Katholieke Hogeschool Sint-Lieven
Departement Industrieel Ingenieur
Opleiding Elektronica optie informatie- en communicatietechnieken
Gebroeders Desmetstraat 1, 9000 Gent

Geautomatiseerde webwinkel en orderverwerking

Eindverhandeling ingediend tot het behalen van
de graad van Industrieel Ingenieur
en aangeboden door: Dieter Plaetinck

Promotor: dr. ir. Annemie Vorstermans
Copromotor: prof. ir. Werner Verschelde

Academiejaar 2006 - 2007

Hierbij geeft de auteur van deze masterproef aan het bestuur van het departement industrieel ingenieur van KaHo Sint-Lieven de uitdrukkelijke toestemming om dit werk in bruikleen af te staan aan eender welke persoon, organisatie of firma, het ten dienste te stellen van de studenten en het geheel of gedeeltelijk te kopiëren.

Dieter Plaetinck
mei 2007

Woord vooraf

Een masterproef tot stand brengen zonder de hulp van derden is een quasi onmogelijke taak. Langs deze weg zou ik graag enkele mensen bedanken voor hun hulp bij deze opgave.

Vooreerst wil ik mijn promotoren bedanken, namelijk Mevr. Annemie Vorstermans, voor de bijstand en advies bij de vele aspecten die aan bod kwamen bij het tot een goed einde brengen van dit project, en Dhr. Werner Verschelde voor zijn nuchtere kijk, relativerend vermogen en objectief advies.

Verder wil ik Lieven bedanken voor zijn hulp die zeker heeft bijgedragen tot de kwaliteit van deze scriptie.

Ik wil ook mijn ouders bedanken, zonder wiens steun deze opleiding, laat staan deze masterproef nooit mogelijk zou geweest zijn.

En tenslotte de Open Source gemeenschap voor de ontwikkeling en ondersteuning van allerlei software zonder dewelke dit eindwerk - net zoals menig ander ICT-project - nooit even succesvol kon beëindigd worden.

Abstract

De applicatie die in dit eindwerk wordt ontwikkeld, bestaat uit meerdere onderdelen: enerzijds de webwinkel naar de klanten toe, en anderzijds de interface voor de beheerder, welke hier naadloos moet bij aansluiten.
Aanbiedingen in de webwinkel moeten eenvoudig kunnen beheerd worden, en binnengekomen bestellingen moeten zo automatisch mogelijk verwerkt worden. Bovendien moet het voor de beheerder eenvoudig zijn om leveringen te plannen, gebaseerd op de bestellingen.
Allereerst zal onderzocht worden welke architectuur voor deze toepassing het meest geschikt is. De oplossing zal gekenmerkt worden door het gebruik van moderne programmeerpatronen en technologieën zoals raamwerken en data-abstractielagen.
Bovendien moet onderzocht worden op welke manieren de veiligheid van de applicatie verhoogd kan worden, zodat een oplossing kan geïmplementeerd worden waarbij de veiligheid en privacy van de gebruiker zo goed mogelijk verzorgd wordt.
Hierbij komt ook een onderzoek naar de werking van het semantisch web en hoe het aspect *trust* moet aangepakt worden in de context van deze technologieën.
Ook het kiezen van een degelijk hardwareplatform is een aspect van dit eindwerk. Kwaliteiten zoals prestaties, mogelijkheden, prijs en veiligheid moeten tegen elkaar afgewogen worden zodat de beste keuze kan gemaakt worden.

Trefwoorden: Webwinkel raamwerk programmeerpatroon semantisch web

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Opdrachtomschrijving

Plaetinck D. Vleeshandel NV[1] is een groothandel in rundsvlees die geïnteresseerd in het gebruiken van moderne technieken om efficiënter te kunnen werken en hun klanten beter te bedienen. In het kader hiervan hebben zij beslist hun website te vernieuwen en er een webwinkel in onder te brengen.
Deze webwinkel moet een overzicht geven van de huidige aanbiedingen en de mogelijkheid bieden voor klanten om artikelen te bestellen.
Ook moet het voor de beheerder eenvoudig zijn klanten te beheren, bestellingen te verwerken en leveringen te plannen.

1.2 Overzicht

In het hoofdstuk *Analyse* wordt dit project gesitueerd in zijn context en wordt een duidelijk overzicht gegeven van wat de vereisten zijn.

Eens duidelijk is waar het juist om gaat en wat verwacht wordt, wordt in hoofdstuk *Technologieën* uitgelegd welke mogelijke oplossingsmethoden bestaan, hoe ze zich verhouden ten opzichte van elkaar en wordt uitgelegd welke methoden en technologieën gekozen werden, en waarom.

Daarna worden in hoofdstuk *Modellering* enkele modellen uitgewerkt die op een schematische manier de structuur van de applicatie en de entiteiten waarmee gewerkt wordt uit de doeken doen. Onder meer het use-case diagram wordt gebruikt evenals het ER-diagram.

Vervolgens wordt in hoofdstuk *Implementatie* concreet besproken hoe dit project werd geïmplementeerd: er wordt inzicht gegeven in de werking van diverse aspecten van

[1] `http://www.plaetinck.be`

de applicatie en de benodigdheden (bibliotheken, plug-in's,...) worden er besproken.

In het hoofdstuk *Semantisch Web* wordt een kijk genomen naar het semantisch web: wat het is, wat het verband is met deze masterproef en hoe dit project zou kunnen worden uitgebreid om gebruik te maken van, en zich te integreren in het Semantisch Web.

In het hoofdstuk *Besluit* wordt tenslotte een terugblik genomen op de masterproef van aanvang tot eind en worden de ervaringen en bedenkingen geplaatst. Bovendien worden in dit hoofdstuk enkele verbeteringen en aanvullingen besproken.

Achteraan deze scriptietext is een cd-rom bijgevoegd met onder andere alle programmeercode en deze scriptie in pdf-formaat.
Bovendien kunnen verdere aanvullingen gevonden worden op
`http://dieter.plaetinck.be/masterproef`.

Hoofdstuk 2

Analyse

2.1 Inleiding

In dit hoofdstuk wordt meer uitleg gegeven over het bedrijf Plaetinck D. Vleeshandel NV. Ook wordt uit de doeken gedaan welke doelstellingen vooropgelegd worden, zowel op gebied van functionaliteiten als werkwijzen.
Bovendien wordt besproken welke middelen hiervoor ter beschikking zijn.

2.2 Situering

Zoals eerder vermeld is deze masterproef uitgevoerd in opdracht van
het bedrijf Plaetinck D. Vleeshandel NV. Dit bedrijf is een groothandel in rundsvlees. Hun meeste klanten zijn warenhuizen, slagers en restaurants.
Om de klanten zo goed mogelijk te kunnen bedienen is gekozen om een webwinkel te gaan gebruiken. Hierop zouden alle artikelen op getoond worden, zodat de klanten deze kunnen bestellen. De beheerder kan dan de bestellingen afhandelen en leveringen plannen.

2.3 Beginsituatie

2.3.1 Infrastructuur

Deze elementen waren beschikbaar om te gebruiken voor dit project.

- *éé*n voldoende krachtige werkstation, verbonden met het internet via een ADSL verbinding.
- webhosting: een account op een shared hosting omgeving, met daarop de website zoals deze was voor aanvang van dit project. Het systeem is gelokaliseerd in een datacenter te Brussel en is publiek bereikbaar via het internet. Er is beschikking over 100MB opslagruimte, 5GB/maand verkeer en 5 mysql databases. Er is

ondersteuning van cgi, perl en php (versie 4) scripttalen en werd gehuurd voor 70 euro per jaar.

2.3.2 Website

De website was origineel ontworpen om te fungeren als een online visitekaartje.
Alhoewel deze nieuwe klanten wel informeerde over het bedrijf, betekende ze voor bestaande klanten geen enkele meerwaarde en voldeed ze geenszins aan de eisen die nu gesteld worden.

2.3.3 Manieren van werken

Aanbiedingen

Van de meeste artikelen werden de prijzen regelmatig mondeling, telefonisch, per fax, ... bekend gemaakt. Aangezien de vleessector een vakgebied is waar prijzen vaak schommelen leidt dit tot werk dat niet alleen veel tijd in beslag neemt, maar bovendien veel ruimte laat voor fouten (niet iedereen is steeds op de hoogte van de prijzen van alle artikelen). Bovendien is het heel gebruikelijk dat klanten telefoneren om zich in te lichten over de prijzen en systematisch pogen een korting te verkrijgen zodat vaak op het moment van de bestelling een unieke prijs wordt toegekend aan een bepaald artikel. Standaardisatie is hier ver te zoeken. Wel is het zo, dat de klanten ingedeeld worden in verschillende groepen (bvb restaurants, groothandels, ...) wat toch min of meer een bepaalde overeenstemming in prijzen oplevert.

Bestellingen

Bestellingen worden momenteel telefonisch, mondeling, per e-mail of geschreven op papier aangevraagd. Vervolgens worden ze door de verantwoordelijke manueel ingevoerd in een rekenblad-programma. Het is zo dat niet altijd alle artikelen kunnen geleverd worden, vandaar dat bestellingen altijd moeten vergeleken worden met de huidige voorraadgegevens en andere bestellingen. Het komt vaak voor dat er contact wordt opgenomen met de klant om de bestelling te wijzigen. Dit is een belangrijk aspect dat ook in de nieuwe oplossing vlot zal moeten geïmplementeerd zijn.

Stockbeheer

Het stockbeheer verloopt volledig manueel. Bij het nakijken van bestellingen komt het zelfs vaak voor dat er naar de frigo's moet gegaan worden om te kijken hoeveel en welke artikelen er nog aanwezig zijn.
Een elektronisch systeem hiervoor voorzien zou een prima project op zich zijn, maar valt buiten het bestek van dit eindwerk.

Leveringsplanning

Planning van leveringen is een gebied waar ICT een degelijke bijdrage kan doen, maar dat is niet genoeg. De uiteindelijke levering hangt immers af van verschillende factoren: de openingsuren van de klant, de ultieme leveringsdag van een bestelling (zoals gekozen door de klant), de beschikbaarheid van chauffeurs, en vooral ook het bestaan van andere bestellingen van andere klanten uit dezelfde regio. Momenteel wordt dit ook volledig manueel geregeld door de bestellingen te groeperen in een rekenblad. De beschikbaarheid van chauffeurs is iets dat moeilijk elektronisch geregeld kan worden, aangezien deze mensen ook andere taken hebben, en hun planning vaak kort op voorhand bepaald wordt (afhankelijk van oa binnengekomen bestellingen).

2.4 Vereisten

De vereisten die gesteld worden aan de webapplicatie zijn als volgt:

2.4.1 Registratiesysteem voor klanten

Bedrijven die willen gebruik maken van de winkel moeten zich kunnen registreren. Na succesvolle registratie moeten zij dan kunnen inloggen als klant.
Echter, gebruikers die zich registreren moeten eerst een zogenaamde "screening" doorgaan vooraleer ze effectief toegelaten worden in het systeem.
Er moet immers steeds zekerheid zijn omtrent de solvabiliteit van bedrijven waarmee gehandeld wordt, en ze moeten een juiste categorie toegewezen worden.
Dit gebeurt manueel door de beheerder van het systeem: hij moet dus kunnen zien welke bedrijven zich geregistreerd hebben en de gegevens van deze bedrijven kunnen opvragen zodat de gegevens kunnen nagetrokken worden.

2.4.2 Leveringsvoorkeuren van klanten

Klanten moeten een profiel kunnen aanmaken, waarop ze hun openingsuren en andere voorkeuren moeten kunnen instellen. Per bestelling moeten ze bovendien een dag kunnen specifiëren waarop ze een de bestelling ten laatste willen ontvangen. Bij het opstellen van leveringsschema's moet met deze factoren rekening worden gehouden.

2.4.3 Beheer van de webwinkel

Het beheer van de webwinkel houdt in: artikelen in de winkel plaatsen, er foto's aan toewijzen, de prijzen instellen en aanpassen.
Maar ook het klantenbeheer hoort hierbij: klanten toewijzen aan een bepaalde categorie, instellingen wijzigen en accounts (de)activeren.

2.4.4 Aankopen van artikelen

Klanten moeten - na registratie - de mogelijkheid hebben om bestellingen te plaatsen van *één* of meerdere artikelen.
Bovendien moet het ook mogelijk zijn voor de beheerder om bestellingen in te voeren.

2.4.5 Verwerking van bestellingen

De beheerder moet overzichten kunnen krijgen van bestellingen. Bovendien moet het duidelijk zijn wat het totaal aantal bestelde exemplaren per artikel is.
Een aspect, eigen aan de vleesindustrie, is dat bestellingen niet sluitend zijn: de beheerder overloopt de bestelling, neemt eventueel contact op met de klant en pas na bevestiging door de beheerder is de bestelling definitief.

2.4.6 Planning van leveringen

Afhankelijk van enkele externe factoren (zoals de beschikbaarheid van chauffeurs) moet de beheerder dan een planning kunnen opstellen door *één* of meerdere chauffeur(s) de levering toe te wijzen van *één* of meerdere bestelling(en).

2.4.7 Aanbiedingen

Er zijn twee soorten aanbiedingen:

- vaste.
- unieke.

Vaste aanbiedingen zijn normaliter continu beschikbaar, maar kunnen in zeldzame gevallen de status 'out of stock' gegeven worden.
Unieke aanbiedingen daarentegen zijn tijdelijk en krijgen de status 'gereserveerd' of 'verkocht' wanneer een bestelling hierop geplaatst resp. goedgekeurd wordt.

2.4.8 Werken met foto's

Foto's moeten kunnen toegewezen worden aan aanbiedingen. Ook moeten ze achteraf makkelijk gewijzigd kunnen worden. Het proces van foto's uitlezen, vewerken[1] en aan aanbiedingen koppelen moet op een eenvoudige wijze verlopen.

2.4.9 Prijzen

Unieke aanbiedingen zijn steeds promoties. Dit houdt in: een prijs die niet alleen laag is, maar ook dezelfde is voor elke categorie van klanten.

[1] Geschikt maken voor het web: grootte aanpassen,compressie toepassen, naam wijzigen,...

Bij vaste aanbiedingen ligt dit anders: deze kunnen ook in promotie gezet worden (een lage prijs, dezelfde voor iedereen), maar gewoonlijk is dit niet het geval. De prijsbepaling van vaste aanbiedingen gebeurt standaard via een speciaal prijzensysteem. Elke klant behoort tot een categorie (bvb slagers, restaurants, groothandels,..) en aan al deze categorieën moet een eigen set van prijzen toegewezen worden. Dit impliceert dat het voor de beheerder het eenvoudig moet zijn om makkelijk de verschillende categorieën en prijzen te beheren. Bovendien wil dit zeggen dat bezoekers die niet ingelogd zijn geen prijs zien bij heel wat artikelen.

2.4.10 Platform

De bestaande hardwarevoorzieningen moeten aangepast of vervangen worden om te voldoen aan de vereisten van de applicatie.

2.4.11 Budget

Aangezien op een relatief korte termijn (enkele jaren a 10 jaar) gestopt zal worden met het bedrijf, betekent dit dat het budget beperkt wordt en eventuele investeringen doordacht moeten gebeuren.

Hoofdstuk 3

Gebruikte technologieën

Dit hoofdstuk verduidelijkt de gebruikte methoden en technologieën, en verklaart de gemaakte keuzes.
Zo komen ondermeer aan bod:

- bestaande webwinkels
- de LAMP-stack
- het CakePHP raamwerk
- de TinyMCE editor
- de Prototype en Scriptaculous bibliotheken
- de Xfce desktop omgeving
- de Imagemagick toolset.

3.1 Vooraf

Vooraleer de technologieën op zich te bespreken is het nodig wat randinformatie te bespreken die een grote invloed gehad hebben op de gemaakte keuzes.

Zowel tijdens de ontwikkeling als in de produktieomgeving werd zoveel mogelijk gebruik gemaakt van gratis te verkrijgen open source software.
De voordelen hiervan zijn de geringe kost, grote ondersteuning door de gemeenschap (alhoewel dat afhangt van project tot project), en de zekerheid dat een produkt niet zomaar stopt met bestaan of vervangen wordt door iets anders. (de zogenaamde "vendor independence"[13])
Voor meer over deze voordelen, zie ook "the economics of open source".[28]
Bovendien is de kwaliteit vaak evenwaardig aan betalende alternatieven. Deze eigenschappen hangen echter af van project tot project en worden dan ook voor elk geval apart bekeken en overwogen.

Er is ook gekozen om - waar mogelijk - gebruik te maken van bestaande software, om zodoende het wiel niet onnodig opnieuw uit te vinden.

3.2 Bestaande webwinkels

Eerst en vooral is gekeken naar enkele bestaande webwinkels.
Onder andere de volgende webwinkels zijn onderzocht:

- Compiere [1]
- osCommerce [2]
- ZenCart [3]
- Extropia Webstore [4]
- Dstore [5]
- bakesale [6]

Wat opviel, is dat *alle* geteste webwinkels te complex zijn - zowel in termen van functionaliteit als de gebruikte code - en vaak ook zijn opgebouwd zonder technieken zoals data-abstractielagen. Bovendien zijn deze producten niet gemaakt om te integreren in een website.
Voor dit project, waarbij de webwinkel moet geïntegreerd worden in een nieuwe website en zowiezo moet aangepast worden aan de specifieke vereisten zijn bestaande webwinkels dus ongeschikt en is er dus voor gekozen om zelf een applicatie te bouwen met behulp van een raamwerk.

3.3 De programmeertaal

Bij aanvang van een nieuw project is *één* van de eerste dingen waarbij moet worden stilgestaan de keuze van de taal die zal gebruikt worden.

3.3.1 Verschillende mogelijkheden

Voor het bouwen van een webapplicatie zijn er enkele vaak gebruikte talen. Deze zijn:

- ASP: een taal ontwikkeld door Microsoft, vaak aangewend met het .NET raamwerk. (ASP.NET)

[1] een ERP systeem waarbij de webwinkel maar een klein aspect is. `http://www.compiere.org/`
[2] de meest gebruikte open source webwinkel. `http://www.oscommerce.com/`
[3] `http://zencart.org/`
[4] `http://www.extropia.com/`
[5] `http://sourceforge.net/projects/dstore/`
[6] `http://bakesale.jeepen.net`

- PHP [7] is een open source scripttaal voor het web.
- Ruby is een open source programmeertaal.
- Java (j2se/j2ee/jsp) is in eerste instantie een programmeertaal voor desktopapplicaties maar wordt ook aangewend op het web.

ASP

ASP wordt ontwikkeld door Microsoft, wat een krachtig bedrijf is en dus een goede toekomstverzekerdheid impliceert. Dankzij het .NET raamwerk (ASP.NET) kunnen relatief snel applicaties ontwikkeld worden.
Er is professionele ondersteuning mogelijk maar er is ook een grote behulpzame gebruikersgemeenschap.
Gebruik van deze omgeving impliceert echter dat er een Windows besturingssysteem moet gebruikt worden met ondersteuning voor IIS. Dit betekent een meerprijs. De api van .NET is echter wel publiek en er is een alternatieve gratis (open source) omgeving voor beschikbaar [8] die ASP.NET applicaties kan draaien, maar hiervoor is echter noch voldoende ondersteuning, noch hosting voor beschikbaar.

PHP

PHP is de vrucht van het werk van een gemeenschap die zowel particulieren als bedrijven huisvest. [9].
Er is een degelijke open source implementatie voor beschikbaar: de "mod_php"module voor de Apache webserver. Hierdoor is er ondersteuning voor veel verschillende platformen (Windows, Linux, BSD,. . .)
Bij deze module wordt PHP-code bij elke paginarequest geïnterpreteerd en uitgevoerd. Er zijn echter wel zogenaamde "optimizers" beschikbaar zijn die objectcode in het geheugen houden. Deze introduceren echter een meerprijs en worden daarom vaak niet gebruikt.
PHP is in eerste instantie ontwikkeld als eenvoudige scripttaal voor het web, maar heeft met de tijd meer functionaliteiten meegekregen zoals exception handling, degelijke object-oriëntatie, abstractie van databanken enz.
PHP is erg populair [10] wat te merken is aan de grote gebruikersgemeenschap.

Ruby

Ruby is een taal die dankzij het "Ruby On Rails" raamwerk in een stijgende lijn aangewend wordt voor de ontwikkeling van web-applicaties. De taal is beknopt en krachtig en dankzij het raamwerk is het vrij eenvoudig complexe webapplicaties te

[7] PHP: Hypertext Preprocessor `www.php.net`
[8] Het Mono project `http://www.mono-project.com`
[9] `http://php.net/links.php`
[10] statistieken van zoekmachines: `http://www.tiobe.com/tpci.htm`

schrijven. [12] Ruby code wordt bij de eerste paginarequest geïnterpreteerd en wordt vanaf dan als objectcode in het geheugen gehouden. Er is echter nog weinig hosting voor te vinden.

Java

Java is ontwikkeld als taal voor portable (desktop/workstation) applicaties (j2se/j2ee), wat bereikt wordt door het gebruik van de virtuele machine. Java beschikt over uitgebreide bibliotheken met ondersteuning voor geavanceerde technologieën zoals transacties.
De jsp technologie wordt vaker aangewend voor "lichtere" taken zoals templating.[11]

3.3.2 De keuze: PHP

Er is uiteindelijk voor PHP gekozen omdat dit een taal is die speciaal ontwikkeld is voor de serverside-logica van webapplicaties.
Met PHP kunnen dmv vrij beknopte code degelijke resultaten behaald worden. Bovendien is er veel php-hosting beschikbaar, en dit tegen prijzen die een stuk lager liggen dan bij de alternatieve technologieën.
Ondanks het feit dat PHP in normale omstandigheden bij elke paginarequest opnieuw geïnterpreteerd moet worden (bij gebruik van de gratis interpreter) geeft dit in praktijk geen problemen. Nog een groot voordeel aan PHP is de overweldigende gebruikersgemeenschap en beschikbare code.

Ondanks het feit dat de recentse versie - PHP 5 - enkele significante verbeteringen heeft gebracht (zoals de verbeterde implementatie van object-oriëntatie en verhoogde uitvoeringssnelheid) is toch gekozen om zoveel mogelijk in PHP 4 te blijven programmeren. De redenen hiervoor zijn als volgt

- De grootste minpunten aan PHP 4 - zoals zichtbaarheid van membermethoden - worden opgevangen door het raamwerk (meer uitleg over het raamwerk in de volgende sectie): voor membermethoden die _als() __volgt() zijn gedefinieerd wordt door het raamwerk de zichtbaarheid private respectievelijk protected geëmuleerd.
- Heel veel hostingproviders blijven trouw aan PHP 4 omdat dit nog altijd een degelijke, stabiele versie is.
- Hostingproviders kiezen hun hardware op zulke manier dat de prestaties er niet méér onder lijden dat een oudere versie wordt gebruikt.
- Bepaalde delen van de code die ontwikkeld werd voor dit project werd vrijgegeven als open source software.[12] Veel gebruikers kiezen nog PHP 4.

[11] Het op een dynamische, modulaire manier opbouwen van de presentatielaag.
[12] De authenticatiecomponent. Zie sectie *Authenticatie* van hoofdstuk *Implementatie* op pagina 33

3.4 Het Raamwerk

3.4.1 Inleiding

Een raamwerk is een verzameling van softwarecomponenten die kunnen gebruikt worden bij het bouwen van een applicatie. Echter de meeste raamwerken bieden ook een set van afspraken en logica die wordt uitgevoerd waar eigen code kan in "tussengevoegd" worden. Op deze manier wordt het ontwikkelen van applicaties nog makkelijker dan bij het louter beschikbaar stellen van code. Het is in deze context dat het begrip "raamwerk" in deze scriptie wordt gebruikt.

3.4.2 De verschillende mogelijkheden

Voor PHP bestaan verschillende raamwerken. De meest voorkomende zijn:

- Zend Framework
- Symfony
- CakePHP
- Code Igniter

Zend framework

Het Zend framework wordt ontwikkeld door Zend, het bedrijf dat meest betrokken is bij de ontwikkling van PHP.
Het is eerder een collectie van componenten dan een echt raamwerk.[22] Het Zend framework beschikt over allerlei modules met functionaliteiten zoals caching, validatie, pdf-generatie,... Voor mogelijkheden die ontbreken - zoals data-abstractie en templating - kunnen 3rd-party plugins gebruikt worden.
Echter: het onbreken van ondersteuning voor PHP4 (de versie die het vaakst wordt gebruikt) is een vrij groot minpunt, evenals het gebrek aan code die helpt bij het ontwikkelen van applicaties (zoals bij een echt raamwerk).

Symfony

Symfony is vergelijkbaar met het Zend framework in die zin dat het veel mogelijkheden biedt. Daarbovenop is het ook een volwaardig raamwerk. Het bevat echter zoveel mogelijkheden die doorgaans zelden worden gebruikt[14] Bovendien zorgt het overmatig gebruik van configuratiebestanden[13], dat het een oninteressante keuze wordt.
Symfony heeft ook vele "vendor dependencies" zoals PEAR[14], een bibliotheek die vaak in de negatieve belangstelling komt door zijn vaak wijzigende api en zogenaamde "bloat".

[13] `http://www.symfony-project.com/book/trunk/19-Mastering-Symfony-s-Configuration-Files`
[14] de PHP Extension and Application Repository, `http://pear.php.net/`

CakePHP

CakePHP hanteert eerder de "less is more" filosofie en probeert alles zo bescheiden mogelijk te houden. Dit maakt het raamwerk een stuk lichter dan bvb Symfony, maar dit impliceert ook dat de mogelijkheden beperkter zijn: zo bevat CakePHP wel een data-abstractielaag maar geen volwaardige ORM-laag[15] zoals Symfony. Ondersteuning voor internationalisatie (i18n) staat ook nog niet zo ver.
CakePHP past ook het principe van "convention over configuration" toe wat resulteert in slechts weinig configuratie-overhead. Verder bevat het wel mogelijkheden zoals templating, ondersteuning voor zowel PHP4 als PHP5, caching,...

Code Igniter

Code Igniter is - net zoals CakePHP - een meer lichtgewicht raamwerk. Het ondersteunt PHP versies 4 en 5 bevat de meest nodige functionaliteiten. Er is echter geen ondersteuning voor associaties tussen entiteiten.

Content Management Systems

Er bestaan ook enkele zogenaamde "Content Management Systems" waarvan de meest voorkomende Drupal en Joomla zijn. CMS-systemen zijn niet ontworpen om te helpen bij het ontwikkelen van applicaties, maar zijn applicaties die ontwikkeld zijn voor het tonen van pagina's en andere informatie.
Beide hebben wel een API die gebruikt kan worden om de applicatie verder uit te breiden met op maat gemaakte logica. (zoals dit eindwerk vereist)

3.4.3 De keuze: CakePHP

Inleiding

Het CakePHP raamwerk is gekozen omdat het het beste geschikt is voor dit eindwerk: het biedt voldoende mogelijkheden om op een relatief snelle manier een degelijke applicatie te bouwen, doch het doet niet meer dan nodig en dus "loopt het niet in de weg". De beschrijving van CakePHP op de officiële website[16] gaat als volgt:

> Cake is a rapid development framework for PHP which uses commonly known design patterns like ActiveRecord, Association Data Mapping, Front Controller and MVC. Our primary goal is to provide a structured framework that enables PHP users at all levels to rapidly develop robust web applications, without any loss to flexibility

[15] een ORM, of *Object-Relational-Mapping* laag zet object-instanties om naar databankrecords en omgekeerd
[16] `http://www.cakephp.org`

Voor deze masterproef is gebruik gemaakt van versie 1.2. Dit is een versie die nog volop in ontwikkeling is, en waarvan tot nu toe enkel een alpha-release is uitgegeven. Dit houdt in dat de API nog lichte veranderingen ondergaat en dat sommige functionaliteiten nog niet volledig ontwikkeld zijn. De basis is echter dezelfde als de stabiele versie 1.1. Ook wordt de broncode steeds ontwikkeld onder het toezicht van, en met hulp van de gebruikersgemeenschap. Bijgevolg kan met vrij grote zekerheid gezegd worden dat er geen noemenswaardige bugs of lekken in zitten.
Het CakePHP raamwerk kan gratis gebruikt worden onder de voorwaarden van de MIT open source licentie.

Om een beter beeld te krijgen over het raamwerk worden hieronder de mogelijkheden en tekortkomingen uitvoeriger besproken.

Mogelijkheden

Model-View-Controller: CakePHP past de MVC-opbouw toe. Deze structuur brengt vele voordelen zoals flexibiliteit, modulariteit en robuustheid.
In CakePHP zijn modellen en controllers geïmplementeerd als PHP-objecten, Views als bestanden met meestal een html of XML indeling. Bij een pagina-oproep instantieert de dispatcher de juiste controller, en roept die de juiste actie van de controller op. (Dit is een publieke methode). De controller voert de nodige domeinlogica uit en doet dan een *render*: Dit is het laden van de juiste view-bestanden, het integreren van deze bestanden in de layout[17], het uitvoeren van de logica die erin staat, en de output naar de gebruiker terugsturen.
Er kunnen bovendien aparte modules gebouwd worden die bepaalde functionaliteiten leveren aan views of controllers (de zogenaamde *helpers* respectievelijk *components*).
Uiteraard kan er met overerving gewerkt worden van modellen en controllers. Om bij views de code netjes te organiseren en herbruikbaarheid te realiseren wordt gebruik gemaakt van zogenaamde *elements*, dit zijn herbruikbare componenten die door verschillende views kunnen worden weergegeven. Met deze manier van werken (elements, views, layouts) wordt templating gerealiseerd.

Validatie: Het is eenvoudig om validatie toe te passen bij het ingeven en bewerken van gegevens (html-formulieren): er kunnen datatypes afgedwongen worden maar ook specifiekere dingen zoals emailadressen of datums. Bij het niet invullen van verplichte velden, of verkeerdelijk ingevulde velden, kan de gebruiker terug hetzelfde formulier getoond worden met de melding wat er mis is met de ingave.

Ajax: Er zijn helpers aanwezig om het werken met Ajax (zie hoofdstuk *Implementatie* op pagina 28) te vereenvoudigen. Hiervoor zijn wel de Scriptaculous en Prototype bibliotheken nodig (zie verder)

[17] Dit is de algemene definitie van een pagina. (met inbegrip van doctype declaratie, het "head"gedeelte van de html pagina, enz.)

Data Sanitization: Dit betekent dat bedreigingen zoals SQL-injecties[21] en XSS [2] worden voorkomen door alle input van de gebruiker op te schonen. (verwijderen van gevaarlijke karakters zoals slashes)

Conventie boven configuratie: CakePHP past het principe van "convention over configuration" toe. Dit wil zeggen dat bepaalde veronderstellingen gemaakt worden over de namen van modellen, tabellen (meervoud, enkelvoud,gebruik van CamelCase, underscores,...) en de locatie van mappen en bestanden. Wanneer hieraan wordt voldaan werkt alles "automagisch". Het is echter steeds mogelijk de configuratie te overschrijven.

Sessies en Request-handling: Dankzij de meegeleverde sessie-component wordt het werken met sessies heel eenvoudig: via een api kan je rechtstreeks gegevens schrijven en opvragen. Alle onderliggende logica wordt automatisch afgehandeld. Er is standaard beveiliging tegen sessie-hijackings.[4]
De meegeleverde request handler is een tool die kan detecteren wat het verwachte antwoord is (een RSS-feed, ajax-antwoord in XML, gewone pagina, ...) en desgewenst antwoorden in het juiste formaat. (de zogenaamde layout wordt automatisch gekozen)

Caching: Het is mogelijk om bepaalde stukken output te cachen, zodat minder code moet uitgevoerd worden, zowel op gebied van databank-queries, als domeinlogica en in de presentatielaag.

Code generatie: Aan de hand van de structuur van een databank kunnen automatisch via het zogenaamde bake-script modellen, controllers en views gegenereerd worden.

Zoekmachine optimalisatie: CakePHP voorziet standaard het gebruik van zoekmachine-vriendelijke adressen.
Een adres zoals `http://www.mijnwinkel.com/kopen/bruine-jas/gele-sjaal` wordt automatisch ontleedt door de dispatcher en de onderdelen van de URL worden via variabelen beschikbaar gesteld binnenin de controller. Dit wordt meer in detail besproken in sectie *opbouw van de URL's* op pagina 29

Test suite: CakePHP bevat een test-suite. Deze is gebaseerd op de SimpleTest [18] suite, de de facto test-suite voor PHP applicaties.

Geen afhankelijkheden: CakePHP is een totaaloplossing. Er zijn geen afhankelijkheden van 3rd-party aanbieders. ("vendor dependencies"). Het is uiteraard altijd mogelijk om de suite uit te breiden, maar dit is geen must en is ook niet echt nodig omdat het raamwerk op zich al veel functionaliteit voorziet.

[18] `http://simpletest.sourceforge.net`

Tekortkomingen

Net zoals andere softwarepaketten heeft CakePHP zijn tekortkomingen. Deze worden hieronder besproken. Voor de effectieve aanpak van deze tekortkomingen wordt verwezen naar het hoofdstuk *Implementatie*. (hoofdstuk 5 op pagina 28)

Authenticatie: CakePHP heeft een wel authenticatie-component maar deze is echter helemaal nog niet voldoende ontwikkeld, daarom moest een eigen systeem ontwikkeld worden.

Authorisatie: CakePHP gebruikt "Acess control lists" om gegevens over gebruikers en hun rechten bij te houden.[19] Dit systeem werkt goed, echter aangezien de gegevens[20] via het mptt-algoritme[27] in de databank worden opgeslaan, is het moeilijk om de software te schrijven die op een eenvoudige manier deze structuur kan beheren. Daarom is gekozen voor een andere oplossing.

Internationalisatie: De ondersteuning voor internationalisatie (het weergeven van dezelfde inhoud in verschillende talen) staat nog niet op punt, dus moest zelf een oplossing bedacht worden om in het Nederlands te werken.

Datalaag: De datalaag is niet zo geavanceerd als ORM-oplossingen zoals propel of hibernate. Er kan via een eenvoudige API wel eenvoudig de juiste data opgevraagd worden, echter het werken met de data gebeurt via arrays en niet met objecten. In praktijk is dit echter niet echt een probleem gebleken.

Transacties: CakePHP biedt nog geen ondersteuning voor transacties. De uiteindelijke versie van 1.2 zou dit moeten ondersteunen maar de functionaliteit is momenteel nog niet uitgewerkt.

3.5 Andere

3.5.1 TinyMCE

TinyMCE [21] is een open source WYSIWG-editor ("what you see is what you get") die zich naadloos integreert in de pagina's van de applicatie. Deze editor is aangewend om het voor de beheerder mogelijk te maken nieuwsberichten op de website te plaatsen en van opmaak te voorzien. De editor is gratis verkrijgbaar onder de LGPL.

[19] `http://manual.cakephp.org/chapter/acl`

[20] De boomstructuren die de permissies beheren van *access requestion objects* die emphaccess controlling objects aanvragen

[21] tinymce.moxiecode.com/

3.5.2 Prototype Javascript raamwerk

Prototype[22] is een JavaScript raamwerk, gratis beschikbaar onder MIT licentie. Prototype bundelt een aantal handige Javascript-methodes en biedt ondersteuning voor Ajax.

3.5.3 Scriptaculous

Scriptaculous[23] is een bibliotheek met uitbreidingen op Prototype (zie 3.5.2). Met dit Javascript raamwerk worden onder andere drag-and-drop-functionaliteit en sorteerbare lijsten mogelijk.

3.5.4 Image Resize Helper

De *Image Resize Helper* [24] is een CakePHP-helper geschreven door een community-lid. Hiermee kunnen on-the-fly de afmetingen van afbeeldingen gewijzigd worden. De code is gratis beschikbaar gesteld (zonder licentie), en is aangepast waar nodig.

3.6 De server

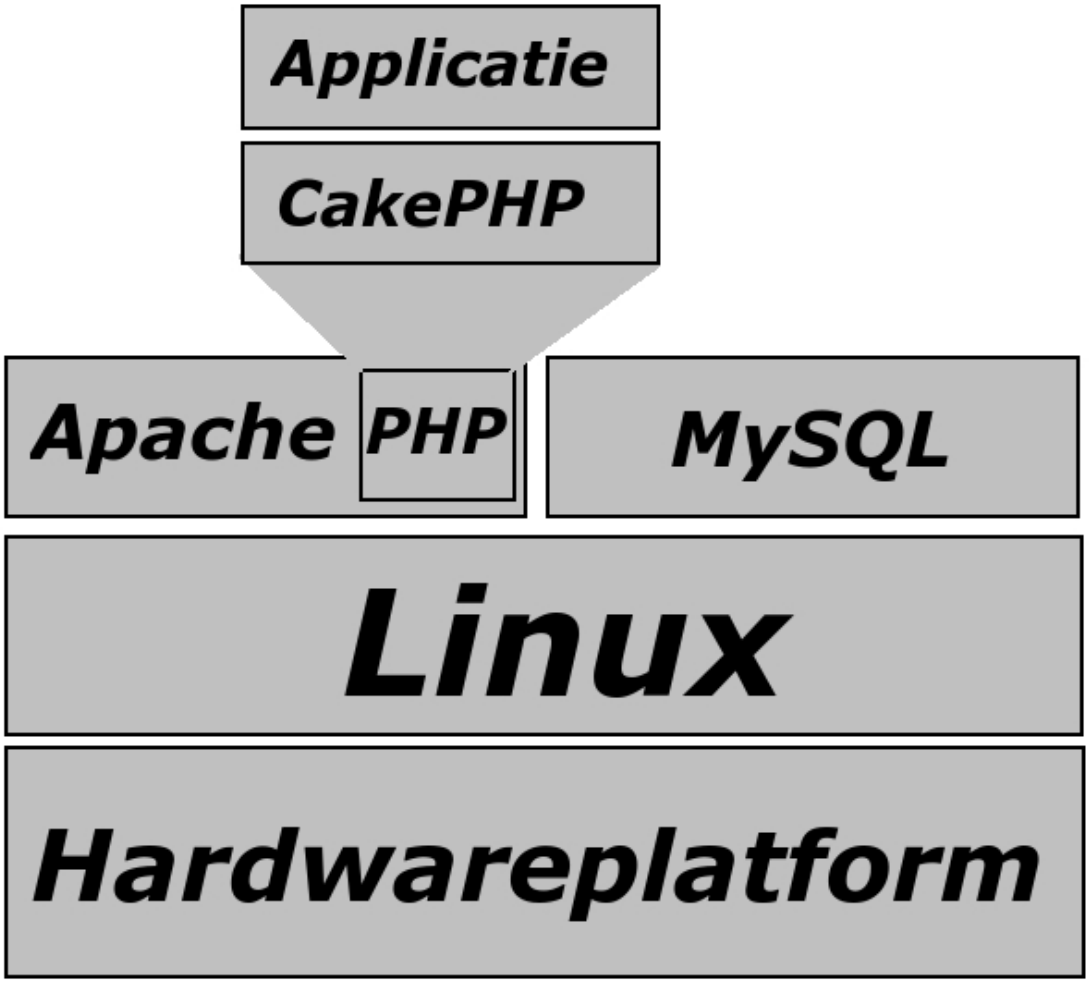
3.6.1 De Lamp-stack

Er is gekozen om de zogenaamde “lamp” stack op de server te gebruiken. Dit is de combinatie van het Linux besturingssysteem, de Apache webserver, de mysql databank, en zoals eerder vermeld: de PHP-taal.
Al deze componenten hebben enkele eigenschappen gemeen: ze zijn gratis, open source en ze hebben stuk voor stuk hun diensten al bewezen heeft in de servermarkt. Bovendien werken ze prima samen.

3.6.2 Shared hosting

In hoofdstuk A op bladzijde 70 wordt berekend wat minimale specficaties van de benodigde server.
Aan de hand hiervan, is gekozen voor een zogenaamde *shared hosting* account met 500MB webruimte en 12GB/maand bandbreedte ter beschikking.
Verdere vereisten zoals de beschikking over een mailserver, https verkeer en dergelijke zijn hierbij ook aanwezig.

[22] http://www.prototypejs.org/
[23] http://script.aculo.us/
[24] http://bakery.cakephp.org/articles/view/image-resize-helper



Figuur 3.1: Structuur van de server

3.7 Het werkstation

3.7.1 Linux, Xfce

Het werkstation werd voorzien van de Linux-distributie Xubuntu.[25]
Aan het besturingssysteem werden enkel de eisen gesteld het beheer van de webapplicatie ermee vlot moet verlopen, en daarvoor is een Linux-gebaseerd systeem prima geschikt. De distributie Xubuntu werd gekozen omdat dit een gebruikersvriendelijk systeem is dat bovendien de Xfce[26] desktop omgeving gebruikt.
Met Xfce's *Thunar volman* is het mogelijk om acties te koppelen aan het optreden van gebeurtenissen zoals het aankoppelen van randapparatuur.
Hiervan werd gebruik gemaakt om de foto's automatisch te verwerken wanneer het digitaal fotoestel werd aangesloten.

3.7.2 Imagemagick

De Imagemagick[27] toolset is een collectie van commandline-tools waarmee afbeeldingen bewerkt kunnen worden. Daarom is deze ook geschikt om te gebruiken in scripts en daarom werd deze dan ook aangewend.

[25] `http://www.xubuntu.org/`
[26] `http://www.xfce.org/`
[27] `http://www.imagemagick.org/`

Hoofdstuk 4

Modellering

Om een inzicht te krijgen in hoe de applicatie werkt, is het eerst nodig te weten hoe ze is opgebouwd. Daarom wordt in dit hoofdstuk de modellering besproken, en in het volgend hoofdstuk de implementatie van de applicatie.
De volgende onderwerpen worden behandeld:

- functionele vereisten (use-cases)
- Entiteiten (ER-diagramma)
- structuur van de applicatie

4.1 Use Cases

Figuur 4.1 op pagina 21 toont de verschillende uses cases. In de volgende secties worden deze meer in detail besproken.

4.1.1 Gast

Gasten zijn personen die de webwinkel bezoeken maar nog niet geregistreerd zijn, of wel geregistreerd zijn maar niet ingelogd.
Gasten moeten de mogelijkheid hebben in te loggen en/of een nieuwe account te registreren. Bij de registratie moeten ze de mogelijkheid hebben hun openingsuren in te stellen (deze kunnen ze echter later nog aanpassen).
Zij moeten ook de mogelijkheid hebben om de aanbiedingen in de winkel te bekijken. Zoals besproken in de analyse mogen zij geen prijzen bij aanbiedingen zien, behalve als deze in promotie staan.
Zij moeten ook andere pagina's met informatie of documentatie kunnen bekijken.

4.1.2 Klant

Klanten: dit zijn personen die ingelogd zijn. Zij moeten de mogelijkheid hebben de webwinkel te verkennen net zoals de gasten en documentatie te bekijken.

Figuur 4.1: Use cases

Echter, voor hen moeten bij alle artikelen de prijzen zichtbaar zijn (uniek voor de groep waartoe ze behoren of promoties). Zij moeten ook kunnen winkelen en dus bestellingen kunnen maken: hiervoor moeten ze artikelen kunnen selecteren uit de winkel, en de hoeveelheid van elk artikel kunnen bepalen. Ook moeten ze per bestelling de uiterste leverdatum kunnen opgeven.
Zij moeten ook overzichten kunnen zien van hun bestellingen, en ze moeten kunnen zien wat de status van elke bestelling is.
Zij hebben een profiel dat ze moeten kunnen beheren (om bijvoorbeeld contactgegevens aan te passen of de openingsuren te veranderen)
Uiteraard moeten ze zich ook kunnen uitloggen.

4.1.3 Beheerder

De beheerder erft heel wat mogelijkheden over van de klant.
Ook hij moet namelijk bestellingen kunnen aanmaken in naam van een klant (indien die bestelling bijvoorbeeld telefonisch is gemaakt) of het profiel van de klant aanpassen.
Hier komen echter een aantal beheerstaken bij:

- klanten: overzicht van de klanten waarop hij moet kunnen zien welke klanten nieuw zijn (en moeten nagetrokken worden), in welke groep elke klant zit, enz. Ook de groepen op zich moeten kunnen beheerd worden: de naam en eigenschappen (bvb geassocieerde prijzen) moet de beheerder kunnen aanpassen per groep.

- bestellingen: de beheerder moet overzichten kunnen krijgen van alle bestellingen, met de mogelijkheid deze bestelling aan te passen en de status ervan te veranderen (naar bijvoorbeeld *in verwerking* of *geleverd*)

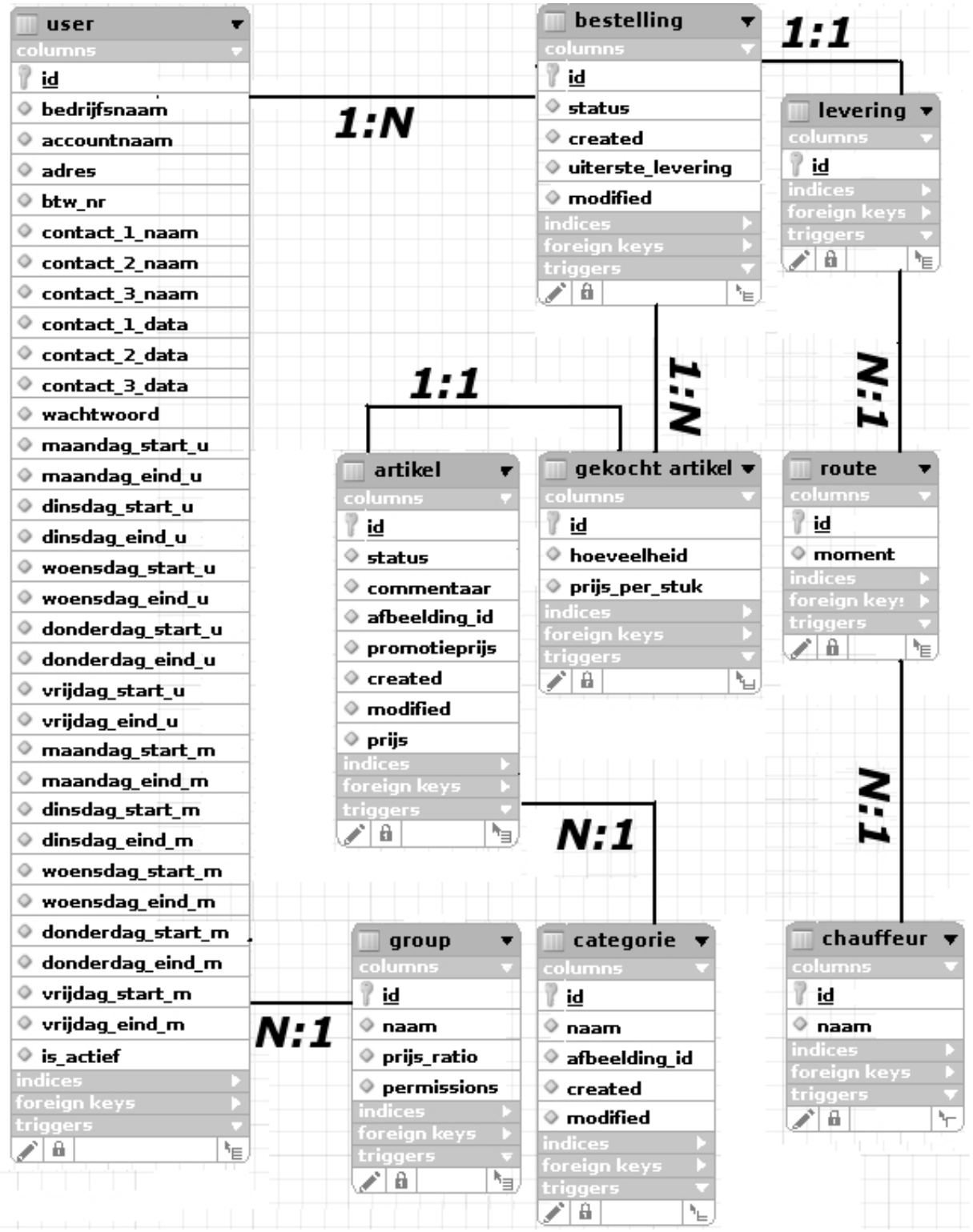
- artikelen: de beheerder moet verschillende categorieën van artikelen kunnen maken en aanpassen, en artikelen aanmaken, er een categorie aan toewijzen en de status per artikel beheren (voorradig, uitverkocht, enz). Ook moet de beheerder afbeeldingen kunnen beheren en koppelen aan artikelen en categorieën.

Tenslotte moet het ook mogelijk zijn voor de beheerder leveringen te plannen.

4.2 Entiteiten

4.2.1 ER-diagramma

Het Entiteiten-Relatie diagramma (figuur 4.2 op pagina 23 toont de relevante entiteiten uit het domeinmodel en de relaties ertussen. Hierna volgt een korte bespreking van de belangrijkste klassen:



Figuur 4.2: Entiteiten-Relatie diagramma

4.2.2 User

De klasse User [1] symboliseert een gebruiker van het systeem. Een gebruiker kan zowel een klant zijn, of de beheerder.

Deze klasse is niet helemaal genormaliseerd. Hier is bewust voor gekozen omdat het het werken ermee vereenvoudigt en het niet echt problemen met zich meebrengt. De eigenschappen zoals openingsuren - welke in een meer genormaliseerde databank in een aparte tabel profielen zou opgeslaan worden - zijn hier bij de gebruiker zelf ondergebracht. Aangezien de relatie tussen een gebruiker en zijn profiel 1:1 is, geeft het geen problemen om dit te doen, en maakt het de verwerking eenvoudiger.
Aangezien klanten zelden meer dan twee telefoonnummers opgeven is ook hiervoor dezelfde aanpak gekozen: meerdere telefoonnummers of faxnummers van *één* of meerdere contactpersonen worden als attributen van deze klasse zelf opgeslaan.

Het attribuut *is_actief* maakt het mogelijk accounts ongeldig te houden tot zolang ze niet goedgekeurd zijn door de beheerder of om accounts achteraf uit te schakelen.

Met de foreign key *klant_category_id* wordt een klant toegewezen aan een instantie van de klasse Group, welke hierna wordt besproken.

4.2.3 Group

Deze klasse zorgt voor de indeling van klanten in categorieën. Elke groep heeft een eigen *prijs_ratio*. Dit is het percentage van de ingestelde prijs van een artikel (behalve bij promoties) dat voor klanten van deze groep moet worden aangerekend.

Het attribuut *permissions* laat ruimte voor een string met permissies voor deze groep. Dit wordt besproken in sectie 5.4 *Authorisatie* op pagina 35.

4.2.4 Artikel

Een artikel is een item dat kan gekocht worden in de winkel. Het moet ofwel een promotieprijs of een gewone prijs ingesteld hebben. Indien beide aanwezig zijn wordt voorkeur gegeven aan de promotieprijs. De gewone prijs is de prijs die verrekend wordt met de *prijs_ratio* van een groep. De promotieprijs is voor iedereen dezelfde. Met het attribuut *afbeelding_id* kan een afbeelding gekozen worden voor een artikel.

Het veld *commentaar* kan gebruikt worden om extra informatie over het artikel in te vullen.

[1] Deze klasse maakt samen met de klasse group deel uit van een herbruikbare component en is daarom Engelstalig. Zie ook sectie *Authenticatie*. (sectie 5.3 op pagina 33)

4.2.5 Categorie

Een categorie dient om artikelen te ordenen. Een categorie kan een naam gegeven worden en een afbeelding toegekend worden.

4.2.6 Bestelling

Een bestelling heeft 3 mogelijke statussen:

- ongeldig
- in verwerking
- geleverd

Ongeldig is de status die een bestelling bij aanmaak wordt gegeven: ze is nog niet goedgekeurd door de beheerder. Na goedkeuring door de beheerder krijgt de bestelling de status *in verwerking*. Een derde status *geleverd* is voorzien, maar niet geïmplementeerd.

Het attribuut *uiterste_levering* wordt gekozen door de klant bij het afronden van de bestelling, en zorgt ervoor dat hiermee kan rekening gehouden worden bij het plannen van leveringen.

4.2.7 GekochtArtikel

De klasse GekochtArtikel koppelt artikelen aan bestellingen. Hier zijn 2 attributen van groot belang:

- hoeveelheid
- prijs_per_stuk

Het attribuut *hoeveelheid* bepaalt het aantal stuks dat van het bepaalde artikel is besteld. Het attribuut *prijs_per_stuk* houdt de prijs bij van een artikel. De reden dat deze hier wordt bijgehouden en niet wordt opgevraagd via het Artikel zelf, is tweeledig:

- de gebruiker kan na bestelling aan een andere groep toegewezen worden
- de prijs van het artikel kan veranderen: het kan in promotie gezet worden, of juist uit promotie gehaald worden, de gewone prijs kan veranderd worden, of de *prijs_ratio* van de groep kan aangepast worden.

In bovenstaande gevallen is het niet gewenst dat de prijs van de bestelling achteraf verandert, daarom wordt de prijs bijgehouden zoals deze was op het moment van de bestelling.

4.2.8 Route

Een route is vrij abstract. Het heeft maar 2 relevante attributen: de *chauffeur_id* om een chauffeur toe te kennen aan een route, en het *moment* specifieert het tijdstip waarop de chauffeur vertrekt.

4.2.9 Levering

Een levering is de klasse die de link legt tussen een bestelling en een route. Een instantie van een levering ontstaat pas wanneer de aflevering van een bestelling effectief wordt ingecalculeerd. Niet wanneer de status van een bestelling op *in verwerking* wordt gezet.

4.2.10 Andere

Afbeelding

De klasse afbeelding is een belangrijke entiteit in het domeinmodel: er is een klasse voor ontworpen (een model in MVC-woordenschat), er is echter geen tabel voor voorzien in de databank. Afbeeldingen worden rechtstreeks op het bestandssysteem opgeslaan. De *id* van een afbeelding is niets anders dan de bestandsnaam zelf. Op die manier kunnen associaties gelegd worden met de klassen Artikel en Categorie. Meer hierover in de sectie *Afbeeldingen* van hoofdstuk *Implementatie* op pagina 44.

Instelling

Instanties van de klasse Instelling worden wel opgeslaan in een tabel in de databank. Deze klasse is echter niet weergeven in het ER-diagramma omdat ze los staat van de andere klassen: er zijn geen associaties met andere klassen, en ze is ook niet relevant voor het domeinmodel.
Ze is echter wel belangrijk in de applicatie: ze bevat “naam-waarde” paren zodat de applicatie kan geconfigureerd worden. Dit wordt verder besproken in sectie *Controle-paneel* van hoofdstuk *Implementatie* op pagina 41

4.3 Applicatie

De applicatie is opgebouwd volgens de MVC-structuur. De diverse voordelen van deze aanpak zijn eerder besproken.
Bovendien bevat het CakePHP raamwerk nog enkele handige functionaliteiten die het werken hiermee aangenamer maken. Voorbeelden hiervan zijn het templating systeem voor de views, maar ook de overerfbare modellen en controllers, waarbij bepaalde functionaliteiten afgehandeld worden in verschillende lagen van de object-hiërarchie (door overerving). Bijvoorbeeld: AppModel erft over van Model en bevat applicatie-specifieke code die geldig is voor alle modellen binnen een applicatie. Model bevat logica zoals het zoeken van alle instanties of het updaten van attributen. Implementatie-specifieke

code (zoals het interageren met databanken) gebeurt in een andere klasse genaamd DboSource

Model Het Model is de interface naar instanties van een bepaalde entiteit. Via het model kan je instanties zoeken, opslaan, bewerken,... Maar Modellen hebben ook (eventueel) enkele membermethoden die logica uitvoeren voor het specifiek model.

View Deze laag is presentatielaag: verschillende html files (die op zich dan weer kunnen refereren naar JavaScript of CSS-bestanden) worden gecombineerd om tot de gewenste output te komen. De views krijgen hun data voorgeschoteld door de controller en moeten deze niet actief opvragen.

Controller De controller is de applicatielaag die interageert met de modellen, de nodige logica uitvoert en uiteindelijk de juiste view oproept.

Hoofdstuk 5

Implementatie

5.1 Inleiding

In dit hoofdstuk wordt de implementatie van het project behandeld. Alhoewel het onmogelijk is om de alle details van de applicatie te belichten, worden toch alle belangrijke aspecten behandeld.
Items die in dit hoofdstuk aan bod komen, zijn:

- algemene aanpak binnenin de applicatie (pagina 29)
- authenticatie: veilig aanmelden van klanten (pagina 33)
- authorisatie: toeganscontrole (pagina 35)
- navigatie doorheen de applicatie (pagina 38)
- controlepaneel voor de instellingen van de applicatie (pagina 41)
- artikelen en categorieën in de webwinkel (pagina 43)
- afbeeldingen: uploaden, geschikt maken voor het web, beheer (pagina 44)
- nieuwsberichten op de website (pagina 47)
- registratieprocedure voor de klant (pagina 49)
- gebruikersbeheer voor de beheerder (pagina 49)
- winkelen: kiezen van artikelen en de bestelling maken (pagina 54)
- de leveringsplanner (pagina 55)
- unit tests (pagina 57)

5.2 Algemene aanpak binnenin de applicatie

5.2.1 Bestandsstructuur

De bestandsstructuur[1] van de webapplicatie wordt weergegeven in tabel 5.1.

map	inhoud
private	
private/cake	het CakePHP raamwerk
private/plaetinck.be/	app_controller.php en app_model.php
private/plaetinck.be/config	configuratiebestanden[2]
private/plaetinck.be/controllers	alle nodige controllers en componenten
private/plaetinck.be/models	alle modellen
private/plaetinck.be/test	alle testcases voor de applicatie
private/plaetinck.be/tmp	tijdelijke bestanden (cache,sessies)
private/plaetinck.be/views	view templates
private/plaetinck.be/views/elements	herbruikbare view templates
www.plaetinck.be	
www.plaetinck.be/css	alle nodige CSS-bestanden
www.plaetinck.be/img	alle afbeeldingen
www.plaetinck.be/js	alle nodige JavasScripts

Tabel 5.1: Bestandsstructuur van de webapplicatie

Enkele opmerkingen:

- Bij voorkeur is enkel de map *www.plaetinck.be* zichtbaar voor het publiek. De bestanden in *private* worden automatisch geinclude.
- De bestanden app_controller.php en app_model.php bevatten de klassen waar alle controllers respectievelijk modellen van de applicatie automatisch van overerven. Logica die elke controller of model moet hebben kan hier dus gedefinieerd worden, en in het geval van app_controller is daar ook gebruik van gemaakt. Zie ook secties *Authorisatie* (pagina 35) en *Controlepaneel* (pagina 41).

5.2.2 opbouw van URL's

Inleiding

Van evengroot belang als de structuur van de mappen en bestanden op het bestandssysteem, is de samenstelling van de URL's die gebruikt worden.
Een goede, coherente opbouw van de URL verhoogt de toegankelijkheid van de applicatie voor zowel klanten als zoekrobotten.

[1] Enkel de relevante mappen worden besproken
[2] Debugging niveau, databank instellingen,...

Een URL in deze webapplicatie ziet eruit als volgt:
http://hostnaam/controller/methode/parameters.
Een voorbeeld is:
http://hostnaam/artikelen/bekijk/1.
Dit roept de *bekijk* methode op van de *ArtikelenController* met als argument 1, zodat het artikel met als id 1 wordt opgevraagd.
Deze manier van werken geldt voor alle entiteiten zoals besproken in hoofdstuk *Modellering* op pagina 20. Voor statische pagina's zoals de informatiepagina of pagina met vaak gestelde vragen wordt de *display* actie van de *PagesController* opgeroepen.

Er zijn echter twee processen waar URL's verwerkt worden vooraleer effectief de juiste methode van een controller wordt opgeroepen.

- vertaling
- routering

Deze worden in de volgende secties besproken.

Vertaling

De namen van methoden en parameters - en soms zelfs volledige controllers - zijn intern in het Engels. Echter, deze applicatie is gericht naar Nederlandstalige gebruikers. CakePHP ondersteunt nog geen internationalisatie[3]. Daarom is een eigen methode ontwikkeld om Nederlandse URL's om te zetten naar Engelstalige. Het bijgevoegd script wordt opgeroepen bij iedere paginarequest, overloopt alle fragmenten van op de gevraagde URL, en vertaalt waar nodig.

```
function translate($var = null,$level="total"){
$translatetable = array('controller'
=> array(        'nieuws' => 'news',
                 'aanbiedingen' => 'offers',
                 'privacyverklaring' => 'privacystatement',
                 'profiel' => 'profile',
                 'hoofdpagina' => 'home',
                 'gebruikers' => 'users',
                 'groepen' => 'groups',
                 'paginas' => 'pages'
        ),
'action'
=> array(        'bekijk' => 'view',
                 'verwijder' => 'delete',
                 'bewerk' => 'edit',
                 'voegtoe' => 'add',
                 'overzicht' => 'index',
                 'registreer' => 'register',
                 'loguit' => 'logout'
        )
```

[3]Ook wel i18n genoemd. `http://en.wikipedia.org/wiki/Internationalization_and_localization`

```
);
                if($var)
                {
                        if($level=="total")
                        {
                                $old = explode('/',$var);
                                $lastone = end($old);
                                if(empty($lastone)) array_pop($old);
                                $new = array();

                                /* translate each part where you have a mapping
                                    table for */
                                $i = 0;
                                for($i ; $i <sizeof($translatetable) ; $i++)
                                {
                                        $levels = array_keys($translatetable);
                                        if(sizeof($old)>$i)     $new[$i] =
                                            translate($old[$i],$levels[$i]);
                                }

                                /* copy remaining entries, if any.  array_merge
                                    () is not suitable for this */
                                for ($i ;$i < sizeof($old); $i++)
                                {
                                        $new[$i] = $old[$i];
                                }

                                /* construct the translated url.  this also adds
                                     a trailing "/" even if it wasn't in the
                                    original */
                                $new_url="";
                                foreach($new as $n)
                                {
                                        $new_url .= $n."/";
                                }

                                return $new_url;
                        }
                        else
                        {
                                if(isset($translatetable[$level]))
                                {
                                        foreach ($translatetable[$level] as
                                            $orig => $new)
                                        {

                                                if($var == $orig) $var = $new;
                                        }
                                }
                                else
                                {
                                        if(DEBUG) echo("translate table for
                                            level ".$level." not found: ".$var."
                                             left untranslated\n");
                                }
                                return $var;
                        }
                }
        }
```

Listing 5.1: translate.php

Op deze manier wordt bijvoorbeeld de URL
http://hostnaam/paginas/bekijk/hoofdpagina
omgezet in *http://hostnaam/pages/display/home*.

Routering

Nadat de nodige parameters vertaald zijn naar het Engels, worden de URLś door de zogenaamde *Router* gehaald. Deze instantie mapt URL's op andere URL's. Het voordeel hiervan is dat kortere, makkelijker te onthouden URL's kunnen gebruikt worden voor bepaalde delen van de applicatie.
Een voorbeeld: De gebruiker vraagt de pagina *http://hostnaam/hoofdpagina* op. Dankzij de vertaling wordt deze URL omgevormd tot *http://hostnaam/home*. Daarna zal de router deze URL omzetten naar *http://hostnaam/pages/display/home*.

Enkele routes worden als voorbeeld getoond. Op deze manier wordt een nieuwe URL opgesteld aan de hand van de opgegeven controller, methode en parameters.

```
Router::connect('/',                                array('controller' => 'pages', '
    action' => 'display', 'home'));
Router::connect('/home',                            array('controller' => 'pages', '
    action' => 'display', 'home'));
Router::connect('/website',             array('controller' => 'pages', 'action'
    => 'display', 'website'));
Router::connect('/faq',                             array('controller' => 'pages', '
    action' => 'display', 'faq'));
Router::connect('/help',                            array('controller' => 'pages', '
    action' => 'display', 'faq'));
```

Listing 5.2: Routes

5.2.3 Accounts

In deze applicatie zijn er 3 soorten gebruikers:

- De *gast* is niet ingelogd.
- De *klant* is ingelogd (nadat de account is goedgekeurd door de beheerder)
- De *beheerder* is de enige account die standaard aanwezig is. Wie ingelogd is als beheerder heeft alle rechten. Dit zijn de accountgegevens:
  - gebruikersnaam: root
  - wachtwoord: root

Meer informatie hieromtrent kan teruggevonden worden in sectie *Authorisatie* op pagina 35.

5.3 Authenticatie

5.3.1 Inleiding

Authenticatie is het proces waarin gecontroleerd wordt of iemand is wie hij/zij beweert te zijn.[6]. Uiteraard is het van belang dat dit proces zo goed mogelijk werkt, m.a.w. dat de controle van de identiteit van de gebruiker zo veilig mogelijk gebeurt.

5.3.2 Gekozen aanpak

Aangezien het officiële authenticatiesysteem voor CakePHP 1.2 nog in ontwikkeling is, is er voor gekozen om zelf een authenticatiesysteem te schrijven.
Het authenticatiesysteem dat voor dit project ontwikkeld werd is ook vrijgegeven als open source component.[4]

5.3.3 Principe

Het principe is het "challenge-response" mechanisme. Hierbij wordt een gebruiker geauthenticeerd door zijn gebruikersnaam op te sturen en een hash van zijn wachtwoord. Niet het wachtwoord zelf. De manier waarop de hash bekomen wordt is bovendien niet constant in de tijd.
Indien de hash-methode constant zou zijn in de tijd, zou dit de zogenaamde "sniffers" de mogelijkheid geven zich valselijk te authenticeren door de hash, die ze onderschept hebben, op te sturen.
In de plaats daarvan verandert de manier waarop de hash bekomen wordt continu. Om dit mogelijk te maken spreken server en client een zogenaamde "seed" af die gebruikt wordt bij het hashen. In praktijk is dit een eenvoudige "salt". Een salt is eenvoudigweg *één* of meerdere karakters die worden toegevoegd aan de string alvorens te hashen. Door deze salt willekeurig te laten genereren door de server bij elke login-aanvraag is er dus een andere hash van het wachtwoord nodig.

5.3.4 Werking

3-fasen model

Het wachtwoord doorloopt 3 fases:

Fase 1: Cleartext Het cleartext wachtwoord is normaal gezien enkel door de gebruiker bekend. Bij voorkeur wordt het nergens opgeslaan, noch verzonden over het netwerk.

Fase 2: Stage 1 hash De *stage 1 hash*, is een hash die via een vastgelegde manier berekend wordt. Deze hash wordt opgeslaan in de databank

[4]Het authenticatiesysteem is onder de naam "dAuth" vrijgegeven. `http://bakery.cakephp.org/articles/view/introduction-to-dauth-v0-3`. Het is verschenen in onder andere "International PHP Magazine"[7]

Fase 3: Stage 2 hash De *stage 2 hash* is de hash die op een unieke manier berekend wordt. Deze wordt over het netwerk gestuurd. Als deze onderschept wordt, is hij waardeloos voor een zogenaamde "sniffer".

Aan de client-zijde gebeuren de berekeningen dmv het JavaScript *d_auth.js*. Aan de serverzijde wordt de *DAuthComponent* gebruikt.

Fallback

Een probleem ontstaat wanneer de gebruiker geen JavaScript-ondersteuning heeft in zijn webbrowser. Hierdoor kunnen er geen bewerkingen op het wachtwoord zoals hashen uitgevoerd worden. In praktijk valt dit weinig voor[8] maar toch kan er beter rekening mee gehouden worden.
Om dit op te vangen, is in dAuth de "fallback" voorzien. Dit houdt in dat als een gebruiker geen JavaScript-ondersteuning heeft, het wachtwoord in cleartext verzonden kan worden. Het voordeel is dat de gebruiker toch kan inloggen, echter de veiligheid wordt erdoor verlaagd.
De fallback is echter configureerbaar door de beheerder op het controlepaneel. (zie sectie *Controlepaneel* op pagina 41) Wanneer *hoge_veiligheid* wordt aangezet, dan wordt geen fallback gebruikt: iemand die probeert in te loggen zonder JavaScript zal een foutmelding zien dat hij JavaScript moet aanzetten, en hij zal geen bruikbaar formulier te zien krijgen om te belemmeren dat het wachtwoord in cleartext wordt verzonden.

Brute-force detectie

Het wachtwoord mag nog zo veilig verstuurd worden, wanneer er geen controle is op het aantal login pogingen, kan na lang proberen iemand het juiste wachtwoord raden en zich valselijk authenticeren. (de zogenaamde "brute-force aanval"). Dit moet vermeden worden. Daarom wordt bij elke inlogpoging bijgehouden welke host de aanvraag deed en op welk tijdstip. Wanneer in een bepaald interval er meer aanvragen gebeuren dan toegelaten, wordt de host geblokkeerd. Standaard is dit 4 pogingen in 5 seconden, maar dit kan aangepast worden in de configuratie van de component. Eveneens kan in de configuratie van de component gekozen worden of er een foutmelding moet verschijnen of de verwerking van de pagina gewoon moet stoppen, wanneer dergelijk misbruik voor het eerst wordt gedetecteerd, of wanneer gevonden wordt dat de host geblokkeerd reeds is.

Sessie hijacking

Overige mogelijke problemen zoals sessie-hijackings.[4] worden afgehandeld door het CakePHP raamwerk.

5.4 Authorisatie

5.4.1 Inleidng

Nadat op een redelijk veilige manier de gebruiker ge-authenticeerd is, moet natuurlijk kunnen gecontroleerd worden wat de permissies voor die gebruiker zijn, met als doel te beslissen of een gebruiker toegang krijgt tot een bepaald object of actie
Dit is het aspect toegangscontrole (*access control*) of authorisatie.
CakePHP heeft standaard een degelijk systeem om toeganscontrole toe te passen: ACL of "access control lists"[5].
Het probleem echter, is de administratie van de acl-regels: de regels zijn opgeslaan via het mptt-algoritme [27].
Zoals reeds besproken in hoofdstuk *Technologieën* is het moeilijk om de software te schrijven die op een eenvoudige manier deze structuur kan beheren. Daarom is gekozen voor een andere oplossing. De oplossing is gebaseerd op een bestaand idee[18] waarbij via tekstregels de permissies voor een groep worden bepaald.

5.4.2 Werking

Opstelling van de permissieregel

De permissieregels dienen om te bepalen of een gebruiker een bepaalde methode van een controller mag oproepen. Via de permissieregels - welke niet meer dan gewone strings zijn - wordt voor elke gebruikersgroep[6] opgeslaan wat diens permissies zijn.
Bovendien werkt het systeem additief, het gaat als volgt:
(De gebruikte variabelen worden uitgelegd in sectie *Controlepaneel* op pagina 41)

- De variabele *standaard_rechten* wordt genomen als startpunt.
- Indien de gebruiker is ingelogd, wordt de variabele *rechten_klant* achteraan toegevoegd aan *standaard_rechten*.
- Wanneer de gebruiker een groep is toegewezen, en die groep heeft een eigen permissieregel (attribuut *permissions*), dan wordt deze regel achteraan toegevoegd aan de bestaande regel.

Verwerking van de permissieregel

Een aldus bekomen permissieregel kan er als volgt uit zien:
!:*,*:display,*:index,!chauffeurs:*,!leveringen:**
De verschillende segmenten worden gescheiden door komma's. Elk segment bestaat dus uit een controller en een actie, gescheiden door een dubbel punt.
Het voorkomen van een bepaalde controller:methode combinatie leidt tot het toekennen

[5] http://manual.cakephp.org/chapter/acl

[6] Het is ook mogelijk rules te maken voor gebruikers onderling, maar voor deze implementatie bleek het voldoende om enkel op basis van groepen te werken.

van het recht, tenzij hij voorafgegaan is door de negatie operator (het uitroepteken). Wildcards (*) duiden op eender welke controller of actie.
De permissieregel wordt steeds van links naar rechts, segment per segment ontleedt. Standaard wordt het recht niet toegewezen, echter tijdens het doorlopen van de regel kan het recht verschillende keren toegewezen en weer afgenomen worden, afhankelijk van het al dan niet voldoen van het aangevraagde controller:methode paar aan het segment dat het gecontroleerd wordt. Na het doorlopen van de hele regel is het uiteindelijk resultaat bekend.
Bovenstaande regel neemt dus eerst alle rechten af, staat daarna de methodes *display* en *index* toe voor eender welke controller, en neemt vervolgens terug rechten af voor bepaalde controllers.

De gebruikte code is de volgende:

```
$ac = $this->checkAccess($this->params['controller'],$this->params['action'],
    $rules);
if(!$ac){
        $this->redirect('/',null,true);
}

function checkAccess($controller, $action, $rules, $allowed = false){
        preg_match_all('@([^:,]+):([^,:]+)@is', $rules, $matches, PREG_SET_ORDER
            );
        foreach ($matches as $match){
                $reverse = false;
                $contr = str_replace('*', '.*', $match[1]);
                $act = str_replace('*', '.*', $match[2]);
                if ($contr{0}=='!'){
                        $contr = substr($contr, 1);
                        $reverse = true;
                }
                if (preg_match('/^'.$contr.'$/i', $controller) && preg_match('/^
                    '.$act.'$/i', $action)){
                        $allowed = !$reverse;
                }
        }
        return $allowed;
}
```

Listing 5.3: CheckAcess

Deze logica wordt uitgevoerd bij elke paginarequest. Hiervoor wordt de *beforeFilter()* callback van de *AppController* gebruikt. De *AppController* is de controller waar elke controller in de applicatie van overerft, en de *beforeFilter()* calback wordt uitgevoerd voor de uitvoering van elke publiek bereikbare methode. Op die manier is elke mogelijke methode in de applicatie beschermd.

De standaard account (root:root) hoort tot de groep *beheer*. Deze groep is de enige die standaard aanwezig is, en heeft als permissie regel **:**.

Ondernomen acties

Wanneer het resultaat van de *CheckAccess()* methode *true* is, doorloopt de controller de opgevraagde methode. Wanneer het resultaat echter *false* is, dan stopt onmiddelijk de uitvoering van de huidige pagina, en wordt een *http redirect* uitgevoerd naar de hoofdpagina van de website.

Hierom moet opgelet worden bij het aanpassen van de permissieregels: indien iemand het recht op het paar *pages:display* wordt ontnomen dan blijft de website naar zichzelf redirecten.

Granulariteit

Deze manier van werken biedt een relatief granulaire controle over de permissies. Voor deze applicatie was het niet nodig om permissies per gebruiker in te stellen, per groep was voldoende. De entiteiten waarvoor permissie gevraagd wordt zijn combinaties van controllers en methode, anders gezegd: het type modellen waarmee gewerkt wordt, en het soort actie ermee wil gedaan worden.
Er is in dit systeem geen notie over de identitieit van het object zelf, waarmee gewerkt wordt. Dit impliceert dat - bij slechts enkele - controllers er nog extra logica is geprogrammeerd om dit te verwezelijken. Een concreet voorbeeld:
Een klant mag zijn eigen profiel bewerken, maar niet dat van anderen. Hij moet dus toegang krijgen tot *klant:bewerk*. Daarom staat in de bijhorende *edit* methode van de *UsersController* logica die controleert welk profiel de klant wil bewerken, en indien nodig toegang weigert.

De standaard regels

De regels die standaard zijn ingesteld zijn de volgende:

- *standaard_rechten* : *!*:*,*:display,*:view,*:index,*:register,*:login, :logout,!chauffeurs:*,!leveringen:*,!groups:*,!afbeeldingen:*, !instellingen:*,!bestellingen:*,!afbeeldingen:*,!users:index*
- *rechten_klant* : *bestellingen:checkout,bestellingen:add, bestellingen:index,bestellingen:cart,bestellingen:addarticle, bestellingen:removearticle,users:bewerk*
- *Groep beheer* : *:*

5.5 Navigatie

5.5.1 De layout

De Layout bevat verschillende items: de twee menu's die op elke sectie van de applicatie aanwezig zijn, en de informatie die wordt weergegeven in het midden van de interface. Standaard wordt hier de hoofdpagina getoond, tenzij de gebruiker ergens naartoe navigeert via het menu of door zelf een specifieke URL op te vragen.

5.5.2 De hoofdpagina

Op de hoofdpagina worden twee elementen weergegeven:

- een overzicht van het laatste nieuws
- een overzicht van de nieuwste artikelen in de winkel

Van hieruit kan verder geklikt worden naar een pagina met meer nieuws, of naar de winkel met alle aangeboden artikelen.

Figuur 5.1: Hoofdpagina

5.5.3 De menu's

Klein menu rechtsboven

Het zogenaamde *Extra Menu* staat rechtsboven en bevat links naar

- de hoofdpagina
- de help-pagina (bevat vaak gestelde vragen en antwoorden)
- een pagina met informatie over de website

Hoofdmenu

Het hoofdmenu bevindt zich links en bestaat uit twee delen: het gemeenschappelijk deel voor iedereen die de applicatie gebruikt, en het onderste deel, dat links toont die relevant zijn voor de rol van de gebruiker (gast, klant, of beheerder).

Het gemeenschappelijk deel bevat links naar

- de pagina met het bedrijfsprofiel
- een infopagina met contactgegevens
- de hoofdpagina
- het overzicht van alle artikelen
- het overzicht van alle categorieën
- de nieuws sectie

Het aangepast gedeelte wordt weergegeven in figuur 5.2. De inhoud verschilt naar gelang de rol van de gebruiker:

- Gast
  - een link om in te loggen
  - een link om te registreren
- Klant
  - een link om uit te loggen
  - een link naar het bestellingenoverzicht
  - een link om het gebruikersprofiel aan te passne
  - het winkelmandje: hierin kunnen artikelen gesleept worden om te bestellen
  - het verwijdervakje: hierin kunnen artikelen gesleept worden vanuit het winkelmandje

Figuur 5.2: Menu's

- Beheerder
  - een link om uit te loggen
  - een link naar de leveringsplanner (zie sectie *Leveringsplanner* op pagina 55)
  - links naar de beheerspagina's voor
    - afbeeldingen (zie sectie *Afbeeldingen* op pagina 44)
    - artikelen en categorieën (zie sectie *Artikelen en categorieën* op pagina 43)
    - bestellingen
    - chauffeurs
    - gebruikers en groepen(zie sectie *Gebruikersbeheer* op pagina 49)
    - instellingen (zie sectie *Controlepaneel* op pagina 41)
    - nieuws (zie sectie *Nieuwsberichten* op pagina 47)

5.6 Controlepaneel

5.6.1 Principe

Vanuit het controlepaneel kunnen algemene instellingen van de applicatie worden beheerd.
Instelingen zijn "naam-waarde"paren waarbij de naam dienst doet als unieke sleutel en als variabele beschikbaar gemaakt wordt in de hele applicatie (door ze op te vragen in de *beforeFilter* methode van de *AppController*)

Instellingen hebben ook een derde attribuut genaamd *aanpasbaar*. Enkel bij de standaard aanwezige paren staat dit veld op 0, bij de andere op 1. Op deze manier kan de beheerder later altijd teruggevallen worden op de standaardwaarden, indien nodig.

5.6.2 Standaard instellingen

De volgende instellingen zijn standaard aanwezig:

naam	waarde
bedrijfsnaam	"Plaetinck D. Vleeshandel NV"
hoofding	"Plaetinck D. Vleeshandel NV online"
google_analytics	""
standaard_rechten	"!*:*,*:display,*:view,*:index,*:register,*:. . . "
rechten_klant	"bestellingen:checkout,bestellingen:index,. . . "
hoge_veiligheid	1
thema	"standaard"

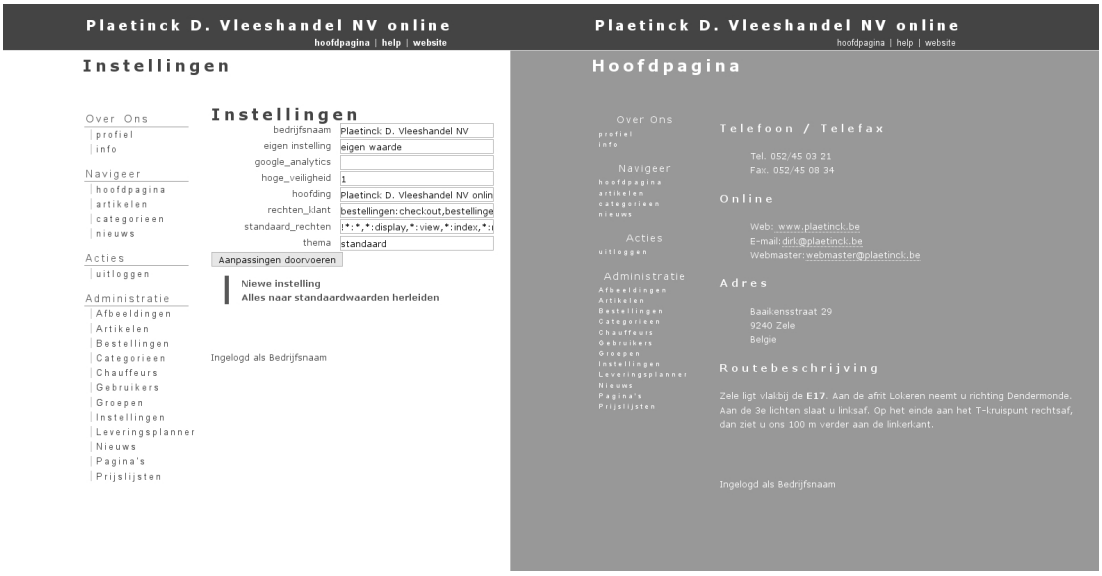
Tabel 5.2: Standaard instelingen

De *bedrijfsnaam* en *hoofding* zijn variabelen die in de view-laag beschikbaar gemaakt worden. Zo wordt de bedrijfsnaam onder andere weergegeven in de titelbalk en op de help-pagina. De hoofding wordt gebruikt op de zogenaamde "banner" van de webapplicatie.

De *google_analytics* variabele staat standaard ingesteld op een lege string. Hier kan echter een gebruikersnummer van de Google Analytics[7] service ingegeven worden. Wanneer de applicatie hier iets anders dan een lege string detecteert, zal het automatisch de ondersteuning voor Google Analytics activeren. Zie hiervoor het bestand *webapp/private/plaetinck.be/views/elements/stats.thtml*.

De instellingen *standaard_rechten* en *rechten_klant* worden in de sectie "Authorisatie" op pagina 35 besproken. De instelling *hoge_veiligheid* werd reeds besproken in de sectie "Authenticatie" op pagina 33.

De instelling *thema* bepaalt de layout van de applicatie. Er zijn 2 thema's meegeleverd met de applicatie: *standaard* en *blauw*.[8]
Uiteraard is het ook mogelijk om op een eenvoudige manier zelf andere layouts toe te voegen, maar dit bespreken zou ons te ver brengen.



Figuur 5.3: De twee meegeleverde layouts. Links het controlepaneel in de *standaard* layout en rechts de infopagina in thema *blauw*

[7]Google Analytics is een dienst die statistieken over bezoeken van een website bijhoudt. `https://www.google.com/analytics/home/`

[8]De blauwe layout is niet in dezelfde mate uitgewerkt als de standaard layout.

Niet alleen bestaande waarden kunnen aangepast worden, er kunnen ook nieuwe instellingen gedefinieerd worden.

5.7 Artikelen en categorieën

Figuur 5.4: Formulieren om artikelen en categorieën aan te maken

Om de webwinkel te beheren kunnen door de beheerder artikelen en categorieën aangemaakt worden. De formulieren die hiervoor gebruikt worden zijn afgebeeld op figuur 5.4. De formulieren die gebruikt worden om bestaande artikelen en categorieën aan te passen zijn quasi identiek aan degene om ze aan te maken.
Aan zowel artikelen als categorieën kunnen afbeeldingen gekoppeld worden. (Het werken met afbeeldingen wordt besproken in sectie *Afbeeldingen*) De formulieren geven automatisch de beschikbare afbeeldingen weer. In het geval van de artikelen worden ook automatisch de beschikbare categorieën weergegeven. Het is ook mogelijk om eerst een nieuwe categorie te maken en die te gebruiken.
Op het overzicht van artikelen wordt bovendien weergegeven hoeveel van elk artikel besteld is.

5.8 Afbeeldingen

5.8.1 Modellering

Afbeeldingen zijn de enige entiteiten die niet in de databank worden opgeslaan. Afbeeldingen worden rechtstreeks op het bestandssysteem opgeslaan. De *id* van een afbeelding is niets anders dan de bestandsnaam zelf. Op die manier kunnen associaties gelegd worden met de klassen Artikel en Categorie. Om dit mogelijk te maken moesten de *save* en *findAll* methoden, die gedefineerd zijn in de generieke klasse *Model* hergedefineerd worden.

```
class Afbeelding extends AppModel
{
        var $useTable = false;
        var $path = IMG_ORIG_DIR;

        function save($data){
                if($data['Naam']) $filename =$data['Naam'];
                else $filename = $data['Afbeelding']['name'];
                $filename = $this->Filename($filename);
                $result =  move_uploaded_file($data['Afbeelding']['tmp_name'],
                    $this->path.$filename);
                return $result;
    }
    function Filename($filename){
        $splitted = explode('.',$filename);
        $filename = $splitted[0]; // get rid of dots and everything behind it.
        while (file_exists($this->path.$filename)){
                $filename .= substr(md5(uniqid(rand(), true)),0,1);
        }
        return $filename;
    }
    function findAll(){
        $output = array();
        if(is_dir($this->path)){
                if ($dh = opendir($this->path)) {
                        while (($file = readdir($dh)) !== false) {
                        if($file != '.' && $file != '..')$output[] = array('
                            Afbeelding' => array('id' => $file));
                        }
                        closedir($dh);
                }
                }
                return $output;
    }
    function del($id = null){
        if($id) return @unlink($this->path.$id);
        else return false;
    }
}
```

Listing 5.4: klasse Afbeelding

5.8.2 Conversie

In plaats van een aanpak waarbij expliciet op voorhand de afbeeldingen naar de juiste afmetingen worden geconverteerd, wordt gebruik gemaakt van de *Image Resize Helper* [9]. Hiermee kan, bij het opvragen van afbeeldingen in een specifiek formaat vanuit de view-laag (bvb miniatuurafbeeldingen of zogenaamde "thumbnails") automatisch de afbeelding gegenereerd worden en aan de gebruiker getoond worden. Daarna wordt de afbeelding gecached zodat ze de volgende keer geladen wordt zoals eender welke afbeelding. De gebruikte formaten zijn 100x00 pixels voor thumbnails, en 300x00 voor grote afbeeldingen. Deze gegenereerde afbeeldingen worden opgeslaan in de map *webapp/www.plaetinck.be/img/imagecache*

5.8.3 Uploaden

Voor het uploaden van afbeeldingen wordt de map
webapp/www.plaetinck.be/img/originelen gebruikt.
Er zijn twee manieren om afbeeldingen te uploaden: via de web interface en volledig automatisch.

Via de web interface

Om bestanden te uploaden via een web-interface is een formulier nodig. Aanvankelijk leek *SwfUpload*[10] hiervoor een geschikte utility, omdat dit toelaat meerdere bestanden tegelijk te uploaden, en een vooruitgangsindicator voorziet. Omdat het gebruik hiervan echter veiligheidsproblemen met zich meebrengt[11] is gekozen om met de gewone voorziening van PHP te werken.

Om een nieuwe afbeelding te uploaden wordt via een eenvoudig formulier op het bestandssysteem een afbeelding uitgekozen. De beheerder kan er ook voor kiezen een alternatieve naam op te geven. Wanneer de overdracht voltooid is, wordt het mimetype gecontroleerd. Dit moet *image/jpeg* of *image/png* zijn, aders wordt de afbeelding automatisch terug gewist. Wanneer hieraan voldaan is wordt een bestandsnaam gekozen. Initieel is deze ingesteld op de originele bestandsnaam, of de nieuwe naam zoals opgegeven door de beheerder in het formulier. Wanneer er echter reeds een afbeelding met deze naam bestaat, worden nieuwe willekeurige namen gegenereerd totdat een naam wordt gevonden die nog niet in gebruik is. Deze bestandsnaam doet dienst als *id* voor associaties met artikelen of categorieën.

[9] `http://bakery.cakephp.org/articles/view/image-resize-helper`

[10] `http://swfupload.mammon.se/`

[11] Aangezien SwfUpload een eigen sessie krijgt moet ofwel de apache module modsecurity uitgeschakeld worden of moet de beveiliging van de webapplicatie verzwakt worden. Zie ook `http://groups.google.com/group/cake-php/browse_thread/thread/5cedbe5a1718e9ce/`

Automatisch vanaf het werkstation

De tweede manier om afbeeldingen te uploaden verloopt volledig automatisch. Hiervoor wordt gebruik gemaakt van de Xfce desktop omgeving, meer bepaald de *Volume manager* van de bestandsbeheertool *Thunar*. De Volume Manager werd zo geconfigureerd dat hij een shellscript oproept wanneer het digitaal fototoestel werd aangesloten op het werkstation. Dit script verkleint eerst de foto's zodat ze drastisch minder plaats innemen, maar toch nog groot genoeg zijn om verwerkt te worden door de *Image Helper*. Hiervoor wordt de Imagemagick toolset gebruikt. Daarna wordt automatisch een connectie met de FTP-server gemaakt, en worden de bestanden in de juiste map geplaatst.

Figuur 5.5: Configuratie van de Volume Manager

```
#!/bin/bash

#Dit script kopieert afbeeldingen vanaf het fototoestel, verkleint ze, en upload
     ze
#naar de opgegeven ftp-locatie

        tmpdir=/tmp/imageuploader
        imagedir=$1/DCIM/101MSDCF
        ftphost=ftp.plaetinck.be
        ftpuser=username
        ftppass=password

if [ -n "$1" ]; then
        mkdir $tmpdir
        cd $1;
        for file in *;
        do
                convert $file -resize x400 "${tmpdir}/${file}";
```

```
        done

        cd $tmpdir;
        ftp -ni $ftppass <<FTP-Session
        user $ftpuser $ftppass
        cd www.plaetinck.be/img/originelen
        binary
        mput *
        bye
FTP-Session
        cd -;
        rm -rf $tmpdir
        exit 0
else
        exit 1
fi
```

Listing 5.5: Shellscript image_uploader

De volgende opties voor de ftp-client waren nodig:

- *n*: zorgt ervoor dat we de logingegevens vanuit het script kunnen opsturen
- *i*: schakelt de interactieve modus uit, zodat niet steeds om bevestiging wordt gevraagd bij het uploaden van afbeeldingen.

Het script is echter vrij rudimentair: het gebruikt steeds alle aanwezige foto's op het fototoestel, en indien er problemen optreden met de FTP-verbinding wordt dit niet opvangen.

5.9 Nieuwsberichten

5.9.1 Ingave

De mogelijkheid is voorzien voor de beheerder om nieuwsberichten te plaatsen op de voorpagina. Hiervoor kan ook opmaak gebruikt worden.

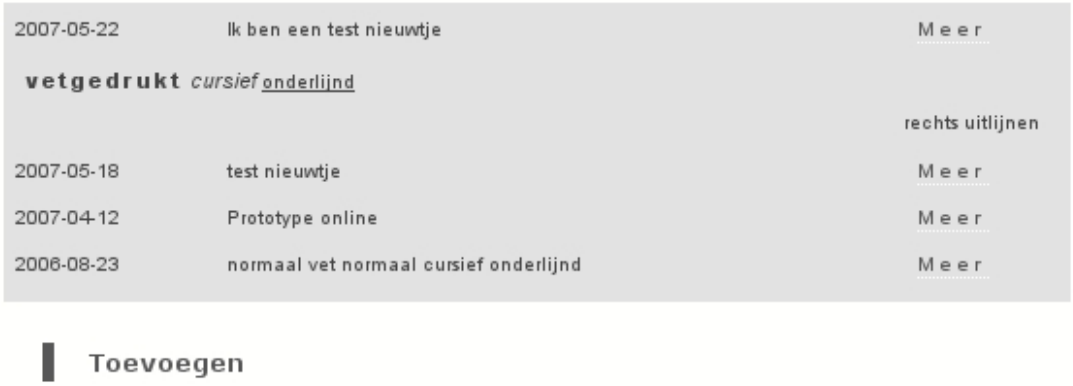
5.9.2 Weergave

Het nieuws element, (*latest_news*) dat onder andere op de voorpagina wordt gebruikt maakt bovendien gebruik van het caching mechanisme van CakePHP. Het nieuws element wordt op deze manier maximum 1 uur lang in de cache gehouden. De reden dat het *latest_offers* element niet gebruikt maakt van caching, is omdat dit opnieuw gegenereerd moet worden afhankelijk van de gebruiker die het opvraagt.

```
<p>Welkom op de website van <?php echo $bedrijfsnaam; ?>
</p>
<?php
echo $this->element('latest_news',array('cache'=>'+1 hour'));
echo $this->element('latest_offers');
?>
```

Listing 5.6: Inhoud hoofdpagina

Figuur 5.6: Plaatsen nieuws



Figuur 5.7: Weergave van nieuws in het *latest_news* element op de voorpagina

Het nieuws wordt standaard ingeklapt weergegeven, dit wil zeggen enkel de titel wordt getoond. Wanneer op *Meer* wordt geklikt, klapt het bericht uit en wordt de inhoud van het bericht getoond. Dit gebeurt in JavaScript via de prototype bibliotheek en vereist geen nieuwe pagina- of ajax-request.

5.10 Registratie als klant

De registratiepagina voor klanten wordt getoond op figuur 5.8. De velden die vet gedrukt zijn, zijn verplicht. Wanneer de gebruiker het formulier verstuurt maar een verplicht veld niet invult, krijgt hij terug hetzelfde formulier te zien met een foutmelding dat hij het desbetreffende veld moet invullen. Ook als een accountnaam gekozen wordt die reeds bestaat wordt hij hiervan op de hoogte gebracht.

Wanneer de klant probeert in te loggen nadat de registratie voltooid is, maar de account nog niet is geactiveerd door de beheerder, zal de klant verwittigd worden dat hij moet wachten tot de account actief is.

5.11 Gebruikersbeheer

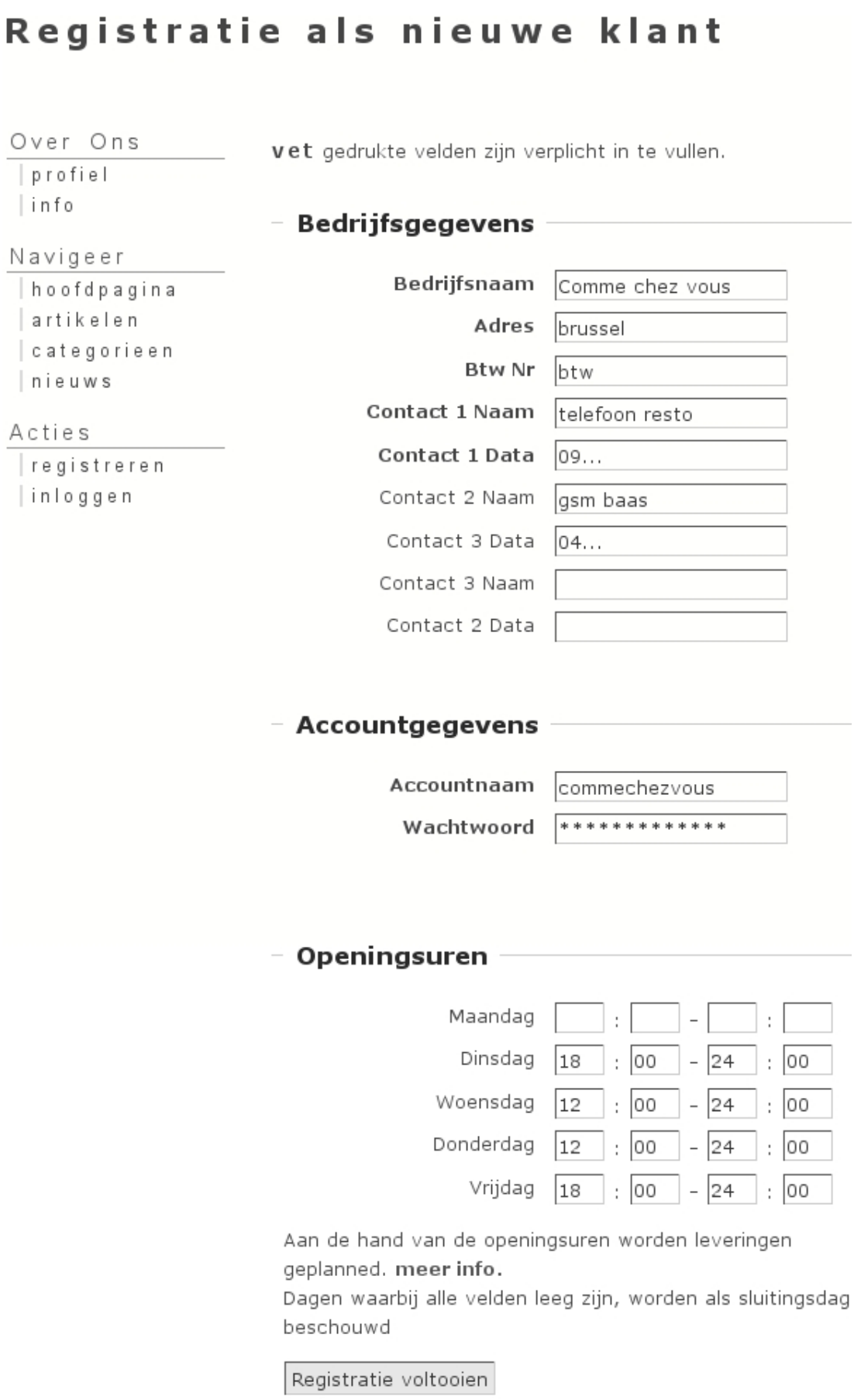
5.11.1 Mogelijkheden

Het gebruikersbeheer is voor de beheerder bereikbaar op
http://hostnaam/gebruikers/.
Hier wordt een overzicht gegeven van alle klanten, en hun attributen zoals *bedrijfsnaam*, *accountnaam*,*groep* en *is_actief*. (figuur 5.9) Per klant zijn ook de volgende acties mogelijk:

- *Bekijk* : gedetailleerd overzicht van alle eigenschappen van het klantprofiel
- *Bewerk* : aanpassingen maken aan een profiel: deze pagina is bijna identiek aan de registratiepagina voor een klant, maar hier kan ook een account op actief of inactief gezet worden.
- *Verwijder* : verwijderen van een entiteit, nadat om bevestiging is gevraagd (figuur 5.10)
- *Login*: inloggen als deze klant. Op deze manier kunnen bestellingen door de beheerder gemaakt worden in naam van een klant.

5.11.2 Interface

Het overzicht biedt enkele handige functionaliteiten.
Zo worden de records *gepagineerd* weergegeven. Dit wil zeggen dat als er meer records zijn dan op een pagina passen, automatisch de inhoud verdeeld wordt over meerdere pagina's. Met de links *vorige* en *volgende* kan van pagina veranderd worden. Deze

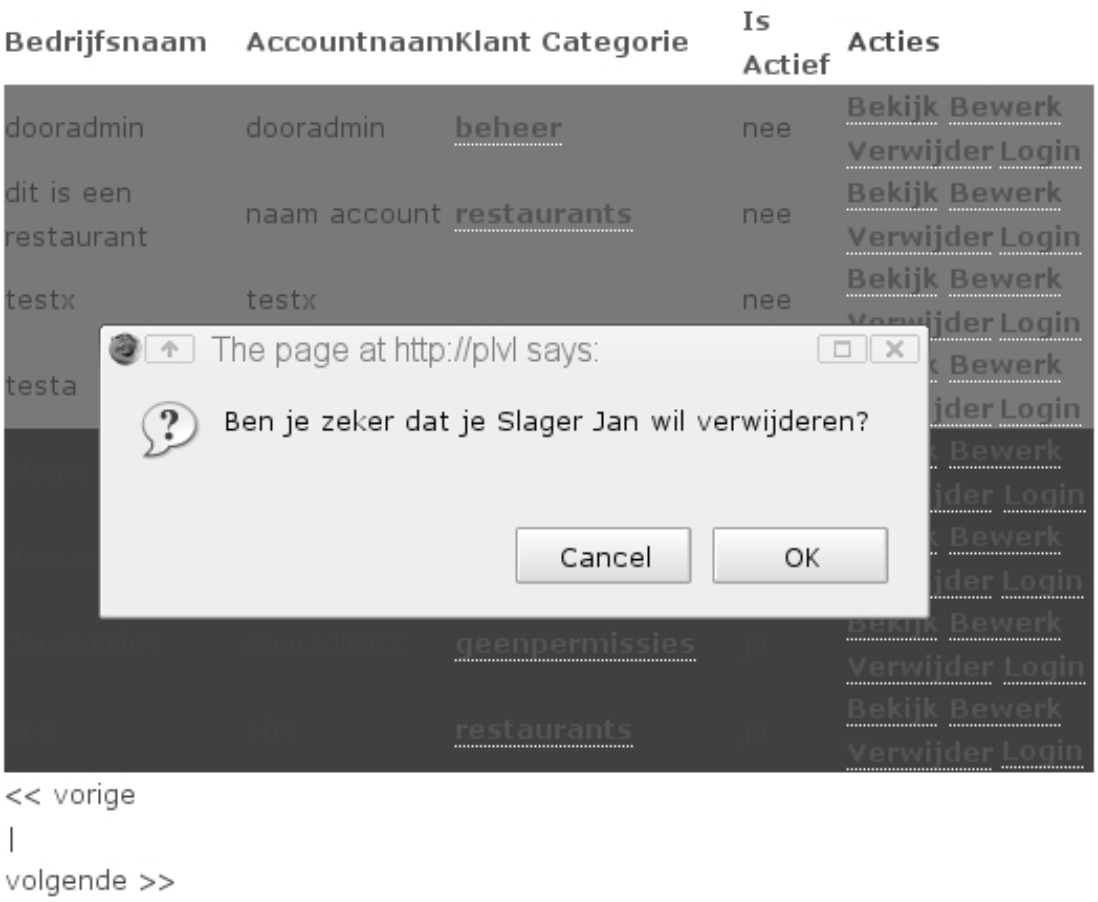


Figuur 5.8: Registratieformulier voor klanten

Bedrijfsnaam	Accountnaam	Klant Categorie	Is Actief	Acties
abc	abc	restaurants	ja	Bekijk Bewerk Verwijder Login
dit is een restaurant	naam account	restaurants	nee	Bekijk Bewerk Verwijder Login
dooradmin	dooradmin	beheer	nee	Bekijk Bewerk Verwijder Login
dooradmin2	dooradmin2	restaurants	ja	Bekijk Bewerk Verwijder Login
dooradmin3	dooradmin3	geenpermissies	ja	Bekijk Bewerk Verwijder Login
slagers 1an	pas	slagers (prijs ratio 50)	ja	Bekijk Bewerk Verwijder Login
testa	testa		nee	Bekijk Bewerk Verwijder Login
testx	testx		nee	Bekijk Bewerk Verwijder Login

<< vorige
|
volgende >>

Figuur 5.9: Gebruikersoverzicht gesorteerd op bedrijfsnaam



Figuur 5.10: Verwijderen van een account

methode is niet enkel een realisatie in de view-laag, maar wordt ook geïmplementeerd in de controller. Op die manier worden de databank niet meer belast dan nodig.

Bovendien zijn de kolommen ook sorteerbaar. Door op de hoofding van een bepaalde kolom te klikken, wordt het overzicht gesorteerd volgens dat attribuut, nogmaals klikken op dezelfde hoofding verandert de richting van sorteren. Ook deze functionaliteit is geïmplementeerd op controllerniveau en resulteert in aangepaste sql-queries.

Ter illustratie: in de figuur met het overzicht (figuur 5.9) wordt de eerste pagina weergegeven, en is gesorteerd volgens bedrijfsnaam. Dit resulteert in de volgende URL:
`http://hostnaam/users/index/page:1/sort:bedrijfsnaam/direction:asc`
De gegenereerde sql-query is de volgende:

```
SELECT
‘User‘.‘id‘, ‘User‘.‘bedrijfsnaam‘, ‘User‘.‘accountnaam‘,
‘User‘.‘adres‘, ‘User‘.‘btw_nr‘,‘User‘.‘contact_1_naam‘,
‘User‘.‘contact_2_naam‘, ‘User‘.‘contact_3_naam‘,
‘User‘.‘contact_1_data‘, ‘User‘.‘contact_2_data‘,
‘User‘.‘contact_3_data‘,‘User‘.‘wachtwoord‘,
‘User‘.‘maandag_start_u‘, ‘User‘.‘maandag_eind_u‘,
‘User‘.‘dinsdag_start_u‘, ‘User‘.‘dinsdag_eind_u‘,
‘User‘.‘woensdag_start_u‘,‘User‘.‘woensdag_eind_u‘,
‘User‘.‘donderdag_start_u‘, ‘User‘.‘donderdag_eind_u‘,
‘User‘.‘vrijdag_start_u‘, ‘User‘.‘vrijdag_eind_u‘,
‘User‘.‘klant_categorie_id‘,‘User‘.‘is_actief‘,
‘User‘.‘maandag_start_m‘, ‘User‘.‘maandag_eind_m‘,
‘User‘.‘dinsdag_start_m‘, ‘User‘.‘dinsdag_eind_m‘,
‘User‘.‘woensdag_start_m‘,‘User‘.‘woensdag_eind_m‘,
‘User‘.‘donderdag_start_m‘, ‘User‘.‘donderdag_eind_m‘,
‘User‘.‘vrijdag_start_m‘, ‘User‘.‘vrijdag_eind_m‘,
‘Group‘.‘id‘, ‘Group‘.‘naam‘,‘Group‘.‘prijs_ratio‘,
‘Group‘.‘permissions‘
FROM ‘users‘ AS ‘User‘ LEFT JOIN ‘groups‘ AS ‘Group‘
ON (‘User‘.‘klant_categorie_id‘ = ‘Group‘.‘id‘)
WHERE 1 = 1 ORDER BY ‘User‘.‘bedrijfsnaam‘ asc LIMIT 20
```

Listing 5.7: Gegeneerde sql-query voor het sorteren van gebruikers volgens bedrijfsnaam.

Al deze mogelijkheden maken intern gebruik van de zogenaamde *Pagination* component die met CakePHP 1.2 meegeleverd wordt.

5.12 Winkelen

5.12.1 Navigatie

Buiten de blok met recentste artikelen op de voorpagina, is het ook mogelijk om een overzicht te krijgen van alle artikelen, van alle categorieën en van *één* specifieke categorie met alle artikelen van die categorie. Deze worden bereikt via de links in het menu, zoals besproken in sectie *Menu* op pagina 39.
Enkel de artikelen met *status* "voorradig"worden weergegeven in de winkel. Bovendien wordt automatisch voor elk artikel de juiste prijs weergegeven afhankelijk van de *prijs_ratio* van de groep van de gebruiker. Artikelen in promotie worden in een opvallender blok weergegeven.

5.12.2 Bestellen

Figuur 5.2 (pagina 40) toonde reeds hoe het winkelmandje van de klant eruit ziet. De klant kan artikelen toevoegen aan zijn winkelmandje door de artikelen er in te slepen. Hiervoor wordt gebruik gemaakt van de drag 'n drop functionaliteiten van de scriptaculous bibliotheek en de Ajax helper van CakePHP. Meermaals hetzelfde artikel toevoegen verhoogt de quantiteit. Artikelen kunnen ook vanuit het winkelmandje naar het verwijdervakje gesleept worden: hierdoor wordt de quantiteit verminderd.
In het winkelmandje worden alle gekozen artikelen, samen met hun prijs per stuk, en hoeveelheid weergegeven. Ook wordt de totaalprijs van alle artikelen getoond, en van zodra *één* of meerdere artikelen zich in het mandje bevinden verschijnt een *checkout* link. Wanneer hier op geklikt wordt, kan de bestelling definitief gemaakt worden. Na het bevestigen van een JavaScript popup kan de klant de gewenste leverdatum invullen. Deze wordt later gebruikt bij het plannen van de leveringen door de beheerder.

5.12.3 Bestellingsoverzicht

De gebruiker heeft ook een bestellingsoverzicht waarop hij reeds gemaakte bestellingen kan zien, en wat hun status is.

5.12.4 Transacties

Wanneer een bestelling wordt opgeslaan in de databank, wordt een record aangemaakt voor de bestelling zelf, en de nodige records voor de objecten van klasse GekochtArtikel. Dit gaat goed zolang er geen technische problemen optreden met de databank, maar wanneer toch problemen met de databank zouden optreden, kunnen sommige queries kunnen falen terwijl succesvol andere uitgevoerd worden.

Het werken met transacties kan dit probleem oplossen, maar helaas zijn transacties nog niet geïmplementeerd in het CakePHP raamwerk. Deze functionaliteit komt er spoedig aan[12] maar kan momenteel dus nog niet gebruikt worden.

[12]Wanneer de finale versie van 1.2 vrijgegeven wordt

5.13 Leveringsplanner

5.13.1 Inleiding

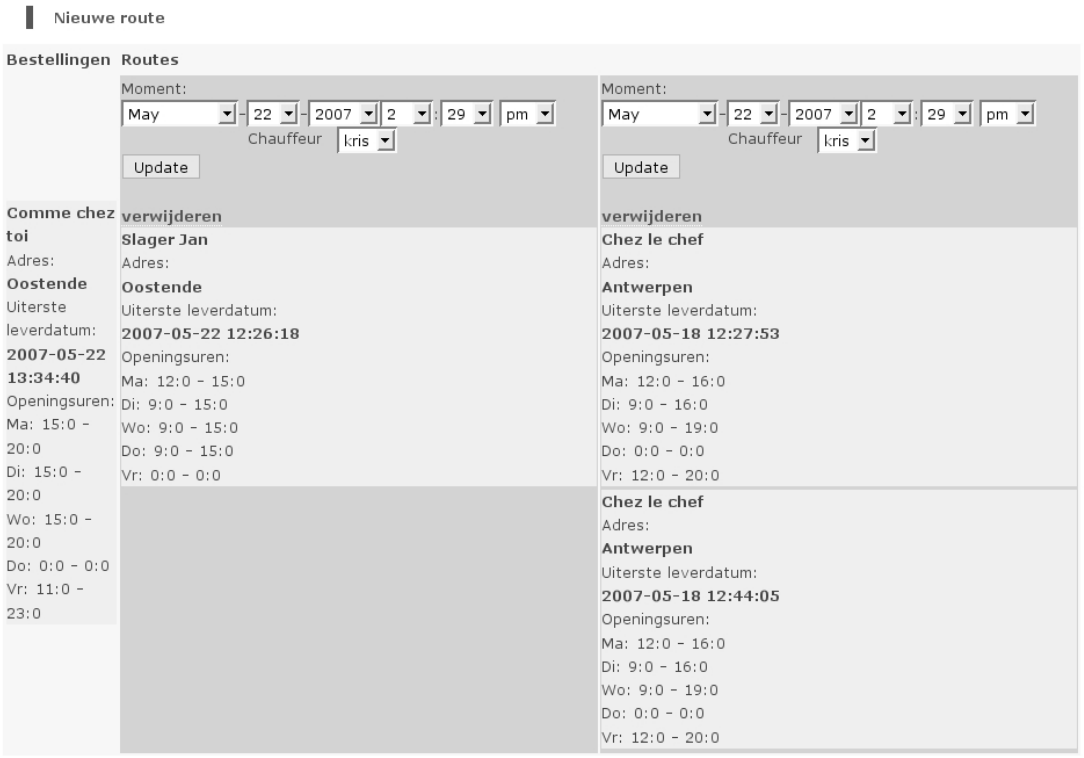
De leveringsplanner is de interface voor de beheerder waar leveringen van bestellingen kunnen gepland worden. Er wordt daarvoor gewerkt met de volgende entiteiten:

- routes
- bestellingen
- leveringen
- chauffeurs

5.13.2 Werking

Een screenshot van de leveringsplanner wordt getoond op pagina 56. Bestellingen met de status *in verwerking* waar nog geen levering is aangekoppeld, worden in de linkerkolom weergegeven. Per bestelling wordt de naam van de klant, de uiterste leverdatum, het adres, en de openingsuren van de klant weergegeven.
Rechts ervan worden de routes afgebeeld: elke route wordt voorgesteld door een kolom. Bovenaan elke route wordt de leverdatum en de naam van de chauffeur gekozen.
Wanneer een bestelling uit de linkerkolom naar *éé*n van de routes gesleept wordt, wordt automatisch een object *Levering* gemaakt die de bestelling aan de route koppelt. Wanneer een bestelling van de ene route naar de andere wordt gesleept, wordt de levering bijgewerkt, en wanneer een bestelling terug naar de linkse kolom wordt gesleept, wordt de levering verwijderd.
Er kunnen zoveel routes aangemaakt worden als nodig en wanneer een route verwijderd wordt, worden automatisch de bijhorende leveringen gewist. (De bestellingen op zich uiteraard niet)

De planner is steeds synchroon met de toestand in de databank. m.a.w. bij elke sleepactie worden automatisch op de achtergrond de nodige operaties uitgevoerd, en bij het openen van de planner wordt automatisch de toestand geladen zoals deze eruit zag bij de laatste bewerking.



Figuur 5.11: De leveringsplanner

5.14 Unit Tests

Het CakePHP raamwerk bevat een test suite. Om hiervan gebruik te maken moeten de volgende stappen ondernomen worden:

- verhoog de *DEBUG* level naar waarde 1,2 of3[13] in *webapp/private/plaetinck.be/config/core.php*
- configureer de instellingen voor de test-databank in *webapp/private/plaetinck.be/config/database.php*
- ga naar *http://hostnaam*/test.php

De locaties van de test cases zijn als volgt:

locatie	soort
webapp/private/plaetinck.be/tests/cases/	Applicatiespecifieke tests
webapp/private/cake/tests/case	Tests op het raamwerk

Tabel 5.3: Locaties van testcases

Ter illustratie volgt een testcase voor het Model Artikel. Dit is een eenvoudige test die controleert of alle records van klasse Artikel ofwel een gewone prijs, ofwel een promotieprijs ingesteld hebben.

```
loadModel('Artikel');

class ArtikelTest extends Artikel {
        var $name = 'Artikel';
        var $useDbConfig = 'test_suite';
}

class ArtikelTestCase extends UnitTestCase {
        var $object = null;

        function setUp() {
                $this->object = new ArtikelTest();
        }
        function tearDown() {
                unset($this->object);
        }
        function testPriceSet() {
                $result = $this->object->findAll();
                $failure = false;
                foreach($result as $a){
                        if(!$a['Artikel']['promotieprijs'] && !$a['Artikel']['
                            prijs']) $failure = true;
                }
```

[13]Zowel 1,2 als 3 zijn geschikte debuglevels. Hun betekenis wordt uitgelegd in de commentaar in het bestand

```
                $this->assertFalse($failure);
        }
}
```

Listing 5.8: een Testcase voor klasse Artikel

Hoofdstuk 6

Het semantisch web

6.1 Inleiding

De oplossing die is gecreëerd in deze masterproef bestaat erin dat gebruikers zelf, via hun webbrowser hun bestellingen invoeren. Ook andere taken, zoals het opvragen van artikelen, het aanpassen van hun profiel en dergelijke, gebeuren allemaal via een interface die door de gebruiker zelf bediend moet worden.
Het doel van dit hoofdstuk bestaat erin te kijken hoe we deze interface generieker kunnen maken, zodat hij niet enkel kan gebruikt worden door mensen, maar ook kan aangesproken worden door andere software.
Immers, in de moderne, hedendaagse wereld probeert de mens zoveel mogelijk werk voor zich te laten doen door machines (computers). Hiervoor is vaak data van een of meer externe bronnen nodig. Dit impliceert dat er een voorziening moet zijn om computerprogramma's met elkaar via een netwerk (het Internet) te laten communiceren, en dat deze computerprogramma's begrip hebben van de inhoud van data, en hierop gepast kunnen reageren. Dit leidt ons tot het Semantisch Web. Maar het semantisch web is uiteraard veel meer dan dat.

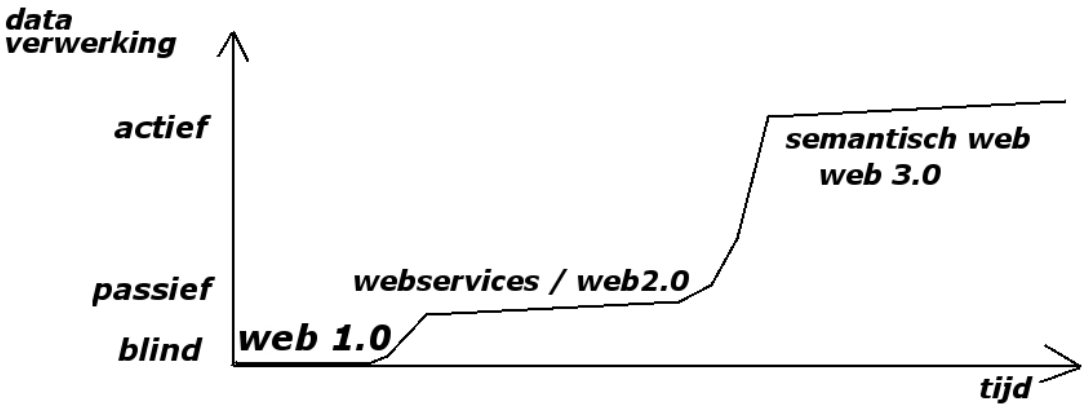
6.2 De basisprincipes

Het semantisch web[1] is een concept dat, hoewel het al enkele praktische verwezelijkingen omvat, zich nog grotendeels in onderzoeksfase bevindt. Momenteel is het W3C[2] onder leiding van Tim Berners-Lee volop bezig met de ontwikkeling van het web in het algemeen, en technieken rond het Semantisch Web in het bijzonder.

Momenteel bevinden we ons al in een overgangsfase: tot voor kort was het web vooral bruikbaar om informatie aan de mens kenbaar te maken: de software vraagt de data op, en toont deze aan de gebruiker (webbrowser) of indexeert deze (zoekmachine). De software heeft echter nog geen weet van de semantische waarde van de opgevraagde

[1] http://www.w3.org/2001/sw/
[2] World Wide Web Consortium: http://www.w3.org/

gegevens. Vandaar de keuze voor de benaming "blinde verwerking" op de figuur. Het aspect van informatie opvragen en tonen aan de gebruiker zal nog niet verdwijnen uit het web, maar er komt echter wel functionaliteit bij.
Waar we naar evolueren is een web waar de informatie ook voor machines toegankelijk wordt gemaakt. Webservices, zoals heden ten dage vaak gebruikt worden op het net maken het mogelijk voor applicaties om informatie "passief te verwerken". De opkomst van web2.0[24] heeft hier ook een grote bijdrage aan geleverd. Gegevens worden door software uitgewisseld en er is enige kennis omtrent de aard van de gegevens. Echter de software is nog niet in staat om op een intelligente manier de gegevens te interpreteren en verwerken.
En dat is exact het doel van het Semantisch Web: door middel van een goed uitge-



Figuur 6.1: Het web wordt intelligenter

dachte structuur, protocollen en entiteiten een web opzetten dat zich leent tot zulke geavanceerde mogelijkheden als applicaties die met elkaar communiceren, informatie extraheren en interpreteren en op een intelligente manier verwerken, met als uiteindelijk doel het dienen van de mens.
Zoals verder zal blijken is het uiteraard mogelijk dat applicaties met onbetrouwbare applicaties communiceren, en/of foutieve informatie ophalen. Trust is dus een essentieel begrip in deze context en wordt verderop in het hoofdstuk belicht.

6.3 Essentiele principes

Het semantisch web steunt op twee essentiele principes: "partial understanding" en "inference"[26]

6.3.1 partial understanding

Wanneer data opgevraagd wordt, kunnen hierin allerlei beweringen staan. (in bijvoorbeeld RDF, zie volgende pagina) Het kan gebeuren dat er entiteiten vernoemd worden

die nog niet bekend zijn. Bewerkingen die voorlopig niet geïnterpreteerd kunnen worden, worden beter bewaard voor wanneer dat wel kan.

6.3.2 inference

Inference betekent gevolgtrekking. Het komt erop neer dat applicaties nieuwe beweringen kunnen genereren adhv de beweringen die ze al kenden (en vertrouwen).

6.4 Essentiele technieken

6.4.1 de URI

De URI, Universal Resource Identifier[3] identificeert resources op het web. De URL, Universal Resource Locater is het meest bekende voorbeeld hiervan. Op het web kan via een URL zowel een resource geïdentificeerd als gelokaliseerd worden. In het semantisch web spelen URI's een belangrijke rol, het zijn de bouwstenen om elementen te identificeren op het web. Een belangrijk aspect van URI's is dat ze niet gecentraliseerd zijn. Dit betekent dat eender wie een URI kan aanmaken voor een resource, zelfs al bezit deze die niet. De gevolgen hiervan zijn dat meerdere URI's hetzelfde object kunnen representeren, bovendien kan niet uitgezocht worden welke URI's naar dezelfde objecten verwijzen. Belangrijk om hierbij op te merken is dat een URI zowel een fysiek item kan representeren, als de webpagina die over dat item gaat.[20]

6.4.2 XML

XML, de Extensible Markup Language[4] is een taal die ontworpen is om allerhande inhoud te publiceren. De nadruk ligt sterk bij de semantische waarden en een stricte structuur. Dit maakt dat XML-documenten eenvoudig leesbaar en interpreteerbaar zijn door machines en/of applicaties. Om problemen te voorkomen wanneer elementen met dezelfde naam in verschillende documenten voorkomen, maar niet hetzelfde object voorstellen, kunnen URI's gebruikt worden als elementnaam. Dit wordt XML namespacing genoemd.

6.4.3 RDF

We hebben URI's om objecten te identificeren, en we hebben een krachtige taal ter beschikking die zich goed laat bewerken en interpreteren door computerprogramma's. De volgende stap is een techniek hebben waarmee beweringen gemaakt kunnen worden die door applicaties echt kunnen geïnterpreteerd worden. Hiervoor is RDF ontwikkeld, het Resource Description Framework[5]. Er zijn twee manieren om met RDF informatie op te slaan, in XML, of in een eenvoudigere plain-text manier genaamd n-triples, wat een subset is van Notation-3.[11] en vaak verkozen wordt bovenop XML omdat het

[3] RFC: http://www.ietf.org/rfc/rfc2396.txt
[4] `http://www.w3.org/XML/`
[5] http://www.w3.org/RDF/

korter en handiger is.[25]
Een eenvoudig rdf-statement in n-triples is dit:
<http://dieter.be> <http://example.org/terms/likes> <www.w3.org/2001/sw> .
Deze uitspraak bevat een

- onderwerp
- predikaat
- lijdend voorwerp

Op deze manier kan dus een lijdend voorwerp aan een onderwerp gekoppeld worden, dmv een predikaat.[6] Hierdoor kan een applicatie de informatie interpreteren. De applicatie kan de informatie nog niet begrijpen (dat is een volgende stap) maar toch al interpreteren.

Aangezien iedereen zomaar alles over iedereen op het web kan publiceren kunnen zich situaties voordoen waarin bijvoorbeeld verschillende bronnen tegenstrijdige berichten plaatsen. Dit is waarom het aspect trust zo belangrijk wordt.

6.4.4 Ontologieën en schema's

Om nu ook nog ervoor te zorgen dat applicaties beter met informatie kunnen werken, moet informatie toegevoegd worden. Op deze manier kan een applicatie weten wat een specifieke term betekent of hoe hij moet gebruikt worden. Hiervoor kan gebruik gemaakt worden van ontologieën en schema's. Schema's en ontologieën leggen de betekenis en relaties vast van termen.

6.4.5 OWL en DAML

Om nu de betekenis en relaties (ontologieën) vast te leggen van termen, is een taal nodig. DAML[7] is hier een taal voor. Later werd OWL[8] ontwikkeld.
Beiden zijn gebaseerd op RDF.
In praktijk kunnen relaties zoals omgekeerden, restricties, lijsten, datatypes enz vast gelegd worden. Hierdoor wordt het mogelijk gemaakt voor programma's om bepaalde redeneringen te maken en conclusies te trekken, afhankelijk van bepaalde gegevens.

6.4.6 Logica en bewijzen

Dit zijn gebieden van het semantisch web maar momenteel nog veel onderzoek naar gebeurt en niet vast ligt.

We hebben gezien hoe we met bepaalde relaties (subklasse, omgekeerde,...) applicaties verbanden kunnen laten trekken. We kunnen echter nog verder gaan, uiteindelijk

[6]Een predikaat beschrijft de relatie. Het bestaat uit een werkwoord en een of meer andere woorden.
[7]de DARPA Agent Markup Language: `http://www.daml.org/2001/03/daml+oil-walkthru`
[8]Web Ontology Language`http://www.w3.org/TR/owl-features/`

is het ook de bedoeling dat applicaties zo goed mogelijk voor zichzelf kunnen denken, vertrekkend vanaf logische statements (die zich verder uitstrekken dan wat met DAML mogelijk is)

Als dit mogelijk is, moeten we ook applicaties dingen kunnen laten bewijzen: ze vertrekken vanaf logische gegevens, en berekenen zelf (via deductie) verdere resultaten.

6.5 Trust

Zoals aangehaald zijn er nog enkele praktische problemen waar oplossingen voor moeten bedacht worden: als iedereen op het web kan beweren wat hij/zij wil dan zal er gegarandeerd onjuiste en zelfs tegenstrijdige informatie gepubliceerd worden. Ook kunnen bijvoorbeeld personen zich uitgeven voor iemand anders.

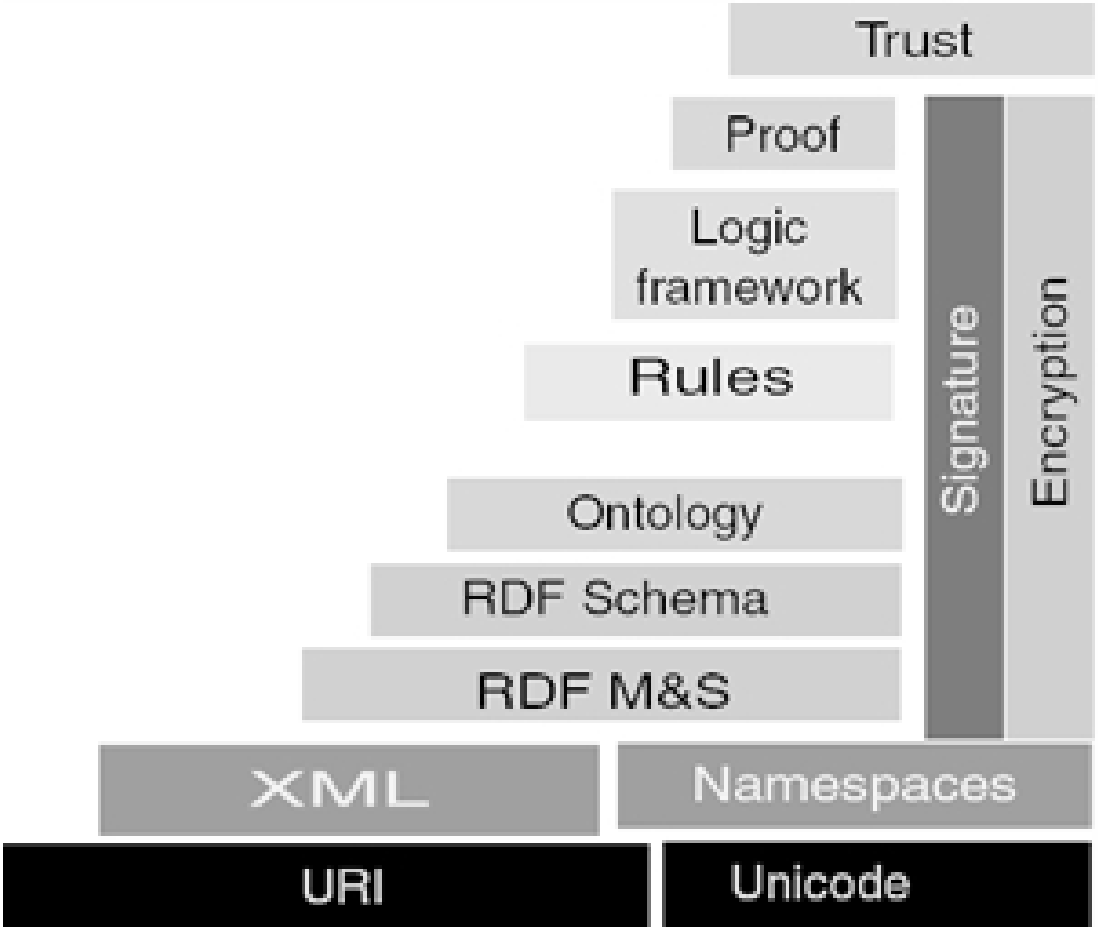
6.5.1 Wat is trust?

Trust betekent letterlijk "vertrouwen".[19] Hoezeer een bron of een stuk informatie wordt vertrouwd hangt af van verschillende factoren.[17] Enerzijds is er het authenticatieprobleem: is een persoon wel wie hij claimt te zijn? Komt deze informatie van de juiste persoon? Hiervoor bestaan oplossingen (zie verder) maar dit is nog niet voldoende. Anderzijds is het niet omdat een stuk informatie gegarandeerd van een bepaalde persoon komt, dat deze informatie ook vertrouwenswaardig is. Dit aspect is een stuk subjectiever, maar minstens even belangrijk. Bovendien is trust contextgebonden: wat een bepaalde persoon zegt over onderwerp A is misschien niet te vertrouwen, maar als dezelfde persoon iets zegt over onderwerp B kan dat een stuk vertrouwenswaardiger zijn. Evengoed moet het mogelijk zijn voor iemand die inhoud wil publiceren, om op deze inhoud restricties op te leggen, en er dus voor te zorgen dat bepaalde mensen de inhoud kunnen raadplegen en andere niet. Op het huidige web wordt dit probleem meestal aangepakt door een vorm van authenticatie samen met toegangscontrole om te beslissen wie wat mag doen. Deze toegangscontrole is echter centraal en dus niet niet geschikt voor een open, gedecentraliseerde werking. Er moet dus een nieuwe techniek ontworpen worden die al deze problemen opvangt op een gedecentraliseerde manier.

6.5.2 Digitale handtekeningen

Uit het werk van de vakgebieden cryptografie en wiskunde is het concept "digitale handtekening"[9] ontstaan. Dankzij de de digitale handtekening kan bewezen worden dat een persoon een document geschreven heeft (de zogenaamde niet-weerlegbaarheid of "non-repudiation"). Bovendien kan een digitale handtekening gebruikt worden om de integriteit van een document te bewaren: als een geëncrypteerd bestand wordt gewijzigd zal een verificatie de incorrectheid aan het licht brengen. Met RDF-beweringen kunnen ook identiteiten gedeclareerd worden. [10] Hiervoor kan een URI gebruikt worden die de publieke sleutel bevat.

[9] `http://en.wikipedia.org/wiki/Digital_signature`



Figuur 6.2: Semantisch Web stack

Om correcte informatie te garanderen in applicaties kan gezorgd worden dat enkel die handtekeningen vertrouwd worden van personen die zelf vertrouwd worden.
Meer geavanceerde implementaties zouden een systeem hebben waarbij een bepaalde hoeveelheid trust of paranoia kan toegewezen worden aan elke sleutel. Dan kan de computer uitmaken in welke mate bepaalde informatie betrouwbaar is.

6.5.3 Web of trust

Van een gebruiker kan niet verwacht worden dat hij alle sleutels gaat toevoegen die hij vertrouwt. Om een tamelijk deel van het web te benutten is een enorme hoeveelheid aan aanvaarde sleutels nodig. Daarom werd het begrip "web of trust"[23] (dat reeds langer bestond, bij PGP[16]) verder verfijnd.

Hoe het werkt

De gebruiker kan de mate van vertrouwen of wantrouwen specifieren voor de sleutels van zijn meest nabije bronnen. Van deze bronnen kan vervolgens ook opgevraagd worden welke sleutels door die bron vertrouwd zijn, en in welke mate, voor elke volgende bron gebeurt hetzelfde, enz. Op deze manier wordt voor elke persoon een uniek web opgebouwd waarin veel meer sleutels voorkomen en waarin gespecifieerd staat in welke mate elke sleutel vertrouwd of gewantrouwd is. Uiteindelijk kan dus voor een specifieke bron door de applicatie gecontroleerd worden in welke mate ze vertrouwd is, en gepast handelen.

Rapportering naar gebruiker toe

Uiteindelijk wordt via een vrij complex systeem dus de hoeveelheid vertrouwen van een bron berekend. Hoe hiermee wordt omgegaan en hoe dit aan de gebruiker wordt duidelijk gemaakt is een vak op zich. Er zijn verschillende manieren om dit aan de gebruiker te melden (geen, een eenvoudige goed/slecht quotering,.. tot een gedetailleerde uitleg van de specifieke boom van vertrouwen naar de specifieke bron)

Een voorstel van Tim Berners-Lee is de zogenaamde "Oh, yeah?" knop[9]. Wanneer hierop geklikt wordt, zou de applicatie een redenering geven om de hoeveelheid vertrouwen te staven.

6.5.4 Friend of a friend

Momenteel wordt hard gewerkt aan een implementatie van een dergelijk systeem: het zogenaamde "friend of a friend"[10][15].
Hierbij wordt via RDF-opgemaakte XML bestanden informatie gepubliceerd over personen. Er kunnen elementaire eigenschappen (zoals naam, e-mail adres,. . .) instaan, maar ook relaties met andere personen op een relatief granulaire manier (ouder van,

[10] `http://www.foaf-project.org/`

vriend van, samen op een foto,...) van personen connecties gemaakt tussen personen. Er is echter nog steeds geen besluit genomen over hoe de URI's er moeten uit zien die personen moeten voorstellen.
Een mogelijkheid is dat openID URI's (zie volgende sectie) hiervoor gebruikt zullen worden.[5]

6.5.5 OpenID

Een hedendaagse implementatie van een gedecentraliseerd authenticatiesysteem is OpenID.[11] Dmv van OpenID kan een gebruiker een URI kiezen waardoor hij wil gerepresenteerd worden. Wanneer de gebruiker zich wil authenticeren bij bvb een webapplicatie, dan zal deze webapplicatie A contact opnemen met applicatie B op de door de gebruiker gespecifieerde URL. Pas wanneer de gebruiker aan B zijn toestemming heeft gegeven om zich te authenticeren bij A, zal B dit doorgeven aan A, en is aldus de identiteit van de gebruiker bewezen bij A.

Dit systeem zorgt er echter niet voor dat berichten getekend kunnen worden. In tegenstelling tot bij gebruik van een digitale handtekening, kan bij OpenID door eender wie beweerd worden dat hij of zij een bericht bezit dat geschreven is door een bepaalde gebruiker. Wegens het ontbreken van een sleutel kan dit niet geverifieerd worden.

Het OpenID systeem ontbreekt dus nog enkele gewenste functionaliteiten die essentieel zijn voor de gewenste mogelijkheden van het Semantisch Web. Het is echter een stap in de richting naar het Semantisch Web, dat langzaamaan meer vorm krijgt.

6.6 het Semantisch Web in context van deze masterproef

Het Semantisch Web staat nog in zijn kinderschoenen: er wordt nog volop gespeculeerd over standaarden, technologieën worden nog volop ontwikkeld en de eerste praktische implementaties beginnen langzaam aan vorm te krijgen.

We willen dus kijken naar hoe dit project kan kaderen in het Semantisch Web, of anders gezegd: hoe zou de webapplicatie kunnen uitgebreid worden zodanig dat deze gebruikt kan worden op een manier die conform is aan de eigenschappen van het Semantisch Web.

Een eerste stap zou kunnen zijn authenticatie: een implementatie van het openID systeem zodat gebruikers hiervan gebruik kunnen maken en eens ingelogd winkelen zoals dat nu ook mogelijk is.

[11] `http://openid.net/`

Deze methode is een kleine verbetering, maar lost het probleem nog niet op: de bedoeling is uiteindelijk dat de software van de klant bestellingen kan plaatsen in de webwinkel. Dit kan zowel expliciet door de klant opgedragen zijn (via clientsoftware) maar evengoed zou software dit autonoom kunnen doen. (gebaseerd op bijvoorbeeld stockvoorraden of andere parameters).

Om dit probleem aan te pakken, kan begonnen worden met een web of trust uit te bouwen vertrekkend vanuit de webwinkel zelf. Klanten kunnen, na registratie en goedkeuring door de beheerder, door de beheerder in het web of trust geplaatst worden.

Van hieruit is het enkel nog kwestie van de bestellingen van de klanten te bezorgen aan de webwinkel (en dat op een manier die confidentialiteit en integriteit garandeert. Een mogelijkheid hiervoor zou zijn dat de klant een soort profielpagina heeft: deze kan gespecifieerd worden (dmv RDF) op de identiteitspagina (de URI hiervan is die van de openID identiteit) van de gebruiker. Op deze profielpagina kan dan informatie over bestellingen geplaatst worden. Een alternatief zou zijn dat bestellingen rechtstreeks als elementen op de identitetispagina van de gebruiker geplaatst worden.

Wanneer bestellingen geplaatst worden met een sleutel die enkel gedeeld wordt door de webwinkel en de klant, is de integriteit en confidentialiteit gegarandeerd.
Eventueel kan ook gebruik gemaakt worden van assymmetrische encryptie.[12]

Het enige wat dan nog moet gebeuren, is de webapplicatie die autonoom op regelmatige tijdstippen zijn web of trust afspeurt op zoek naar nieuwe bestellingen. Wanneer deze gevonden worden kunnen deze verwerkt worden op de gebruikelijke manier, of -uiteraard- ook op een betere “Semantisch Web”-manier.

[12]In dat geval moet door de klant of door de clientsoftware het bericht geëncrypteerd worden met de publieke sleutel van de webwinkel.

Hoofdstuk 7

Besluit

7.1 Ervaringen

De opdracht, opgesteld door Plaetinck D. Vleeshandel NV was het bouwen van een webapplicatie die functioneert als informatieve website, maar vooral ook een webwinkel huisvest.

Een taak die op het eerste zicht eenvoudig leek, maar waar toch al snel vele aspecten bij kwamen kijken. Zo zijn verschillende bestaande webwinkels onderzocht, zijn allerlei programmeertalen, raamwerken, applicatie-architecturen en programmeerpatronen bestudeerd, is er voor de server een nieuw platform uitgekozen, is onderzoek gebeurd naar allerhande aspecten omtrent veiligheid en privacy, en is onderzocht hoe een applicatie als deze zou kunnen kaderen in het Semantisch Web.

Dat dit project enorm leerrijk is gebleken, zal dan ook niet als een verrassing komen. Ik heb veel bijgeleerd over zowel bestaande technologieën, als die van de toekomst. (Zo is het Semantisch Web een technologie waar ik naar uitkijk)

Ook heeft dit eindwerk voor bepaalde inzichten over het bedrijfsleven gezorgd. Een van de dingen die me zeker is opgevallen, is hoe een bedrijf volledig ondgergedompeld zou kunnen worden in de ICT-wereld. Alles kan verbeterd worden, overal kan ICT een meerwaarde betekenen.

7.2 Resultaat

De huidige oplossing kan zeker als solide basis gebruikt geworden, maar verdere ontwikkeling is echter sterk aan te raden vooraleer de applicatie in productie te brengen.

7.3 Verbeteringen en uitbreidingen

7.3.1 Noodzakelijk

De items in deze sectie zijn noodzakelijk. Zonder de verbeteringen en uitbreidingen die hier opgesomd worden hoort dit project niet in productie gebracht te worden:

- Privacyverklaring: wegens wettelijke verplichtingen is het een must om een privacy verklaring te gebruiken voor dit soort doeleinden. De verklaring moet bovendien goedgekeurd zijn door een officiële instantie.[3]
- Het uploadscript: dit hoeft niet gebruikt te worden, maar als het gebruikt wordt, is het raadzaam het te verbeteren op enkele gebieden. (geëncrypteerd opslaan en verzenden van wachtwoord, fout-afhandeling, rapportage naar de beheerder,. . .)
- Https: Vooral voor gebruikers zonder JavaScript is dit essentieel voor een veilige authenticatie, maar alle gebruikers hebben er baat bij als alles geëncrypteerd verstuurd wordt (privacy)
- Transacties: momenteel wordt nog gewerkt zonder transacties. Zolang er geen fouten optreden geeft dit geen problemen, maar hier mag niet vanuit gegaan worden. Deze functionaliteit zal geïmplementeerd worden in het CakePHP raamwerk, maar is nu nog niet aanwezig.
- Prijsaanpassingen: Als de prijs wordt aangepast door de beheerder net nadat de klant het heeft toegevoegd aan zijn bestelling, maar de bestelling nog niet is voltooid, zou de klant hier een melding van moeten krijgen.

7.3.2 Optioneel

De items in deze sectie zijn optioneel: het zijn extra's die een meerwaarde kunnen betekenen maar die niet noodzakelijk zijn voor een degelijke en betrouwbare werking van de applicatie.

- aparte print-layout: printvriendelijke pagina's met minder kleuren, hoger contrast en zonder menu's, banners,. . .
- rss-feeds van artikelen, nieuwsberichten,. . . verhoogt de flexibiliteit naar de gebruikers toe.
- pdf-generatie: rapporten voor de beheerder met bestellingen, of rapporten voor de chauffeurs met daarop hun route en de bestellingen die ze moeten uitvoeren.
- E-mail notificaties zouden klanten kunnen op de hoogte brengen wanneer ze zijn aanvaard in het systeem, of wanneer de status van hun bestelling is veranderd.
- Nu wordt gewerkt met eenvoudige prijs-ratio's die de prijs van elk artikel met *één*zelfde factor vermeningvuldigen. Het werken met aparte prijslijsten voor elke groep zou meer controle geven aan de beheerder.

Bijlage A

Berekening serververeisten

A.1 Inleiding

In dit hoofdstuk wordt getracht een beeld te krijgen van de benodigde bandbreedte en opslagruimte voor de server. Aangezien het door allerlei factoren (onvoorspelbare serverbelasting, moeilijk te voorspellen bestandsgroottes,...) niet mogelijk is om zoiets accuraat uit te rekenen werd gebruik gemaakt van schattingen.
De schattingen werden ruim gekozen zodat er een "worst case scenario" wordt voorgesteld. Wanneer verschillende methoden mogelijk zijn, wordt de intensiefste gekozen (bvb webmail ipv pop-toegang)

Een bijkomend probleem is dat er twee standaarden worden gebruikt omtrent het werken met hoeveelheden bytes, namelijk de SI en IEC standaarden. [1] In deze berekening is voor de IEC-standaard gekozen, deze is niet alleen de recentste, maar sluit bovendien meest aan bij het worst-case-scenario omdat het delen door 1000 tot hogere resultaten leidt dan bij 1024.

A.2 Veronderstellingen

A.2.1 Algemeen

eenheid	hoeveelheid
vaste aanbiedingen tegelijk aanwezig	100
unieke aanbiedingen tegelijk aanwezig	100
bezoekers per maand [1]	500
aantal thumbnails bekeken per bezoek	150
aantal grote foto's bekeken per bezoek	15
aantal grote html pagina's bezocht per bezoek	15
aantal kleine html pagina's bezocht per bezoek	15
aantal Ajax requests per bezoek	10
aantal Ajax requests per dag voor beheerder	100
aantal grote html pagina's bezocht per dag voor beheerder	100
aantal kleine html pagina's bezocht per dag voor beheerder	100
levensduur vaste aanbieding (in dagen)	30
levensduur unieke aanbieding (in dagen)	2
aantal e-mail accounts	4
type e-mail	webmail
aantal keer mail bekijken per gebruiker per dag	100
aantal keer mail lezen of schrijven per gebruiker per dag	50
grootte bijvoegsel (*attachment*)	5 MB
aantal keer up/downloaden bijvoegsel per gebruiker per dag	5

Tabel A.1: Algemene veronderstellingen voor berekening serververeisten

Er wordt geen rekening gehouden met het gebruik van caching. M.a.w. we gaan er vanuit dat de browser van de klant dezelfde afbeeldingen steeds opnieuw downloadt.

A.2.2 Bestandsgroottes

soort	grootte (kB)
thumbnail aanbieding	10
afbeelding aanbieding	100
webapplicatie [2]	$10 * 10^3$
mailbox	$50 * 10^3$

[1] Bezoek: naar de website gaan, verschillende pagina's en artikels bekijken, eventueel een bestelling doen, en het bezoek stoppen

[2] deze schatting is gemaakt door rekening te houden met de grootte van het CakePHP raamwerk, de benodige andere tools (TinyMCE, Prototype, Scriptaculous,...) , bijkomende bestanden en de bestanden van de huidige website.

A.2.3 Overdracht

De elementen die hier besproken worden, zijn gegevens die uitgewisseld worden over het netwerk, maar nog niet vroeger besproken zijn: zo worden bijvoorbeeld de html pagina's gegenereerd aan de hand van *één* of meerdere bestanden.
Items die exact worden doorgestuurd zoals ze zijn opgeslaan (zoals afbeeldingen) worden hier niet meer vermeld.

soort	grootte (kB)
kleine html pagina	5
grote html pagina	15
Ajax request en response	5

A.2.4 Andere

Het aanpassen van gegevens op de site, beheren van de webserver, e.d. zijn activiteiten die een minieme hoeveelheid bandbreedte eisen en verwaarloosd worden.

A.3 Benodigde opslagruimte

- Webapplicatie: $10 * 10^3$ kB = 10MB
- Foto's webwinkel: 100 * 10 kB + 100 * 100 kB = 11MB
- E-mail: 4 mailboxen * $50 * 10^3$ kB/mailbox = 200MB

Totaal $10 * 10^3$ kB + 100 * 10 kB + 100 * 100 kB + 4 * $50 * 10^3$ kB = 221 MB.

A.4 Benodigde bandbreedte

Bezoeker

500 bezoekers/maand* (150 thumbnails * 10kB/thumbnail + 15 grote foto's * 100kB/-grote foto + 15pagina's*(5+15)kB/pagina + 10 ajax requests * 5kB/ajax request) = 1675 MB/maand

Beheerder

- vervangen foto vaste aanbiedingen: (100 kB + 10 kB)*50 = 5.5 MB/maand
- vervangen foto unieke aanbiedingen: (100 kB + 10 kB)*50 * 15 = 82.5 MB/maand
- Ajax requests: 100 keer/dag * 5kB * 31 dagen/maand = 16 MB/maand
- Html pagina's: 100 * (5kB+15kB) = 2 MB/maand

E-mail

- pagina's (grote voor mail overzicht, kleine voor mail lezen/schrijven): (100*15kB + 50*5kB)* 4 gebruikers * 31 dagen = 217 MB/maand
- bijvoegsels: 4 gebruikers * 2 bijvoegsels/dag * 5 MB/bijvoegsel * 31dagen/maand = 1240 MB/maand

Totaal

1675 MB/maand + (5.5+82.5+16+2) MB/maand + (217+1240) MB/maand = 3238 MB/maand

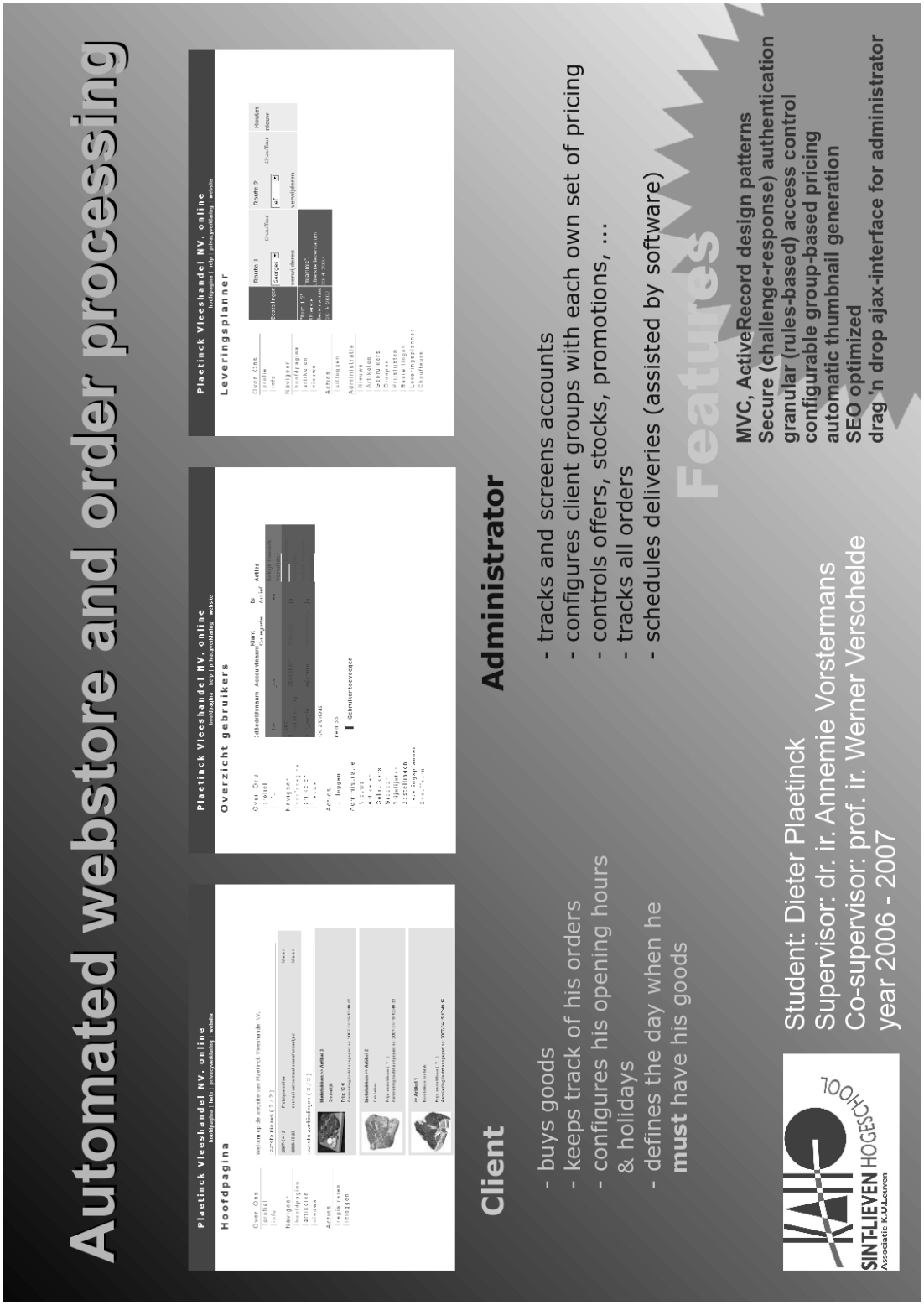
Bijlage B

Beschrijving van deze masterproef onder de vorm van een wetenschappelijk artikel

Bijlage C

Poster

De poster op de volgende pagina werd gebruikt ter promotie van deze masterproef op de opendeurdag van KaHo Sint-Lieven.



Figuur C.1: Poster opendeurdag

Bijlage D

Inhoud van de CD-rom

De bijgevoegde cd-rom bevat:

- De benodigde code inclusief benodigde bibliotheken
- Deze scriptie in pdf-formaat
- De Engelstalige paper (bijlage B)
- De poster (bijlage C)

Bijlage E

Installatie instructies

E.1 Benodigdheden

- webserver met PHP ondersteuning. Versie 4 is voldoende maar PHP5 valt aan te raden.
- SQL-server. Bij voorkeur MySQL.
- Optioneel: Xfce 4.4 (werkt op Linux,BSD, Mac OS, cygwin,...) voor de automatische uploads van afbeeldingen.

E.2 Uit te voeren stappen

- Kopieren van de webapp map naar de map die de webserver kan gebruiken. Bij voorkeur is enkel de map *www.plaetinck.be* zichtbaar voor de buitenwereld.[1]. Nodige bibliotheken en raamwerken zijn meegeleverd.
- Volgende mappen moeten beschrijfbaar zijn door de applicatie[2]:
  - *webapp/private/plaetinck.be/tmp*
  - *webapp/www.plaetinck.be/img/imgcache*
  - *webapp/www.plaetinck.be/img/originelen*
- Importeren van het sql bestand in de databank.
- Aanpassen van *webapp/private/plaetinck.be/config/database.php*
- Optioneel:
  - Het shellscript in een locatie naar keuze plaatsen.
  - De parameters in het shellscript aanpassen
  - Thunar Volman configureren om het shellscript te gebruiken. De opdracht hiervoor is: *locatie-van-het-script %m*

[1] Op de Apache server kan dit mbhv virtualhost
[2] Op Linux kan dit dmv het chmod commando

Lijst van figuren

Lijst van tabellen

Listings

Bibliografie

[1] Veelvouden van bytes. *WikiPedia (http://nl.wikipedia.org/wiki/Veelvouden_van_bytes)*.

[2] The cross site scripting (xss) faq. *Cgisecurity.com (http://www.cgisecurity.com/articles/xss-faq.shtml)*, 2002.

[3] Privacy. *Belgische Overheid* *http://privacy.fgov.be/*, 2004.

[4] Php security guide. *PHP Security Consortium (http://phpsec.org/projects/guide)*, 2005.

[5] Openid for the semantic web. *Rojo Networks* *http://www.rojo.com/story/SBCD09egSFE68c4q*, 2006.

[6] What is authentication? *Search Security* *http://searchsecurity.techtarget.com/sDefinition/0,290660,sid14_gci2116%21,00.html*, 2006.

[7] Article: Get started with dauth v0.3 authentication system. *International PHP Magazine* *http://www.php-mag.net/magphpde/magphpde_news/psecom,id,26679,nodeid,5.%html*, 2007.

[8] Browser statistics: Web statistics and trends. *W3 Schools* *http://www.w3schools.com/browsers/browsers_stats.asp*, 2007.

[9] Tim Berners-Lee. Cleaning up the user interface. *W3* *http://www.w3.org/DesignIssues/UI.html*, 1997.

[10] Tim Berners-Lee. Semantic web road map. *W3* *http://www.w3.org/DesignIssues/Semantic*, 1998.

[11] Tim Berners-Lee. Notation 3, a readable language for data on the web. *W3* *http://www.w3.org/DesignIssues/Notation3*, 2006.

[12] David A. Black. Ruby and rails: In your face... but out of your way. *InfoQ (http://www.infoq.com/articles/Ruby-and-Rails-In-your-face)*, 2006.

[13] Dana Blankenhorn. Control is the real open source advantage. *Zdnet (http://blogs.zdnet.com/open-source/?p=844)*, 2006.

[14] Fabio Cevasco. Rails-inspired php frameworks. *Herald.com (http://www.infoq.com/articles/Ruby-and-Rails-In-your-face)*, 2006.

[15] Leigh Dodds. An introduction to foaf. 2004.

[16] Patrick Feisthammel. Explanation of the web of trust of pgp. *Rubin.ch http://www.rubin.ch/pgp/weboftrust.en.html*, 2004.

[17] Timothy Finin and Anupam Joshi. Agents, trust, and information access on the semantic web. 2002.

[18] Felix Geisendörfer. A lightweight approach to acl - the 33 lines of magic. *ThinkingPHP.org http://www.thinkingphp.org/2006/10/03/a-lightweight-approach-to-acl-the%-33-lines-of-magic/*, 2006.

[19] Yolanda Gil and Varun Ratnakar. Trusting information sources one citizen at a time. *USC Information Sciences Institute*, 2002.

[20] Sandro Hawke. How we identify things (on the semantic web) ? *W3C http://www.w3.org/2001/03/identification-problem/*, 2001.

[21] Byte internetdiensten. Php sql injection. *Byte internetdiensten (http://www.byte.nl/docs/Php-Sql-Injection.html)*.

[22] Ivo Jansch. Quick zend framework review. *Achievo.org blog (http://www.achievo.org/blog/archives/29-Quick-Zend-Framework-review.ht%ml)*, 2006.

[23] James Hendler Jennifer Golbeck, Bijan Parsia. Trust networks on the semantic web. *University of Maryland*, 2003.

[24] Tim O'Reilly. What is web 2.0? *O'Reilly http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web%-20.html*, 2005.

[25] Sean B. Palmer. a roughguide to notation3. *Infomesh http://infomesh.net/2002/notation3/*.

[26] Sean B. Palmer. The semantic web, taking form. *Infomesh.net http://infomesh.net/2001/06/swform/*.

[27] Gijs Van Tulder. Storing hierarchical data in a database. *Sitepoint.com (http://www.sitepoint.com/article/hierarchical-data-database)*, 2003.

[28] Philip E. Varner. The economics of open source. *University of Maryland (http://www.cs.virginia.edu/~pev5b/writing/econ_oss/advantages.html)*, 1999.