Newyse CMS

Afstudeerscriptie

Naam: Elwin Vreeke

Werkgever: Maxxton

Begeleider Maxxton: Dhr. Jean-Pierre Mampaey

Universiteit: Technische Universiteit Delft

Begeleider TU Delft: Dr. Kees van der Meer

Inhoud

1. Samenvatting

In een tijdsbestek van ongeveer anderhalf jaar is een idee over een uitbreiding van het Newyse backoffice pakket met een frontoffice gedeelte uitgegroeid tot een volwaardig product.

Voor aanvang van het project was Newyse maar op 1 manier ontsloten naar het internet toe: via een SOAP interface. De doelstelling van dit project is geweest om dit verder uit te breiden en Newyse te voorzien van een internet gedeelte met een compleet CMS systeem dat volledig geïntegreerd is in Newyse. Om dit te realiseren is het project opgedeeld in drie verschillende delen:

1) CMS keuze
2) Integratie
3) Pilot project

1) Eén van de belangrijkste eisen aan het CMS was dat het mogelijk was om het CMS aan te kunnen passen en uit te breiden zodanig dat een integratie met Newyse gerealiseerd kon worden. De licentie moet het dan toestaan dit aangepaste CMS in de markt te zetten. Aangezien er in de organisatie veel kennis is van Java is het ook een eis dat het een CMS zou zijn dat in Java ontwikkeld is.

Het CMS Liferay (Liferay, 2007) sluit het beste aan bij deze eisen. Het is een portal systeem dat het mogelijk maakt om losse componenten (portlets) te maken die elk een eigen stuk functionaliteit bieden.

2) Belangrijke technieken voor de integratie van Newyse en een CMS zijn iBatis (Apache iBatis, 2008) voor het afhandelen van de mapping van domain objecten naar de database entity objecten en Wicket dat een geschikte techniek is voor het ontwikkelen van de gebruikers interface.
Voor iBatis is voornamelijk gekozen vanwege de flexibiliteit in het gebruik van eigen SQL statements. Dit is een zwaarwegend element vanwege de specifieke kennis van SQL binnen het team. Wicket (Apache Wicket, 2008) is het juiste framework vanwege de overeenkomsten met Swing: het is '*model based*' en '*event driven*'. Het feit dat Wicket gebruik maakt van en server-side state is ook een voordeel bij het maken van reserveringen, waarbij het belangrijk is gedurende het gehele proces de juiste informatie te hebben. Voor de koppeling tussen Wicket en iBatis wordt er gebruik gemaakt van Spring (Spring, 2007).

Om het mogelijk te maken op de website een te boeken accommodatie te zoeken op basis van kenmerken is er besloten om gebruik te maken van SOLR (Apache SOLR, 2008). Deze op Lucene (Apache Lucene, 2008) gebaseerde zoekindex maakt het mogelijk om in de grote aantallen verschillende vakanties te zoeken op basis van allerlei uiteenlopende kenmerken, die op verschillende elementen van toepassing zijn.

3) Uiteindelijk is al het voorgaande toegepast in een pilot project om een daadwerkelijke integratie te realiseren. Voor “De Krim”, een klant van Maxxton en Newyse gebruiker, werd er een website ontwikkeld op basis van Liferay met een sterke integratie met Newyse.

Het resultaat is een website die een totaal oplossing biedt voor de klant. De klant heeft een uitgebreid CMS tot zijn beschikking waarmee het op een zeer flexibele manier de website kan inrichten en beheren. Door de integratie kan het zelf ook componenten op de website plaatsen die direct communiceren met Newyse waardoor er altijd actuele informatie op de website komt te staan.

De bezoeker van de website kan op een snelle en intuïtieve manier zoeken naar accommodaties op basis van actuele beschikbaarheid en kan dit omzetten in een reservering.

2. Inleiding

Dit document is het verslag van mijn afstudeerproject voor mijn opleiding Technische Informatica aan de Technische Universiteit Delft.

Mijn afstudeerproject heb ik uitgevoerd bij Maxxton uit Middelburg.

Al voor de periode van het afstudeerproject ben ik werkzaam bij Maxxton geweest. Ik was actief als Java ontwikkelaar voor het hoofdproduct van Maxxton: Newyse.
Newyse is een backoffice applicatie voor de recreatieve sector.

Begin 2007 ontstonden er serieuze ideeën om Newyse verder uit te breiden met een front office gedeelte. In de beginfase is er aanvankelijk veel gediscussieerd en nagedacht om te kunnen bepalen wat de doelstellingen en gevolgen van die uitbreiding zouden zijn. De eerste ideeën gingen nog uit van het uitbesteden of inkopen van een bestaand product waarmee Newyse gekoppeld zou kunnen worden. Tijdens deze voorbereidende fase is gebleken dat er geen geschikte partij te vinden was die een oplossing kon bieden. Vanaf dat moment is er besloten om het in eigen beheer te gaan ontwikkelen. Het idee was om een geïntegreerde website te kunnen bieden. De exacte specificaties van het gewenste eindresultaat waren nog niet gedefinieerd.

Vanaf dat moment ging het idee spelen om dit project in te zetten voor mijn afstudeerproject.
Het onderzoek dat uitgevoerd wordt als onderdeel van mijn afstudeerproject zou een geschikte oplossing voor dit probleem moeten aanwijzen. Van mij werd ook, tot op zekere hoogte, verwacht dat ik sturing zou geven aan de realisatie van dit project.

2.1. Maxxton

Maxxton is in 1998 ontstaan als een dienstverlenend bedrijf dat consultancy verleende op ICT-gebied aan bedrijven in de verblijfsrecreatiesector. De eerste klanten waren vakantieparken in de Benelux van uiteenlopende omvang.

De ervaring met consultancy leidde ertoe dat Maxxton in 2001 (toen nog onder de naam Recreanet) het bedrijf C.A.T. overnam, leverancier van een van de meest gangbare reserveringssystemen: Macro+. Dit stelde Recreanet in staat een grotere invloed te hebben bij het doorvoeren van wijzigingen, en voegde bij de kennis van de markt ook inhoudelijke kennis van automatiseringssystemen.

Het vermogen om zelf aan het reserveringssysteem te sleutelen kon echter niet de meest in het oog lopende beperkingen van Macro+ verhelpen: het was een systeem dat decentraal was opgezet. Voor grotere organisaties met meerdere parken en boekingskanalen schoot het tekort.

In die tijd is het plan opgevat zelf een nieuw systeem van de grond af aan op te bouwen. Dat werd Newyse. Vanaf 2003 is aan Newyse gebouwd en begin 2005 zijn de eerste klanten live gegaan met het pakket.

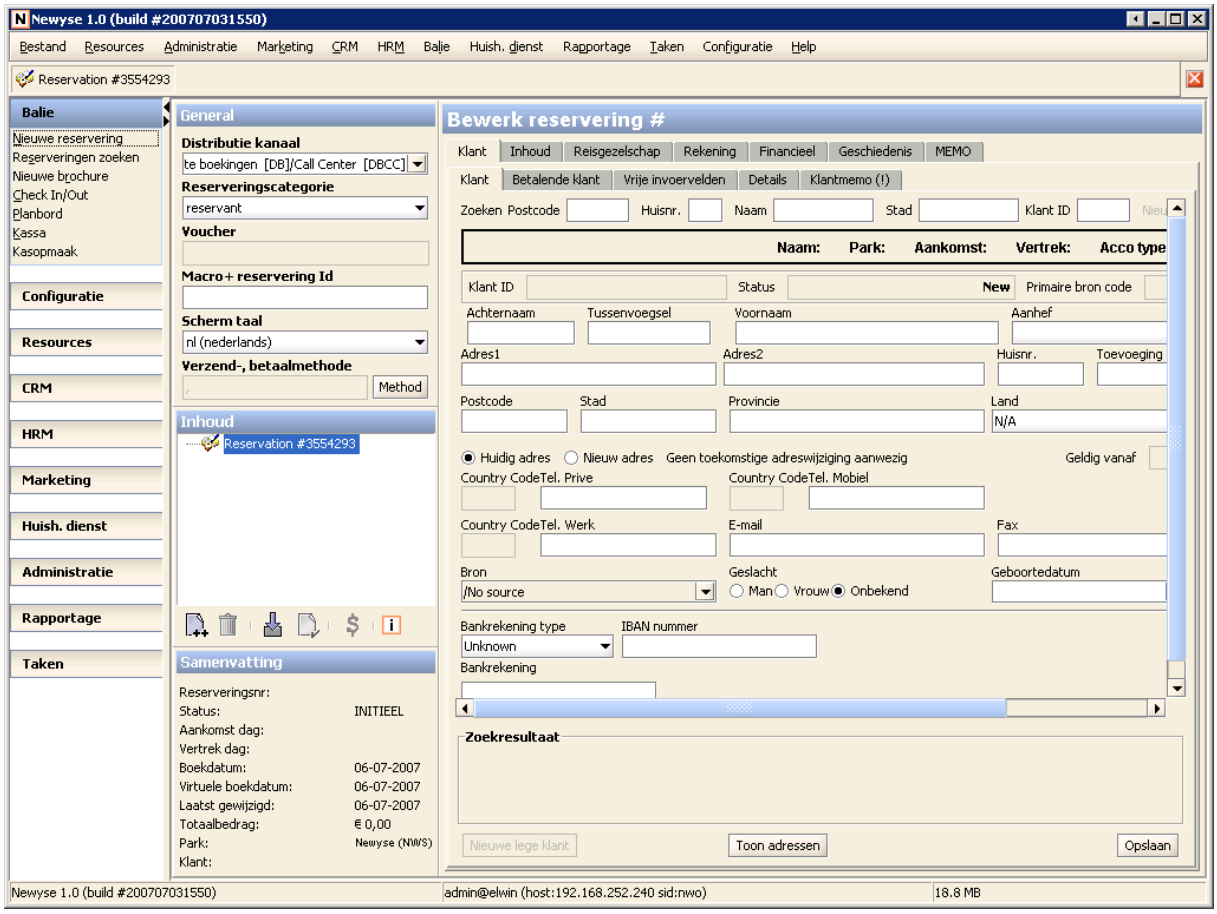
Maxxton heeft het ontwikkelproces altijd in eigen hand gehouden, hoewel wel altijd ontwikkelcapaciteit is ingehuurd. Sinds oktober 2005 werkt een team programmeurs en testers in Pune (India) aan de verdere ontwikkelingen in het pakket. Vanuit het kantoor in Middelburg wordt voornamelijk support en consultancy verleend in combinatie met onderzoek en productontwikkeling.

Naast activiteiten rond het reserveringssysteem Newyse houdt Maxxton zich ook bezig met het koppelen van externe systemen en randapparatuur. Zo levert hij ook toegangscontrolesystemen en bijbehorende software. Het product Wyseguard is daar het beste voorbeeld van. In dit verband zijn deze activiteiten echter minder relevant.

2.2. Newyse

Newyse is een pakket dat voor een recreatieondernemer in de eerste plaats als een reserveringssysteem functioneert. Het biedt de mogelijkheid reserveringen via een callcenter of een boekingswebsite te laten maken. Het is een pakket dat in principe op een centrale server draait. Alle gebruikers loggen daarop in, zodat de informatie waarover een ieder beschikt dezelfde is. Distributiekanaal-management is een functionaliteit die in het hart van het pakket ligt. Het stelt gebruikers in staat de inhoud die wordt aangeboden (accommodaties, maar ook arrangementen en producten) te variëren in prijs, representatie en boekingsmomenten, al naar gelang het kanaal via welke het wordt aangeboden.

Verder bevat het pakket onder meer mogelijkheden om huishoudelijke dienst planningen te genereren (voor schoonmaakwerkzaamheden tijdens wisselmomenten), afrekeningen te genereren voor touroperators en eigenaren en de postkamer te beheren. Het uitgangspunt van Maxxton bij verdere ontwikkelingen is dat alle bedrijfsprocessen die op enigerlei wijze aan reserveringen te relateren zijn, in het pakket geautomatiseerd moeten zijn.

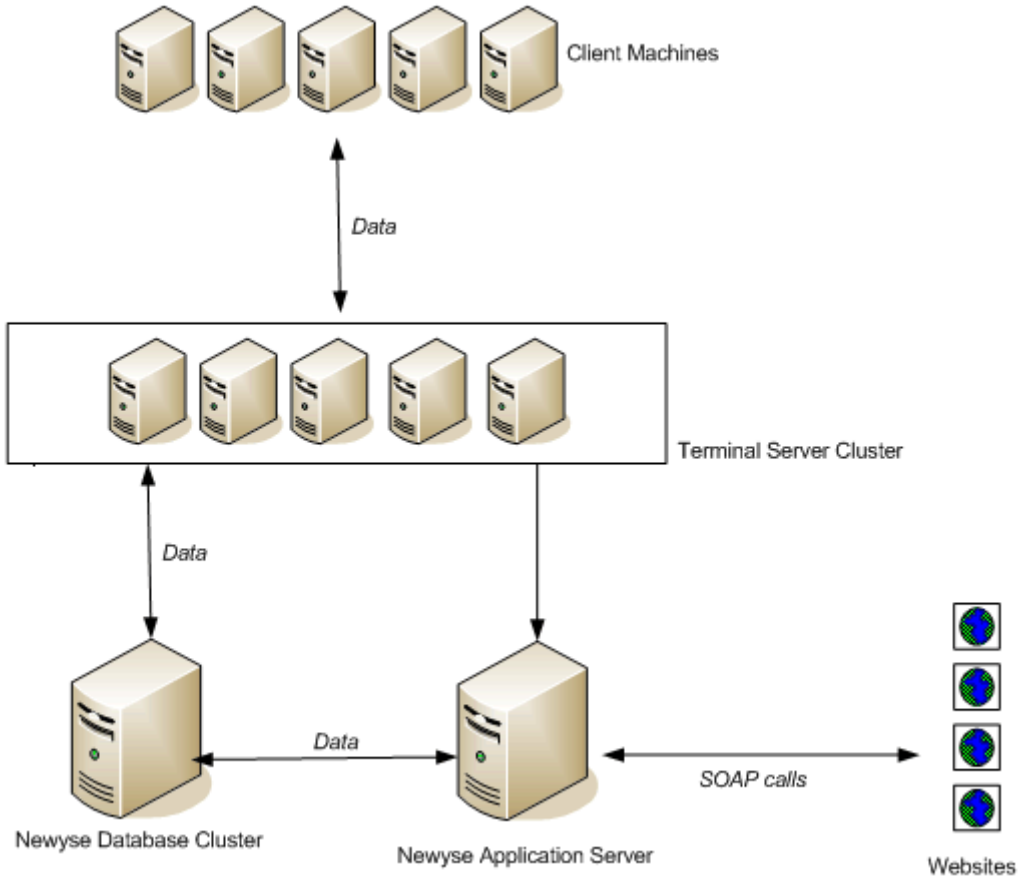


Figuur 1 : Voorbeeld van het maken van een reservering met Newyse

2.2.1. Technische omschrijving Newyse

Newyse is een client server applicatie geschreven in Java. Het onderliggende framework is het BC4J framework van Oracle. Newyse wordt dan ook ontwikkeld in JDeveloper. Newyse is opgezet volgens

het ASP [1]model. Dat wil zeggen dat de klanten zelf niet de software aanschaffen en op de eigen pc installeren, maar dat ze betalen voor het gebruik hiervan. Klanten maken gebruik van terminal servers waar zij een webstart client kunnen opstarten. Hieronder een schematische weergave van de opstelling:



Figuur 2: Server opstelling voor Newyse

De terminalservers uit het schema draaien op Windows server edities. Het database cluster bestaat uit 7 nodes van Sun servers met RHEL als besturingssysteem en de Oracle database software. Achter dit cluster draait nog een storage. Op de applicatie server draait ook RHEL en de Oracle Applicatie server.

[1] Application Service Provider

Iedere client draait zijn eigen Newyse versie op de terminalservers. Dit is geschreven in Java met JDeveloper. De client is een Swing client die werkt met 'bound components'. Deze componenten zijn door middel van een BC4J[2] framework gebonden aan de database en worden daardoor voorzien van de juiste data.

De core is een verzameling van Business Components en ondersteunende Java klassen. Veel van onze logica ligt vast in de database in '*triggers'* en '*stored procedures'*. Op de applicatie server draaien '*core*' instanties van Newyse. Dit is ontsloten naar de buitenwereld door middel van een Web Service (SOAP).

[2] Business Components for Java

2.3. Doelstelling

Voor Maxxton als organisatie is het de doelstelling om het product Newyse door te ontwikkelen en te voorzien van een gebruiksvriendelijke en met Newyse geïntegreerde website. Om dit te bereiken heeft Maxxton besloten om een integratie met een CMS te realiseren.

Het totale project is opgedeeld in drie verschillende deelprojecten, namelijk:

1) Keuze van een CMS
2) De integratie van het CMS met Newyse
3) Een pilot project

1) Mijn taak zal het dan ook zijn om het kader te definiëren waarmee er een keuze gemaakt kan worden voor een CMS, rekening houdend met de eisen vanuit de organisatie. Met behulp van dit afgebakende kader moet het dan mogelijk worden om een afgewogen beslissing te nemen over het te gebruiken CMS ten behoeve van de integratie.

2) Vervolgens zal ik verantwoordelijk zijn voor de technische projectleiding door onderzoek te doen naar mogelijk te gebruiken technieken. Het resultaat van dit onderzoek moet worden dat er een keuze gemaakt wordt uit de beschikbare technieken om de integratie van het CMS met Newyse te kunnen realiseren. Deze keuze moet te valideren zijn.

3) De opgedane kennis, het gekozen CMS en de gerealiseerde integratie met Newyse zal in de praktijk gebracht worden door middel van een pilot project. Met behulp van het pilot project wordt er een praktische invulling van de gekozen technieken ontwikkeld.

2.4. Rolverdeling binnen het project

Binnen dit project zal het mijn taak zijn om een leidende rol te nemen in het onderzoeken van de verschillende mogelijkheden, de beschikbare technieken en producten die gebruikt kunnen worden voor het succesvol voltooien van dit project.

Op procesmatig niveau zal ik advies uit brengen over de uit te voeren werkzaamheden en het opzetten van de planning hiervoor. Naast het projectmatig sturen, begeleiden en adviseren gedurende dit project zal ik zelf ook actief mee werken aan de daadwerkelijke realisatie van de integratie van het CMS.

Bij de fase CMS selectie ben ik verantwoordelijk geweest voor het inventariseren van de verschillende eisen, het maken van het overzicht van de CMS pakketten, het bepalen van de gewichten van de eisen, het maken van de vergelijkingen van de CMS pakketten en adviseren in de keuze voor het CMS. Natuurlijk heb ik dit hele proces niet alleen kunnen uitvoeren. Er is regelmatig overleg en discussie geweest met collega's en de projecteigenaar Jean-Pierre Mampaey. Met Rob Sonke heb ik veel kunnen discussiëren en ideeën uit kunnen wisselen over de keuze van het CMS.

Bij de integratie fase ben ik werkzaam geweest als technisch projectleider. In deze rol heb ik de projectleider geadviseerd op inhoudelijk niveau en voorzien van input voor de planning. Daarnaast ben ik inhoudelijk contactpersoon geweest naar de klant toe. Bij de keuze van de verschillende technieken heb ik een adviserende en sturende rol gehad waarbij er regelmatig overleg en discussie was met collega's. In deze fase werd ik niet alleen bijgestaan door Rob Sonke, maar ook door Thijs Vonk. De daadwerkelijke ontwikkeling van de integratie is in teamverband uitgevoerd, waarbij ik ook actief geweest in de rol van software ontwikkelaar.

3. Probleem stelling

3.1. Functionele beschrijving

Maxxton heeft uit strategisch oogpunt besloten om van een afstand te kijken naar het eigen pakket om te kunnen beoordelen wat er nog ontbreekt aan Newyse. Newyse is een veelzijdig, zeer flexibel en op veel punten compleet pakket. Echter een webinterface naar Newyse bestaat nog niet. Vanuit reacties van de huidige klanten blijkt dat hier wel behoefte aan is. De wenselijkheid van de webinterface uit zich op twee verschillende manieren:

- Een webgebaseerde variant van Newyse
- Websites voor Sales & Marketing, CRM[3].

De wens voor een webgebaseerde variant van Newyse ontstaat vooral vanuit kleinere parken of parken die werken met een grote spreiding van de accommodaties, waardoor er geen gebruik gemaakt wordt van grote centrale recepties. Het in het huidige ontwerp niet reëel om op elke locatie Newyse beschikbaar te maken wegens technische problemen en kosten[4]. Zodra er een webgebaseerde Newyse variant is, zou het veel toegankelijker zijn om Newyse ook op de kleinere locaties te gebruiken.

Belangrijker is de wens om van Newyse een totaaloplossing te maken voor de parken. Op dit moment biedt Maxxton met Newyse een oplossing voor de verschillende interne bedrijfsprocessen op de parken. Bijna alle aspecten op een park worden door Newyse ondervangen, maar het is nog niet mogelijk om direct een totaalpakket af te leveren inclusief een nauw met Newyse geïntegreerde website. Juist terwijl een goed geïntegreerde website ervoor kan zorgen dat een klant meer invloed, controle en mogelijkheden tot monitoren krijgt over het totale verkoopkanaal.

Deze vergrote invloed en controle kunnen ontstaan doordat de klant de beschikking krijgt over actuele (boekings)gegevens en de mogelijkheid om configuratie in Newyse direct effect te laten hebben op de website. Vanuit de markt bestaat de behoefte om snel in te kunnen spelen op actuele situaties. Door middel van een Newyse CMS (een CMS geïntegreerd met Newyse) wordt de klant in staat gesteld om ook op de website gebruik te maken van de flexibiliteit van Newyse.

> *Voorbeeld: op het moment dat blijkt dat er op een bepaald park voor het komende weekend nog veel plaatsen beschikbaar zijn, dan wil de marketingafdeling de mogelijkheid hebben om hier snel op in te spelen. Dit zou is mogelijk door een speciale aanbieding te configureren in Newyse. Deze nieuwe aanbieding moet dan ook direct (of in elk geval spoedig na aanmaken) beschikbaar zijn op internet om effect te kunnen hebben.*

[3] CRM: Customer Relationship Management
[4] De reden hiervan komt aan bod bij het technische gedeelte.

3.2. Functionele eisen

De uitdagingen van dit project zitten in het overbrengen van de flexibiliteit van Newyse op het te ontwikkelen CMS, maar met behoud van beheersbaarheid en performance. Newyse is ontworpen om een oplossing te bieden binnen de grotere organisaties binnen de recreatie sector en is zeer configureerbaar en aanpasbaar, zodat het goed aansluit bij de wensen van de klant. Daardoor zijn er weinig beperkingen en veel mogelijkheden in Newyse. Deze flexibiliteit moet behouden en toegepast worden binnen de website. Aangezien *flexibiliteit* vaak een negatieve invloed heeft op de *performance* en *beheersbaarheid* van een pakket is het noodzakelijk om tijdens het overbrengen van de flexibiliteit van Newyse op het CMS terdege rekening te houden met de *performance* en *beheersbaarheid* van het CMS zodat alle drie de aspecten gegarandeerd kunnen worden.

Deze aspecten zijn zowel van toepassing op het tonen van informatie (parkinformatie, accommodatie-informatie) op de website als op het maken van nieuwe reserveringen.

Maxxton heeft besloten dat er een nieuwe methode van reserveren voor op de website ontwikkeld moet worden. De nieuwe methode moet gebruik gaan maken van zoeken op basis van kenmerken van de verschillende accommodaties, zodat de gast snel kan bepalen welke accommodatie er geschikt is. De complexiteit hiervan is hoog, vooral als gevolg van de flexibiliteit zoals Newyse deze biedt. Aangezien het de gast is die gebruik gaat maken van dit systeem en niet een getrainde en geïnstrueerde medewerker, zullen het gebruiksgemak en de performance een cruciale rol spelen.

De websites moeten een communicatiemiddel vormen voor de parken richting de uiteindelijke klanten, ondersteund door de beschikbare informatie uit Newyse.

Kortom de belangrijkste functionele eisen aan het CMS zijn:
flexibiliteit van Newyse handhaven, goede performance en beheersbaarheid.

3.3. Technische beschrijving

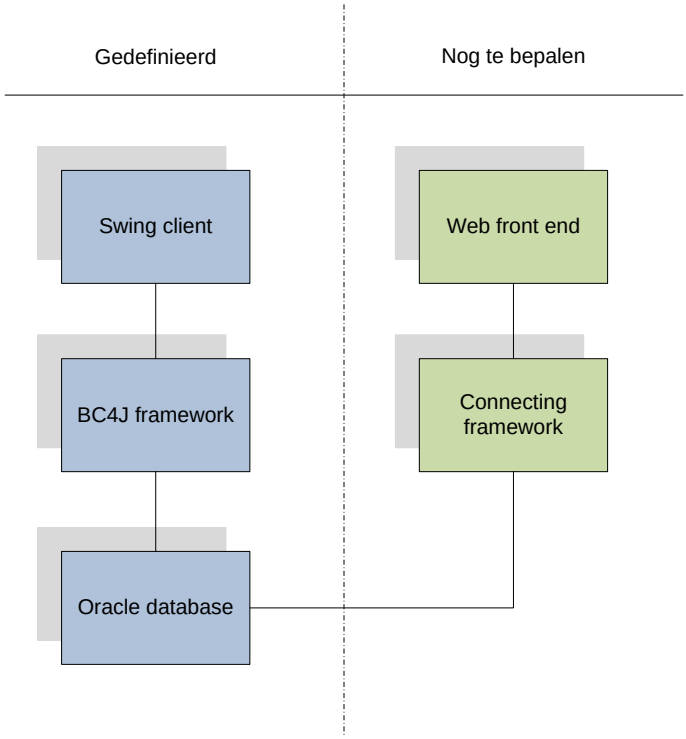
Op dit moment is het gebruik van Newyse geregeld via terminal servers, waarbij de gebruiker direct inlogt op de terminal server en daar via een webstart[5] de Newyse applicatie opstart.

Dit gebeurt onder meer vanwege het beperken van het dataverkeer richting de gebruiker en het beperken van het geheugen gebruik.

Door een integratie van het CMS met Newyse zal het mogelijk zijn om efficiënter gegevens uit te wisselen.

In de huidige opzet van Newyse wordt er gebruik gemaakt van een Java Swing client applicatie die communiceert met de (Oracle) database door middel van het Oracle BC4J EJB Framework. Voor de integratie met een CMS zal er gewerkt moeten worden met een ander alternatief framework. Tevens zal er ook voor de interface van de website en het CMS voor een techniek gekozen moeten worden.

[5] Met webstart wordt een link op een internet site bedoeld waarmee een Java programma opgestart kan worden.



Figuur 3 : Schematische weergave van gebruikte technieken en de onderlinge verbanden.

In Figuur 3 wordt op een schematische wijze getoond welke technieken er gebruikt worden en wat er nog ingevuld moet worden. Voor de integratie zal het noodzakelijk zijn om te bepalen welke technieken er gebruikt gaan worden.

Eén van de aspecten van de realisatie van de CMS integratie met Newyse is dat deze integratie het mogelijk moet gaan maken om op internet te werken met een nieuwe manier van zoeken. Deze nieuwe manier van zoeken houdt in dat er op een zeer flexibele manier op verschillende kenmerken op verschillende niveaus gezocht kan worden[6].

De performance van het zoeken moet de hoogste prioriteit hebben. Om dit te kunnen realiseren moet er onderzocht gaan worden welke technieken hiervoor beschikbaar zijn en welke techniek daarvan het meest geschikt is voor dit doel.

[6] Meer details over de werkwijze van het zoeken komen later aan bod

3.4. Probleemstelling 1: Newyse CMS

Wat is de beste aanpak voor de keuze van een CMS (dat gebruikt gaat worden met Newyse)?

Aan welke criteria moet het CMS gaan voldoen?

Welk CMS is het meest geschikte CMS voor deze integratie?

3.5. Probleemstelling 2: Integratie Newyse CMS en Newyse

Hoe moet de integratie van CMS en Newyse technisch gezien gerealiseerd worden?

Met als deelvragen:

- *wat is 'de huidige situatie'?*
- *wat zijn de functionele requirements?*
- *waar zitten de technische knelpunten?*
- *wat zijn mogelijke technische oplossingen?*
- *welke oplossing kunnen we hier het beste kiezen?*

3.6. Probleemstelling 3: Webgebaseerde Newyse variant

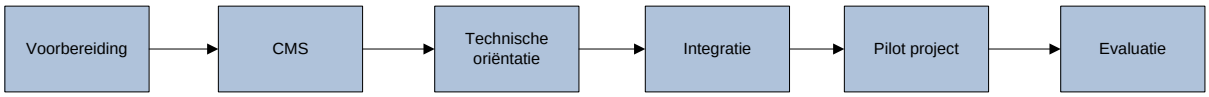
Wat is de beste aanpak om een webgebaseerde variant van Newyse te ontwikkelen?

Welke technieken zijn er beschikbaar en welke is hiervoor het meest geschikt?

Om de omvang van het project te kunnen beperken en binnen bepaalde kaders te kunnen houden is ervoor gekozen om Probleemstelling 3 buiten dit project te laten vallen.

4. Plan van aanpak

Voor de realisatie van dit project is het project opgedeeld in zes verschillende fases. Voor elk van de verschillende fases wordt er beschreven hoe de fase aangepakt zal worden.



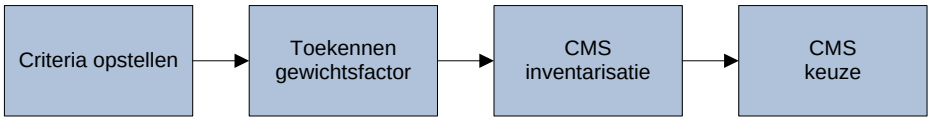
Figuur 4 : Fases binnen het project

4.1. Fase I: Voorbereiding

Voorafgaand aan het project is één van de overwegingen geweest of er eventuele geschikte partners waren om het CMS door te laten ontwikkelen of dat alles intern ontwikkeld zou gaan worden. Het gaat hier om een strategische beslissing die al genomen is voor aanvang van het project. Hierbij is besloten om het zelf in eigen hand te nemen.

De voorbereidingsfase heeft gezorgd voor de initiële ideeën voor dit project. Tijdens deze fase is er vooral uit strategisch oogpunt gekeken naar de strategische wenselijkheid van het project en de mogelijke alternatieven.

4.2. Fase II: CMS



Figuur 5 : Workflow CMS selectie

Het selecteren van het juiste CMS is opgedeeld in vier verschillende stappen.
Ten eerste wordt er begonnen met het opstellen van een grote criteria lijst die allemaal (in meer of mindere mate) van belang zijn voor de keuze van het CMS.

In de tweede stap zal aan deze criteria een gewicht toegekend worden. Dit gewicht drukt de mate van belang uit van een criterium voor het slagen van de CMS integratie. Dit om een relevante vergelijking te kunnen realiseren.

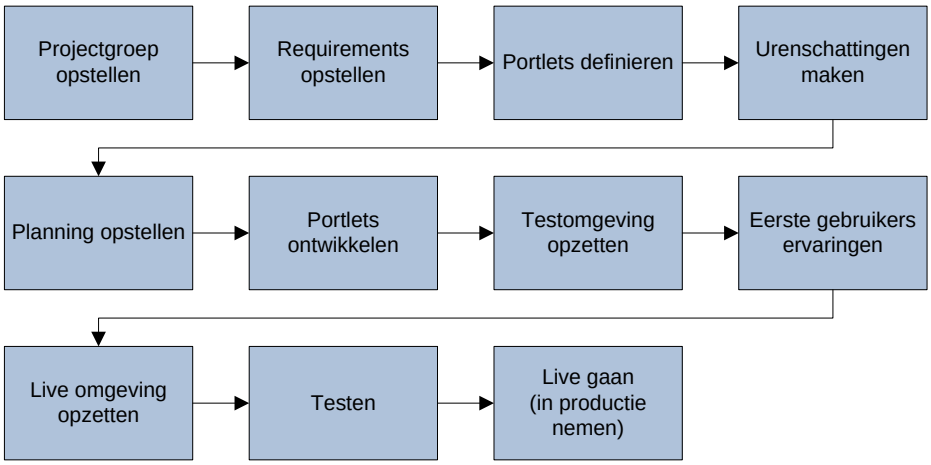
4.3. Fase III: Technische oriëntatie

Inventarisatie van de verschillende beschikbare technieken die bruikbaar zijn voor de realisatie van dit project en het opdoen van kennis hierover. Hiervoor wordt gebruik gemaakt van bestaande literatuur en referenties.

4.4. Fase IV: Integratie

Het realiseren van een integratie tussen het CMS en Newyse. Voor het realiseren van deze integratie moet er gezocht worden naar de juiste technieken op het gebied van een web application framework (de techniek waarmee de user-interface wordt ontwikkeld) en een connecting framework, dat ervoor moet zorgen dat de communicatie tussen het web application framework en de database geregeld wordt.

4.5. Fase V: Pilot project



Figuur 6: Stappenplan realisatie in pilot project

In fase V wordt er een praktische realisatie van het CMS en de ontwikkelde integratie tussen het CMS en Newyse ontwikkeld. Deze praktische realisatie zal direct een concreet project worden voor één van onze klanten.

De uitvoering van dit pilot project is geen statisch project maar eerder een dynamisch proces. Gedurende looptijd van het pilot project wordt er regelmatig kritisch gekeken naar de juiste benadering van het project. Het gevolg hiervan is dat de details van het uiteindelijke doel steeds aan verandering onderhevig zijn.

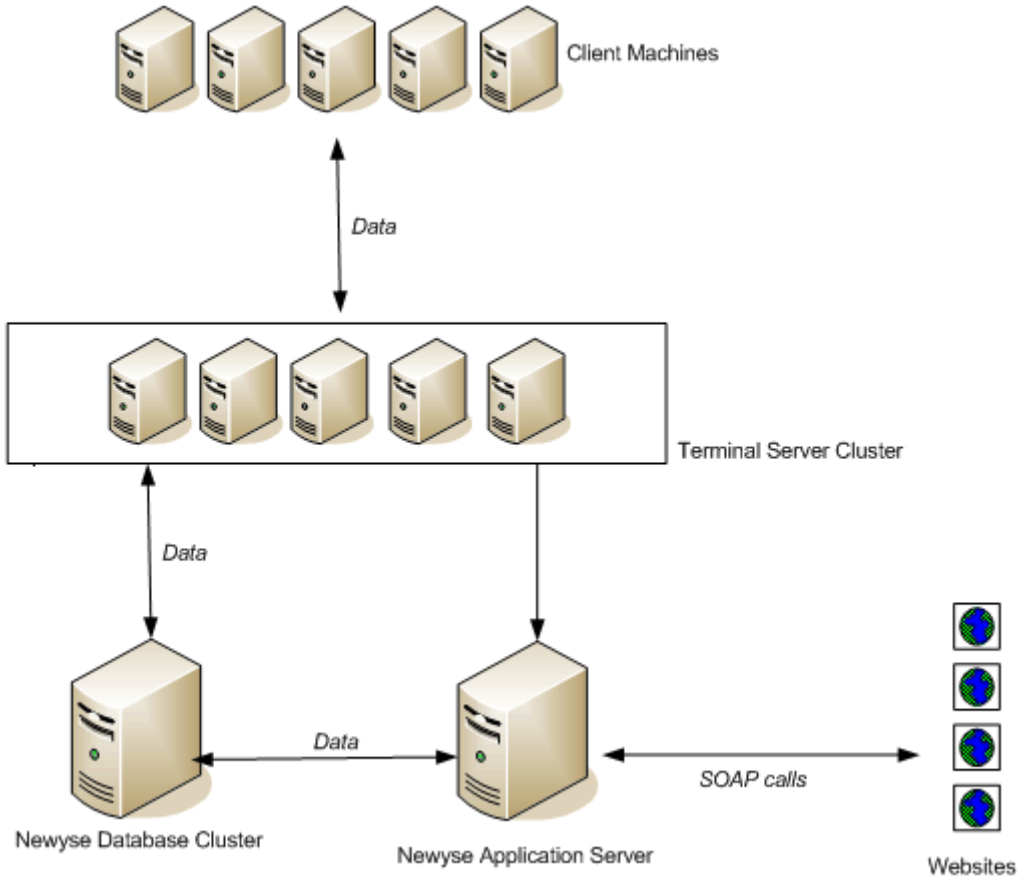
4.6. Fase VI: Evaluatie

Evaluatie van het bereikte resultaat. Zijn de probleemstellingen beantwoord?

Is het pilot project een succes geworden?

5. Newyse CMS

Bij de introductie van Newyse is er al een kort overzicht gegeven van de huidige structuur. Hieronder volgt nogmaals dit schema.

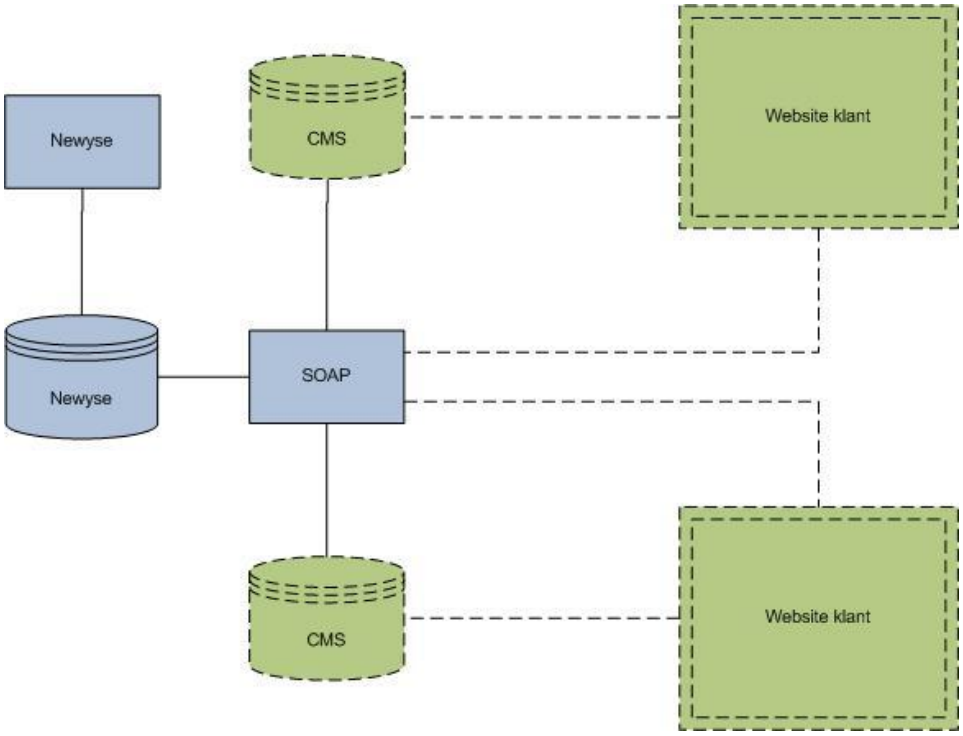


Figuur 7 : Newyse structuur

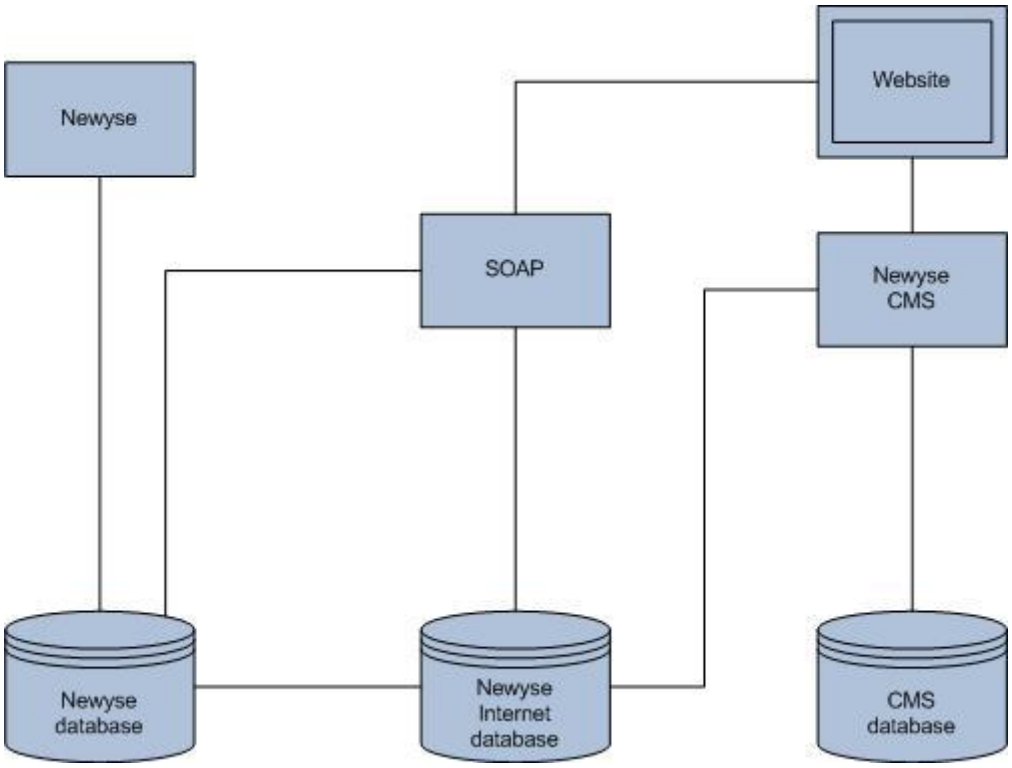
In Figuur 7 wordt schematisch de opstelling getoond van de gebruikte servers en apparatuur voor Newyse. Op de Newyse applicatie server draait een Newyse versie met een webservice (een SOAP service). De verschillende websites maken gebruik van deze webservice. Deze websites worden gemaakt door andere partijen. Zoals ook bij de doelstelling is aangegeven zou Newyse met meer dan alleen een webservice verbonden moeten worden met het internet.

Om dat te kunnen realiseren is het noodzakelijk om hiervoor een Newyse CMS aan te bieden. Het voordeel hiervan moet zijn dat het mogelijk is om het CMS verder uit te breiden en daarmee een zeer nauwe integratie met Newyse kunnen bereiken.

Het Newyse CMS moet direct de Newyse database kunnen aanspreken of eventueel een tussenliggende Newyse Internet Database. Dit wordt hieronder schematisch getoond.



Figuur 8 : Begin situatie



Figuur 9 : Schematische weergave van het beoogde eindresultaat (in beperkte vorm)

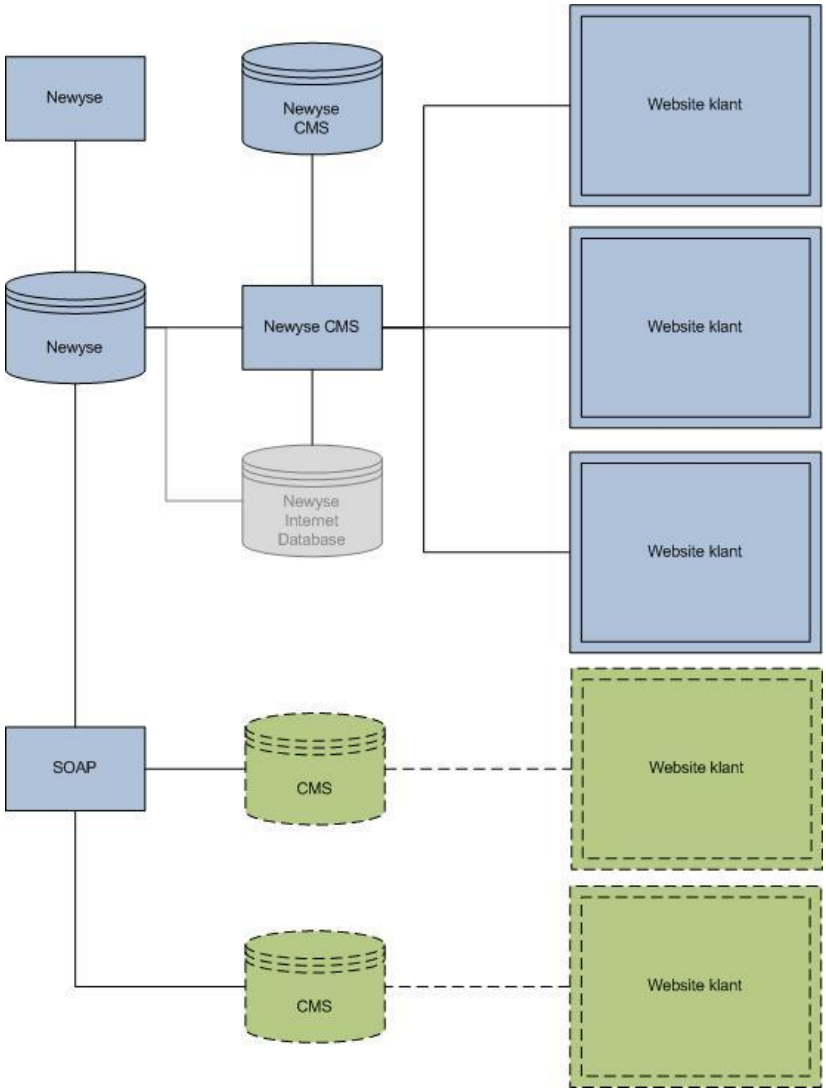
Figuur 8 geeft een schematische weergave aan van de structuur zoals deze voor aanvang van het project was. De onderbroken lijnen (en groene kleur) geven (deel)systemen weer die niet onder het beheer van Maxxton vallen. De blauwe elementen zijn in beheer van Maxxton. Opvallend hieraan is dat Maxxton wel de benodigde informatie aanlevert aan de verschillende CMS systemen van de klanten, maar dat het nog niet de mogelijkheid heeft om dit ook in zijn geheel aan te bieden aan de klanten.

Figuur 9 geeft is een schematisch weergave van het beoogde eindresultaat zoals wij dit bedacht hadden bij aanvang van dit project (zomer 2007). Kenmerk hiervan is dat alle blokken in deze situatie onder het beheer en verantwoordelijkheid van Maxxton vallen.

In Figuur 9 is tevens een Newyse internet database te zien. De huidige Newyse database is zoveel mogelijk genormaliseerd en geoptimaliseerd voor het gebruik met Newyse en een correcte dataopslag. Deze database is niet geoptimaliseerd om snel in te kunnen zoeken. Om binnen het CMS snel bepaalde zoekacties uit te kunnen voeren is het waarschijnlijk noodzakelijk om een aparte database te maken geoptimaliseerd is voor gebruik met internet zoeken. De eerste gedachte van deze database is dat hierin in elk geval berekende prijzen staan i.p.v. prijsregels die de prijs definiëren.

Om tot deze situatie te kunnen komen is het noodzakelijk dat de eisen voor een CMS duidelijk worden en er een CMS gekozen wordt dat aan deze eisen voldoet, zodat er een integratie van het CMS met Newyse gerealiseerd kan worden.

In Figuur 10 wordt de mogelijke structuur van de eindsituatie getoond. Daarin valt te zien dat de ondersteuning voor websites van externe partijen via de SOAP aanwezig blijft. Maar er ontstaat ook een situatie waarbij het beheer in het geheel bij Maxxton komt te liggen.



Figuur 10 : Mogelijke eindsituatie

Aangezien er al vele CMS systemen beschikbaar zijn is er besloten om eerst te gaan kijken of er al een geschikt CMS bestaat dat gebruikt kan worden en eventueel uitgebreid.

5.1. Aanpak CMS keuze

De basis voor het maken van de juiste keuze voor een CMS wordt gevormd door een goede aanpak.

Er is uitgegaan van de kracht van de organisatie, dit is vertaald naar eisen waaraan de geschikte CMS systemen moeten voldoen. Dit zorgt direct voor een afbakening van het CMS terrein. Op basis van een aantal van deze eisen vallen er direct een aantal CMS systemen af.

De belangrijkste eis vanuit de organisatie is het gebruik van een CMS dat is ontwikkeld in de programmeertaal Java. Hiervoor zijn enkele redenen van toepassing. De eerste reden is dat binnen de organisatie er veel kennis en ervaring bestaat met Java, dit zorgt voor een snellere ontwikkeling en een makkelijke inzetbaarheid van personeel tussen de verschillende projecten.

Daarnaast biedt een CMS dat ook in Java is ontwikkeld goede integratie mogelijkheden, het kan eenvoudig en snel aan elkaar gekoppeld worden.

Vervolgens is er gekozen om een overzicht van (overige) CMS kenmerken op te stellen en hieraan een gewicht te koppelen. Dit gewicht representeert dan de mate van belang van het kenmerk in de keuze van een CMS.

De volgende stap is het opstellen van een verzameling van CMS systemen die in elk geval voldoen aan de eerste beperkingen. Voor elk van deze CMS systemen is een overzicht gemaakt van de verschillende kenmerken om te kijken aan welke deze voldoen.

Door dit overzicht te combineren met de lijst met kenmerken inclusief de verschillende waarderingen is het mogelijk om een overweging te maken welk CMS het meest in aanmerking komt.

5.2. Beperkingen & Eisen

Om tot de keuze van een juist CMS te kunnen komen is het van belang om goed vast te stellen wat precies de beperkingen en eisen zijn voor het te gebruiken CMS.

Als uitgangspunt voor het opstellen van de eisen is voor de basis uitgegaan van de lijst zoekcriteria zoals deze gebruikt worden op de website: www.cmsmatrix.org (CMS Matrix, 2007).

Op deze website is het mogelijk om vele bestaande CMS systemen met elkaar te vergelijken om te bepalen of hier een geschikt systeem tussen zit.

De website is een goed startpunt voor het opstellen van een lijst met eisen aangezien hier al direct een hele grote groep met verschillende eisen gegeven wordt. Dit kan dan als basis gebruikt worden voor de eigen lijst met eisen, zolang er deze lijst maar wel continue kritisch bekeken wordt.

Het verkregen overzicht van eisen mag niet als compleet gezien worden. Voor een verantwoorde keuze voor een CMS moet dit overzicht nog wel aangevuld worden met de specifieke wensen voor het project. Het aanvullen van deze lijst is ontstaan door binnen het projectteam te discussiëren over de gewenste functionaliteiten van het te kiezen CMS.

Het resultaat van het verwerken van de verschillende eisen is dat er een opsomming ontstaan is van verschillende criteria voor het CMS. Hieronder zal er een overzicht getoond worden van de verschillende eisen en de toepassing daarvan op het Newyse CMS. Om een goede vergelijking en afweging te maken is er aan elk kenmerk een gewicht gehangen. Dit gewicht per kenmerk is specifiek

voor het huidige project. Met behulp van dit gewicht worden de eisen op prioriteit gesorteerd. De mogelijke gewichten zijn (de prioriteiten worden later gebruikt in de CMS matrix):

- **Noodzakelijk (prioriteit 1)**
  Dit kenmerk is van zodanig groot belang dat het onmogelijk is om een CMS te kiezen dat niet voldoet aan een kenmerk van dit type.
- **Zeer gewenst (prioriteit 2)**
  Dit kenmerk is van groot belang. Het ontbreken van één kenmerk met dit gewicht bij een CMS zal niet perse betekenen dat het uitgesloten wordt, maar is wel een zwaarwegend nadeel.
- **Gewenst (prioriteit 3)**
  Het zou fijn zijn als een CMS hierin kan voorzien, dat zal de doorslag kunnen geven ten opzichte van een ander CMS.
- **Neutraal (prioriteit 4)**
  Dit kenmerk heeft geen invloed op de keuze van het CMS.
- **Ongewenst (prioriteit 5)**
  Dit kenmerk zorgt voor minpunten in een onderlinge vergelijking tussen CMS systemen als het wel aanwezig is.
- **Zeer ongewenst (prioriteit 6)**
  Als een CMS voldoet aan dit kenmerk, dan valt het direct af voor de selectie.

Kenmerk	Gewicht	Prioriteit
Friendly URLs	noodzakelijk	1
Interface Localization	noodzakelijk	1
Licentie	noodzakelijk	1
Multi-lingual Content	noodzakelijk	1
Pluggable API	noodzakelijk	1
Programmeertaal	noodzakelijk	1
Server Page Language	noodzakelijk	1
SSL Compatible	noodzakelijk	1
Template Language	noodzakelijk	1
Themes / Skins	noodzakelijk	1
WYSIWYG Editor	noodzakelijk	1
Commercial Support	zeer gewenst	2
Developer Community	zeer gewenst	2
Inlog geschiedenis	zeer gewenst	2
Multi-Site Deployment	zeer gewenst	2
Online Administration	zeer gewenst	2
Online Help	zeer gewenst	2
Advanced Caching	gewenst	3
Captcha	gewenst	3
Commerciele handleidingen	gewenst	3
Content Scheduling	gewenst	3
Database	gewenst	3

Drag-N-Drop Content	gewenst	3
Email verificatie	gewenst	3
Image Resizing	gewenst	3
Kosten	gewenst	3
Load Balancing	gewenst	3
Uploaden meerder bestanden	gewenst	3
Metadata	gewenst	3
Multi-lingual Content Integration	gewenst	3
Notificatie bij problemen	gewenst	3
Openbaar forum	gewenst	3
Openbare mailinglist	gewenst	3
Open source	gewenst	3
Package Deployment	gewenst	3
Page Caching	gewenst	3
Pluggable Authentication	gewenst	3
Sessie management	gewenst	3
Site Map	gewenst	3
SSL Logins	gewenst	3
SSL Pages	gewenst	3
Exporteren van statische content	gewenst	3
Statistieken	gewenst	3
Sub-sites / Roots	gewenst	3
URL Rewriting	gewenst	3
Web-based Style/Template Management	gewenst	3
Web-based Translation Management	gewenst	3
Audit Trail	neutraal	4
FAQ Management	neutraal	4
Inline Administration	neutraal	4
Search Engine	neutraal	4
Type applicatie server	neutraal	4

Een overzicht van de verschillende kenmerken gesorteerd op prioriteit. Het gewicht zegt in deze tabel niets over de voorkeur van het betreffende kenmerk, maar alleen wat de invloed van het kenmerk op de keuze van het CMS is. Als voorbeeld kan er gekeken worden naar het kenmerk ‘Kosten’. Als gewicht hangt hier aan ‘Gewenst’. Dit betekent niet dat het gewenst is dat er kosten aan het CMS zitten, maar dat de kosten van een CMS niet de hoogste prioriteit hebben bij het maken van de keuze.

Systeemeisen

- **Applicatie server** *geen eisen* *neutraal*
  Ondanks dat op dit moment gebruik gemaakt wordt van de Oracle applicatie server voor de Newyse applicatie is dit geen beperking voor de keuze van een applicatie server.
- **Kosten** *€ 0,00 / gratis* *gewenst*
  Gezien het feit dat er voldoende keus is in de markt van de verschillende CMS systemen is het niet noodzakelijk om veel kosten te maken aan een CMS. Zeker niet als bedacht wordt dat een betaald systeem vaak snel in de tienduizenden euro's kost en dat het CMS waarschijnlijk geen zeer complex systeem moet worden. Dit is echter zeker geen harde eis en als blijkt dat een CMS wel degelijk geld kost, maar dat uit de andere kenmerken blijkt dat het wel degelijk (aanzienlijk) beter geschikt is, dan gaat dat CMS wel voor.
- **Database** voorkeur: *mysql* *gewenst*
  Ondanks dat op dit moment er een Oracle database in gebruik is voor de Newyse applicatie is dit zeker geen eis voor het CMS. Er zal namelijk niet zeer veel data en intelligentie in het CMS terecht komen, waardoor de Oracle database niet die meerwaarde kan bieden om de hoge licentie kosten van een Oracle database te compenseren.
  Een mysql database moet dan ook zeker kunnen voldoen en heeft niet de nadelen van de hoge licentiekosten. Overigens is het niet strict noodzakelijk om een mysql-database te gebruiken, een eventuele andere variant hiervan zou ook kunnen voldoen (Oracle, Postgres, mSQL, enz)
- **CMS licentie** *vermarktbaar* *noodzakelijk*
  De licentie waarmee het CMS wordt uitgegeven is van zeer groot belang in de keuze van het CMS. De licentie moet het namelijk toestaan om het CMS naar wens aan te passen en om deze evt. aangepaste versie onder de eigen naam in de markt te kunnen zetten. Indien dit niet het geval is, is het niet mogelijk om zomaar zonder problemen een CMS aan klanten uit te leveren.
- **Open source** voorkeur: *ja* *gewenst*
  Van een CMS kan de broncode openbaar zijn of niet. Ook dit kan in de keuze voor een CMS meegenomen worden. Dit kenmerk ligt dicht tegen de kosten en/of licentie aan, maar mag zeker niet samengenomen worden daarmee. Het 'open source' zijn van een CMS kan bruikbaar zijn doordat eventuele gebreken of andere interpretaties van functionaliteiten zelf bekeken en aangepast kunnen worden. Er kunnen echter nog steeds beperkingen in de licentie zitten waardoor dit niet toegestaan is.
  Vaak wordt 'open source' ook gezien als synoniem met gratis of zonder kosten. Dit is niet waar. Het licentie model kan nog steeds kosten aan het gebruik afdwingen. Een andere mogelijkheid is dat de ontwikkelkosten voor integratie zeer hoog worden als gevolg van gebrek aan support en documentatie.
- **Programmeertaal** voorkeur: *Java* *noodzakelijk*
  Bij voorkeur wordt er gekozen om een CMS te gebruiken dat gemaakt is in de taal Java. Maxxton heeft hiermee voldoende kennis in huis en tot de beschikking bij Cybage om zonder eerst een andere techniek aan te leren aanpassingen door te voeren in het bestaande systeem. Daarnaast maakt dit de communicatie tussen het CMS en Newyse eenvoudiger doordat ze beide gebruik maken van dezelfde techniek.

Beveiliging

- **Captcha** *gewenst*
  Beveiliging tegen computersystemen die misbruik trachten te maken van invulformulieren. Meestal wordt dit gedaan d.m.v. een tekst in een afbeelding, die de gebruiker moet overtypen.

- **Email verificatie** *gewenst*
- **Inlog geschiedenis** zeer *gewenst*
- **Pluggable authentication** *gewenst*
  Biedt het CMS ondersteuning voor een externe manier van authentificatie? Dit zal vooral een rol gaan spelen bij persoonlijke omgevingen voor de gasten en op het moment dat gebruikers van Newyse automatisch gekoppeld moeten worden aan het CMS.
- **Notificatie bij problemen** *gewenst*
  Als het CMS bij meerdere verschillende klanten de productie omgeving gaat regelen is het noodzakelijk dat het CMS voorzien is van een mogelijkheid om zelf problemen vast te stellen en deze ook te kunnen melden (bijvoorbeeld dmv. een sms of email).
- **Ondersteuning voor het SSL protocol** noodzakelijk
  Het CMS moet de mogelijkheid bieden om beveiligde verbindingen op te zetten of om dit zeer makkelijk te kunnen implementeren. Dit is vooral zeer van belang bij betalen via internet, maar kan ook gebruikt worden voor het inloggen door gasten op een persoonlijke omgeving.

Support

- **Professionele support beschikbaar** zeer gewenst
  Een CMS met een professionele support afdeling kan in veel gevallen zorgen voor snelle en gerichte ondersteuning in het geval van problemen. Dit kan vaak meer opleveren (door het voorkomen of oplossen van problemen) dan dat het kost.
- **Developer community** *zeer gewenst*
- **Online help** *zeer gewenst*
- **Pluggable API** *noodzakelijk*
  Beschikt het CMS over een mogelijkheid om het CMS zelf uit te breiden met eigen ‘modules’ en deze te koppelen d.m.v. een API (een gedefinieerde interface).
- **Openbaar forum** *gewenst*
  Door middel van een openbaar forum is het mogelijk om in contact te komen met andere gebruikers en elkaar te helpen bij het oplossen van problemen.
- **Openbare mailinglist** *gewenst*
  Een alternatief voor het forum is een mailinglist van gebruikers (en ontwikkelaars) voor het bespreken van problemen.

Gebruiksgemak

- **Drag-N-Drop** *gewenst*
  Het zou handig zijn als de gebruiker d.m.v. ‘drag-and-drop’ de indeling van een pagina zou kunnen wijzigen.
- **Friendly URLs** *noodzakelijk*
  Een friendly URL betekent dat elke willekeurige pagina bereikbaar moet kunnen zijn onder een leesbaar adres. Bijvoorbeeld: www.voorbeeld.nl/klant/21 i.p.v. www.voorbeeld.nl/index.html?klant=21. Dit is ook handig voor de bezoekers, zodat deze de pagina adressen kan onthouden en eenvoudig terug kan komen, maar het is vooral belangrijk voor zoekmachines. Zoekmachines hebben namelijk deze friendly URLs nodig om de pagina’s goed te kunnen indexeren.
- **Automatisch afbeeldingsformaat wijzigen** *gewenst*
- **Uploaden van meerdere bestanden tegelijk** *gewenst*
- **Server page language** *noodzakelijk*
- **Template language** *noodzakelijk*

- **WYSIWYG editor** *noodzakelijk*
  "What You See Is What You Get" editor, dat houdt in dat een gebruiker van het CMS makkelijk de opmaak van teksten kan beïnvloeden en direct daarvan het resultaat ziet. Hierbij kan gedacht worden aan de manier waarop een tekst programma werkt.

Performance

- **Advanced caching** *gewenst*
  Door middel van een efficiënt gebruik van caching wordt de performance verbeterd. Veel uitgevoerde queries op de database kunnen gebruik maken van een cache om de database te ontlasten.
- **Load balancing** *gewenst*
  Load balancing kan handig zijn, maar het is mogelijk een nog betere oplossing om dit door de applicatie server te laten afhandelen en niet door het CMS.
  Een ander alternatief is om een hardware matige oplossing in te zetten, dit is de beste oplossing qua performance, maar hieraan zitten ook hoge kosten vast.
- ***Pagina caching*** *gewenst*
- ***Static Content Export*** *gewenst*

Management

- **Content scheduling** *gewenst*
- **Online administration** *zeer gewenst*
- **Ondersteuning voor goede logging en statistieken** *gewenst*
  Het moet in het CMS mogelijk zijn om het gebruik van de website op een uitgebreide manier te volgen. Het gaat dan natuurlijk om de fouten die optreden, maar ook het normale gebruik moet te volgen zijn. Onder andere de manier waarop een klant door de website loopt en hoe het op een bepaalde plek uitkomt. Of bijvoorbeeld hoe een reservering geboekt wordt, of juist afgebroken.
  Dit zal zeer waarschijnlijk een zelf te maken onderdeel moeten worden, aangezien de wensen te specifiek zijn. Het zou natuurlijk wel handig zijn als een aantal basisfunctionaliteiten al in het CMS naar voren komen.
- **Package deployment** *gewenst*
- **Sub-sites / Roots** *gewenst*
- **Ondersteuning voor eenvoudig gebruik van themes** *noodzakelijk*
  Het moet eenvoudig mogelijk zijn om met het CMS het uiterlijk van een website aan te passen. Hiervoor wordt gebruik gemaakt van themes. De themes zelf definiëren in dit geval hoe de website er uit komt te zien. Deze zal niet met behulp van het CMS gemaakt moeten worden, maar het moet wel mogelijk zijn om eenvoudig een theme toe te passen.
  Dit valt grotendeels ook wel onder een aantal criteria, die genoemd worden, maar is zodanig van belang dat het extra onder de aandacht gebracht moet worden
- **Web-based style / Template management** *gewenst*
- **Web-based translation management** *gewenst*

Flexibiliteit

- **Interface localization** *zeer gewenst*

Aangezien het de bedoeling is dat het CMS ingezet gaat worden bij meerdere klanten, waarbij het ook mogelijk is dat het in het buitenland wordt ingezet, zou het zeer handig zijn als het CMS direct ondersteuning biedt om de interface in verschillende talen aan te bieden.

- **Metadata** *gewenst*
- **Multi-lingual content** *noodzakelijk*

  In de recreatie branch is het niet onwaarschijnlijk dat er buitenlandse gasten zijn die een reservering willen maken. Onze klanten verwachten dan ook zeker de mogelijkheid om een website in meerdere talen te kunnen aanbieden, zodat ook buitenlandse bezoekers de website kunnen lezen.
- **Multi-site deployment** *zeer gewenst*
- **URL rewriting** *gewenst*

5.3. CMS vergelijking

Door middel van zoeken en vergelijken van CMS systemen op basis van de CMS systemen zoals deze bekend zijn bij *cmsmatrix.org* (CMS Matrix, 2007) en de vastgestelde eisen zoals die eerder zijn bepaald blijven er een aantal CMS systemen over die interessant zijn om te bespreken.

De verschillende kenmerken zijn samen met de prioriteit ervan in een matrix gezet om een snel overzicht te creëren wat de belangrijkste plus- en minpunten zijn van de verschillende CMS systemen.

				Jahia ECM			Magnolia 3.0			
Product		GX WebManager 8.3	InfoGlue 2.0	Community	Professional	Liferay Portal/CMS	Community	Enterprise	MMBase 1.6	OpenCms 6.2.3
Kenmerk	**Prioriteit**	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
Applicatie server	4	any J2EE 1.4 compatible AS	Tested in Tomcat, Resin and Websphere	J2EE		Tested compatibility with Borland ES 6.5, JBoss 4.0.2, JOnAS 4.4.3, JRun 4 Updater 3, OracleAS 10.1.2, Orion 2.0.6, Pramati 4.1, RexIP 2.5, Sun JSAS 8.01, WebLogic 8.1 SP4, WebSphere 5.1	J2EE	J2EE		Tomcat, JBoss, Bea Weblogic.
Verwachte kosten	3	Niet vermeld op de site, maar niet gratis en afhankelijk van de verschillende modules die gebruikt worden.	€ 0,00	€ 0,00	€ 19990,00 per JVM	€ 0,00	€ 0,00	€ 7990,00 per server per jaar	€ 0,00	€ 0,00
Database	3	MySQL, Oracle, MSSQL	Oracle, Microsoft SQL Server or MySQL, DB2, Sybase iAnywhere	Postgres, Oracle, MySQL, MS SQL server, proprietary		DB2, Firebird, Hypersonic, InterBase, JDataStore, MySQL, Oracle, PSQL	Java Content Repository (JCR)	Java Content Repository (JCR)	Any	MySQL, PostGreSQL, Oracle, MSSQL
Licentie	1	Flexible	GNU GPL	JCDDL	JSSL	MIT Open Source Neen-restrictive business friendly license	GNU LGPL	GNU LGPL	Mozilla Public License	GNU LGPL
Ontwikkeltaal	1	Java(J2EE)/XML/XSLT/Java script/JSP	Java	Java 1.4. +		Java 1.4. +	Java	Java	Java 1.3+	Java 1.3+

Beveiliging	**Prioriteit**	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
Audit Trail	4	Ja	Ja	Ja		Ja	Ja	Ja	Beperkt	Ja
Captcha	3	Nee	Nee	Ja		Ja	Nee	Nee	Nee	Nee
Email Verification	3	Ja	Nee	Nee		Gratis toevoeging	Ja	Ja	Nee	Nee
Login History	2	Ja	Nee	Ja		Ja	Ja	Ja	Beperkt	Ja
Pluggable Authentication	3	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Extra kosten
Problem Notification	3	Ja	Nee	Ja		Nee	Ja	Ja		Ja
Session Management	3	Nee	Nee	Gratis toevoeging		Ja	Ja	Ja	Ja	Nee
SSL Compatible	1	Ja	Ja	Ja		Ja	Ja	Ja	Nee	Ja
SSL Logins	3	Ja	Ja	Ja		Ja	Ja	Ja	Nee	Ja
SSL Pages	3	Ja	Nee	Ja		Ja	Ja	Ja	Nee	Ja

Support	**Prioriteit**	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
Commercial Manuals	5	Ja	Ja	Nee	Ja	Nee	Nee	Ja	Nee	Ja
Commercial Support	2	Ja	Ja	Nee	Ja	Ja	Nee	Ja	Ja	Ja
Developer Community	2	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Online Help	2	Nee	Nee	Beperkt		Nee	Nee	Nee	Ja	Ja
Pluggable API	1	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Public Forum	3	Nee	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Public Mailing List	3	Nee	Ja	Ja		Ja	Ja	Ja	Ja	Ja

Gebruiksgemak	**Prioriteit**	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
Drag-N-Drop Content	3	Ja	Beperkt	Nee		Ja	Ja	Ja	Nee	Beperkt
Friendly URLs	1	Ja	Ja	Ja		Ja	Ja	Ja	Beperkt	Ja
Image Resizing	3	Ja	Ja	Ja		Ja	Ja	Ja	Nee	Ja
Mass Upload	3	Ja	Nee	Ja		Ja	Ja	Ja	Ja	Ja
Server Page Language	1	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Template Language	1	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
WYSIWYG Editor	1	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja

Performance	**Prioriteit**	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
Advanced Caching	3	Ja	Ja	Nee	Ja	Ja	Ja	Ja	Nee	Ja
Load Balancing	3	Ja	Ja	Nee	Ja	Ja	Ja	Ja	Nee	Extra kosten
Page Caching	3	Ja	Ja	Ja		Ja	Ja	Ja	Nee	Ja
Static Content Export	3	Ja	Nee	Nee		Ja	Ja	Ja	Nee	Ja

Beheer	Prioriteit	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
Content Scheduling	3	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Inline Administration	4	Ja	Ja	Ja		Ja	Ja	Ja	Beperkt	Beperkt
Online Administration	2	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Package Deployment	3	Ja	Beperkt	Beperkt		Ja	Nee	Ja	Ja	Beperkt
Sub-sites / Roots	3	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Themes / Skins	1	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Nee
Web Statistics	3	Ja	Nee	Ja		Beperkt	Nee	Nee	Beperkt	Nee
Web-based Style/Template Management	3	Ja	Ja	Ja		Ja	Ja	Ja	Beperkt	Beperkt
Web-based Translation Management	3	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Nee

Flexibiliteit	Prioriteit	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
Interface Localization	1	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja
Metadata	3	Ja	Ja	Ja		Ja	Ja	Ja	Nee	Ja
Multi-lingual Content	1	Ja	Ja	Nee	Ja	Ja	Ja	Ja	Nee	Ja
Multi-lingual Content Integration	3	Ja	Ja	Ja		Ja	Ja	Ja	Nee	Beperkt
Multi-Site Deployment	2	Ja	Ja	Nee	Ja	Ja	Ja	Ja	Nee	Ja
URL Rewriting	3	Ja	Ja	Ja		Ja	Ja	Ja	Ja	Ja

Built-in Applications	Prioriteit	GX WebManager	InfoGlue	Community	Professional	Liferay Portal / CMS	Magnolia	Magnolia	MMBase	OpenCms
FAQ Management	4	Ja	Ja	Ja		Beperkt	Nee	Nee	Ja	Extra kosten
Search Engine	4	Ja	Gratis toevoeging	Ja		Ja	Ja	Ja	Nee	Ja
Site Map	3	Ja	Ja	Ja		Ja	Ja	Ja	Nee	Ja

Zie **Extra informatie CMS systemen** voor meer specifieke informatie over de verschillende CMS systemen. Hierin wordt per CMS een korte uitleg en beschrijving gegeven.

5.4. CMS keuze

Op basis van de vastgestelde criteria en de eigenschappen van de verschillende CMS pakketten blijven er twee CMS pakketten over die in aanmerking lijken te komen om ingezet te worden als het Newyse CMS. De overige CMS hebben allemaal teveel zwaarwegende nadelen, die vooral te vinden zijn op het gebied van de kosten en/of de licentie.

De twee CMS pakketten die over blijven zijn '*GX WebManager*' (GX, 2007) en '*Liferay*' (Liferay, 2007).
Er zijn zes factoren van doorslaggevend belang geweest voor de keuze van het CMS:

1) Licentie
2) (Aanschaf)kosten
3) Documentatie
4) Community
5) Staat van dienst
6) Functionaliteiten

GX WebManager komt in aanmerking door de ondersteuning van de grote hoeveelheid gewenste functionaliteiten en de bewezen staat van dienst bij vele grote bedrijven. *Liferay* komt ook in aanmerking als gevolg van de ondersteuning van een grote hoeveelheid van de gewenste functionaliteiten en heeft een aantal grote en bekende klanten, maar in mindere mate dan GX Webmanager (zeker op het moment dat deze keuze gemaakt werd, nu zijn er ook bij Liferay meer bekende namen als klant bijgekomen en is Sun een officiële partner geworden).

Liferay heeft echter een uitgebreider aanbod in de online documentatie en grotere en actievere community die werkzaam is met het CMS. Het gebruikte licentie model van Liferay sluit aan bij de beoogde toepassing van het CMS door Maxxton.

Tot slot worden er voor Liferay geen aanschaf en licentie kosten gerekend, in tegenstelling tot GX WebManager.

Op grond van deze factoren en de verschillen tussen de twee overgebleven CMS pakketten hebben we voor Liferay gekozen.

Om deze keuzen voor *Liferay* verder te kunnen valideren zal er een prototype van de huidige Maxxton website in Liferay ontwikkeld worden om vooral goed te ervaren hoe Liferay in het gebruik is. Hiertoe worden er nog geen aanpassingen aan het CMS zelf gedaan.

De vervolgfase hiervan is het ontwikkelen van een eenvoudige website voor het gebruik in de *Guest Service Terminal (GST)*. Dit zijn 'standalone' kasten die op een park geplaatst kunnen worden, waarmee een gast zichzelf kan aanmelden op het park (inchecken), de reservering kan betalen en de sleutelpasjes voor de accommodatie op kan halen. Hiervoor moeten er wel enkele aanpassingen in het CMS gemaakt worden om alvast een basis communicatie op te kunnen zetten met Newyse.

Tijdens deze twee fases is het goed mogelijk om te bepalen of het verwachte potentieel van Liferay ook daadwerkelijk aanwezig is. Het is ook goed mogelijk om op deze manier te werken, doordat het een gratis beschikbaar open source project is. Dit zorgt ervoor dat er direct en zonder al te veel risico op een praktische wijze georiënteerd kan worden met het CMS.

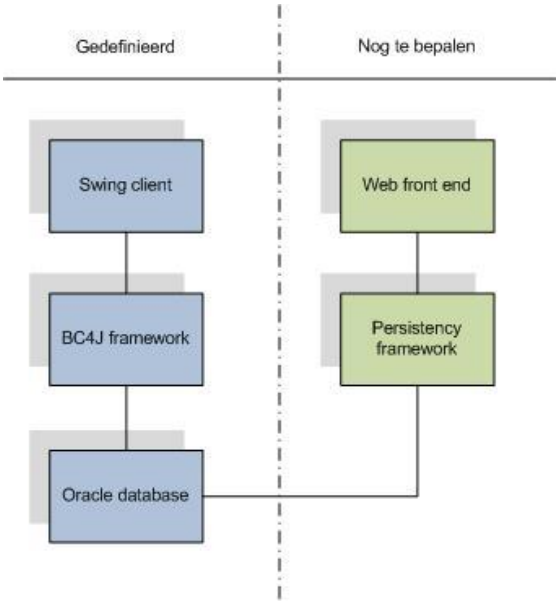
Indien er tijdens dit proces toch belangrijke gebreken naar voren komen dan is GX WebManager een mogelijk alternatief.

Ondanks dat Liferay een gratis beschikbaar ‘open source’ project is, staat er een professioneel bedrijf achter dit pakket. Dit commercieel ingestelde bedrijf behaalt de omzet uit het bieden van professionele ondersteuning en het geven van cursussen.

Om het optimale uit Liferay te halen en zo snel mogelijk overweg te kunnen met Liferay zal er ook gebruik gemaakt worden van de mogelijkheid om een cursus te volgen. Gedurende de eerste ontwikkelperiode zal er ook een support contract afgesloten worden om de ontwikkeling te versnellen en problemen op te lossen. Na het aflopen van het eerste support contract zal opnieuw overwogen worden of het support contract verlengd zal worden (indien Liferay voldoet aan de verwachtingen).

6. Technische integratie

Nu het te gebruiken CMS bekend is (Liferay) wordt het noodzakelijk om de mogelijkheden en problemen voor de integratie van het CMS te onderzoeken. Zoals in Figuur 3 is aangegeven moet er een ‘web front end’ techniek en een ‘persistancy framework’ gekozen worden. In dit hoofdstuk worden de gekozen technieken beschreven en de keuze voor die technieken gevalideerd.



Figuur 11 : Te bepalen technieken

Het is belangrijk dat er een geschikt ‘web application framework’ en een ‘persistency framework’ worden gekozen, zodanig dat deze frameworks het ontwikkelen en uitbreiden van het Newyse CMS makkelijker maken. Behalve het makkelijker maken van het ontwikkelen voor het Newyse CMS moeten deze technieken er ook voor zorgen dat de oplossing sneller en stabieler wordt. Bij een goed gekozen set aan technieken is het ook mogelijk om gebruik te maken van re-usability. Code die op veel plaatsen terug komt kan dan vanuit meerdere plaatsen gebruikt worden zonder dit steeds opnieuw te moeten doen.

6.1. Database en Applicatieserver

Met het gekozen CMS Liferay zijn er veel combinaties van te gebruiken databases en applicatie servers mogelijk.

Application Servers	
BES	Borland ES 6.5
GER	Apache Geronimo 2.x
GLF	Sun GlassFish 2 UR1
JB	JBoss 4.0.x, 4.2.x
JON	JOnAS 4.8.x
JR	JRun 4 Updater 3
O	OracleAS 10.1.3.x
ORI	Orion 2.0.7
P	Pramati 5.0
REX	RexIP 2.5
SUN	SUN JSAS 9.1
WL	WebLogic 8.1 SP4, 9.2, 10
WS	WebSphere 5.1, 6.0.x, 6.1.x

Servlet Containers	
JET	Jetty 5.1.10
RES	Resin 3.0.19
TOM	Tomcat 5.0.x/5.5.x

Databases	
AD	Apache Derby
DB2	IBM DB2
FIRE	Firebird
HYP	Hypersonic
INF	Informix
INT	InterBase
JDS	JDataStore
MY	MySQL
O	Oracle
PSQL	PostgresSQL
SAP	SAP
SQL	SQL Server
SYB	Sybase

Figuur 12 : Liferay ondersteuning voor applicatie servers en databases *(bron: www.liferay.com)*

Er is gekozen om gebruik te maken van een Oracle database omdat Newyse ook al op een Oracle database draait. Dit zorgt ervoor dat de structuur en gebruikte technieken gelijk blijven en daardoor zeer geschikt zijn voor integratie. Door het gebruik van Oracle voor Newyse is er al veel kennis en ervaring opgedaan met het werken met een Oracle database.

Om diezelfde reden is er ook gekozen voor het gebruik van de Oracle applicatieserver.

Als servlet container is er gekozen voor Apache Tomcat 6 (Apache Tomcat, 2007). Dit is de huidige versie van de bekende servlet container. Ondanks dat deze versie van Tomcat nog niet in het overzicht op de Liferay website staat[7], wordt deze nieuwste versie ook ondersteund. Tomcat kan beschouwd worden als 'proven-technology' en is een geschikte keus aangezien dit een eenvoudige en snelle implementatie biedt van de Java Servlet Specificaties (Sun, 2008). Als alternatief kan er ook gekozen worden voor de twee producten Jetty (Jetty, 2008) en Resin(Caucho, 2008). Het grote voordeel in de keuze van deze servlet container is dat de twee belangrijkste gekozen technieken allemaal ondersteuning bieden voor het gebruik van deze drie belangrijkste servlet containers. Het is dan ook eenvoudig om van servlet container te wisselen.

[7] Dit was op het moment van beslissen, maar tegenwoordig wordt Liferay ook aangeboden in een bundel met Tomcat 6.

6.2. Communicatie CMS en database

Voor de realisatie van de communicatie tussen het CMS en de database zijn er meerdere technieken beschikbaar. Een greep uit deze technieken zal hier besproken worden en er zal aangegeven worden waarom er voor gekozen is om juist die techniek wel (of niet) te gebruiken voor dit project.

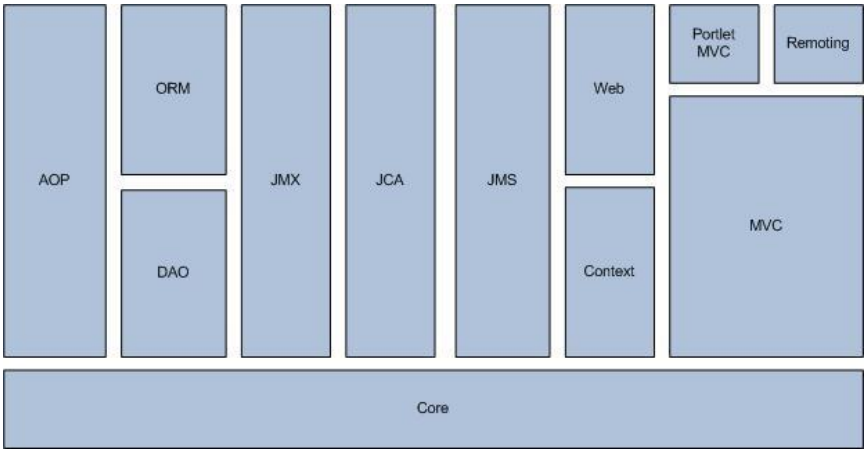
6.2.1. Spring

Voor de basis van het project wordt er gebruik gemaakt van een relatief nieuwe techniek: Spring. In het boek 'Spring in Action' (Walls, et al., 2007) wordt Spring op de volgende manier samengevat: "*Spring is a lightweight dependency injection and aspect-oriented container and framework".*

Ondanks dat deze omschrijving maar beperkt van omvang is, omvat dit wel de kern van Spring.

Het geeft aan dat het gaat om een lichtgewicht framework. De term lichtgewicht is in dit geval zelfs op twee manieren van toepassing; het framework zelf is niet groot (net iets meer dan 2,5 MB), maar het gebruik van Spring kan er ook voor zorgen dat er minder overhead ontstaat in de code. Dat is vooral een gevolg van *Depency Injection* (hierover later meer).

Deze korte omschrijving bevat veel verschillende elementen en dit blijkt ook wel als we er een schematische weergave van de beschikbare modules van Spring bijhalen.



Figuur 13 : De verschillende modules van Spring

Elke module is gebaseerd op de *core* module van Spring. Deze zorgt voor de basis van Spring. Alle overige modules zijn optioneel en kunnen naar wens ingezet gaan worden. De *core* module van Spring is verantwoordelijk voor de grote kracht van Spring; *Dependency Injection*.

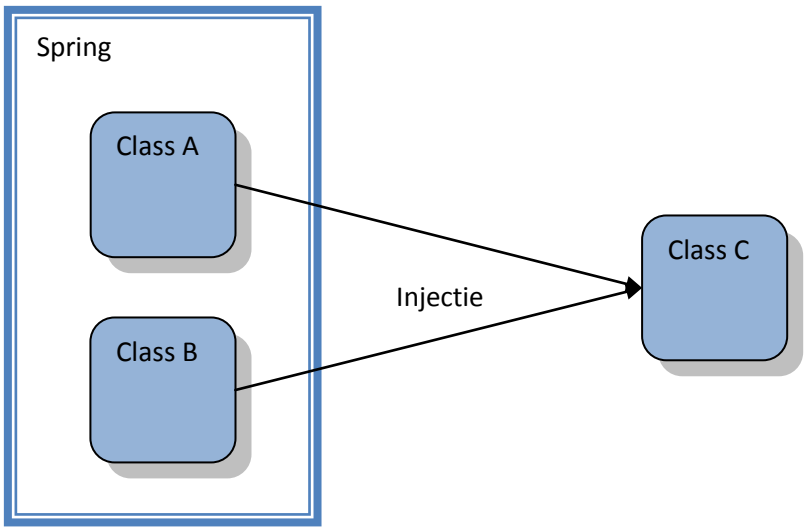
"*Spring deals with the plumbing*" is een uitspraak uit de reader van de Cursus (Interface21, 2007) om aan te geven dat het doel van Spring is om het vuile werk uit handen te nemen, zodat alle tijd en aandacht besteedt kan worden aan het daadwerkelijk oplossen van het domein probleem. *Dependency Injection* is een goed voorbeeld hiervan.

Met *Dependency Injection* wordt bedoeld dat er op een andere manier omgegaan wordt met de verantwoordelijkheid van Java classes met betrekking tot de afhankelijkheden. Nagenoeg elke applicatie zal bestaan uit meerdere Java klassen. Deze Java klassen hebben elk hun eigen afhankelijkheden. Normaal gesproken is het zo dat de Java klasse zelf verantwoordelijk is voor het

verkrijgen van de afhankelijkheden. Het nadeel hiervan is dat de code veel afhankelijkheden zal bevatten en het daardoor lastig is om de verschillende onderdelen te testen of een verandering van ontwerp door te voeren.

Met behulp van *Dependency Injection* wordt deze verantwoordelijkheid bij de Java klasse zelf weggehaald. De afhankelijkheden worden niet meer door de Java klasse zelf verkregen, maar er komt een extra entiteit bij die deze afhankelijkheden injecteert in de Java klasse.

Spring neemt deze rol van die extra entiteit op zich.



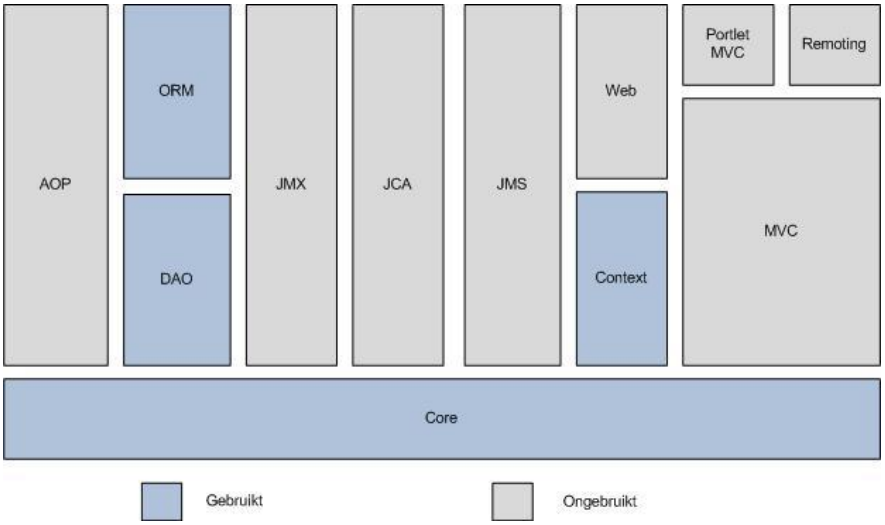
Figuur 14 : Principe van Dependency Injection

De Spring module DAO[8] in combinatie met de ORM[9] module is een hulpmiddel om op een nette manier gebruik te maken van andere ORM tools zoals Hibernate en iBatis. Hierover later meer.

De context module is een verzamel module voor verschillende kleine functionaliteiten van Spring. Voor dit project wordt er vooral gebruik van gemaakt voor de I18N[10] functionaliteiten.

Uiteindelijk wordt er voor dit project maar een gedeelte van alle Spring modules gebruikt. In het volgende schema wordt aangegeven welke modules gebruikt zullen gaan worden voor het project.

[8] Data Access Object
[9] Object-relational mapping
[10] Internationalization



Figuur 15 : De gebruikte Spring modules

In eerste instantie is er gekozen om alleen gebruik te maken van de modules ORM, DAO en Context (afgezien van de noodzakelijke core).

Doordat er nu al gebruik gemaakt wordt van het Spring framework is het goed mogelijk om in de toekomst gebruik te gaan maken van de andere modules van Spring. Bijvoorbeeld voor het gebruik van JMX[11] voor monitoring en management tools.

6.2.2. Persistency framework

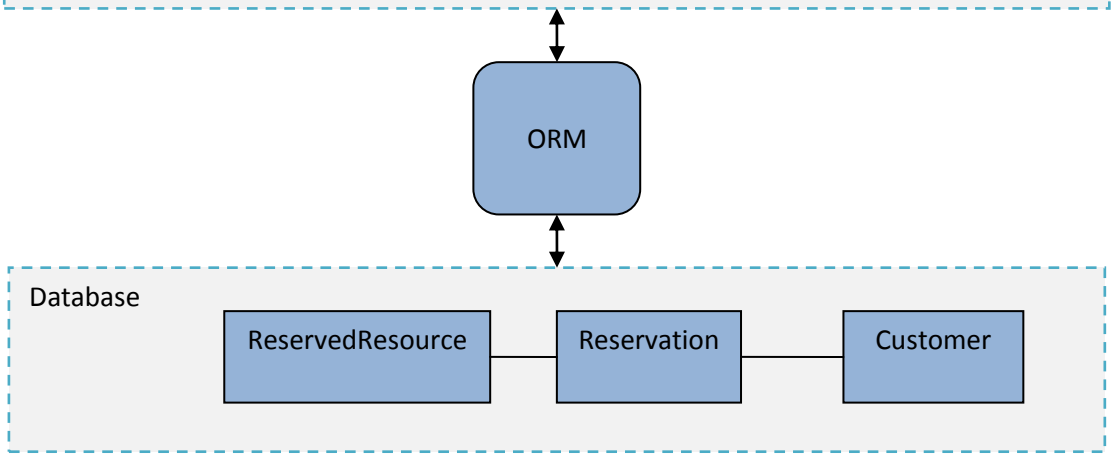
Een 'persistency framework' zorgt ervoor dat de state van een applicatie bewaard blijft ook als deze afgesloten wordt. Het is dus mogelijk om een 'persistancy framework' te hebben dat de state van een applicatie opslaat in bestanden. In veruit de meeste gevallen zal de data opgeslagen worden in een database.

Een object-relational mapping framework is een vorm van een 'persistency framework'. Een ORM framework zorgt voor het vereenvoudigen van de communicatie met de database. Het maakt het eenvoudiger om domein data objecten te vertalen naar database domein objecten. Op deze manier wordt het ontwikkelen eenvoudiger en sneller.

Door middel van het gebruik van ORM is het mogelijk om binnen de Java code gebruik te blijven maken van domein (Java) objecten. Het is dan mogelijk om met behulp van Java objecten, ook wel POJO's[12] genoemd, representaties te creëren van datamodellen uit de database. Bij het opslaan en/of ophalen van gegevens uit de database zorgt het ORM framework voor het omzetten van de POJO naar de verschillende attributen van het database model.

[11] Java Management Extensions
[12] Plain Old Java Object

```
public class Reservation
{
  private String _reservationNumber;

  private List<ReservedResource> _reservedResources;

  private Customer _customer;

  public String getReservationNumber()
  {
    return _reservationNumber;
  }

  public void setReservationNumber(String reservationNumber)
  {
    _reservationNumber = reservationNumber;
  }

  public List<ReservedResource> getReservedResources()
  {
    return _reservedResources;
  }

  public void setReservedResources(List<ReservedResource> reservedResources)
  {
    _reservedResources = reservedResources;
  }

  public Customer getCustomer()
  {
    return _customer;
  }

  public void setCustomer(Customer customer)
  {
    _customer = customer;
  }
}
```



Figuur 16 : Vereenvoudigd voorbeeld van de werking van ORM

Er zijn verschillende opties voor het te kiezen ORM framework. Twee van de belangrijkste en bekendste frameworks zijn Hibernate en iBatis. Deze twee verschillende technieken zijn allebei opties om te gebruiken bij de integratie van het CMS met Newyse.

Hibernate

Het door Liferay standaard ondersteunde ORM framework is Hibernate (Hibernate, 2008).

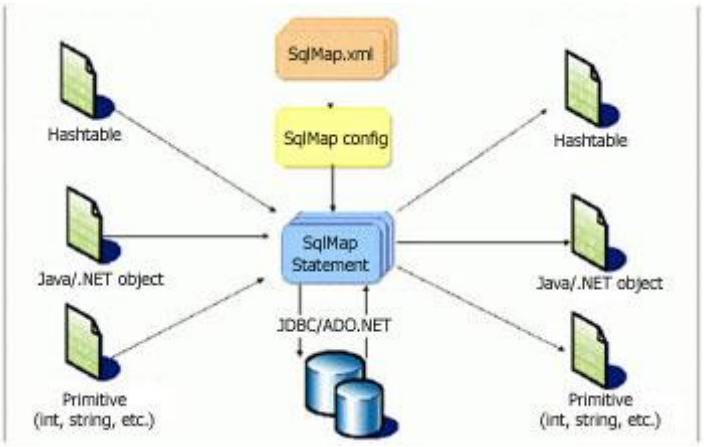
Hibernate zorgt voor de mapping van Java objecten naar database tabellen en kan hier zelf de SQL queries voor genereren. Dit maakt het gebruik van Hibernate eenvoudig, maar het beperkt ook direct de mogelijkheden.

Het is met Hibernate namelijk niet goed mogelijk om gebruik te maken van zeer specifieke eigen database queries. Dit is voor dit project toch een grote beperking, aangezien er binnen het team veel kennis bestaat over SQL en er veel gebruik gemaakt wordt van het optimaliseren van de SQL code.

Daarnaast verlangt Hibernate een bepaald database model, hierdoor is het een geschikte techniek om te gebruiken in combinatie met een nog te maken database model. Aangezien we in dit project al te maken hebben met een bestaand en complex database model is ook dit een nadeel voor het gebruik van Hibernate.

Het voordeel van Hibernate is dat het beschikt over een uitstekend cache model. Deze zorgt ervoor dat er niet onnodig vaak database queries uitgevoerd worden op de database.

iBatis



Figuur 17 : iBatis flow (bron: ibatis.apache.org)

iBatis (Apache iBatis, 2008) is volgens de eigen website een ‘Data Mapper’ framework, maar wordt vaak onder gebracht bij de ORM frameworks als gevolg van de vele overeenkomsten. In Spring in Action (Walls, et al., 2007) omschrijft Craig Walls iBatis als een ‘object-query mapping’ oplossing. De term ‘object-query mapping’ is veel treffender voor iBatis aangezien iBatis de ontwikkelaar de volledige controle geeft over de uit te voeren queries. iBatis helpt er vervolgens bij om met behulp van deze queries gegevens uit een database om te zetten naar Java objecten.

De grote kracht van dit framework zit hem in de combinatie van de eenvoud en controle over het uit te voeren SQL. Aangezien er binnen Maxxton voor Newyse veel gebruik gemaakt worden van database stored procedures en er veel SQL kennis beschikbaar is, is het van belang om gebruik te

kunnen maken van de stored database procedures en om controle te kunnen hebben over de uit te voeren SQL queries.

iBatis biedt deze mogelijkheden volop. Door gebruik te maken van .xml configuratie bestanden waar zelf SQL code aan toegevoegd kan worden is er veel controle over het gedrag van iBatis.

Neem nu bijvoorbeeld de reservering uit Figuur 16. Om het Java object te vullen met behulp van iBatis vanuit de database moet er een xml configuratie bestand aangemaakt worden, waarin de SQL map gedefinieerd staat.

```
<select id="getReservation"
    parameterClass="java.lang.Long"
    resultClass="com.example.Reservation">
      SELECT
        reservation_id,
        reservation_number
      FROM
        reservation r
      WHERE
        Reservation_id = #reservationId#
 </select>
```

```
Reservation reservation = IBatis.getObject("getReservation", id);
```

Figuur 18: iBatis voorbeeld reserveringsobject

Zodra de configuratie van iBatis goed is ingesteld, is het gebruik van de database modellen binnen de Java code haast elementair geworden. Binnen de Java code is het verder niet noodzakelijk om na te denken over de omzetting van Java object naar database model via SQL code. Dit wordt allemaal geregeld door iBatis.

Een bijkomend voordeel voor de performance is dat iBatis gebruik maakt van een caching model. De al uitgevoerde queries worden opgenomen in de cache, zodat deze bij een volgende executie sneller geretourneerd kunnen worden. Voor de gevallen waarbij caching niet gewenst is, is het ook mogelijk om dit uit te schakelen (dit is in ons geval bijvoorbeeld toe te passen op de beschikbaarheid, waarbij er geen valse beschikbaarheid mag ontstaan als gevolg van caching). Het caching model van iBatis is zeker niet zo goed als het caching model van Hibernate. Het is echter goed genoeg om relatief eenvoudige queries op een effectieve manier te cachen. Voor de moeilijkere queries is Hibernate beter, maar in de te gebruiken situaties weegt de vrijheid van de eigen SQL queries op tegen het betere caching model. Vooral aangezien de complexere queries vaak maar eenmalig uitgevoerd zullen worden, omdat ze afhankelijk zijn van unieke situaties. In die gevallen zou het effect van een caching model teniet gedaan worden.

De keuze valt in dit geval op iBatis omdat het belang van de advanced cache van Hibernate niet opweegt tegen de flexibiliteit van iBatis wat betreft het zelf kunnen maken en controleren van de SQL code.

6.2.3. Web application framework

Wicket

Figuur 19: Apache Wicket Logo

Wicket (Apache Wicket, 2008) is een Java web application framework. Met behulp van Wicket is het mogelijk om in Java web applicaties te schrijven. Het grote voordeel van Wicket is dat het eenvoudig aan te leren is en dat het zorgt voor een duidelijke scheiding van de Java code ten opzichte van de HTML code. Deze scheiding is een voordeel voor zowel de Java ontwikkelaar als de designer, omdat beiden een eigen bestand hebben om in te werken. Dit zorgt ervoor dat ze redelijk onafhankelijk van elkaar kunnen werken en elkaar niet snel in de weg zitten. In tegenstelling tot het gebruik van Struts en JSP[13] & JSF[14] zorgt Wicket voor schone en leesbare HTML code. Tevens biedt Wicket standaard ondersteuning voor het gebruik van server-side state.

Alternatieven

Er zijn veel beschikbare web application frameworks in de ontwikkeltaal Java. De twee bekendste en meest gebruikte frameworks voor web application frameworks in combinatie met een portal CMS zijn op dit moment wel Struts 2 (Apache Struts, 2007) en JSF (Sun JSF, 2007). Eén van de meest opvallende nadelen van deze twee frameworks is dat deze beide veel meer impact hebben op de opmaak van de HTML (ook wel 'intrusive' genoemd). Dit is duidelijk te maken door voor de drie technieken een klassiek Hello World voorbeeld te geven.

```
<html>
  <head>
  </head>
  <body>
    <h1 wicket:id="msg"></h1>
  </body>
</html>
```

```
public class HelloWorld extends Page
{
   public HelloWorld()
   {
      super();
      add(new Label("msg", "Hello World");
   }
}
```

Figuur 20 : Wicket voorbeeld: Hello World

[13] JavaServer Pages
[14] JavaServer Faces

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>

<html>
<head>
    <title>HelloWorldJSF</title>
</head>

<body>
    <f:view>
  <h:form>
      <h3>
    <h:outputText value="#{helloWorldBean.hello}" />
      </h3>
  </h:form>
    </f:view>
</body>
</html>
```

```
public class HelloWorldBean {

    private String hello = "Hello World!";

    public String getHello () {
      return this.hello;
    }
}
```

Figuur 21 : JSF Voorbeeld Hello World

Het valt direct op dat bij JSF er gebruik gemaakt wordt van tag-libraries. Deze moeten om te beginnen eerst geïmporteerd worden en vervolgens worden deze tags ook door de HTML code heen gebruikt om alles goed te kunnen renderen. Hierdoor wordt de leesbaarheid en aanpasbaarheid aanzienlijk kleiner.

Waarom Wicket

De grote kracht van Wicket is dat het “component based” is, wat in houdt dat het eenvoudig is om zelf basis componenten te ontwikkelen (eventueel op basis van de al bestaande componenten). Deze nieuwe componenten kunnen dan zeer eenvoudig ingezet worden in de rest van de applicatie.

Ook de duidelijk leesbare en normaal te renderen HTML bestanden zijn zeer handig. Dit stelt het ontwikkelteam in staat om een designer een pagina in HTML te laten ontwerpen. De ontwikkelaar kan dit dan snel en zonder problemen overnemen in de broncode. Ook andersom is het mogelijk om de designer achteraf even snel de HTML bestanden te bekijken (en te laten renderen), zonder dat daarvoor de server moet draaien. Dit kan allemaal geheel onafhankelijk gebeuren.

Voor het project is de ondersteuning van de server-side state ook een zeer positief kenmerk van Wicket. Bij een server-side state wordt de state van de pagina op de server bijgehouden in plaats van in de client. Dit is een geschikte aanpak voor bijvoorbeeld het boekingsproces. Tijdens het boeken moet er constant een garantie en validatie van de datamodellen zijn. Door deze op de server te

bewaren is er meer controle hierop, onder andere om de flow van de reservering te kunnen controleren en bij overgangen bepaalde stappen af te dwingen.

Dit voordeel van de server-side state is in sommige gevallen echter ook een nadeel. Want bij componenten die eigenlijk helemaal niet afhankelijk zijn van een bepaalde sessie en daarom ook geen server-side state nodig hebben, kan dit zelfs een probleem vormen. Het bijhouden van deze server-side state is namelijk sessie afhankelijk. Het gevolg hiervan is dan ook dat het component niet meer beschikbaar is indien de sessie niet meer beschikbaar is. Dit is goed in het geval van een boekingsengine, maar voor het puur ophalen van informatie uit Newyse en het weergeven daarvan is het niet gewenst dat het component kan verlopen, doordat de sessie verloopt. Dit probleem is echter opgelost door een mogelijkheid in te bouwen dat het component een timer implementeert, waardoor de sessie steeds vernieuwd wordt, voordat deze dreigt te verlopen.

Rekening houdend met de bestaande situatie heeft de keuze voor Wicket ook het voordeel dat het ontwikkelen in Wicket zeer veel lijkt op het ontwikkelen van een Swing applicatie, waardoor het voor het bestaande ontwikkelteam een lage leercurve heeft. De overeenkomsten tussen Wicket en Swing applicaties zitten in het *model based* en *event driven* ontwikkelen.

De websites moeten meerdere talen ondersteunen. Dit is ook al een belangrijke eis geweest bij het kiezen van het CMS. Het is natuurlijk ook zeer belangrijk voor het te kiezen web application framework. Wicket biedt hiervoor standaard ondersteuning dmv van resourcebundles per pagina. Deze zijn op een zeer eenvoudige manier te integreren in de pagina's.

```
<html>
  <head>
  </head>
  <body>
    <wicket:message key="message"/>
  </body>
</html>
```

```
Message=Hello World
```

```
Message=Hallo Wereld
```

Figuur 22 : Voorbeeld Wicket HTML bestand voor weergave van een taalafhankelijke tekst (statisch)

Ondanks al deze voordelen is Wicket niet direct te gebruiken als web application framework in combinatie met Liferay. Dit komt omdat Wicket wel ondersteuning heeft voor de JSR 168 specificatie (Java Specification Request, 2003), maar nog niet de JSR-286 specificatie (Java Community Process, 2008). Voor de integratie van Wicket met Liferay is er in eerste instantie gebruik gemaakt van een work-around waarbij er zowel in Wicket als in Liferay enkele aanpassingen zijn gemaakt, waardoor het mogelijk werd om Liferay portlets te ontwikkelen met Wicket.
Sinds de ondersteuning van JSR 286 in Liferay, is het alleen noodzakelijk om Wicket aan te passen zodat dit ook de JSR 286 specificatie ondersteunt. Na deze aanpassingen in Wicket is het wel mogelijk om een Liferay portlet te ontwikkelen in Wicket. Het voordeel van deze aanpassing (ten opzicht van

een aanpassing voor de JSR 168 specificatie) is dat op deze manier alleen Wicket aangepast moet worden (om Wicket ook JSR 286 ondersteuning te geven), Liferay blijft op deze manier standaard.

Deze aanpassingen in Wicket zijn gemaakt om de nieuwe portlet 2.0 specificatie te ondersteunen en zijn dan ook een nuttige toevoeging op het framework. De aanpassingen zijn ook naar de ontwikkel community opgestuurd om te kunnen opnemen in het framework.

6.3. Zoeken

De nieuwe manier van zoeken is gebaseerd op het facet searching, oftewel filteren op basis van kenmerken. De resultaten zijn alle accommodaties die nog beschikbaar zijn die voldoen aan alle geselecteerde criteria. Om de gebruiker hier prettig mee te kunnen laten werken is het van groot belang dat de response tijden erg kort zijn, waardoor de gebruiker snel door de verschillende resultaten heen kan lopen.

Het grote nadeel van deze aanpak is dat de configuratie in Newyse zodanig flexibel is dat de verschillende kenmerken op object niveau kunnen verschillen, terwijl er standaard wel op type niveau gezocht moet worden. Het resultaat wordt dat er in een zeer grote set van resultaat mogelijkheden gezocht moet worden.

In Newyse bestaat er geen overzicht van vastgestelde prijzen. Het is namelijk van te voren niet direct aan te geven wat de prijs van een reservering is aangezien deze afhankelijk is van veel verschillende factoren. Daarom zijn alleen de prijsregels opgeslagen.

Om een voorbeeld te geven: Een prijs van een reservering kan oa. afhankelijk zijn van het distributiekanaal waarmee geboekt wordt, het aantal personen, het soort personen (baby's, kinderen, volwassenen, 50-plussers), de eigenaar van het object (en het contract daarmee), de periode van aankomst, de duur van het verblijf, beschikbare kortingen, enz.

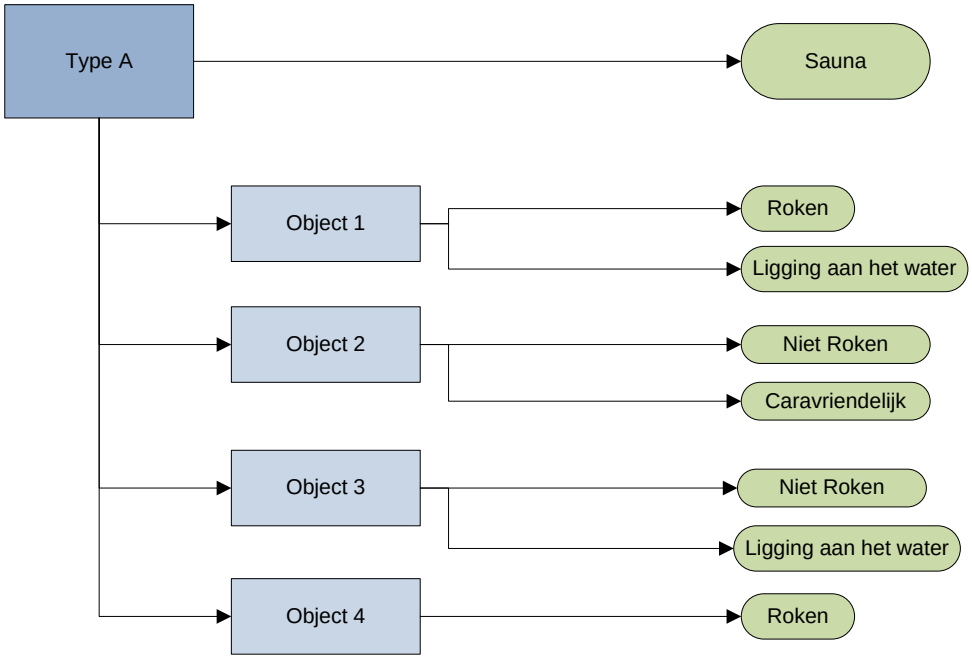
Wanneer deze prijzen van te voren berekend zouden worden, dan ontstaat dus die enorme set aan prijzen. Een rekenvoorbeeld:

```
Stel er zijn 20 accommodatie types, 10 distributiekanalen, maximaal 5 personen, 10
kortingen, 5 soorten contracten en de prijzen worden berekend voor 1,5 jaar vooruit
met een maximale verblijfsduur van 28 dagen, dan zouden er al 2.535.000.000 prijzen
berekend moeten worden (en bijgehouden).

Toelichting: rekening houdend met een samenstelling van het reisgezelschap bestaande
uit maximaal 5 personen met een onderverdeling naar 4 groepen (baby, kind, volwassene,
50+) zorgt voor 125 mogelijkheden.

1,5 jaar is gelijk aan 52*1.5 = 78 weken, met twee aankomsten per week (maandag en
vrijdag) zorgt dit voor 156 mogelijke aankomsten.

Een maximale verblijfsduur van 28 dagen met twee mogelijk aankomsten per week (maandag
en vrijdag) zorgt voor 13 verschillende verblijfsperiodes.
```

Figuur 23: Rekenvoorbeeld aantallen te berekenen prijzen

Er bestaan op dit moment al meerdere andere websites die werken met het principe van zoeken op kenmerken. Waarom is dit dan een zodanig grote uitdaging? Om dit duidelijk te maken is het nodig om meer inzicht te geven in het datamodel binnen Newyse.

Binnen Newyse wordt er onderscheidt gemaakt tussen een accommodatietype en een accommodatieobject. Een type is een manier van groeperen van verschillende accommodaties objecten. Binnen het type zal er een set van kenmerken zijn dat voor alle objecten van toepassing is, maar voor elk specifiek object kunnen er extra kenmerken gedefinieerd worden.



Figuur 24 : Kenmerken op type en object niveau

Binnen Newyse wordt er in principe standaard geboekt op accommodatieniveau en niet direct op object niveau. Deze keuze zorgt ervoor dat de parken een grotere flexibiliteit krijgen voor het indelen van de verschillende reserveringen om aan zoveel mogelijk wensen te voldoen.

```
Als er een klant belt die graag een reservering wil maken voor een accommodatietype A
met sauna en een ligging aan het water, dan voldoen zowel object 1 als object 3 aan
deze eis.
Door nu de boeking vast te leggen op accommodatietype niveau, samen met de gewenste
kenmerken, blijven beide objecten in principe nog beschikbaar. Het is nu dus nog
mogelijk om bij een volgende boeking te kunnen voldoen aan de wens om een niet roken
object aan het water te reserveren. Dit was niet mogelijk geweest als de eerste
reservering direct op object 3 geplaatst zou zijn.
```

Tevens brengt deze keuze ook extra moeilijkheden met zich mee voor bijvoorbeeld het zoeken op kenmerken voor de internetsite. Het wordt nu noodzakelijk om te gaan zoeken op twee niveaus tegelijk. Het is immers niet mogelijk om alleen op accommodatietype niveau te zoeken. Dit zorgt er namelijk voor dat mogelijk niet alle kenmerken van de objecten meegenomen worden. Aan de andere kant is het ook niet mogelijk om alleen op object niveau te zoeken. Op die manier worden niet alleen de kenmerken van het type over het hoofd gezien (deze kunnen eventueel nog wel gepropageerd worden richting de objecten), maar op de website moeten de resultaten ook getoond worden als beschikbare accommodatie types.

6.4. SOLR

In Figuur 9 is er voor het eerst gesproken over een Newyse Internet database die speciaal ingericht zou moeten worden voor het snel kunnen zoeken op internet. Bij het maken van dit ontwerp werd er nog uitgegaan dat dit een tabel zou zijn met daarin vooraf bepaalde prijzen per (o.a.) periode en accommodatie. Pas later is ook het idee ontstaan om zoeken op basis van facetten te gaan gebruiken voor de internetomgeving.

Er is in eerste instantie gekeken of het facet gebaseerd zoeken eventueel door een externe partij gerealiseerd kon worden. Dit aangezien het om hele specifieke kennis gaat.

Al snel is gebleken dat ook de externe partijen tegen problemen aan liepen (o.a. het probleem van de accommodatietypes/objecten) en er geen kant en klare oplossingen hiervoor beschikbaar waren. Daarom is besloten om een bestaande en beproefde techniek te gebruiken en dit verder uit te breiden naar de eigen wensen.

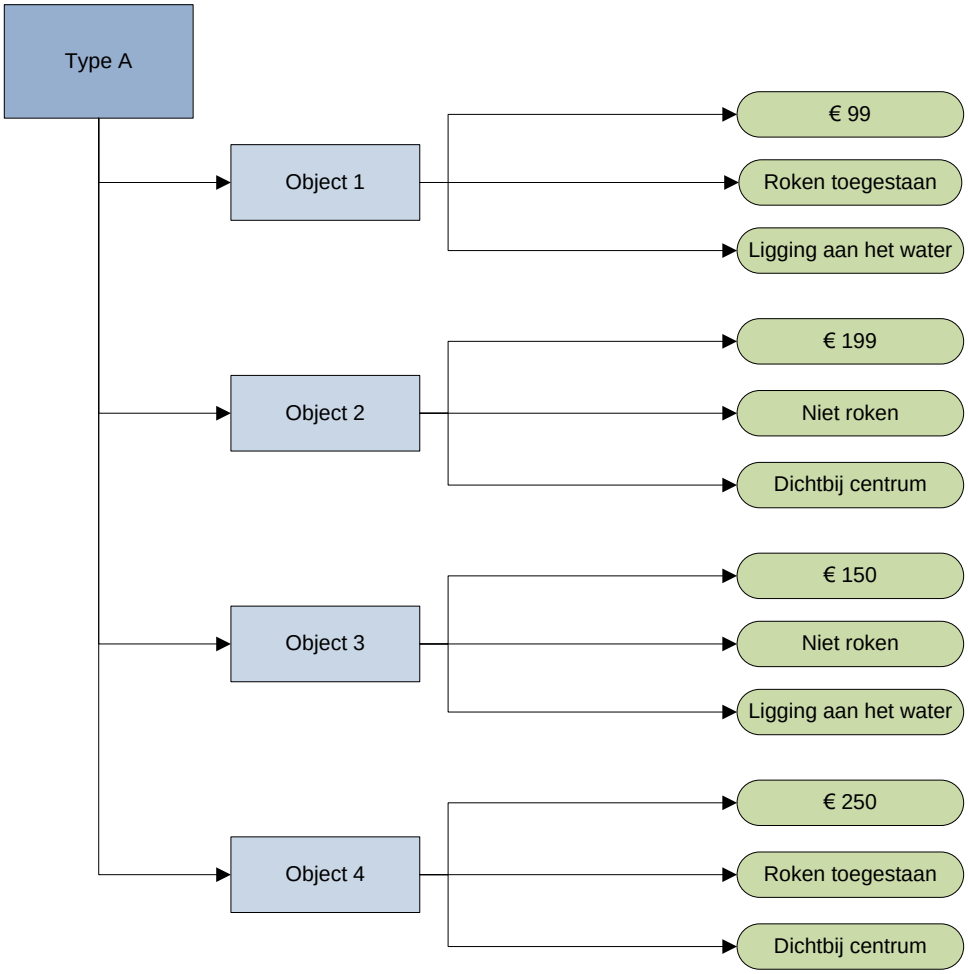
Er is gekozen om het zoeken te baseren op de techniek van SOLR (Apache SOLR, 2008) . Dit is een zoek server gebaseerd op Lucene (Apache Lucene, 2008). Het vullen van de zoekindex wordt gedaan door middel van XML bestanden.

Deze techniek is geoptimaliseerd voor het gebruik op druk bezochte websites en zeer geschikt voor het snel filteren van een grote resultaatset op basis van facetten. Een facet is een groep van zoekcriteria. Door het aanbieden van facetten wordt het voor de gebruiker eenvoudig om een snelle selectie te maken op basis van zijn of haar wensen.

6.4.1. Techniek

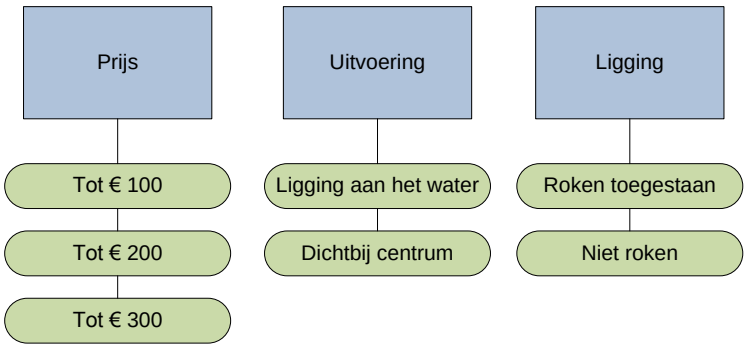
Lucene is een full-text search engine die geschreven is in Java. Lucene maakt het mogelijk om zeer snel te zoeken in stukken tekst. SOLR is een zoek server die gebaseerd is op deze full-text search engine en het facet gebaseerd zoeken combineert met het snelle zoeken van Lucene. Het basisprincipe achter SOLR is een index gebaseerd op zogenaamde documenten. Deze documenten definiëren waar er allemaal op gezocht kan worden (in onze situatie stelt een document een vakantie voor met een aankomst datum, verblijfsduur, accommodatie, enz.). Over deze documenten wordt dan gezocht door middel van facetten. Deze facetten zijn in principe groeperingen van de velden uit het document die beschikbaar zijn gemaakt voor het zoeken. Op deze manier ontstaat er een lijst met alle mogelijke verschillende opties waarop gezocht kan worden.

Om het begrip facet beter toe te lichten nemen we het voorbeeld uit Figuur 24 en breiden dit uit met enkele extra kenmerken.



Figuur 25: Objecten met kenmerken

Als we voor het voorbeeld uit Figuur 25 facetten zouden aanmaken, dan Figuur 26 daar een resultaat van kunnen zijn.



Figuur 26: Voorbeeld facetten (met bijbehorende kenmerken)

De blauwe blokken in Figuur 26 stellen de mogelijke facetten voor en de groene elementen zijn kenmerken die ingedeeld zijn onder de verschillende facetten. De gebruiker kan hiermee snel op een facet (prijs, uitvoering en/of ligging) zoeken en krijgt dan snel een resultaat set die voldoet aan de gemaakte keuze binnen een facet.

Er kan bijvoorbeeld gekozen worden voor het facet ligging, waarbij er dan gekozen kan worden uit de verschillende waardes (bijv. 'Ligging aan het water', 'Dichtbij centrum').

Het selecteren van een kenmerk uit een facet zorgt voor het beperken van de beschikbare documenten.

6.4.2. Implementatie

Voor de implementatie van de SOLR zoek index zijn er twee belangrijk problemen:

- Beperken van de set met zoekresulaten
- Het type / object probleem

Om de SOLR index goed in te kunnen richten is het noodzakelijk geweest om een specifieke implementatie te ontwikkelen voor het verkrijgen van een goede index.

De eerste stap hierin was om de resultaatset zo ver mogelijk te verkleinen, met behoudt van de belangrijke (zoek) informatie.

Een belangrijke stap hierin is het opslaan van alle geldige kortingen op een object binnen 1 zoekveld. Dit zorgt ervoor dat er altijd maar 1 rij voor een object is ook al zijn er meerdere geldig. Als prijs wordt dan de goedkoopste oplossing opgeslagen.

Een andere optimalisatie is het niet opslaan van een object per aankomst datum en verblijfsduur, maar alleen voor een bepaalde aankomstdatum en een maximale verblijfsduur. Immers als een object nog beschikbaar is voor 1 september voor 4 weken, dan is deze ook beschikbaar op 1 september voor een week.

Om het zoeken op type mogelijk te kunnen maken, maar dan wel inclusief de kenmerken van de objecten, is het noodzakelijk om een index op te bouwen die gebaseerd is op twee verschillende documenttypes. Namelijk op accommodatietype en objecten.

Het uitvoeren van een zoek actie zal dan inhouden dat er eerst gekeken wordt of er objecten beschikbaar zijn voor alle mogelijke geselecteerde facetten. De tweede stap zal zijn het zoeken (en tonen) van alle accommodatietypes op basis van deze objecten.
Op deze manier wordt het type/object probleem ook opgelost.

```
Voorbeeld: Een belangrijk voorbeeld van het type/object probleem (dat met deze aanpak
opgelost wordt) is bijvoorbeeld de situatie waarbij er 1 object beschikbaar is dat als
kenmerk heeft cara-vriendelijk en 1 object beschikbaar dat als kenmerk heeft
huisdieren-welkom. Als er alleen op accommodatietype niveau gezocht wordt, dan is het
mogelijk om een accommodatie te selecteren dat voldoet aan zowel cara-vriendelijk als
huisdieren-welkom, terwijl dit in werkelijkheid onmogelijk is.
```

6.5. Portlets

Het CMS Liferay is een zogenaamd portal systeem. Dit houdt in dat een website bestaat uit een framework, waarbinnen allemaal portlets geplaatst kunnen worden. Een portlet is een zelfstandig element in een portal systeem dat verantwoordelijk is voor de weergave (en functionaliteit) van zijn eigen gedeelte. Binnen de JSR-168[15] specificatie van portlets is het niet toegestaan om (directe) communicatie tussen de verschillende portlets te hebben.

Liferay houdt zich aan deze standaard zodat het mogelijk is om portlets volgens de specificatie te ontwikkelen die eventueel ook in een ander portal omgeving gebruikt kunnen worden. Tevens is Liferay druk bezig met de implementatie van de portlet 2.0 specificatie (Java Community Process, 2008).

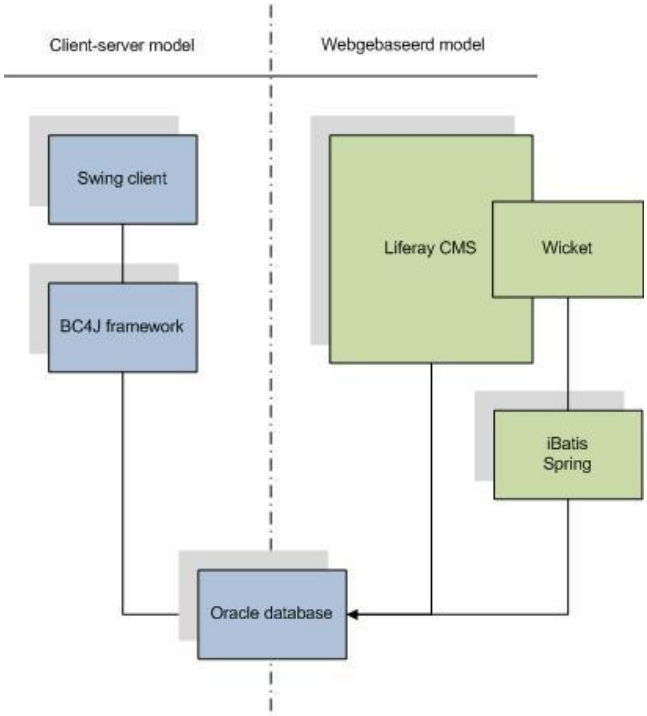
Deze portlets zijn een belangrijk aspect voor de integratie van het CMS met Newyse. Voor de integratie worden er enkele maatwerk portlets ontwikkeld die ervoor zorgen dat het CMS met Newyse geïntegreerd wordt. Deze portlets zullen worden beschreven in het volgende hoofdstuk.

6.6. Resultaat integratie

Het resultaat van deze integratie van het Liferay CMS met Newyse, waarbij gebruik gemaakt wordt van de verschillende technieken zoals Wicket, Spring en iBatis, zorgt voor een solide basis die eenvoudig verder uit te breiden valt. Door gebruik te maken van een portal systeem en de verschillende portlets te ontwikkelen met behulp van Wicket wordt het eenvoudig om nieuwe uitbreidingen voor de integratie achteraf er aan toe te voegen.

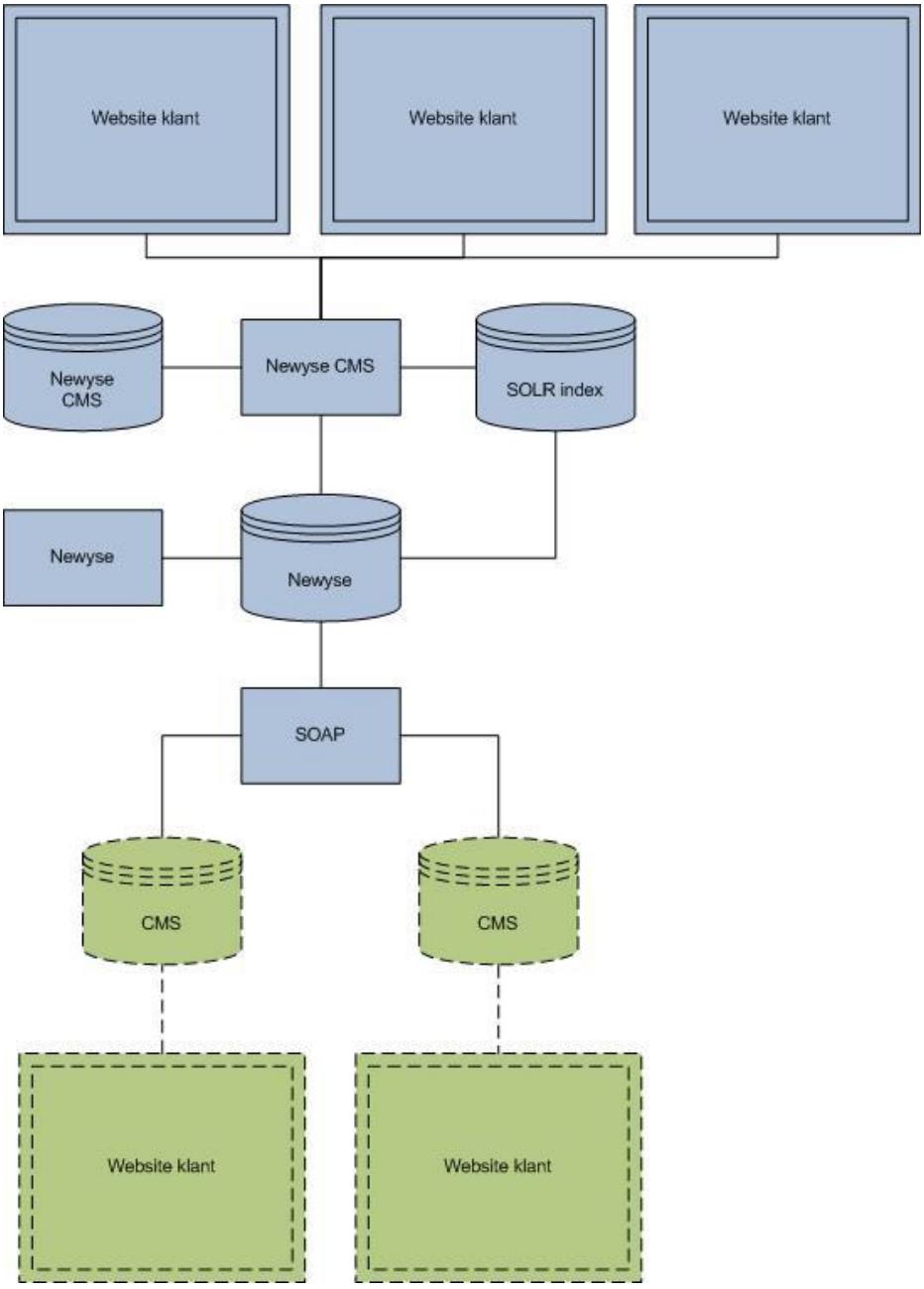
In Figuur 3 wordt er al een schematische weergave getoond van de nog te maken keuzes qua technieken. Deze keuzes zijn nu inmiddels verder ingevuld en daarmee komt het er schematisch zo uit te zien.

[15] Zie voor meer informatie ook de portlet specificaties JSR-168: http://developers.sun.com/portalserver/reference/techart/jsr168/pb_whitepaper.pdf



Figuur 27 : Bijgewerkte versie van de schematische weergave van gebruikte technieken

Er kan nu ook teruggekeken worden naar de schematische weergave van de verschillende onderdelen die te maken hebben met de structuur zoals die verwacht werd (zie Figuur 10).



Figuur 28 : Behaald eindresultaat

Opvallend bij het vergelijken van Figuur 10 en Figuur 28 is in de eerste plaats het ontbreken van de Internet database en de toevoegen van de SOLR index. Het idee van de Newyse Internet Database is geweest dat dit een mogelijkheid zou kunnen zijn om het zoeken naar beschikbaarheid en prijzen te kunnen versnellen. Dit is in ook de functie van de SOLR index. De SOLR index is er voor om op een snelle manier accommodaties te kunnen vinden op basis van kenmerken, inclusief de prijs.

7. Pilot project

Om de nieuwe technieken en de gekozen weg met Liferay te kunnen testen en te evalueren is er een pilot project opgezet.

Er is gekozen voor een pilot project voor de Newyse klant “De Krim”. “De Krim” wil graag een nieuwe website, met een nieuw ontwerp en nieuwe functionaliteiten.

Voor ons is “De Krim” een geschikte partij om een pilot project mee op te starten. Dit heeft met verschillende factoren te maken. Niet alleen is het een partij die al zeer intensief gebruik maakt van de mogelijkheden van Newyse, maar het is ook een partij die hier voor open staat en input wil geven. Op die manier heeft het resultaat van het pilot project ook direct waarde en zal het achteraf op een goede en kritische manier beoordeeld worden.

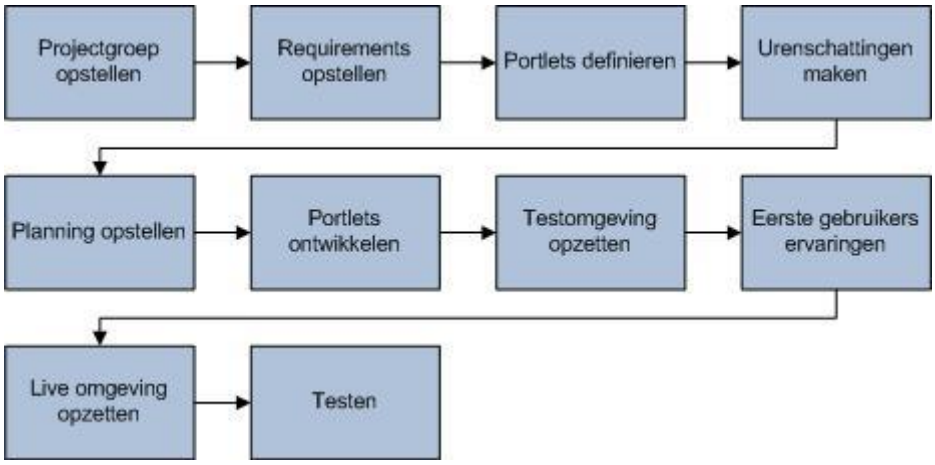
Qua Newyse configuratie beschikken ze ook over enkele uitdagende configuraties vooral met betrekking tot de nieuwe manier van zoeken. Er is namelijk een combinatie beschikbaar van accommodaties in het beheer van het park en (privé) eigenaren. Deze eigenaren zorgen ervoor dat er verschillende afspraken bestaan per object over hoe het verhuurd mag worden, wat niet mag en de inrichting van de objecten verschillen enorm.

Deze combinatie zorgt er vooral voor dat het zoeken een complexe functie wordt.

Deze klant beschikt over twee verschillende bungalowparken. Op deze twee parken staan 20 verschillende accommodaties. Er wordt ook gebruik gemaakt in de configuratie van de verschillende kenmerken van de accommodaties. Deze staan zowel op het object niveau als op het accommodatietype niveau gedefinieerd.

Deze configuratie zorgt ervoor dat het zeker een uitdagend pilot project zal worden, maar dat het nog wel te overzien valt. Op deze manier kan er goed beoordeeld worden wat de invloed van het aantal bezoekers op de performance zal zijn en wat of de gekozen technieken kunnen voldoen aan de verwachtingen van de schaalbaarheid.

7.1. Aanpak



Figuur 29 : Verschillende onderdelen pilot project

Voor het opzetten van het pilot project voor “De Krim” wordt het Liferay CMS geïntegreerd met Newyse op basis van de eerder gekozen technieken. Hierbij worden er standaard componenten ontwikkeld die direct communicatie ondersteuning richting Newyse bieden om altijd te voorzien in de nieuwste informatie.

De start van het pilot project is het opstellen van de projectgroep.

Vervolgens moet de eisen voor de nieuwe website opgesteld worden. Hierin moet worden meegenomen welke specifieke componenten er ontwikkeld moeten worden om een goede integratie met Newyse te ontwikkelen. Tevens moet het duidelijk zijn wat de klant van het project verwacht.

Met behulp van deze lijst met eisen en componenten is er een individuele urenschatting gemaakt per onderdeel. Op basis van deze urenschattingen is er een indeling gemaakt in verschillende milestones. De voorwaarde voor elke milestone is dat deze nieuwe functionaliteit en/of indrukken biedt naar de klant toe om gedurende het project een indruk te krijgen van de voortgang.

De combinatie van de eisen, de componenten, de urenschattingen en de milestones zorgt dat er een eerste versie van de planning gemaakt kan worden. Zie hiervoor de bijlage **Fout! Verwijzingsbron iet gevonden.**.

7.1.1. Projectgroep

Voor aanvang van het project is er een projectgroep samengesteld. Deze projectgroep is verantwoordelijk voor het gehele project, waarbij ieder persoon een bepaald gedeelte van het project voor zijn rekening neemt.

Bij Maxxton bestaat de projectgroep uit een totaal van 6 mensen

Projecteigenaar	Jean-Pierre Mampaey
Projectleider	Edwin de Vries
Technisch projectleider	Elwin Vreeke
Software ontwikkelaar	Rob Sonke
Software ontwikkelaar	Thijs Vonk
Software ontwikkelaar	Elwin Vreeke
Designer	Michel van Beek

7.1.2. Requirements

Om tot een goed eindresultaat te kunnen komen is het noodzakelijk om voor aanvang duidelijk te krijgen wat de verwachtingen en eisen van alle partijen zijn.

Daarom is er in de beginfase veel overleg geweest met “De Krim” over de verwachtingen en eisen van het eindresultaat. Deze eisen zijn vastgelegd in de bijlage Pakket van eisen pilot project.

7.1.3. Portlet overzicht

Op basis van de wensen die naar voren komen in het pakket van eisen voor de website is er een lijst samengesteld van de verschillende custom portlets die ontwikkeld moeten worden. Het maken van een overzicht van de te maken portlets maakt het mogelijk om een planning op te zetten.

De complete lijst met portlets is hier bewust weggelaten, aangezien dit verder weinig toegevoegde waarde heeft. Het opstellen van de lijst is wel van essentieel belang geweest voor het verloop van het project.

7.1.4. Urenschattingen

Op basis van de gemaakte lijst met portlets is er in overleg met alle betrokken ontwikkelaars een urenschatting opgesteld zodat het mogelijk werd om een planning op te zetten.

7.1.5. Initiële planning

Met behulp van de eerdere lijst met te ontwikkelen portlets en de bijbehorende urenschattingen was het mogelijk om een initiële planning op te zetten. In mijn rol als technisch projectleider ben ik verantwoordelijk geweest voor het verzamelen van de juiste informatie, het maken van de urenschattingen (in overleg met de andere ontwikkelaars) en het opzetten van de initiële planning om op basis van de inhoudelijke kennis een logische volgorde neer te zetten. Het is nadrukkelijk niet mijn verantwoordelijkheid geweest om de planning up-to-date te houden en de planning te

bewaken. Dit is de rol van de algemene projectleider. Het adviseren voor de planning op basis van inhoudelijke kennis is wel onderdeel van mijn verantwoordelijkheid.

7.1.6. Portlet ontwikkeling

In deze fase worden alle gedefinieerde portlets ook daadwerkelijk ontwikkeld.

7.1.7. Testomgeving opzetten

Er moet een omgeving opgezet worden waarbij er een applicatie server opgezet wordt waarop vervolgens het Newyse CMS draait. Vervolgens is het nodig om een indruk te geven van het uiterlijk, de inhoud en de mogelijkheden van het CMS.

7.1.8. Gebruikers ervaringen

Zodra er een test omgeving beschikbaar is kunnen de gebruikers voor het eerst gaan rond kijken en een eerst indruk krijgen van het Newyse CMS. Op basis van de ontvangen feedback kunnen er dan al in een vroeg stadium wijzigingen doorgevoerd worden.

7.1.9. Live omgeving opzetten

Dit is in principe hetzelfde als het opzetten van een test omgeving, maar dan nu met een zo schoon mogelijke database, waarop dan een begin gemaakt kan worden met het vullen en inrichten van de uiteindelijke live omgeving. Hierbij wordt dan ook de klant betrokken zodat deze zelf verantwoordelijk wordt voor de inhoud.

7.1.10. Testen

De laatste fase van het pilot project is de test fase. Behalve dat er tussentijdse tests zijn uitgevoerd zal er op het laatst een totale test uitgevoerd worden.
Voor de testfase zal het projectteam bij Maxxton uitgebreid worden met enkele testers. Het gaat hier dan juist om personen die nog niet betrokken zijn geweest bij het hele project, zodat er op een onbevooroordeelde manier getest kan worden. Deze zullen dan de website in het geheel gaan testen.
Vervolgens zal ook de klant nog gaan testen (in eerste instantie zelf, later ook met behulp van een onafhankelijk testpanel).

7.2. Portlets

Een portlet is, zoals eerder al genoemd, een zelfstandig stukje informatie dat in een portal geplaatst kan worden. Deze portlets spelen een sleutelrol binnen de integratie van het CMS Liferay met het eigen backoffice pakket Newyse. Om het mogelijk te maken de website te voorzien van dynamische en realtime informatie die gebaseerd is op de bestaande gegevens uit Newyse, zal er gebruik gemaakt worden van portlets.

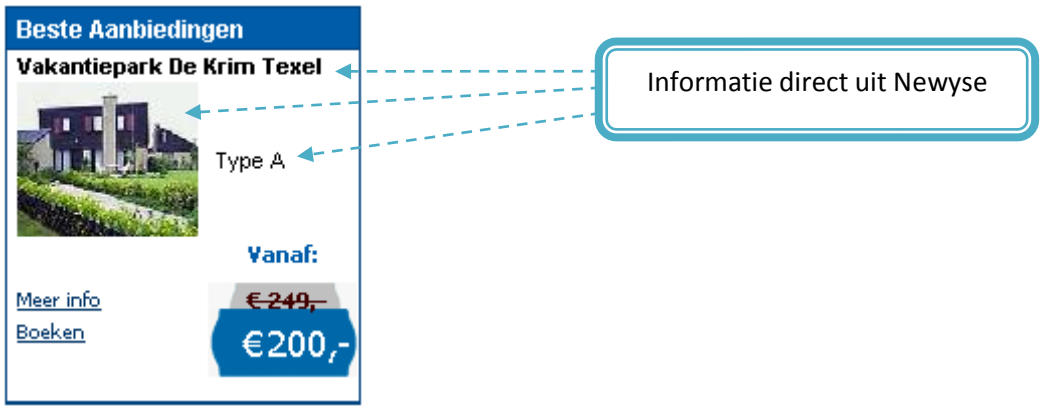
Voor elk specifieke informatie en/of functionele behoefte zal er een portlet op maat ontwikkeld worden met behulp van de eerder beschreven technieken. De uiteindelijke eindgebruiker kan dan eenvoudig één (of meerdere) van deze portlets op de website plaatsen om dynamische informatie uit Newyse op te halen (en eventueel informatie aan te passen of toe te voegen).

In dit hoofdstuk wordt er een voorbeeld gegeven van een aantal van deze verschillende portlets.

7.2.1. Accommodatietype informatie

De accommodatietype informatie portlet is een van de meest eenvoudige portlets qua opzet. Deze portlet zal namelijk direct informatie uit Newyse ophalen over een bestaand accommodatietype.

Deze informatie kan dan getoond worden op de website. Het voordeel hiervan is dat de informatie nu voortaan op één locatie kan worden ingevoerd en opgeslagen. Een aanpassing van de informatie in Newyse heeft ook gelijk tot gevolg dat het op de website aangepast wordt.



Figuur 30 : Voorbeeld van hoe de accommodatie portlet er uit kan zien

```
public class HomePage
{
  @SpringBean private AccommodationTypeDao _accoTypeDao;

  public HomePage(final PageParameters parameters)
  {
    AccommodationTypeFilter filter = new AccommodationTypeFilter();
    // Vul enkele filter kenmerken in

    AccommodationType accommodationType = _accoTypeDao.getAccommodationType(filter);

    // Gebruik het accommodation type voor het tonen op de pagina.
  }
}
```

Hierboven ziet u een fragment van een Wicket pagina waarbij er een aantal punten aandacht verdienen. De Java klasse begint met '*@SpringBean'*, hiermee wordt aangegeven dat Spring ervoor zorgt dat de Java klasse AccommodationTypeDao wordt geïnjecteerd in deze klasse. Dit attribuut wordt dan ook niet gecreëerd in deze klasse. Daar zorgt Spring voor. Met behulp van de volgende configuratie weet Spring dan welke Java klasse geïnjecteerd moet worden.

```
<bean id="accommodationDao"
           class="maxxton.newyse.web.service.accommodation.dao.AccommodationDaoImpl">
    <property name="dataSource" ref="dataSource" />
    <property name="sqlMapClient" ref="sqlMapClient" />
</bean>
```

Nadat de DAO[16] door Spring aan de Java klasse is toegevoegd kan deze direct gebruikt worden. Deze specifieke DAO is verantwoordelijk voor het ophalen van gegevens over de accommodatie type uit de database. De connectie met de database wordt in dit geval verzorgd door iBatis.

```
public class AccommodationTypeDaoImpl extends SqlMapClientDaoSupport implements
AccommodationTypeDao
{
  public AccommodationType getAccommodationType(AccommodationTypeFilter filter)
  {
    return (AccommodationType) getSqlMapClientTemplate().
                                    queryForObject("Resource.getResource", filter);
  }
}
```

Fragment uit de AccommodationTypeDao, waarbij iBatis gebruikt wordt om een AccommodationType object uit de database op te halen. In een configuratie bestand van iBatis wordt vastgelegd op welke manier iBatis informatie uit de database kan halen en hoe dit omgezet kan worden in een Java object.

[16] DAO – Data Access Object

```
<select id="getAccommodation" resultMap="accommodationResult"
cacheModel="accommodationcache">
  SELECT
    ob.object_id,
    ob.asset_id,
    ob.code,
    NVL(i18n.name, di18n.name) as name,
    NVL(i18n.short_description, di18n.short_description) as short_description,
    NVL(i18n.description, di18n.description) as description,
    ob.propertymanager_id,
    ob.imagemanager_id
  FROM objectbase ob
  JOIN language l on (l.short_name = #language#)
  LEFT JOIN i18n_objectbase i18n on (i18n.object_id = ob.object_id and
                                    i18n.language_id = l.language_id)
  LEFT JOIN i18n_objectbase di18n on (di18n.object_id = ob.object_id and
                             di18n.language_id = contextpkg.getDefaultLanguageId)
  WHERE ob.object_id = #objectId#
</select>
```

7.2.2. Brochure aanvraag

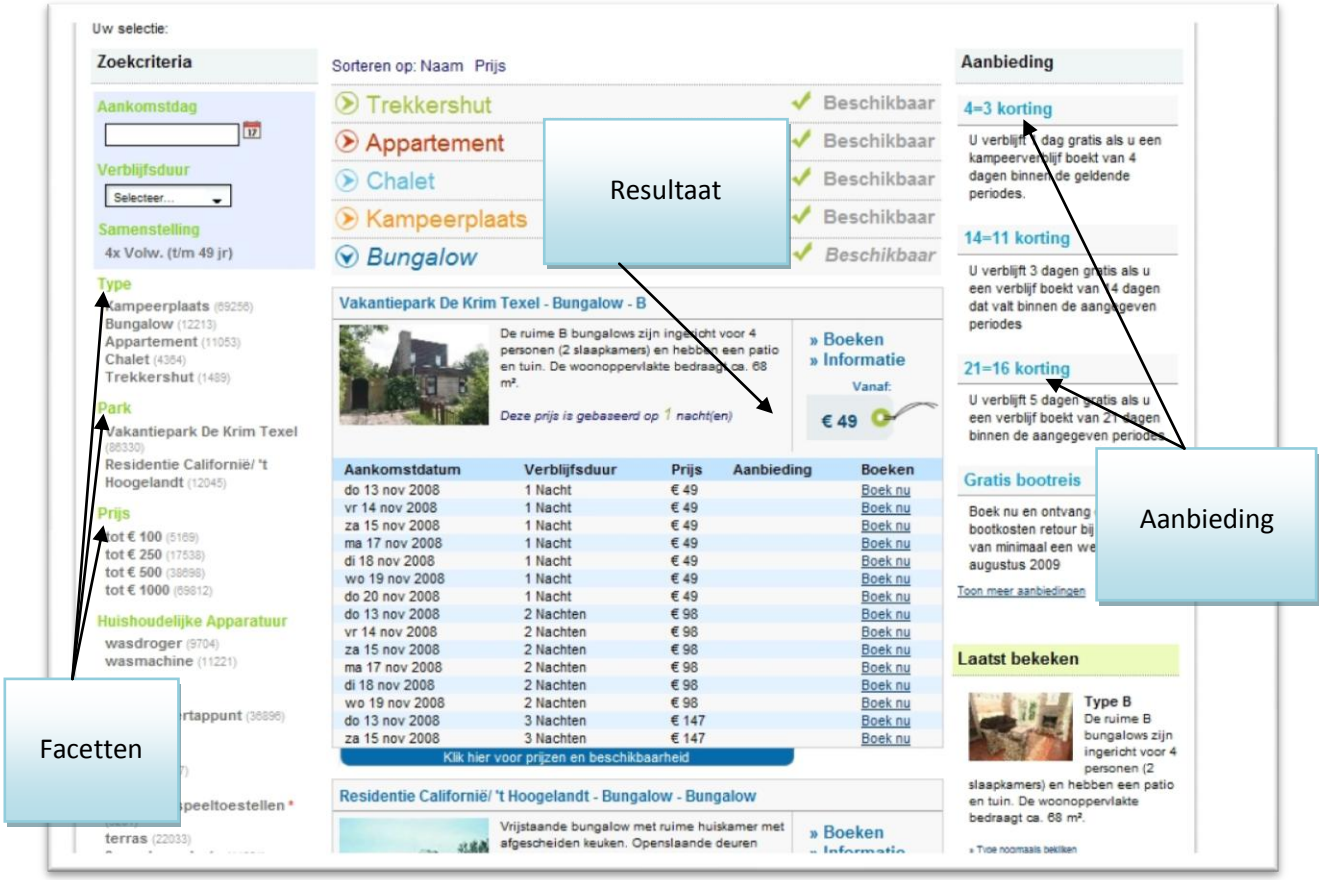
De brochure aanvraag portlet gaat iets verder dan de accommodatietype portlet. Deze portlet maakt het niet alleen mogelijk om informatie uit Newyse op te halen, maar ook om er weer nieuwe informatie in terug te zetten. Door middel van deze portlet is het mogelijk om op een snelle en eenvoudige manier een bezoeker de gelegenheid te geven om zelf een brochure aan te vragen.

7.2.3. Zoek en Boek

De Zoek en Boek portlet is de meest complexe portlet van allemaal. Deze maakt ook direct gebruik van alle eerder beschreven technieken. Het zorgt er eerst voor dat het voor de bezoeker mogelijk is om op basis van kenmerken de juiste accommodatie te kunnen vinden. En deze bezoeker kan dit ook direct voortzetten in het maken van een daadwerkelijke reservering.

Een van de eerste ideeën was om het zoeken en het boeken onder te brengen in verschillende portlets om de flexibiliteit van het inrichten van de website verder te vergroten. Zodra het allemaal binnen 1 portlet zit is het lastig om hier veel variatie in aan te brengen. Doordat tijdens de ontwikkeling van de portlets de portlet 2.0 specificatie JSR 286, waarin onderlinge gegevensuitwisseling is toegestaan, nog niet voltooid was is er besloten om alles toch in 1 portlet te integreren. Op deze manier is het goed mogelijk om de zoekresultaten en reserveringsgegevens continue door te geven naar de volgende stappen.

Ook de zoek afhankelijke informatie zoals laats bekeken objecten en resultaat afhankelijke kortingen zitten op dit moment binnen 1 portlet geïntegreerd.



Figuur 31 : Zoekresultaat

7.3. Resultaat

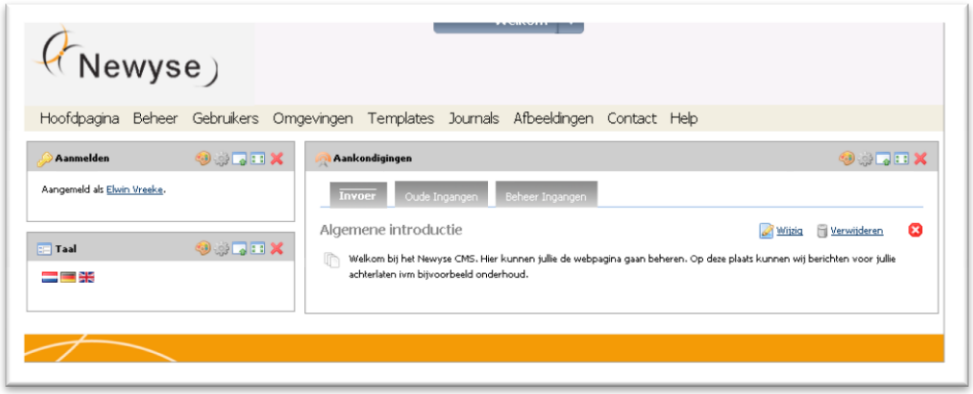
Het resultaat van dit pilot project is een zeer flexibele website geworden met een nauwe integratie met Newyse. Het is voor de gebruikers eenvoudig om de indeling en het uiterlijk van de pagina zelf aan te passen, maar ook is het voor een eindgebruiker van het CMS eenvoudig geworden om Newyse informatie te tonen. Dit is gerealiseerd d.m.v. de (maatwerk) portlets met Newyse integratie. Deze portlets kunnen direct uit het menu in het CMS gekozen en geplaatst worden, waarbij het eenvoudig is om een combinatie van Newyse CMS en Newyse informatie op een pagina te tonen.

Figuur 32 : Voorpagina van de website

Figuur 33: Combinatie CMS en Newyse informatie op de zoekpagina

In de bovenstaande figuren wordt van twee pagina's in kleur aangegeven welke gedeeltes uit Newyse komen (rechts) en welke gedeeltes informatie uit het CMS (links) bevatten.

Figuur 34 : Voorbeeld portal systeem met de mogelijkheid tot slepen van onderdelen



Figuur 35 : Newyse CMS gedeelte van de website

8. Evaluatie

De planning is om na afloop en voltooiing van het pilot project een evaluatie te houden over het gelopen project. Hierbij worden de verschillende betrokkenen ondervraagd over de bevindingen wat betreft het verloop van het project en het eindresultaat.
Hierbij wordt dan de mening gevraagd van zowel Java ontwikkelaars, ontwerpers, gebruikers als directie.

9. Conclusies & Aanbevelingen

Voor Maxxton als bedrijf is de doelstelling van het project geweest om een integratie te ontwikkelen tussen Newyse en een internet CMS. Deze integratie moest voorzien in de behoefte om van Newyse een totaal oplossing te maken en op deze manier te ontsluiten naar het internet.

Door middel van het pilot project zijn de gekozen producten en technieken toegepast voor een praktische oplossing. De opgeleverde website voldoet aan de gestelde verwachtingen:

- Facet gebaseerd zoeken op basis van actuele beschikbaarheid
- Lage responsetijden (een goede performance)
- Flexibiliteit in het maken en vullen van de website
- Directe en actuele informatie tonen uit Newyse op de website

Van het gekozen CMS 'Liferay' is gebleken dat het flexibel genoeg is geweest om het te kunnen koppelen met Newyse en om hier zelf de eventuele aanpassingen in aan te brengen. Opvallend aan 'Liferay' is gebleken dat de in eerste instantie terughoudende benadering van een portal CMS onterecht is gebleken. Juist dit portal systeem is een zeer gebruiksvriendelijk element geworden in het CMS en voorziet daarmee in een belangrijk onderdeel geworden van het uiteindelijke resultaat. Dit portal systeem maakt het ook zeer goed mogelijk om nieuwe, losstaande portlets te ontwikkelen, met of zonder integratie met Newyse.

De complexiteit en flexibiliteit van 'Liferay' als CMS is ook tevens één van de gevaren in het gebruik van 'Liferay'. Er zijn immers zoveel mogelijkheden, dan het niet moeilijk voor te stellen is dat een eindgebruiker de weg kwijt raakt. 'Liferay' heeft hiervoor wel de mogelijkheid om door middel van rechten en rollen een eenvoudigere opstelling neer te zetten.

Wicket voldoet aan de verwachtingen. Het is eenvoudig in gebruik en herkenbaar voor de ontwikkelaars met Java Swing ervaring. De mogelijkheid om zelf eenvoudig nieuwe componenten te maken is krachtig en nuttig.

iBatis als Object-Query mapper zorgt voor een eenvoudige koppeling tussen een Java object en een database tabel met behoudt van de kracht van SQL. Dankzij een goed cache model van iBatis worden er veel SQL aanroepen voorkomen, maar het blijft wel mogelijk om geen gebruik van de cache te maken als dat niet gewenst is.

Het zoeken met behulp van SOLR is voor een klant zoals "De Krim" een geschikte oplossing. Met een dataset van die grote biedt SOLR een uitstekende zoek server met facet mogelijkheden. Voor een grotere klant zal de schaalbaarheid verder onderzocht moeten worden.

9.1. Aanbevelingen voor de toekomst

9.1.1. Inter-portlet communicatie

In de eerste portlet specificatie (JSR 168: Portlet specificatie) is het niet toegestaan was om inter-portlet communicatie uit te voeren. Daardoor werden wij verplicht tot het maken van 1 portlet die verantwoordelijk is voor het hele zoeken en het maken van een reservering. Anders zou het teveel problemen met zich mee zou brengen om dit in verschillende portlets onder te brengen.

Sinds 12 juni 2008 is echter de nieuwe specificatie goedgekeurd (JSR 286: Portlet 2.0 specificatie), waarbij het nadrukkelijk wel is toegestaan om communicatie tussen verschillende portlets uit te voeren. Ook heeft 'Liferay' al aangegeven bezig te zijn met het implementeren van deze specificatie, waardoor deze specificatie al standaard door het gekozen CMS ondersteund zal worden.

Deze ontwikkelingen moeten in de gaten gehouden worden en zodra dit mogelijk is toegepast worden op de huidige integratie. Het gebruik van losse portlets geeft een website beheerder nog meer vrijheid in het indelen van de website.

9.1.2. Webgebaseerde Newyse variant

De derde probleemstelling valt buiten de scope van dit project en is dan ook niet uitvoerig onderzocht in de afgelopen periode. Wel blijkt de integratie met Newyse en de ontwikkeling van specifieke portlets zodanig succesvol te zijn, dat het aan te bevelen is om bij het eventueel alsnog overwegen om een weggebaseerde Newyse variant te ontwikkelen niet alleen kritisch te onderzoeken of dit nog steeds noodzakelijk is, maar om ook te overwegen of het CMS 'Liferay' hier niet voor geschikt is.

Dit zal vooral veel potentie hebben wanneer er maar behoefte is aan enkele modules uit Newyse, waardoor die functionaliteit in een aparte portlet gestopt kan worden.

10. Bibliografie

Een deel van de in deze bibliografie genoemde werken zijn niet direct als referentie gebruikt binnen dit afstudeerrapport. Toch zijn deze opgenomen in de bibliografie aangezien deze hebben bijgedragen aan de algemene beeldvorming over de betreffende onderwerpen en daarmee aan het tot stand komen van dit afstudeerrapport.

Apache iBatis. 2008. Apache iBatis. *Apache iBatis.* [Online] Apache, 2008. http://ibatis.apache.org/.

Apache Lucene. 2008. Apache Lucene. *Apache Lucene.* [Online] Apache, 2008. http://lucene.apache.org/java/docs/index.html.

Apache SOLR. 2008. SOLR. *SOLR.* [Online] 2008. http://lucene.apache.org/solr/.

Apache Struts. 2007. Apache Struts 2. [Online] 2007. http://struts.apache.org/2.x/index.html.

Apache Tomcat. 2007. Apache Tomcat. *Apache Tomcat.* [Online] Apache, 2007. http://tomcat.apache.org/.

Apache Wicket. 2008. Apache Wicket. *Apache Wicket.* [Online] 2008. http://wicket.apache.org/.

Caucho. 2008. Resin. [Online] 2008. http://caucho.com/.

CMS Matrix. 2007. CMS Matrix. [Online] 2007.

Cotterell, Bob Hughes and Mike. 1999. *Software Project Management (Second edition).* s.l. : McGraw-Hill International (UK) Limited, 1999.

Earl, Micheal J. 2000. *Management Strategieën en Informatietechnologie.* sl : Academic Service, 2000.

GX. 2007. GX Webmanager. [Online] 2007.

Heuvel, Hans de Bruijn & Ernst ten. 2008. *Management in Networks.* s.l. : Routledge, 2008.

Hibernate. 2008. Hibernate. [Online] 2008. http://www.hibernate.org.

Interface21. 2007. *Core Spring from the Source.* 2007.

Java Community Process. 2008. JSR 286: Portlet 2.0 specificatie. *Java Specification Request 286.* [Online] Java Community Process, 2008. http://jcp.org/en/jsr/detail?id=286.

Java Specification Request. 2003. JSR 168: Portlet specificatie. [Online] 2003. http://jcp.org/en/jsr/detail?id=168.

Jetty. 2008. Jetty. [Online] 2008. http://jetty.mortbay.org/jetty/.

King, Christian Bauer & Gavin. 2006. Java Persistance with Hibernate. s.l. : Manning Publications, 2006, pp. 37-104.

Liferay. 2007. Liferay. [Online] 2007.

Machacek, Rob Harrop and Jan. 2005. *Pro Spring.* s.l. : Apress, 2005.

Spring. 2007. Spring source. *Spring Source.* [Online] 2007.

Sun. 2008. Java Servlet. [Online] 2008. http://java.sun.com/products/servlet/index.jsp.

Sun JSF. 2007. JavaServer Faces. [Online] 2007.

Walls, Craig en Breidenbach, Ryan. 2007. *Spring in Action.* sl : Manning, 2007.

11. Bijlagen

11.1. Extra informatie CMS systemen

11.1.1. GX WebManager

http://www.gx.nl

Het GX WebManager CMS is een Nederlands CMS met veel grote en bekende klanten.

Zelf zeggen ze hierover op de website:

"*GX heeft in de afgelopen tien jaar honderden weboplossingen gerealiseerd voor organisaties uit verschillende landen en in uiteenlopende branches zoals Telecom, Financiële Dienstverlening, Professionele Sportorganisaties, Gemeentelijke Overheid, Uitgeverijen & Media, Retail en Corporate Accounts.*"

Enkele van de bekende klanten zijn: "Nationale Postcode Loterij", "KPN" en "Menzis". Het gaat hier om een professioneel product van een leverancier met veel ervaring op het gebied van CMS systemen.

Voordelen

Het CMS is ontwikkeld door een professioneel bedrijf met vele jaren ervaring met CMS systemen. Het huidige CMS is al uitvoerig in gebruik bij vele verschillende klanten, wat kan resulteren in een bekende en stabiele omgeving.

Het GX Webmanager CMS biedt ook erg veel mogelijke opties aan. Veel van deze losse opties zijn dan ook los aan te schaffen.

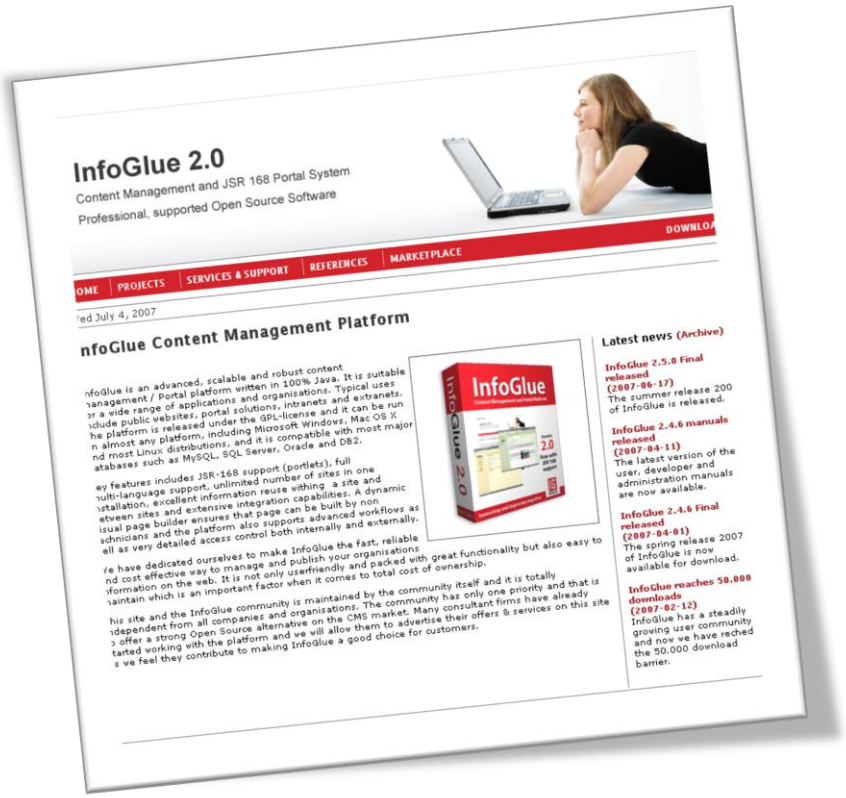
Nadelen

Het los aanschaffen van de verschillende modules kan gezien worden als een voordeel. Immers je schaft alleen die onderdelen aan die nodig zijn en de rest blijft achterwege. Dit zorgt voor lagere (let op lagere en niet perse lage) kosten.

De website is niet zeer duidelijk in het geven van informatie. Het is dan ook duidelijk dat het geen gratis pakket is en er zeker een flink bedrag voor betaald moet worden.

11.1.2. InfoGlue

http://www.infoglue.org



InfoGlue Content management systeem is een open source project, waarbij het wel ondersteunt wordt door een profesionele organisatie.

Voordelen

Het InfoGlue project is een open source project waardoor het eenvoudig is om de broncode te bemachtigen. Tevens biedt InfoGlue voldoende functionaliteiten die nodig zijn voor het Newyse CMS.

Nadelen

Een groot nadeel van het InfoGlue CMS is dat het uitgebracht wordt onder de GNU GPL licentie. Dit houdt in dat het een open source pakket is en dat het ook aangepast en gedistribueerd mag worden, maar dat de broncode van de gewijzigde versie ook openbaar gemaakt moet worden.

Dit is niet de bedoeling als we een Newyse CMS willen gaan ontwikkelen.

Er is geen probleem notificatie (geen sms of email bij problemen), maar eventuele problemen met de server kunnen ook op een andere manier in de gaten gehouden worden.

11.1.3. Jahia

http://www.jahia.org

Jahia is een professioneel CMS, dat verkrijgbaar is in vier verschillende versies.
Van een gratis 'Community' versie tot een zeer duur 'Enterprise' pakket.

De verschillende versies hebben zeer uiteenlopende functionaliteiten. Deze worden goed weergegeven door de op de site vermelde 'Features Matrix': zie www.jahia.org/matrix

Het heeft dan ook geen zin om Jahia in z'n geheel te beoordelen, de versie zijn te veel verschillend. Er is dan ook gekozen om twee versies te vergelijken:

Community en Professional

Community

Voordelen
Het grote voordeel van deze versie zijn uiteraard de kosten; het gaat om een gratis te gebruiken versie.

Nadelen
Een zeer belangrijk gemis aan deze versie is dat het geen ondersteuning heeft voor websites in meerdere talen. De interface zelf kan wel gebruikt worden in meerdere talen. Door het gebrek aan taalondersteuning is dit geen optie.

Professional

Voordelen

Eigenlijk alle gezochte functionaliteiten zijn aanwezig, waaronder ook de ondersteuning voor websites in meerdere talen.

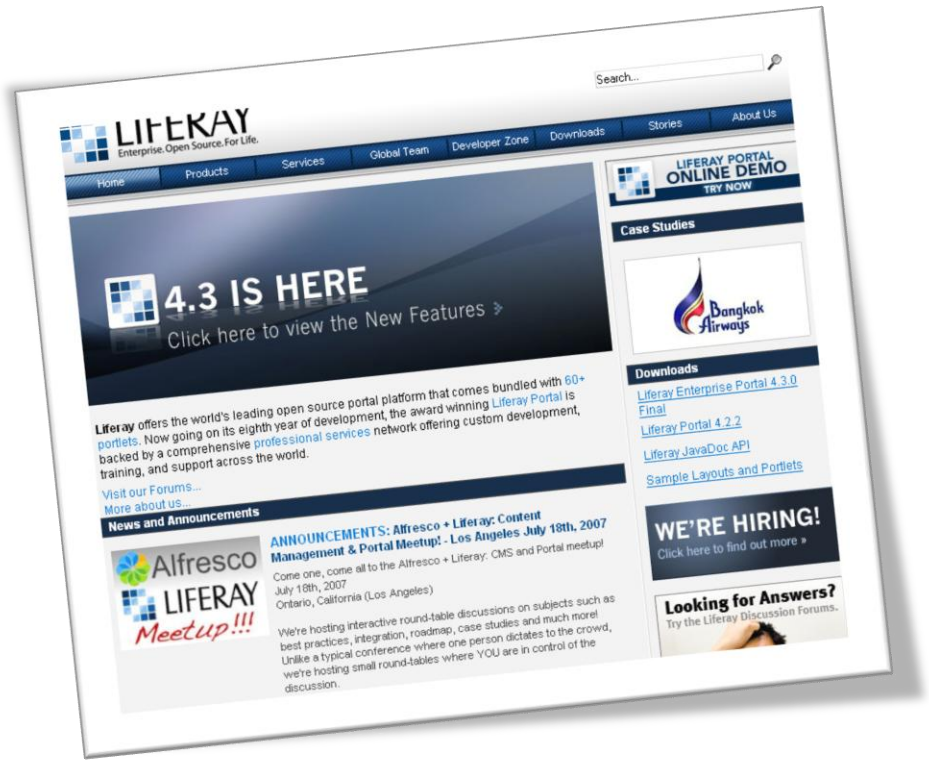
Nadelen

Er zitten zeer hoge kosten aan het gebruik van de Jahia Professional versie.

Het gaat hier om € 19990,00 per JVM (java virtual machine). Dat houdt in dat dit betaald moet worden voor elke instantie die opgestart wordt. Bij gebruik van loadbalancing worden er al snel meerdere instanties gebruikt.

11.1.4. Liferay

http://www.liferay.com



Liferay is een professioneel ontwikkeld open source CMS.

Voordelen

- De kosten, want het is een gratis pakket.
- De licentie is volledig vrij. Het CMS mag aangepast worden en zonder voorwaarden weer worden gedistribueerd/verkocht.
- Er is een grote en actieve community en veel documentatie, op deze manier is er bij problemen veel informatie te vinden.

Nadelen

- Er is geen probleem notificatie (geen sms of email bij problemen), maar eventuele problemen met de server kunnen ook op een andere manier in de gaten gehouden worden.

11.1.5. Magnolia

http://www.magnolia.info

11.1.6. MMBase

http://www.mmbase.org

11.1.7. OpenCMS

http://www.opencms.org

Ook OpenCMS is een open source CMS en wordt gedistribueerd onder de GNU LGPL licentie.

Het nadeel is dat het maar beperkt ondersteuning biedt voor meertalige inhoud en geen themes heeft. Twee belangrijke criteria voor het CMS.

11.2. Pakket van eisen pilot project

Deze bijlage is in een apart bestand opgenomen. Zie hiervoor *bijlage_pakket_van_eisen.pdf*