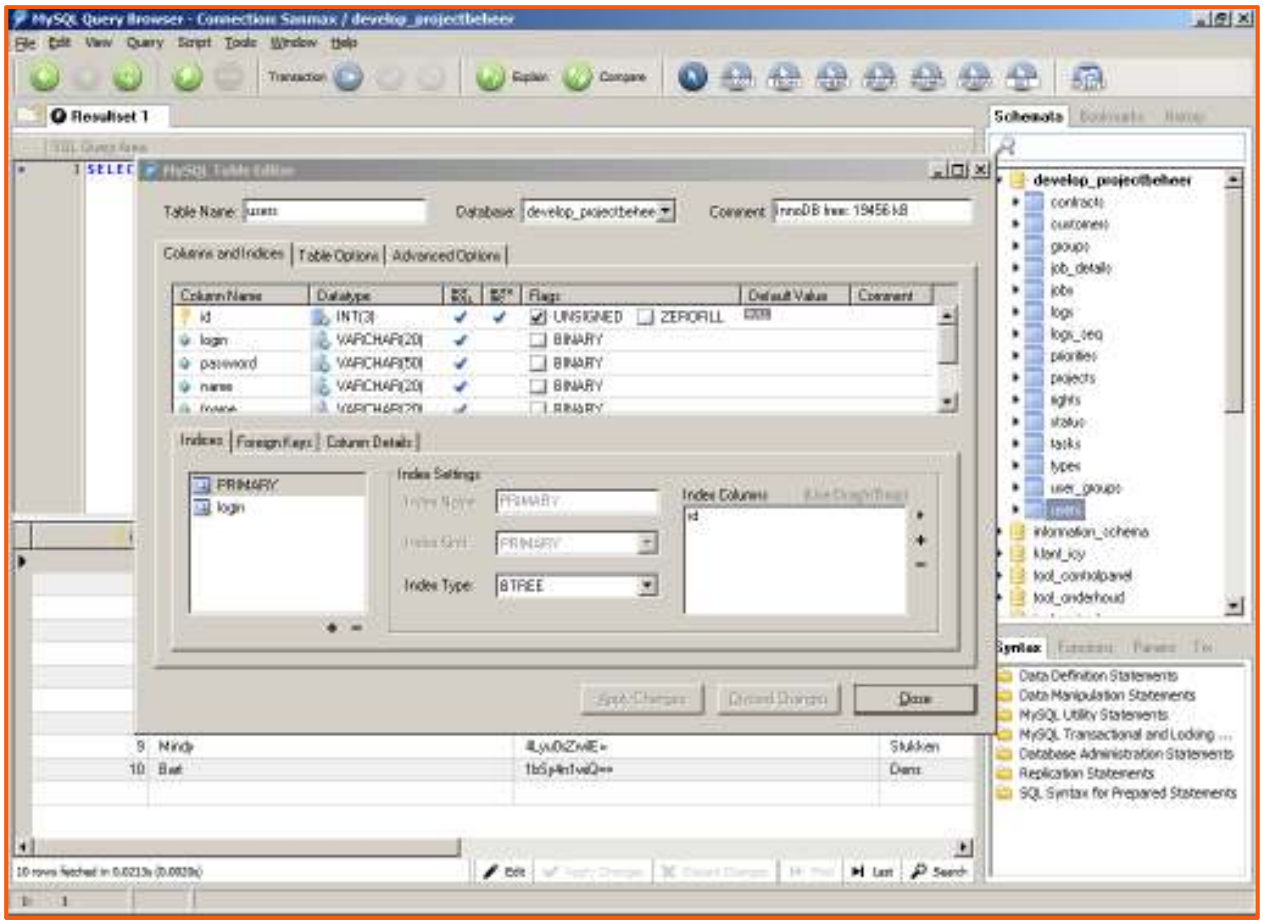
Zoek zeker ook eens via uw favoriete zoekmachine naar "postgresql vs. mysql".
U zult ook daar heel wat interessante en uitgebreide vergelijkingen en reviews terugvinden.

4.1.2 Administratie tools

Een bekende en veel gebruikte MySQL-frontend (applicatie met gebruikersinterface) is phpMyAdmin. Via deze webgebaseerd MySQL-administratie applicatie kunnen de MySQL databases op een gebruiksvriendelijke en relatief makkelijke manier beheerd worden. Ook dit is een open-source project dat dus gratis gebruikt kan worden.

Naast phpMyAdmin biedt men op de website van MySQL zelf ook nog een handige tool aan. Met behulp van MySQL Query Browser kan u niet alleen SQL-queries genereren, uitvoeren en optimaliseren, maar ook uw volledige database beheren met behulp van de object browser. Zo kunt u create- en insert-statements uitvoeren, maar ook via invoervelden aanpassingen en gegevens doorvoeren naar de database toe. Ook een handig functie is dat uw verschillende SQL-statements, en meer bepaald de output, kan vergelijken. Ingewikkeldere queries kunnen zo vergeleken worden met elkaar. Ook deze tool is op zijn beurt gratis.

Tijdens het opstellen van de scripts gebruikte ik zowel phpMyAdmin als MySQL Query Browser. In eerste instantie maakte ik een tabel aan via phpMyAdmin. Deze code exporteerde ik dan in een tekstbestand en hier werkte ik op verder voor de rest van de tabellen in de Query Browser. Zo maakte ik een volledig script voor het aanmaken van de nodige tabellen in de database. Een deel van dit script vindt u terug in het volgende onderdeel.



MySQL Query Browser (Table Editor)

4.2 Script

4.2.1 Voorbereiding

4.2.1.1 Storage engines: InnoDB vs. MyISAM

Allereerst wordt er een database aangemaakt waarin mijn tabellen terecht zullen komen. Het aanmaken van een database kan op een simpele manier gebeuren via phpMyAdmin. Er hoeft in principe enkel een naam voor de database ingesteld te worden. De database kreeg hier de naam 'tool_projectbeheer'. Natuurlijk kunnen er ook instellingen gedaan worden om bepaalde MySQL gebruikers verschillende rechten te geven. In mijn geval hebben er twee users toegang tot de database, namelijk mijn eigen MySQL-gebruiker 'davy' en een gebruiker 'projectbeheer' die ook gebruikt zal worden in de configuratiebestanden om vanuit de PHP webapplicatie een verbinding te maken naar de databank.

Het is ook belangrijk om na te denken welke functionaliteit de database zal krijgen. Afhankelijk hiervan kan voor de tabellen eventueel een ander storage engine (≈ type) gekozen worden. Ik koos niet voor de standaard storage engine 'MyISAM', maar voor 'InnoDB'. Wat zijn de verschillen tussen beide die voor mijn applicatie van belang kunnen zijn ?

	InnoDB	MyISAM
default Deze storage engine is standaard ingesteld.	ß	
transacties Transactie veilige ondersteuning: een handeling wordt ofwel volledig voltooid ofwel niet. (werken met rollback, commit,…).		ß
referentiële integriteit Mogelijkheid tot het leggen van relaties tussen tabellen door het gebruik van "foreign key constraints". Er wordt een controle gedaan om te voorkomen of juist te bekomen dat child-records al dan niet dezelfde actie ondergaan als de actie die gebeurt op het parent-record.		ß
locking niveau Voor het opvragen van data is dit minder belangrijk. Maar bij het toevoegen of aanpassen van gegevens zal de performantie bij rij-level locking groter zijn.	**rij**	**tabel**
opvraag duur Duur van het uitvoeren van een select-statement.	**<**	
aanpas duur Duur van het uitvoeren van een insert-, update- or delete-statement.	**>**	

Vooral de ondersteuning van referentiële integriteit (foreign keys constraints) gaf de doorslag om InnoDB tabellen te gebruiken. MyISAM is echter het meest geschikt wanneer er grotendeels informatie opgehaald zal worden uit de tabel. Wanneer er heel wat meer interactie met de database (zowel ophalen als wegschrijven) zal plaatsvinden, geniet InnoDB de voorkeur.

4.2.2 Script

De belangrijkste onderdelen van het script dat gebruikt wordt om de database op te vullen met tabellen vindt u hieronder terug in het onderdeel ‘projecten’. Dit script is zo opgebouwd dat na een uitvoering alle tabellen terug geïnitialiseerd zijn.

```
SET FOREIGN_KEY_CHECKS=0;
DROP TABLE IF EXISTS `projects`;
SET FOREIGN_KEY_CHECKS=1;
```

Omdat bij het resetten van de database het mogelijk is dat er (test-)gegevens aanwezig zijn in de tabellen en er dus gebruik gemaakt wordt van foreign keys moet de controle op deze foreign keys tijdig uitgeschakeld worden. Indien dit niet gebeurt zullen de tabellen niet verwijderd kunnen worden, of moet dit in een bepaalde volgorde gebeuren. (Error: Cannot delete or update a parent row: a foreign key constraint fails). Om het mij zelf gemakkelijk te maken wordt de controle tijdig op false gezet. Alle tabellen, wanneer ze bestaan, worden verwijderd. Verder wordt de controle terug aangezet wanneer er gestart wordt met het eigenlijk creëren van de tabellen. Vervolgens wordt de tabel aangemaakt, voor de tabel ‘projects’ wordt het volgende statement gebruikt:

```
CREATE TABLE `projects` (
 `id`             INT ( 5 )         UNSIGNED NOT NULL AUTO_INCREMENT ,
 `name`           VARCHAR( 20 )     NOT NULL ,
 `description`    BLOB              NOT NULL ,
 `customer_id`    INT ( 4 )         UNSIGNED NOT NULL ,
 `user_id`        INT ( 3 )         UNSIGNED NOT NULL ,
 `hours_budget`   INT ( 3 )         NULL ,
 `hours_perform`  INT ( 3 )         NULL ,
 `cost`           FLOAT ( 6 ,2 )    NULL ,
 `priority_id`    INT ( 2 )         NOT NULL ,
 `status_id`      INT ( 2 )         NOT NULL ,
 `date_end`       DATE              NULL ,
 PRIMARY KEY (`id`) ,
 FOREIGN KEY (`customer_id`) REFERENCES `customers` (`id`) ON DELETE NO ACTION,
 FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION,
 UNIQUE (`name`)
)ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Het statement start, zoals steeds, met een create table gevolgd door de naam van de tabel.

```
`cost`           FLOAT ( 6 ,2 )    NULL ,
```

De volledige tabelstructuur zal binnen de haken gedefinieerd worden. Dit gebeurt per kolom. De definitie van een kolom bestaat uit een kolomnaam, een data type[1)] en enkele opties. Onderstaand voorbeeld heeft de naam ‘cost’ en is het van type float (decimaal getal). Het bestaat in totaal uit 6 getallen, waarvan 2 na de komma. Het veld is niet verplicht want het kan en mag de waarde null bevatten.

[1)] http://dev.mysql.com/doc/refman/5.1/en/data-type-overview.html

```
PRIMARY KEY (`id`) ,
 FOREIGN KEY (`customer_id`) REFERENCES `customers` (`id`) ON DELETE NO ACTION,
 FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION,
 UNIQUE (`name`)
```

Aan het einde van de create kunnen de verschillende keys gedefinieerd worden. De key-definitie kan ook gebeuren per kolom, maar ik koos ervoor dit op het einde te doen om het zo overzichtelijk te houden. Allereerst wordt aangegeven welke kolom als primary-key zal fungeren. De 'on delete no action' is een foreign key constraint[1)] die voorzien kan worden omdat er gebruik gemaakt wordt van de storage-engine innodb. Wanneer in dit geval een klant verwijderd zou worden waaraan een project gekoppeld is, zal dit niet lukken omdat er staat dat er in dat geval "no action" mag plaatshebben. Andere mogelijkheden zijn vb.: "cascade" (wel mee verwijderen) of "set null" (null waarde in de plaats zetten, indien mogelijk).

```
)ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Tot slot zullen buiten de haken nog enkele eigenschappen van de tabel meegegeven worden, zoals de storage engine die gebruikt wordt (hier: InnoDB). Ook de karakterset die gebruik wordt, wordt hier gespecificeerd. Ik gebruik hiervoor utf8 (8-bit Unicode Transformation Format). Via dergelijke karakterset worden unicode-tekens (= internationale standaard voor de identificatie van grafische tekens en symbolen) omgezet en opgeslagen als een stroom van bytes, een zogenaamde tekencodering.

De volledige syntax en alle mogelijkheden voor een create van een tabel kan bekeken worden in de reference manual van MySQL.[2)]

```
ALTER TABLE `projects`
  ADD FOREIGN KEY (`status_id`) REFERENCES `status` (`id`) ,
  ADD FOREIGN KEY (`priority_id`) REFERENCES `priorities` (`id`);
```

Wanneer bepaalde tabellen nog niet aangemaakt zijn waarnaar eigenlijk een link gelegd zou moeten worden, kunnen deze foreign keys later nog toegevoegd worden via een alter table statement. Ook hier is dit het geval, de tabellen 'priorities' en 'status' waren nog niet gemaakt en er kon dus toen geen link naar gelegd worden.

```
INSERT INTO `priorities` VALUES
  (1 ,'low' ,'BLUE') ,  (2 ,'medium' ,'GRAY') ,
  (3 ,'high' ,'ORANGE') ,  (4 ,'very high' ,'RED');
```

Tot slot zijn in het script ook al een aantal inserts voorzien. Zo worden bij het uitvoeren van het script een aantal tabellen opgevuld. Tabellen als 'status', 'priorities', 'rights',… krijgen een beginwaarde omdat het in de applicatie niet voorzien is om wijzigingen en dus ook toevoegingen op deze tabel toe te passen.

Het volledige script vindt u achteraan deze bundel in de bijlage terug. Let wel op, dat script komt niet meer volledig overeen met het gegevensmodel en de databasestructuur. Tijdens en vooral naar het einde van de ontwikkeling zijn er nog een aantal wijzigingen aangebracht.

---

1) http://dev.mysql.com/doc/refman/5.0/en/innodb-foreign-key-constraints.html
2) http://dev.mysql.com/doc/refman/5.0/en/create-table.html

4.2.3 Aanpassingen

Gedurende de ontwikkeling zijn er, zoals net vermeld, nog enkele aanpassingen doorgevoerd aan de databasestructuur. Een reden van deze wijzigingen is te wijten aan de manier waarop de programmatie moet plaatsvinden. Zo is tijdens de analyse een rechtensysteem uitgewerkt dat feitelijk niet of moeilijker geïmplementeerd kan worden in het systeem dat voorzien wordt binnen de programmeeromgeving, maar bij aanvang van de analyse nog niet bekend was. Het is dan ook makkelijker en logischer om dit laatste te gaan toepassen en dus veranderingen door te voeren aan de databasestructuur.

Een tweede reden voor wijzigingen is het niet nauwkeurig genoeg verkrijgen van feedback op de analyse. De aanpassingen meestal bleven echter beperkt tot het vergroten van een veldlengte en het wijzigen van de constraints (not null, unique,…). Al moesten er soms nog enkele velden bijgevoegd worden.

Hieronder vindt u een opsomming van de belangrijkste aanpassingen en de bijhorende reden:

De grootste verandering gebeurde door het rechtensysteem dat toegepast zal worden. De tabellen ‘rights’ en ‘rightgroups’ worden volledig aangepast. De koppeltabel ‘rightgroups’ zal zelfs volledig verdwijnen. Het onderdeel van de rechten komt echter in een apart onderdeel van de ontwikkeling uitvoerig aan bod. Ik ga dan ook hier niet verder op de databasestructuur in, omdat dit in dit deel van de bundel ook nog niet volledig duidelijk zal zijn.

~~**rightgroups**~~

In de tabel met de projectenbeschrijvingen wordt de unique constraint verwijderd. Er kunnen dus projecten aangemaakt worden met de zelfde naam. Een voorbeeld: het is mogelijk dat Sanmax een project ‘webshop’ heeft lopen voor klant x en voor klant y.

projects

name	~~u~~

Ook in de tabelprojecten komen enkele noemenswaardige veranderingen. Allereerst kunnen heel wat velden de waarde ‘null’ wel aannemen en zijn ze dus optioneel. Enkel het interne nummer en de bedrijfsnaam blijven verplicht. Dit omdat niet altijd alle gegevens bekend zijn bij Sanmax, waar ik bij de analyse wel vanuit ging. Verder is er een veld ‘int_nr’ en ‘acc_nr’ toegevoegd voor een uniek intern- en boekhoudnummer en is er een commentaarveld voorzien. Tot slot is het veld met de status-id die verwees naar de tabelstatussen verwijderd geworden en is dit vervangen door een simpele boolean die bijhoudt of een klant afgesloten is of niet.

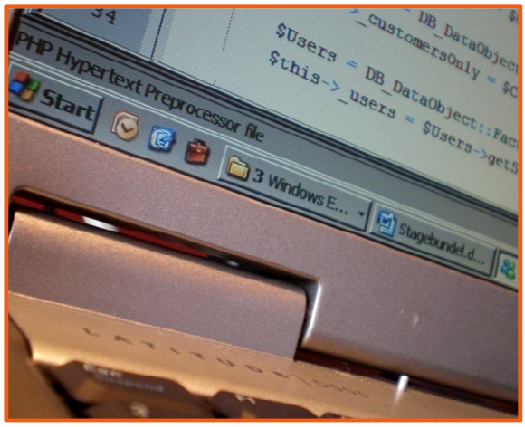
customers

int_nr	u	
acc_nr	u o	
company		
comment	o	
done	o	
~~status_id~~		~~f~~ ~~status~~

5 ONTWIKKELING

Ook in dit hoofdstuk zal slechts een onderdeel van de applicatie toegelicht worden. Ook zal niet alles in detail behandeld worden omdat het dan haast onmogelijk is om een goed overzicht te behouden. Zoals bij de analyse neem ik hier ook enkel de projectenbeheer pagina's onder de loep. Alle andere pagina's zijn wat structuur en opbouw betreft gelijkaardig. Zijn er toch noemenswaardige uitzonderingen of interessante gebruikte technieken dan worden deze apart behandeld.

5.1 Inleiding



Na de analyse en het opstellen van de databank tijd voor het "echte" werk: de ontwikkeling van de webapplicatie. Dit zal gebeuren met de analyse in het achterhoofd en meerbepaald de ontworpen use case diagrammen die de functionaliteit van de te ontwerpen applicatie duidelijk zichtbaar maken.

De ontwikkeling zal gebeuren in, wat men noemt, een LAMP omgeving. Dit is een acroniem voor een set van gratis vrije softwarepakketten die samen gebruikt worden voor het draaien van dynamische websites. Via dit acroniem worden onder andere de webserverinfrastructuur, de ontwikkelsoftware en het software distributiepakket gedefinieerd. Dit acroniem staat in dit geval voor het besturingssysteem Linux, de Apache webserver, het database management systeem MySQL en de scripttaal PHP. Een variant kan bijvoorbeeld WAMP zijn, waarbij Microsoft Windows als besturingssysteem gebruikt wordt.

In het Plan van Aanpak vermeld ik dat voor de ontwikkeling een bepaalde structuur gehanteerd wordt. Er zal dan ook gebruik gemaakt worden van een zelfontwikkeld framework dat gebaseerd is op het MVC-model. Voor bepaalde functies worden libraries uit het Zend-framework opgeroepen.

Verder wordt er voor de connectie naar de MySQL databank gebruik gemaakt van een PEAR-package. Dankzij dit package kunnen er op een relatief eenvoudige manier gegevens opgevraagd, weggeschreven en aangepast worden in de databank doordat er gebruik gemaakt wordt van dataobjecten. De eigenlijke interface die men te zien krijgt als gebruiker wordt gegenereerd met de template engine Smarty. Via deze templates worden de opgevraagde gegevens getoond of zal men gegevens ingeven die later toegevoegd zullen worden aan de database.

Al deze onderdelen worden op de volgende pagina's gedetailleerd besproken en eventueel met voorbeelden uit de code verduidelijkt. Verder vermeld ik aan het einde van dit hoofdstuk ook nog enkele handige tools die ik gebruikt heb om het werk te vereenvoudigen of efficiënter te laten gebeuren.

5.2 PHP: Hypertext Preprocessor

PHP is de scripttaal waarin ontwikkeld zal worden. PHP stond in eerste instantie voor Personal Home Page. Sinds PHP 3.0 is de betekenis een recursief acroniem geworden waarbij de betekenis staat voor "PHP: Hypertext Preprocessor". Via deze server-side scripttaal worden dynamische webpagina's ontwikkeld. Zoals de naam verklapt zal informatie omgezet worden naar hypertext (meestal HTML en/of XHTML).

Een klein stukje geschiedenis: PHP is in 1994 ontworpen door Rasmus Lerdorf, een software engineer bij 's werelds grootste IT-bedrijf IBM. De taal was indertijd duidelijk geïnspireerd van de programmeertaal Perl. Getuige daarvan is het dollarteken ($) dat gebruikt wordt om een variabele aan te duiden. De eerste publieke versie kwam uit in 1995, net als de tweede versie. In 1998 volgde de derde versie en in 2000 PHP 4.0. Sinds juli 2004 is de huidige versie, 5.0, uitgebracht. Deze versie kent al enkele nieuwe subversies. De meest recente en stabiele versie is PHP 5.2.2, deze is uitgebracht op 3 mei 2007.

Ondanks het feit dat PHP 5 al bijna 3 jaar geleden is uitgekomen beperken de meeste webservers zich nog steeds tot de 4.0 versie. Zijn de verschillen dan zo beperkt ? Toch niet. Sinds PHP 5 is het object georiënteerd programmeren aanzienlijk verbeterd. Er kan een hogere snelheid gehanteerd worden en de XML-bibliotheek is uitgebreid. Verder zijn er, zoals steeds, een aantal functies bijgekomen. Zo kan bijvoorbeeld er sinds PHP 5 een try-catch geprogrammeerd worden. En is het werken met AJAX (Asynchrone JavaScript and XML) en JSON (JavaScript Object Nation) ook aanzienlijk vergemakkelijkt.

5.2.1 .htaccess

Dezelfde situatie bij de servers van Sanmax. Standaard is hierop PHP 4 als (standaard) Apache module geïnstalleerd. Om toch gebruik te kunnen maken van de functionaliteiten van PHP 5 heeft men deze ook draaien in CGI-mode (Common Gateway Interface). Via een aanpassing in het .htaccess bestand zal aangegeven worden dat van deze module gebruik gemaakt moet worden voor het uitvoeren van de PHP scripts. Dit gebeurt met de volgende regel:

```
AddHandler php5-cgi .php
```

Via het AddHandler statement zullen de SSI (Server Side Includes) settings overschreven worden. Dit zijn de standaard server configuratie instellingen. Via bovenstaand statement wordt dus aangegeven dat gebruik gemaakt moet worden van de php5-cgi module voor .php bestanden.

Een .htaccess bestand, of ook wel "distributed configuration file" genoemd, geeft de mogelijkheid om configuratie aanpassingen te doen per directory. De aanpassingen aangegeven door dit bestand worden toegepast op de map en alle onderliggende mappen en hun bestanden. Bij elke benadering van een directory zal het .htaccess bestand ingelezen worden en zal er dus rekening gehouden worden met wat er in deze file aangegeven staat. Andere mogelijkheden van een .htaccess bestand zijn bijvoorbeeld het genereren van error-pagina's, het beveiligingen van een website of directory,….

5.3 Framework

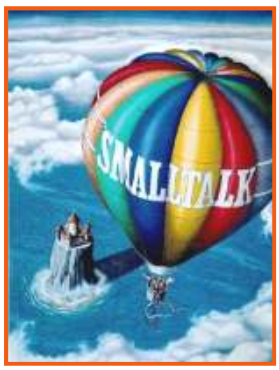
Het framework dat gebruikt wordt binnen Sanmax is door hen zelf ontwikkeld, en meerbepaald door Andries Seutens. Het door Andries ontworpen framework kan vergeleken worden met een beperktere versie van het open-source Zend-framework. Het Sanmax framework werd gelijktijdig ontwikkeld met het Zend-framework en zal, indien nodig, ook deels gebruik maken van verschillende componenten van dit framework, die elk afzonderlijk opgeroepen kunnen worden. Andries is zelf ook een Zend-medewerker[1)], hij werkt op vrijwillige basis mee aan de ontwikkeling van het Zend-framework.

Zoals al aangehaald is het framework gebaseerd op het MVC-model. Dit staat voor Model View Controller model, een werkwijze die de applicatie opdeelt in drie onderdelen. Via een bepaalde structuur en enkele bestanden die zorgen dat alles volgens die structuur loopt, zal de webapplicatie alle benodigdheden terugvinden en dus zijn werk kunnen doen.

Het grote voordeel van het werken met een framework is dat alles min of meer gestandaardiseerd is. Dankzij de overzichtelijke structuur weet u als ontwikkelaar direct wat u waar kan terugvinden of moet aanpassen. De programmatie verloopt dus efficiënter en sneller. Het enige nadeel is dat deze manier van werken een kleine aanpassing vraagt, maar eens u het systeem door hebt, is het heel plezant werken.

5.3.1 MVC-model

Het framework is gebaseerd op het MVC-model. Niet meer dan logisch dat ik dit model dan gedetailleerd onder de loep zal nemen. MVC staat voor Model View Controller en is een structuur dat het ontwerp van een, meestal complexere, toepassing opdeelt in drie eenheden met een verschillende verantwoordelijkheid. Doordat deze verantwoordelijkheden gescheiden worden, bevordert het de leesbaarheid en herbruikbaarheid van code. Een ander voordeel is dat veranderingen in de interface niet direct invloed moeten hebben op de code die instaat voor de communicatie met de database.



Ook hier kort de geschiedenis schetsen. De eerste implementatie van een MVC-model vond plaats in 1979 door Trygve Reenskaug. De implementatie was beschreven in een paper “Applications Programming in Smalltalk-80™: How to use Model-View-Controller MVC”[2)]. Smalltalk is een objectgeoriënteerde programmeertaal die ontwikkeld werd bij Xerox PARC.

[1)] http://framework.zend.com/contributors
[2)] http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

De eigenlijke bedoeling is om de presentatie layer of user interface te scheiden van de business logic. Men wil dus minimum een two-tier systeem bekomen. Zo kan men de zelfde gegevens op verschillende manieren tonen (als tabel, grafiek,…) zonder aanpassingen te moeten doen in de business logic.

U zult echter merken dat het MVC-model drie segmenten heeft. Om aan bovenstaande criteria te voldoen zouden twee segmenten toch volstaan, waarom dan drie ? De presentatielaag wordt nogmaals opgedeeld in een presentatie- en applicatiegedeelte. Daardoor dat MVC ook zo populair is in webapplicaties. De presentatie zal meestal gebeuren met behulp van een HTML pagina. De applicatielaag bevat de code-behind, deze zal reageren op een event of een handeling van de gebruiker. De business logic blijft bestaan en zorgt voor de verwerking van de gegevens die uit de database gehaald zullen worden.

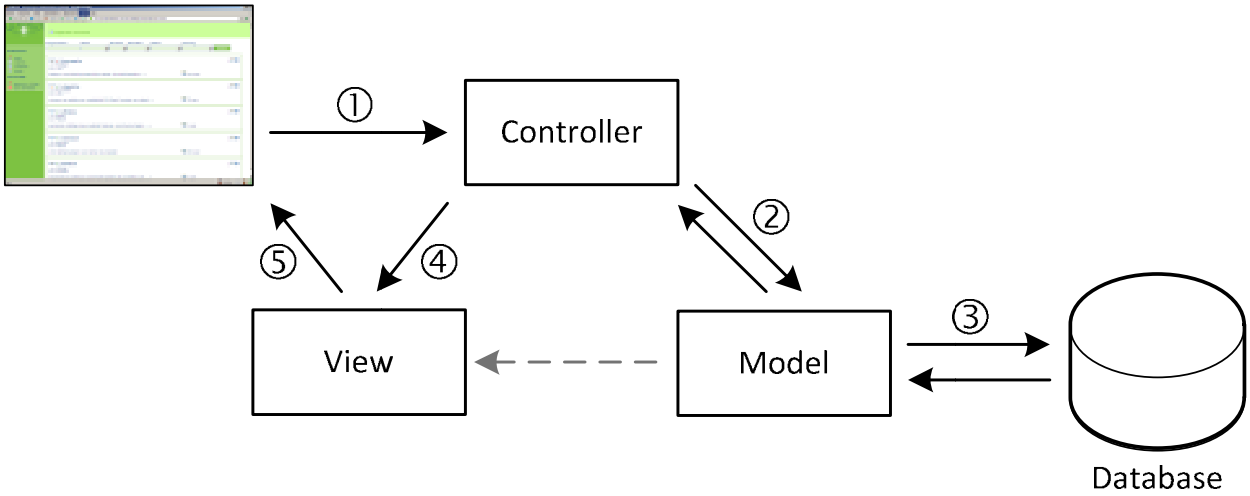
Om nu de termen te gebruiken die gehanteerd worden in MVC-model, nog even overlopen: De drie eenheden waarin de applicatie opgedeeld zal worden zijn de Model (datamodel), View (datapresentatie) en Controller (applicatielogica). Elk onderdeel behoeft de nodige uitleg zodat van het volledige model een totaal beeld geschetst kan worden.

Allereerst de **model (datamodel)**. Dit model representeert de informatie waarmee de applicatie werkt. Aan ruwe gegevens wordt een betekenis gegeven, door ondermeer relaties te leggen tussen de data logica. De daadwerkelijke opslag van data zal gebeuren met behulp van een persistent opslagmedium. Via een datalaag, die niet per se een onderdeel is van het MVC-model, zullen de gegevens opgehaald en weggeschreven worden naar de dataopslag. Het datamodel is onafhankelijk van zowel de datapresentatie als van de applicatielogica. Het zal dan bijvoorbeeld ook geen variabele mogen of kunnen bevatten die verwijzen naar de andere onderdelen.

Als tweede onderdeel is er de **view (datapresentatie)**. Alle gegenereerde of opgevraagde informatie wordt aangeleverd via een model en via de view getoond. User interface elementen worden in dit onderdeel gedefinieerd. Het tonen van gegevens kan op verschillende manieren gebeuren, met het gebruik van templates maar ook via een transformatieve methode zoals XSL (Extensible Stylesheet Language: definieert hoe XML getoond moet worden).

Tot slot de **controller (applicatielogica)**, deze reageert op events, die meestal veroorzaakt zullen worden door handelingen van de gebruiker van de applicatie. De controller ontvangt en vertaalt input naar aanvragen voor de model of voor de view. Controllers zijn verantwoordelijk voor het aanroepen van methodes gedefinieerd in het datamodel. Deze methodes zullen de status van het datamodel veranderen en mogelijk de status van de datapresentatie.

Op welke manier deze drie onderdelen samenhangen en met elkaar communiceren, wordt duidelijk in onderstaand schema en bijhorende beschrijving.



1 De gebruiker van de webapplicatie zal in interactie zijn met de user interface. Meestal zal de gebruiker een handeling doen, zoals het klikken van een knop. De controller behandelt het event.
2 De controller benadert de model. Afhankelijk van de gebeurtenis zullen er gegevens opgevraagd, weggeschreven, aangepast of verwijderd moeten worden.
3 Dit gebeurt vanzelfsprekend in communicatie met de databank.
4 De view zal de model gebruiken om de user interface te genereren met de passende gegevens die opgevraagd werden. De view ontvangt de data van de model, meestal via een omweg langs de controller. De model zelf heeft in feite geen weet van wat er in de view getoond zal worden, ze levert enkel de data aan.
5 De user interface wordt aangepast en wacht op verdere instructies van de gebruiker, waarbij de doorlopen cyclus terug van vooraf aan zal starten.

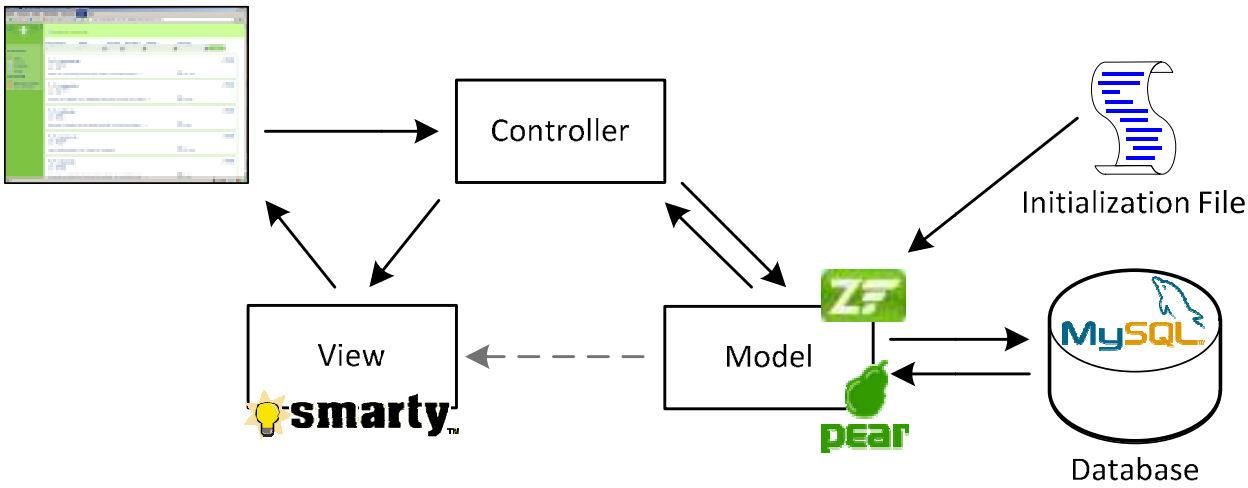
Om bovenstaande theoretische beschrijving nog beter te verduidelijken, even een concreet voorbeeld:

1 De Sanmax medewerker typt een projectnaam in de zoekbalk op de overzichtspagina van de projecten en klik op de knop zoeken. De controller ziet dat er een bepaalde actie plaatsvindt en voert de nodige code uit.
2 Hierbij roept de controller de model op verantwoordelijk voor de projectgegevens. De nodige gegevens worden opgehaald die voldoen aan de zoekopdracht op basis van de parameters die men ingevuld heeft.
3 Deze gegevens worden via een gegenereerd SQL-statement uit de MySQL databank gefilterd en opgehaald.
4 De opgevraagde gegevens zullen in een array geplaatst worden en via de controller aan de view teruggegeven worden. In de view zullen deze gegevens in een template doorlopen worden met een loop en een voor een getoond worden.
5 Het browser scherm wordt opnieuw ingeladen en alle projecten die voldoen aan de zoekopdracht worden getoond. De gebruiker kan, indien gewenst, een nieuwe handeling doen, bijvoorbeeld zijn zoekopdracht specificeren en de cyclus een tweede maal doorlopen.

5.3.2 Sanmax Framework

En nu concreet: hoe zit dat Sanmax framework in elkaar ? In dit onderdeel bekijk ik op welke manier de verschillende segmenten van het MVC-model concreet opgevuld worden binnen Sanmax. Vanaf hier zullen er ook geregeld fragmenten uit de code getoond worden om zeer specifiek aan te tonen op welke manier ik te werk ging.

Allereerst de concrete invulling. We nemen er terug het schema bij dat de structuur van een MVC-systeem aantoonde, maar nu ingevuld met de verschillende componenten die we daadwerkelijk gaan gebruiken.



- **Database:**
  Het volledige gegevensbeheer gebeurt in een MySQL databank.

- **Model:**
  De communicatie met de databank gebeurt met dataobjecten. Dit met behulp van het package DB_dataobject uit de PHP Extensie and Application Repository of kortweg PEAR. Dit package genereert SQL-statements en fungeert via een object interface naar de database tabellen. Er zijn ook een aantal models die niet naar de database toe werken maar gegevens teruggeven die in de code zelf gedefinieerd zijn. Ook mogelijk is het ophalen van gegevens uit bijvoorbeeld een ini (zie rechtensysteem) of xml bestand, dit zal gebeuren met behulp van een library van het Zend-framework (Registry).

- **Controller:**
  De controller is op een specifieke manier opgebouwd. Per controller zullen een aantal “actions” gedefinieerd worden. Deze actions zijn de eigenlijke events die zullen reageren op een bepaalde handeling van de gebruiker. De controllerbestanden zijn volledig in PHP geschreven.

- **View:**
  Het samenstellen van de user interface gebeurt met behulp van de template engine Smarty. De opbouw van deze pagina’s gebeurt in HTML code. De dynamische gegevens zullen met behulp van smarty-expressies getoond worden. Deze expressies hebben een specifieke syntax. Zo zullen ze bijvoorbeeld steeds tussen accolades ({ }) getoond worden.

De specifieke componenten worden in een apart hoofdstuk nog eens gedetailleerd besproken.

De eigenlijke applicatie moet natuurlijk zijn weg vinden binnen de structuur van de applicatie. Waar moet gezocht worden achter de models en waar vind ik de controllers terug ? Via welke HTML pagina toon ik het eindresultaat op het scherm van de gebruiker ? Deze structuur wordt vastgelegd aan de hand van een aantal voorwaarden. Zo moet er een bepaalde mappenstructuur en naamgeving aangehouden worden.

De globale structuur vindt u hieronder terug. Let op: het gaat hier om de concrete indeling van mijn applicatie. Op de root van de website directory staan heel wat subdirectories, zoals u ziet. Iedere directory heeft een bepaalde functie en een eigen inhoud. Bepaalde aspecten zullen pas echt duidelijk worden nadat u het vervolg van dit hoofdstuk overlopen hebt. Het is dus een goed idee om dit overzicht ook naderhand nog een keer onder de loep te nemen. De uitleg bij elke directory is zeer beknopt. Meer informatie per onderdeel vindt u terug in het onderdeel van de bundel waarvan u het volgnummer in de laatste kolom terugvindt.

 controllers	In de map 'controllers' bevindt zich voor iedere controller een PHP bestand terug. In elk bestand zullen per controller een aantal methodes gedefinieerd worden. Deze methodes komen overeen met een action. Een action is meestal een reactie op een event dat veroorzaakt wordt door de gebruiker.	*5.3.6*
 etc	In deze map staan een aantal configuratiescript. Het configuratiescript voor de connectie naar de databank, het script waarin alle mogelijk rechten opgesomd worden en de gegevens voor het laten genereren van de dataobject classes. Verder ook enkele txt-bestanden voor het opmaken van mails.	*5.3.4*
 libs	In de libraries vindt u verschillende submappen terug. Waaronder de libraries voor het Sanmax-framework, het Zend-framework en Smarty. Verder is er ook een map 'DB' terug te vinden. In deze map zijn de 'models' geplaatst. Voor elke tabel is er een PHP script met een class afgeleid van de DB_dataobject class.	*5.3.4* *5.3.6*
 schemas	In deze map bevinden zich twee ini-bestanden met gegevens over de database tabellen. Ze bevatten dan ook de naam van de database. Een bestand wordt automatisch aangemaakt bij het genereren van de dataobjecten, het bestand dat eindigt op links.ini is zelf aangemaakt.	*5.3.4*
 templates	Onder templates worden de Smarty-templates geplaatst. Voor elke controller wordt er hier een submap voorzien met identiek dezelfde naam. In deze submap wordt dan voor elke action een HTML bestand gemaakt waar de eigenlijke gegevens in getoond zullen worden. Ook de benaming van dit bestand is van belang.	*5.3.5*

Enkele minder relevante, maar daarom niet minder belangrijke, mappen:

 templates_c	In deze map worden de gecompileerde templates geplaatst. Deze map is misschien minder relevant voor de ontwikkeling zelf, maar moet wel verplicht aanwezig zijn.	*5.3.5*
 js	De JavaScript-bestanden komen in de map 'js' terecht.	*5.5*
 images	In deze map staan alle afbeeldingen en iconen gebruikt op de website.	

Voorts zijn er op de root ook nog een drietal bestanden terug te vinden die zorgen voor de eigenlijke initialisatie van de webapplicatie (en het framework). Ik start dan ook met deze te bespreken.

5.3.3 application.php

Het allereerste bestand dat zal opgeroepen worden is het 'bootstrap.php' bestand. Deze file kan vergeleken worden met de standaard index.php of index.html. Alle aanvragen zullen via dit centraal bestand binnenkomen. Het bootstrap bestand zet het framework op en initialiseert de code en scripts die hiervoor noodzakelijk zijn.

In feite doet deze bootstrap in mijn geval niets anders dan de application activeren. Zo maakt hij een object van de class SanmaxIntern aan en roept de publieke functie run() op.

bootstrap.php

```
set_include_path(get_include_path() . PATH_SEPARATOR . './libs');

require_once 'application.php';

$Sanmax = new SanmaxIntern();
$Sanmax->run();
```

Op die manier komen we al snel bij het tweede bestand 'application.php' terecht.

application.php

```
require_once 'Sanmax/Application.php';

class SanmaxIntern extends Sanmax_Application
{
```

Bij het bekijken van de class ziet u al direct dat het gaat om een afgeleide class. Er bestaat met andere woorden nog een superclass 'Sanmax_Application'. Deze class is een abstracte class. Dit wil zeggen dat deze class een of meerdere ongedefinieerde methoden bevat. Deze methoden worden geïmplementeerd in de subclass van de abstracte class. Doordat er gebruik gemaakt wordt van een abstracte class kan er ook geen rechtstreeks object aangemaakt worden van deze class maar moet men werken via een subclass. Door middel van overerving kan men in die subclass methodes van de abstracte class gebruiken.

Wanneer er gebruik gemaakt wordt van een class die gedefinieerd is in een ander (PHP) bestand moet dit aangegeven worden. Via de functie: 'require_once' wordt het opgeroepen script in feite eenmalig toegevoegd aan het lopende script, zodat er zonder problemen gebruik kan gemaakt worden van de daarin gedefinieerde functies. U vraagt zich misschien af waarom het niet nodig is om aan te geven dat 'Application.php' zich bevindt in /libs/Sanmax/ en dat het volstaat om aan te geven dat men moet gaan zoeken in de directory /Sanmax/. De reden hiervoor is dat we in de 'bootstrap.php' een include path gedefinieerd hebben met de functie set_include_path. De daarin aangeven paden zullen van links naar rechts doorlopen worden en men zal op zoek gaan naar het bestand. Zoals u ziet is daarin ook de directory /libs/ opgegeven.

```
    public function __construct()
    {
```

Voorts vinden we in de 'application.php' een publieke functie '__construct' terug. Deze functie wordt automatisch uitgevoerd wanneer een andere functie in die class aangesproken zal worden. Met andere woorden, wanneer de functie 'run()' opgeroepen zal worden, zal ook de code in deze functie uitgevoerd worden. Deze functie bevat heel wat 'require_once' functies

die er voor zorgen dat alle nodige bestanden ingeladen worden, zoals de nodige Sanmax Framework, Zend, PEAR en Smarty libraries.

```
public function run()
{
```

De belangrijkste functie is de 'run()' functie, diegene die opgeroepen werd vanuit 'bootstrap.php'. Deze bekijken we dan ook gedetailleerd, omdat ze de basis vormt van het eigenlijke framework.

```
session_start();
ob_start();
```

Allereerst zal de sessie data geïnitialiseerd worden. Deze sessie zal later gebruikt worden voor het bijhouden van alle gegevens van de aangemelde gebruiker. Ook wordt de output buffering aangezet, output zal niet vanuit de scripts getoond worden maar zal opgeslagen worden in een interne buffer waar ze later eventueel uitgelezen kan worden.

```
$config  = new Zend_Config_Ini('./etc/config.ini', 'production');
$acl     = new Zend_Config_Ini('./etc/acl.ini', 'production');
Zend_Registry::set('config',$config);
Zend_Registry::set('acl',$acl);
```

Een volgende stap is het uitlezen van twee belangrijke ini-bestanden (= initialization file). Hiervoor wordt een library ('Zend/Config/Ini.php') van het Zend-framework gebruikt, die dan ook in de '__construct' functie opgeroepen werd. Er wordt bij het uitlezen een sectie gespecificeerd. Wanneer de gegevens uit het ini-bestand ingelezen is, zullen deze geregistreerd worden in een variable. Ook dit gebeurt via een Zend-library ('Zend/Registry.php') die al opgevraagd werd. Wat bevindt zich nu in die initialisatiebestanden ? In config.ini wordt de configuratie bepaald van de database. De antwoorden op de vragen: "Wat zijn de gegevens van de database ?", "Waar vind ik de classes terug voor het aanmaken van de dataobjecten ?".

/etc/config.ini

```
; Production site configuration data
 [production]
db.database          =
  mysql://gebruikersnaam:wachtwoord@localhost/tool_projectbeheer
db.schema_location   =
  /home/data/websites/www/localhost/htdocs/projectbeheer/schemas
db.class_location    =
  /home/data/websites/www/localhost/htdocs/projectbeheer/libs/DB
db.require_prefix    =
  /home/data/websites/www/localhost/htdocs/projectbeheer/libs/DB
db.quote_identifiers =
  true
db.class_prefix      =
  DB_
```

Er kunnen meerdere secties gespecificeerd worden. Naast de 'production'-sectie die hierboven getoond wordt, zou er bijvoorbeeld ook een 'staging'-sectie aanwezig kunnen zijn. Dit maakt het mogelijk door een enkele verandering in 'application.php' over te schakelen van de testomgeving naar de eigenlijke productieomgeving. Het bestand 'acl.ini' wordt gebruikt voor het rechtenbeheer. Hier wordt later nog op teruggekomen.

```
$options = &PEAR::getStaticProperty('DB_dataobject', 'options');
$options = $config->db->toArray();
```

Voor het gebruiken van de dataobjecten moet er natuurlijk ook naar de database gekeken worden naar wat er in de database aanwezig is. Zoals al aangegeven wordt hiervoor gebruik gemaakt van een package van PEAR. Er wordt hier dan ook een functie opgeroepen van PEAR. Zoals u ziet wordt er gewerkt met een referentie (&), dit om ervoor te zorgen dat de link naar het object blijft bestaan. Er worden ook opties meegegeven, deze opties zijn afkomstig uit de variabele $config die we zojuist opgevuld hebben met de gegevens uit het ini-bestand.

```
$Smarty  = new Smarty;
$Smarty->force_compile = true;
$Smarty->plugins_dir[] = dirname(__FILE__) . '/libs/Smarty/plugins';
```

Ook van Smarty, dat gebruikt zal worden voor het genereren van de templates, zal er een instantie aangemaakt worden. Er worden ook direct twee eigenschappen gespecificeerd. Door de force_compile op true te zetten wordt aangegeven dat de templates bij elke aanroep (opnieuw) gegenereerd moeten worden. Deze opties moeten expliciet aangegeven worden want standaard staat dit op false. De gegenereerde templates komen terecht in de map ‘templates_c’. Ook hier wordt later op terug gekomen. Verder wordt ook het pad waar men de nodige plugins kan terugvinden, aangegeven.

```
$controller = strtolower($this->getController());
$action = strtolower($this->getAction());
```

Vanaf hier start de eigenlijke verwijzing naar de juiste pagina. Dit bestand zal namelijk, bij iedere aanvraag of handeling, bepalen welke pagina de gebruiker uiteindelijk te zien zal krijgen. De handeling die de gebruiker doet, zal bepaald worden aan de hand van de URL naar waar de gebruiker doorverwezen wordt of waartoe hij probeert toegang te krijgen. De URL is opgebouwd als volgt: http://domein/controller/action. Waarbij de controller staat voor de controller zoals we die tot nu toe kennen. En de action staat voor een bepaalde methode in die controller-class, of noem het een event dat gebeurt. Dit is momenteel misschien nog niet volledig duidelijk maar zal bij de bespreking van een controller wel duidelijk worden.

Hoe worden deze namen nu uit de URL gehaald ? Daarvoor moeten er enkele bestanden onder de loep genomen worden. Allereerst zal er in de code gewoon verwezen worden naar de URL zoals die hierboven opgebouwd werd. Namelijk naar http://domein/controller/action. Met behulp van het .htacces bestand zal deze URL herschreven worden. Dat gebeurt op de volgende manier. (Ik bekijk hier slechts één rewrite rule, er zijn er nog meer gedefinieerd, maar dit is momenteel de belangrijkste):

.htaccess

```
RewriteEngine on

RewriteRule ^([a-zA-Z-]+)/([a-zA-Z-]+)/?$
/bootstrap.php?controller=$1&action=$2&tpl=template [QSA,L]
```

Door gebruik te maken van de RewriteRule worden de URL die de naam van de controller en de action bevat herschreven naar de URL die het bootstrap bestand zal oproepen. De naam van de controller en action zal uitgelezen worden uit de huidige URL en meegegeven worden als parameters bij bootstrap.php. Als template wordt template meegegeven.

Deze parameters worden in abstracte class dan weer opgehaald. Dat gebeurt telkens wanneer deze class opgeroepen wordt, dus ook weer met een '__construct' functie:

/libs/Sanmax/Application.php

```
    protected function __construct()
    {
        ini_set('magic_quotes_runtime', false);
        ini_set('magic_quotes_gpc', false);

        $this->_controller = $this->_formatControllerName($_GET['controller']);
        $this->_action     = $this->_formatActionName($_GET['action']);
        $this->_template   = $this->_formatTemplateName($this->_controller);
    }

    public function getController()
    {
        return $this->_controller;
    }
    ...

    protected function _formatControllerName($unformatted)
    {
        return str_replace(' ', '_',
                   ucwords(strtolower(str_replace('-', ' ', $unformatted))));
    }
    ...
```

Met behulp van $_GET[ ] worden de parameters opgehaald uit de URL en, na ze om te vormen naar het juiste formaat, weggeschreven in een variabele van de class ($this). Met behulp van de nodige getters kunnen deze variabelen dan op hun beurt opgevraagd worden. Wat dus ook in het 'application.php'-bestand gebeurde met de 'getController()' en 'getAction()' functie.

Nu er geweten is welke controller en action er getoond moet worden kunnen deze variable toegewezen worden aan een template. Als u application.php bekijkt zal u zien dat er op momenteel heel wat meer code aanwezig is. Deze code is ter controle van de rechten van de gebruiker. De controle die er gedaan wordt, is of de gebruiker wel de rechten heeft om die controller en bijhorende actie op te roepen. Het volledig rechtensysteem wordt echter apart besproken dus beperken we ons hier tot de relevante code.

```
        $Smarty->assign('controller', $this->getController());
        $Smarty->assign('action',     $this->getAction());
        $Smarty->assign('template',   $this->getTemplate());

        $Smarty->display('Common/' . $_GET['tpl'] . '.html');
    }
```

De opgevraagde controller, action en template zullen toegewezen en getoond worden. De assign methode wordt gebruikt om de template variabelen in te vullen. Met de methode display wordt het type template opgehaald. In dit geval is dat template.html, maar er zijn ook nog andere mogelijkheden. (Zie bijvoorbeeld Modal Popup (5.5.1)).

Dit wat de globale code betreft. Op de volgende pagina's zal een voorbeeld, dat van de projectbeheer-pagina nog concreter aangetoond worden, en dat zowel voor de model, de controller als de view. Eens dat u dit principe volledig door hebt, is het voor alle andere onderdelen in feite hetzelfde.

5.3.4 Model (PEAR DB_dataobject)

Het eerste grote onderdeel dat in detail besproken wordt, is de model. Zoals al aangegeven representeert dit model de informatie waarmee de applicatie werkt. In het geval van mijn applicatie zal de informatie opgehaald worden uit een MySQL databank en geplaatst worden in dataobjecten. Met deze dataobjecten zal dan in de applicatie gewerkt.

Voor het beheer en gebruik van de dataobjecten in de ontwikkelde applicatie wordt gebruik gemaakt van een package uit “The PHP Extension and Application Repositry” of kortweg PEAR, namelijk het package “DB_dataobject”. Wat bevat dit package en wat is de eigenlijke functie van zo een dataobject ? Met behulp van dit package kan u gebruik maken van een SQL Builder. Deze builder zal deels automatisch SQL statements opbouwen aan de hand van de objectvariabelen en enkele gedefinieerde methodes. Een dataobject zelf handelt eigenlijk als dataopslag voor een rij uit een specifieke tabel.

5.3.4.1 Generator

Dit package bevat ook een generator die de nodige configuratie en basis classes aanmaakt. Het aanmaken van deze classes gebeurt dus niet handmatig, althans niet wat de class variabelen en basis statements betreft, maar daarop kom ik zo dadelijk nog terug. Allereerst moet de generatie plaatsvinden, hiervoor moeten er eerst een aantal bestanden aangemaakt of juist geconfigureerd worden. Allereerst is er de initialization file ‘generate.ini’ nodig. In deze file zullen een aantal eigenschappen meegegeven worden.

/etc/generate.ini

```
[DB_dataobject]
database          =
  mysql://gebruikersnaam:wachtwoord@localhost/tool_projectbeheer
schema_location   =
  /home/data/websites/www/localhost/htdocs/projectbeheer/schemas
class_location    =
  /home/data/websites/www/localhost/htdocs/projectbeheer/libs/DB
require_prefix    =
  /home/data/websites/www/localhost/htdocs/projectbeheer/libs/DB
quote_identifiers = true
class_prefix      = DB_
```

Hierin wordt de database gespecificeerd. Voor iedere tabel in deze database zal een dataobject class aangemaakt worden. Ook de locatie waar men de schema’s en de classes moet plaatsen. Tot slot wordt nog aangegeven dat de identifiers omringt mogen worden met aanhalingstekens en dat we voor elke gegenereerde class een prefix voorzien. In het volgende voorbeeld zal voor de tabel ‘projects’ dus een class aangemaakt worden die ‘DB_projects’ zal noemen.

Het effectief genereren van de databaseclassen gebeurt door het surfen naar een URL die er als onderstaande uitziet. Op de localhost van Sanmax staat een pagina met alle links voor alle projecten. Er wordt een PHP script (generate.php) opgeroepen dat als parameter het pad van bovenstaande ini-file meekrijgt en aan de hand daarvan de dataobjecten zal genereren.

```
/generate.php?ini=/home/data/websites/www/framework.davy.be/etc/generate.ini
```

Naast het creëren van de DB-classes zal er ook een schema aangemaakt worden. Dit schema wordt geplaatst in de aangegeven directory, in mijn geval /schemas. Het zal de naam van de database krijgen en dat wordt voor mijn situatie: tool_projectbeheer.ini. Dit bestand bevat de configuratie van de database, en meerbepaald van de verschillende tabellen. Per tabel zijn er twee blokken voorzien, een voor de tabel zelf en een voor de keys van de tabel. Voor de tabel projecten ziet dit er als volgt uit:

/schemas/tool_projectbeheer.ini

```
 [projects]
id = 129
name = 130
description = 194
customer_id = 129
user_id = 129
hours_budget = 1
hours_perform = 1
cost = 1
priority_id = 129
status_id = 129
date_end = 6

[projects__keys]
id = N
```

Let wel op: het is niet de bedoeling om aanpassingen te doen in dit bestand. Dit bestand wordt automatisch mee gegenereerd bij de generatie van de dataobject-classes. Wanneer er dus aanpassingen gebeuren aan de structuur of veldtypes van een tabel is het de bedoeling dat al deze bestanden opnieuw gegenereerd worden. Aangezien de ontwikkelaar nooit rechtstreeks met dit bestand zal werken, is het ook niet echt nodig om het te begrijpen, toch overloop ik het even. Want, ook al komt u er niet rechtstreeks mee in contact, belangrijk is dit bestand wel !

De eerste blok bevat de naam van de tabel en alle kolomnamen van die tabel. Het cijfer dat er steeds getoond wordt is de binaire waarde voor het veldtype. Zo staat bijvoorbeeld is 1=integer, 2=string, 6=date, 128=not null, 129=integer en not null, enzovoorts. De tweede blok bevat de keys van de tabel.

Wanneer er gebruik gemaakt gaat worden van de DB_dataobject methodes zoals getLinks(), en joinAdd() zal er nog een extra bestand voorzien moeten worden. Dit bestand zal dezelfde naam krijgen maar eindigen op .links.ini. Dit bestand moet wel handmatig aangemaakt worden.

/schemas/tool_projectbeheer.links.ini

```
 [projects]
customer_id = customers:id
user_id = users:id
priority_id = priorities:id
status_id = status:id
```

In dit bestand zal ook per tabel een blok voorzien worden. De velden die refereren naar een veld in een andere tabel zullen hierin opgegeven worden. De veldnaam van de tabel wordt gelijkgesteld aan de tabel- en veldnaam naar waar het refereert, waarbij de laatst vernoemden gescheiden worden door een dubbele punt.

5.3.4.2 DB_dataobject

Concreet voor de tabel 'projects' betekent het laten genereren van de dataobjects dat het bestand 'projects.php' aangemaakt wordt in de directory /libs/DB. Dit bestand bevat na de generatie de volgende automatisch gegenereerde code:

/libs/DB/Projects.php

```
<?php
/**
 * Table Definition for projects
 */
require_once 'DB/dataobject.php';

class DB_Projects extends DB_dataobject
{
    ###START_AUTOCODE
    /* the code below is auto generated do not remove the above tag */

    var $__table = 'projects';           // table name
    var $id;            // int(5) not_null primary_key unsigned auto_increment
    var $name;              // string(120)  not_null
    var $description;       // blob(65535)  not_null blob binary
    var $customer_id;       // int(4)  not_null multiple_key unsigned
    var $user_id;           // int(3)  not_null multiple_key unsigned
    var $hours_budget;      // int(3)
    var $hours_perform;     // int(3)
    var $cost;              // real(6)
    var $priority_id;       // int(2)  not_null multiple_key
    var $status_id;         // int(2)  not_null multiple_key
    var $date_end;          // date(10)  binary

    /* ZE2 compatibility trick*/
    function __clone() {
        return $this;
    }

    /* Static get */
    function staticGet($k,$v=NULL) {
        return DB_dataobject::staticGet('DB_Projects',$k,$v);
    }

    /* the code above is auto generated do not remove the tag below */
    ###END_AUTOCODE
}
```

Zoals u merkt wordt bovenaan het bestand toegevoegd waarin de class DB_dataobject zich bevindt. De class waarvan deze gegenereerde class een aantal methodes zal overerven. Verder zien we ook duidelijk dat het prefix dat gespecificeerd werd in het initialisatie bestand in de class naam duidelijk voorkomt. De 'autocode' zoals aangegeven in commentaar bevat alle velden van de tabel in de vorm van een variabele en enkele extra functies zoals een clone-functie voor het makkelijk klonen van een dataobject.

In de class DB_dataobject, worden heel wat methodes uitgeschreven die, dankzij de overerving, ook opgeroepen kunnen worden op een object van een specifieke dataobject class. Het is echter ook mogelijk om zelf een aantal, meer specifieke, methodes uit te schrijven. Ook daarvoor zal gebruik gemaakt worden van methodes uit de superklasse. Het lijkt me dan ook nuttig de meest belangrijke methodes even kort te bespreken. Alle methodes kan u terugvinden in de online PEAR-manual[1)]. De Engelse handleiding is het meest up-to-date, ik raad dan ook aan deze te gebruiken, ondanks de aanwezigheid van een Nederlandstalige.

factory()	Laad de klasse op basis van de tabelnaam die als parameter wordt meegegeven. Alle onderstaande methodes kunnen toegepast worden op het object dat men terugkrijgt als resultaat van deze functie. Parameter: string $table – automatisch de eerste gevonden rij fetchen Return: object / false
query()	Verstuur een volledige (insert/select/update/delete) query naar de databank. Parameter: string $string – SQL Query
find()	Het samenstellen en uitvoeren van de huidige query, gebaseerd op de object-variabele en eventuele whereAdd's. Parameter: boolean $autofetch – automatisch de eerste gevonden rij fetchen Return: int – aantal gevonden rijen
fetch()	De fetch methode haalt de volgende rij op en vult de objectvariabelen op met de gegevens uit de rij. Return: boolean – true bij succes, false bij wanneer het fetchen niet meer lukt
get()	Een simpele get (select) aanvraag, standaard op basis van de key van de tabel of op basis van een specifieke opgegeven veldnaam. Parameter: $value (primary key) / $key, $value (voor een specifieke veldnaam) Return: int – aantal gevonden rijen
insert()	Toevoegen van data aan de database, gebaseerd op de waarde aanwezig in de variabelen van huidig object. Parameter: $value / $key, $value (wanneer voor een specifieke veldnaam) Return: int – id (primary key) van de toegevoegd rij, 0 wanneer niet gelukt
update()	Updaten van data in de database, gebaseerd op de waarde aanwezig in de variabelen van het huidige object. Parameter: Dataobject $original (zal enkel de wijzigingen aanpassen het dataobject) Return: int – aantal aangepaste rijen
delete()	Verwijderen van data uit de database, op basis van de primary key van een object of door het toevoegen van een whereAdd. Parameter: boolean $use_where (=DB_DATAOBJECT_WHEREADD_ONLY) bij whereAdd Return: int – aantal verwijderde rijen
toArray()	Maakt een array van het huidig resultaat. Parameter: string $format – voor het eventueel formatteren van de gegevens
setFrom()	Kopieert de items van een array of een object in het huidige object. Dit zal

[1)] http://pear.php.net/manual/en/package.database.db-dataobject.php

	bijvoorbeeld gebruikt worden wanneer de gegevens van een formulier ($_POST) in een object opgenomen zullen worden (voorwaarde: veldnamen van het formulier komen overeen met de variabelen van het dataobject). Parameter: array $from / object $from
selectAdd()	Toevoegen van een kolom aan een select-statement. Standaard wordt bij een select-query alle velden geselecteerd (select * from). Om dit te voorkomen kan u eerst een lege selectAdd() doen zonder argumenten en dan elk benodigd veld toevoegen aan de select. Parameter: kolomnaam (eventueel voorafgaand door de tabelnaam.)
whereAdd()	Toevoegen van voorwaarde aan het 'where' gedeelte van een query. Parameter: string $cond, de conditie waaraan voldaan moet worden (leeg = reset) string $opt, hoe samenvoegen van meerdere ? "OR" of "AND" (=default)
orderBy()	Toevoegen van een 'order by'-conditie. Parameter: string $order – een of meerdere kolomnamen
groupBy()	Toevoegen van een 'group by'-conditie Parameter: string $group – een of meerdere kolomnamen
joinAdd()	Toevoegen van een ander bestaand dataobject voor het kunnen samenstellen van een join-query. (Let op: zoals al vermeldt, moet hiervoor een .links.ini-bestand aangemaakt zijn in de schemas-directory) Parameter: object $obj – het te joinen object string $joinType – "LEFT", "INNER" of "RIGHT" string $joinAs – voor het hernoemen van de tabel waarop gejoind wordt string $joinCol – de kolom die overeen moet komen (bij meerdere links)

Deze functies zullen steeds opgeroepen worden op basis van een dataobject, wat logisch is, aangezien het methoden zijn die in de superklasse van een dataobject class gedefinieerd zijn. Deze methoden kunnen dus opgeroepen worden in de dataobject class zelf, maar ook in de controller. Dit laatste is geen enkel probleem als het gaat om een simpele get-select. Maar zeker af te raden wanneer het om een uitgebreide methode gaat met bijvoorbeeld verschillende join- en where-condities.

5.3.4.3 Voorbeeld

Nu we alle methoden overlopen hebben bespreek ik even een zelfgedefinieerde functie. Deze functie is een van de uitgebreidste functies die er in het project terug te vinden is, maar bevat wel heel wat methoden die hierboven theoretisch uitgelegd zijn. Ik beschrijf kort even het doel van deze functie zodat u de code, dankzij de nodige commentaar, goed kan begrijpen.

De functie 'search(Array $parameters)' is gedefinieerd in de class 'DB_Projects' en zal, zoals de naam het verklapt, een of meerdere projecten zoeken. Er zal bij het oproepen van deze functie vanuit de controller een array meegegeven worden. Deze array bevat een aantal zoekopties die de gebruiker heeft opgegeven in de zoekbalk. De gebruiker kan in formuliervelden aangeven dat men wil zoeken op projectnaam, op klant,… en zal door te klikken op een knop deze zoekfunctie oproepen. De functie zal uiteindelijk een array teruggeven met de projecten en heel wat extra relevante informatie die aan de zoekopdracht voldoen.

/libs/DB/Projects.php

```
public function search(Array $parameters)
{
  $validKeys = get_class_vars(get_class($this));
  foreach ($parameters as $key => $value) {
    if (array_key_exists($key, $validKeys)) {
        if (strlen($value) > 0){
          if ($key=='user_id' OR $key=='customer_id' OR $key=='status_id')
              $this->whereAdd("projects.$key = $value");
            else if ($key=='hours_budget')
              $this->whereAdd("projects.$key $value");
            else{
              $value = preg_replace('/[^a-zA-Z0-9]/', '%', $value);
              $this->whereAdd("projects.$key LIKE '%$value%'");
            }
        }
      }
  }
```

Opvragen van de eigenschappen (variabelen) van een klasse, in dit geval, deze klasse. De array die meegegeven werd als parameter gaat doorlopen worden. Er zal voor elk element in de array gecontroleerd worden of er hiervoor een variabele bestaat in de class. Afhankelijk van de key wordt er een where statement toegevoegd. Meestal zal het gaan om een “projects.$key = $value” die in het where-gedeelte wordt geplakt. Maar bij bijvoorbeeld ‘hours_budget’ kan de waarde gelijk zijn aan “between 50 and 100”. Voor alle andere keys, zoals de projectnaam zullen er jokerstekens toegepast worden. Alle tekens die niet alfanumerieke of numerieke tekens zullen vervangen worden door een jokerteken (%) Voor en na de waarde die de gebruiker ingevoerd heeft zal ook een jokerteken (%) geplaatst worden. De ingave “D^avy” zal dus “%D%avy%”worden.

```
  $this->selectAdd();
  $this->selectAdd('projects.*');
```

Met de selectAdd() zonder argument wordt voorkomen dat bij de select alle gegevens opgehaald worden van alle (gekoppelde) tabellen. Zonder deze regel zou de select in principe beginnen als volgt: select * . Daarna selecteren we alle velden uit de projecttabel (‘projects.*’). Merk op dat er hier enkele aanhalingstekens gebruikt worden en bij de whereAdd() hierboven dubbele aanhalingstekens. De reden hiervoor is dat de variabele bij de whereAdd() herkend moeten kunnen worden als variabele en niet als tekst.

```
  $status = DB_dataobject::Factory('status');
  $this->joinAdd($status);
  $this->selectAdd('status.id as status_id');
  $this->selectAdd('status.style as status_style');
  $this->selectAdd('status.name as status_name');
```

Een record uit de ‘projects’-tabel bevat een aantal id’s die verwijzen naar een andere tabel. Het id tonen zou niet echt informatief zijn voor de gebruiker, daarom wordt de bijhorende benaming en eventuele nog andere gegevens opgehaald uit die tabel. Hiervoor moet die tabel gejoind worden op de huidige tabel. Dat gebeurt met behulp van de joinAdd()-methode. Hierna kunnen er met de functie selectAdd() naar believen kolommen van deze tabel toegevoegd worden. Eventueel met een alias benaming. (Dit is noodzakelijk bij twee identieke kolommen, hier bijvoorbeeld id).

```
    $priorities = DB_dataobject::Factory('priorities');
    $this->joinAdd($priorities);
    $this->selectAdd('priorities.style as priority_style');
    $this->selectAdd('priorities.name as priority_name');
    $customers = DB_dataobject::Factory('customers');
    $this->joinAdd($customers);
    $this->selectAdd('customers.company');
    $this->selectAdd('customers.id as cust_id');
    $this->selectAdd('customers.name as cust_name');
    $this->selectAdd('customers.fname as cust_fname');
    $this->selectAdd('customers.city as cust_city');
    $users = DB_dataobject::Factory('users');
    $this->joinAdd($users);
    $this->selectAdd('users.login as user_login');
```

Dit gebeurt ook voor de andere refererende velden op identiek dezelfde manier. Vergeet vooral niet telkens een object aan te maken van de dataobject class voor een bepaalde tabel.

```
    $this->orderBy('projects.status_id');
    $this->orderBy('projects.priority_id');
```

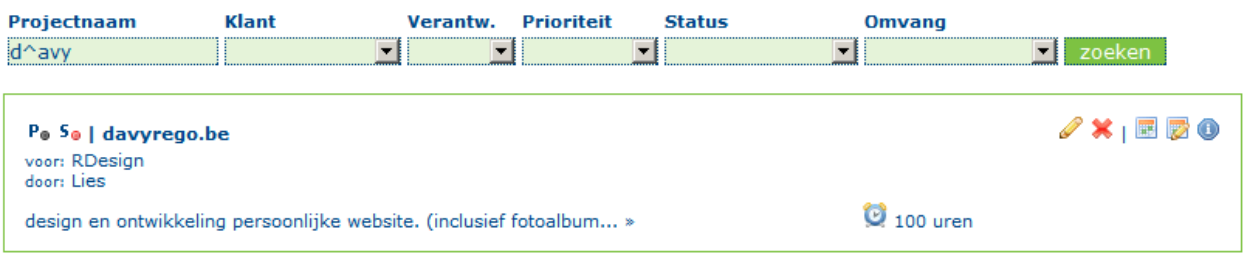
Alles wat uiteindelijk geselecteerd wordt, zal worden gesorteerd op stutus_id en priority_id.

```
    $this->find();
    $projects = array();
    while ($this->fetch()) {
        array_push($projects, $this->toArray());
    }
    return $projects;
}
```

Uiteindelijk wordt de query samengesteld en uitgevoerd. Wanneer de query uitgevoerd is, kan gestart worden met het fetchen van de rijen. Bij het fetchen wordt het object en meerbepaald zijn variabele opgevuld met de gegevens uit de opgehaalde rij. Deze gegevens worden op hun beurt in een array gestopt. Een array die voor de while-lus wordt geïntialiseerd, wordt opgevuld met die array's.

Als eindresultaat zal dus een array teruggegeven worden die voor elk project dat voldeed aan de zoekactie een array bevat.

Om u een idee te geven van wat deze functie nu eigenlijk als eindresultaat heeft, wordt deze functie uitgevoerd. Deze functie wordt opgeroepen wanneer er op de overzichtspagina van de projecten een naam ingevoerd wordt in de zoekbalk bovenaan. Ik neem het voorbeeld dat hierboven ook al aangehaald was en ik zoek op de naam "D^avy". Het resultaat ziet u hieronder:



Maar wat is er nu eigenlijk gebeurd ? De query die bovenstaande functie samengesteld en uitgevoerd heeft, is de volgende:

SQL-query

```
SELECT projects.* ,
       status.id as status_id ,
       status.style as status_style ,
       status.name as status_name ,
       priorities.style as priority_style ,
       priorities.name as priority_name ,
       customers.company ,
       customers.id as cust_id ,
       customers.name as cust_name ,
       customers.fname as cust_fname ,
       customers.city as cust_city ,
       users.login as user_login
FROM   `projects`
INNER JOIN `tool_projectbeheer`.`status`
      ON `tool_projectbeheer`.`status`.`id`=`projects`.`status_id`
INNER JOIN `tool_projectbeheer`.`priorities`
      ON `tool_projectbeheer`.`priorities`.`id`=`projects`.`priority_id`
INNER JOIN `tool_projectbeheer`.`customers`
      ON `tool_projectbeheer`.`customers`.`id`=`projects`.`customer_id`
INNER JOIN `tool_projectbeheer`.`users`
      ON `tool_projectbeheer`.`users`.`id`=`projects`.`user_id`
WHERE ( projects.name LIKE '%d%avy%' )
ORDER BY projects.status_id , projects.priority_id;
```

Het resultaat dat uiteindelijk gevonden en gefetched wordt is het volgende, weliswaar ingekort, maar het geeft u toch een impressie van wat het resultaat is.

Resultaat

```
DB_Projects: find: CHECK autofetchd
DB_Projects: find: DONE
DB_Projects: FETCH: ...
DB_Projects: fetchrow LINE: id = 1
DB_Projects: fetchrow LINE: name = davyrego.be
DB_Projects: fetchrow LINE: description = design en ontwikkeling ...
DB_Projects: fetchrow LINE: customer_id = 1
...
DB_Projects: fetchrow LINE: cust_fname = Davy
DB_Projects: fetchrow LINE: cust_city = Alken
DB_Projects: fetchrow LINE: user_login = Lies
DB_Projects: fetchrow: projects DONE
DB_Projects: FETCH: N;
DB_Projects: FETCH: Last Data Fetch'ed after 0.00111198425293 seconds
```

Nog even kort en misschien ook wel interessant om te weten is hoe u deze resultaten kan opvragen. Door het toevoegen van onderstaande regel zal er bij de uitvoer van een pagina, waarop met dataobjecten gewerkt wordt, een output getoond worden. Dit is zeer handig, zeker wanneer de data niet juist getoond of weggeschreven wordt. Waarschijnlijk is er dan een fout geslopen in de opbouw van de methode. Dankzij het debuggen kan deze op een relatief makkelijke manier opgespoord worden. Het getal dat als argument meegegeven wordt bepaald in welke vorm de informatie teruggegeven wordt, van beperkt (1) tot zeer gedetailleerd (5).

```
db_dataobject::debuglevel(5);
```

5.3.4.4 Validate

Een niet te vergeten onderdeel in de dataobject class is de validatie functie. Deze functie zal bij het toevoegen of editeren van gegevens opgeroepen kunnen worden om de ingegeven gegevens te valideren en indien nodig een foutmelding te geven. Met andere woorden, niets meer of minder dan een invoercontrole. Ik bespreek slechts een gedeelte uit deze functie omdat het principe voor ieder klasse attribuut hetzelfde zal zijn.

/libs/DB/Projects.php

```
public function validate()
{

  if (!Validate::string($this->name, array(
       'min_length' => 2,
       'max_length' => 120,
  ))) {
     Sanmax_Error::push('name', 'u heeft een ongeldige naam opgegeven');
  }

  if (strlen($this->hours_budget) && !Validate::number($this->hours_budget,
array(
       'min' => 0,
       'max' => 999,
  ))) {
     Sanmax_Error::push('hours_budget', 'ongeldige urenbudget opgegeven');
}

  return !Sanmax_Error::hasErrors();
}
```

De Validate-functie maakt gebruik van een Validate-methode uit het package PEAR. Via deze package kunnen data als numerieke (number) en alfanumerieke velden (strings), emails, datums, … gevalideerd worden. Deze methode wordt hier dan ook opgeroepen, er wordt een bepaalde syntaxis en opbouw gehanteerd.

Wanneer je een invoercontrole wilt toevoegen aan een niet verplicht veld is dit ook mogelijk. In de if-voorwaarde zal dat een extra controle gedaan worden op de lengte van het veld. Pas als de lengte groter is dan nul en de validatie niet zal kunnen plaatvinden zal er een error gepushed worden. Voor verplichte velden is de werkwijze identiek hetzelfde alleen zal enkel de Validate functie gecontroleerd worden.

Aan het einde van de de functie zal aangegeven worden of er errors gevonden zijn of niet. Afhankelijk hiervan zullen de eventueel gepushte errors getoond kunnen worden in de template. Het errorsysteem is een onderdeel van het Sanmax framework.

Hoe deze validate-functie opgeroepen wordt en hoe de errors op het scherm getoond zullen worden vindt u terug in onderdeel 5.3.6.2.

5.3.4.5 Logs

Alle acties naar de database zullen ook gelogd worden. Hiervoor worden in de dataobject klasse de standaard insert, update en delete methodes overschreven. We maken een functie aan met dezelfde naam als de parent functie. Door parent::insert() terug te geven verandert er uiteindelijk niets aan deze methode, buiten de ene regel voor het wegschrijven van de log. Het wegschrijven van de log is ook een specifiek onderdeel van het Sanmax framework.

/libs/DB/Projects.php

```
function insert()
{
      Sanmax_Logs::create($this, 'insert');
      return parent::insert();
}
```

Dit gebeurt identiek voor de update en delete methode. Als we even de class voor het creeren van de logfiles bekijken die we terugvinden in de library map Sanmax, zien we dat er gewoon een dataobject aangemaakt wordt waarbij de verschillende attributen geïnitaliseerd worden en uiteindelijk weggeschreven zullen worden naar de databank.

De $this in de functie hierboven, of de $log in de functie create bevat de eigenlijke beschrijving van wat er juist gebeurd is. Dit is een volledige string met alle nodige informatie, met onder andere de tabel waarop de actie gebeurd is, de volledige query die uitgevoerd is, enzovoorts.

/libs/Sanmax/Logs.php

```
class Sanmax_Logs
{
    function create($log, $action)
    {
        $Logs = new DB_Logs();

        $Logs->description = serialize($log);
        $Logs->category    = $action;
        $Logs->date        = DB_dataobject_Cast::sql('NOW()');
        $Logs->user_id     = $_SESSION['user_id'];

        $Logs->insert();
    }
}
```

5.3.5 View (Smarty Templates)

Het tweede onderdeel van het MVC-model is de view. De eigenlijke datapresentatie. De aangeleverde informatie zal getoond worden aan de gebruiker. In deze applicatie bestaat de user interface uit HTML pagina's, zo lijkt het toch op het eerste gezicht. Maar eigenlijk gaat het om templates. Templates die opgebouwd worden met behulp van smarty, een template systeem voor PHP.

Wat houdt smarty juist in ? Smarty zorgt er simpelweg voor dat u de PHP code uit de HTML code kunt laten. Er ontstaat zo enerzijds een pagina met PHP en anderzijds met HTML code. Beide bestanden kunnen afzonderlijk onderhouden worden. Bij grotere projecten kan dat zelfs door verschillende personen. De designer zal de lay-out en schikking verzorgen van de HTML bestanden. De ontwikkelaar bekommert zich om de programmeercode. Een aanpassing in de programmeercode behoeft dus niet telkens een aanpassing in template. De gegevens die toegewezen worden vanuit de programmeercode zullen met behulp van expressies getoond worden op de HTML pagina.

Door dat smarty hier geïmplementeerd zit in het framework als datapresentatie is het gebruik niet identiek hetzelfde dan wanneer men enkel smarty zal gebruiken in een webapplicatie, wat dus ook perfect mogelijk is. Op de officiële website noemt men smarty zelf ook een framework omdat het zich niet beperkt tot enkel tag-replacement (het vervangen van tags) maar er ook effectieve handelingen en bewerkingen kunnen gebeuren met de gegevens die men terugkrijgt van de PHP-code. Zo kan er op een snelle manier een tabel getoond worden met alle gegevens uit een array. Maar er zijn ook tal van template functies die er voor zorgen dat de template designer op een snelle manier HTML controls kan definiëren. Zo kan met een enkele expressie een volledige selectbox (dropdown) getoond en opgevuld worden.

Andere technische voordelen zijn de beveiliging. De templates zelf bevatten een PHP code en kunnen dus moeilijker onderschept worden. Daarnaast is dit template systeem ook extreem snel ondanks de vele functies die beschikbaar zijn. De meeste plug-ins worden 'on-demand' ingeladen, met andere woorden: pas als deze daadwerkelijk nodig zijn.

5.3.5.1 "Standaard" smarty

Toch gaan we even een standaard smarty applicatie bekijken, zonder de implementatie in het framework, om u smarty zo beter te laten begrijpen als alleenstaande template engine. Het gaat hier om een heel simplistisch voorbeeld, enkel om een impressie te geven.

We gaan eerst het PHP bestand opmaken. Allereerst is het natuurlijk wel nodig om de smarty classes te downloaden en in een directory te plaatsen op de host.

```
Require ('/smarty/Smarty.class.php');
$smarty = new Smarty();
```

Deze class zal dan opgeroepen worden in het PHP bestand. Waarna een instantie van smarty aangemaakt kan worden.

```
$smarty->template_dir = '/templates/';
$smarty->config_dir = '/configs/';
$smarty->compile_dir = '/templates_c/';
```

Voordat je een template kan gebruiken moet smarty weten waar hij de templates en de configuratiebestanden kan vinden. Smarty zal de templates ook compileren en heeft dus ook een directory nodig waar hij de gecompileerde versies kwijt kan. Deze directory moet beschrijfbaar zijn (dit kan door deze te chmod'en[1)] naar 775).

```
$smarty->assign('voornaam','Davy');
$smarty->assign('naam','Rego');
```

Met behulp van de assign methode gaan we variabele declareren die gebruikt kunnen en zullen worden in de template.

```
$smarty->display('ik.tpl');
```

De display methode zorgt op zijn beurt dat de template effectief getoond zal worden. De template zal gecompileerd worden wanneer dit nog niet gebeurd is. Afhankelijk van de instellingen is het ook mogelijk dat deze compilatie telkens zal gebeuren, wat enkele milliseconden extra tijd kost natuurlijk.

De template die hierboven getoond wordt, moet natuurlijk ook nog aangemaakt worden. In dit geval zal dat een bestand zijn met de extensie .tpl. Maar dit kan in principe ook een .html of .php extensie zijn.

```
<html>
  <body>
    Hallo, ik ben {$voornaam} {$naam}.
  </body>
</html>
```

Dit is maar een zeer klein voorbeeld, naarmate de applicatie groter en dus ook ingewikkelder gaat worden zal het nut duidelijker te voorschijn komen. Zo ook met mijn applicatie. Maar zoals nu al duidelijk is geen PHP geknoei meer in de templates met de echo's of quotes.

5.3.5.2 Smarty in het framework

Smarty in mijn webapplicatie wordt iets anders toegepast. Aangezien smarty slechts een onderdeel is in het MVC-model. De assign en display methode wordt maar op een enkele plaats opgeroepen en dat is in de application.php die al besproken werd aan het einde van onderdeel 5.3.3. Hieronder even de assign's en display uit die file ter opfrissing.

```
        $Smarty->assign('controller', $this->getController());
        $Smarty->assign('action',     $this->getAction());
        $Smarty->assign('template',   $this->getTemplate());
        $Smarty->display('Common/' . $_GET['tpl'] . '.html');
    }
```

[1)] Chmod: change modus, hiermee worden rechten van bestanden en mappen ingesteld. Voor drie groepen (owner, group, other) kunnen er read, write en execute rechten toegekend worden. Meer informatie en een online chmod-calculator vindt u op http://www.classical-webdesigns.co.uk/resources/whatchmod.html.

De variabele die dus toegekend worden zijn de controller, de action en de template. Afhankelijk van de template zal ook een bepaald HTML bestand opgevraagd worden. Dit bestand is een eerste template die we tegenkomen. Er zijn in het totaal drie verschillende common templates voorzien.

/templates/Common/template.html

```
{include file='Common/header.html'}

<table width="100%" cellpadding="5" cellspacing="0">
  <tr>
    {if $smarty.session.user_id}
    <td class="menu">
      <center><img src="/images/admin/logo.gif" alt="Sanmax" /></center>
      <br /><br />
            <div class="menubox">
{load_module controller="Menu" action="build" template="Common/menu.html"}
            </div>
      <br/>
    </td>
   {/if}
    <td class="template">
{load_module controller="$controller" action="$action" template="$template"}
    </td>
  </tr>
</table>

{include file='Common/footer.html'}
```

De eerste template is de standaard template die voor het merendeel van de pagina's in de applicatie zal opgeroepen worden. Naast de header en footer wordt ook hier een menu module geladen. De belangrijkste module die ingeladen wordt is diegene die de variabele $controller, $action en $template bevat. Zoals u ziet zijn dit de variabele die vanuit de application.php toegekend werden aan smarty.

/templates/Common/header.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
    <title>SANMAX - PIS (Project Information System)</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta http-equiv="pragma" content="no-cache" />
    <meta http-equiv="Cache-Control" content="no-cache" />
    <meta name="author" content="Sanmax bvba Genk - http://www.sanmax.be" />
</head>
<body>
```

De header bevat naast een DOCTYPE-declartie, die aangeeft aan welke HTML versie de pagina voldoet, ook de begin tags <html>, <head> en <body>. In de head sectie, die hier overigens ook afgesloten wordt onder andere de titel en een aantal meta-tags voor deze applicatie bepaald. Er zullen hier ook JavaScript en CSS bestanden opgeroepen kunnen worden. In de footer wordt de de body- en html-tag afgesloten.

/templates/Common/footer.html

```
</body>
</html>
```

/templates/Common/iframe.html

```
{include file='Common/header.html'}
{load_module controller="$controller" action="$action" template="$template"}
{include file='Common/footer.html'}
```

Een andere algemene template is de iframe. Het enige verschil met de vorige is dat hier geen menu getoond zal worden. Deze template zal bijvoorbeeld opgeroepen worden in popups. Een voorbeeld waar deze template gebruikt wordt vindt u terug in hoofdstuk 5.5.1.

/templates/Common/ajax.html

```
{load_module controller="$controller" action="$action" template="$template"}
```

Tot slot een AJAX template, specifiek voor de pagina's waarin AJAX-componenten gebruikt worden. Hier worden de header en de footer niet toegevoegd omdat deze hier niet nodig zijn. Het gaat namelijk niet om een html bestand zoals de tags in de header en footer wel zouden aangeven. Een voorbeeld van deze template kan u terugvinden onder 5.5.3.

Tot zover de globale templates. Elke pagina of zelfs paginaonderdeel zal ook een template moeten hebben. Zo is er bijvoorbeeld een template nodig voor het tonen van een overzicht met alle projecten. Een template met een formulier voor het toevoegen en aanpassen van projecten. Eventueel een template die de details van een project toont in een popup. Per onderdeel van de applicatie, of beter gezegd, per controller zal er een submap aangemaakt worden in de map templates. Omdat de uitleg van deze templates zeer nauw aansluit met wat er in de controller gebeurt zullen deze templates in detail in het onderdeel van de controller behandeld worden. Zo zal de samenhang duidelijker zijn.

5.3.5.3 Smarty functies en variable modifiers

Maar toch nog even over smarty zelf: om de gegevens, HTML componenten,… straks te tonen in de template heeft men in smarty heel wat handige functies voorzien. Ik overloop even de belangrijkste en meest door mij gebruikte functies in het volgende overzicht.

{foreach}	Deze expressie wordt gebruikt voor het doorlopen van enkelvoudige associatieve array. Waarbij de elementen van de array dan getoond worden. Zo kan bijvoorbeeld een array met projecten doorlopen worden. `{foreach from=$projects item=$project}` `  Project: {$project}  ` `{foreachelse}` `  Geen projecten gevonden !` `{/foreach}`

{if}	De voorwaarde controle beter gekend als de selectie of het if-then-else statement. De syntax van de vergelijking is in overeenstemming met die van PHP, heel wat mogelijkheden met andere woorden.
	`{if $name == 'Davy'}`
	`  U bent Davy, juist he ?`
	`{elseif}`
	`  U bent Davy niet, u bent {$name} !`
	`{/if}`

{include}	Gebruikt voor het toevoegen van andere templates in de huidige template. Zoals bijvoorbeeld gebruikt om de header en footer toe te voegen in de algemene template.html of om bijvoorbeeld een zoekbar aan een overzichtspagina toe te voegen. `{include file='Common/header.html'}`
{html_options}	Deze functie genereert de html <select><option> tags, met andere woorden de gekende selectbox. Deze zal opgevuld worden met waarden die men uit een array haalt. Zo is het bijvoorbeeld mogelijk om een selectbox op te vullen met de klantnamen (op voorwaarde dat deze in een array zitten). De options eigenschap is de array, selected zal een specifieke option selecteren. `{html_options name=customer_id options=$customers selected=$smarty.post.customer_id}`

{html_select_date}
{html_select_time}
	Het generen van selectboxen voor het selecteren van de datum of de tijd. Er kunnen hier heel wat extra opties toegevoegd worden. Zo kan er een beperking gedaan worden van het aantal te selecteren jaren. Of kan men er voor kiezen om de tijd in een 12 of 24 uren weergave te tonen, met of zonder de seconden, met een interval van 5 minuten enzovoorts…
	`{html_select_date end_year='+5'}`
	`{html_select_time use_24_hours=true display_seconds=false minute_interval=5}`

{debug}	Deze expressie zal er voor zorgen dat er een popup verschijnt bij het laden van de pagina die alle toegewezen variabele aan de opgeroepen template zal tonen. Wanneer er bepaalde gegevens niet getoond worden kan je zo controleren of de gegevens fout doorgegeven worden of dat er gewoon een syntax fout in de templates is gekropen. `{debug}`

Naast deze functies zijn er ook een “variable modifiers”. Deze zullen vooral dienen om het formaat van de variabele aan te passen. Deze modifiers zullen achter de variabele tussen de accolades geplakt worden. Enkele voorbeelden:

|capitalize	Alle woorden in de variabele met een hoofdletter beginnen.
|date_format	Voor het formatteren van datum en tijd. Er wordt hier gewerkt met een specifiek syntaxis voor het formatteren van de datum en tijd. Bijvoorbeeld: `%m/%d/%y` = 01/12/05
|truncate	Voor het afkappen van variabele op een bepaalde lengte. `{$project.name|truncate:30:"..."}`
|default	Voor het geven van een standaardwaarde aan de variabele moest deze zelf geen waarde bevatten. `{$customer.phone|default:'Geen telefoonnummer'}`

Een overzicht van alle functies en variable modifiers vindt u terug in de manual van Smarty op hun website[1)].

[1)] http://smarty.php.net/manual/en/

5.3.6 Controllers

Tot slot de controller of applicatielogica. Het is de controller die zal reageren op events. De controller ontvangt en vertaalt input naar aanvragen voor de model of voor de view. De controller zal gegevens ophalen, eventueel verwerken en daarna bezorgen aan de view om daar getoond te worden. Per onderdeel van de applicatie zal er een controller voorzien worden. Ik zie bijvoorbeeld projectbeheer als een onderdeel. We voorzien dan ook een controller 'Projects.php'. Voor de belangrijkste tabellen in de database zal je normaliter een controller aanmaken omdat al deze gegevens beheerd moeten worden, maar dat is zeker geen vaste regel. Zo is er in mijn applicatie ook een controller voorzien voor de startpagina ('Home.php') terwijl er daarvan helemaal niets terug te vinden is in de database.

Wat wel een regel is, is dat er voor elke controller een map met dezelfde naam aanwezig is onder de directory /templates. In deze specifieke mappen zullen dan de templates terechtkomen waaraan de controller zijn gegevens zal teruggeven.

De structuur van de controller zal dus gedeeltelijk of soms zelfs volledig terug te vinden zijn in de structuur van de templates. We nemen de proef op de som. We plaatsen links de inhoud van de controller class 'Projects' en rechts de verschillende templates:

/Controllers/Projects.php	/Templates/Projects
class Projects {	
public function add()	add.html (include form.html)
public function edit()	edit.html (include form.html)
public function delete()	*behoeft geen template*
public function overview()	overview.html (include searchbar.html en vcard.html)
public function detail()	detail.html
}	

Wat deze functies juist doen en hoe de samenhang gerealiseerd wordt met de templates wordt duidelijk in onderdeel 5.3.6.2. Eerst bespreek ik nog een aantal specifieke functies en regels voor de controller zelf.

Allereerst is er in de meeste controllers een initialisatie functie voorzien. In deze functie worden heel wat gegevens opgehaald uit (data)models voor het vullen van bijvoorbeeld selectboxen. Deze gegevens moeten bijvoorbeeld opgehaald worden in het onderdeel add, edit en overview voor het vullen van een selectbox in de zoekbalk. Dat zou betekenen dat drie keer dezelfde code gedeclareerd zou moeten worden. Daarom werken we met een initialisatie functie. Deze functie is vergelijkbaar met de __construct, alleen dat deze laatste bij elke oproep van de class uitgevoerd zal worden en de init-functie slechts opgeroepen zal worden wanneer dit expliciet vermeld staat.

We noemen deze functie 'init()'. Deze functie zal binnen een andere functie opgeroepen kunnen worden met behulp van onderstaande code. $this verwijst naar de eigen classe, aangezien de functie in dezelfde klasse gedefinieerd is.

```
$this->init();
```

In het voorbeeld Projects.php ziet de init() functie er als volgt uit:

/Controllers/Projects.php

```
protected $_status            = array();
protected $_priorities        = array();
protected $_customersAll      = array();
protected $_customersOnly     = array();
protected $_admins            = array();
protected $_hours             = array();

public function init()
{
    require_once 'DB/FillModel.php';

    $Status = DB_dataobject::Factory('status');
    $this->_status = $Status->getSelect();

    $Priorities = DB_dataobject::Factory('priorities');
    $this->_priorities = $Priorities->getSelect();

    $Customers = DB_dataobject::Factory('customers');
    $this->_customersAll = $Customers->getSelect();

    $Customers = DB_dataobject::Factory('customers');
    $this->_customersOnly = $Customers->getSelect(DB_Customers::QUERY_ONLYACTIVE);

    $Users = DB_dataobject::Factory('users');
    $this->_admins = $Users->getSelectAdmin();

    $this->_hours = FillModel::fillHoursBudget();
}
```

Bovenaan wordt er binnen de class een protected array aangemaakt. Het zijn deze array's die gevuld zullen worden met de opgehaalde gegevens. Zoals u merkt worden de meeste gegevens uit de database gehaald. Er wordt dan ook gewerkt met een dataobject en een methode gedefinieerd in die specifieke dataobject class. Bij de laatste regel is dit echter anders. Er wordt hier ook beroep gedaan op een model, maar deze haalt geen gegevens op uit de databank. Dit toont aan dat een model niet steeds een dataobject class moet zijn. Het bestand 'FillModel.php' bevat een aantal functies, zoals fillHoursBudget(), die statische gegevens zal teruggeven. Hiervoor moet wel dit bestand toegevoegd worden, zoals u bovenaan de functie ziet.

/libs/DB/FillModel.php

```
public static function fillHoursBudget()
{
      $hours = array();
      $hours[null]                  = " ";
      $hours["between 0 and 20"]    = "Klein (<20h)";
      $hours["between 20 and 100"]  = "Middel (<100h)";
      $hours["> 100"]               = "Groot (>100h)";
      return $hours;
}
```

Let op, het gaat hier om een statische functie. Dat ziet u ook aan de manier waarop deze functie opgeroepen wordt in de controller.

5.3.6.1 Controller ft. template.

Overview

Wanneer we nu een specifieke actie onder de loop zullen nemen zal u al snel merken dat de samenhang met de template duidelijk te zien zal zijn. De gegevens die de controller ophaald zullen teruggegeven worden aan de template, waar ze dan op hun beurt getoond zulen worden. Het duidelijkste voorbeeld hiervan is het tonen van een overzichtspagina. Als voorbeeld ook hier weer de overzichtspagina van de projecten. Om nog even de link te leggen met de applicaton.php en de .htaccess file. Onderstaande functie zal opgeroepen worden wanneer er gesurft wordt naar /projects/overview, waarbij $controller = 'projects' en $action = 'overview'.

/Controllers/Projects.php

```
    public function overview()
    {
        $this->init();
        $Projects = DB_dataobject::Factory('projects');
```

Allereerst wordt er in deze functie gebruik gemaakt van de initialisatie functie. Alle arrays zullen opgevuld worden met de juiste data en de class variabelen zullen gebruikt kunnen worden. Als tweede zal er een dataobject aangemaakt worden voor de tabel 'projects'. Op dit dataobject zal afhankelijk van de request methode een bepaalde methode opgeroepen worden.

```
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $projects = $Projects->search($_POST);
        }
```

Wanneer men de pagina benadert zullen er in eerste instantie geen projecten zichtbaar zijn. Dit om lange laadtijd te vermijden. Men zal via de zoekbalk kunnen zoeken naar bepaalde projecten of gewoon op de zoekknop klikken om een volledig overzicht te krijgen van alle projecten. Er gebeurt op dat moment een POST en dus is de request_mode='POST'.

```
        if ($_SERVER['REQUEST_METHOD'] === 'GET') {
            $id = (int) $_GET['id'];
            $projects = $Projects->getById($id);
        }
```

Na het wijzigen van een project zal er echter terug doorverwezen worden naar de overzichtspagina. Als parameter zal er het id meegegeven worden van het gewijzigde project. Deze wordt uit de URL opgehaald via $_GET[]. Op dat moment zal op de overzichtspagina enkel het gewijzigde project getoond worden.

```
        return array('projects' => $projects,
                     'prioritiesopties' => $this->_priorities,
                     'customersopties' => $this->_customersAll,
                     'hoursopties' => $this->_hours,
                     'usersopties' => $this->_admins,
                     'statusopties' => $this->_status);
        }
```

Uiteindelijk zullen alle opgevraagde gegevens in de vorm van een array teruggegeven worden. Ook de gegevens die via de init() functie opgehaald worden. Deze zullen in dit geval gebruikt worden voor het opvullen van de selectboxen in de zoekbalk.

De gegevens worden dus “gereturned” aan de template. De template zal deze arrays nu verwerken en tonen op een specifieke manier die bepaald wordt door de designer van de template-paginas.

/Templates/Projects/overview.html

```
<div id="header">
	<h3>
		{html_image file='/images/icons/overview.gif'}
		Projecten overzicht
	</h3>
</div>

{include file='Messages/Error.html'}
{include file='Messages/Result.html'}

<form name="frm_project_search" method="post" action="/projects/overview">
	{include file="Projects/searchbar.html"}
</form>
<br />

{if $projects}
	{foreach from=$projects key=id item=project}
		{include file="Projects/vcard.html" showactions=true}
	{/foreach}
{else}
	<center>
		{html_image file='/images/icons/exclamation.gif'}
		Specifieer uw zoekopdracht of klik op onmiddelijk 'zoeken' om alle
projecten te tonen.<br/>
		Er zijn geen projecten teruggevonden in de databank.
	</center>
{/if}
```

Op de overzichtspagina wordt als eerste de zoekbalk geinclude tussen formulier-tags. De methode die gebruikt wordt is een POST en de actie is /projects/overview. Wanneer er dus op de zoekknop, die getoond wordt in de volgende file (searchbar.html) geklikt zal worden zal de code uitgevoerd worden van de methode ‘overview’ in de controller ‘projects’. Bij de if-controle op de request_method zal diegene met als resultaat = ‘POST’ uitgevoerd worden.

Verder zal er, wanneer er projecten zijn, voor ieder project dat gevonden wordt een vcard.html geinclude worden. Hiervoor wordt de array ‘$projects’ doorlopen die we via de controller teruggegeven hebben. Eén project uit die array wordt een item genoemd, we geven hier de naam ‘project’. Wanneer we in de vcard de gegevens van dat project willen tonen zullen we de attributen van dat item oproepen door ze de prefix ‘project.’ te geven. Dit zal bij de bespreking van het vcard.html bestand duidelijker worden.

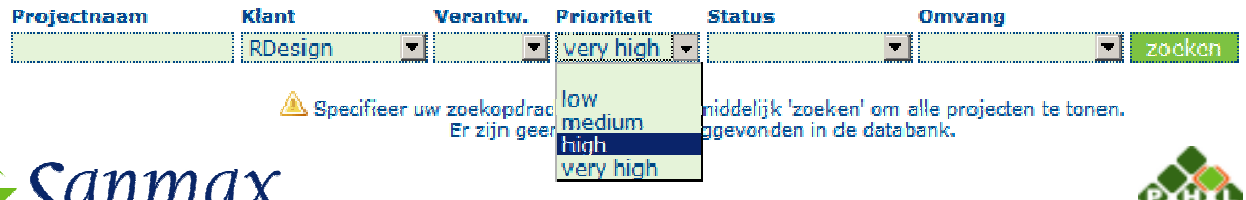
Wanneer er geen projecten gevonden worden ({else}) zal er hiervan een melding gemaakt worden. Wanneer de pagina voor de eerste keer opgeroepen wordt zal deze melding ook verschijnen, aangezien er dan nog geen projecten opgehaald zijn. De gebruiker zal op de knop ‘zoeken’ moeten klikken om een volledig overzicht te krijgen van alle projecten of hij kan zijn zoekopdracht specificeren door gebruik te maken van de verschillende velden op de zoekbalk.

/Templates/Projects/searchbar.html

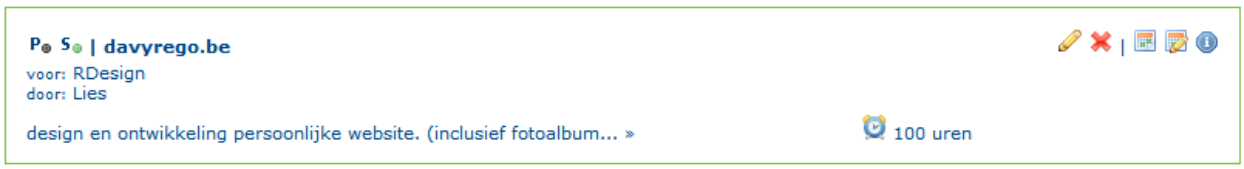
```
<table>
  <tr>
    <th>Projectnaam</th>
    <th>Klant</th>
    <th>Verantw.</th>
    <th>Prioriteit</th>
    <th>Status</th>
    <th>Omvang</th>
    <th> </th>
  </tr>
  <tr>
    <td>
      <input type="text" name="name" class="control"
value="{$smarty.post.name}" />
    </td>
    <td>
      {html_options name=customer_id options=$customersopties
selected=$smarty.post.customer_id}
    </td>
    <td>
      {html_options name=user_id options=$usersopties
selected=$smarty.post.user_id}
    </td>
    <td>
      {html_options name=priority_id options=$prioritiesopties
selected=$smarty.post.priority_id}
    </td>
    <td>
      {html_options name=status_id options=$statusopties
selected=$smarty.post.status_id}
    </td>
    <td>
      {html_options name=hours_budget options=$hoursopties
selected=$smarty.post.hours_budget}
    </td>
    <td><input type="submit" name="search" class="button" value="zoeken"
/></td>
  </tr>
</table>
```

In de zoekbalk wordt een tekstveld voorzien waarin men de naam of een gedeelte ervan kan invullen waar men kan op zoeken. Verder zijn er heel wat selectboxen waaruit men bepaalde element kan selecteren. Deze selectboxen worden opgevuld met de gegevens uit de arrays die we ook teruggaven aan deze template. De naam die de selectbox krijgt, komt overeen met het veld waarop gezocht zal worden in de search-functie (zie voorbeeld 5.3.4.4).

De $smarty.post.variabele_naam haalt de waarde op van de variabele_naam die het resultaat is van een POST van een formulier. (in php: $_POST['variabele_naam'];). Deze waarde kan getoond worden of kan gebruikt worden om in een selectbox een bepaald element te laten selecteren. Zodat wanneer men effectief op de zoekknop klikt het geselecteerde element ook na het herladen van de pagina geselecteerd zal blijven.



Voor elk gevonden project zullen er ook details getoond worden. Dit gebeurt door voor elk project een vcard te includen. De naam vcard komt van de wel gekende vCards die meegestuurd kunnen worden met een e-mail of die je terugvindt in uw adresboek van uw e-mailcliënt. Ook daar wordt er op een 'kaartje' heel wat relevante informatie getoond. En dat is hier niet anders, weliswaar geen personen- maar projectgegevens. Wat er allemaal getoond wordt en op welke manier wordt overlopen aan de hand van de html-code.



Een belangrijke opmerking: let op de variabele die gebruikt wordt. We gebruiken hier $project. De naam die meegegeven werd als itemeigenschap in de foreach in overview.html.

/Templates/Projects/vcard.html

```
<table class="vcard_project">
  <tr>
    <td width="70%">
      {if $project.priority_style eq 'BLUE'}
        {html_image file='/images/icons/prior_blue.gif'}
      {elseif $project.priority_style eq "RED"}
        {html_image file='/images/icons/prior_red.gif'}
      {elseif $project.priority_style eq "ORANGE"}
        {html_image file='/images/icons/prior_orange.gif'}
      {else}
        {html_image file='/images/icons/prior_gray.gif'}
      {/if}
      {if $project.status_style eq 'GREEN'}
        {html_image file='/images/icons/status_green.gif'}
      {elseif $project.status_style eq "RED"}
        {html_image file='/images/icons/status_red.gif'}
      {elseif $project.status_style eq "ORANGE"}
        {html_image file='/images/icons/status_orange.gif'}
      {else}
        {html_image file='/images/icons/status_gray.gif'}
      {/if}
      <strong>| {$project.name|stripslashes}</strong>
    </td>
```

De prioriteit en de status van een project wordt getoond door middel van een icoon. Het getoonde icoontje is afhankelijk van de kleur die meegegeven werd als status_style. Als u nu terug zal kijken naar de model in 5.3.4.3. zal u zien dat alle gegevens of attributen van een project ook daadwerkelijk opgehaald moesten worden en meegeschreven zijn in de array. Deze array werd teruggekregen bij het oproepen van de methode in de model, de controller gaf deze op zijn beurt terug aan de template.

```
    <td align="right" width="30%">
      {if $showactions}
```

De controle die hier gebeurt, gebeurt op een variabele die meegegeven werd in de include expressie op de overview.html. In de include expressie is showactions=true vermeldt. Hierdoor zal de code binnen de if expressie dus getoond worden. Indien deze parameter op false zou staan, werden de iconen met eventueel bijhorende link niet getoond.

```
        {if isset($smarty.session.permissions.Projects)
            and in_array('edit', $smarty.session.permissions.Projects)}
          <a href="/projects/edit/?id={$project.id}">
              {html_image file='/images/icons/pencil.gif'}
          </a>
        {/if}
```

Er zullen een aantal icoontjes getoond worden waaraan een actie gekoppeld is. Bovenstaande icoon zal bijvoorbeeld doorverwijzen naar de edit-pagina voor dat bepaalde project. Het id van het project wordt meegegeven zodat met op de edit-pagina weet welke gegevens ingeladen en aangepast moeten kunnen worden. De action “edit” wordt verderop in dit hoofdstuk nog besproken. Het tonen van die iconen bevindt zich in een if-structuur. De controle die hier gebeurd is of de gebruiker wel voldoende rechten bezit om deze actie te mogen uitvoeren. Indien hij niet voldoende rechten heeft, heeft het ook geen zin om dit icoon te tonen, en zal het dus verborgen worden. Het hele rechtensysteem wordt besproken onder hoofdstuk 5.4.

```
        {if isset($smarty.session.permissions.Projects)
            and in_array('delete', $smarty.session.permissions.Projects)}
            <a href="/projects/delete/?id={$project.id}">
              <img src="/images/icons/pencil_delete.gif"
              onclick="return confirm('Ben je zeker dat je project
{$project.company} wilt verwijderen?')" />
            </a>
        {/if}
```

Bijna identiek gaat het er aan toe bij het verwijdericoon. Alleen zal hier niet doorverwezen worden naar een nieuwe pagina. Wanneer er geklikt wordt op het icoon zal er een confirm-window tonen waarbij de gebruiker gevraagd wordt of hij zeker is om het item te verwijderen. Indien hij op ja klikt, zal de link (/projects/delete/?id={$project.id}) gevolgd worden, in het andere geval zal er niets gebeuren. Dankzij deze werkwijze is het niet nodig om een pagina aan te maken waarop je dan de gebruiker moet laten bevestigen of de delete mag doorgaan. Het verwijderen wordt dus enkel afgehandeld door de controller. Ook dit wordt duidelijk op de volgende pagina’s.

```
        {if isset($smarty.session.permissions.Tasks)
            and in_array('overview', $smarty.session.permissions.Tasks)}
              |
            <a href="/tasks/overview/?id={$project.id}">
              {html_image file='/images/icons/task.gif'}
            </a>
        {/if}
        {if isset($smarty.session.permissions.Tasks)
            and in_array('add', $smarty.session.permissions.Tasks)}
            {if $project.status_id <> 5 && $project.status_id <> 4}
               <a href="/tasks/add/?id={$project.id}">
                 {html_image file='/images/icons/task_add.gif'}
               </a>
            {/if}
```

```
        {/if}
      {/if}
      {if isset($smarty.session.permissions.Projects)
         and in_array('detail', $smarty.session.permissions.Projects)}
        <a href="/iframe/projects/detail/?id={$project.id}"
         class="lWOn" title="project {$project.name|stripslashes}">
          {html_image file='/images/icons/information.gif'}
        </a>
      {/if}
    </td>
  </tr>
```

Ook bij de knoppen voor het takenoverzicht per project, het aanmaken van een nieuwe taak voor een project en het opvragen van de projectdetails in een modal popup (zie 5.5.1) worden gelijkaardig getoond.

```
  <tr>
    <td valign="top">
      <small>voor:</small>
      {if isset($smarty.session.permissions.Customers)
         and in_array('detail', $smarty.session.permissions.Customers)}
        <a href="/iframe/customers/detail/?id={$project.cust_id}"
         class="lWOn" title="klant {$project.company|stripslashes}">
           {$project.company|stripslashes}
        </a>
      {else}
        {$project.company|stripslashes}
      {/if}
      <br />
      <small>door:</small> {$project.user_login|stripslashes}
      <br /><br />
      {$project.description|stripslashes|truncate:68:"...":true}
      {if $project.description|count_characters > 68}
        <a href="#" onmouseover="this.T_TITLE='Omschrijving:
{$project.name|stripslashes}';
           return escape('{$project.description|stripslashes}')">
        &raquo;</a>
      {/if}
    </td>
    <td valign="bottom">
      {if $project.hours_budget}
        {html_image file='/images/icons/clock.gif'}
        {$project.hours_budget} uren<br />
      {/if}
    </td>
  </tr>
</table>
```

Voorts worden alle gegevens getoond. Bepaalde variabele worden aangepast van formaat met behulp van de “variable modifiers”. Wanneer de beschrijving van een project meer dan 68 karakters telt zal deze afgebroken worden omdat de structuur van de pagina anders door elkaar gehaald zal worden. Er wordt dan een extra link geplaatst. Wanneer men hier met de muisaanwijzer over zal gaan, krijgt men de volledige omschrijving te zien (zie 5.5.2).

Edit

In de applicatie zullen niet alleen gegevens getoond worden maar moeten en ook kunnen toegevoegd en aangepast worden. Daar waar de add-action bijna gelijkaardig is met de edit-action nemen we deze laatste even onder de loep.

/Templates/Projects/vcard.html

```
public function edit()
{
      $this->init();
```

Net als op de overzichtspagina wordt ook hier de initialisatiefunctie opgeroepen. Deze is nodig voor het opvullen van een aantal selectboxen.

```
      if ($_SERVER['REQUEST_METHOD'] === 'GET') {
          $id = (int) $_GET['id'];
          $Project = DB_DataObject::Factory('projects');
          $Project->get($id);
          $project = $Project->toArray();

          return array('project' => $project,
            'statusopties' => $this->_status,
            'prioritiesopties' => $this->_priorities,
            'customersopties' => $this->_customersAll,
            'usersopties' => $this->_admins);
      }
```

Wanneer de editeerpagina benaderd wordt zal er een een id meegegeven worden van het project waar men een aanpassing wil aan doen. De gegevens van dit project zullen opgehaald worden. Via de methode get() die opgeroepen wordt vanuit het dataobject. Het resultaat wordt in een array geplaatst en deze array zal teruggegeven worden. Samen met de nodige andere arrays voor het opvullen van de checkboxen.

```
      if ($_SERVER['REQUEST_METHOD'] === 'POST') {
          $id = (int) $_POST['id'];
          $Project = DB_DataObject::Factory('projects');
          $Project->id = $id;
          $Project->setFrom($_POST);
```

Wanneer er een POST gebeurt op editeerpagina zal er ook een id aanwezig zijn. Deze keer niet in de URL, maar als waarde op het formulier zelf (hidden). Deze id wordt opgehaald als formuliervariabele en een dataobject wordt aangemaakt met dat id. Alle andere formuliervariabele worden ook in het object geplaatst. Vandaar dat het zo belangrijk is dat de veldnamen van het formulier overeenkomen met de attributen van de dataobject class.

```
          if ($Project->status_id==4 OR $Project->status_id==5){
            $Project->date_end = date("Y-m-d H:i:s");
```

Wanneer er voor een project de status “done” (4) of “invoice” (5) geselecteerd is bij de status zullen er een aantal extra handelingen moeten gebeuren. Allereerst zal de einddatum van het project bepaald worden op het moment dat de status gewijzigd wordt. Via de php-functie data halen we de huidige datum op in het juiste formaat.

```
$Tasks = DB_DataObject::Factory('tasks');
$tasks = $Tasks->getTasksForProject($id, DB_Tasks::QUERY_ONLYACTIVE);
if (count($tasks) > 0){
      Sanmax_Error::push('projects', 'Project kan de status
            done of invoice niet aannemen. Er zijn nog actieve
            taken. U kan deze nu aanpassen:');
      Sanmax_Error::toSession();
      header("Location: /Tasks/overview/?id=$id");
      exit(0);
}
```

Verder zal er ook gecontroleerd worden of er nog actieve taken aanwezig zijn voor het project dat men op done of invoice wil zetten. Wanneer er nog actieve taken zijn, zal deze wijziging niet doorgevoerd worden en zal men hiervan melding krijgen. Men zal doorverwezen worden naar de overzichtspagina van de taken en zal enkel de taken van het te editeren project te zien krijgen (vandaar dat het id wordt meegegeven in de URL). Men kan deze taken dan eerst afronden en zal dan pas in staat zijn om het project effectief af te sluiten.

```
} else {
  $Project->date_end = null;
}
```

In het andere geval krijgt de einddatum de waarde null. Wanneer een attribuut van een dataobject de waarde null toegewezen krijgt, wil dit zeggen dat dit element niet opgenomen zal worden in de update (of insert) query. U moet hier wel mee opletten, het gaat hier om een PHP null waarde, en niet om een null waarde in MySQL. Stel dat men een afgesloten taak terug opent, dan zal de einddatum die aangegeven stond nog steeds in de databank staan aangezien deze niet overschreven zal worden. Daarom dat we zodadelijk een extra update zullen doen.

```
if (!$Project->validate()) {
  Sanmax_Error::toSession();
  return array('project' => $_POST,
        'statusopties' => $this->_status,
        'prioritiesopties' => $this->_priorities,
        'customersopties' => $this->_customersAll,
        'usersopties' => $this->_admins);
}
```

Maar vooraleer we de update gaan doorvoeren zullen we eerste de gegevens valideren. Dit gebeurt door de validatiemethode op te roepen die gespecificeerd is in de dataobject class. Deze methode geeft een true terug wanneer er geen errors zijn, en een false wanneer er wel errors zijn, vandaar dat ook hier weer het tegengestelde zal gecontroleerd worden in de if. Wanneer een invoercontrole faalt, zal de error getoond worden, en worden de arrays teruggegeven aan de template. De gegevens moeten immers blijven staan, het is niet de bedoeling dat men alles terug moet ingeven. Voor de projectinformatie wordt de volledige $_POST teruggegeven. Dit is immers ook een array die de formuliervariabelen bevat.

```
$Project->query("UPDATE {$Project->__table} SET date_end=null WHERE
id = {$Project->id}");
$Project->update();
```

Allereerst zal het einddatumveld een MySQL null waarde toegekend krijgen. Moest er geen einddatum opgegeven zijn, is dit veld in ieder geval leeggemaakt. Daarna vindt de standaard update methode plaats.

```
            $_SESSION['result'] = 'Project werd aangepast!';
            header("Location: /Projects/overview/?id=$id");
            exit(0);
      }
}
```

Tot slot zal er in sessie variabele meegegeven worden dat het project aangepast is. De gebruiker zal doorverwezen worden naar de overzichtspagina en zal het aangepaste project te zien krijgen. De exit(0) zorgt ervoor dat de huidige pagina ook daadwerkelijk verlaten wordt.

Zoals ik aan het begin van dit onderdeel al vermeldde zijn de add- en edit-action bijna gelijkaardig. Dat is ook te merken aan de opbouw van de templates. Voor beide is er een aparte template voorzien, maar ze includen alle twee de form.html.

/Templates/Projects/add.html

```
<div>
      {include file='Messages/Error.html'}
      {include file='Messages/Result.html'}
      <form name="project_add" method="post" action="/Projects/add">
            {include file='Projects/form.html'}
      </form>
</div>
```

/Templates/Projects/edit.html

```
{include file='Messages/Error.html'}
{include file='Messages/Result.html'}
<form name="project_edit" method="post" action="/Projects/edit">
      {include file='Projects/form.html'}
      <input type="hidden" name="id" value="{$project.id}" />
</form>
```

Het enige verschil tussen beide is action die verschilt en het verborgen veld bij de editpagina waarin het project id zal bijgehouden worden. Bij de add pagina is dit niet nodig aangezien de id van een project een primary key met auto increment is. Op de volgende pagina bekijken we de bewust form.html.

Let ook op de includes van de Message/Error.html en Message/Result.html. Dit zijn de pagina's waarnaar eventuele errors of session results getoond zullen worden. De result.html ziet er als volgt uit. Via een smarty expressie wordt gecontroleerd of er een sessie-variabele id bestaat. Is dat het geval wordt de div-tag met het resultaat getoond. Uiteindelijk wordt de sessie variabele terug leeggemaakt. De error afhandeling werkt volgens hetzelfde principe.

/Templates/Message/result.html

```
{strip}
{if $smarty.session.result}
      <div id="msg_info">
            {html_image file='/images/icons/information.gif'} 
            {$smarty.session.result}
      </div>
      {clear_result}
{/if}
{/strip}
```

/Templates/Projects/form.html

```
<table width="100%">
  <tr>
    <td>Klant *</td>
    <td>
      {if $customer.id}
        <input type="hidden" name="customer_id" value="{$customer.id}" />
        <strong>{$customer.company}</strong>
      {else}
        {html_options name=customer_id options=$customersopties
            selected=$project.customer_id}
      {/if}
    </td>
  </tr>
  <tr>
    <td>Naam *</td>
    <td>
       <input type="text" size="72" name="name" value="{$project.name|stripslashes}"/>
    </td>
  </tr>
  <tr>
    <td valign="top">Omschrijving *</td>
    <td>
      <textarea onkeydown="Telkarakters(this.form.description,this.form.count,500);"
      onkeyup="telkarakters(this.form.description,this.form.count,500);"
      cols="70" rows="4" name="description">
         {$project.description|stripslashes}
      </textarea>
      <input type="text" size="2" name="count" class="control_plain"
          readonly="readonly" value=""/>
    </td>
  </tr>
  <tr>
    <td>Verantwoordelijke *</td>
    <td>{html_options name=user_id options=$usersopties
          selected=$project.user_id}</td>
  </tr>
  <tr>
    <td>Voorziene uren</td>
    <td><input type="text" size="5" name="hours_budget"
          value="{$project.hours_budget}"/> </td>
  </tr>
  <tr>
    <td>Prioriteit *</td>
    <td>{html_options name=priority_id options=$prioritiesopties
          selected=$project.priority_id}</td>
  </tr>
  <tr>
    <td>Status *</td>
    <td>
      {html_options name=status_id options=$statusopties selected=$project.status_id }
      {if $action=="edit" && $project.date_end}
        sinds {$project.date_end|date_format:'%d-%m-%Y'}
      {/if}
    </td>
  </tr>
  <tr>
    <td> </td>
    <td>
      <input type="submit" name="submit" class="button" value="Opslaan"/>
      <input type="reset" name="cancel" class="button" value="Annuleren"
          onclick="location.href = '/projects/overview/';"/>
    </td>
  </tr>
</table>
```

In dit formulier wordt dezelfde werkwijze toegepast als bij het tonen van een overzicht of als bij de zoekbalk. Alleen dat de waarde nu in een tekstveld geplaatst zullen worden. Nogmaals vermelden, want het is oh zo belangrijk, de namen van de tekstvelden moeten dezelfde naam hebben als de naam van de attributen. Dit om er voor te zorgen dat de controller formuliervariabelen kan omzetten naar een dataobject dat alle nodige gegevens bevat. En natuurlijk dat bij het editeren de huidige gegevens juist en volledig getoond worden op de juiste plaats in het formulier.

Delete

Net zoals toevoegen (add) en wijzigen (edit) een actie zijn, geldt dat ook voor het verwijderen (delete). Echter is er hier geen template voorzien, onder andere omdat er gewerkt wordt met het confirm-window die de bevestiging voor zijn rekening neemt.

Wanneer men dus akkoord gaat met het verwijderen van het project zal er worden doorverwezen naar de URL /projects/delete/?id={$project.id}. Ook hier wordt het id van het project meegegeven zodat men weet over welk project het gaat.

/Controllers/Projects.php

```
public function delete()
{
  if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $id = (int) $_GET['id'];
    $Project->whereAdd("id = $id");
    $deleted = $Project->delete(DB_DATAOBJECT_WHEREADD_ONLY);
```

Het id wordt uit de URL gehaald en zal gebruikt worden in de where-conditie van het statement dat men aanmaakt. Uiteindelijk zal het project verwijderd worden uit de databank. De parameter DB_DATAOBJECT_WHEREADD_ONLY zorgt ervoor dat er daadwerkelijk rekening gehouden wordt met de where-conditie.

```
    if ($deleted > 0){
      $_SESSION['result'] = 'Project werd verwijderd uit de database!';
    } else {
      Sanmax_Error::push('projects', 'Project werd NIET verwijderd uit de
      database! Er zijn nog onderliggende taken gekoppeld aan dit project.');
      Sanmax_Error::toSession();
    }
    header('Location: /Projects/overview');
    exit(0);
  }
```

Is dit project nu ook daadwerkelijk verwijderd ? Dit zal zeker niet altijd zo zijn. Dit wordt gecontroleerd door het aantal verwijderde rijen dat de delete methode teruggeeft.

Bij het aanmaken zijn er namelijk een aantal constraints (on delete no action) toegevoegd aan refererende sleutels bij verschillende tabellen zodat een parent-row niet verwijderd kan worden wanneer er aan die rij child-rows gekoppeld zijn. Er wordt in dat geval een gepaste melding gegeven. Uiteindelijk zal de gebruiker doorverwezen worden naar de overzichtspagina.

5.4 Login & Rechtensysteem

5.4.1 Inleiding

Aangezien de webapplicatie gegevens bevat die bepalend zijn voor een gebruiker of in dit geval medewerker van Sanmax, is het logisch dat deze gebruiker geïdentificeerd moet kunnen worden. De gebruiker laten inloggen is daarvoor een zeer goede methode. Niet alleen is de gebruiker dan gekend bij het systeem, ook is het systeem beveiligd voor ongewenst bezoek.

Voor het inloggen wordt gebruik gemaakt van een sessie. Een sessie is een cookie die actief blijft zolang de browser actief is. Wanneer je de browser afsluit, zal de session cookie dus vervallen en verwijderd worden. In een cookie wordt informatie weggeschreven die lokaal bij de gebruiker bewaard blijft, in dit geval gedurende de sessie. In deze cookie zullen gegevens weggeschreven worden zoals de login van een gebruiker en andere configuratie-instellingen zodat alle rechten van die gebruiker bewaard worden.

Ook in mijn applicatie heb ik zo een sessie cookie voorzien. Vooraleer een dergelijke sessie cookie aangemaakt wordt, moet natuurlijk ook gecontroleerd worden of de gebruiker wel mag inloggen. Dat gebeurt door de logingegevens die hij ingeeft te vergelijken met de gegevens aanwezig in de databank, en meer bepaald in de tabel ‘users’.

Naast het feit dat een gebruiker kan inloggen is er ook een rechtensysteem voorzien. Hiermee is het mogelijk om bepaalde gebruikers, of om nauwkeuriger te zijn, bepaalde groepen gebruikers acties al dan niet te laten kunnen ondernemen. In feite is het logisch dat een stagiair niet de mogelijkheid moet hebben om een project toe te voegen, daar waar een admin wel beschikt over die mogelijkheid of dat een medewerker zichzelf niet meer of minder rechten kan geven en het volledige gebruikersbeheer kan door elkaar halen. Naast de inloggegevens zullen er dus ook rechtengegevens bijgehouden moeten worden en ook dat zal gebeuren in een sessie cookie.

5.4.1.1 Het inloggen

Wanneer een gebruiker nog niet aangemeld is, is het logisch dat hij geen pagina’s mag zien. De gebruiker zal dan ook altijd op de inlogpagina terecht moeten komen. Deze controle gebeurt op de plaats waar de templates getoond worden. Zoals u zich nog herinnert, gebeurde dit in de application.php met behulp van de $smarty->display() functie. Het is dan ook in deze file dat er een eerste controle zal gebeuren of er al een sessie aangemaakt is voor de gebruiker of niet.

application.php

```
if((isset($_SESSION['permissions'][$this->getController()])
and in_array($this->getAction(), $_SESSION['permissions'][$this-
>getController()])) or ($controller == 'users' and $action == 'login')
or ($controller == 'users' and $action == 'lostpass') ) {
      $Smarty->assign('controller', $this->getController());
      $Smarty->assign('action',     $this->getAction());
      $Smarty->assign('template',   $this->getTemplate());
} else {
      Sanmax_Error::push('login',"geen rechten voor $controller.$action");
      $message = Sanmax_Error::get();
      $Smarty->assign('message', $message);
      $Smarty->assign('controller', 'users');
      $Smarty->assign('action',     'login');
      $Smarty->assign('template',   'Users/login.html');
}
```

Wanneer er in de sessie een variabele een array bestaat permissions met als element de opgevraagde controller en in deze array de opgevraagde action voorkomt zal de gebruiker toegang krijgen tot de opgevraagde controller, action en template. Met andere woorden de gebruiker zal de opgevraagde pagina te zien krijgen. Er zijn hierop twee uitzonderingen: als de gebruiker de login pagina (/users/login) opvraagt of de pagina voor het opvragen van zijn verloren paswoord (/users/lostpass) zal hij hier wel toegang krijgen. Logisch, want een gebruiker die zijn paswoord kwijt is, kan natuurlijk ook niet inloggen. In alle andere gevallen wordt de gebruiker doorverwezen naar de loginpagina. Wanneer iemand voor het eerst naar de webpagina surft, zal hij uiteraard op de loginpagina terechtkomen. Ook omdat dit zo in het .htaccess bestand aangegeven is.

.htaccess

```
RewriteRule ^$ /users/login
RewriteRule ^ajax/([a-zA-Z-]+)/([a-zA-Z-]+)/?$
/bootstrap.php?controller=$1&action=$2&tpl=ajax     [QSA,L]
RewriteRule ^iframe/([a-zA-Z-]+)/([a-zA-Z-]+)/?$
/bootstrap.php?controller=$1&action=$2&tpl=iframe   [QSA,L]
RewriteRule ^([a-zA-Z-]+)/([a-zA-Z-]+)/?$
/bootstrap.php?controller=$1&action=$2&tpl=template [QSA,L]
```

Met behulp van een RewriteRule zal verwezen worden naar /users/login. Er volgen natuurlijk nog heel wat RewriteRule's maar deze zullen enkel toegepast worden wanneer er naar een specifieke pagina gesurft wordt. Vraagt de gebruiker de standaard URL op, intern is dat bijvoorbeeld http://projectbeheer, zal hij uitkomen op http://projectbeheer/users/login.

Zoals u al kon raden aan de hand van de URL vindt het afhandelen van de login plaats in de controller 'users' met als action (of in de methode) 'login'. Het lijkt me dus interessant ook deze methode even onder de loep te nemen.

/users/login

```
public function login()
{
  if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    if(session_is_registered('user_id')){
      header('Location: /home/overview');
      exit(0);
    }
  }
```

Wanneer men surft naar /users/login en men is al ingelogd (er is al een sessie variabele 'user_id' geregistreerd) zal men doorverwezen worden naar de startpagina.

```
  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $login = $_POST['login'];
    $pass = $_POST['pass'];
```

Wanneer men zijn gegevens ingeeft en op de knop aanmelden klikt, zullen de formuliergegevens in variabelen geplaatst worden.

```
    $User = DB_DataObject::Factory('users');
    if ($User->canLogin($login,$pass)){
```

Men zal controleren of de gebruiker kan inloggen. De methode canLogin zal controleren of de combinatie gebruikersnaam – paswoord voorkomt in de database. Het paswoord staat gecodeerd in de database. Het ingevoerde paswoord moet dus ook eerst gecodeerd worden om vergeleken te kunnen worden. Verder wordt ook nog gecontroleerd of de gebruiker die wil inloggen geen lid is van de usergroup 'not active' (id=1). Vanzelfsprekend mag een niet actieve gebruiker ook niet meer kunnen inloggen. Wanneer dit niet mogelijk is zal een error teruggegeven worden die in het else gedeelte van deze if getoond zal worden.

```
      $User->get('login',$login);
      $_SESSION['user_id'] = $User->id;
      $_SESSION['user_login'] = $User->login;
      $_SESSION['permissions'] = $User->getPermissions($User->id);
```

Wanneer de gebruiker kan inloggen, worden alle gegevens van de gebruiker opgehaald aan de hand van de login. Dit hoeft niet aan de hand van het id, want de login is ook uniek in de tabel users. Er zullen dan een aantal sessie variabelen aangemaakt worden zoals het id, de loginnaam en alle permissies. Merk op dat er hier geen session_start() opgeroepen wordt. Dit is namelijk al gebeurd in de application.php. De permissies worden opgehaald aan de hand van de user id. Aangezien rechten toegekend zijn aan een groep, worden in de methode getPermissions alle groepen doorlopen waaraan de gebruiker gekoppeld is. Voor elke groep worden op hun beurt alle gekoppelde rechten aan die groep opgehaald.

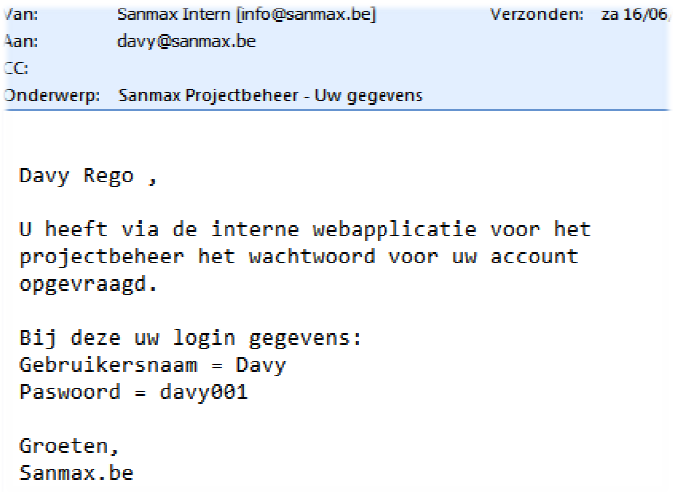
```
      $Logs = DB_DataObject::Factory('logs');
      $Logs->addLog($User->id,'info',"$User->login heeft zich aangemeld");
      header("Location: /home/overview");
      exit(0);
    }
```

Uiteindelijk zal de aanmelding van de gebruiker in de logs-tabel weggeschreven worden en de gebruiker doorverwezen worden naar de startpagina.

```
    else{
      $Logs = DB_DataObject::Factory('logs');
      $Logs->addLog(1,'error',"$login probeerde zich tevergeefs aan te
melden");

      $message = Sanmax_Error::get();
      return array('message' => $message);
    }
  }
}
```

Wanneer de gebruiker niet kan inloggen, wordt de foutmelding getoond en wordt er tevens een logrecord weggeschreven, zodat men eventueel misbruik kan opsporen.

Wanneer een gebruiker niet meer kan inloggen omdat hij zijn gegevens vergeten is, is er een functie voorzien voor het opvragen van die gegevens. Via de pagina /users/lostpass kan men die gegevens aanvragen door het e-mailadres op te geven. Op dit e-mailadres gebeuren een aantal controles. Allereerst zal gecontroleerd worden of het gaat om een geldige en bestaande e-mailadres. Via de validate-functie van PEAR is het mogelijk om te controleren of het domein gedeelte van een e-mailadres bestaat. Als tweede zal gecontroleerd worden of het adres ook daadwerkelijk voorkomt in de databank. De enige voorwaarde om deze functie te gebruiken, is dus dat de gebruiker zijn eigen e-mailadres kent, meestal zal dit toch het e-mailadres van het bedrijf zijn waar men de tool gebruikt (in dit geval dus voornaam@sanmax.be). Wanneer er een gebruiker met dergelijk e-mailadres teruggevonden wordt, zullen de inloggegevens opgehaald worden, het paswoord wordt gedecodeerd en alles wordt samen verstuurd in een e-mailbericht naar het opgegeven e-mailadres.


Van: Sanmax Intern [info@sanmax.be] Verzonden: za 16/06
Aan: davy@sanmax.be
CC:
Onderwerp: Sanmax Projectbeheer - Uw gegevens

Davy Rego ,

U heeft via de interne webapplicatie voor het projectbeheer het wachtwoord voor uw account opgevraagd.

Bij deze uw login gegevens:
Gebruikersnaam = Davy
Paswoord = davy001

Groeten,
Sanmax.be

Er is dus geen tussenkomst nodig van de administrator. Hij kan ook op geen enkel moment de wachtwoorden opvragen. De administrator kan enkel bij het toevoegen van een nieuwe gebruiker een wachtwoord opgeven. Of hij kan dit wachtwoord random laten genereren. Wanneer een nieuwe gebruiker toegevoegd wordt, zal deze gebruiker op zijn e-mailadres dat opgegeven wordt een bericht ontvangen met zijn nieuwe gegevens. De gebruiker kan natuurlijk te allen tijde zijn eigen paswoord wel aanpassen.

5.4.2 Rechtensysteem

Tot slot nog heel even stilstaan bij het rechtensysteem. Alle mogelijke rechten worden beschreven in een .ini bestand, namelijk in /etc/acl.ini. Hier kunt u naar believen rechten toevoegen. Al is de standaard opbouw van een recht 'acl.controller.action'.

/etc/acl.ini

```
acl.Logs.delete              =     "logs verwijderen"
acl.Logs.overview            =     "logs opvragen en tonen"
acl.Logs.detail              =     "log-details bekijken"
acl.Contracts.add            =     "contract toevoegen"
acl.Contracts.edit           =     "contract aanpassen"
```

Dit bestand bevat naast het recht ook een omschrijving van elk recht.

Al deze rechten zullen toegekend worden aan een groep. Voor het ophalen van de rechten uit het initialisatiebestand zal een model gebruikt worden. Dit is dus een voorbeeld dat een model niet altijd gegevens uit een databank moet halen.

De model die hiervoor gebruikt wordt is PermissionModel.php. Dit model bevat een enkele functie getPermissions die alle rechten uit het .ini-bestand zal ophalen en in een array plaatsen. Er wordt daarvoor gebruik gemaakt van de Registry library van Zend. (Deze werd ook al eens gebruikt voor gegevens uit de config.ini te halen).

/libs/DB/PermissionModel.php

```
class PermissionModel
{
    public function getPermissions()
    {
        $acl = Zend_Registry::get('acl');
        return $acl->acl->toArray();
    }
}
```

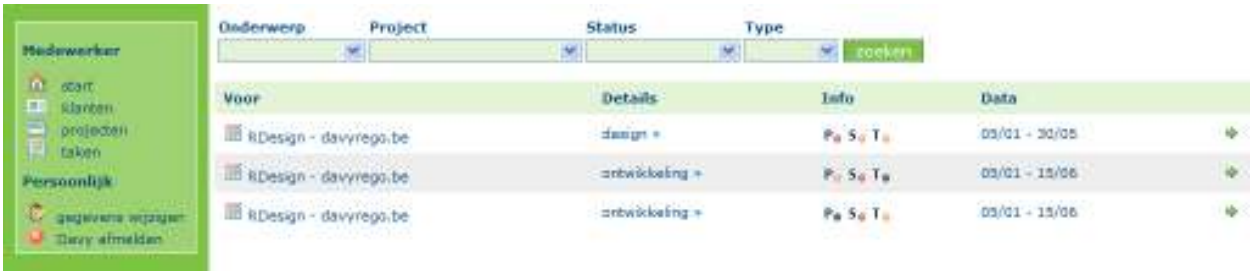
De administrator zal per groep rechten kunnen toekennen door deze rechten aan te vinken. Bij het opslaan zullen de rechten weggeschreven worden in de tabel 'Rights'. De belangrijkste gegevens die deze tabel bijhoudt, zijn de naam van de controller, de action en het group_id.

Wanneer een gebruiker zich aanmeldt, hebben we gezien dat de rechten waarover de gebruiker beschikt ook weggeschreven worden in een sessie. Bij elke handeling zal application.php controleren of de gebruiker over het recht beschikt om deze actie te mogen uitvoeren. Ook bij het tonen van bepaalde knoppen wordt deze controle uitgevoerd.

Dat gebruikers die tot verschillende groepen behoren daadwerkelijk andere rechten en mogelijkheden hebben, wordt aangetoond met de printscreens op de volgende pagina en met bijbehorende uitleg.

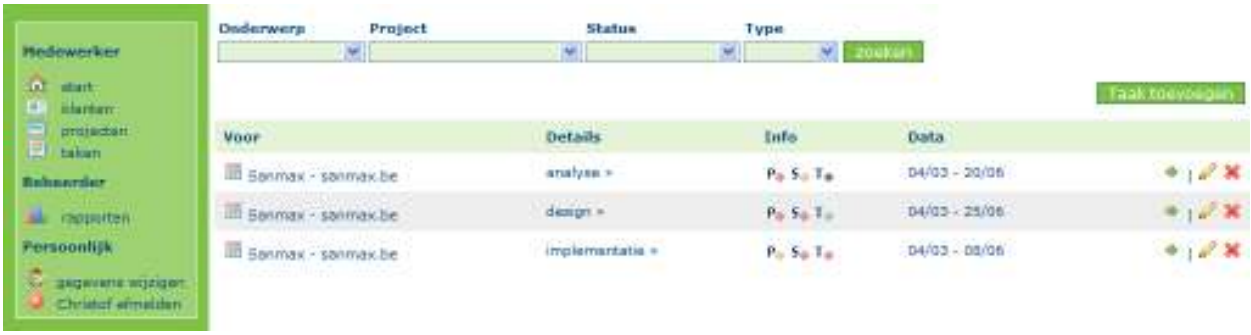
Stagiair:

Deze medewerker zal in het takenoverzicht enkel zijn taken te zien krijgen en enkel binnen deze taken kunnen zoeken. Omdat het hier over een stagiaire gaat, kan hij zijn taken ook niet aanpassen, hij kan enkel op het pijltje (uiterst rechts klikken) om jobs en jobdetails onder een taak te boeken. De taken zelf worden door de admin voor hem gegenereerd. Let ook op de beperkte functies waarover de stagiaire beschikt.



Developer:

Deze medewerker krijgt via het takenoverzicht ook enkel zijn taken te zien maar kan deze wel aanpassen of verwijderen. Hij/zij kan ook taken toevoegen via de knop 'Taak toevoegen' rechtsboven. Verder beschikt deze medewerker over een extra beheerfunctie in zijn menu, namelijk het genereren en opvragen van rapporten.



Admin:

Deze medewerker, of beter administrator, kan via het takenoverzicht en meer bepaald via de zoekbalk bovenaan alle taken opvragen van eender welke medewerker. Standaard krijgt ze ook enkel zijn/haar taken te zien. Ze kan ook taken aanpassen of verwijderen. Het is ook duidelijk dat een admin beschikt over heel wat meer functies dan dat een gewone medewerker doet.

5.5 JavaScripts, DHTML & AJAX

JavaScript is een scripttaal die vooral gericht is op het gebruik in webapplicaties. Wat de syntaxis betreft sluit JavaScript nauw aan bij Java. Maar daar houdt de vergelijking dan ook op. Toch verwarren heel wat mensen, die niet zo thuis zijn in het ICT gebeuren, beide talen met elkaar. De bedoeling van JavaScript is om webpagina's interactiever te maken.

DHTML, Dynamic HTML, is een term die gebruikt wordt voor dergelijke interactieve websites. Deze pagina's zijn opgebouwd uit de standaard HTML code in combinatie met een scripttaal zoals JavaScript en een CSS file voor de opmaak.

En tot slot AJAX. Asynchronous Java and XML is een techniek die de laatste jaren zeer bekend is geworden sinds de opkomst van de Web 2.0 websites. Web 2.0 is een term die gebruikt wordt voor de webapplicaties van dit moment. Webapplicaties die de vorm aannemen van een heus interactief platform, dat volgens sommigen zelfs de huidige lokaal geïnstalleerde software overbodig zou maken. AJAX speelt hierin een grote rol. Zoals de naam al verklapt, worden de gegevens asynchroon opgevraagd van de webserver. Het handige gevolg hiervan is dat de pagina's niet meer volledig opnieuw moeten ingeladen of vernieuwd worden, dankzij een XMLHttpRequest. Ook hier wordt JavaScript gebruikt om al de verschillende componenten van AJAX met elkaar te binden.

Op de volgende pagina's vindt u deze technieken toegepast in zeer concrete onderdelen die ook geïmplementeerd zijn geworden in mijn applicatie.

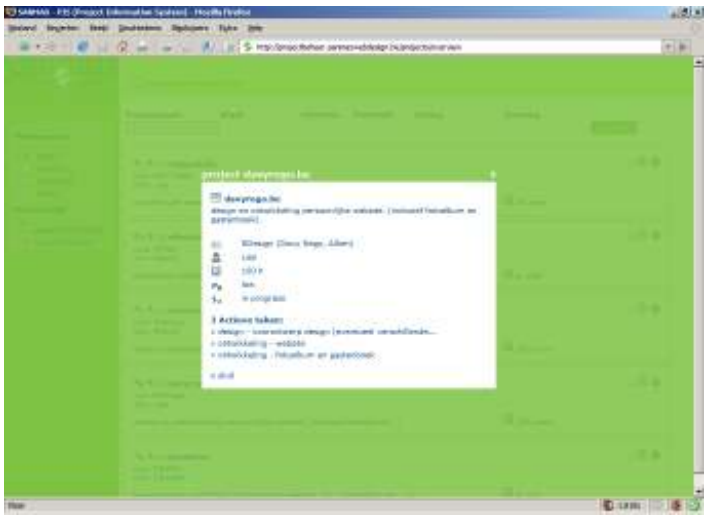
Had u graag voorbeelden gezien van Web 2.0 websites en webapplicaties ?
Typ dan zeker eens "web 2.0 examples" in het zoekveld van uw zoekrobot.

Of… een prachtig voorbeeld en aanrader van mijnentwege: www.netvibes.com !

5.5.1 JS: Modal Popup

Naam: LightWindow v1.2.1
Uitgever: Kevin Miller
Website: http://stickmanlabs.com/lightwindow

Een modal popup is een popup die binnen de webpagina zal verschijnen. De huidige pagina zal vervagen naar de achtergrond en de popup zal getoond worden in een kader op de zelfde website. Er zal dus geen nieuw browserscherm geopend worden. Deze popup's komt u onder allerhande namen tegen, zoals modal window, light box, slimbox,… genoemd. De specifieke popup die ik gebruikt heb, noemt lightwindow. Heel wat demo's en de scripts zijn terug te vinden op de website hierboven vermeld.



Waarom deze modal popup gebruiken ? De standaardpopup's worden door de meeste gebruikers ervaren als vervelend. Een logisch gevolg hiervan is dat de meeste browsers tegenwoordig ook standaard uitgerust zijn met een popup blocker. Die voorkomt dat er om de haverklap een nieuwe popup openspringt. Een goede oplossing voor de vele reclame en spam-popup's maar wel nadelig voor de popup's die toch enige waarde hebben.

5.5.1.1 Implementatie

De stappen die doorlopen moeten worden om deze modal popup te kunnen gebruiken in de applicatie zijn de volgende:

Het JavaScript dat zich bevindt in het bestand lightWindow.js (let op de js-extensie) wordt in de map /js geplaatst. In principe is het niet nodig om aanpassingen te doen in dit JavaScript, maar bij enige kennis van deze taal is dit wel perfect mogelijk. De opmaak gebeurt via een CSS-bestand dat u terug kan vinden in de style-map (templates/Style/lightWindow.css). Dit is een duidelijke toepassing van DHTML.

Bovenstaande bestanden, zowel de .js als de .css, zijn nodig om de eigenlijke modal popup op te kunnen roepen. Er moet dus in principe op elke pagina waar deze modal popup gebruikt zal worden aangegeven worden waar men deze bestanden kan vinden.

/templates/Common/header.html

```
<link href="/templates/Style/lightWindow.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="/js/lightWindow.js"></script>
```

Om dit vele knip- en plakwerk te voorkomen gaan we deze gewoon eenmaal toevoegen in de algemene header.html. Dit bestand wordt bij elke pagina toegevoegd. De scripts zullen dus voor iedere pagina opgeroepen en gebruikt kunnen worden.

Het eigenlijke oproepen of tonen van een modal popup gebeurt wanneer men op een link klikt waarbij in de <a>-tag de class “lWon” is toegevoegd. Deze naam komt ook terug in het JavaScript-bestand.

/templates/Projects/vcard.html

```
<a href="/iframe/projects/detail/?id={$project.id}" class="lWOn"
title="project {$project.name|stripslashes}">
      {html_image file='/images/icons/information.gif' title="details"}
</a>
```

In dit voorbeeld wordt er op de vcard van een project een info-knop geplaatst waarop men kan klikken om alle details te zien van een bepaald project. Er is dus een pagina detail.html gemaakt die de detailgegevens van het project waarvan het id meegegeven wordt zal tonen. Het enige probleem is dat wanneer we deze pagina zullen oproepen via de standaard URL zoals gewoonlijk (/projects/detail/?id={$project.id}), dan zal het menu ook verschijnen in de modal popup en dat is natuurlijk niet de bedoeling.

Om dit op te lossen wordt er voor de URL /iframe toegevoegd. En zo belanden we terug bij het .htaccess bestand. Bij het bekijken van het application.php bestand in hoofdstuk 5.3.3 heb ik dit ook al kort besproken. Er werd toen uitgelegd hoe men wist welke controller en action opgeroepen werd. Via een RewriteRule in het .htacces bestand (hieronder afgedrukt in het lichter blauw) werden de controller en action uit de huidige URL gehaald en als parameters meegegeven aan het bootstrap.php bestand. De RewriteRule die toen besproken is geworden, was echter niet de enige. Voordat de al besproken RewriteRule toegepast werd, wordt er nog getracht een andere RewriteRule uit te voeren, en het is deze die nu nodig is.

.htaccess

```
RewriteEngine on

RewriteRule ^iframe/([a-zA-Z-]+)/([a-zA-Z-]+)/?$
/bootstrap.php?controller=$1&action=$2&tpl=iframe   [QSA,L]
RewriteRule ^([a-zA-Z-]+)/([a-zA-Z-]+)/?$
/bootstrap.php?controller=$1&action=$2&tpl=template [QSA,L]
```

Wanneer de opgeroepen URL van de vorm iframe/controller/action is zal de eerste RewriteRule toegepast worden. U ziet dat er als template ‘iframe’ wordt meegegeven en niet de standaard ‘template’ die normaal gebruikt wordt. Door de [L] die vermeld staat tussen vierkante haken aan het einde van de RewriteRule wordt aangegeven dat het rewrite-proces stopt wanneer deze regel is toegepast en er geen andere meer uitgevoerd zal worden.

application.php

```
$Smarty->display('Common/' . $_GET['tpl'] . '.html');
```

In de application.php file zal via de Smarty-methode display nu iframe.html opgeroepen worden in plaats van de gebruikelijke template.html. En deze ziet er als volgt uit, zonder menu.

/templates/common/iframe.html

```
{include file='Common/header.html'}
{load_module controller="$controller" action="$action" template="$template"}
{include file='Common/footer.html'}
```

5.5.2 JS: Tooltip

Naam: Tooltips 3.45
Uitgever: Walter Zorn
Website: http://www.walterzorn.com/tooltip/tooltip_e.htm

Ieder die wel al eens een informatieve website ontwikkeld heeft, weet dat het niet altijd even gemakkelijk is om alle gegevens kwijt te geraken op de beperkte ruimte die er op een webpagina is. Zeker wanneer de resolutie beperkt is tot 1027 x 768 pixels. En niet alleen dat is een probleem, zelfs bij een grotere schermresolutie is het ook belangrijk om de pagina's overzichtelijk te houden en de gebruiker niet te overspoelen met informatie. Langs de andere kant kan bepaalde informatie voor de gebruiker toch wel relevant zijn. Er zijn verschillende oplossingen mogelijk. U kunt de extra informatie tonen in een nieuw venster, of zelfs in een modal window zoals hierboven uitgelegd… De simpelste en gebruiksvriendelijkste manier is misschien wel het tonen van de informatie in een tooltip. Een tooltip is een soort tekstballon die te voorschijn zal komen wanneer er over een bepaalde tag gegaan zal worden. In het afgebeelde voorbeeld wordt de omschrijving afgebroken omdat deze te lang is in het standaard getoonde overzicht. Door over de pijltjes te gaan met de muisaanwijzer krijgt de gebruiker een tooltip te zien met de volledige beschrijving voor dat project.

5.5.2.1 Implementatie

Ook voor het implementeren van de tooltip zijn een aantal handelingen nodig. Allereerst zal er ook een JavaScript bestand in de juiste map geplaatst moeten worden. In dit geval is dat de wz_tooltip.js die we in de map /js plaatsen. In deze file is er een configuratiesectie voorzien waarin je de standaardeigenschappen van een tooltip kan bepalen. Deze standaardwaarden kunnen echter ook per tooltip individueel aangepast en dus overschreven worden.

/js/wz_tooltip.js

```
...
var ttBgColor     = "#e5f3d9";
var ttBgImg       = "";
var ttClickClose  = false;
var ttDelay       = 0;
var ttFontColor   = "#004684";
var ttFontFace    = "Verdana, Arial, Helvetica, sans-serif";
var ttOpacity     = 100;
var ttShadowColor = "#e5f3d9";
var ttShadowWidth = 2;
...
```

In het JavaScript bestand kan de opmaak bepaald worden. Zoals de achtergrondkleur, een achtergrondafbeelding, tekstkleur, het lettertype, maar ook de doorzichtigheidgraad, een eventuele schaduw(kleur) die getoond moet worden, de vertraging waarna de tooltip moet verschijnen en nog veel meer van dergelijke opties.

/templates/Common/footer.html

```
<script type="text/javascript" src="/js/wz_tooltip.js" ></script>
</body>
```

Het JavaScript wordt opgeroepen in de footer.html. Daar waar dit HTML bestand bij iedere template toegevoegd wordt. Het JavaScript wordt net voor het afsluiten van de body-sectie toegevoegd. Dit is niet strikt noodzakelijk, zolang het JavaScript maar toegevoegd wordt na de laatste tag waarin een tooltip opgeroepen wordt.

/templates/Projects/vcard.html

```
<a href="#" onmouseover="this.T_TITLE='Omschrijving:
{$project.name|stripslashes}';
return escape('{$project.description|stripslashes}')">
   &raquo;
</a>
```

De tooltip wordt opgeroepen wanneer we met onze muisaanwijzer over de pijltjes zullen gaan, vandaar het gebruik van de onmousover-functie. Het belangrijkste dat binnen de aanhalingstekens gedefinieerd moet worden, is de return van de tekst die in de tooltip getoond zal worden. In dit geval dus de beschrijving van het project. Het is belangrijk om de stripslashes van Smarty toe te passen, anders zal de tekst mogelijk niet getoond worden. Alle speciale karakters moeten ook omgezet worden naar HTML entities, anders zal de tooltip ook niet getoond worden.

Met behulp van allerhande commando's kan elke tooltip individueel aangepast worden. Zo wordt bij bovenstaand voorbeeld ook een titel toegevoegd. Het is ook mogelijk om de standaardconfiguratiewaarden te overschrijven.

/templates/Projects/vcard.html

```
<a href="#" onmouseover="this.T_TITLE='Omschrijving:
{$project.name|stripslashes}'; this.T_FONTCOLOR='#FF0000'; this.T_OPACITY=50;
return escape('{$project.description|stripslashes}')">
   &raquo;
</a>
```

Zo is het bijvoorbeeld mogelijk om de standaardkleur die in wz_tooltip.js bepaald werd te overschrijven met een zelfgekozen kleur (hier rood) door bovenstaande aanpassing door te voeren. Of om de tooltip toch een doorzichtigheidgraad van 50% te geven terwijl standaard ingesteld staat dat de tooltip niet doorzichtig zal zijn. We krijgen als resultaat een tooltip zoals hierlangs afgebeeld.

Sinds 13 juni 2007 is er een nieuwe versie (v 4.01) van dit tooltip-script beschikbaar op de hierboven vermelde website. Deze versie is heel wat makkelijker en logischer te implementeren en te configureren en voorzien van tal van extra mogelijkheden. Hebt u als lezer van dit rapport de intentie om deze tooltip te gebruiken, dan raad ik u aan om de nieuwe versie te gebruiken. Alle uitleg vindt u terug op de website vermeldt aan het begint van dit artikel. De vorige versie, die ik gebruik, is ook nog terug te vinden op die website.

5.5.3 AJAX: VAT

Tot slot een toepassing van AJAX. Zoals al uitgelegd is het dankzij het gebruik van AJAX niet nodig om een pagina volledig te laten vernieuwen bij de aanpassing van gegevens. Ook in de ontwikkelde webapplicatie zit hiervan een toepassing. In de inleiding bleef de uitleg betreffende AJAX beperkt. We bekijken de gebruikte technieken die gecombineerd worden bij het gebruik van AJAX even nader:

- XHTML en CSS voor de presentatie volgens de standaarden van het W3C
- Het Document Object Model voor het dynamisch tonen van informatie en voor interactie.
- XML en XSLT voor de opslag, aanpassing en transport van gegevens. In mijn geval wordt dit vervangen door JSON (JavaScript Object Notation).
- Het XMLHttpRequest object voor asynchrone communicatie.
- JavaScript om alles aan elkaar te binden.

Online kan je heel wat AJAX libraries en toolkits downloaden die het merendeel van bovenstaande technieken op een handige manier zal implementeren, waardoor u zich daar dus niet al te veel van moet aantrekken en direct aan de slag kunt. Een voorbeeld daarvan is scriptaculous. De documenten worden vanzelfsprekend in de map /js geplaatst. Het scriptaculous pakket bevat heel wat AJAX controls en knappe effecten, maar voor een echte AJAX request is het bestand prototype.js noodzakelijk.

Ik zal gebruikmaken van een AJAX toepassing op de toevoeg- en wijzigpagina van de klanten. De gebruiker heeft de mogelijkheid om het BTW nummer van een klant in te geven. Aan de hand van dit BTW nummer zullen, indien juist en gekend, de klantgegevens opgehaald worden. Er wordt hiervoor gebruik gemaakt van een service aangeboden door de Europese Commisie. Het gaat om een webservice[1)] opgebouwd via WSDL (Web Service Definition Language) en deze kan door iedereen vrij aangesproken worden met behulp van een SOAP (Service Oriented Architecture Protocol) service. Door het gebruik van AJAX gebeurt dit uiteraard zonder dat de pagina opnieuw ingeladen wordt.

Ook op de website van de Europese Commissie[2)] kan deze webservice opgeroepen worden via een formulier. Wanneer we bijvoorbeeld het BTW nummer van Sanmax ingeven krijgen we het volgende te zien op de website:

VAT number	BE 462846881
Member State	BE
Name	BVBA SANMAX CONSULTANCY
Address	HERENGRACHT 59 3665 AS

[1)] http://ec.europa.eu/taxation_customs/vies/api/checkVatPort?wsdl
[2)] http://ec.europa.eu/taxation_customs/vies/en/vieshome.htm

5.5.3.1 Implementatie

/templates/common/header.html

```
<script type="text/javascript" src="/js/scriptaculous/prototype.js"></script>
```

Zoals al vermeld hebben we het script prototype.js nodig van het scriptaculous-pakket. Dit script wordt toegevoegd aan de header file, zodat elke pagina hierover kan beschikken. Functies in deze JavaScript zullen ook nog vanuit andere pagina's opgeroepen worden.

/templates/Customers/form.html

```
<script type="text/javascript" src="/js/vat.js"></script>
```

Als tweede zal er een JavaScript aangemaakt worden dat de request zal 'uitzenden' en het resultaat (response) zal verwerken en tonen in het juiste formaat en op de juiste plaats. Dit script moet vanzelfsprekend ook opgeroepen worden. Omdat dit slechts eenmaal gebruikt zal worden, gebeurt dit bovenaan in de form.html. En dus niet in de header.html wat in principe ook zou kunnen.

```
<input type="button" name="getAddress" class="button" value="Zoek"
onclick="return queryVat();" />
```

Er moet ook effectief een handeling gebeuren om de request te activeren. Dat gebeurt op deze pagina door de klikken op de zoekknop. Zoals u ziet wordt bij een onclick het resultaat van de functie queryVat() teruggegeven.

De functie queryVat() is de functie die zich bevindt in het vat.js bestand dat we zonet hebben toegevoegd aan form.html. Ik overloop dit bestand onderdeel per onderdeel.

/js/vat.js

```
function queryVat() {
    var loading = $('loading');
    loading.innerHTML = '<img src="/images/lightwindow/ajax-loading-small.gif" alt="" />';
```

Allereerst zal er bij het starten van deze functie een loading-afbeelding getoond worden.

/templates/Customers/form.html

```
<div id="loading" style="float: left"></div>
```

Deze wordt getoond in een div in de form.html die de naam 'loading' kreeg. Deze naam is aangegeven binnen de haakjes na het dollarteken. Het is dus belangrijk dat deze naam zowel hier, als in het de template, hetzelfde is.

```
    var company = $('company');
    var address = $('address');
    var city    = $('city');
    var zip     = $('zip');
    company.value = "";
    address.value = "";
    city.value    = "";
    zip.value     = "";
```

Voorts worden er ook variabele gemaakt van de velden die effectief opgevuld moeten worden, aangezien we hier later een waarde zullen moeten aan kunnen toekennen. Van het moment dat deze variabele aangemaakt zijn, geven we hun als waarde ene lege string. Dit om te voorkomen dat wanneer men een tweede (fout) BTW nummer ingeeft dat niet kan worden gevonden, de gegevens van de eerste aanvraag blijven staan.

Hierna volgt de eigenlijke code die de handeling zal doen. Er wordt een variabele opt aangemaakt die zal handelen als JSON-object. Deze variabele zal aan het einde van deze functie meegezonden worden met de AJAX request.

```
    var opt = {
        method: 'post',
        postBody: 'vat=' + $F('vat') + '&country=' + $F('country'),
```

Er wordt aangegeven dat de methode die gebruikt wordt een POST-methode is. Aangezien er op de zoekknop geklikt zal worden. De body die meegezonden zal worden bevat het BTW (VAT) nummer en de landcode (vb.: 'BE' voor België).

```
        onSuccess: function(t) {
            var result  = eval('(' + t.responseText + ')');
```

Naast de method en postBody, bevat deze ook een onSuccess sectie. In deze sectie wordt de functie gedeclareerd die het resultaat dat, in dit geval teruggekregen wordt van de webservice, zal verwerken en uiteindelijk ook zal tonen op de webpagina. Het resultaat (responseText) dat men terugkrijgt zal in een variabele result geplaatst worden waar verder mee gewerkt zal worden.

```
        var myCompany     = result.name.substring(7);
        var myCompanyType = result.name.substring(0,7);
        if (myCompanyType=="M      " || myCompanyType=="MV     "){
            myCompanyType="";
        }
        var myAddress  = theAddress[0];
        var ZipCity    = theAddress[1].split('  ');
        var myZip      = ZipCity[0];
        var myCity     = ZipCity[1];

        var CompanyArray = myCompany.split(" ");
        var i=0;
        for (i=0;i<CompanyArray.length;i++){
            CompanyArray[i] =
(CompanyArray[i].substring(0,1).toUpperCase()).concat(CompanyArray[i].substring(1).toL
owerCase());
        }
        myCompany = "";
        for (i=0;i<CompanyArray.length;i++){
            myCompany += CompanyArray[i].concat(" ");
        }
        myAddress =
(myAddress.substring(0,1).toUpperCase()).concat(myAddress.substring(1).toLowerCase());
        myCity =
(myCity.substring(0,1).toUpperCase()).concat(myCity.substring(1).toLowerCase());
```

Zoals u waarschijnlijk al heeft opgemerkt, wordt het resultaat anders getoond in mijn applicatie dan op deze website van de Europese commissie. Allereerst wordt de straat, postcode en gemeente uit elkaar gehaald. Ook het anders schikken en het omzetten naar kleine letters, uitgezonderd de beginhoofdletter gebeurt in bovenstaande code.

```
            company.value = myCompany.concat(" ",myCompanyType);
            address.value = myAddress;
            city.value    = myCity;
            zip.value     = myZip;

            loading.innerHTML = '';
        },
    }
```

De in het juiste formaat gegoten gegevens zullen uiteindelijk toegekend worden als waarde aan de variabele waarmee de tekstvelden in het formulier opgevuld worden. Het resultaat is dus gevonden en wordt getoond, wat betekent dat de laadanimatie verborgen mag worden.

```
    new Ajax.Request('/ajax/customers/validatevat', opt);
}
```

Het laatste stukje code, maar eigenlijk het belangrijkste uit de hele functie, is het versturen van de request. Er wordt een URL opgeroepen met de controller en de action. Verder worden de options die hierboven gespecificeerd werden ook meegegeven. Ajax.Request is een functie die uitgeschreven is in het JavaScript prototype.js.

/js/scriptaculous/prototype.js

```
Ajax.Request = Class.create();
Ajax.Request.Events =
  ['Uninitialized', 'Loading', 'Loaded', 'Interactive', 'Complete'];

Ajax.Request.prototype = Object.extend(new Ajax.Base(), {
  _complete: false,

  initialize: function(URL, options) {
    this.transport = Ajax.getTransport();
    this.setOptions(options);
    this.request(URL);
  },
...
```

Let ook hier weer op met de gevormde URL. Deze wordt voorafgegaan door /ajax. Het .htaccess bestand zal ook hier weer een RewriteRule toepassen en er voor zorgen dat de juiste template opgeroepen wordt.

.htaccess

```
RewriteRule ^ajax/([a-zA-Z-]+)/([a-zA-Z-]+)/?$
/bootstrap.php?controller=$1&action=$2&tpl=ajax   [QSA,L]
```

In dit geval zal de /templates/Common/ajax.html opgeroepen. Dit bestand ziet er als volgt uit:

/templates/Common/ajax.html

```
{load_module controller="$controller" action="$action"
template="$template"}
```

Er wordt op deze pagina dus geen menu getoond, maar ook geen header.html en footer.html ingeladen. Enkel de module die de controller, action en template bevat.

In de Ajax.Request werd dus een URL opgegeven die er als volgt uitzag: /ajax/customers/validatevat. Dit betekent dus ook dat de actie 'validatevat' in de controller 'customers' voorzien zal moeten worden. In deze actie zal de eigenlijke webservice opgeroepen worden en het resultaat van deze webservice teruggegeven worden, wat dan op zijn beurt verwerkt zal worden in de JavaScript functie die op de vorige pagina's al werd geëxpliceerd.

/controllers/Customers.php

```
public function validatevat()
{
	require_once 'Zend/Json.php';
```

De gegevens die teruggegeven zullen worden, zijn in de vorm van een JSON-object gegoten. Om deze te kunnen verwerken, wordt er gebruik gemaakt van een classe van het Zend-Framework. Deze moet dan vanzelfsprekend opgeroepen worden.

```
	$country = $_POST['country'];
	$vat     = $_POST['vat'];

	$data = array(
	    'countryCode' => $country,
	    'vatNumber'   => $vat
	);
```

De geposte gegevens zijn, zoals eerder vermeld, de landcode en het BTW nummer. Deze variabele worden in het juiste formaat gegoten (hier: array) met de juiste benamingen zodat ze herkend en verwerkt kunnen worden door de webservice.

```
	$Soap   = new
SoapClient('http://ec.europa.eu/taxation_customs/vies/api/checkVatPort?wsdl');
	$valid  = $Soap->checkVat($data);
```

Via een SOAP-service zullen we de eigenlijke webservice aanspreken. Service Oriented Architecture Protocol is een protocol voor het verzenden van op XML-gebasseerde berichten via het internet. Deze service moet wel geactiveerd en ondersteund worden door de server.

checkVatPort?wsdl

```
<element name="checkVat">
	<complexType>
		<sequence>
			<element name="countryCode" type="xsd:string"/>
			<element name="vatNumber" type="xsd:string"/>
		</sequence>
	</complexType>
</element>
```

De functie die de controle zal doen noemt checkVat en verwacht een array met de twee variabelen 'countryCode' en 'vatNumber'. Deze functie is ook duidelijk te herkennen in het WSDL-bestand dat opgeroepen wordt. De SOAP service zal voor deze functie een object van de 'stdClass' teruggeven met alle nodige attributen.

```
object(stdClass)#23 (6) { ["countryCode"]=> string(2) "BE"
["vatNumber"]=> string(9) "462846881" ["requestDate"]=> string(10)
"2007-06-15" ["valid"]=> bool(true) ["name"]=> string(25) "BVBA SANMAX
CONSULTANCY" ["address"]=> string(23) "HERENGRACHT 59 3665 AS" }
```

```
            $prop   = get_object_vars($valid);
            $result = array();
            foreach ($prop as $key => $value) {
                $result[$key] = $value;
            }

        return array('json' => Zend_Json::encode($result));
    }
```

Wanneer het BTW nummer geldig is, zal het object '$valid' gegevens zoals bedrijfsnaam en adres, bevatten. Deze gegevens worden in een array geplaatst door de objectvariabelen een voor een te doorlopen. Uiteindelijk zal het resultaat omgezet worden naar een JSON-object dat dan teruggegeven zal worden aan de functie.

Om helemaal in orde te zijn met de regels van het framework moet in principe ook een template aangemaakt worden voor bovenstaande action. In de map Customers plaatst u dan ook best een met de actie gelijknamige HTML file. In deze file plaatsen we enkel de expressie {$json} omdat volgens het MVC-model de output in feite in de view getoond moet worden.

/templates/Customers/validatevat.html

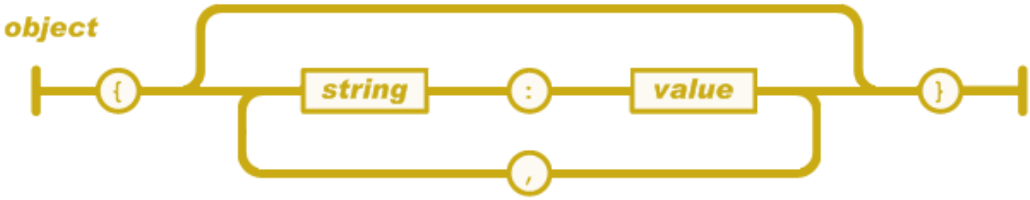
```
{$json}
```

5.5.3.2 JavaScript Object Notation

Even wat meer uitleg over die JSON-objecten. JavaScript Object Notation is een subset voor de programmertaal JavaScript. Het wordt, zoals u al merkte, gebruikt voor het uitwisselen van datastructuren, met name in webapplicaties die asynchroon gegevens ophalen van de webserver (= toepassing van AJAX).

JSON is een goed alternatief voor XML, dankzij zijn eenvoud. De gegevens worden uitgewisseld in de vorm van JavaScript-expressies. Door simpelweg de JSON-expressies te evalueren kunnen deze ingelezen worden in een JavaScript. Het is dus niet nodig om een apart parser te gebruiken.

Een JSON object is opgebouwd volgens de volgende structuur:



Concreet betekend dit de Array die we op het einde van de action returnen het volgende JSON-object bevat:

```
["json"]=> string(156) "{
  "countryCode":"BE",
  "vatNumber":"462846881",
  "requestDate":"2007-06-15",
  "valid":true,
  "name":"BVBA SANMAX CONSULTANCY",
  "address":"HERENGRACHT 59\n3665 AS"
}
```

5.6 Cronjobs

Om de projectopvolging zo goed mogelijk te laten plaatsvinden, heb ik gebruik gemaakt van het proces cron in mijn applicatie. Cron is een geavanceerde taakplanner voor Unix-based systemen. Je kunt commando's of, in mijn geval, (PHP) scripts met bepaalde tijdsintervallen of op een bepaald tijdstip laten uitvoeren. Let op, cronjobs configureren is niet voor iedereen mogelijk. Je moet hiervoor allereerst shell toegang hebben tot de server. Omdat dit bij Sanmax niet voor iedereen mogelijk is, heeft Christof (toegang tot de server met root-paswoord) mijn cronjob geconfigureerd. Het opstellen van het script is echter door mezelf gebeurd.

In mijn specifiek geval zullen de prioriteiten van een taak of een project gecontroleerd en indien nodig aangepast worden. Dit zal voor de taken gebeuren op basis van de resterende tijd tot aan de deadline (einddatum) van de taak. De prioriteit van een project zal bepaald worden aan de hand van de prioriteiten van de onderliggende taken. De administrator of medewerker hoeft dus zelf niet de controle te gaan doen of hij een taak nu al op prioriteit medium of high moet zetten of hij deze nog kan laten staan op low, dit zal dagelijks volledig automatisch gebeuren.

Deze oplossing zal ook voorkomen dat een project of taak in de vergeethoek zal geraken. Taken worden in het takenoverzicht onder andere gesorteerd op prioriteit. Wanneer een taak of project toegevoegd wordt met een prioriteit "low" en een deadline die pas binnen 5 maanden bereikt wordt, is de kans dus groot dat dit project wel eens uit het oog verloren wordt. Dankzij deze cronjob zal dit niet meer gebeuren. Naargeland de deadline dichterbij komt zal de prioriteit dus automatisch aangepast worden en de taak ook naar bovenschuiven in het takenoverzicht.

Om dit script goed te kunnen bereiken is het niet onbelangrijk om te weten welke prioriteiten er zijn, wat hun id is en vanaf welk moment ze toegepast zullen worden. Het gene dat aangepast zal worden is de 'priority_id' in de tabellen 'tasks' en 'projects'.

Id	Prioriteit	Toegepast (duur tot deadline)
1	Very High	< 1 week
2	High	< 2 weken
3	Medium	< 3 weken
4	Low	> 3 weken

Het "algoritme" dat gebruikt wordt om het toe te wijzen id te bereken is het volgende, met ernaast een concreet voorbeeld.

$$\frac{\text{eindatum-huidige datum}}{\text{7 dagen}} \qquad \frac{29/06 - 16/06}{\text{7 dagen}} = \frac{13}{7} = 1{,}86$$

Het resultaat van deze berekening zal naar boven afgerond worden. Wanneer het resultaat van de berekening 1,86 is zal dus prioriteit 2 (High) toegepast worden. Wanneer het resultaat (veel) groter is dan 4 zal automatisch prioriteit 4 toegekend worden.

Het eigenlijke script dan. Dit script is geplaatst in de map /etc, maar heeft uiteindelijk niets meer met het framework of het MVC-model te maken. Toch zullen we gebruik maken van enkele functionaliteiten van het framework.

\etc\jobs\updatepriorities.php

```
<?php
session_start();
$_SESSION['user_id'] = 1;

$stage = $argv[1] ? $argv[1] : 'staging';
$paths = array(
    get_include_path(),
    dirname(dirname(dirname(__FILE__))) . '/libs'
);

set_include_path(implode(PATH_SEPARATOR, $paths));
set_time_limit(0);
ini_set('memory_limit', '128M');

require_once 'DB/dataobject.php';
require_once 'DB/dataobject/Cast.php';
require_once 'DB/Logs.php';
require_once 'Sanmax/Logs.php';
require_once 'Zend/Config/Ini.php';

$Config  = new Zend_Config_Ini(dirname(dirname(__FILE__)) . '/config.ini', $stage);
$options = &PEAR::getStaticProperty('DB_dataobject', 'options');
$options = $Config->db->asArray();
```

Zoals het gebruik van dataobjecten en het het wegschrijven van de logs. Al deze scripts moeten natuurlijk opgevraagd worden. Alsook de eigenlijke configuratiefile voor de connectie naar de databank.

```
$Logs = DB_dataobject::Factory('logs');
$Logs->addLog(1,'CronJobs',"Start uitvoeren updatepriorities.php");

$Projects = DB_dataobject::Factory('projects');
$Projects->find();
while ($Projects->fetch()) {
  echo "\nProject {$Projects->name}:\n";

  $average = 0;
  $Tasks = DB_dataobject::Factory('tasks');
  $Tasks->selectAdd('(FLOOR(DATEDIFF(date_end, NOW())/7)) AS my_priority_id');
  $Tasks->whereAdd('status_id <> 4 and status_id <> 5');
  $Tasks->whereAdd("project_id = '{$Projects->id}'");
  $Tasks->whereAdd("(FLOOR(DATEDIFF(date_end, NOW())/7)) <> priority_id");
  $count = $Tasks->find();
```

Alle projecten zullen overlopen worden, en binnen een project alle taken. De belangrijkste gegevens, die van een taak opgehaald zullen worden is de berekening van het algoritme dat op de vorige pagina uit de doeken gedaan werd. Vertaald in MySQL wordt dit: (FLOOR(DATEDIFF(date_end, NOW())/7). Niet alle taken zullen opgehaald worden, taken met als status_id 4 (done) of 5 (invoice) zullen niet meer gecontroleerd worden. Ook de taken die door toepassing van het algoritme geen ander priority_id zouden krijgen worden niet opgehaald, dit enkel om de server niet overbodig handelingen te laten doen.

```
while ($Tasks->fetch()) {
      $priorityId = $Tasks->my_priority_id <= 4 ? $Tasks->my_priority_id : 4;

      echo "\n\t" . $Tasks->id . " - " . $Tasks->description . ":\n";
      echo "\tstatus van {$Tasks->priority_id} naar {$priorityId}\n";

      $TaskUpdater = clone $Tasks;
      $TaskUpdater->priority_id = $priorityId;
      $TaskUpdater->update();

      $average += $priorityId;
  }
```

Voor elke taak zal het priority_id aangepast worden. Als het resultaat van de berekening kleiner of gelijk is aan 4 wordt dat resultaat toegewezen. Is het resultaat groter (wat betekent dat er in ieder geval nog meer dan 3 weken tot de deadline zijn, dan wordt de priority_id automatisch herleid tot 4. Uiteindelijk zal hier dus ook een update van doorgevoerd worden en is de prioriteit van de taak aangepast.
Voor het berekenen van de priority_id van het project worden de priority_id's van de onderliggende taken opgeteld.

```
  if (!$average > 0) {
      continue;
  }
```

Wanneer de som van alle priority_id's van de taken niet groter is dan nul wil dit zeggen dat er geen onderliggende taken gevonden zijn die voldeden aan de voorwaarden. Er zal dan verder gegaan worden, en meer bepaald zal het volgende project ingeladen worden en men zal stoppen met de overige berekeningen.

```
  $average = round($average / $count);
  echo "\tproject status van {$Projects->priority_id} naar {$average}\n";

  $ProjectUpdater = clone $Projects;
  $ProjectUpdater->priority_id = $average;
  $ProjectUpdater->update();
}
```

Voor de berekening van het gemiddelde zal de som gedeeld worden door het aantal gevonden taken. De variabele $count kreeg zijn inhoud via de find()-methode op het dataobject Tasks. Dit resultaat wordt standaard afgerond (kleiner dan 0,5 na de komma naar onder, gelijk of groter dan 0,5 naar boven). En ook hier zal op zijn beurt het priority_id geupdate worden.

```
$Logs->addLog(1,'CronJobs',"Einde uitvoeren updatepriorities.php");
```

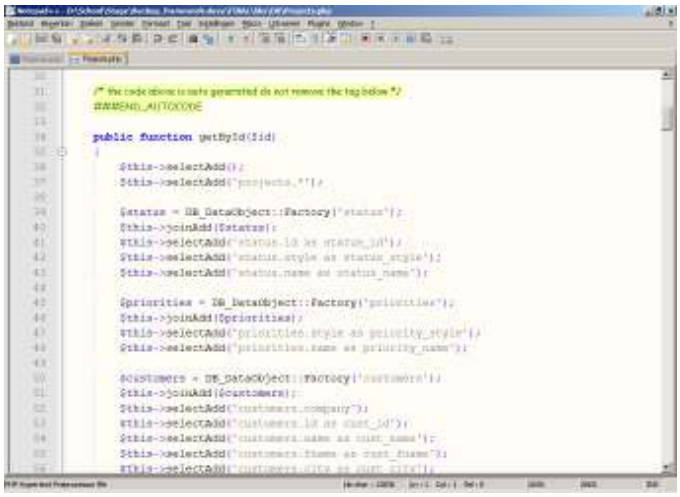
Uiteindelijk zal er nog een log weggeschreven worden die aangeeft dat de cronjob uitgevoerd en voltooid is. Zodat men ook vanuit de webapplicatie de logs kan bekijken en kan controleren of de cronjob daadwerkelijk uitgevoerd is.

Dit script is een mooi en duidelijk voorbeeld van de automatisatie van de projectopvolging en zal bij Sanmax dagelijks om middernacht uitgevoerd worden.

5.7 Tools

In dit onderdeel bespreek ik een aantal handige tools die ik gebruikt heb gedurende de ontwikkeling. Per toepassing schets ik in het kort wat hun functionaliteit is, eventueel aangevuld met een klein voorbeeld.

5.7.1 Notepad ++



Naam: Notepad++
Uitgever: sourceforge.net
Website: notepad-plus.sourceforge.net

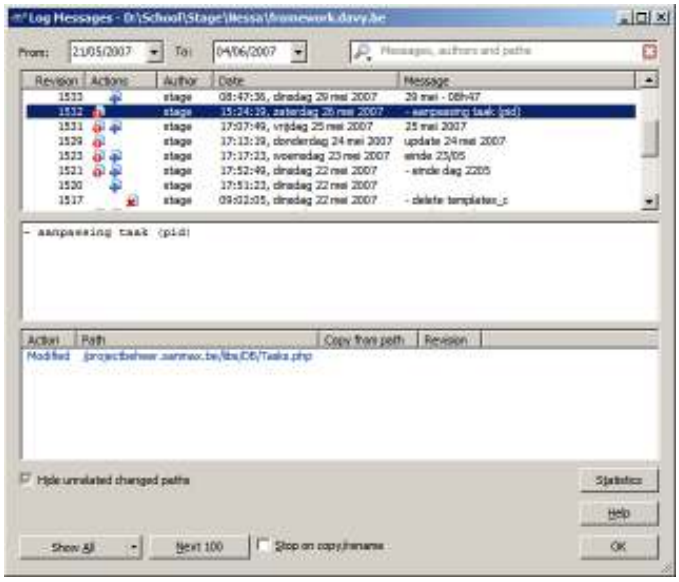
Notepad ++ is een source code editor. Het is in feite een uitgebreide versie van de standaard notepad of kladblok die in Windows ingebakken zit. Dit project is onder de GNU licentie, wat dus betekent dat het gratis te gebruiken is. Notepad ++ was bij mij al bekend voor ik mijn stage begon. Het is een snel programma dat heel wat talen ondersteunt, zoals C, C++, Java, XML, HTML, PHP, JavaScript, CSS, SQL, VB/VBS, ini-files,… en zo kan ik nog wel even doorgaan. Zo voorziet men voor elke taal een bepaalde opmaak van de code zodat de structuur van de verschillende tags duidelijk wordt. Verder ook een auto-aanvulfunctie, die voor elke taal weliswaar beperkt is tot de bekendste functies. In tegenstelling tot andere editor software start dit programma snel op en zijn er geen overbodige plugins voorzien. Ik kan het gerust omschrijven als de editor software voor de programmeur die echt nog zelf elk karakter van zijn code wil schrijven.

Een andere mogelijkheid die gebruikt kon worden is het meest bekende webapplicatie ontwikkelingsprogramma Dreamweaver, sinds kort uitgebracht door Adobe, voordien door Macromedia. Voor Dreamweaver is er ook een plug-in voorzien voor het werken met de Smarty template engine. Maar zoals aangehaald, start dit programma veel langzamer op en zitten er heel wat handige, maar in mijn geval, overbodige tools in.

Nog een ander programma is Zend Studio. Ik heb deze tool even aan het werk gezien bij de collega's en dit was wel handig. Dit programma biedt bijvoorbeeld goede autocomplete functies voor allerhande PHP functies. Daarbij komt ook nog dat deze software duidelijk de structuur van het framework kon herkennen. Zo wist het perfect autoaanvullingen te doen voor de namen van dataobjecten en hun attributen. Een handige tool, maar toch koos ik voor Notepad. Omdat ik op deze manier het programmeren het snelst onder de knie zou krijgen, aangezien alles zelfs getypt moest worden.

5.7.2 Tortoise SVN

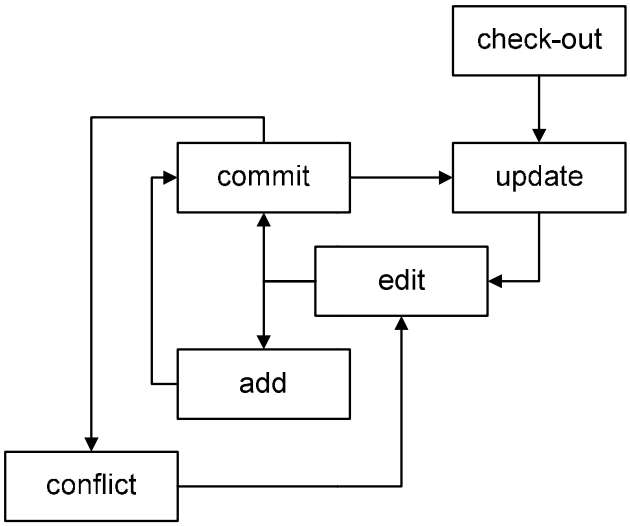
Naam: TortoiseSVN
Uitgever: Tigris
Website: subversion.tigris.org



Om te weten wat de software Tortoise SVN doet, moet u eerst weten waarvoor SNV staat. Subversion, of kortweg SVN, is een tool waarmee het versiebeheer van een softwareontwikkelingsproject uitgevoerd wordt. Met dergelijke tool kunnen de bestanden in hun verschillende stadia van hun ontwikkeling beheerd worden. Zo kan men verschillende versies van bestanden beheren. Dergelijke systemen bewaren allerhande informatie zoals de wijziging die doorgevoerd wordt en eventueel een reden. Zo is het mogelijk om later bij problemen terug te schakelen naar een vorige versie van een bepaald bestand. Verder is het op deze manier ook gemakkelijk om met meerdere aan een project te werken. Aangepaste bestanden kan u door middel van een update gemakkelijk binnenhalen en bij conflicten zal ook daarvan een melding gegeven worden en kan u kiezen voor een bepaalde versie van het bestand.

Tortoise SVN is een Windows client voor Subversion die ook weer als open source gratis gedownload kan worden. Het is bij Sanmax de bedoeling dat iedereen die aan een project werkt op zijn pc een kopie heeft staan van het project op de server, de working copy. Ook dit is een voordeel van het Subversion. Op deze manier hebt u in feite een back-up op uw eigen pc in tegenstelling tot wanneer u rechtstreeks op de server zou werken.

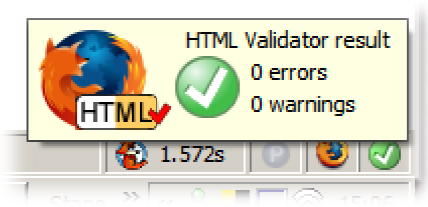
Om even de werking van SVN te schetsen onderstaand schema en bijhorende standaard procedure die gehanteerd zullen worden:



- Maak een eenmalige ‘check-out’ van het project waarmee u bezig bent.
- Het is verstandig een nieuwe ‘update’ te doen. Dan weet u zeker dat u met de nieuwste versie van de files aan de slag gaat.
- U gaat nu een of meerdere bestaande file(s) bewerken (‘edit’).
- U kunt ook nu weer een ‘update’ uitvoeren om te kijken of er niet toevallig iemand aan een of meerdere van de door u bewerkte files heeft gewerkt en vóór u een ‘commit’ heeft uitgevoerd.
- Als dat niet het geval is, geeft u een ‘commit’ om de source bij te werken (‘merge’).
- Als u een nieuwe file of een nieuwe directory met een of meerdere files aan de source wilt toevoegen, gebruikt u de optie ‘add’. Daarna geeft u een ‘commit’ om de file of map met file(s) daadwerkelijk aan de source toe te voegen.

5.7.3 HTML Validator

Naam: HTML Validator
Uitgever: Marc Gueury
Website: users.skynet.be/mgueury/mozilla



HTML Validator is een extensie voor de browser Mozilla Firefox. Deze plug-in zal de HTML code valideren van de webpagina die opgeroepen wordt via de browser. Via een icoon in de statusbar zal aangegeven worden of de pagina al dan niet fouten bevat. Via een detail scherm zullen dan de fouten getoond worden en, indien gekend, suggesties meegegeven worden hoe deze fouten voorkomen kunnen worden.

De extensie is gebaseerd op Tidy en OpenSP. Dit zijn twee algoritmes voor het valideren van HTML code die ontwikkeld worden door het Web Consortium, beter bekend onder de naam W3C. Het Tidy-algoritme controleert vooral de HTML tags en de XHTML syntaxis. OpenSP is een iets wat strenger algoritme. Het gaat om een SGML-parser (Standard Generalizes Markup Language) gebaseerd op DTDs (Document Type Definition). Ik heb gebruik gemaakt van beide algoritmes tijdens het controleren van mijn webpagina's. Allereerst zal de SGML-parser de pagina controleren. Worden er geen fouten gevonden wordt het Tidy-algortime toegepast.

Al deze benamingen zeggen misschien niet zo heel veel. Daarom enkele voorbeelden van fouten die gecontroleerd zullen worden en die ikzelf geregeld tegengekomen ben. Allereerst zal de opbouw gecontroleerd worden van een document. Alle tags die geopend worden, worden deze ook correct en op de juiste plaats afgesloten ?

✗ `<td> <strong> beschrijving </td> <strong>`
✓ `<td> <strong> beschrijving </strong> </td>`

Daarnaast zal de inhoud van de tags bekeken worden, worden de juiste opties meegegeven met de juiste syntaxis ?

✗ `<tr height="20px">`
✓ `<tr style="height: 20px">`

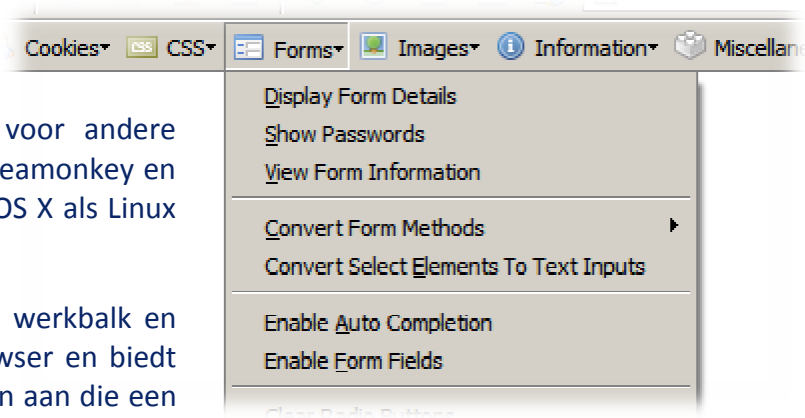
Ook de eigenlijke inhoudt wordt gecontroleerd, wordt er bijvoorbeeld gebruik gemaakt van HTML entities (= tekenentiteitreferentie) ?

✗ `Ontwikkeling website & forum`
✓ `Ontwikkeling website &amp; forum`

Er bestaan ook nog allerhande alternatieven voor HTML validatie, al gaat het in de meeste gevallen om online validators. De bekendste, ook van W3C is de validator op de website validator.w3.org. De code wordt dan ingeladen, verstuurd naar een server en daar gecontroleerd. Het leuke aan de validator die ik gebruik is dat controle lokaal en direct kan gebeuren, zonder extra handelingen of wachttijden.

5.7.4 Webdeveloper tool

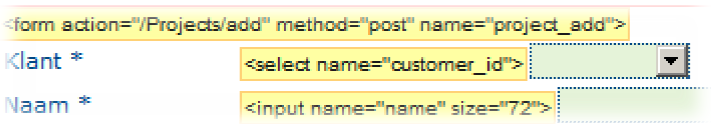
Naam: Webdeveloper tool
Uitgever: Chris Pederick
Website: chrispederick.com/work/web-developer



Ook dit is een extensie die opgeroepen zal worden via de browser Mozilla Firefox. Deze tool is echter ook beschikbaar voor andere browsers zoals Flock, Mozilla en Seamonkey en dit zowel op een Windows, MAC OS X als Linux platform.

Deze extensie voegt een handige werkbalk en een nieuw menu toe aan de browser en biedt werkelijk heel wat functionaliteiten aan die een goede webdeveloper meermaals nodig zal hebben. Een korte opsomming van wat ik intensief gebruikt heb bij deze tool:

- Cookies: Zowel sessie- als domein cookies op een eenvoudige en snelle manier bekijken en leegmaken. Ik heb gebruikt gemaakt van deze functies bij het ontwikkelen van de inlogpagina en het rechtensysteem.
- CSS: Het uitschakelen van de toegepaste stylesheets of net het toepassen van een bepaalde stylesheet. Ik heb deze tool gebruikt voor het uittesten van een stylesheet die toegepast zal worden bij het afprinten van de rapportering pagina. Via de opties: 'Display CSS by Media Type: Print' kon ik op een snelle manier controleren of de opmaak van de print-pagina in orde was.
- Forms: Via deze optie kan alle informatie van een formulier opgevraagd worden. Zo kunnen met een klik op de knop alle formulier details getoond worden. Een zeer handige functie om bijvoorbeeld te kijken welke benaming een bepaald veld gekregen heeft.

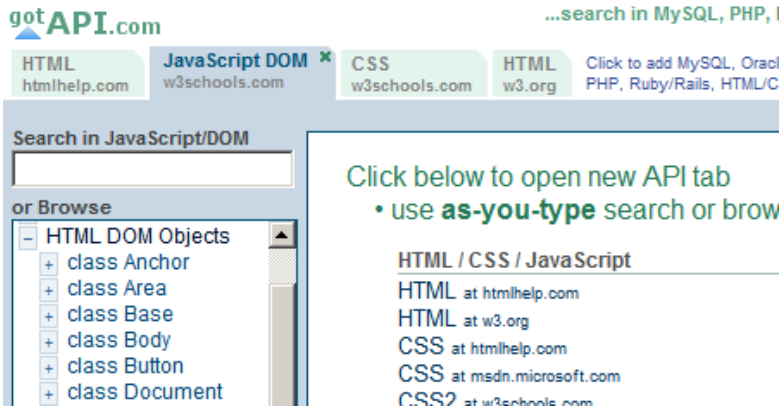


- Verder ook nog tools voor het beheer van de afbeeldingen op een pagina. Ook algemene informatie die opgevraagd kan worden over alle tags aanwezig in de code. Een handige resize functie die de webpagina snel in een andere resolutie toont zodat u ook hier rekening mee kunt houden.
- En tot slot zijn er ook nog enkele validatie tools ingebouwd. Deze zullen meestal externe websites oproepen voor de validatie van de webpagina. Hiervan heb ik dan ook geen gebruik gemaakt, ook omdat ik hiervoor al beroep doe op de HTML Validator extensie.

Deze tool is in mijn ogen niet alleen een aanrader voor iedere webontwikkelaar, maar kan ook een leuk hebbeding zijn voor de intensievere surfer onder ons.

5.7.5 gotAPI

Naam: gotAPI
Uitgever: LogPerspective Inc
Website: www.gotapi.com



De laatste in het rijtje is niet echt een gereedschap, maar daarom niet minder handig. “Documentation at your fingertips”, zo omschrijft de website zichzelf en dat kan ik alleen maar bevestigen.

Iedereen die wel eens websites ontwikkeld heeft, zal wel eens een manual geraadpleegd hebben of erger, op zoek gegaan zijn naar een goede manual of tutorial via uw zoekrobot. Het kan echter efficiënter. Gotapi.com kan je beschouwen als een groot naslagwerk. Op deze website vindt u namelijk heel wat handleidingen terug en dat op één enkele pagina. Let wel op, gotAPI centraliseert de verschillende manuals op hun website… maar zelf maken ze geen manuals aan. De handleidingen van onder andere PHP en MySQL die u op gotapi.com terugvindt, zijn dezelfde als die u vindt op de officiële websites. Maar alleszins hebt u geen drie browserschermen meer nodig met de PHP, de MySQL en de JavaScript manual. Zeker bij het werken in een framework of met AJAX, waar meerdere talen met elkaar gecombineerd worden, is dit zeer handig. Via tabbladeren kan je switchen tussen de verschillende programmeertalen en manuals.

Wanneer u naar de website surft, zal u standaard twee schermen te zien krijgen. Links een zoekbalk waar u kunt zoeken in HTML tags, rechts een overzicht van alle beschikbare ‘APIs’ die u kan toevoegen als tabblad aan uw gepersonaliseerde pagina.

De APIs die ik geregeld gebruikte waren:

- HTML (htmlhelp.com)
- CSS (w3schools.com)
- JavaScript/DOM (w3schools.com)
- PHP (php.net)
- MySQL (mysql.com)

Maar u vindt er ook nog andere terug zoals APIs rond XML, Java, PHP Frameworks zoals Symphony en CakePHP. En de lijst wordt voortdurend aangevuld. Als u een webapplicatie-ontwikkelaar bent, zou deze website haast verplicht in uw favorietenlijst moeten voorkomen !

6 PERSOONLIJKE RAPPORTERING

In dit onderdeel kijk ik met een kritische blik terug op een aantal gebeurtenissen gedurende mijn stageperiode. Wat heb ik bijgeleerd en kon/kan ik Sanmax bijbrengen? Waar liep het (bijna) fout? Wat kon er beter? Ik heb gekozen om een aantal van mijn bedenkingen neer te schrijven in column-stijl. Ook vindt u in dit hoofdstuk de mening van onder andere mijn stagebedrijf over het verloop van mijn stage terug.

6.1 Column “Saved by a ‘s’ !”

Op maandag 14 mei ontsnapte ik op het nippertje aan een kleine catastrofe. Moest het geluk niet aan mijn zijde gestaan hebben dan waren alle data uit de huidige tool verwijderd.

Even de situatie schetsen. Tijdens het programmeren van een validatie methode als invoercontroles was het nodig om te kijken in welk formaat bepaalde gegevens momenteel opgeslagen worden. Zoals steeds gebruik ik hiervoor de tool van MySQL zelf, MySQL Query Browser. Via deze tool kunnen er gemakkelijk en snel queries uitgevoerd worden, tabeldefinities opgevraagd worden en scripts samengesteld worden. Deze tool gebruik ik echter ook voor het uitvoeren van mijn script. Een script dat er voor zorgt dat alle tabellen terug gereset en geïntialiseerd worden met de standaard (test-)gegevens.

MySQL Query Browser werkt op die manier dat wanneer u gegevens opvraagt uit of een query uitvoert op een bepaalde tabel de database waarin deze tabel zich bevindt, geactiveerd zal worden als standaard. Het is dus mogelijk om meerdere databases te beheren via deze tool, maar er zal telkens maar één actief zijn. Ik had dus gegevens opgevraagd uit een tabel uit de database van de huidige PMtool. Nadat ik gevonden had wat ik zocht kon ik dus verder programmeren. De gemaakte validatie methode even uittesten, wat gegevens toevoegen en aanpassen, enzovoorts. En na dit alles, naar goede gewoonte, mijn script een keer uitvoeren zodat alle tabellen terug gereset worden.

En hier ging het fout, door onoplettendheid klikte ik op de execute-knop en voerde ik mijn script uit. U raadt het al, het werd uitgevoerd in de PMtool-database die nog actief was. Zoals u al dan niet al gemerkt heeft, begint mijn script met het droppen van alle tabellen, waarna ze een voor een terug aangemaakt zullen worden. Na enkele seconden had ik al door dat er iets niet klopte, en wanneer ik surfte naar de huidige webapplicatie was het helemaal duidelijk. Er was iets serieus misgegaan !

Na de nodige zweetdruppels en enkele diepe zuchten bleken de negatieve gevolgen nog zeer beperkt. Gelukkig had ik enkele dagen voordien al mijn tabellen naar het meervoud omgezet. MySQL gaf namelijk problemen op de tabellen ‘Group’ en ‘Right’ aangezien dit gereserveerde benamingen zijn. Het was dus nodig om deze tabellen te hernoemen. Om een zekere uniformiteit te behouden heb ik dan maar alle tabelnamen in het meervoud geplaatst. Een zeer goede daad, zo bleek achteraf, aangezien de tabellen in de database van PMtool ook in het

enkelvoud staan. Dit had rampzalige gevolgen kunnen hebben. De tabellen 'customer', 'project', 'task', 'job',… met al hun gegevens waren dan verloren gegaan.

De schade bleef dus gelukkig beperkt. Enkel de tabel 'Groups' en 'Rights' die ook in de PMtool-database in het meervoud stonden, moesten terug aangepast worden. Maar aangezien het hier respectievelijk ging om een zeer kleine tabel en een standaardtabel was dit geen moeilijke, laat staan ondoenbare, klus. En na een klein half uur functioneerde alles terug alsof er niets gebeurd was.

Dit voorval heeft gelukkig geen negatieve gevolgen gehad, integendeel. Ook binnen Sanmax kan men hier uit leren. Tot die dag werd er geen back-up genomen van de interne gegevens, en meer bepaald de data uit de PMtool. Misschien toch geen slecht idee om dit naar de toekomst wel te gaan doen. Dit om te voorkomen dat op een milliseconde maanden aan gegevens verloren gaan. Dit hoeft daarom niet steeds door een menselijke handeling te zijn, maar ook hardware problemen bij de interne server kunnen deze gevolgen teweegbrengen.

"We back up our data on sticky notes because sticky notes never crash."

Een andere tip die meegegeven kan worden is niet iedereen zomaar alle rechten te geven op een resource. Zo was het voor mij bijvoorbeeld voldoende geweest om enkel selects uit te kunnen voeren op de bestaande database. Moest het toch nodig zijn om de gebruiker aanpassingen te laten doorvoeren, dan kan het altijd handig zijn om de database even te exporteren en een kopie aan te maken. Hierin kan dan wel ten volle gewerkt, en per ongeluk verwijderd, worden zonder dat dit enige gevolgen heeft.

6.2 Column "Het (on)nut van een (goede) analyse ?!"

"Waarom een dergelijke diepgaande en uitgebreide analyse maken ?", dat was de vraag die mij wel eens gesteld werd door mijn collega's. En diezelfde vraag heb ik een aantal weken later, zeker naar het einde toe, voor mezelf ook een aantal keren trachten te beantwoorden. Tevergeefs… of niet ?

Het is niet pessimistisch om te stellen dat een goede analyse niet garant staat voor een goed eindresultaat. Een analyse moet zonder meer begrijpbaar en overzichtelijk zijn. De inhoud is gedetailleerd doch goed afgebakend. Naast het feit dat een analyse zo zorgvuldig moet gebeuren zijn er nog twee belangrijke voorwaarden waaraan ook voldaan zal moeten worden.

Allereerst is het nodig om een analyse grondig te bespreken met de opdrachtgever. Deze verantwoordelijkheid ligt bij de analist zelf. Hij moet de opdrachtgever of klant benaderen en vragen om samen de analyse te overlopen en te bespreken. Dit kan in de vorm van een presentatie, een kort overlegmoment of zelfs een gezellige zakenlunch. Ook de opdrachtgever speelt echter een grote rol in het goede verloop van deze bespreking. De analist verwacht steeds een eerlijke repliek en een doordachte feedback. De opdrachtgever als de uiteindelijke uitvoerder heeft niets aan een goede analyse wanneer deze later toch niet volledig blijkt te voldoen aan de verwachtingen en de eisen van de opdrachtgever. Het snel doorbladeren van een analyse of het intensief ja-knikken tijdens de bespreking lijkt voor de opdrachtgever op dat moment misschien de beste oplossing - dan ben ik er vanaf - maar achteraf kan dat grote, zelfs ongewenste, gevolgen teweegbrengen. Enkele uren uittrekken om een analyse nauwkeurig onder de loep te nemen en je bemerkingen te noteren is heus geen overbodige luxe.

"My team has created a very innovative solution, but we're still looking for a problem to go with it."

De tweede belangrijke regel kan je pas zinvol noemen wanneer aan de eerste voorwaarde voldaan is. Wanneer er overeengekomen is met de opdrachtgever dat de analyse voldoet aan de verwachtingen kan er gestart worden met de eigenlijke uitwerking en ontwikkeling van het project. De grote pijnpunten die men zou kunnen tegenkomen zijn dankzij een goede analyse al weggewerkt. De analyse zal door de ontwikkelaar verder gebruikt worden als een leidraad gedurende de programmatie. Ook op basis van deze analyse zal men een planning opstellen en aan de opdrachtgever kunnen meedelen wanneer men het project afgerond ziet. De programmeur zal hierbij natuurlijk rekening houden met een speling van een x-aantal dagen of weken, afhankelijk van de omvang van het project. Ook moet er een uitgebreide testfase en implementatie periode voorzien worden.

Nieuwe ideeën of plotse geniale invallen kunnen zeer leuk zijn voor het uiteindelijke resultaat, maar heel wat minder plezant voor de ontwikkelaar die zich zo goed mogelijk aan zijn planning tracht te houden. Bij de start van een project moet dit dus zo grondig mogelijk en vanuit allerhande invalshoeken bestudeerd worden. Het is aan de uitvoerder van het project om er voor te zorgen dat de opdrachtgever zich bewust is van het belang van de analyse. Gebruik gerust de opgestelde documenten, of een samenvatting ervan, als een contract. De klant verklaart zich zo akkoord met het eindresultaat dat vooropgesteld wordt.

Natuurlijk moeten we dit alles nuanceren. Een project kan niet van de eerste keer perfect uitgewerkt worden en voldoen aan de eisen van de opdrachtgever. Zelfs niet bij de beste analyse waarbij de opdrachtgever al het nodige deed dat van hem verwacht werd. Bepaalde onderdelen zullen altijd voor interpretatie vatbaar blijven. Daarom lijkt het verstandig om toch een aantal aanpassingmogelijkheden te voorzien, want het is natuurlijk ook niet de bedoeling om de opdrachtgever met de rug tegen de muur te zetten. Laat hem bijvoorbeeld de mogelijkheid om gedurende de ontwikkeling 2 kleine en 1 grote aanpassing door te voeren. Wanneer dit vooraf, in overleg met de opdrachtgever bepaald is, kan men hier ook rekening mee houden bij het opstellen van de planning en de tijd dat men hier aan besteedt mogelijk mee incalculeren.

Een goede analyse, maar ook en vooral de omkadering ervan, kan leiden tot een goed eindresultaat. Waarbij zowel de ontwikkelaar tevreden kan zijn over het verloop en het eigenlijke resultaat, alsook de opdrachtgever die over de nodige inspraak zal beschikken bij aanvang van het project en (beperkt) kan beschikken tijdens de ontwikkeling.

Beluister deze column ook via podcast op mijn website www.davyrego.be.

De mening van Sanmax

Davy heeft dankzij een eigen grondige analyse van het project potentiële pijnpunten voorkomen. Op eigen initiatief kwam hij regelmatig met nieuwe ideeën en oplossingen op de proppen, die het gebruiksgemak kwalitatief verhoogde. Davy denkt vooruit en is nooit in problemen gekomen met de vooropgestelde planning. Ook wanneer er nieuwe zaken werden toegevoegd aan het project kwam de deadline nooit in het gedrang. In de testfase werden aanpassingen onmiddellijk doorgevoerd.

Het intranet dat Davy heeft ontwikkeld is veel gebruikersvriendelijker dan het intranet waar Sanmax voordien mee werkte. Werkelijk een plezier om mee te werken ! Voor Sanmax is deze stagiair een meerwaarde geweest !

Stagebegeleidster Sanmax
Lies Ceuppens

6.4 De mening van OFFITEL

Even kort voorstellen wie of wat Offitel is: we zijn een call- en contactcenter. Sanmax maakt gebruik van onze accommodatie, wat dus betekent dat ook Davy gedurende 3 maanden samen met ons onder één dak heeft doorgebracht.

Ik moet dan ook toegeven dat we enkele keren gebruik gemaakt hebben van zijn algemene ICT-kennis. Zo zocht hij samen met ons naar oplossingen voor enkele kleine problemen in het databankprogramma MS Access, dat we gebruiken voor het beheer van onze gegevens voor onze telefoniesoftware.
Daarnaast heb ik samen met Davy een praktisch probleem in onze software besproken, waarmee we al langer zaten. Al snel kwam hij met een oplossing, die we dan ook doorgegeven hebben aan onze softwareleverancier die deze aanpassing dan ook zal doorvoeren.

Kortom, Davy was voor ons een aangename “collega” en heeft ook ons wat bijgebracht. Verder verzekerde hij ons dat we eventueel ook na zijn stage nog beroep op hem mogen doen.

Zaakvoeder Offitel
Luc Vannitsen

Sanmax

PHL

7 CONCLUSIE

Na het voltooien van mijn stage kan ik afsluiten met het cliché dat ik heel wat heb bijgeleerd. Al is dat echt wel het geval. De wereld van dynamische websites, die voor mij bij aanvang zo goed als onbekend was, is voor mij opengegaan. De mogelijkheden zijn heel uitgebreid en het is tof om in een gestructureerde omgeving te werken dankzij het gebruik van een framework. Er is in mijn ogen duidelijk nog heel wat potentieel voor de webapplicaties, web 2.0 in het achterhoofd houdende. Mede omwille van deze reden vind ik het spijtig dat deze materie niet meer aan bod gekomen is tijdens de opleiding. Al zal dit vanaf volgend academiejaar wel het geval zijn door de invoering van de afstudeerrichtingen.

Het eindresultaat van 14 weken zwoegen, mag gezien worden. De door mij ontwikkelde webapplicatie wordt ook daadwerkelijk gebruikt bij Sanmax voor de interne administratie. De beheerapplicatie werd op mijn laatste werkdag gedoopt als de “PIS-tool”. Een ludieke afkorting voor “Project Information System”. Natuurlijk zijn er nog een aantal schoonheidsfoutjes opgedoken, deze zijn al of zullen nog weggewerkt worden. In overleg met mijn stagebegeleider zal ik de eerste weken na mijn stage beschikbaar blijven om eventueel nog een aantal aanpassingen te doen.

Voor mij persoonlijk sluit ik met deze conclusie niet alleen mijn stagerapport af, maar ook mijn opleiding Toegepaste Informatica, en meer nog, mijn studentenleven. In de afgelopen drie jaar is er bij momenten wel eens gezucht en gepuft, maar dat zal in de toekomst niet anders zijn. Met het nodige karakter en doorzettingsvermogen is alles, of alleszins veel, mogelijk. En dankzij het totaalpakket van de opleiding aan de PHL, waartoe ook deze praktijkervaring behoorde, is er een goede basis gelegd voor een mooie toekomst.

> *A goal without a plan is just a wish.*
>
> *Larry Elder, Amerikaans talkshow host*

8 BIJLAGEN

8.1 Activiteitenverslagen

8.1.1 Week 02: 05-03 – 16-03

Op maandag 4 maart begon ik mijn stage bij Sanmax bvba te Genk. Sanmax is een dynamisch internetbedrijf dat heel wat toepassingen en services aanbiedt zoals webdesign, database management, webhosting, e-marketing, webapplicaties, enzovoort.

De eerste dag was het natuurlijk even wennen aan de nieuwe omgeving. Maar al bij al verliep deze aanpassing redelijk vlot. Ik kreeg de nodige gegevens waarmee ik toegang had tot de nodige resources nodig voor de uitwerking van mijn project. Zoals toegang tot de MySQL-databanken, alle tot nu toe verwezenlijkte projecten, een e-mailaccount van Sanmax en toegang tot de huidige beheertool die ik als opdracht zal moeten herwerken.

Ook de inhoud van het door mij uit te werken project werd uit de doeken gedaan. Het doel van het project is om een functionele en overzichtelijke webapplicatie uit te werken voor het beheer van de klanten en de bijhorende projecten, die intern bij Sanmax zal draaien. Een webapplicatie voor de eigen administratie van Sanmax.

Momenteel gebruikt men een gratis webapplicatie die niet volledig voldoet aan de eisen en beperkt is in functionaliteit.

Samen met mijn stagebegeleidster Lies bekeken we het huidige systeem en bespraken we de tekortkomingen. Niet enkel het probleem werd behandeld, maar er werd ook hardop nagedacht over de manier waarop deze tekortkomingen kunnen weggewerkt worden. Ook had ik een kort gesprek met zaakvoerder Pascal, om even te polsen wat zijn ervaringen waren met de huidige tool.

Hoe begin je nu aan dergelijke project? Wanneer ik direct zou gestart zijn met de programmatie was de kans relatief groot dat ik tijdens de ontwikkeling op heel wat obstakels zou stoten. Ook vanuit de opleiding is altijd (terecht) gepleit voor een goede analyse die dan een gefundeerde basis zal vormen voor de verdere uitwerking. Het leek me dus evident om een grondige en volledige analyse uit te werken. Al blijkt wel dat men aan deze stap in de werkelijkheid niet zo heel veel belang hecht. Bij grotere projecten wordt er natuurlijk "nagedacht" en worden er enkele bemerkingen genoteerd. Maar er wordt niet geanalyseerd volgens een bepaalde standaard.

Nadat ik de huidige tool zelf grondig bekeken had, enkele korte “interviews” met enkele medewerkers had en het plan van aanpak opstelde, kon ik begonnen met de analyse. Allereerst werd de informatieanalyse volgens FCO-IM uitgewerkt. Aan de hand van de opgestelde feittype expressies en met behulp van de tool CaseTalk werden de nodige diagrammen gegenereerd. Met deze gegevens kan men in CaseTalk een relationeel schema laten genereren, op basis

waarvan het gegevensmodel zal opgesteld worden. Aan de hand van dit model zal op zijn beurt de databasestructuur afgeleid worden. Deze databasestructuur werd gecontroleerd door Christof, de senior webdeveloper van Sanmax. Hij bezorgde me dan ook de nodige feedback en informatie over de werkwijze die bij Sanmax gehanteerd wordt. Zoals het gebruik van Engelse termen, het gebruik van een underscore in plaats van een koppelteken, enzovoorts… Met deze opmerkingen werd uiteraard rekening gehouden en de nodige aanpassingen aan het databasemodel werden doorgevoerd.

Het tweede grote deel van de analyse dat uitgewerkt werd is de design met UML, in feite is dit de object georiënteerde analyse. Niet de volledige OO-analyse zal uitgewerkt hebben maar enkel de use-case diagrammen, dit omdat PHP niet echt een object georiënteerde taal is. Dit gebeurde onder andere in overleg met Mevr. Peetermans, wiens mening ik hierover gevraagd had. Met behulp van deze use-case diagrammen kan op een overzichtelijke en begrijpelijke manier de functionaliteiten van het te realiseren systeem weergegeven worden. Dit onderdeel van de analyse kan in latere fases (ontwikkeling, implementatie en testfase) ook gebruikt worden als checklist om te controleren of alle vooropgestelde eisen voldaan zijn. Voor het uitwerken van de use-case diagrammen werd gebruik gemaakt van Visual Paradigm, de tool die men momenteel op 2TIN gebruikt. En dus niet met Rational Rose zoals men dit academiejaar op 3TIN gebruikt heeft. Dit was natuurlijk even aanpassen, maar het werken met Visual Paradigm bood aanzienlijk meer mogelijkheden.

Tot slot werd de structuur van de webapplicatie ook in een diagram gegoten, dit met behulp van MS Visio. Dit volgens een specifieke werkwijze gedefinieerd in een online tutorial gebaseerd op het boek Information Architecture for the World Wide Web 1).

Tussentijds werd er als “ontspanning” ook al gewerkt aan de opmaak van het stagerapport volgens de BIN-normen met gebruik van alle functionaliteiten binnen MS Word 2007, zodat er op het einde van de stage hier niet al te veel tijd ingestoken zal moeten worden.

Buiten het nog eens goed nalezen van de nodige documentatie zit de analyse er dus zo goed als volledig op. In de komende werkdagen zal dus de MySQL databank opgezet worden waarna in de resterende weken de eigenlijke programmatie kan plaatsvinden. Er zijn al enkele bergen (werk) verzet, maar de “Mount Everest” staat momenteel nog te wachten. Hoe ik dit aanpak leest u ongetwijfeld in het volgende activiteitenverslag binnen 2 weken.

8.1.2 Week 04: 21-03 – 30-03

Na een tof maar vermoeiend verlengd weekend door de schoolvierdaagse naar de CeBIT beurs in Hannover hervatte ik op woensdag het werk bij Sanmax. (De uitgewerkte opdracht vindt u terug in meegeleverde document.)

Nadat ik in de eerste twee weken de analyse uitwerkte kon ik de databank waarmee de applicatie zal communiceren opzetten. Zoals gebruikelijk voor een PHP webapplicatie maak ik gebruik van een MySQL databank. MySQL is een open source, en dus gratis te gebruiken, relationeel database management systeem (RDBMS). De taal die gebruikt wordt voor het communiceren met dergelijke database is, zoals de naam het al verklapt, SQL. Dit was geen enkel probleem aangezien er tijdens mijn opleiding intensief met SQL gewerkt werd, op enkele aanpassingen en uitbreidingen in de syntaxis na. Ik maakte een volledig script voor het aanmaken van de tabellen binnen de database en het vullen van bepaalde standaardtabellen. Hiervoor maakte ik gebruik van twee MySQL-administratie applicaties. Enerzijds is er het webgebaseerde phpMyAdmin, dat intern bij Sanmax op de server draait. Anderzijds heb ik ook gewerkt met MySQL Query Browser. Dit is een gratis 'frontend' (applicatie met gebruikersinterface) waarin korte maar ook gecompliceerde statements in gegenereerd en uitgevoerd kunnen worden. Kortom een zeer gebruiksvriendelijke programma dat voor intensieve gebruikers van MySQL zeker eens het bekijken waard is.

Het enigste obstakel dat ik tegenkwam tijdens het genereren was het type van tabel dat gekozen moest worden. Standaard zijn de MySQL tabellen van het type MyIsam, maar op aanraden van Werner bekeek ik ook een keer de andere types. Het type InnoDB leek interessant. In tegenstelling tot MyIsam wordt hier locking voorzien op rijniveau in plaats van op tabelniveau. Een nuttige extra aangezien het wel zal voorkomen dat men gelijktijdig naar eenzelfde tabel weg zal schrijven (bijvoorbeeld: wanneer elke medewerker 's morgens bij aanvang van het werk zijn taken voor die dag zal definiëren in het systeem). Een nog groter voordeel is dat er gewerkt kan worden met referenties naar andere tabellen, zodat er gewerkt kan worden met hetzelfde principe als er steeds tijdens onze opleiding gebeurde. Wanneer er bijvoorbeeld een record verwijderd zal worden dat refereert naar een bestaand record in een andere tabel, zal voorkomen kunnen worden dat dit record zomaar verwijderd kan worden.

Een tweede hoofdactiviteit tijdens de derde week was het samenstellen van een presentatie. Op eigen initiatief heb ik de vraag gesteld of het mogelijk was om de uitgewerkte analyse te presenteren. Niet alleen om te laten zien op welke manier ik het huidige en het toekomstige systeem geanalyseerd heb, maar ook om feedback te kunnen krijgen. Zodat ik niet met verkeerde opvattingen of functionaliteiten aan de ontwikkeling zou beginnen. Na een aantal dagen vertraging, deed ik op maandag 26 maart via een PowerPoint-presentatie de analyse uit de doeken. Ik vertelde aan de zaakvoerder en mijn stagebegeleidster waarom een dergelijke analyse zo belangrijk is, op welke manier ik te werk ging en welk resultaat ik hoopte te bereiken. Met behulp van enkele diagrammen en de nodige explicatie trachtte ik een overzichtelijke en begrijpbare kijk op het toekomstige systeem te creëren. De reacties na de presentatie waren positief. Naast enkele interessante opmerkingen die ik zal meenemen tijdens de programmatie werd er ook goed gereageerd op de lay-out van de PowerPoint-presentatie.

De rest van de vierde week besteedde ik uit aan het kennismaken en het onder de loep nemen van Smarty. Eerst bekeek ik enkele 'simpelere' Smarty-toepassingen die ik terugvond op de website van Smarty (smarty.php.net). Later bouwde ik deze toepassingen verder uit om zelf ook al enkele regels code Smarty getypt te hebben. Zo was er een gastenboek script, waarbij de gebruiker berichten kan lezen en zelf wegschrijven in het gastenboek. Ik breidde deze applicatie uit zodat de gebruiker een dag kon selecteren en enkel de berichten van die dag te zien kreeg. Een relatief kleine uitbreiding, maar wel ideaal als eerste kennismaking met Smarty en het opfrissen van PHP.

Aan het einde van de week was het dan tijd voor het grotere werk. Samen met Andries bekeek ik een webapplicatie die hij momenteel aan het ontwikkelen is voor voetbalclub KRC Genk, voor het beheer van de seat-bezetting in hun stadion. Ook hier wordt gebruikt gemaakt van Smarty, in combinatie met een framework. Dit framework werd deels door Andries zelf uitgewerkt, gebaseerd op het bestaande Zend framework (framework.zend.com). Ook de applicatie die ik zal uitwerken zal volgens dit principe gebeuren. Het is dan ook de bedoeling dat ik dezelfde structuur en werkwijze zal aanhouden als in het project dat Andries aan het uitwerken. Zowel Andries als Werner, beiden al kennis gemaakt met dit Smarty framework, waarschuwden me wel dat het even kan duren voor de 'klik' er komt. Ze raden me dan ook aan om hier toch een dikke week voor uit te trekken. Maar eens ik het principe door heb zou het wel makkelijker werken zijn, voegden ze er aan toe.

Aangezien ik toch van plan was om tijdens de paasvakantie door te werken zal dit geen probleem vormen. Ik kan me beter goed voorbereiden dan onbezonnen te starten met programmeren en om de regel vast te lopen. Het is wel mogelijk dat ik in de vakantie een aantal dagen van thuis uit werk, aangezien ik in principe momenteel niets nodig heb op de server. Moest dit wel zo zijn kan ik nog altijd van thuis uit via de openVPN verbinding connectie maken naar de server bij Sanmax. Toch zal ik grotendeels in Genk zelf aanwezig zijn, opdat ik dan bij onduidelijkheden direct opklaring kan vragen aan de specialisten ter plaatse.

8.1.3 Week 06: 02-04 – 13-04

Gedurende de eerste week van de eigenlijke paasvakantie bekeek ik het volledige framework waarin gewerkt zal worden. Nadat ik uitgebreid en gedetailleerd kennis heb kunnen maken met de al bestaande applicaties die volgens dezelfde structuur opgebouwd zijn als ik zal moeten doen, kan ik ook u een beter overzicht geven van op welke manier er gewerkt wordt.

Er zijn eigenlijk drie grote onderdelen die gebruikt worden in een dergelijke webapplicatie. Allereerst is er de template engine Smarty. Smarty is dus in feite niet het framework, waar mee gewerkt zal worden zoals ik eerder vermeldde. Smarty zorgt er enkel en alleen voor dat de PHP code volledig gescheiden blijft van de eigenlijke design. Er wordt in feite een presentatie layer gecreëerd. Het grote voordeel hiervan is dat designers zich niets moeten aantrekken van de eigenlijke programmatie. En dat programmeurs zich kunnen concentreren op de ontwikkeling. Natuurlijk is er wel nog de nodige communicatie nodig tussen beide groepen, maar dit alles verloopt veel makkelijker. De gegevens die weergegeven worden zullen via eenvoudige expressies verwerkt worden in de template. Je kan dit gerust vergelijken met het gebruik van de servlet (vgl. php) en de .jsp-pagina (template) bij een Java webapplicatie. Met dit principe had ik in week 4 al kennisgemaakt, door de demo-applicatie van de Smarty-website onder de loep te nemen. Maar doordat er met een framework gewerkt zal worden zal de manier van werken lichtjes aangepast moeten worden. Maar daarover leest u zodadelijk meer.

Een tweede onderdeel is de connectie naar de databank toe. Een MySQL databank in mijn geval. Hiervoor wordt gebruik gemaakt van een package van PEAR. PEAR, PHP Extension and Application Repositor, is net zoals het volledige PHP-concept een open-source community die heel wat handige en bruikbare packages, extensies en bibliotheken aanbiedt. Het package dat gebruikt wordt is het DB_dataobject package. Via classes uit dit package kan op een relatief eenvoudige manier gecommuniceerd worden met de databank. Dit package bevat een SQL builder en een object interface naar de database toe. Er wordt dus met dataobjecten gewerkt. De classes voor elke tabel aanwezig in de databank worden automatisch gegenereerd, en deze classes bevatten ook een aantal default methodes zoals standaard insert(), update(), delete(), find(),… methodes. Verder kunnen er natuurlijk onderaan deze automatisch gegenereerde classes ook nog nieuwe functies toegevoegd worden, waarbij bijvoorbeeld parameters meegegeven kunnen worden en dergelijke.

En het laatste onderdeel is het eigenlijk framework. Dit framework is door Sanmax (Andries Seutens) zelf ontwikkeld. Tegelijkertijd met de ontwikkeling werd ook het Zend-framework gerealiseerd. Het door Andries uitgewerkte framework is vergelijkbaar met dit framework, rekening houdend dat het open-source Zend-framework natuurlijk uitgebreider is. Het framework is gebaseerd op het MVC (Model View Controller)-model. Dit model deelt een applicatie op in drie onderdelen: het datamodel (model), de datapresentatie (view) en de applicatielogica (controller). Deze structuur zal ook terugkomen in de directory-structuur van de webapplicatie. Zo zal er een map genaamd ‘/controller’ zijn, in deze map bevinden zich PHP classes die op events (meestal handelingen van de gebruiker) zullen reageren. In de map ‘/libs/DB’ bevinden zich de classes die communiceren met de database (m.b.v. PEAR, zie vorige alinea). Tot slot zullen in het mapje ‘/templates’ de eigenlijke pagina’s of view geplaatst worden en dat met behulp van Smarty-templates. Om dit framework fatsoenlijk te laten werken moet

er natuurlijk rekening gehouden worden met een aantal regels. Zo moet de in de template-folder per onderdeel een mapje aangemaakt worden dat dezelfde naam heeft als de bijhorende class in de controller-folder. In deze classes worden op hun beurt functies aangemaakt die dezelfde naam hebben als de template-pagina's. Ik zal in mijn stagerapport vanuit een gedetailleerder en een technischer standpunt deze werkwijze toelichten.

Tijdens de tweede week van de paasvakantie werkte ik mijn eerste pagina's uit. Zo zorgde ik er voor dat de klanten beheerd kunnen worden. De gegevens van klanten kunnen toegevoegd, gewijzigd en verwijderd worden. Ook kan er een handig overzicht opgevraagd worden. Kleine moeilijkheden die ik tegenkwam waren het werken met array's in PHP en het teruggeven van data die uit meerdere tabellen kwam. Maar na eenmaal de juiste werkwijze gevonden te hebben is het principe steeds hetzelfde. Ook werkte ik al aan de pagina's voor het beheer van de gebruikers, hiermee bedoel ik de medewerkers die de webapplicatie zullen gebruiken. Ook hier kunnen gebruikers toegevoegd, aangepast of verwijderd worden. Het is ook de bedoeling dat gebruikers aan een groep toegekend kunnen worden, en deze groep zal op zijn beurt rechten krijgen. Op deze manier zal in het laatste stadia, voor de testfase en implementatie, ook de toegankelijkheid van elke gebruiker gecontroleerd en beheerd worden.

De komende weken zullen ook de andere pagina's uitgewerkt worden. Als eerste de project-pagina en daarna de onderliggende taken- en jobs-pagina's. De programmatie hiervan zal iets ingewikkelder zijn aangezien er hier steeds met meer en meer gegevens uit verschillende tabellen rekening gehouden zal moeten worden. Maar aangezien de stageperiode, mede dankzij het doorwerken in de paasvakantie, nog niet in de helft zit, zal dit normaal gezien geen probleem zijn.

8.1.4 Week 08: 16-04 – 27-04

In de voorbije weken hield ik me verder bezig met de ontwikkeling van de webapplicatie. De beheer pagina voor de klanten is al volledig uitgewerkt, echter zullen hier nog aanpassingen aan gebeuren naar gelang de ontwikkeling vordert. Meer hierover leest u in de loop van dit verslag.

De pagina voor het beheer van de gebruikers, dus de medewerkers van Sanmax, werd in de eerste dagen nog wat geoptimaliseerd. Zo voorzag ik de mogelijkheid om via een pagina mails te versturen naar een gebruiker die men selecteert, zodat men niet via een mailclient (zoals MS Outlook of Mozilla Thunderbird) telkens de interne mails moet verzenden.

Een probleem dat ik tegenkwam en dat ik in de toekomst zeker nog moet bekijken, is het verwijderen van gebruikers die gekoppeld zijn aan een groep. Of algemener het verwijderen van objecten die refereren naar andere bestaande objecten. Momenteel heb ik dit opgelost door eerst de objecten te verwijderen waarnaar gerefereerd wordt (met behulp van een where) en dan het eigenlijk object zelf. Echter is hier nog een andere mogelijkheid:, doordat ik bewust gekozen heb voor tabellen van het type InnoDB kunnen er acties voorzien worden die uitgevoerd worden voor refererende sleutels bij een delete (en een update). Het gaat hier dan om de zogenaamde ‘foreign key constraints’. Door aan de refererende sleutel een ON DELETE CASCADE gedeelte toe te voegen kan ik aangeven dat de ‘child’-records mee verwijderd moeten worden. Alternatieven voor het CASCADE gedeelte zijn SET NULL, NO ACTION of RESTRICT. Zoals vermeld werk ik momenteel nog op de ‘makkelijkere’ manier, het resultaat is het zelfde maar de performantie is op deze manier minder. Wanneer ik op het einde de nodige tijd over heb zal ik bovenstaande werkwijze echter zeker implementeren, maar het afleveren van een totaal afgewerkt project is momenteel prioritair.

Zoals op mijn planning stond werkte ik in de tweede week aan de pagina’s voor het projecten en contracten beheer (toevoegen, aanpassen, verwijderen,…). Net zoals bij het klantenbeheer wordt er bij de overzichtpagina van de projecten gewerkt met ‘vcards’. Voor elk project wordt een vcard getoond op de pagina met de nodige informatie. Toch is het niet mogelijk om alle informatie in een oogopslag te tonen omdat dit te veel plaats in beslag zou nemen OF het overzicht eigenlijk geen overzicht meer zou zijn. Daarom zocht ik naar enkele opties. Zo werk ik voor de meest relevante informatie met tooltips. Technisch wordt er gebruikt gemaakt van een javascript dat een DHTML tooltip toont met de nodige informatie in. Om alle informatie te tonen gebruik ik een detailpagina die in pop-up getoond zal worden. Door het aanmaken van de projectenpagina heb ik ook nog een aanpassing doorgevoerd bij het klantenbeheer. Ook voor elke klant kan men een detailpagina laten genereren die dan de voor die klant aangemaakte projecten weergeeft.

En zo kom ik bij het laatste onderdeel terecht dat op de planning stond, het beheer van de contracten. Hier wordt het mogelijk gemaakt om de onderhoudscontracten te beheren van een klant. Een contract wordt dus gekoppeld aan een klant en niet aan een project, zoals tijdens de analyse overeengekomen werd met de geïnterviewde medewerkers. Zo is het mogelijk voor de klant om een contract af te sluiten voor meerdere projecten. Naast het beheergedeelte voor de contracten voor de beheerder zal de nodige contract informatie ook getoond worden op de detailpagina van de klanten.

Het principe en de structuur van een dergelijke webapplicatie heb ik nu zeker en vast door. Maar doordat de webapplicatie tot op een aanzienlijk niveau uitgediept is moet er met steeds meer bovenliggende gegevens (en bijhorende refererende tabellen) rekening gehouden worden en dat is van tijd tot tijd wat zoeken.

Voorlopig zit ik dus nog op schema, al heb ik daarvoor de laatste vrijdag wel wat 'overuren' moeten maken. Dit komt natuurlijk ook deels door de verschillende dagen dat ik niet aanwezig was in de afgelopen weken. Enerzijds door de terugkomdag op school, maar anderzijds door de sollicitatiegesprekken die ik had bij KBC te Leuven. Ondertussen heb ik hier al positieve repliek op gehad en mag ik vrijdag 4 mei een laatste keer teruggaan voor de concrete contractbespreking. Door de vrees dat ik in tijdsgebrek zou komen heb ik er voor gekozen om maandag de brug niet te maken en dus gewoon te gaan werken. Maar nogmaals, voorlopig heb ik me nog steeds aan mijn planning kunnen houden en ik hoop dat ook dit zo blijft.

8.1.5 Week 10: 30-04 – 11-05

Ook in dit activiteitenverslag heel wat informatie over de ontwikkeling van de webapplicatie. De beheerpagina van de taken, job en jobdetails zijn zo goed als afgewerkt.

De programmatie is in de voorbije weken redelijk goed opgeschoten. Zeker wanneer je rekening houdt met het feit dat de functies die momenteel geschreven heel wat complexer zijn dan die van pakweg het klantenbeheer. Er moeten over verschillende tabellen gegevens opgehaald worden en berekend worden. Ter illustratie onderstaande schermafbeelding. U ziet hier een overzicht pagina van de jobs (en hun jobdetails) voor één taak. Naast de vele gegevens die opgehaald moeten worden zoals de project, taak, job, jobdetails en contract gegevens gebeuren er ook tal van berekeningen op deze pagina. Zo wordt de duur van een jobdetail berekend aan de hand van de begin- en eventuele eindtijd. Aan de hand van deze duur wordt een subtotaal berekend per job en uiteindelijk zal de som van deze subtotalen vergeleken worden met de totaal voorziene duur van de taak. Moest deze duur overschreden worden zal ook dit aangegeven worden (voorlopig met een rode vlag, maar dit zal nog vervangen worden door een duidelijkere melding). Zoals u merkt, niet de eenvoudigste pagina, maar hij functioneert momenteel wel hoe het moet. Ook het toevoegen en aanpassen van een job is uitgewerkt. Enkel het toevoegen en aanpassen van jobdetails moet nog wat beter uitgewerkt worden. Dat is voor begin volgende week.

Verder werkte ik ook al aan het log-systeem. Alle gebeurtenissen naar de databank toe worden gelogd. De standaard insert, update en delete-functies worden overschreven in de dataobject class, er wordt zo een functie toegevoegd die het wegschrijven naar de log-tabel verzorgd. De loggegevens worden momenteel 'ruuw' weggeschreven, er is dus enkele kennis voor nodig om te begrijpen wat er juist gebeurd is, indien er nog tijd over is zal dit zeker nog aangepast worden. Het bijhouden van de gegevens gebeurd ombeperkt, al is wel voorzien dat de administrator de tabel met log-gegevens wel op elk moment leegmaken.

Enkele algemene opmerkingen op mijn vorig verslag of algemenere activiteiten die toch het vermelden waard zijn:

Het probleem wat het verwijderen van gegevens betreft die gekoppeld zijn aan andere gegevens is ondertussen ook opgelost. De gekoppelde gegevens worden niet automatisch mee verwijderd zoals ik eerst had willen uitwerken (met ON DELETE CASCADE). Dit zou te extreme gevolgen hebben moest men ooit per ongeluk de verkeerde handeling uitvoeren. Daarom heb ik besloten om het statement ON DELETE NO ACTION 1) toe te voegen. Dit zorgt er voor dat wanneer er gegevens gelinkt zijn er niets zal gebeuren. Er zal echter gecontroleerd worden wanneer deze situatie zich voordoet en de gebruiker zal hiervan op de hoogte gebracht worden. Wil hij de gegevens toch verwijderen, zal hij eerst alle gekoppelde gegevens moeten verwijderen. Het is wel zo dat gegevens bijna nooit verwijderd zullen worden, maar gearchiveerd zullen worden door hun status op ‘done’ te plaatsen. Enkel foutieve of overbodige gegevens zullen daadwerkelijk verwijderd moeten worden.

Ook voor de opmerking van Christof in verband met het niet overtollig gebruik van pop-up’s zocht ik een oplossing. Een trendy en tevens perfect alternatief voor een pop-up is een modal window 2) (javascript). Een modal window is een pop-up die binnen je webpagina zal

verschijnen. De achtergrond zal vervagen en de gegevens worden getoond in een modal window. Voor alle duidelijkheid opent er dus geen extra browser scherm.

Wat staat er nu voor de komende weken nog op het programma ? Nog voldoende ! Allereerst zullen begin volgende week de add- en edit-pagina nog verder uitgewerkt worden voor de jobdetails. Daarna zullen de pagina's voor de rapportage uitgewerkt worden, een zeer belangrijk onderdeel voor Pascal als zaakvoerder. De huidige tool komt vooral op het gebied van rapportage te kort. Ik zal hier omtrent volgende week even kort samen zitten met Pascal om de eisen nog een keertje op een rij te zetten.

Verder moet er ook nog een startpagina voorzien worden waar de gebruiker een handig overzicht zal krijgen met zijn taken, jobs, …. En tot slot, ook belangrijk en niet zo simpel: het inlog- en rechtensysteem dat nog verder uitgewerkt moet worden.

Of er tijd zal over zijn voor de implementatie zal afhangen van de problemen die ik bij bovenstaande taken zal tegenkomen. Er werd me alleszins al gemeld dat ik voor de implementatie eventueel beroep kan doen op andere medewerkers.

Een leuk extra: op donderdag was er een probleem met de server waardoor er intern niet getest kon worden. Ik heb me toen vooral beziggehouden met mijn stagebundel, maar ook de andere medewerkers van Sanmax konden niet echt doorwerken. Er zijn toen een aantal filmpjes gemaakt voor de medewerkers pagina van Sanmax website. Ook mijn collega-stagiair Mindy en ik konden niet ontsnappen aan het oog van de camera. Het resultaat kan u bekijken op http://www.sanmax.be/team_stagiairs.php.

8.1.6 Week 12: 14-05 – 25-05

Het voorlaatste activiteitenverslag en ook dit handelt enkel over de programmatie. In de afgelopen weken ben ik al weer goed opgeschoten en de webapplicatie begint vorm te krijgen. Desondanks zal er in de laatste twee weken nog hard gewerkt moeten worden om de applicatie volledig af te werken zoals gewenst.

Allereerst werden er nog een aantal aanpassingen nodig. Enerzijds omwille van de webapplicatie die Mindy (stagiair Xios) zal uitwerken en waarbij ze gebruik zal maken van mijn tabel Customers. En anderzijds door een aantal bijkomende eisen, nieuwe of nog meer uitgewerkte ideeën die bij mijn stagebegeleider en de zaakvoerder opkwamen. De aanpassingen voor Mindy bleven beperkt. Voor haar was het enkel noodzakelijk dat een klant ook een BTW-nummer toegewezen kon krijgen. Dit was in mijn systeem niet voorzien. Een veld toevoegen aan de database en in de pagina is natuurlijk snel gebeurd, was het niet zo dat ik dit veld ook liefst gevalideerd had. In de validate class van Pear zitten per land een aantal specifieke validatie-functies. Voor België is er zo een VAT-validation voorzien, die via een bepaald algoritme de geldigheid van het BTW-nummer zal controleren. Maar aangezien Sanmax ook enkele buitenlandse klanten heeft en in het buitenland andere formaten gehanteerd worden is deze validatie beperkt. De Europese Commisie biedt echter op zijn website een formulier aan die gebruikt kan worden voor het valideren van BTW-nummers voor BTW-nummers binnen Europa. De webservice, die achter dit formulier zit, opgebouwd via WSDL (Web Service Definition Language) kan echter vrij gebruikt wordt door iedereen. Via een SOAP-service kan deze aangesproken worden. SOAP is een protocol voor het verzenden van het op XML-gebasseerde berichten via het internet. Tot zo ver de technischere uitleg, wat is nu het leuke aan deze service. De functie geeft niet alleen terug of het ingegeven BTW-nummer geldig is, maar zal ook de bedrijfsgegevens (naam en adres) teruggeven. Ik heb deze functie zo geïmplementeerd dat wanneer men een BTW-nummer opgeeft, deze gegevens opgehaald kunnen worden en getoond worden zonder dat deze nog ingegeven moeten worden. Om de pagina niet te laten vernieuwen bij deze actie maak ik gebruik van AJAX (Asynchronous JavaScript and XML) met behulp van de javascripting van script.aculo.us. Ik zal dit onderdeel nog extra in detail bespreken in mijn stagerapport.

De andere aanpassingen kwamen naar boven op woensdag 16 mei. Ik zat toen samen met de zaakvoeder Pascal, mijn stagebegeleidster Lies en webdevelopers Christof en Andries. Ik demonstreerde in het kort de al uitgewerkte onderdelen en kreeg de nodige feedback. Een aantal aanpassingen die doorgevoerd moesten worden werden doorgegeven, alsook een aantal nieuwe en goede ideeën kwamen bovendrijven. Enige vraag die ik, samen met de rest, me toen stelde was of het nog wel haalbaar was al deze ideeën binnen de beperkte termijn van 4 weken in realiteit om te zetten. Er werd dan ook duidelijk aangegeven dat ik zelf moest uitmaken wat haalbaar was en niet, maar men gaf toch ook mee welke functies men toch graag uitgewerkt gezien had. Ander opmerking die ik hierbij, nogmaals, kan geven is dat het nut van het maken van een analyse en in dit geval een goede feedback op deze analyse toch wel belangrijk is. Een aantal aanpassingen die ik moet doorvoeren hadden voorkomen kunnen worden moest de analyse goed doorgenomen geworden zijn. Het toevoegen van velden bij een klant zoals een boekhoudnummer en een intern nummer waarmee de kaften van de klanten genummerd worden had men mij tijdens de analyse kunnen meegeven zodat deze daarin mee opgenomen

konden worden. Ik zal mijn analyse (diagrammen en dergelijke) niet meer gaan aanpassen. Het is uiteraard niet de bedoeling om de analyse aan te passen aan de programmatie. De wijzigingen zal ik wel apart vermelden bij het eigenlijke programmatie-onderdeel.

Een laatste belangrijke aanpassing die nog zal gebeuren is het automatisch controleren en updaten van prioriteiten. Afhankelijk van de duur tot de einddatum van een taak of een project zal de status van de taak of het project aangepast moeten worden. Dit zal gebeuren met behulp van een cronjob. Een cronjob is een unix-commando dat op geplande tijdstippen uitgevoerd zal worden. Zo is het ook mogelijk om PHP code te laten uitvoeren en meer specifiek een update-commando. Afhankelijk van de openstaande duur tot de deadline zal de taak dus mogelijk van prioriteit veranderen. Dit om te voorkomen dat een project toegevoegd wordt met de prioriteit low, steeds onderaan de lijst terecht komt, en men een week voor de deadline ineens opmerkt dat er nog niet begonnen is aan dit project. Ook deze issue zal dus nog weggewerkt worden.

In de afgelopen week verwerkte ik ook het login-systeem en de startpagina in de applicatie. De gebruiker kan inloggen aan de hand van zijn gebruikersnaam en zijn paswoord. Dit paswoord staat versleuteld in de database, maar niet volgens de standaard MD5-functie. Deze keuze is bewust. Via de veilige MD5-functie kan een string gecodeerd worden maar niet gedecodeerd. Omdat er intern gewerkt wordt is deze strenge beveiliging in principe niet noodzakelijk. Zo schrijft men bij Sanmax zelfs voor openbare webapplicaties de paswoorden meestal zonder enige codering weg naar de databank, om zo makkelijk de paswoorden op te kunnen vragen wanneer de gebruiker deze verloren zou zijn. Maar om toch elke gebruiker de mogelijkheid te geven een persoonlijk paswoord te gebruiken dat niet zomaar door iedereen uit de MySQL-database gelezen kan worden koos ik er voor om het paswoord toch te coderen. Dit met een functie die het ook mogelijk maakt het paswoord te decoderen. Het is namelijk zo dat een gebruiker die zijn paswoord vergeten is via het email adres dat gekend is bij het systeem zijn gegevens terug kan opvragen. Het paswoord wordt dan gedecodeerd en via mail naar de gebruiker verzonden. Zo moet of kan de administrator ook niet tussenkomen in het paswoord beheer. Wanneer de administrator een gebruiker toevoegt zal hij natuurlijk een paswoord instellen voor de gebruiker. Al is deze in staat om het wachtwoord op elk moment te wijzigen.

Wanneer de gebruiker ingelogd is zal er een sessie aangemaakt worden die onder andere het gebruikers-id zal bewaren gedurende de sessie. Aan de hand van dit id zal specifieke informatie (taken, jobs,…) voor deze gebruiker uit de database gefilterd worden. Zo zal de gebruiker bijvoorbeeld in het takenoverzicht enkel zijn taken terugvinden, wat logisch is. Ook iedere gebruiker komt na het inloggen op een gespecialiseerde startpagina terecht, voorzien van enkele leuke snufjes.

In de laatste twee weken zal nog hard gezwoegd moeten worden. Wat er sowieso nog door moet is de rapportering voor de zaakvoerder, met eventueel de mogelijkheid om deze te exporteren naar een pdf-formaat zodat deze ook makkelijker bewaard en/of afgeprint kunnen worden. En last but not least, het rechtensysteem. Dat zal bepalen wat een gebruiker allemaal kan. Voorlopig is het zo dat iedereen die inlogt alle functies in het menu ter beschikking krijgt, zoals het toevoegen van projecten, klanten en gebruikers,… terwijl deze eigenlijk alleen door een beheerder uitgevoerd zouden mogen worden.

8.1.7 Week 14: 29-05 – 08-06

Bij aanvang van de stage leken die 3 maanden wel eens lang te kunnen duren, maar ze zijn in feite omgevlogen. Bij deze dan ook het laatste activiteitenverslag. Voor de laatste twee weken stond de rapportering en het implementeren van het rechtensysteem nog op de planning.

Wat de rapportering betreft heb ik mij door tijdsgebrek en in overleg met mijn stagebegeleider beperkt tot een HTML rapport. Aan het exporteren naar een pdf-formaat ben ik dus niet meer gekomen. Echter heb ik getracht het HTML rapport zo te laten genereren en enkele functies toe te voegen zodat de rapportering heel gedetailleerd kan gebeuren en ook de mogelijkheden wat printen betreft uitgebreid zijn. Diegene die het rapport gaat genereren heeft verschillende mogelijkheden en kan heel wat gegevens specificeren. Zo kan men een rapport genereren op basis van een klant (en dus alle projecten van een klant), op basis van één project of op basis van een medewerker (alle projecten waar de medewerker aan mee werkt). Men kan ook kiezen welke gegevens men allemaal wilt tonen (zoals: medewerkers, contracten, tijdsduur,…) en tot op welk niveau (tasks, jobs, jobdetails). Dankzij het gebruik van een extra stylesheet (.css) die aangeroepen zal worden wanneer men deze pagina wilt printen, kan ik een afgeprint rapport een eigen lay-out toewijzen. Een ander voordeel is dat men bijvoorbeeld het menu op de website zelfs niet mee zal afprinten, aangezien ik dit in de stylesheet verberg. Verder heb ik ook het logo van Sanmax toegevoegd bovenaan het rapport. Wanneer men nu een pdf-printer zou installeren op de computer kan men perfect deze HTML pagina afprinten in pdf-formaat en zo eventueel aan de klant bezorgen of bijhouden in het archief. Al is het natuurlijk ook helemaal geen probleem om een HTML pagina op te slaan. Een ander leuk extra is dat alle parameters in de URL van het rapport getoond worden, zo kan men de URL van gegenereerd rapport perfect opslaan als favorieten in de browser en dit later opnieuw laten genereren en opvragen met identiek dezelfde specificaties en eventueel aangepaste gegevens.

Het andere grote onderdeel wat nog uitgewerkt moest worden was het rechtensysteem. Het is immers de bedoeling dat een medewerker die tot de groep ‘developer’ of ‘stagiair’ behoord niet hetzelfde kan als een medewerkers die opgenomen is in de groep ‘admin’. Het is met andere woorden de bedoeling om rechten toe te wijzen aan den groep. Na heel wat wikken en wegen en overleg met Andries hebben we besloten om gebruik te maken van een .ini-bestand waarin we alle mogelijke rechten eenmalig gaan specifieren. Een recht is opgebouwd uit de naam van de controller en de bijhorende actie (= een structuur aangenomen in het framework, wordt in detail behandeld in het stagerapport). Dit ini-bestand zal dan opgevraagd worden bij het uitlezen van de rechten. Bij het toevoegen of wijzigen van een groep kan men dan aanvinken over welke rechten men beschikt, wat dan op zijn beurt weer bijgehouden wordt in de databank. Bij het inloggen worden dan de rechten waarover een gebruiker beschikt uit de database opgehaald en weggeschreven als een array in de sessie van die gebruiker. Bij elke oproep van een controller en bijhorende action, of simpel weggezegd een bepaalde pagina, zal er gecontroleerd worden of men over de rechten beschikt om deze pagina daadwerkelijk op te roepen. Voorts zal ook op basis van deze rechten het menu opgebouwd worden en bepaalde knoppen of links al dan niet getoond worden. Wanneer een gebruiker tot meerdere groepen behoort, zal de gebruiker beschikken over de som van deze rechten, wat dus betekend dat hij over alle rechten zal beschikken die er aangegeven zijn, ook al zou hij tot tweede groep behoren met heel wat minder rechten.

Om bovenstaande uitleg van het rechten systeem even te verduidelijken vergelijk ik de taken overzicht pagina's van medewerkers die behoren tot de groep 'stagiair', 'developer' en 'admin'.

Stagiair:
Deze medewerker zal in het takenoverzicht enkel zijn taken te zien krijgen en enkel binnen deze taken kunnen zoeken. Omdat het hier over een stagiaire gaat kan hij zijn taken ook niet aanpassen, hij kan enkel op het pijltje (uiterst rechts klikken) om jobs en jobdetails onder een taak te boeken. De taken zelf worden door de admin voor hem gegenereerd. Let ook op de beperkte functies waarover de stagiaire beschikt.

Developer:
Deze medewerker krijgt via het takenoverzicht ook enkel zijn taken te zien maar kan deze wel aanpassen of verwijderen. Hij/zij kan ook taken toevoegen via de knop 'Taak toevoegen' rechtsboven. Verder beschikt deze medewerker over een extra beheerfunctie in zijn menu, namelijk het genereren en opvragen van rapporten.

Admin:
Deze medewerker, of beter administrator, kan via het takenoverzicht en meer bepaald via de zoekbalk bovenaan alle taken opvragen van eender welke medewerker. Standaard krijgt ze ook enkel zijn/haar taken te zien. Ze kan ook taken aanpassen of verwijderen. Het is ook duidelijk dat een admin beschikt over heel wat meer functies dan dat een gewone medewerker doet.

Gedurende de laatste week hield mijn stagebegeleidster zich bezig met het ingeven van klanten, projecten en taken in het systeem. Regelmatig werden er nog een aantal aanpassingen doorgevoerd om het gebruik nog makkelijker te maken of kleine bugs die er uit gehaald werden. Ook gedurende de laatste week stelde ik mijn systeem nog een keer voor aan alle medewerkers via een demo. En dit vooral met het oog op de functionaliteit die zij intensief zullen gebruiken. Zo toonde ik hoe men oa. gegevens moest toevoegen aan het systeem, hoe met een jobdetail moet afsluiten, enzovoorts. De reacties waren heel positief en niemand had echt grote opmerkingen bij het systeem.

Op donderdag werd mijn stagebeoordelingsformulier ingevuld door mijn stagebegeleidster en hebben we dit kort overlopen. Ook hier niets dan positiefs. In de kwalitatieve evaluatie schrijft men: "Het intranet dat Davy heeft ontwikkeld is veel gebruikersvriendelijker dan het intranet waar Sanmax voordien mee werkte. Werkelijk een plezier om mee te werken ! Voor Sanmax is deze stagiair een meerwaarde geweest!". Zeer leuk om te horen en te weten dat het doorwerken in onder andere de paasvakantie en een x-aantal overuren gedurende de laatste weken, tot soms zelfs half 7 's avonds, ook daadwerkelijk geloond hebben. Het is ook mogelijk dat ik in de komende dagen nog binnenspring bij Sanmax om hier en daar eventueel wat bug-fixing te doen.

De laatste werkdag bedankte ik mijn collega's van Sanmax, maar ook van de andere bedrijven die gebruiken maken van dezelfde accommodatie, met een kleine traktatie. Ik had ook een afrondend gesprek met zaakvoerder Pascal D'Helft die me graag na mijn stage bij Sanmax aan het werk had gezien en het toch wel spijtig vond dat ik vanaf september bij KBC aan de slag zal gaan. "Moest het daar niet naar je zin zijn, mag je altijd terug langskomen.", liet hij mij weten.

Ook de zaakvoerder Luc Vannitsen van Offitel bedankte mij. Ik heb namelijk tijdens de pauzes ook voor hun gezocht naar een aantal oplossingen voor problemen in o.a. hun telefoniesoftware. Hij drong er zelfs op aan om ook dit in mijn stagebundel te vermelden.

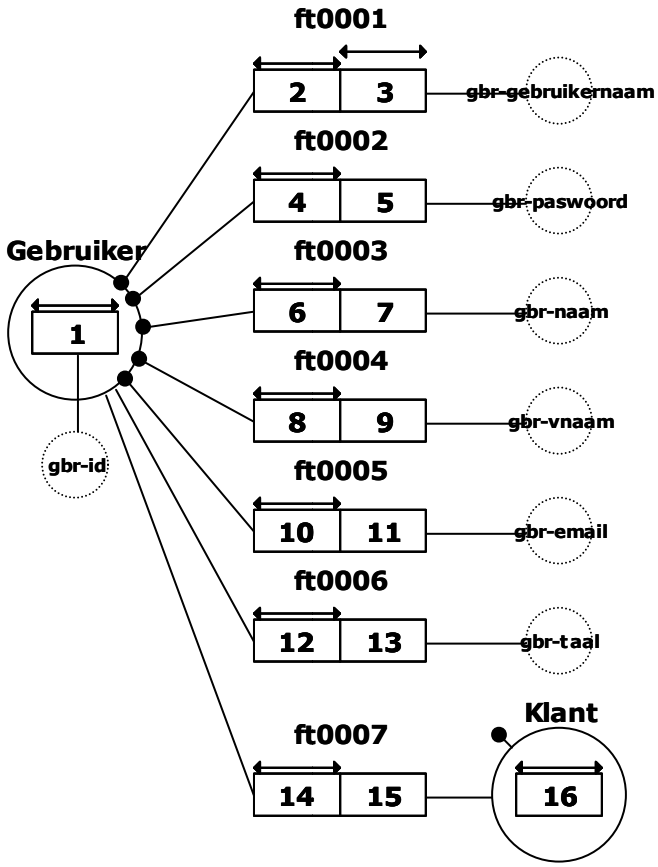
8.2 Analyse

8.2.1 Informatieanalyse

8.2.1.1 Informatie Grammatica Diagrammen

Allereerst zal er in de webapplicatie ingelogd moeten worden. Dit niet enkel om veiligheidsredenen maar ook omdat niet alle gebruikers dezelfde mogelijkheden zullen hebben binnen de applicatie. De gebruiker zal naast een unieke nummer, ook een unieke aanmeldnaam toegewezen krijgen met bijhorend persoonlijk paswoord. Naast deze gegevens wordt ook een naam, voornaam en e-mail adres bijgehouden voor elke gebruiker. Optioneel is er de mogelijkheid om een voorkeurstaal aan een gebruiker toe te wijzen. Dit kan nuttig zijn wanneer de applicatie in meerdere talen beschikbaar zal zijn. Tot slot wordt ook voorzien dat een gebruiker gekoppeld kan worden aan een klant. De klant zal ook dan moeten kunnen inloggen met gebruikersgegevens. Deze functionaliteit valt echter buiten de grenzen van dit project maar wordt voorzien wanneer men, medewerkers maar dus ook klanten, deze applicatie extern zou kunnen benaderen.

Gebruiker:
Gebruiker 010
Gebruiker 010 heeft als gebruikersnaam “Davy”
Gebruiker 010 heeft als paswoord “*5B27BB68A8CF550F086404732865CEF9194E7BD2”
Gebruiker 010 heeft als naam “Rego”
Gebruiker 010 heeft als voornaam “Davy”
Gebruiker 010 heeft als email “davy@sanmax.be”
Gebruiker 010 heeft als voorkeurstaal “nl”
Gebruiker 010 is Klant null

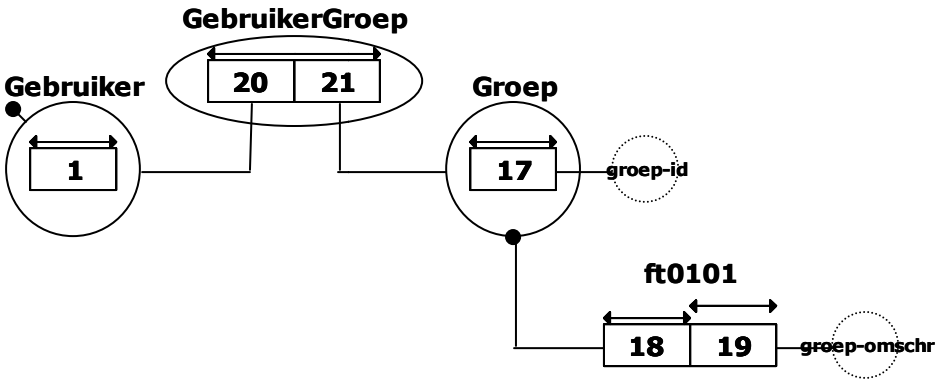


Een gebruiker behoort tot een groep. Deze groep zal de functie van de gebruikers specificeren. Voorbeelden van groepsomschrijvingen kunnen zijn: webdeveloper, design, html, admin,… Het is mogelijk dat een gebruiker tot meerdere groepen behoord. Zo kan bijvoorbeeld de medewerker verantwoordelijk voor het design ook tot de groep van administrators behoren, om zo de zaakvoerder bij te staan in het beheer van de klanten en projecten.

Dat een gebruiker in een groep terecht komt heeft ook nog een andere reden. Per groep worden er namelijk een aantal rechten toegekend. Er zullen heel wat rechten gespecificeerd worden. Enkele voorbeelden van rechten om u een idee te geven: task.add, project.delete, user.edit,…. Een recht heeft ook een veld ‘alert’. Dit geeft wanneer het op true staat aan dat er een alert of bevestiging getoond moet worden wanneer dit recht opgeroepen zal worden. (Vb.: bij het recht user.delete zal alert op true staan, de gebruiker zal immers moeten bevestigen dat men de gebruiker daadwerkelijk wil verwijderen.) Doordat een gebruiker tot meerdere groepen kan behoren zou er ook de mogelijkheid zijn dat bepaalde rechten elkaar overrulen. Er zullen daarom enkel rechten toegekend worden die toegepast zullen worden (allow), en geen ontkennende rechten (deny), zodat er niet gewerkt moet worden met prioriteiten. Dus, ofwel heeft een gebruiker een recht toegekend en kan hij de functie uitvoeren, ofwel heeft de gebruiker het recht gewoon niet.

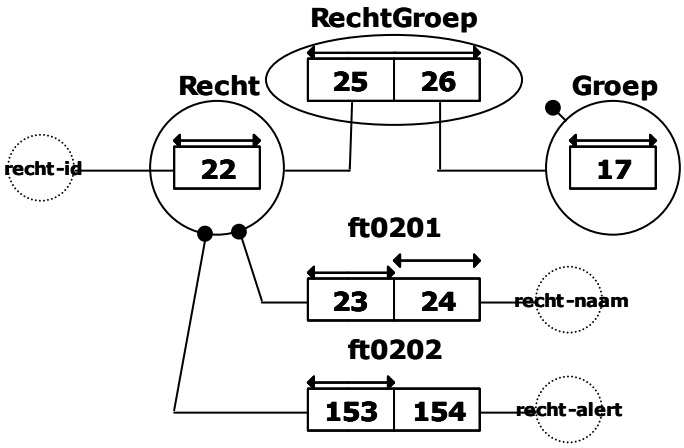
Groep:
Groep 03
Groep 03 heeft als omschrijving “webdeveloper”

GebruikerGroep:
Gebruiker 010 is gekoppeld aan Groep 03



Rechten:
Recht 043
Recht 043 heeft als naam “task.add”
Recht 043 heeft een alert false

RechtGroep:
Recht 043 is gekoppeld aan Groep 03

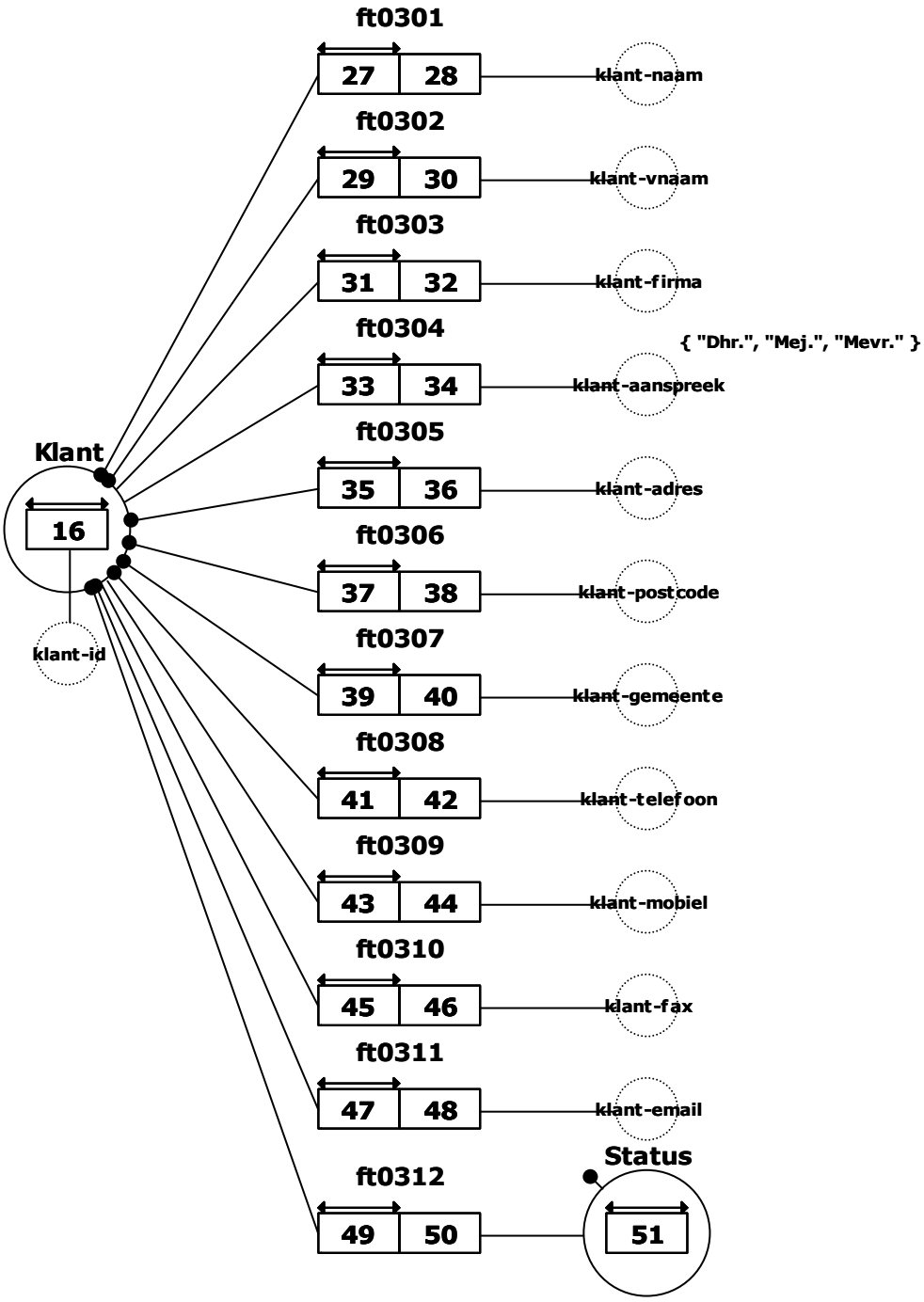


De belangrijkste doelstelling van deze applicatie is het beheer van de projecten die uitgevoerd worden voor klanten. Het is logisch dat er heel wat klantgegevens bijgehouden zullen moeten worden. Elke klant krijgt een unieke nummer. Een klant kan een particulier zijn, maar ook een verantwoordelijke of afgevaardigde van een firma. Wanneer een contactpersoon meerdere firma's zou vertegenwoordigen zullen deze gegevens telkens als een nieuwe klant weggeschreven worden. Een bedrijf zou ook verschillende contactpersonen kunnen hebben voor bijvoorbeeld verschillende projecten. Momenteel zaten er al dergelijke gegevens in het klantenbestand van Sanmax, dus deze veronderstelling is niet onrealistisch en er moet dus zeker rekening mee gehouden worden. Zo komt de firma 'KRC Genk' verschillende keren voor in het bestand, maar wel telkens met een ander contactpersoon.

Er worden heel wat contactgegevens bijgehouden, zo kan men de klant indien nodig ook snel contacteren, via telefoon, mail of zelfs via briefwisseling. Daarom dat ook een aanspreektitel bijgehouden zal worden. Een aantal andere contactgegevens zijn optioneel, zoals een mobiel telefoonnummer, een fax,….

Een klant kan ook een bepaalde status bevatten. Standaard zal een klant actief zijn, maar het is mogelijk dat alle projecten voor een bepaalde klant afgewerkt is. De klant zal dan niet verwijderd worden, maar op afgewerkt geplaatst worden. In het huidige systeem werd er als tussenoplossing een A geplaatst voor het klant-id, het werken met een apart veld lijkt mij een logischere oplossing.

Klant:
Klant 0043
Klant 0043 heeft als naam "Janssens"
Klant 0043 heeft als voornaam "Peter"
Klant 0043 is van de firma "Argipel"
Klant 0043 heeft als aanspreektitel "Dhr."
Klant 0043 heeft als adres "Steenweg 430"
Klant 0043 heeft als postcode "3600"
Klant 0043 heeft als gemeente "Genk"
Klant 0043 heeft als telefoon "012345678"
Klant 0043 heeft als gsm "0456789012"
Klant 0043 heeft als fax "012345679"
Klant 0043 heeft als email "janssens.peter@argipel.be"
Klant 0043 heeft als Status 04

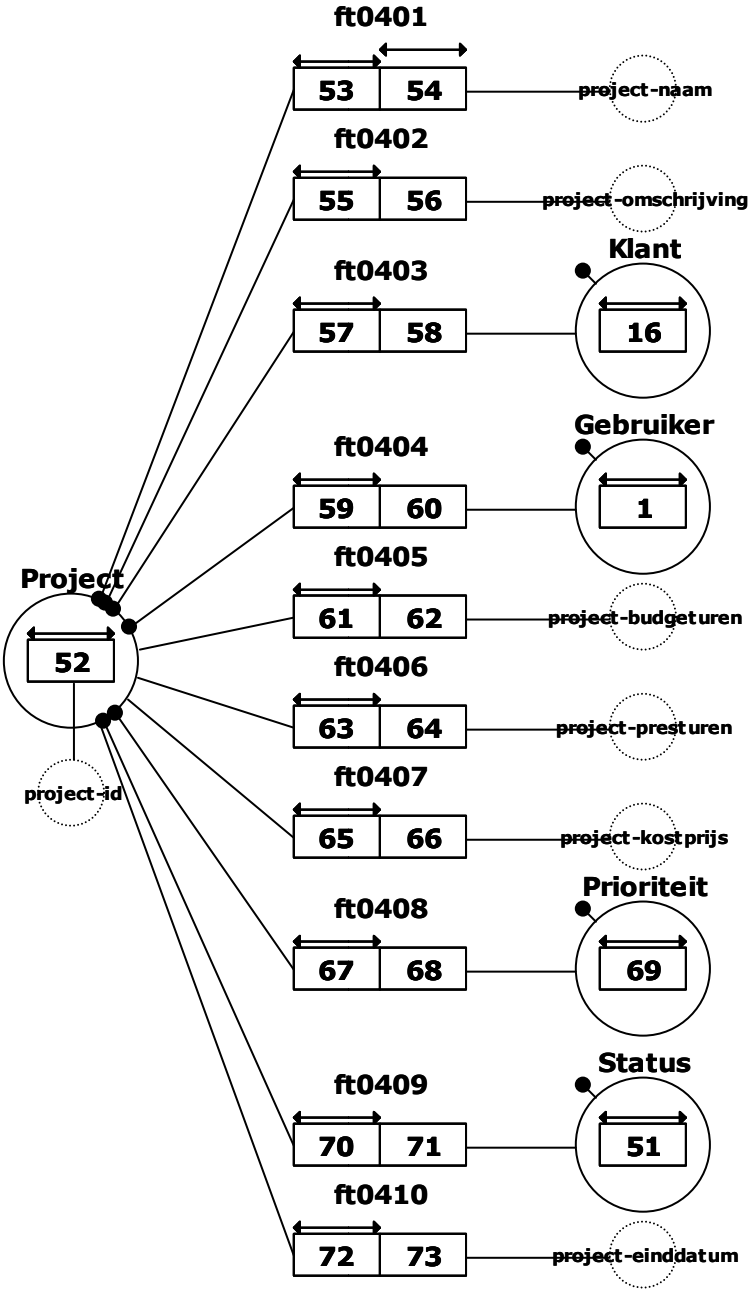

ft0301
27
28
klant-naam
ft0302
29
30
klant-vnaam
ft0303
31
32
klant-firma
ft0304
{ "Dhr.", "Mej.", "Mevr." }
33
34
klant-aanspreek
ft0305
35
36
klant-adres
ft0306
37
38
klant-postcode
ft0307
39
40
klant-gemeente
ft0308
41
42
klant-telefoon
ft0309
43
44
klant-mobiel
ft0310
45
46
klant-fax
ft0311
47
48
klant-email
ft0312
49
50
Klant
16
klant-id
Status
51

Een van de andere belangrijke onderdelen is het beheer van de projecten. Een project heeft een unieke nummer en naam. Twee projecten met dezelfde naam zijn dus logischerwijze niet mogelijk. Naast een naam zal een project ook een duidelijke omschrijving bevatten. Een project zal aan een klant gekoppeld worden, het is mogelijk dat een klant één of meerdere projecten heeft. Daarnaast zal een project ook gekoppeld worden aan een gebruiker, een medewerker van Sanmax die verantwoordelijk zal zijn voor dit project.

Het financiële aspect van een project was niet noodzakelijk, maar wordt wel al voorzien voor de toekomst. Zo kan een budget in uren opgegeven worden, en zal ook het aantal gepresteerde uren bijgehouden worden. U kunt stellen dat dit berekend kan worden aan de hand van onderliggende gegevens van de tijdsduur van taken, maar er wordt hier toch een apart veld voorzien. Zo kan men eventueel fictieve gepresteerde uren opgeven in plaats van enkel en alleen de mogelijkheid te hebben om ze te automatisch te laten berekenen. Ook een kostprijs per uur zal gedefinieerd worden, omdat dit per project kan verschillen of door de jaren heen kan veranderen.

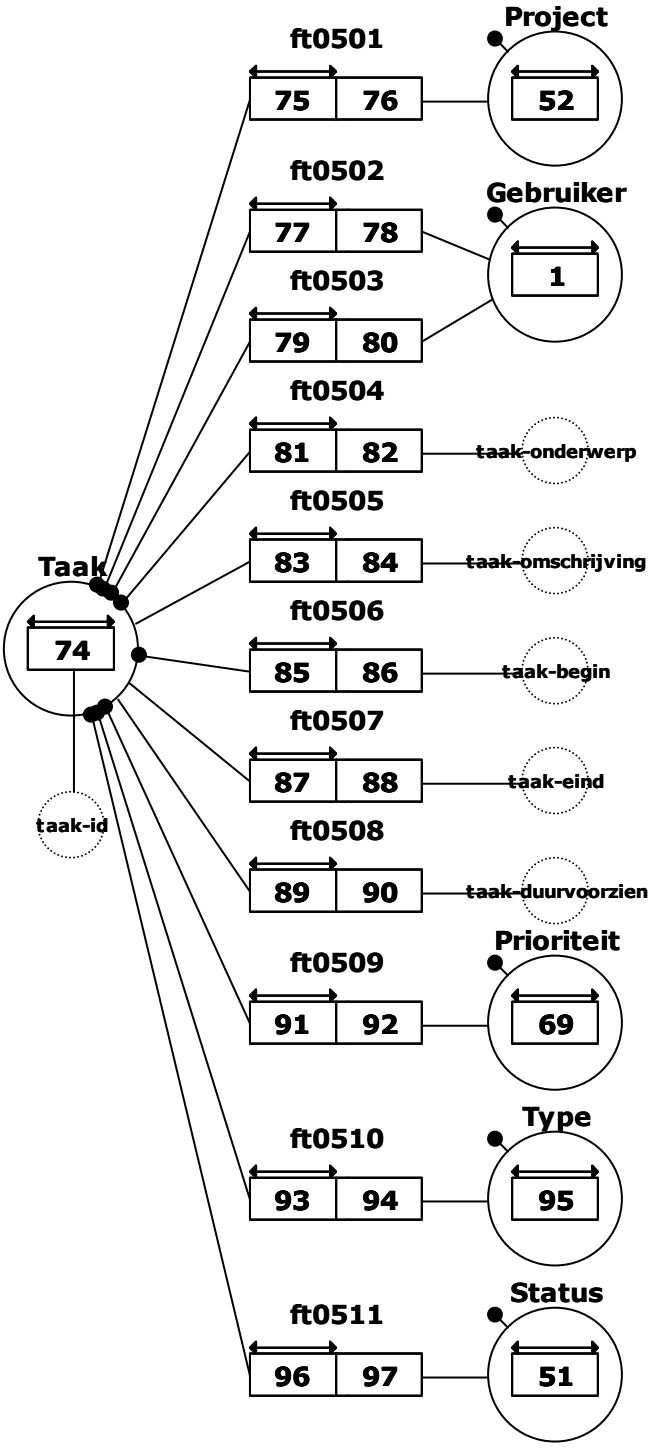
Verder heeft een project ook een prioriteit zodat men weet welk project voorrang moet krijgen op andere projecten. Een project kan ook in een bepaalde status zijn. Afhankelijk van de status zal softwarematig bepaald worden wat de gebruiker kan met die project. Wanneer het project nog niet gestart is zal men geen taken kunnen toevoegen, wanneer alle taken afgewerkt zijn zal het project kunnen voltooid worden,…. Wanneer een project de status afgewerkt krijgt, zal er een einddatum bijgehouden worden.

Project:
Project 0037
Project 0037 heeft als naam “Website Argipel”
Project 0037 heeft als omschrijving “Design en ontwikkeling van dynamische website”
Project 0037 wordt uitgewerkt voor Klant 00043
Project 0037 werd aangemaakt door Gebruiker 002
Project 0037 heeft als budget 80 uren
Project 0037 heeft als gepresteerde uren 75 uren
Project 0037 heeft als kostprijs per uur 17,5 EUR
Project 0037 heeft als Prioriteit 03
Project 0037 heeft als Status 04
Project 0037 werd afgewerkt op 2007-02-29 11:35:45


ft0401
53
54
project-naam
ft0402
55
56
project-omschrijving
Klant
ft0403
57
58
16
Gebruiker
ft0404
59
60
1
ft0405
Project
61
62
project-budgeturen
52
ft0406
63
64
project-presturen
project-id
ft0407
65
66
project-kostprijs
Prioriteit
ft0408
67
68
69
Status
ft0409
70
71
51
ft0410
72
73
project-einddatum

Een taak is een onderdeel van één project. Een project kan logischerwijze meerdere taken bevatten. Een taak wordt aangemaakt door een gebruiker, maar deze gebruiker kan verschillende zijn van de gebruiker die de taak zal uitwerken. Zo is het perfect mogelijk dat de zaakvoerder een taak aangemaakt voor een stagiair. Een taak heeft een onderwerp en kan eventueel een bijkomende omschrijving hebben. Verder kan een bepaalde deadline toegekend worden en de voorziene uren dat de medewerker aan deze taak zou mogen werken. Ook een taak heeft een prioriteit, die onafhankelijk is van de prioriteit van het project. Daarnaast heeft een taak ook een status en een type.

Taak:
Taak 00234
Taak 00234 is een onderdeel van Project 0037
Taak 00234 aangemaakt voor Gebruiker 002
Taak 00234 aangemaakt door Gebruiker 002
Taak 00234 heeft als onderwerp “ontwerp design”
Taak 00234 heeft als omschrijving “uittekenen van ontwerp voor lay-out website”
Taak 00234 werd aangemaakt op 2007-01-05 16:23:45
Taak 00234 heeft als deadline 2007-01-07 12:00:00
Taak 00234 heeft als voorziene uren 5
Taak 00234 heeft Prioriteit 02
Taak 00234 is van het Type 01
Taak 00234 heeft als Status 04


Project
ft0501
75
76
52
ft0502
Gebruiker
77
78
1
ft0503
79
80
ft0504
81
82
taak-onderwerp
ft0505
Taak
83
84
taak-omschrijving
ft0506
74
85
86
taak-begin
ft0507
87
88
taak-eind
taak-id
ft0508
89
90
taak-duurvoorzien
Prioriteit
ft0509
91
92
69
Type
ft0510
93
94
95
Status
ft0511
96
97
51

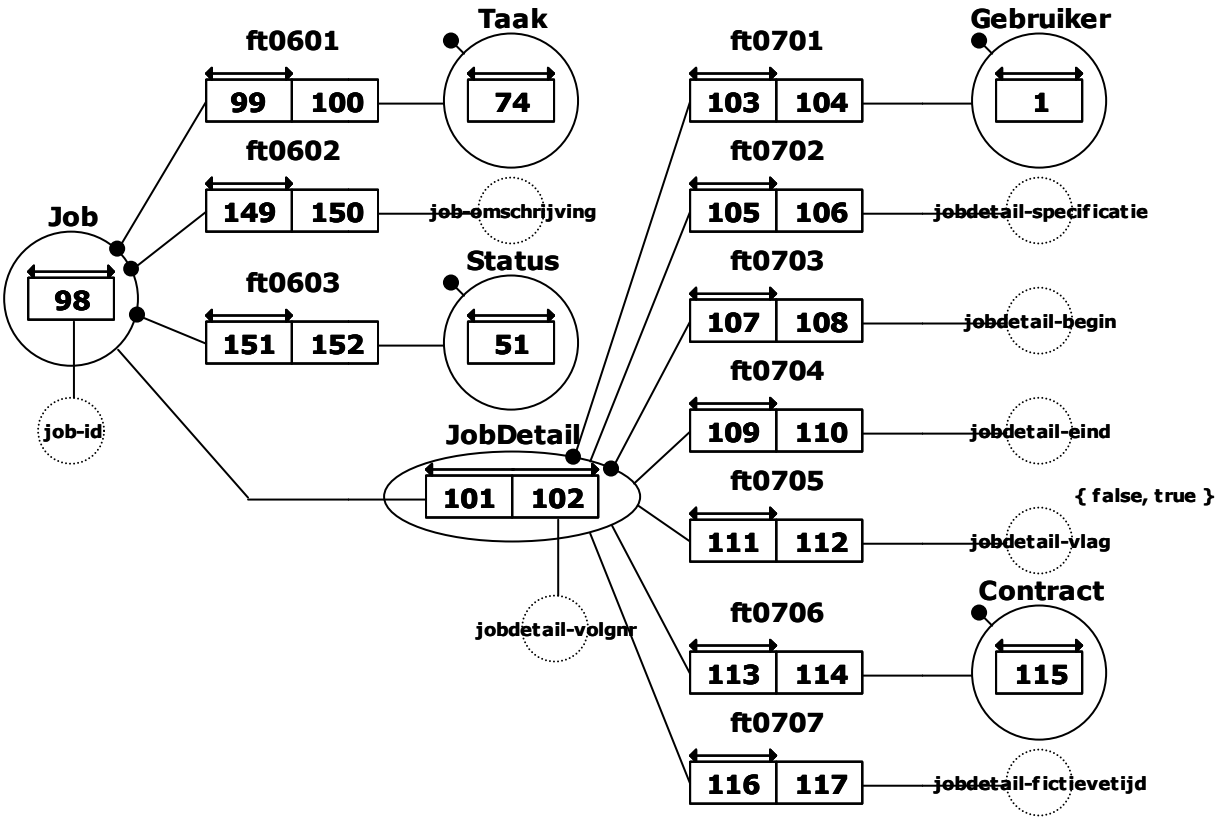
Een taak kan op zijn beurt opgedeeld worden in één of meerdere jobs. Wanneer een medewerker een volledige taak of een onderdeel van aan taak afwerkt is hij bezig met een job. In de huidige situatie wordt een job aangemaakt voor een taak, voor één gebruiker en voor één tijdstip. Wanneer de gebruiker langer dan één dag aan een job werkt, zal hij voor elk moment een nieuwe job moeten aanmaken. Met het toekomstig systeem is dit niet nodig, een gebruiker zal 's anderendaags aan een job kunnen verder werken. Sterker nog, er kunnen meerdere gebruikers aan een job werken. Een job is dus gekoppeld aan een taak en heeft een algemene beschrijving. Een job heeft ook een status.

Aan een job kan op verschillende momenten en/of door verschillende gebruikers gewerkt worden, dit is wat we een jobdetail (= onderdeel van een job) noemen. Men kan indien gewenst voor elke deel-job (jobdetail) het uitgevoerde werk nog specificeren, al is dit niet noodzakelijk. Verder heeft een job een begin en, wanneer deze beëindigt is, een eindtijd. Een gebruiker kan een deel-job vlaggen. Dit betekend dat die bepaalde job bij hem extra gemarkeerd zal worden, een geheugensteuntje.

Momenteel was er al een webapplicatie ontwikkeld voor de onderhoudscontracten. Dit zijn contracten die Sanmax afsluit per klant. De klant koopt in principe een aantal werkuren aan, waarin voor hun gewerkt zal worden. Dat kan gaan van updaten van de website, een onderhoud tot een restyling… Aangezien deze webapplicatie geïmplementeerd moet worden voorzien we dat een gebruiker kan aangeven dat het bepaalde werk waar hij mee bezig is zal afgehouden worden van de uren in het onderhoudscontract. Hier kan men zich baseren op het berekenen van de tijd aan de hand van de begin en eindtijd, maar men kan ook zelf een fictieve tijd opgeven die aangerekend zal worden. Dit zal zeker niet bij iedere job het geval zijn, let dus goed op, het gaat hier om een optionele functionaliteit.

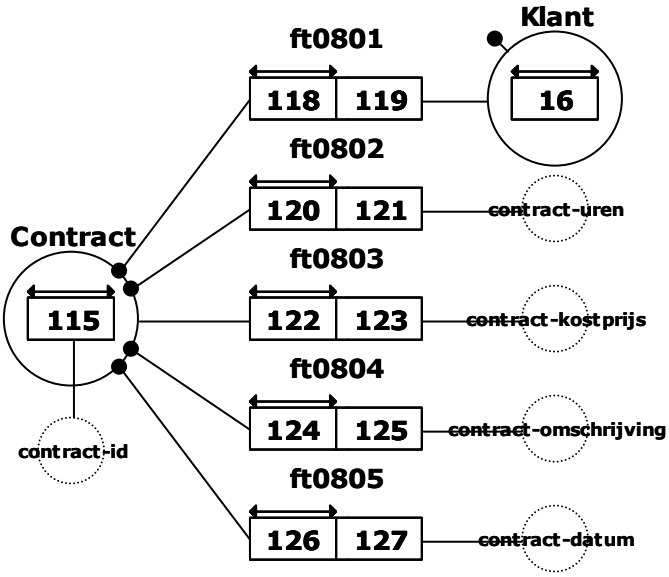
Job:
Job 00343
Job 00343 werkt aan Taak 00234
Job 00343 heeft als omschrijving "eerste voorontwerp design"
Job 00343 heeft als Status 04

JobDetail:
Job 00343 met volgnummer 01
Job 00343 met volgnummer 01 wordt uitgevoerd door Gebruiker 002
Job 00343 met volgnummer 01 heeft als specificatie "aanmaken header en footer"
Job 00343 met volgnummer 01 is gestart op 2007-01-06 08:30:00
Job 00343 met volgnummer 01 is gestopt op 2007-01-06 12:00:00
Job 00343 met volgnummer 01 is gevlagd true
Job 00343 met volgnummer 01 wordt opgenomen in het Contract 00000000004
Job 00343 met volgnummer 01 heeft als fictieve tijd 4 uur



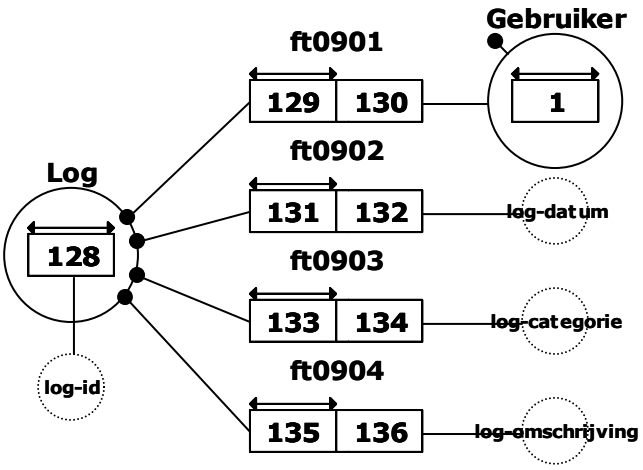
Zoals als uitgelegd kan een klant een onderhoudscontract afsluiten. Dit contract houdt in dat men een bepaald aantal uren 'aankoopt' waarin de medewerkers van Sanmax aan de website kunnen werken. Naast de gegevens van de klant en het budget van het contract in uren, bevat dit ook de kostprijs per uur. Ook hier weer, dit is een financiële informatie die momenteel nog niet verplicht is, maar wel al voorzien wordt naar de toekomst toe.Verder heeft een contract een korte beschrijving en een datum waarop het contract is aangemaakt.

Contract:
Contract 00000000004
Contract 00000000004 is opgesteld voor Klant 0043
Contract 00000000004 bevat 50 uren
Contract 00000000004 heeft een kostprijs per uur van 15 EUR
Contract 00000000004 heeft als omschrijving "onderhoud website"
Contract 00000000004 is opgemaakt op 2007-01-07 13:02:43



Alle handelingen die zullen plaats hebben op het systeem zullen worden gelogd. Een gebeurtenis achter een log zal worden aangeroepen door een actie van een gebruiker, vanzelfsprekend wordt deze gebruiker bijgehouden. Zodat men bij eventueel misbruik of bij grote vergissingen kan nagaan bij welke gebruiker het misgelopen is. Zoals in elk logsysteem wordt natuurlijk de datum bijgehouden. Een log zal tot een bepaalde categorie behoren. De meeste logs zullen informatief zijn, en zal dan een omschrijving hebben zoals: “Taak: aanpassen taak #”, “Gebruiker: toevoegen gebruiker #’,…. Een andere mogelijke categorie zou zijn ‘ERROR’, waarbij logs worden bijgehouden wanneer er een fout tegengekomen wordt, en dit als feedback naar de ontwikkelaar of onderhouder van de applicatie toe.

Log:
Log 04384
Log 04384 heeft als Gebruiker 003
Log 04384 werd weggeschreven op 2007-01-05 16:23:34
Log 04384 heeft van categorie “INFO”
Log 04384 heeft als omschrijving “TAAK: Toevoegen Taak 00234”



Zoals al vermeld heeft iedere taak een type. Voorbeelden van taak-types zijn: nieuw, controle, aanpassing,…. Elke type kan slecht één keer voorkomen en krijgt een specifieke kleur toegekend. Nieuwe taken zullen zo bijvoorbeeld met het woord 'nieuw' en in het groen aangegeven worden.

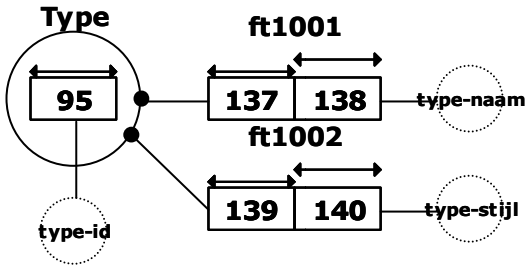
De status geeft de vorderingen van de verschillende gekoppelde onderdelen weer. Zo heeft elke klant, elk project, elke taak en elke job een status. Men kan zo kijken op de vier verschillende niveaus of men al begonnen is het met het werken aan een bepaald onderdeel, of men nog moet beginnen en of dat het onderdeel al afgewerkt is. Ook hier heeft iedere status een unieke naam en kleur.

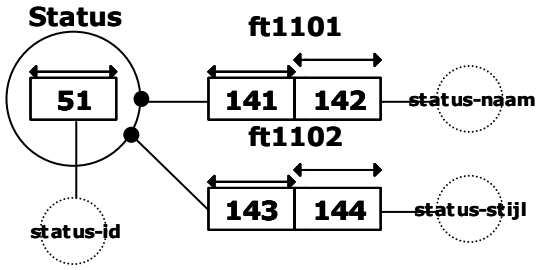
Tot slot kan aan een project en aan een taak ook nog een prioriteit toegewezen worden. Zodat de medewerkers snel kunnen zien welke onderdelen prioritair afgewerkt moeten worden. De verschillende mogelijke prioriteiten kunnen zijn: laag, gemiddelde, hoog en heel hoog, en deze ook allemaal met een specifieke bijhorende kleur.

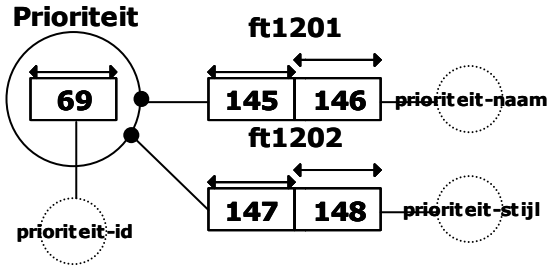
Type:
Type 01
Type 01 heeft als naam "nieuw"
Type 01 heeft als stijl "green"

Status:
Status 04
Status 04 heeft als naam "afgewerkt"
Status 04 heeft als stijl "green"

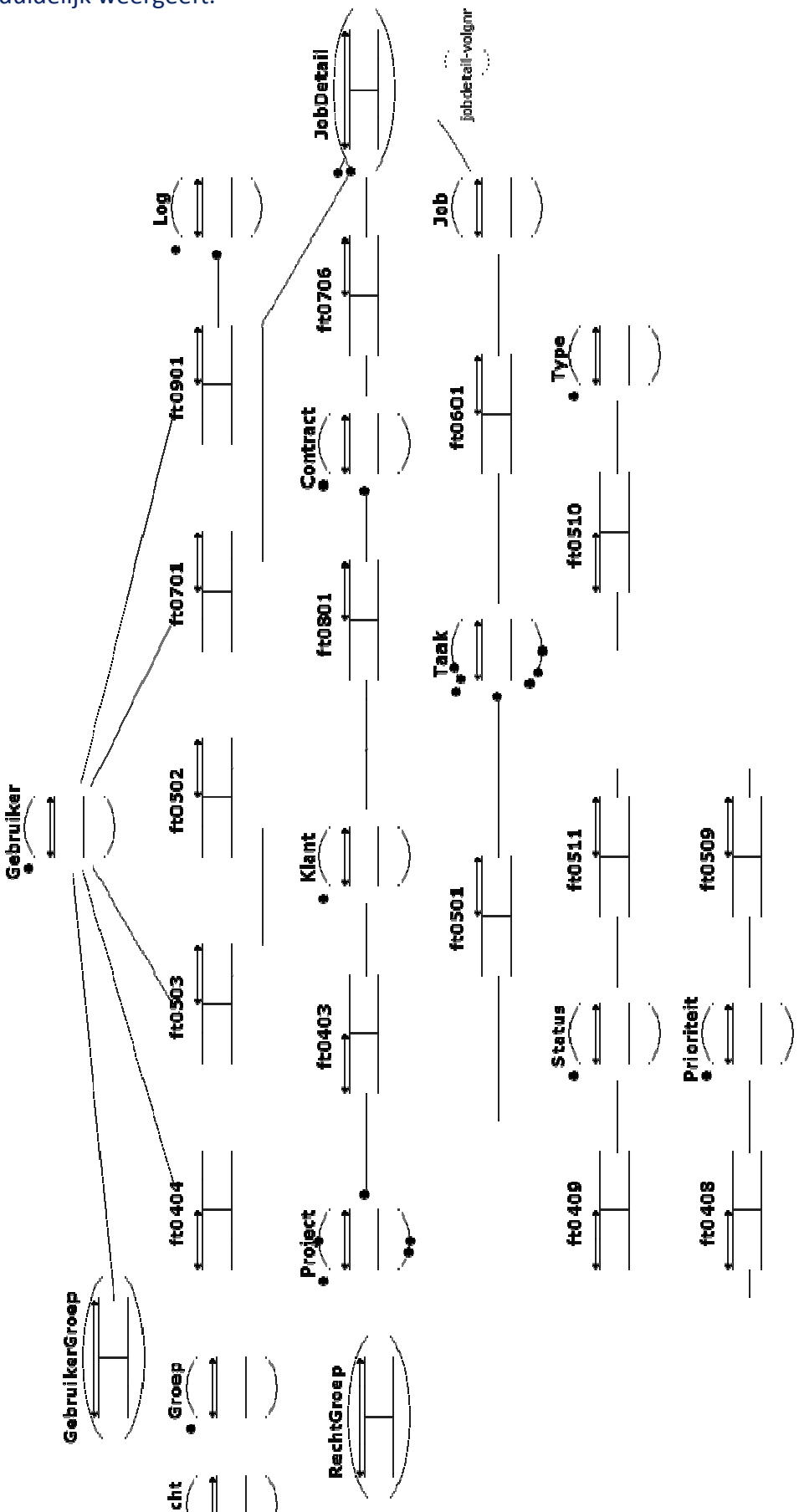
Prioriteit:
Prioriteit 03
Prioriteit 03 heeft als naam "hoog"
Prioriteit 03 heeft als stijl "orange"







Tot slot het volledige Informatie Grammatica Diagram dat de samenhang tussen de onderdelen nogmaals duidelijk weergeeft.



8.2.1.2 Gegevensmodel:

Gebruikers

gbr-id		
gbr-gebruikersnaam		
gbr-paswoord		
gbr-naam		
gbr-vnaam		
gbr-email		
gbr-taal	o	
klant-id	o	↳ Klanten

Groepen

groep-id		
groep-omschrijving	u	

GebruikerGroepen

gbr-id		↳ Gebruikers
groep-id		↳ Groepen

Rechten

recht-id		
recht-naam	u	
recht-alert		

RechtGroepen

recht-id		↳ Rechten
groep-id		↳ Groepen

Klanten

klant-id		
klant-naam		
klant-vnaam		
klant-firma		
klant-aanspreek		
klant-adres		
klant-postcode		
klant-gemeente		
klant-telefoon		
klant-mobiel	o	
klant-fax	o	
klant-email		
klant-status		↳ Statussen

Projecten

project-id		
project-naam	u	
project-omschrijving		
klant-id		↳ Klanten
gebruiker-id		↳ Gebruikers
project-budgeturen	o	
project-presturen	o b	
project-kostprijs	o	
prioriteit-id		↳ Prioriteiten
status-id		↳ Statussen
project-einddatum		

Taken

taak-id		
project-id		↳ Projecten
gbr-aangemaakt-id		↳ Gebruikers
gbr-uitvoerder-id		↳ Gebruikers
taak-onderwerp		
taak-omschrijving	o	
taak-begin		
taak-eind	o	
taak-duurvoorzien	o	
prioriteit-id		↳ Prioriteiten
type-id		↳ Types
status-id		↳ Statussen

Jobs

job-id		
taak-id		↳ Taken
job-omschrijving		
status-id		↳ Statussen

JobDetails

job-id		↳ Jobs
jobdetails-volgnr		
gbr-id		↳ Gebruikers
jobdetail-specificaties	o	
jobdetail-begin		
jobdetail-eind	o	
jobdetail-vlag	o	
contract-id	o	↳ Contracten
jobdetail-fictievetijd	o	

Contracten

contract-id		
klant-id		↳ Klanten
contract-uren		
contract-kostprijs	o	
contract-omschrijving		
contract-datum		

Logs

log-id		
gbr-id		↳ Gebruikers
log-datum		
log-categorie		
log-omschrijving		

Prioriteiten

prioriteit-id	
prioriteit-naam	u
prioriteit-stijl	u

Statussen

status-id	
status-naam	u
status-stijl	u

Types

type-id	
type-naam	u
type-stijl	u

8.2.1.3 Databasestructuur:

Op basis van bovenstaand gegevensmodel zal dan het database ontwerp gemaakt worden. De structuur van de database zal normaliter gelijkaardig blijven aan die van het gegevensmodel. Eventuele veranderingen zijn afhankelijk van het type database waar mee gewerkt wordt, maar ook eventuele standaarden waar men zich binnen de programmeeromgeving of intern binnen de onderneming zal aan houden..

De structuur van het bovenstaande gegevensmodel zal dus behouden blijven. Er zullen echter enkele aanpassingen gebeuren wat de naamgeving betreft, hierbij is rekening gehouden bij enkele interne gewoontes die men handhaaft bij Sanmax. De tabelnamen blijven in het meervoud. Dit ook om problemen te voorkomen. Zo zouden de tabelnamen 'group' en 'right' fouten geven, aangezien dit gereserveerde benamingen zijn binnen MySQL. 'Groups' en 'Rights' zullen dus geen probleem vormen. Alle veldnamen zullen naar het Engels omgezet worden. Ook elke prefix die verwijst naar de tabel zal weggelaten worden. Tabelnamen worden in het enkelvoud geplaatst en diegene die bestaan uit samengevoegde namen zullen gescheiden worden door een underscore (liggend streepje, _). Er zal ook een underscore gebruikt worden bij de scheiding in veldnamen (in plaats van het koppelteken, -).

Bepaalde velden zullen ook een ander type krijgen dan verwacht, dit is namelijk afhankelijk van de mogelijkheden afhankelijk van het type databank. Tijdens dit project wordt er gewerkt met een MySQL databank.

users

id		
login		
password		
name		
fname		
email		
language	o	
client_id	o	↳ customers

groups

id		
name	u	

user_groups

user_id		↳ users
group_id		↳ groups

rights

id		
name	u	
alert		

right_groups

right_id		↳ rights
group_id		↳ groups

customers

id		
name		
fname		
company		
title		
address		
zip		
city		
phone		
mobile	o	
fax	o	
email		
status_id		↳ status

projects

id		
name	u	
description		
client_id		↳ customers
user_id		↳ users
hours_budget	o	
hours_perform	o b	
cost	o	
priority_id		↳ priorities
status_id		↳ status
end_date	o	

tasks

id		
project_id		↳ projects
author_id		↳ users
user_id		↳ users
subject		
description	o	
date_start		
date_start	o	
planned_hours	o	
priority_id		↳ priorities
type_id		↳ types
status_id		↳ status

jobs

id		
task_id		↳ tasks
description		
status_id		↳ status

job_details

job_id		↳ jobs
nr		
user_id		↳ users
specification	o	
date_start		
date_end	o	
flag	o	
contract_id	o	↳ contracts
fict_time	o	

contracts

id		
client_id		↳ customers
hours		
cost	o	
description		
datum		

logs

id		
user_id		↳ users
date		
category		
description		

priorities

id		
name	u	
style	u	

status

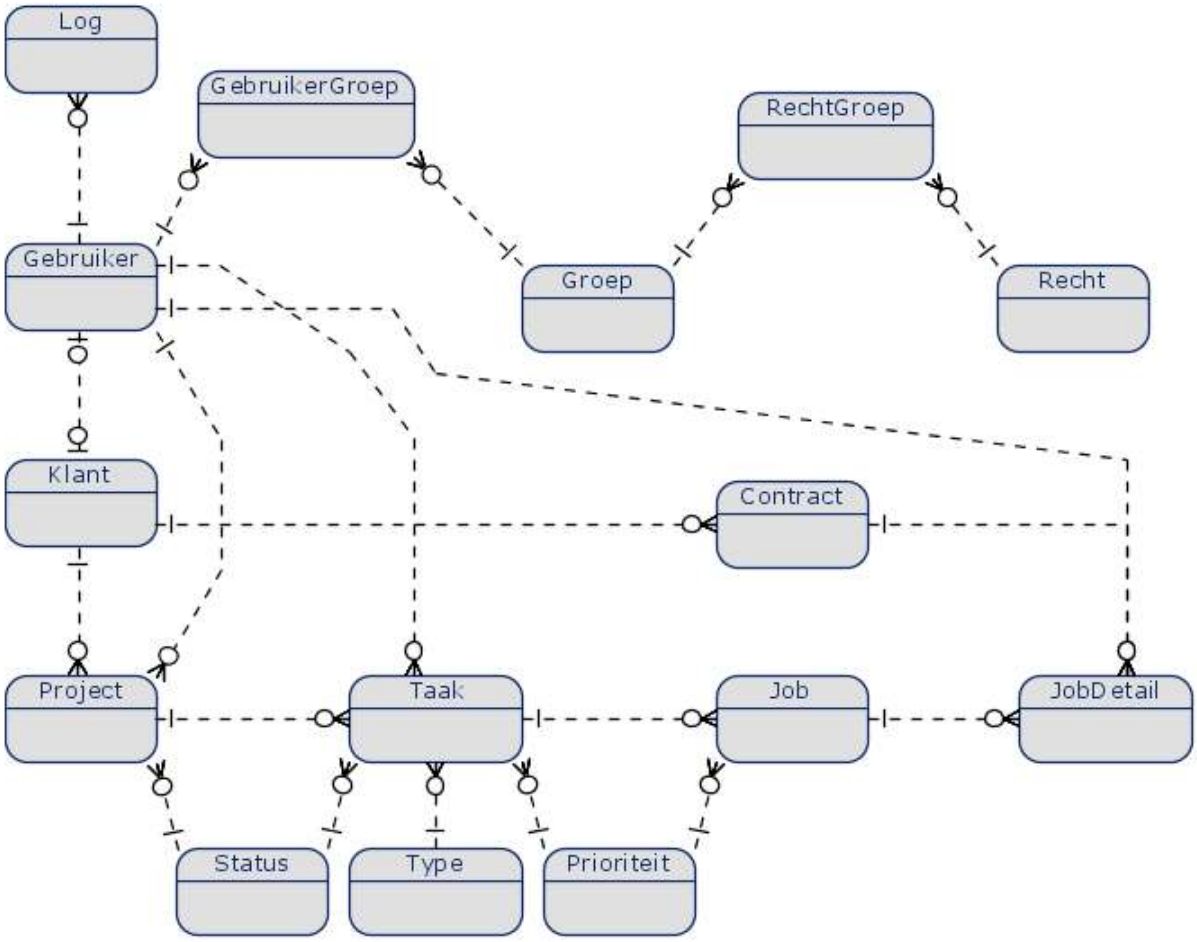
id		
name	u	
style	u	

types

id		
name	u	
style	u	

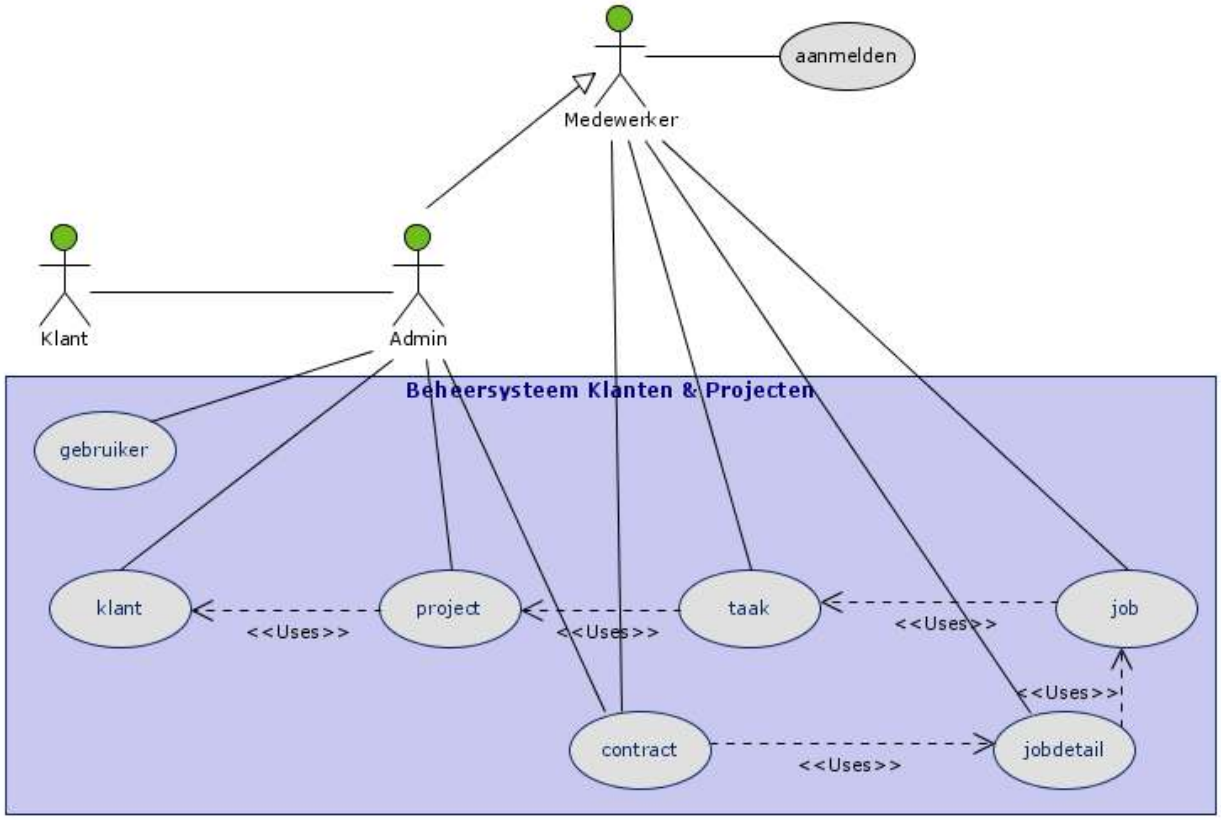
8.2.1.4 Entity-Relationship Diagram

Onderstaand ERD is gebaseerd op het Nederlandstalig gegevensmodel.



8.2.2 Object Georienteerde Analyse

8.2.2.1 Use-case-diagrammen



Aanmelden:

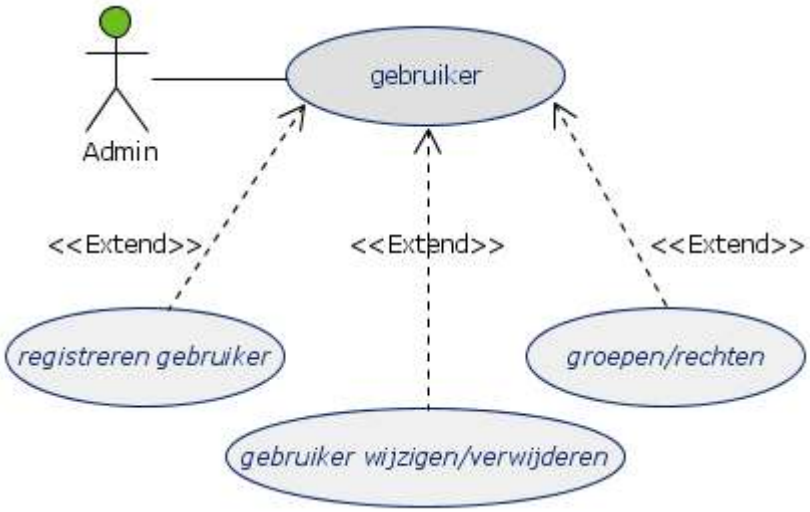
Naam	**Aanmelden van de medewerker**
Samenvatting	De medewerker zal zich aanmelden op het systeem.
Actors	Medewerker (eventueel als Admin)
Startvoorwaarde	De medewerker moet beschikken over de nodige inloggegevens. Meerbepaald over een gebruikersnaam en een paswoord.
Beschrijving	*Actor ingave* / *Systeem bewerking* 1 Gebruikersnaam ingeven 2 Paswoord ingeven 3 Gegevens valideren (fout: 1) 4 Actie selecteren 5 Actie uitvoeren (herhaling: 4) 6 Afmelden
Uitzondering	De aanmeld gegevens van de medewerker kloppen niet. Optioneel de mogelijkheid geven om de administrator hiervan op de hoogte te brengen, paswoord op te vragen via mail...
Resultaat	De gebruiker is aangemeld en kan bepaalde functies, afhankelijk van zijn recht en, oproepen.

Gebruiker:

Naam	**Beheer van de gebruikers**
Samenvatting	Een administrator zal de gebruikers van het systeem beheren. Gebruikers kunnen worden toegevoegd, gekoppeld worden aan groepen (en zo de nodige rechten bekomen), aangepast of verwijderd worden.
Actors	Admin
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren.
Beschrijving	*Actor ingave* / *Systeem bewerking* 1 Actie selecteren: - nieuwe gebruiker registreren - gebruiker wijzigen of verwijderen - groepen en rechten beheren 2 Geselecteerde actie uitvoeren (herhaling: 1)
Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen. De medewerker (= geen admin) heeft niet genoeg rechten om deze functie op te roepen

Resultaat	De gebruiker kan alle gebruikers binnen het systeem beheren.

Specifieker:



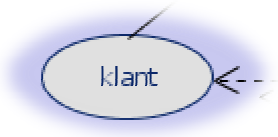
Naam	**Registratie van een gebruiker**		
Samenvatting	Een nieuwe gebruiker zal op het systeem geregistreerd worden.		
Actors	Admin		
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren.		
Beschrijving		*Actor ingave*	*Systeem bewerking*
	1	Gegevens gebruiker ingeven	
	2	Gebruiker toewijzen aan groep(en)	
	3		Gegevens valideren (ok: melding + wegschrijven DB) (fout: melding + 1) (herhaling: 1)
Uitzondering	(+ zie uitzonderingen bovenliggende use-case)		
Resultaat	Een nieuwe gebruiker is aan het systeem toegevoegd.		

Naam	**Wijzigen / verwijderen van een gebruiker**		
Samenvatting	De gegevens van een bestaande gebruiker zullen gewijzigd of verwijderd worden.		
Actors	Admin		
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren. Er moet een of meerdere gebruikers aanwezig zijn in het systeem.		
Beschrijving		*Actor ingave*	*Systeem bewerking*
	1	Gebruiker selecteren en een bijhorende functie oproepen. - wijzig - verwijder	
	2		Gegevens valideren (ok: melding + aanpassing DB) (fout: melding + 1)

		(herhaling: 1)
Uitzondering	De gebruiker is nog gekoppeld aan actieve projecten, taken, jobs of jobdetails en kan dus niet zomaar verwijderd worden. (+ zie uitzonderingen bovenliggende use-case)	
Resultaat	De gegevens van een bestaande gebruiker zijn gewijzigd in of verwijderd uit het systeem.	

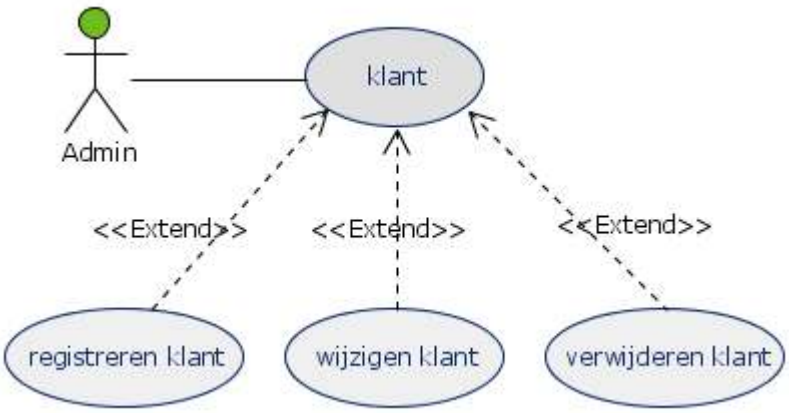
Naam	**Het beheer van groepen en rechten.**		
Samenvatting	Groepen kunnen worden toegevoegd of aangepast. En aan deze groepen kunnen rechten gekoppeld worden, die van toepassing zullen zijn op alle gebruikers binnen een groep.		
Actors	Admin		
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren.		
Beschrijving		***Actor ingave***	***Systeem bewerking***
	1	Actie selecteren: - groepen toevoegen - groepen aanpassen - groepen verwijderen	
	2		Actie uitvoeren en de nodige pagina en componenten tonen.
	3	Bij het toevoegen en aanpassen: mogelijkheid om op een efficiënte manier de nodige rechten aan een groep toe te kennen.	
	4		Gegevens valideren (ok: melding + aanpassing DB) (fout: melding + 3) (herhaling: 1 of 3)
Uitzondering	De groep bevat nog actieve gebruikers en kan dus niet zomaar verwijderd worden. (+ zie uitzonderingen bovenliggende use-case)		
Resultaat	De gegevens van een groep zijn toegevoegd aan, gewijzigd in of verwijderd uit het systeem.		

Klant:



Naam	**Beheer van de klanten**	
Samenvatting	Een administrator zal de klanten van Sanmax beheren. Klanten kunnen toegevoegd, gewijzigd of verwijderd worden uit het systeem.	
Actors	Admin	
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren.	
Beschrijving	*Actor ingave*	*Systeem bewerking*
	1 Actie selecteren: - nieuwe klant registreren - klant wijzigen - klant verwijderen	
	2	Geselecteerde actie uitvoeren (herhaling: 1)
Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen. De medewerker (= geen admin) heeft niet genoeg rechten om deze functie op te roepen	
Resultaat	De gebruiker kan alle klanten binnen het systeem beheren.	

Specifieker:



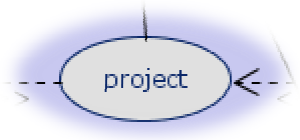
Naam	**Registratie van een klant**		
Samenvatting	Een nieuwe klant zal op het systeem geregistreerd worden.		
Actors	Admin		
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren.		
Beschrijving		*Actor ingave*	*Systeem bewerking*
	1	Contactgegevens klant ingeven	
	2		Gegevens valideren (ok: melding + wegschrijven DB) (fout: melding + 1) (herhaling: 1)
	3	Eventueel mogelijkheid om al een project aan te maken voor deze klant. (use-case 'project')	
	4		Doorverwijzen naar de pagina voor het aanmaken van een project. (vb. door het meegeven van een parameter met het klant-id).
Uitzondering	(+ zie uitzonderingen bovenliggende use-case)		
Resultaat	Een nieuwe klant is aan het systeem toegevoegd.		

Naam	**Wijzigen van een klant**		
Samenvatting	De gegevens van een bestaande gebruiker zullen gewijzigd worden.		
Actors	Admin		
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren. Er moet een of meerdere klanten aanwezig zijn in het systeem.		
Beschrijving		*Actor ingave*	*Systeem bewerking*
	1	Klant selecteren en de nodige gegevens aanpassen.	
	2		Gegevens valideren (ok: melding + aanpassing DB) (fout: melding + 1) (herhaling: 1)
Uitzondering	(+ zie uitzonderingen bovenliggende use-case)		
Resultaat	De gegevens van een bestaande klant zijn gewijzigd in het systeem.		

Naam	**Verwijderen van een klant**	
Samenvatting	De gegevens van een bestaande gebruiker zullen definitief verwijderd worden.	
Actors	Admin	
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren. Er moet een of meerdere klanten aanwezig zijn in het systeem.	
Beschrijving	***Actor ingave***	***Systeem bewerking***
	1 Klant selecteren en aangeven dat men deze wilt verwijderen.	
	2	Bevestiging vragen
	3 Bevestiging geven (ok) of annuleren (fout)	
	4	Gegevens valideren (ok: melding + aanpassing DB) (fout: melding + 1) (herhaling: 1)
Uitzondering	De klant heeft nog een project lopen en zal men dus niet kunnen verwijderen. (+ zie uitzonderingen bovenliggende use-case)	
Resultaat	De gegevens van een bestaande klant zijn verwijderd uit het systeem.	

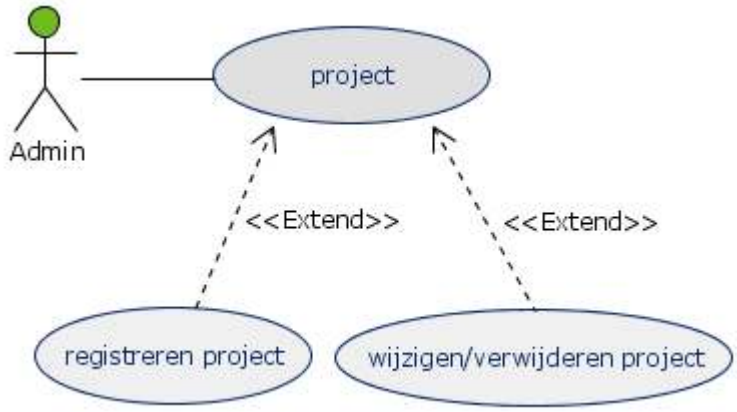
Voor de overige use-cases zal ik de extended use-cases niet meer zo gedetailleerd uittypen. Enerzijds is het principe van toevoegen, wijzigen en verwijderen overal min of meer gelijkaardig. En anderzijds zou dit dit analyse document te complex maken en zou het overzicht mogelijk verloren kunnen gaan. En dat is natuurlijk niet de bedoeling. Daarom kan je bovenstaande use-case beschrijvingen als norm nemen voor alle gelijkaardige handelingen, dus toevoegen, wijzigen en verwijderen, bij de overige use-cases.

Project:



Naam	**Beheer van de projecten**	
Samenvatting	Een administrator zal de projecten beheren. Projecten kunnen toegevoegd, gewijzigd (zowel inhoud als status) of verwijderd worden uit het systeem.	
Actors	Admin	
Startvoorwaarde	De medewerker moet ingelogd zijn en tot de groep Admin behoren. Voor het wijzigen en verwijderen moeten er bestaande projecten in het systeem aanwezig zijn.	
Beschrijving	*Actor ingave*	*Systeem bewerking*
	1 Actie selecteren: - nieuwe project toevoegen - project wijzigen of verwijderen	
	2	Geselecteerde actie uitvoeren (herhaling: vanaf 1)
Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen. De medewerker (= geen admin) heeft niet genoeg rechten om deze functie op te roepen	
Resultaat	De gebruiker kan alle projecten binnen het systeem beheren.	

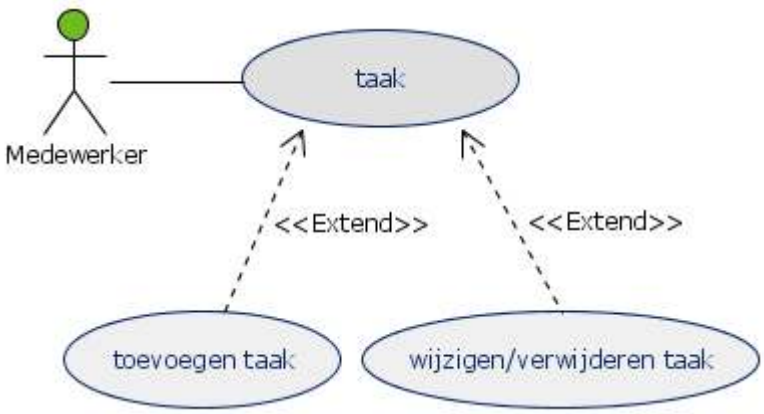
Specifieker:



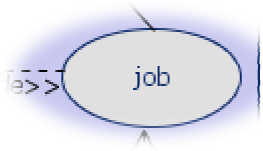
Taak:

Naam	**Beheer van de taken**	
Samenvatting	Een medewerker zal taken kunnen beheren. Men kan een of meerdere taken toevoegen voor een project, zowel voor zichzelf als voor een andere medewerker. Deze taken kunnen ook gewijzigd worden (zowel inhoud als status) en indien nodig verwijderd worden.	
Actors	Medewerker	
Startvoorwaarde	De medewerker moet ingelogd zijn. Voor het toevoegen moet er een project bestaan waaraan taken toegevoegd kunnen worden. Voor het wijzigen en verwijderen moeten er een bestaande taak in het systeem aanwezig zijn.	
Beschrijving	*Actor ingave*	*Systeem bewerking*
	1 Actie selecteren: - nieuwe taak toevoegen - bestaande taak wijzigen of verwijderen	
	2	Geselecteerde actie uitvoeren (herhaling: vanaf 1)
Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen.	
Resultaat	De gebruiker kan alle projecten binnen het systeem beheren.	

Specifieker:

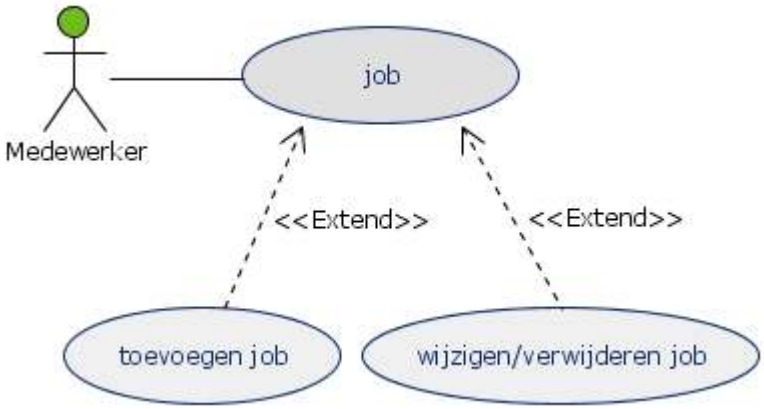


Job:



Naam	**Beheer van de jobs**	
Samenvatting	Een medewerker zal jobs kunnen beheren. Men kan een of meerdere jobs toevoegen voor een taak, aan een job zal gewerkt kunnen worden door een of meerdere gebruikers, gespreid over een of meerdere dagen (zie jobdetails). Deze jobs kunnen ook gewijzigd worden (zowel inhoud als status) en indien nodig verwijderd worden.	
Actors	Medewerker	
Startvoorwaarde	De medewerker moet ingelogd zijn. Voor het toevoegen moet er een taak bestaan waaraan een job toegevoegd kan worden. Voor het wijzigen en verwijderen moeten er een bestaande job in het systeem aanwezig zijn.	
Beschrijving	*Actor ingave*	*Systeem bewerking*
	1 Actie selecteren: - nieuwe job toevoegen - bestaande job wijzigen of verwijderen	
	2	Geselecteerde actie uitvoeren (herhaling: vanaf 1)
Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen.	
Resultaat	De gebruiker kan alle jobs binnen het systeem beheren.	

Specifieker:



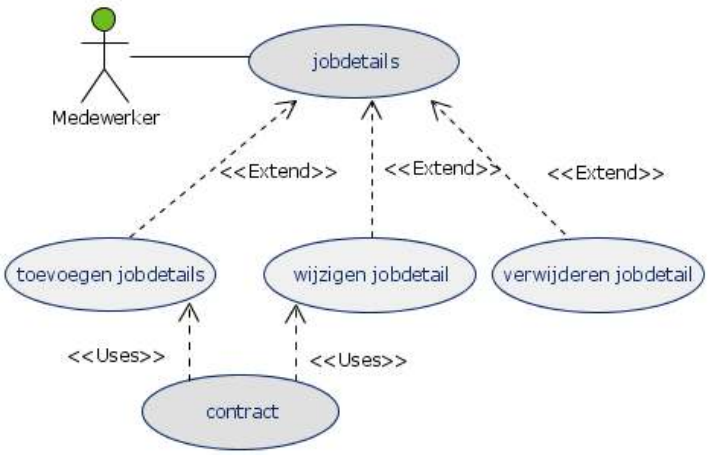
Jobdetail:

Naam	**Beheer van de jobdetails**
Samenvatting	Een medewerker zal jobdetails voor zichzelf kunnen beheren. Een jobdetail is het eigenlijke werk van één medewerker aan een job. Het is mogelijk dat er meerdere jobdetails per job geregistreerd zullen worden. Ook kan mijn jobdetails wijzigen (vb. beïndigen van een jobdetails) en eventueel verwijderen.
Actors	Medewerker
Startvoorwaarde	De medewerker moet ingelogd zijn. Voor het toevoegen moet er een job bestaan waaraan een jobdetail toegevoegd kan worden. Voor het wijzigen en verwijderen moeten er een bestaande jobdetail in het systeem aanwezig zijn.
Beschrijving	

	Actor ingave	*Systeem bewerking*
1	Job selecteren	
2		Eventueel bestaande gegevens (zoals jobdetails ophalen)
3	Actie selecteren: - nieuwe jobdetail toevoegen (4) - bestaande jobdetail wijzigen of verwijderen (5)	
4	Eventueel aangeven of de uren dat gewerkt wordt, zoals aangegeven in het jobdetail, aangerekend moeten worden bij het onderhoudscontract.	
5		Geselecteerde actie uitvoeren (herhaling: vanaf 1)

Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen.
Resultaat	De gebruiker kan alle jobs binnen het systeem beheren.

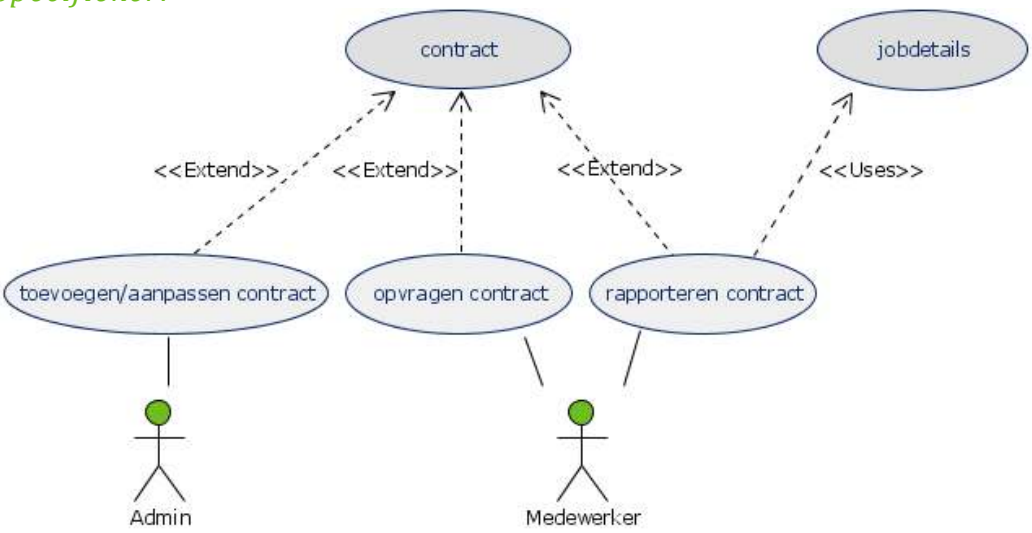
Specifieker:



Contract:

Naam	**Beheer van de contracten**	
Samenvatting	Een medewerker zal alle gegevens van de contracten kunnen vragen en eventueel hierop rapporteren (oa. via zijn jobdetails). Het aanmaken van contracten gebeurt door een medewerkers (admin) met voldoende rechten. Zoals al aangegeven in de beschrijving van de use-case 'jobdetails' kan elke medewerkers aangeven dat zijn werkuren mee gecalculeerd moeten worden voor een onderhoudscontract. Een goede optie zou zijn dat de administrator de vermindering van de uren van een onderhoudscontract kan controleren en eventueel zijn goedkeuring moet geven.	
Actors	Admin, Medewerker	
Startvoorwaarde	De medewerker moet ingelogd zijn. Voor het toevoegen van contracten moet de medewerkers ingelogd zijn als admin. Ook moeten de klant bestaan waarvoor een contract zal opgemaakt worden. Voor het opvragen van een overzicht of rapporteren op een contract (via jobdetails) moet er uiteraard een contract aangemaakt zijn.	
Beschrijving	*Actor ingave*	*Systeem bewerking*
	1 Actie selecteren: - nieuwe contract toevoegen - contract aanpassen (vb: jobdetails goedkeuren) - overzicht contracten opvragen	
	2	Geselecteerde actie uitvoeren (herhaling: 1)
Uitzondering	De medewerker is niet (meer) ingelogd, mogelijkheid geven tot inloggen.	
Resultaat	De gebruiker kan alle jobs binnen het systeem beheren.	

Specifieker:



Opgelet: zoals al eerder vermeldt is een admin een medewerker met meer en uitgebreidere rechten. Een admin kan dus perfect contracten opvragen of rapporteren op een contract, aangezien hij ook als medewerker kan acteren.

Verder kan of zal er in dit systeem ook nog een log-systeem voorzien worden om alle mogelijke handelingen die er gebeuren bij te houden. Dit is echter niet verwerkt in het use-case diagram aangezien dit op de achtergrond zal gebeuren bij elke handeling. De administrator zal deze logs wel kunnen opvragen via pagina met een eenvoudig overzicht. Dit is echter weinig relevant voor het systeem wat klant- en projectbeheer betreft en is dus niet opgenomen.

Ook zal de admin eventueel ook de mogelijkheid hebben om records toe te voegen aan de tabel ‘prioriteit’, ‘status’ en ‘type’ of om aanpassingen te doen aan bestaande records (vb. de kleur opstelling). Deze acties zijn echter irrelevant voor de werking van het systeem en zullen waarschijnlijk ook weinig tot niet gebruikt worden. Ook deze werden op hun beurt niet toegevoegd omwille van bovenstaande argumenten.

8.2.3 Website Architectuur

8.2.3.1 Levels

Allereerst zal de structuur van de website uitgeschreven worden. Het uitschrijven van de structuur betekend concreet dat men zal werken met verschillende levels. De startpagina zal op het eerste level gelegen zijn. De globale pagina's die ook gelinkt zullen zijn op de homepagina zijn van het tweede level. Binnen deze pagina's kan dan nog eens locale navigatie aanwezig zijn die dan van het derde level zullen zijn, enzovoorts. Het aantal levels is in principe onbeperkt, toch is het aan te raden de website niet te diep uit te werken. De kans wordt dan groter dat de structuur en navigatie voor de gebruiker niet meer duidelijk en gebruikersvriendelijk zal zijn.

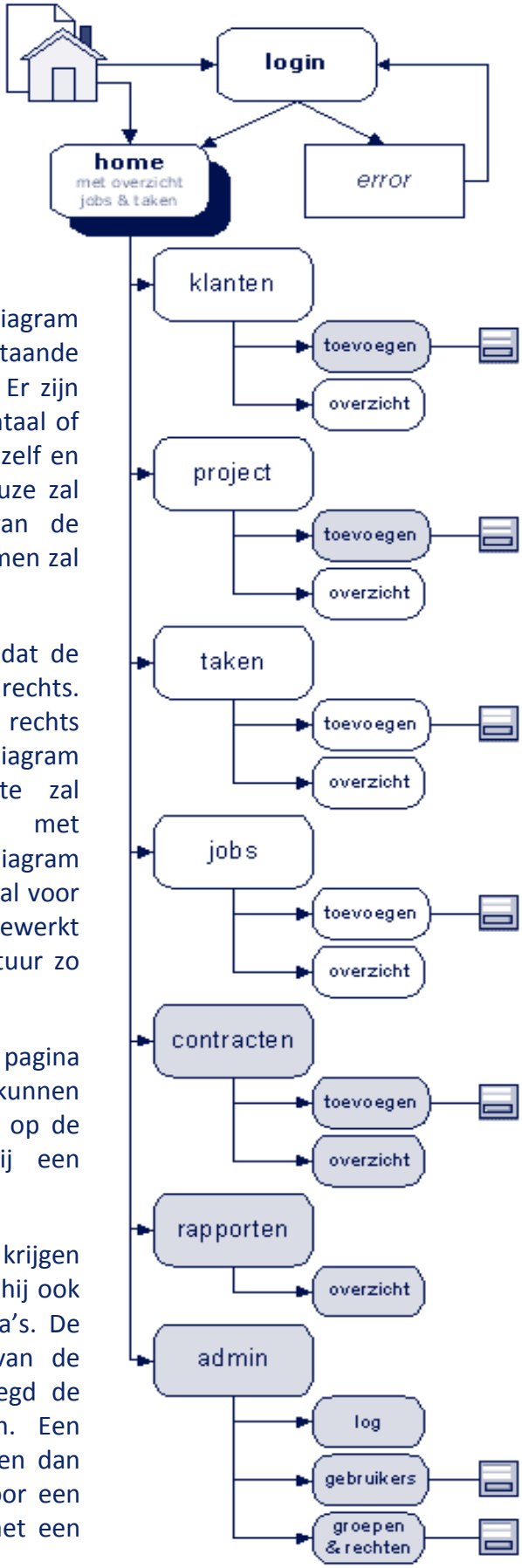
Wanneer we de structuur opstellen zal ook gewerkt worden met een specifieke nummering. De reden hiervoor is dat het mogelijk is dat bepaalde pagina's dezelfde naam zullen dragen, daarom geven we elke pagina een unieke nummer. Aan de hand van deze nummering zal ook het level afgeleid kunnen worden.

De structuur van de door mij uitgewerkte website zal er als volgt uitzien. Let wel op, dit schema is vereenvoudigt en beperkt gehouden met als doel u een beeld van de werkwijze te scheppen. Het onderdeel 'Toevoegen' moet hier in de ruime zin bekeken worden. Onder deze pagina wordt alles verstaan dat aanpassingen teweeg zal brengen in de gegevens. Naast het eigenlijk toevoegen van gegevens wordt ook het aanpassen en verwijderen van gegevens hier geplaatst, aangezien dit toch binnen hetzelfde level geplaatst zal worden.

De structuur van de door mij uitgewerkte website zal er als volgt uitzien:

1 Home
- 1.1 Klant
  - 1.1.1 Toevoegen
  - 1.1.2 Overzicht
- 1.2 Project
  - 1.2.1 Toevoegen
  - 1.2.2 Overzicht
- 1.3 Taken
  - 1.3.1 Toevoegen
  - 1.3.2 Overzicht
- 1.4 Jobs
  - 1.4.1 Toevoegen
  - 1.4.2 Overzicht
- 1.5 Rapporten
  - 1.5.1 Overzicht
- 1.6 Contracten
  - 1.6.1 Toevoegen
  - 1.6.2 Overzicht
- 1.7 Admin
  - 1.7.1 Log
  - 1.7.2 Gebruikers
  - 1.7.3 Groepen & rechten

Afhankelijk van de rechten van de aangemelde gebruiker zal deze structuur verschillen. Ook is het mogelijk dat de eigenlijke indeling van de pagina's en mappen zal verschillen. Aangezien er gewerkt zal worden met een framework zal er waarschijnlijk een bepaalde structuur aangehouden moeten worden.



8.2.3.2 Diagram

Naast het uitschrijven zal ook hier weer een diagram gemaakt worden. In dit diagram zal bovenstaande structuur (de levels) logischerwijze terugkomen. Er zijn twee soorten diagrammen mogelijk, een horizontaal of verticaal diagram. Deze keuze is voor de analist zelf en maakt geen verschil voor de uitwerking. De keuze zal meestal afhankelijk zijn van de diepgang van de structuur en de manier waarop men de diagrammen zal willen afprinten in de documentatie.

Ik koos een horizontaal diagram. Dit betekend dat de levels uitgewerkt zullen worden van links naar rechts. Met level 1 uiterst links en de volgende levels rechts ervan. De volledige website wordt hier in één diagram verwerkt. Wanneer men een grote website zal uitwerken kan men het best werken met subdiagrammen. Men zal dan één globaal diagram maken met de level 1 en 2 pagina's. Bijkomend zal voor elke level 2 pagina's alle onderliggende levels uitgewerkt worden in een nieuw diagram. Dit om de structuur zo overzichtelijk mogelijk te houden.

Mijn uitgewerkt diagram vindt u rechts op deze pagina terug. Van op de startpagina zal de gebruiker kunnen inloggen. Bij het juist inloggen komt hij terecht op de startpagina, bij het fout aanmelden zal hij een foutmelding krijgen en opnieuw kunnen inloggen.

Op de startpagina zal de gebruiker een overzicht krijgen van zijn huidige taken en jobs. Via het menu zal hij ook kunnen navigeren naar de onderliggende pagina's. De mogelijkheden zullen echter afhankelijk zijn van de rechten die aan deze gebruiker, of beter gezegd de groep waartoe hij behoort, zijn toegewezen. Een administrator zal meer pagina's kunnen opvragen dan een gewone gebruiker. De pagina's specifiek voor een administrator zijn in dit diagram aangegeven met een donkerdere achtergrond.

8.3 Database

8.3.1 Script

```
-- VOORKOMEN DAT DE FOREIGNKEYS GECONTROLEERD ZULLEN WORDEN
-- ERROR 1217: Cannot delete or update a parent row: a foreign key constraint fails
SET FOREIGN_KEY_CHECKS=0;

-- DROPPEN (BESTAANDE) TABELLEN
DROP TABLE IF EXISTS `users`;
DROP TABLE IF EXISTS `groups`;
DROP TABLE IF EXISTS `user_groups`;
DROP TABLE IF EXISTS `rights`;
DROP TABLE IF EXISTS `customers`;
DROP TABLE IF EXISTS `projects`;
DROP TABLE IF EXISTS `tasks`;
DROP TABLE IF EXISTS `jobs`;
DROP TABLE IF EXISTS `job_details`;
DROP TABLE IF EXISTS `contracts`;
DROP TABLE IF EXISTS `logs`;
DROP TABLE IF EXISTS `priorities`;
DROP TABLE IF EXISTS `status`;
DROP TABLE IF EXISTS `types`;

-- HET CONTROLEREN VAN FOREIGN KEYS TERUG AANZETTEN
SET FOREIGN_KEY_CHECKS=1;

-- AANMAKEN TABELLEN

CREATE TABLE `users` (
   `id`           INT ( 3 )      UNSIGNED NOT NULL AUTO_INCREMENT ,
   `login`        VARCHAR( 20 )  NOT NULL ,
   `password`     VARCHAR( 50 )  NOT NULL ,
   `name`         VARCHAR( 20 )  NOT NULL ,
   `fname`        VARCHAR( 20 )  NOT NULL ,
   `email`        VARCHAR( 30 )  NOT NULL ,
   `language`     CHAR( 3 )      NULL DEFAULT 'nl' ,
   PRIMARY KEY (`id`) ,
   UNIQUE (`login`)
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `groups` (
   `id`           INT( 2 )       UNSIGNED NOT NULL AUTO_INCREMENT,
   `name`         VARCHAR( 20 )  NOT NULL,
   PRIMARY KEY (`id`) ,
   UNIQUE (`name`)
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `user_groups` (
   `user_id`      INT( 3 ) UNSIGNED NOT NULL ,
   `group_id`     INT( 2 ) UNSIGNED NOT NULL ,
   PRIMARY KEY (`user_id` ,`group_id`) ,
   FOREIGN KEY `fk_user` (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,
   FOREIGN KEY `fk_group` (`group_id`) REFERENCES `groups` (`id`) ON DELETE NO ACTION
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `rights` (
  `id`            INT(11)        UNSIGNED NOT NULL AUTO_INCREMENT,
  `group_id`      INT(11)        UNSIGNED NOT NULL,
  `controller`    VARCHAR(255)   NOT NULL,
  `action`        VARCHAR(255)   NOT NULL,
  `actionid`      VARCHAR(6)     NOT NULL DEFAULT '*',
  FOREIGN KEY `fk_groupr` (`group_id`) REFERENCES `groups` (`id`) ON DELETE NO ACTION,
  PRIMARY KEY  (`id`)
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;
```

```
CREATE TABLE `customers` (
   `id`           INT ( 4 )      UNSIGNED NOT NULL AUTO_INCREMENT ,
   `int_nr`       INT ( 3 )      UNSIGNED NOT NULL ,
   `acc_nr`       INT ( 9 )      UNSIGNED NULL,
   `fname`        VARCHAR( 20 )  NULL ,
   `name`         VARCHAR( 20 )  NULL ,
   `company`      VARCHAR( 60 )  NOT NULL ,
   `title`        VARCHAR( 5 )   NULL ,
   `address`      VARCHAR( 60 )  NULL ,
   `zip`          VARCHAR( 6 )   NULL ,
   `city`         VARCHAR( 30 )  NULL ,
   `country`      VARCHAR( 30 )  NULL DEFAULT 'BE',
   `vat`          VARCHAR( 30 )  NULL ,
   `phone`        VARCHAR( 15 )  NULL ,
   `mobile`       VARCHAR( 15 )  NULL ,
   `fax`          VARCHAR( 15 )  NULL ,
   `email`        VARCHAR( 40 )  NULL ,
   `done`         BOOL           NULL DEFAULT 0,
   `comment`      BLOB           NULL ,
  PRIMARY KEY (`id`),
  UNIQUE (`int_nr`),
  UNIQUE (`acc_nr`)
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `projects` (
   `id`           INT ( 5 )      UNSIGNED NOT NULL AUTO_INCREMENT ,
   `name`         VARCHAR( 120 ) NOT NULL ,
   `description`  BLOB           NOT NULL ,
   `customer_id`  INT ( 4 )      UNSIGNED NOT NULL ,
   `user_id`      INT ( 3 )      UNSIGNED NOT NULL ,
   `hours_budget` INT ( 3 )      NULL ,
   `hours_perform`INT ( 3 )     NULL ,
   `cost`         FLOAT ( 6 ,2 ) NULL ,
   `priority_id`  INT ( 2 )      NOT NULL ,
   `status_id`    INT ( 2 )      NOT NULL ,
   `date_end`     DATE           NULL ,
  PRIMARY KEY (`id`) ,
  FOREIGN KEY (`customer_id`) REFERENCES `customers` (`id`) ON DELETE NO ACTION,
  FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `tasks` (
   `id`           INT ( 5 )      UNSIGNED NOT NULL AUTO_INCREMENT ,
   `project_id`   INT ( 5 )      UNSIGNED NOT NULL ,
   `author_id`    INT ( 3 )      UNSIGNED NOT NULL ,
   `user_id`      INT ( 3 )      UNSIGNED NOT NULL ,
   `subject`      VARCHAR ( 30 ) NOT NULL ,
   `description`  BLOB           NULL ,
   `date_start`   DATETIME       NOT NULL ,
   `date_end`     DATETIME       NULL ,
   `hours_planned`INT ( 3 )      NULL ,
   `priority_id`  INT ( 2 )      NOT NULL ,
   `type_id`      INT ( 2 )      NOT NULL ,
   `status_id`    INT ( 2 )      NOT NULL ,
  PRIMARY KEY (`id`) ,
  FOREIGN KEY (`project_id`) REFERENCES `projects` (`id`) ON DELETE NO ACTION,
  FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION,
  FOREIGN KEY (`author_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `jobs` (
   `id`           INT ( 5 )      UNSIGNED NOT NULL AUTO_INCREMENT ,
   `task_id`      INT ( 5 )      UNSIGNED NOT NULL ,
   `subject`      VARCHAR ( 75 ) NOT NULL ,
   `status_id`    INT ( 2 )      NOT NULL ,
  PRIMARY KEY (`id`) ,
  FOREIGN KEY (`task_id`) REFERENCES `tasks` (`id`) ON DELETE NO ACTION
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;
```

```
CREATE TABLE `job_details` (
   `id`           INT ( 5 )      UNSIGNED NOT NULL AUTO_INCREMENT,
   `job_id`       INT ( 5 )      UNSIGNED NOT NULL ,
   `user_id`      INT ( 3 )      UNSIGNED NOT NULL ,
   `specification`BLOB           NULL ,
   `date_start`   DATETIME       NOT NULL ,
   `date_end`     DATETIME       NULL ,
   `flag`         BOOL           NULL ,
   `contract_id`  INT ( 11 )     NULL DEFAULT NULL,
   `time_fict`    TIME           NULL ,
   PRIMARY KEY (`id`) ,
   FOREIGN KEY (`job_id`) REFERENCES `jobs` (`id`) ON DELETE CASCADE,
   FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `contracts` (
   `id`           INT ( 11 )     NOT NULL AUTO_INCREMENT,
   `customer_id`  INT ( 4 )      UNSIGNED NOT NULL ,
   `hours`        TIME           NOT NULL ,
   `cost`         FLOAT ( 6 , 2 ) NULL ,
   `description`  BLOB           NOT NULL ,
   `date`         DATE           NOT NULL ,
   PRIMARY KEY (`id`) ,
   FOREIGN KEY (`customer_id`) REFERENCES `customers` (`id`) ON DELETE NO ACTION
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `logs` (
   `id`           INT            NOT NULL ,
   `user_id`      INT ( 4 )      UNSIGNED NOT NULL ,
   `date`         TIMESTAMP      NOT NULL ,
   `category`     VARCHAR ( 20 ) NOT NULL ,
   `description`  TEXT           NOT NULL ,
   PRIMARY KEY (`id`) ,
   FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCADE
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `priorities` (
   `id`           INT  ( 2 )     NOT NULL ,
   `name`         VARCHAR ( 20 ) NOT NULL ,
   `style`        ENUM ('RED' ,'GREEN' ,'YELLOW' ,'ORANGE' ,'BLUE' ,'WHITE' ,'BLACK'
,'GRAY') NOT NULL DEFAULT 'GRAY' ,
   PRIMARY KEY (`id`)
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `status` (
   `id`           INT  ( 2 )     NOT NULL ,
   `name`         VARCHAR ( 20 ) NOT NULL ,
   `style`        ENUM ('RED' ,'GREEN' ,'YELLOW' ,'ORANGE' ,'BLUE' ,'WHITE' ,'BLACK'
,'GRAY') NOT NULL DEFAULT 'GRAY' ,
   PRIMARY KEY (`id`)
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE `types` (
   `id`           INT ( 2 )      NOT NULL ,
   `name`         VARCHAR ( 20 ) NOT NULL ,
   `style`        ENUM ('RED' ,'GREEN' ,'YELLOW' ,'ORANGE' ,'BLUE' ,'WHITE' ,'BLACK'
,'GRAY') NOT NULL DEFAULT 'GRAY' ,
   PRIMARY KEY (`id`)
)
ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_general_ci;
```

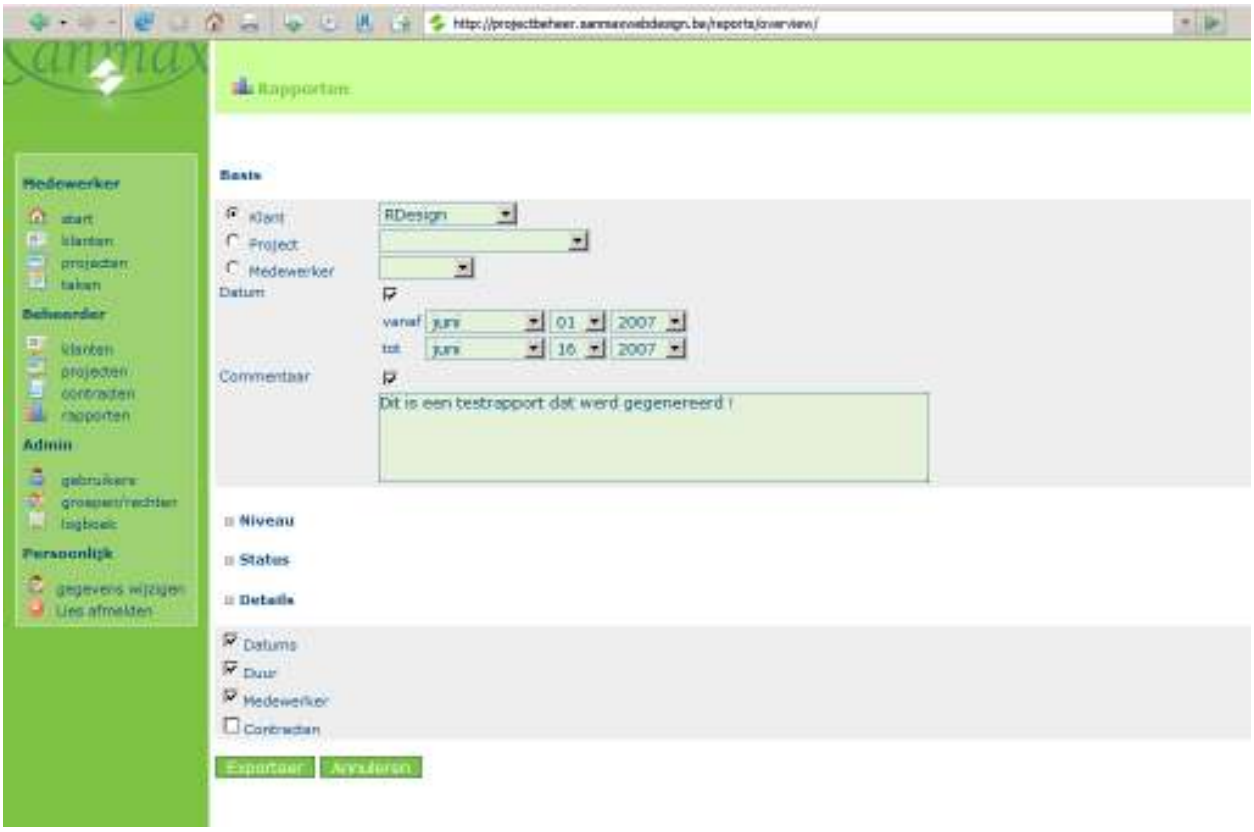
```
-- LEGGEN VAN DE NOG NIET GELEGDE REFERENTIES
ALTER TABLE `projects`
  ADD FOREIGN KEY (`status_id`) REFERENCES `status` (`id`) ,
  ADD FOREIGN KEY (`priority_id`) REFERENCES `priorities` (`id`);

ALTER TABLE `tasks`
  ADD FOREIGN KEY (`status_id`) REFERENCES `status` (`id`) ,
  ADD FOREIGN KEY (`priority_id`) REFERENCES `priorities` (`id`) ,
  ADD FOREIGN KEY (`type_id`) REFERENCES `types` (`id`);

ALTER TABLE `jobs`
  ADD FOREIGN KEY (`status_id`) REFERENCES `status` (`id`);

ALTER TABLE `job_details`
  ADD FOREIGN KEY (`contract_id`) REFERENCES `contracts` (`id`) ON DELETE NO ACTION;


-- VULLEN TABELLEN
INSERT INTO `priorities` VALUES
  (1 ,'very high' ,'RED') ,
  (2 ,'high' ,'ORANGE') ,
  (3 ,'medium' ,'BLUE') ,
  (4 ,'low' ,'GRAY')
;

INSERT INTO `status` VALUES
  (1 ,'not started yet' ,'RED') ,
  (2 ,'in progress' ,'GREEN') ,
  (3 ,'waiting' ,'ORANGE') ,
  (4 ,'done' ,'GRAY'),
  (5 ,'invoice' ,'GRAY')
;

INSERT INTO `types` VALUES
  (1 ,'new' ,'GREEN') ,
  (2 ,'check' ,'ORANGE') ,
  (3 ,'bug' ,'RED') ,
  (4 ,'todo' ,'GRAY'),
  (5 ,'change' ,'ORANGE')
;

INSERT INTO `groups` VALUES
  (1 ,'not active') ,
  (2 ,'admin') ,
  (3 ,'developer') ,
  (4 ,'design') ,
  (5 ,'input &amp;amp; html') ,
  (6 ,'customer') ,
  (7 ,'stagiair')
;

-- OPVULLEN REALISTISCHE GEGEVENS
INSERT INTO `users` VALUES
  (1 , 'Pascal' , '***' , 'D&#039;Helft', 'Pascal' , 'pascal.dhelft@sanmax.be', 'nl') ,
  (2 , 'Lies' , '***' , 'Ceuppens', 'Lies' , 'lies@sanmax.be', 'nl') ,
  (3 , 'Christof' , '***' , 'Schraepen', 'Christof' , 'christof@sanmax.be', 'nl') ,
  (4 , 'Andries' , '***' , 'Seutens', 'Andries' , 'andries@sanmax.be', 'nl'),
  (5 , 'Joeri' , '***' , 'Erregeerts', 'Joeri' , 'joeri@sanmax.be', 'nl') ,
  (6 , 'Jochen' , '***' , 'Maenen', 'Jochen' , 'jochen@sanmax.be', 'nl') ,
  (7 , 'Werner' , '***', 'Van Gelder', 'Werner' , 'werner@sanmax.be', 'nl') ,
  (8 , 'Davy' , '***' , 'Rego', 'Davy' , 'davy@sanmax.be', 'nl'),
  (9 , 'Mindy' , '***' , 'Stukken', 'Mindy' , 'mindy.stukken@gmail.com', 'nl') ,
  (10, 'Bart' , '***' , 'Dens', 'Bart' , 'bart@sanmax.be', 'nl')
;
```

```
INSERT INTO `user_groups` VALUES
  (1 , 2) ,
  (2 , 2) ,
  (2 , 4) ,
  (2 , 5) ,
  (3 , 3) ,
  (4 , 3) ,
  (5 , 3) ,
  (6 , 5) ,
  (7 , 3) ,
  (8 , 7) ,
  (9 , 7) ,
  (10, 1)
;

INSERT INTO `rights` VALUES
  (1,1,'Users','login','*'),
  (2,1,'Users','lostpass','*'),
  (3,1,'Users','logout','*'),
  (4,2,'Logs','delete','*'),
  (5,2,'Logs','overview','*'),
  (6,2,'Logs','detail','*'),
   ...
  (143,7,'Users','login','*'),
  (144,7,'Users','lostpass','*'),
  (145,7,'Users','personal','*'),
  (146,7,'Users','logout','*'),
  (147,7,'Customers','overview','*'),
  (148,7,'Customers','detail','*')
;

INSERT INTO `logs` VALUES
  (1 , 1, NOW(), 'reset', 'Leegmaken Logboek (uitvoeren SQL-Script)')
;
```

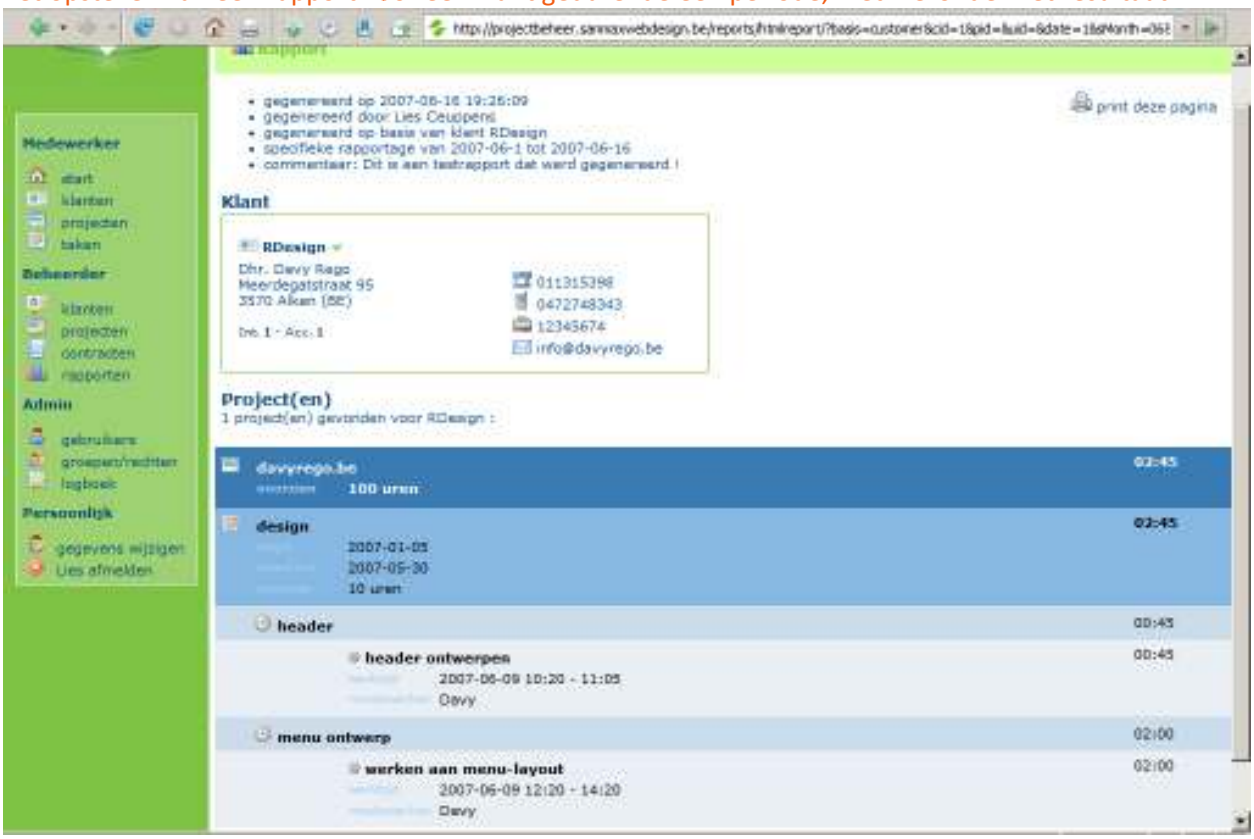
Dit is het aangepast script zoals het uiteindelijk geïmplementeerd geworden is. Dit script komt dus niet meer volledig overeen met de databasestructuur zoals aangegeven in de analyse. De belangrijkste aanpassingen zijn weergegeven bij onderdeel 4.2.3.

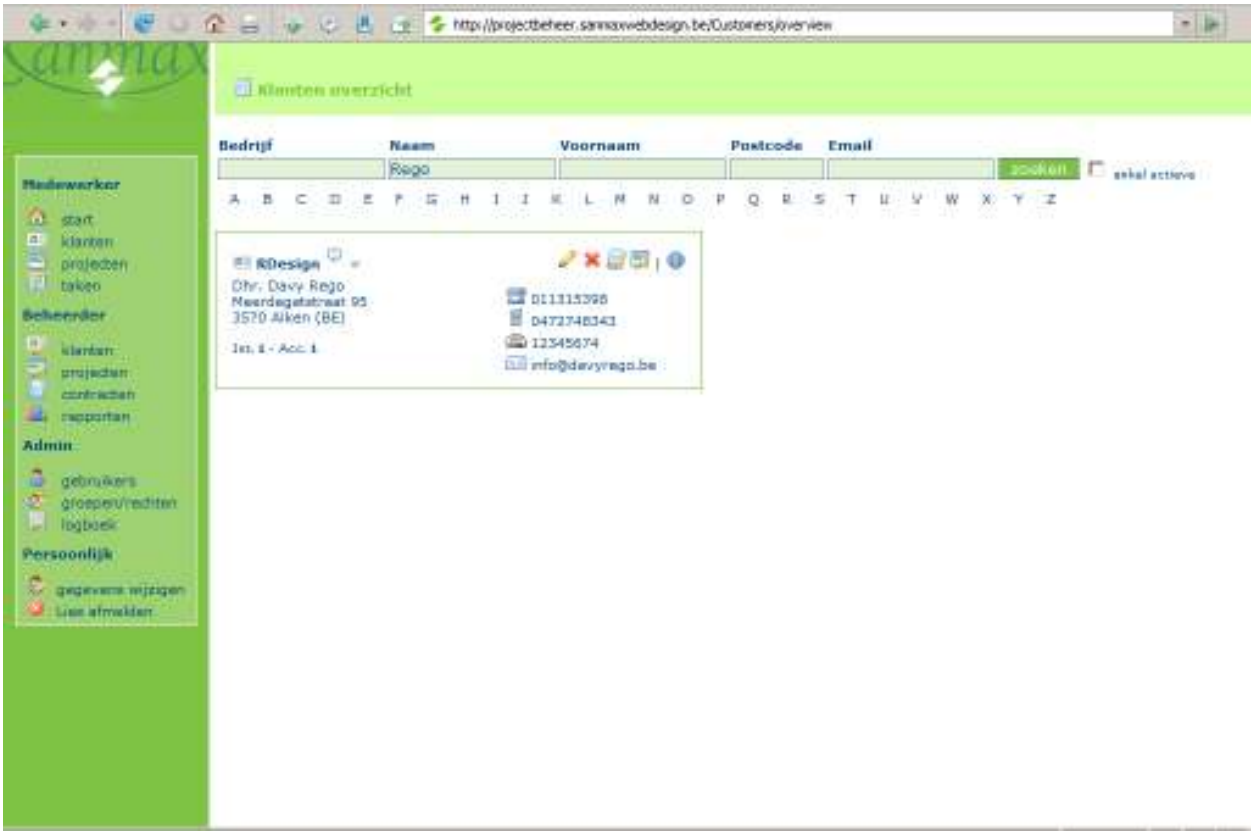
8.4 Printscreens

Enkele printscreens uit de ontworpen PIS-tool:

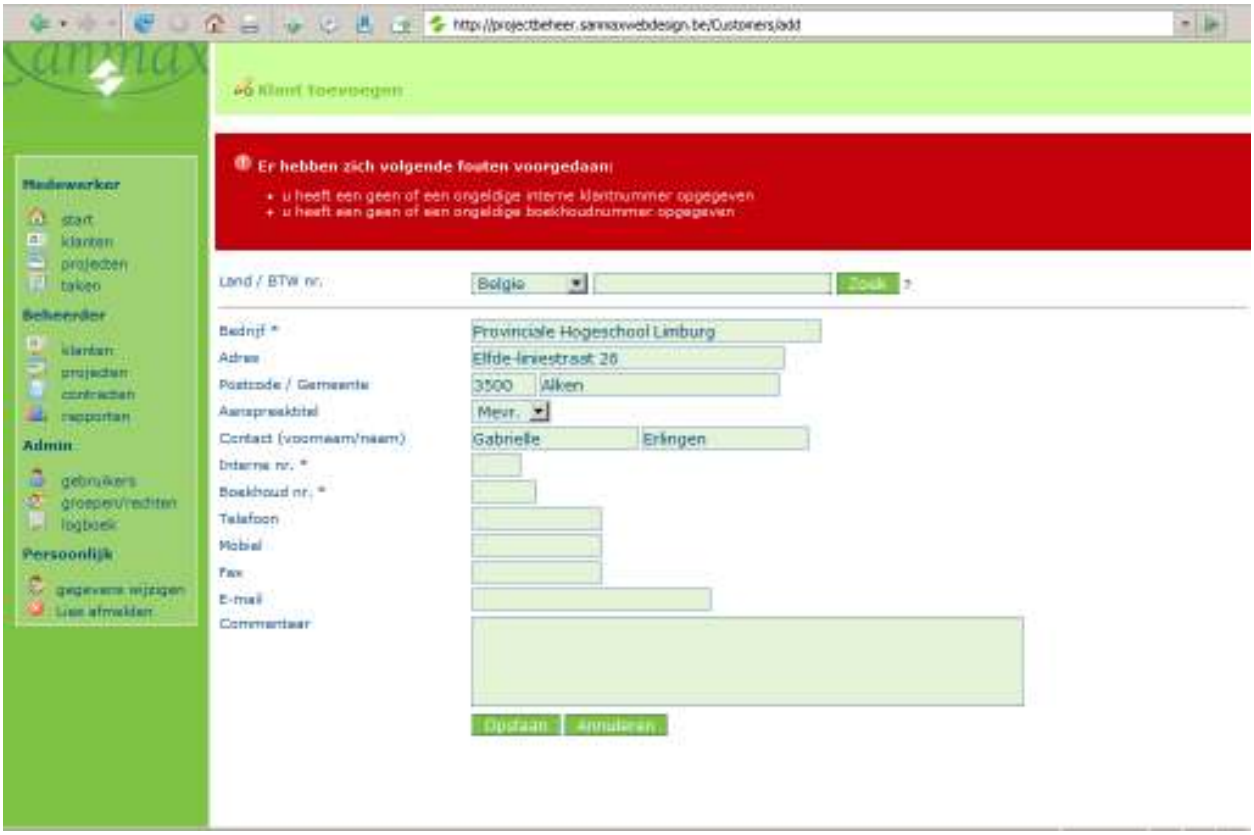


Het opstellen van een rapport voor een klant gedurende een periode, met hieronder het resultaat.

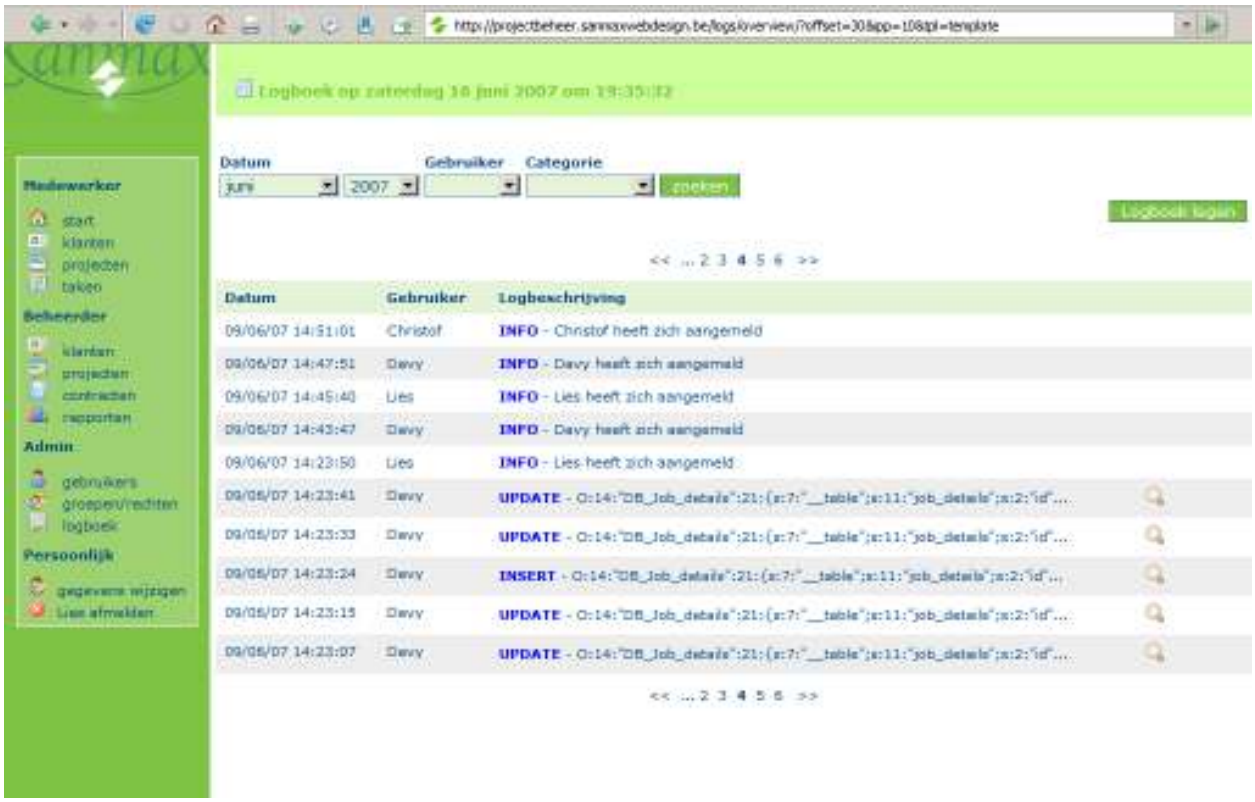




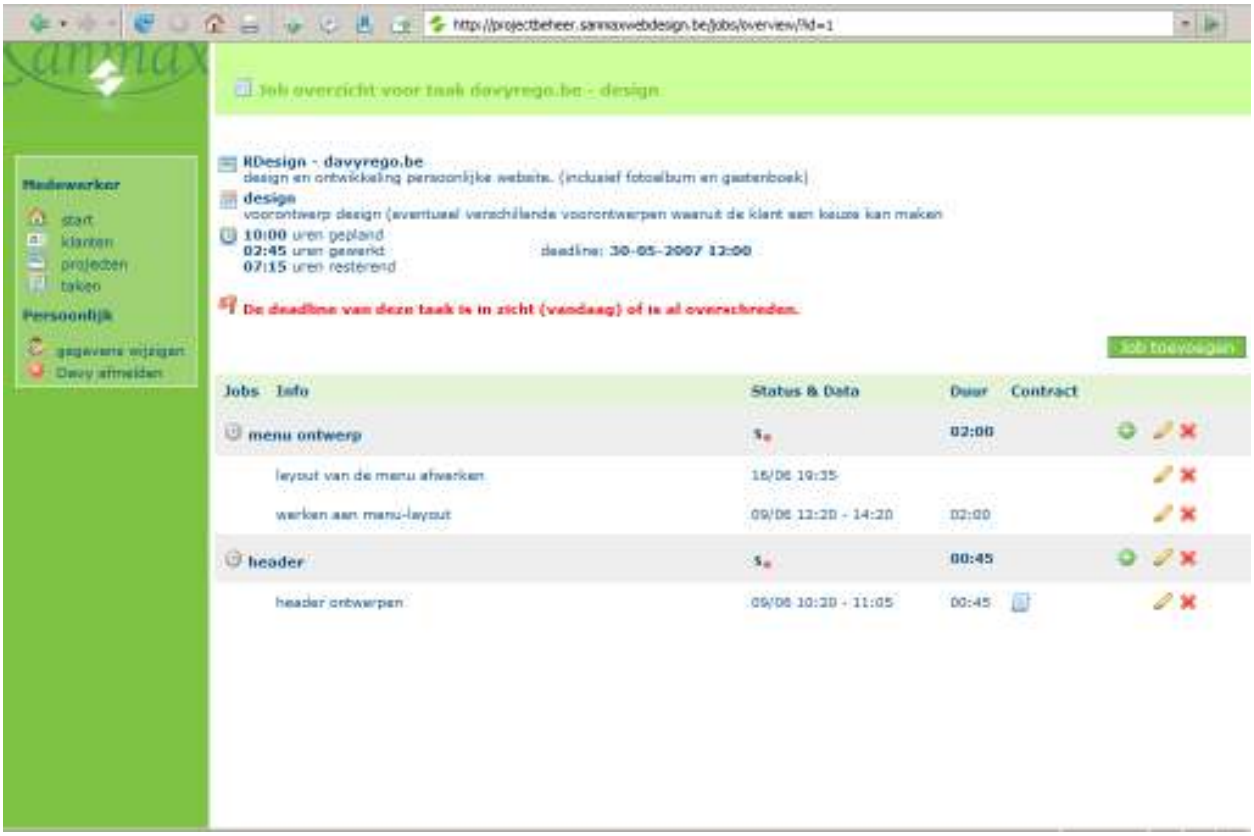
Het zoeken van een klant op de overzichtspagina met behulp van de zoekbalk.



Het proberen toevoegen van een nieuwe klant door een administrator. Merk de foutmeldingen op van de invoercontrole.



Het logboek bekeken door een administrator. Om vertraging te voorkomen worden de logs per tien getoond en kan er doorheen gebladerd worden.



Joboverzicht van een medewerker voor een taak met de geposte jobdetails. Er wordt ook een melding gegeven dat de deadline voor deze taak overschreden is.

9 BIBLIOGRAFIE

Elektronische bronnen

Sanmax bvba	http://www.sanmax.be
Wikipedia	http://www.wikipedia.org
PMTool	http://sourceforge.net/projects/PMtool
CaseTalk	http://www.casetalk.com
Visual Paradigm	http://www.visual-paradigm.com
IBM Rational Rose	http://www.ibm.com/software/rational
MySQL	http://www.mysql.com
phpMyAdmin	http://www.phpmyadmin.net
InnoDB	http://www.innodb.com
PostgreSQL	http://www.postgresql.org
Zend Framework	http://framework.zend.com
PEAR	http://pear.php.net
Smarty	http://smarty.php.net
GotAPI	http://www.gotapi.com
Scriptaculous	http://script.aculo.us
JavaScript Object Notation	http://www.json.org

Boeken

Bakem, G., Zwart, J.P. en van der Lek, H.,
Volledig Communicatiegeoriënteerde Informatiemodellering FCO-IM,
Academic Service , Den Haag, 2005, ISBN 9039524181

Rosenfeld, L. en Morville, P.,
Information Architecture for the World Wide Web,
O'Reilly , 2006 , ISBN 0596527349

Gheorghe Lucian, Hayder Hasin en Maia João Prado
Smarty, PHP Template Programming and Applications,
Packt Publishing, 2006, ISBN 190481140X

(Online) Tijdschriften

Site Diagrams: Mapping an Information Space
Withrow Jason
http://www.boxesandarrows.com/view/site_diagrams_mapping_an_information_space

Test Pattern: Model View Controller
Moore Jeff – php|architect (Volume 6 Issue 5 – p46-49)
http://www.phparch.com

Andere publicaties

Cartoons by Randy Glasbergen (http://www.glasbergen.com)