"Spaarpost"

een component based ontwerp met Petri netten

van den Borne, Ruud

Award date:
2003

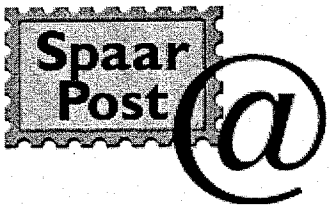
TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

"SpaarPost"

een component based ontwerp met Petri Netten

Afstudeerverslag van
Ruud van den Borne



Opleiding	Technische Informatica Vakgebied Informatiesystemen Technische Universiteit Eindhoven
Afstudeerdocent	prof. dr. K.M. van Hee
Periode	februari – november 2003
Bedrijf	Deloitte
Afdeling	Vakdirectoraat Consultancy
Begeleider	drs. R.A. van der Toorn

Samenvatting

Dit afstudeerrapport is een ontwerpdocument voor het project SpaarPost, enerzijds uitgevoerd namens SpaarPost zelf en Trusted Party Deloitte, anderzijds namens Technische Universiteit Eindhoven.

Het concept van SpaarPost is als volgt:
Op een centrale plek op Internet geeft de consument aan van welke bedrijven en over welke producten of diensten hij regelmatig, via e-mail, informatie wenst te ontvangen. Slechts het achterlaten van een e-mail adres is voldoende. Bedrijven kunnen zich gratis presenteren op de website van Stichting SpaarPost en vervolgens e-mailen naar geïnteresseerde potentiële klanten. Ze voldoen hiermee aan de nieuwe Europese wetgeving en geven blijk op een maatschappelijk verantwoorde manier gebruik te maken van commerciële e-mail.

Snelheid van ontwikkeling, flexibiliteit en correctheid zijn drie sleutelwoorden voor het project. Deze wensen hebben onder anderen geleid tot een keuze om te gaan ontwikkelen met ColdFusion van Macromedia. De basis voor een goede implementatie is een correct ontwerp. Allereerst zijn daarom de requirements vastgesteld in de vorm van informele use-cases. Deze zijn later formeler gemaakt in modellen in de vorm van Component-nets (C-netten), een vorm van Petri Netten. C-netten worden gebruikt in een componentsgewijze ontwikkeling. Het voordeel van deze methode van modelleren is dat er analyse gedaan kan worden op correctheid van de modellen en dat de applicatie een gestructureerde hiërarchie bevat. Door componenten en subcomponenten te gebruiken kan een overzichtelijk model worden gegenereerd op elk niveau. Er zijn een aantal componenten te onderscheiden binnen de applicatie.

In de eerste plaats is er voor de consumenten een interface nodig waarvoor men nieuwsbrieven kan selecteren en gewenste informatie over kan brengen. Vervolgens is er de behoefte voor de bedrijven om op eenvoudige wijze nieuwsbrieven op te stellen en te verzenden. Om dit alles te beheren moet er een component zijn om de database te bewerken, een Content Management Applicatie (CMA). Ten slotte moeten er ook werkelijk nieuwsbrieven verstuurd gaan worden en is er behoefte aan een component die de mailing gaat verzorgen.

Bij het implementeren van deze componenten wordt gebruik gemaakt van de FuseBox-methode, waarbij elke component los van elkaar ontwikkeld wordt, als een zekering (fuse) in een zekeringenkast (FuseBox). In samenwerking met een hostingpartij en webdesigner worden bestaande componenten hergebruikt en geconfigureerd. Een CMA, een tool om nieuwsbrieven te bewerken en een mailingcomponent zijn beschikbaar en resulteren uiteindelijk in een snelle realisatie van de applicatie.

Om het ontwerp compleet te maken is ook nog vermeld hoe de infrastructuur (de fysieke hardware) is ingericht.

Voorwoord

Dit afstudeerrapport sluit een periode af van een driejarige informaticastudie binnen de vakgroep Informatiesystemen aan Technische Universiteit Eindhoven.

Het ontwerp beschreven in dit verslag is in opdracht van Deloitte uitgevoerd, ten behoeve van het Nationaal Initiatief SpaarPost.

Mijn speciale dank gaat uit naar mijn begeleider Robert van der Toorn voor zijn intensieve begeleiding en mijn afstudeerdocent prof. K.M. van Hee voor zijn inspirerende en kritische ideeën. Vanaf begin tot eind ben ik intensief betrokken bij het project, wat mogelijk is gemaakt door initiatiefnemers Ruud Kanters en Lex Pieterse. Ook hiervoor bedankt. Als laatste wil ik Erik-Jan Jaquet danken voor de ondersteuning en de fijne samenwerking tijdens de implementatie.

4 november, 2003
Ruud van den Borne

Inhoudsopgave

1
Inleiding

1.1. Achtergrond

De hoeveelheid junkmail is de afgelopen jaren sterk toegenomen en oefent een significante druk uit op het internet. Het onderwerp staats reeds ter discussie sinds 1975. Ook uit andere perspublicaties blijkt dat er duidelijk iets mis is met het imago van de commerciële e-mail. Dat is jammer, want direct marketing via e-mail kan uitstekend voorzien in de informatiebehoefte van de consument. Feit is dat een aantal bedrijven e-mail een zeer slechte reputatie bezorgt. Zo slecht, dat het Europese Parlement heeft besloten er wat aan te gaan doen, door middel van Richtlijn 2002/58, waarin het bedrijven verboden wordt consumenten, zonder voorafgaande toestemming, e-mail te sturen. Uiterlijk 31 oktober 2003 moeten de EU-lidstaten deze richtlijn in hun wetgeving hebben opgenomen.
De wens van consumenten om via e-mail geïnformeerd te worden blijft, evenals de behoefte bij het bedrijfsleven om e-mail naar prospects te sturen. Dit laatste maakt de nieuwe wetgeving bijna onmogelijk.

Het Nationaal Initiatief "Stichting SpaarPost" onderkent dat deze maatschappelijke ontwikkelingen (wildgroei, irritatie en regelgeving) zorgen voor een nieuwe fase in het commerciële e-mail verkeer.
De kracht van het concept ligt in de eenvoud. Op een centrale plek op Internet geeft de consument aan van welke bedrijven en over welke producten of diensten hij regelmatig, via e-mail, informatie wenst te ontvangen. Daarbij hoeft hij geen lange vragenlijsten met persoonlijke gegevens in te vullen. Slechts het achterlaten van zijn e-mail adres is voldoende. Dit is niet alleen gebruikersvriendelijk, maar komt ook aan de veelgehoorde wens van absolute privacy tegemoet. SpaarPost garandeert immers dat het adres niet wordt doorgegeven aan derden.
Bedrijven kunnen zich gratis presenteren op de website van Stichting SpaarPost en vervolgens e-mailen naar geïnteresseerde potentiële klanten tegen zeer aantrekkelijke tarieven. Ze voldoen hiermee aan de nieuwe Europese wetgeving en geven blijk op een maatschappelijk verantwoorde manier gebruik te maken van commerciële e-mail.

Voor het Nationaal Initiatief "Stichting SpaarPost" is brede maatschappelijke steun. Het Platform Internet voor Alledag heeft het initiatief geadopteerd. Het Platform Internet voor Alledag biedt ondersteuning aan levensvatbare Internet projecten met nationale uitstraling.

Deloitte&Touche, die nu al een grote bijdrage heeft geleverd bij de opzet en inrichting van het Nationaal Initiatief "Stichting SpaarPost", vervult als Trusted Party een belangrijke controlerende rol en garandeert dat SpaarPost zich aan haar privacy toezeggingen houdt.

1.2. Leeswijzer

Dit rapport richt zich op het ontwerp van de SpaarPost applicatie. Het volgende hoofdstuk, de user requirements legt informeel de requirements vast voor het systeem, zoals deze met de opdrachtgevers besproken zijn. Hoofdstuk 3, de business architectuur, is de basis voor het software ontwerp. De eisen worden formeler opgesteld en er worden modellen van de deelcomponenten gemaakt in de vorm van Petri Netten.
In hoofdstuk 4 worden de hoofdcomponenten onderscheiden en alle subcomponenten daaraan toegewezen, zodat uiteindelijk per hoofdcomponent één globaal model wordt verkregen. Hoofdstuk 5 geeft weer hoe de logische componenten tot software componenten verwerkt zijn en wat de structuur is van de applicatie.
Ten slotte geeft het laatste hoofdstuk een kort overzicht van de infrastructuur, om het ontwerp compleet te maken.

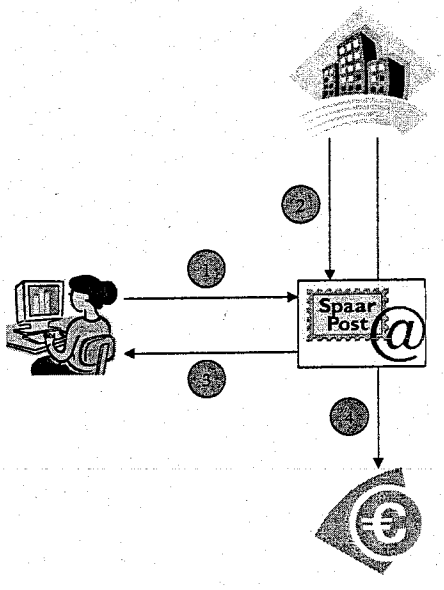
2
User requirements

2.1. De business case

Most e-business projects should start with the design of the way of doing business: the business model. Essentially, it provides the design rationale for e-commerce systems from a business point of view. In our view, the main goal of a business model is to answer the question: "who is offering what to whom and expects what in return". [AGV2000]

Dit citaat geeft kort maar krachtig weer waar een business model in zou moeten voorzien. Voor het SpaarPost concept geldt dit natuurlijk ook. Aan de hand van Figuur 2.1 wordt dit concept hieronder beknopt beschreven.

Stichting SpaarPost stelt de consument, op eenvoudige wijze, in staat om op haar internetsite zijn informatievoorkeuren vast te leggen. Slechts het intikken van een e-mailadres en het aanklikken van bedrijven is voldoende (actie 1 in figuur). Hiermee geeft de consument aan van welke bedrijven hij regelmatig, via e-mail, informatie over producten of diensten wenst te ontvangen.



Figuur 2.1

Nadat een bedrijf op de site is geplaatst, kan zij alle verdere handelingen om een e-mail te versturen zelf verrichten. Hierbij maakt Stichting SpaarPost gebruik van "Straight Through Processing". Dit werkt als volgt:

Een bedrijf dat een boodschap wil verzenden naar geïnteresseerde potentiële klanten, laadt haar (volledig in eigen beheer) opgemaakte e-mail in de speciaal daarvoor ontwikkelde e-mail manager van Stichting SpaarPost (actie 2). Vervolgens zendt het bedrijf deze e-mail naar Stichting SpaarPost. Zij ontvangt dan van Stichting SpaarPost een proef e-mail, inclusief een

opgave van het aantal consumenten dat heeft aangegeven interesse te hebben in de producten en diensten van het desbetreffende bedrijf. Aan de hand hiervan kan het bedrijf zelf bepalen of zij de e-mail daadwerkelijk wil gaan versturen. Indien zij besluit de e-mail te verzenden dan gaat deze nogmaals naar Stichting SpaarPost. Stichting SpaarPost koppelt deze boodschap dan aan de door haar beheerde e-mailadressen en stuurt deze vervolgens door naar de consument (actie 3). Wanneer een mailingactie is voltooid worden de resultaten geregistreerd en worden de bedrijven gefactureerd voor het aantal verstuurde e-mails. Alle bovenbeschreven handelingen die uitgevoerd worden bij Stichting SpaarPost zijn volledig geautomatiseerd en dit hele proces kan in enkele seconden worden uitgevoerd.

Voor iedere e-mail verzonden via Stichting SpaarPost, wordt uiteindelijk een geldbedrag toegevoegd aan het Stichting SpaarPost Duurzaamheidfonds (actie 4). Indien de ontvanger van de e-mailboodschap geïnteresseerd is in het aanbod kan hij rechtstreeks, zonder tussenkomst van Stichting SpaarPost, doorklikken naar de site van de afzender.

2.2. Stakeholders

Er zijn vier stakeholders die een aandeel hebben in het systeem.

1. De consumenten
   Deze kunnen de site bekijken en via login de gewenste nieuwsbrieven selecteren en accountinformatie bewerken
2. De bedrijven
   Om nieuwsbrieven op te stellen, te beheren en beschikbaar te maken
3. SpaarPost
   Om de gehele applicatie van user interface tot database
4. De founding partners en fondsen
   Zij spelen geen actieve rol in het systeem. Founding partners investeren in SpaarPost en SpaarPost investeert in de fondsen.

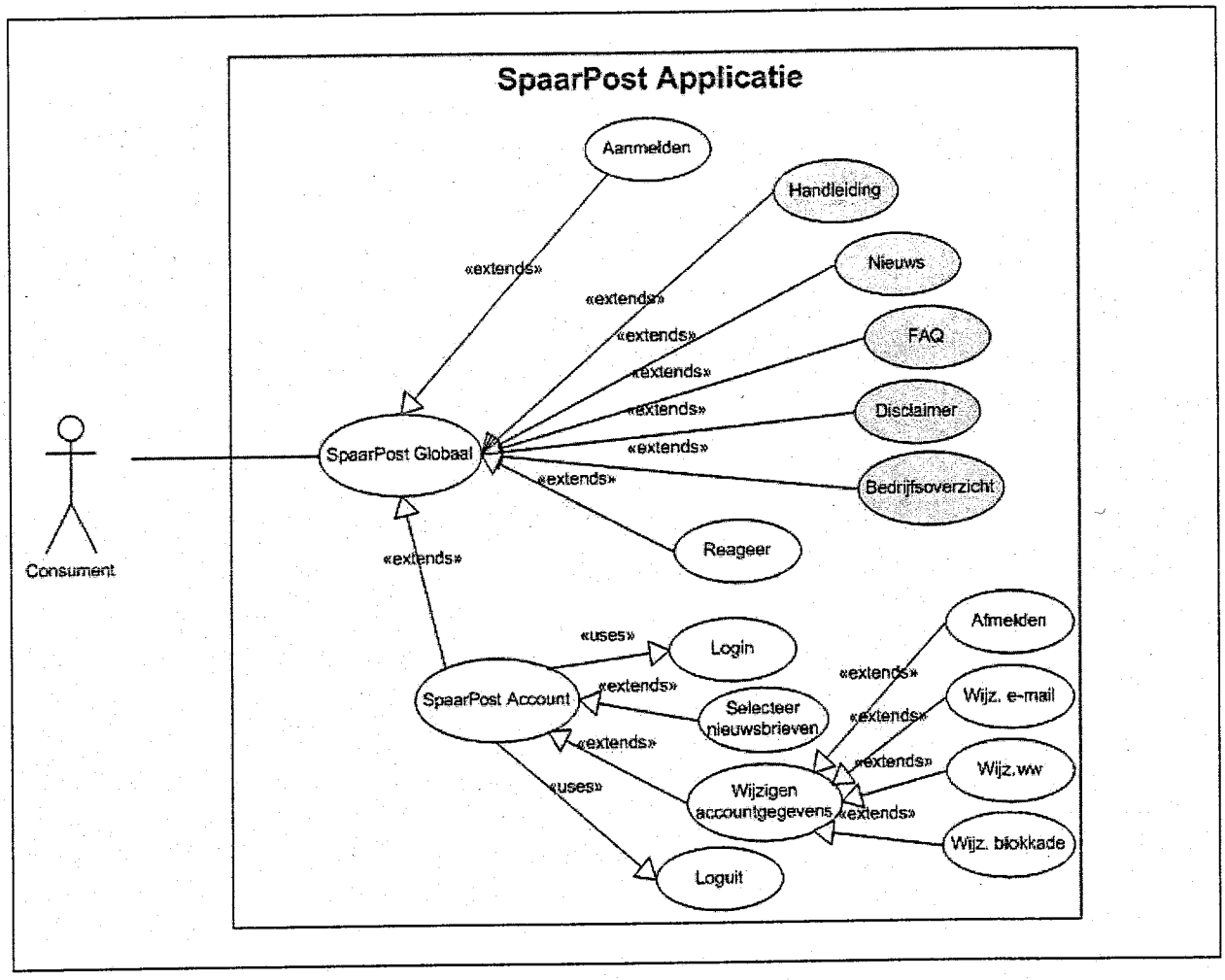
2.3. Functionele eisen: Use-cases

In de eerste fase van het ontwerp worden de requirements vastgelegd, de zgn. requirements-analysefase. Hierin worden de requirements in modellen vastgelegd. Bij het modelleren van een systeem is het belangrijk om de functionaliteit in kaart te brengen zoals deze wordt gezien door de ogen van de gebruikers. Dit kan worden gemodelleerd door middel van use-cases. Allereerst wordt de context van het systeem in kaart gebracht door middel van een UML ontwerp. Een requirements document is gemaakt van de globale functionaliteit. De use-cases worden nader toegelicht door middel van een tekstuele omschrijving. Om het software ontwerp visueel te maken is bij het ontwerp gebruik gemaakt van UML-views met als basis de use-cases voor de functionaliteit; een use-case beschrijft een compleet stuk functionaliteit dat een systeem aanbiedt aan een gebruiker en dat een voor de gebruiker observeerbaar resultaat oplevert. Allereerst beginnen we het ontwerp met een informele beschrijving van de use-cases. Dit doen we met behulp van een use-case diagram en bijbehorende beschrijvingen. Elke eerder gedefinieerde stakeholder uitgezonderd de laatste is eveneens een actor in het systeem, dat wil zeggen dat deze de use-case uitvoert. Een actor communiceert met een systeem door het sturen of ontvangen van berichten of informatie en kan dus zowel een mens als een ander systeem representeren. Voor elke actor stellen we een use-case diagram op. Dit diagram is een uitgangspunt voor een meer formele beschrijving van het gedrag van het de

applicatie. Hier komen we later op terug. Het feit dat een bepaalde actor deelneemt in een bepaalde use-case wordt weergegeven met een communicates-relatie, een lijn tussen actor en use-case.

Actor 1: Consument

Figuur 2.2 geeft het use-case diagram voor de actor *consument* weer.



Figuur 2.2 Use-case diagram consumenten

Deze figuur laat zien dat de SpaarPost applicatie een aantal use-cases bevat. De consument communiceert met de omvattende use-case *SpaarPost Globaal,* welke weer andere use-cases gebruikt: *Aanmelden, Handleiding, Nieuws, FAQ, Disclaimer, Bedrijfsoverzicht, SpaarPost Account* en *Reageer.*

Aanmelden
De potentiële gebruiker krijgt de mogelijkheid een e-*mail adres* en een *wachtwoord* in te geven dit is mogelijk nadat de gebruiker op het hoofdscherm van de website de optie *<Aanmelden>* heeft gekozen. De gebruiker krijgt de mogelijkheid zijn e-mail adres in te voeren en zelf een wachtwoord te kiezen. Hierna krijgt de gebruiker een e-mail in zijn inbox. Door deze te bevestigen wordt de gebruiker actief lid.

Een aantal use-cases waarvoor de gebruiker niet ingelogd hoeft te zijn, zijn de scenario's waarbij de gebruiker statische pagina's gaat bekijken. Deze betreffen *Handleiding, Nieuws, Frequently Asked Questions (FAQ), Disclaimer* en een *Bedrijfsoverzicht.*

Handleiding	-	Een handleiding voor de consumentensite.
Nieuws	-	Bevat het laatste nieuws van SpaarPost en houdt een archief bij.
FAQ	-	Bevat vragen die vaak gesteld worden en hun antwoorden.
Disclaimer	-	Noodzakelijke disclaimer.
Bedrijfsoverzicht	-	Geeft een overzicht van alle bedrijven die lid zijn van SpaarPost en waarvan men dus nieuwsbrieven kan ontvangen.

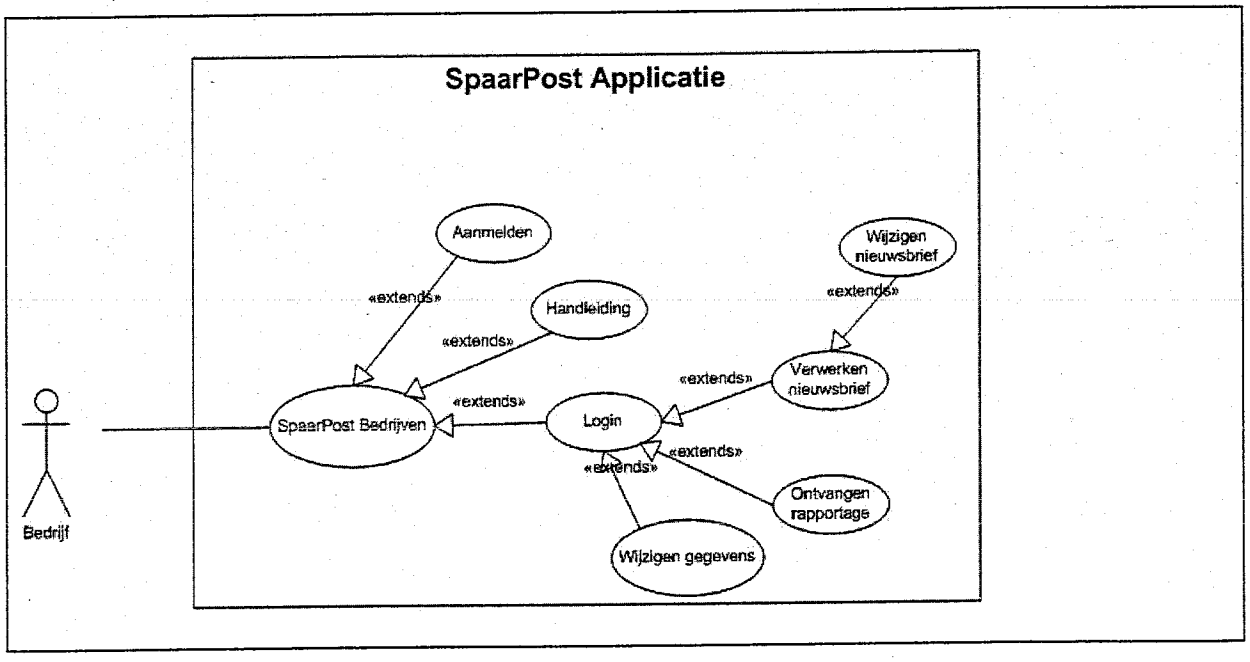
Reageer
Dit is een functionaliteit die door middel van een link benaderbaar is en voor iedereen toegankelijk (dus ook zonder ingelogd te zijn). Alle opmerkingen en vragen kunnen hier verstuurd worden. Dit versturen gaat via een online formulier.

SpaarPost Account
Deze use-case biedt een interface voor alle informatie die beschikbaar is wanneer is ingelogd. Daarom is er de <<*uses*>> relatie met *login* en *loguit*. Als een gebruiker ingelogd is heeft hij de mogelijkheid een selectie te maken uit nieuwsbrieven (de *Selecteer Nieuwsbrieven* use-case) en de mogelijkheid zijn eigen profiel te wijzigen (*Wijzigen Accountgegevens*).

Met deze use-cases is het informele gedeelte voor de actor *consument* compleet. De volgende actor in de applicatie is het *bedrijf.*

Actor 2: Het bedrijf

Het bedrijf zal de nieuwsbrieven geheel zelf moeten gaan verwerken, zonder tussenkomst van SpaarPost. In Figuur 2.3 is het use-case diagram te vinden van deze actor.



Figuur 2.3 Use-case diagram Bedrijven

SpaarPost Bedrijven
Dit is de algemene use-case die de interface aanbiedt voor de bedrijvensite. Via deze use-case kan een bedrijf zich *aanmelden* via een online formulier. SpaarPost neemt dan zelf contact op met het bedrijf en bepaalt of deze toegevoegd kan worden en onder welke voorwaarden. Door middel van een *login* die SpaarPost verstrekt kan het bedrijf inloggen op de site. Eenmaal ingelogd kunnen *nieuwsbrieven* worden *verwerkt* en kunnen *rapportages* worden bekeken van eerder verzonden mailings.
Ook kunnen dan statistieken bekeken worden van reeds verzonden nieuwsbrieven (*Ontvangen rapportage*). Hierbij wordt vooral gedacht aan hoeveelheden (verzonden en "gebounced").
Om het geheel compleet te maken is ook op dit gedeelte van de site een *Handleiding* beschikbaar.

Actor 3: SpaarPost

De derde en laatste actor is *SpaarPost* zelf. Deze heeft de gehele site en database in beheer. Kort samengevat moet SpaarPost beschikken over een CMA (Content Management Applicatie) waarin teksten op de site beheerd kunnen worden en de database aangepast kan worden op een gebruiksvriendelijke wijze. Omdat hiervoor een standaard component aanwezig is die zich al bewezen heeft, is het niet zinvol om ook hiervoor use-cases op te stellen.

2.4. Niet-functionele eisen

Behalve de functionele eisen zijn er uiteraard ook de niet-functionele eisen waar de applicatie aan moet voldoen. Het realiseren van deze niet-functionele eisen is vooral afhankelijk van de infrastructuur en ontwikkelomgeving. Deze keuzes worden later in dit verslag gemotiveerd. Allereerst een kort overzicht van de niet-functionele eisen:

Voor het SpaarPost project betekent *schaalbaarheid* een grote belasting aankunnen en snel en probleemloos kunnen groeien. Een snelle groei van het aantal gebruikers in combinatie met intensief gebruik van de applicatie kan voor de web- of mailservers een grote belasting tot gevolg hebben. De computercapaciteit moet in dit geval uitgebreid kunnen worden. Door de applicatie te hosten bij een professionele organisatie met ervaring op dit gebied is deze schaalbaarheid geen probleem.

Ook *performance* is een belangrijk aspect voor de applicatie. Lange wachttijden zijn onacceptabel. Enerzijds is deze performance afhankelijk van de hardware, anderzijds kan ook de programmacode hier een belangrijke rol bij spelen. Hardware is om dezelfde reden als schaalbaarheid geen discussiepunt. De hostingorganisatie beschikt over moderne apparatuur en technieken voor caching indien nodig. Wat betreft de software zijn vooral de queries hier een knelpunt. Om deze reden is hier dan ook veel aandacht aan besteed. Veel tijdswinst is te halen door queries efficiënt op te stellen. Twee instructies die dezelfde antwoorden geven, maar anders geformuleerd zijn, kunnen zeer uiteenlopende performances hebben. Door bijvoorbeeld een join te vervangen door een subquery (of andersom), kan de performance met een factor honderd verbeterd worden.
Een ander punt waar queries aandacht op verdienen is de *consistentie*. Door het aanbrengen van een duidelijke structuur met try-catch commando's in de queries welke schrijven in de database wordt de consistentie in de database gehandhaafd. Soms moeten meerdere queries achtereenvolgend uitgevoerd worden. Als er één van deze queries fout gaat, moet een algehele

rollback plaatsvinden. Een commit vindt pas plaats als alles in het try-block succesvol doorlopen is.

Ook moet de applicatie *flexibel* worden opgebouwd zodat het ontwikkelen en inzetten van nieuwe componenten goed wordt ondersteund. De keuze van programmatuur is daarbij van groot belang. Tijdens het ontwikkeltraject van SpaarPost is uiteindelijk de keuze gevallen op ontwikkeling met Coldfusion van Macromedia. Deze omgeving maakt het uitstekend mogelijk om componenten eenvoudig in te passen.

Ten slotte is een van de aantrekkelijke aspecten van het SpaarPost dat e-mailadressen niet worden vrijgegeven aan derden. Volledige garantie dat internetverkeer niet kan worden onderschept en ontcijferd is nooit te geven, echter een aantal maatregelen zijn getroffen om database en applicatie te beveiligen. De servers zijn goed afgeschermd voor aanvallen van buitenaf en firewalls zorgen dat alleen vertrouwde informatie ontvangen en verstuurd wordt. Autorisatie en authenticatie worden gebruikt om toegang tot database of server te regelen. Eventueel is het mogelijk om alle verkeer over het Internet te versleutelen door middel van Secure Socket Layer (SSL). Meer over de infrastructuur en beveiliging is te vinden in het gelijknamige hoofdstuk (6).

3
Business architectuur

3.1. Use-case modellering

In het vorige hoofdstuk hebben we een informele opsomming gegeven van de benodigde use-cases. Om een stap dichter bij de ontwikkeling van software te komen gaan we de use-cases formeler vastleggen. Dit helpt in een vroegtijdig stadium eventuele fouten en inconsistentie te constateren, wat uiteindelijk leidt tot een kostenbesparing. Immers, fouten die laat in de ontwikkeling nog geconstateerd worden zijn vele malen kostbaarder dan wanneer deze tijdig gevonden en hersteld kunnen worden.
Allereerst gebruiken we een formelere tekstuele omschrijving, met naam, doel en precieze omschrijving, gevolgd door pre- en postcondities van de use-case. Sommige use-cases bevatten nog een bepaalde controle, welke nodig is om de use-case succesvol te doorlopen.

Consumenten

Use-case C.1

Naam: SpaarPost Globaal
Doel: Dient als interface naar de applicatie voor de consument en coördineert alle functionaliteiten van SpaarPost.
Beschrijving:

1. Start sessie
2. Start homepage
3. Keuze
    a) Als Aanmelden wordt gekozen, start use-case [Aanmelden]
    b) Als SpaarPost Account wordt gekozen, start use-case [SpaarPost Account]
    c) Als Handleiding wordt gekozen, start use-case [Handleiding]
    d) Als Nieuws wordt gekozen, start use-case [Nieuws]
    e) Als FAQ wordt gekozen, start use-case [FAQ]
    f) Als Disclaimer wordt gekozen, start use-case [Disclaimer]
    g) Als Bedrijfsoverzicht wordt gekozen, start use-case [Bedrijfsoverzicht]
    h) Als Reageer wordt gekozen, start use-case [Reageer]
4. Einde sessie

Preconditie: -
Postconditie: De consument bevindt zich op de SpaarPost website

Use-case C.2

Naam: Aanmelden
Doel: Het registreren bij SpaarPost
Beschrijving:

1. Invoeren registratie
2. Als de registratie geldig is:
    a) Ontvangen bevestigingsmail
    b) Bevestigen bevestigingsmail
    c) Bevestiging in browser (als bevestiging geldig is, anders foutmelding)
3. Als de registratie ongeldig is:
    Foutmelding in browser

Controles:
- E-mailadres uniek
- E-mailadres correct
- Wachtwoord correct

Preconditie: De consument is niet ingelogd en bevindt zich op de site
Postconditie: De consument is aangemeld of is niet aangemeld

Use-case C.3 t/m C.7

Deze use-cases zijn triviaal (oproepen van statische schermen)

Use-case C.8

Naam: Reageer
Doel: Het versturen van een formulier naar SpaarPost met een opmerking
Beschrijving:

1. Invullen e-mailadres
2. Invullen opmerking
3. Versturen formulier

Controles:
- E-mailadres geldig
- Opmerking ingevuld

Preconditie: De consument bevindt zich op de site
Postconditie: De consument bevindt zich op de site

Use-case C.9

Zoals uit de use-case *SpaarPost Globaal* blijkt, is *SpaarPost Account* een onderdeel van de globale use-case en wordt gestart wanneer de consument inlogt.

Naam: SpaarPost Account

Doel: Dient als interface naar de applicatie voor de consument en coördineert de functionaliteiten van SpaarPost betreffende accountinformatie. Hiervoor dient de consument in te loggen
Beschrijving:

1. Start accountsessie
2. Start use-case [Login]
3. Keuze tussen use-cases totdat uitgelogd is:
    a) Als Selecteer nieuwsbrieven wordt gekozen, start use-case [Selecteer nieuwsbrieven]
    b) Als Wijz. accountgegevens wordt gekozen, start use-case [Wijz. accountgegevens]
4. Start use-case [Loguit]
5. Einde accountsessie

Preconditie: De consument bevindt zich op de site maar is niet ingelogd
Postconditie: De consument bevindt zich op de site maar is niet (meer) ingelogd

Use-case C.10

Naam: Login
Doel: Toegang verkrijgen tot het persoonlijke gedeelte van SpaarPost
Beschrijving:

1. Login met e-mailadres en wachtwoord
2. Als login geldig: Persoonlijke startpagina op scherm
3. Als login ongeldig: Foutmelding op scherm

Controles:
- E-mailadres bestaat
- Wachtwoord geldig

Preconditie: De consument is niet ingelogd
Postconditie: De consument is ingelogd of niet ingelogd

Use-case C.11

Naam: Selecteer Nieuwsbrieven
Doel: Meest relevante onderdeel van SpaarPost, het aan- (en uitzetten) van bepaalde nieuwsbrieven.
Beschrijving:

Aanzetten:

1. Selecteer een subcategorie uit een overzicht met hoofd- en subcategorieën
2. Selecteer een nieuwsbrief

Uitzetten:

Deselecteer de nieuwsbrief in je selectie

Preconditie: De consument is ingelogd
Postconditie: De consument is ingelogd en heeft zijn selectie aangepast

Opmerking: Op dit moment wordt na elke (de)selectie deze selectie bewaard. Mocht dit een grote belasting voor het systeem blijken kan op een later tijdstip een andere techniek gebruikt worden, waarbij de nieuwsbrieven allereerst geselecteerd worden en vervolgens een bevestiging gegeven moet worden voordat de selectie bewaard wordt.

Use-case C.12

Naam: Wijzigen accountgegevens
Doel: Deze use-case biedt een interface voor het wijzigen van alle persoonlijke gegevens van de consument
Beschrijving:

1. Keuze tussen use-cases:
    a. Als Wijz. e-mail wordt gekozen, start use-case [Wijz. e-mail]
    b. Als Wijz. ww wordt gekozen, start use-case [Wijz. ww]
    c. Als Wijz. blokkade Account wordt gekozen, start use-case [Wijz. blokkade]
    d. Als Afmelden wordt gekozen, start use-case [Afmelden]
2. Start scherm wijzigen accountgegevens en selecteer te wijzigen optie.

Preconditie: De consument is ingelogd
Postconditie: De consument is ingelogd en bevindt zich op de juiste locatie om zijn gegevens te wijzigen

Use-case C.13

Naam: Loguit
Doel: Het persoonlijke gedeelte van de SpaarPost site te verlaten
Beschrijving:

1. Kies uitloggen op het scherm
2. Bevestiging van uitloggen op het scherm

Preconditie: De consument is ingelogd
Postconditie: De consument is uitgelogd

Use-case C.14

Naam: Wijzigen e-mail
Doel: Het wijzigen van het e-mailadres
Beschrijving:

1. Invullen nieuw e-mailadres en wachtwoord
2. Als ww geldig:
    a. Ontvangen bevestigingsmail
    b. Bevestigen wijziging
    c. Bevestiging of foutmelding op scherm
3. Als ww ongeldig: Foutmelding op scherm

Controles:
- zie aanmelden
- Wachtwoord correct

Preconditie: De consument is ingelogd
Postconditie: De consument heeft zijn e-mailadres bij SpaarPost gewijzigd of dit is mislukt.

Use-case C.15

Naam: Wijzigen ww
Doel: Het wijzigen van het wachtwoord
Beschrijving:

1. Invullen 2x nieuwe wachtwoord en 1x oude wachtwoord
2. Als wachtwoorden geldig:
    a. Bevestiging op scherm
    b. Ontvangen e-mail met nieuwe wachtwoord
3. Als ww ongeldig: Foutmelding op scherm

Controles:
- Wachtwoorden correct

Preconditie: De consument is ingelogd
Postconditie: De consument heeft zijn wachtwoord bij SpaarPost gewijzigd of dit is mislukt.

Use-case C.16

Naam: Wijzigen blokkade
Doel: Het wijzigen of invoeren van een blokkade om een periode in te stellen waarin de consument geen e-mail wenst te ontvangen
Beschrijving:

1. Keuze:
    a. Selecteer begindatum
    b. Selecteer einddatum
2. Als data geldig (einddatum ≥ begindatum en einddatum ≥ vandaag):
    Blokkade op scherm

3. Als data ongeldig: Foutmelding op scherm

Controles:
- Begindatum ≤ Einddatum en Einddatum na vandaag

Preconditie: De consument is ingelogd
Postconditie: De consument heeft zijn blokkade bij SpaarPost gewijzigd of dit is mislukt.

Use-case C.17

Naam: Accepteren e-mail
Doel: Het bevestigen van een wijziging in de database ter controle
Beschrijving: Triviaal

Use-case C.18

Use-case *Afmelden* wordt gestart wanneer de consument kiest om zich af te melden. Door het e-mailadres en wachtwoord juist in te vullen kan een consument zich afmelden.

Naam: Afmelden
Doel: Het registreren bij SpaarPost
Beschrijving:

1. Invoeren wachtwoord
2. Bevestiging van afmelding op scherm

Preconditie: De consument bevindt zich op de site
Postconditie: De consument is afgemeld

Bedrijven

Use-case B.1:

Naam: Aanmelden
Doel: Het sturen van een formulier met het verzoek om het bedrijf aan te melden
Beschrijving:

1. Invullen bedrijfsgegevens
2. Versturen aanmelding

Controles:
- Alle velden ingevuld

Preconditie: Het bedrijf is nog niet aangemeld
Postconditie: De consument heeft zijn blokkade bij SpaarPost gewijzigd of dit is mislukt.

Use-case B.2

Naam: Handleiding
Doel: Statische weergave van een handleiding hoe nieuwsbrieven op te stellen
Beschrijving: Triviaal

Preconditie: De gebruiker bevindt zich op de bedrijvensite
Postconditie: De gebruiker beschikt over een handleiding voor het opstellen van nieuwsbrieven

Use-case B.3

Naam: Login
Zie Use-case C.11

Use-case B.4

Naam: Verwerken nieuwsbrief
Doel: Van opstellen nieuwsbrief tot aan de werkelijke verzending
Beschrijving:

1. Selecteren nieuwsbrieftype

2. Workflow:
    a. Creëer nieuwe draft
    b. Verstuur proef
    c. Ontvangen proef
    d. Goedkeuren proef of: Afkeuren en terug naar a.
    e. Verzenden nieuwsbrief

Opmerking: Na het selecteren van een nieuwsbrieftype (1), kan de gebruiker een keuze maken uit een nieuwsbrief in de workflow, in elke status tot de nieuwsbrief verzonden is.

Controles:
- Controleren op rechten voor goedkeuren of verzenden

Preconditie: De gebruiker is ingelogd op de bedrijvensite
Postconditie: De gebruiker heeft een actie uitgevoerd op een nieuwsbrief, ergens in de workflow

Use-case B.5

Naam: Ontvangen rapportage
Doel: Het resultaat bekijken van een verzonden nieuwsbrief-mailing
Beschrijving:

1. Selecteren nieuwsbrieftype

2. Selecteren verzonden nieuwsbrief

3. Selecteren rapportage

Controles:
- Controleren op rechten voor bekijken rapportage

Preconditie: De gebruiker is ingelogd op de bedrijvensite
Postconditie: De gebruiker ziet de rapportage van een verzonden mailing

Use-case B.6

Naam: Wijzigen gegevens
Doel: Wijzigen e-mailadres of wachtwoord
Beschrijving: Zie C.13, C.15, C.16

Tot op dit punt hebben we de requirements vastgelegd in UML-views en beschrijvingen. Het voordeel van UML is dat het krachtig genoeg is om software systemen visueel en efficiënt te specificeren. Via UML diagrammen kunnen de user requirements, de statische eigenschappen en het dynamische gedrag eenvoudig beschreven worden. Een nadeel is echter dat de mogelijkheid ontbreekt om wiskundige technieken op de modellen toe te passen voor validatie of analyse van het systeem.

UML vs. Petri-Net
Allereerst voorzien Petri Netten in een grafische techniek om systemen te beschrijven, die eenvoudig te begrijpen is en veel lijkt op de zgn. state-transition diagrams. Verder kunnen eigenschappen als parallellisme, concurrency en synchronisatie eenvoudig gemodelleerd worden met behulp van Petri Netten. Bovendien zijn er vele technieken beschikbaar om Petri Netten te analyseren (bijvoorbeeld ExSpect). Tenslotte zijn Petri Netten uit te breiden, met kleur om objectattributen en methoden te modelleren, met tijd om het gedrag van objecten te kwantificeren en met hiërarchie om het net te verdelen in componenten en zo structuur te geven aan modellen. [SS2000]
Objecten reageren op events die intern gecreëerd worden of op events die extern gecreëerd worden in andere objecten. De collaboration diagrammen in UML geven de event flows tussen de objecten weer, terwijl statechart diagrammen de levensduur van een object weergeven. Het gedrag van objecten is event-based in relatie met andere objecten en kan als CPN gemodelleerd worden.Een aantal views die grafisch ondersteund worden door UML en in dit ontwerp gebruikt zullen gaan worden zijn de component view, het sequence diagram en het class diagram. Activiteiten diagrammen en state diagrammen worden niet gebruikt maar in plaats daarvan worden CPN's gebruikt.[GM1998]

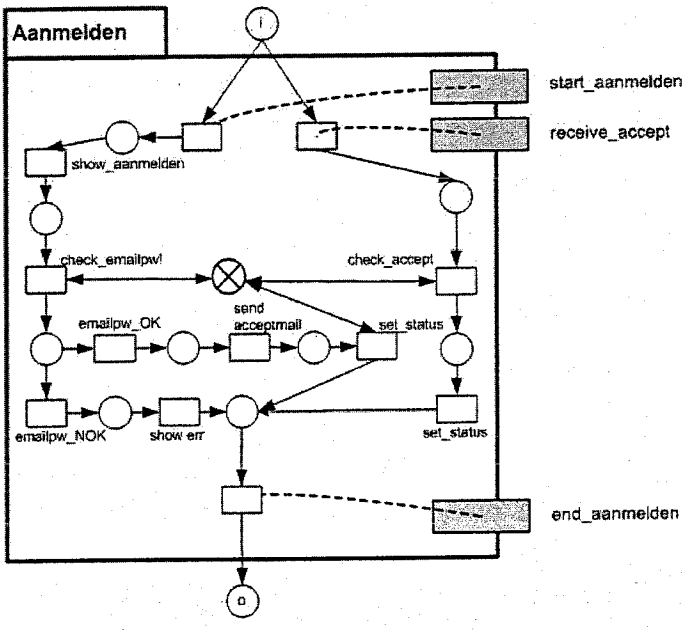
3.2. Workflowmodellering met C-netten

In dit onderdeel beschrijven we de use-cases in zogenaamde Component-nets (C-netten). Hierbij zien we elke use-case als een kleine component met één inputplaats en één outputplaats. C-netten zijn tevens WorkFlow-nets (WF-netten). [AHT2000]
Alle C-netten zijn voor dit ontwerp gesloten, minimale componenten (ook wel atomair genoemd), wat wil zeggen dat ze verder niet meer opgedeeld worden in kleinere componenten. Een uitzondering is het C-net voor het wijzigen van accountdata en die voor het verwerken van nieuwsbrieven (B.4). Referentie naar andere componenten binnen de bestaande componenten kan met behulp van zogenaamde "component placeholders". Elke component placeholder (CP) beschrijft de functionaliteit van een component die gebruikt wordt binnen een andere component. Elke transitie binnen de component wordt gelabeld. Deze labels hebben vaak contact met labels buiten de component of met de database. Ze dienen in feite als communicatiemiddel met de buitenwereld voor die component.

Voor het modelleren van elke use-case zijn C-netten opgesteld. Deze beschrijven in tegenstelling met wat de tekstuele beschrijving van de use-case doet vermoeden, niet de functionaliteit gezien vanuit de actor, echter de functionaliteit gezien vanuit de applicatie zelf. Zo bevat elke use-case een model dat een groot raakvlak met de uiteindelijke implementatie kent. De netten bevatten de acties die het systeem ook werkelijk uit zal moeten gaan voeren. Veel transities communiceren met de database of betreffen user interface-acties, omdat dit vaak voorkomende acties zijn binnen een dynamische webapplicatie. Het aanroepen van andere componenten gaat via de eerder genoemde labels. Verder zijn de C-netten uitgebreid met datastores om interactie met de database aan te tonen. Elke datastore geeft daarbij een onderdeel van de database aan.

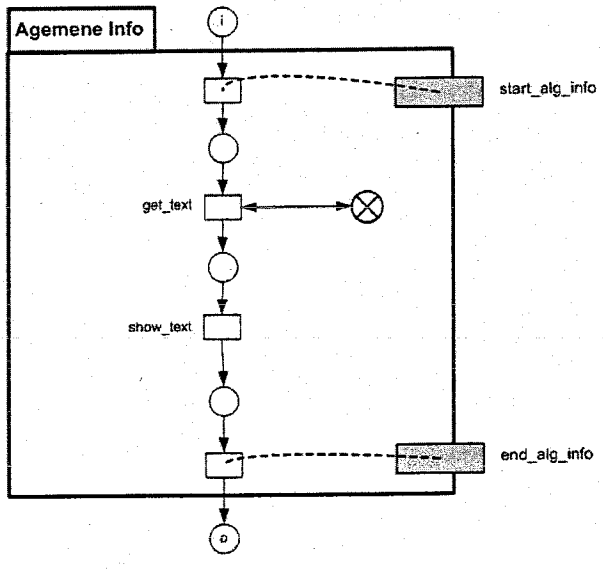
C.1 SpaarPost Globaal
Triviaal

C.2 Aanmelden



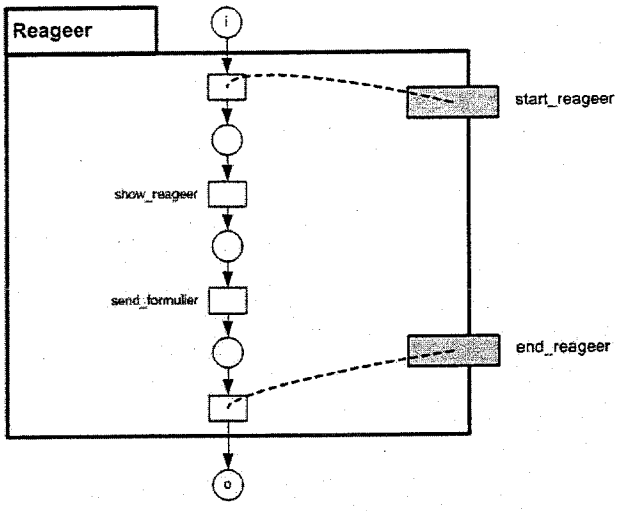
Use-case aanmelden bestaat kan op twee manieren aangeroepen worden. Enerzijds via de site, door een e-mailadres en wachtwoord in te voeren, anderzijds door het bevestigen van een accept-mail.

C.3 t/m C.7



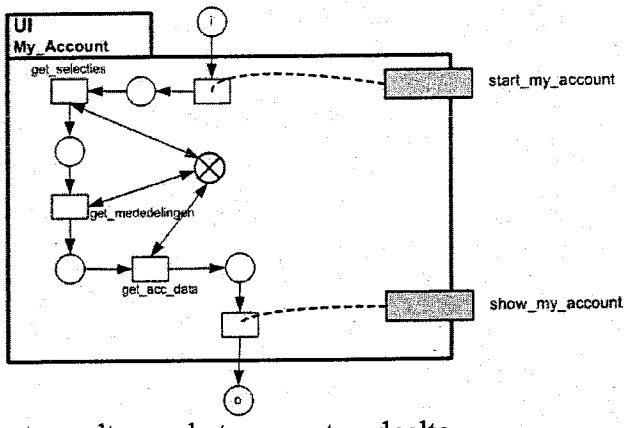
Het ophalen van teksten uit de database om de statische pagina te kunnen laten zien.

C.8 Reageer



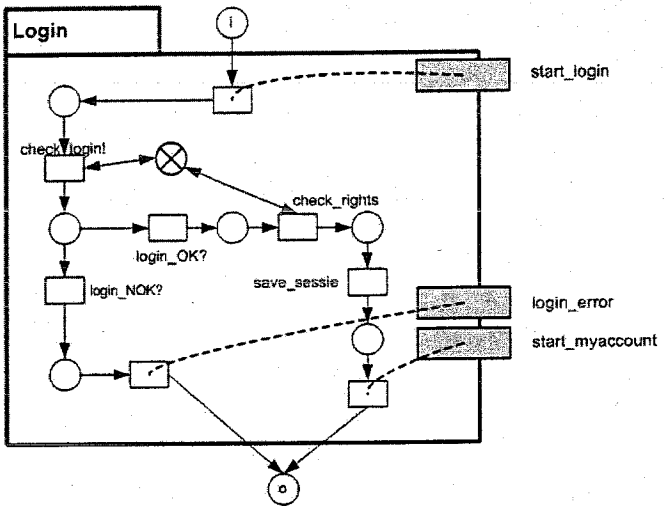
Het laten zien en verzenden van het reactieformulier.

C.9 SpaarPost Account



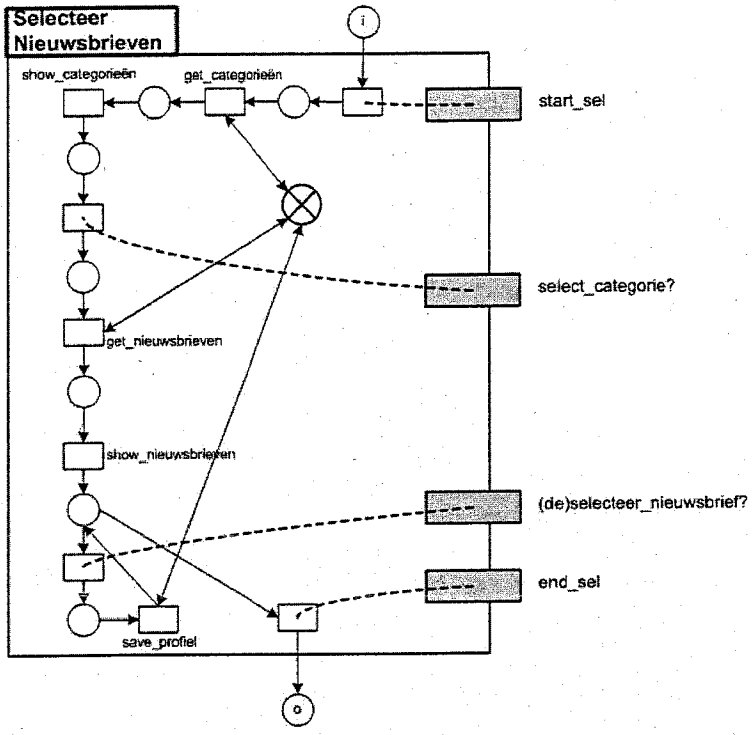
User interface die gestart wordt voor het account-gedeelte

C.10 Login



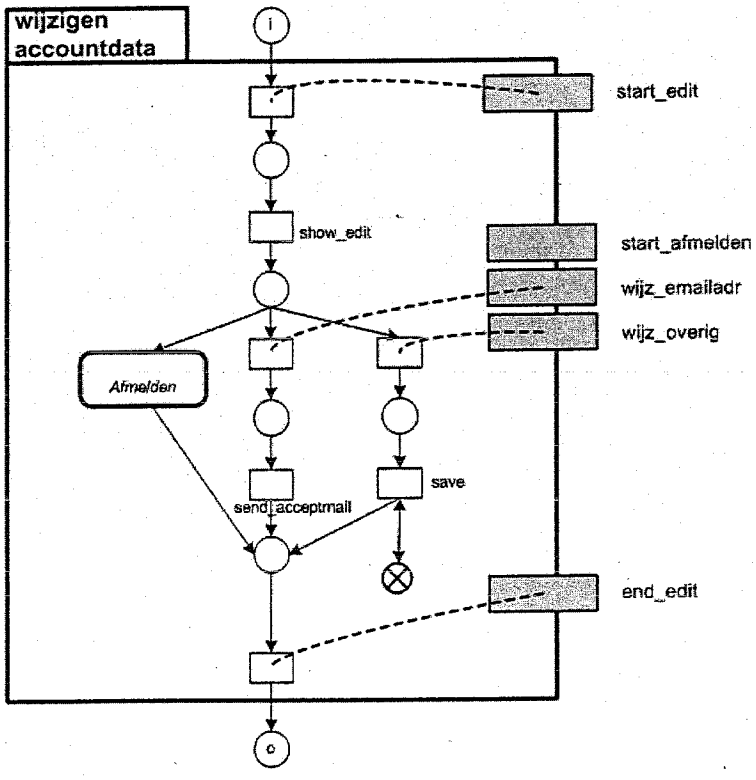
Het controleren van de login met de database.

C.11 Selecteer Nieuwsbrieven



Selectie van achtereenvolgens de categorie en de nieuwsbrief

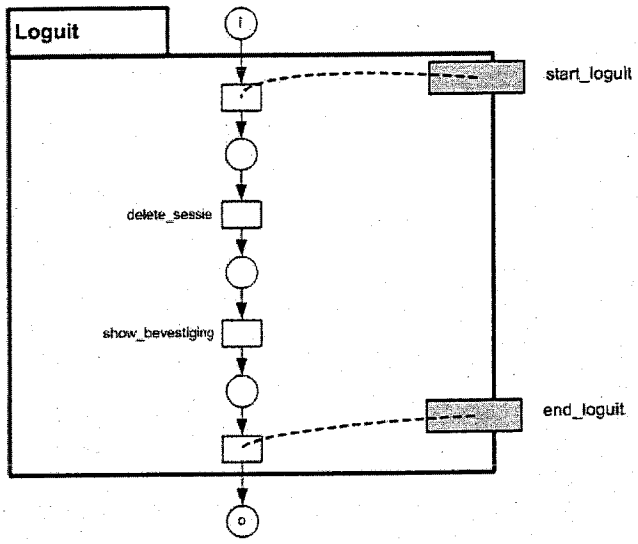
C.12 Wijzigen accountgegevens



Figuur 3.1

Het wijzigen van de persoonlijke accountgegevens. Bij het wijzigen van een e-mailadres wordt een e-mail gestuurd ter controle.

C.13 Loguit



C.14 Wijzigen e-mail

Zie wijzigen accountgegevens

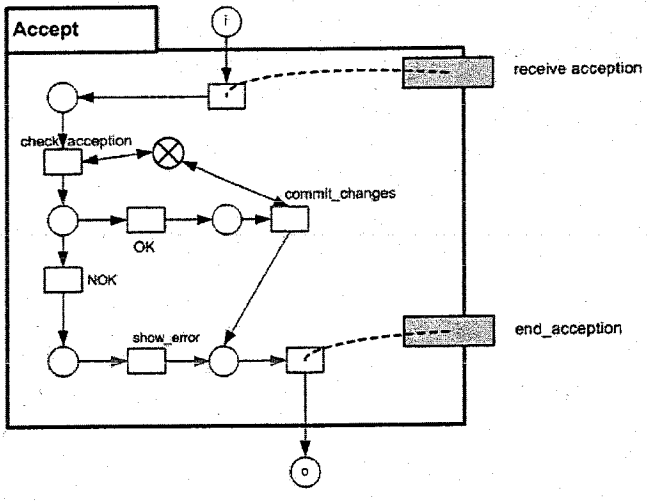
C.15 Wijzigen ww

Zie wijzigen accountgegevens

C.16 Wijzigen blokkade

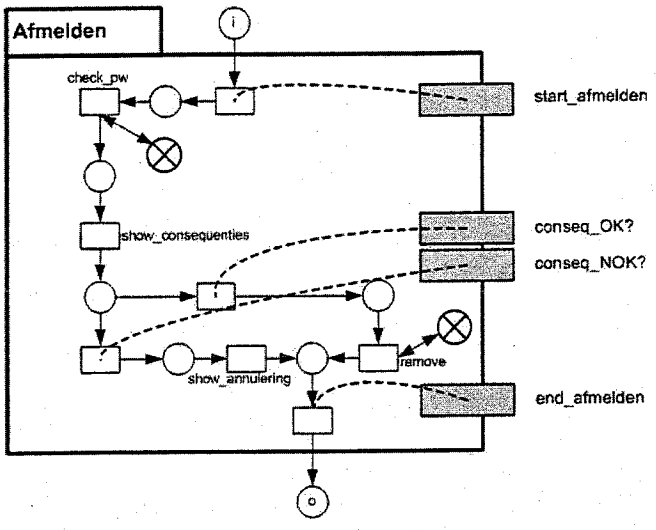
Zie wijzigen accountgegevens

C.17 Accepteren e-mail

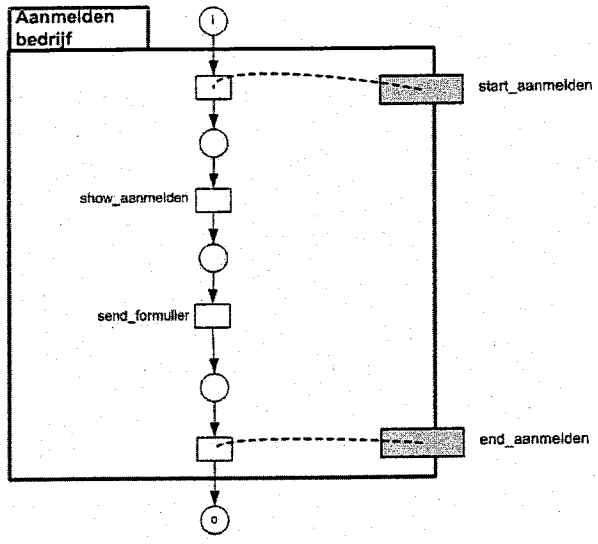


Voor sommige wijzigingen moet een e-mail worden bevestigd ter controle.

C.18 Afmelden



B.1 Aanmelden



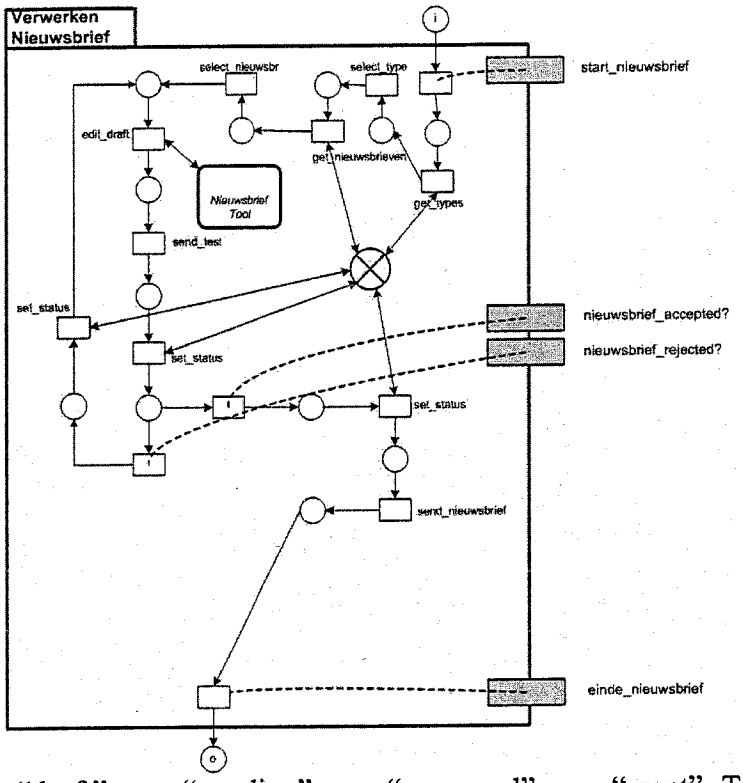
B.2 Handleiding

Triviaal (oproepen van statische schermen)

B.3 Login

Zie C.11

B.4 Verwerken nieuwsbrief



Een workflow van "draft" naar "pending" naar "approved" naar " sent". Tijdens draft-status kan de nieuwsbrief bewerkt worden. Dit gebeurt in de component "Nieuwsbrief Tool". Dit is dan ook de enige component die niet als atomair wordt beschouwd.

B.5 Ontvangen rapportage

Triviaal: Het selecteren van een nieuwsbrief, na kiezen "rapportage" ophalen van de rapportage uit de database en tonen op scherm.

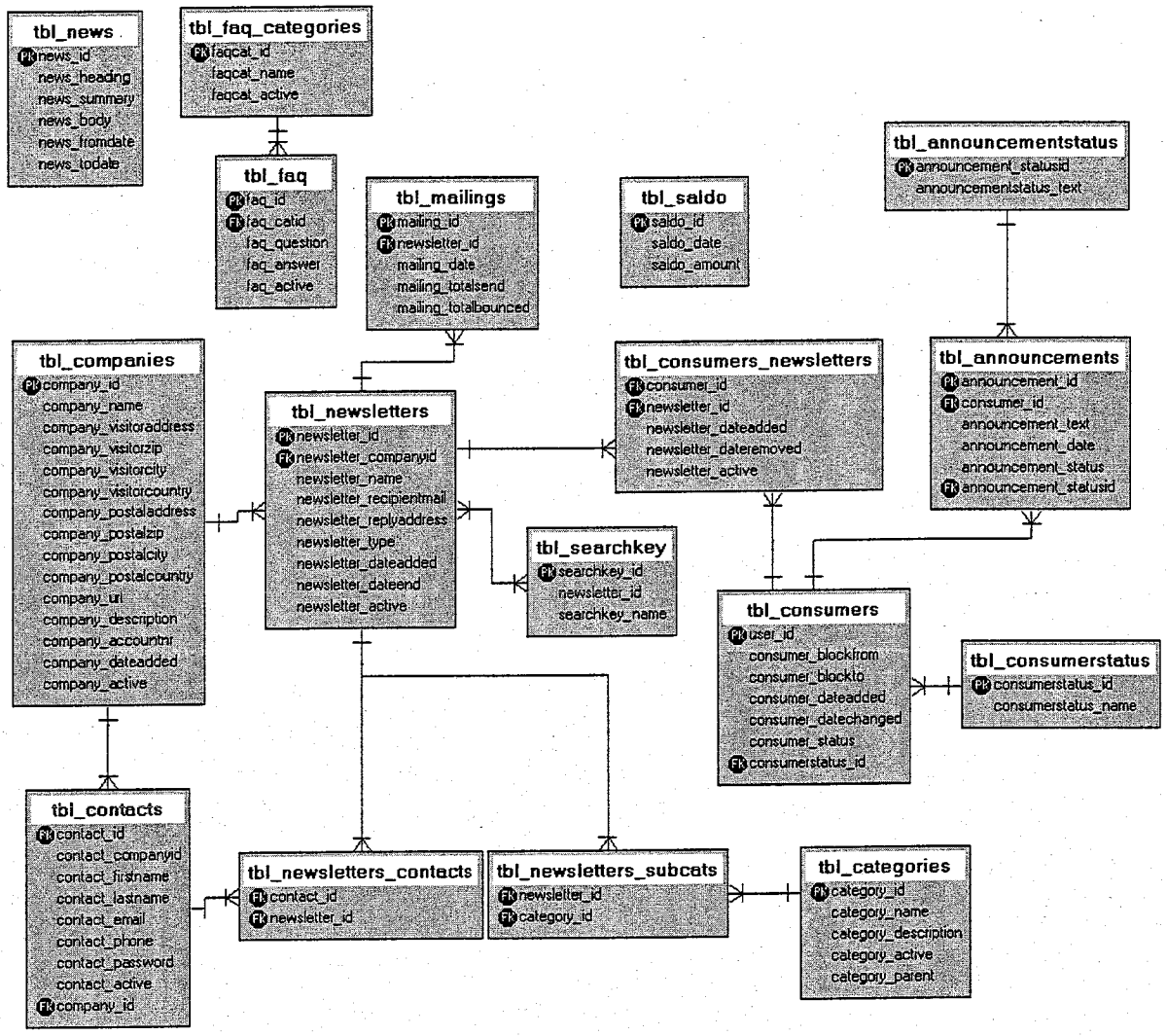
Natuurlijk kunnen er, zoals in elke software applicatie, excepties optreden. Door fouten gemaakt door de gebruiker (expres of per ongeluk) of door technische fouten. Het is onmogelijk om al deze excepties te modelleren. In de software worden deze zoveel mogelijk opgevangen middels gebruiksvriendelijke schermen. Ook kan op de server aangegeven worden wat er in het geval van een kritieke applicatiefout getoond moet worden.

Wel een belangrijke eis bij het opstellen van de C-netten, is dat deze *sound* zijn. Informeel kan gezegd worden dat alle use-cases dan kloppende transacties zijn in het systeem, met een begin en een gegarandeerd einde [CHS2003]. Een definitie van soundness is te vinden in Bijlage A. Het is eenvoudig te constateren dat alle eerder opgestelde netten zich gedragen als State Machine WorkFlow nets (SMWF). Omdat elke SMWF sound is (zie ook Bijlage A), is hiermee soundness bewezen.

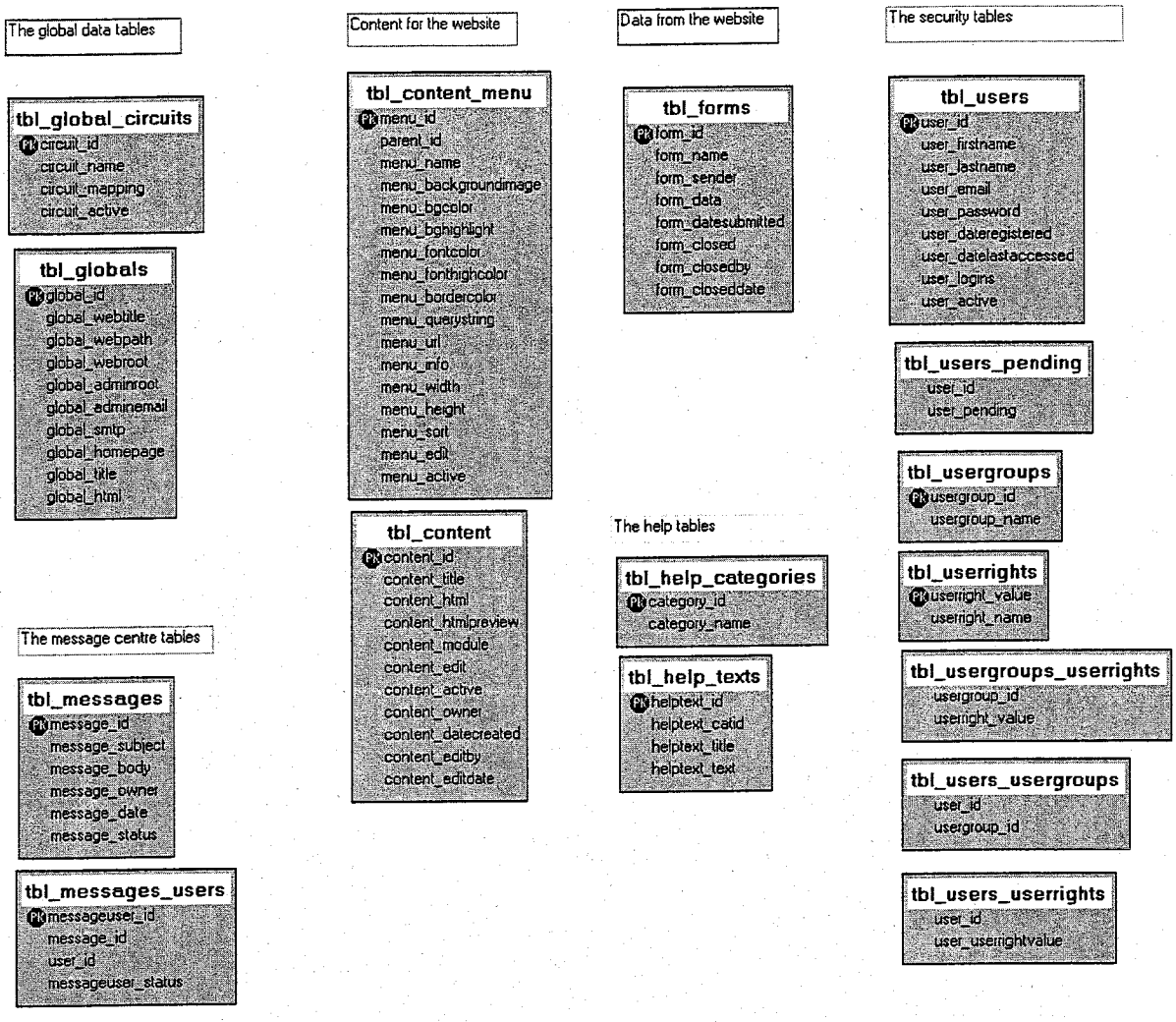
3.3. Het datamodel

Zoals vele dynamische webapplicaties zijn user interface en database voor SpaarPost erg belangrijk. Door deze logica duidelijk te scheiden kan later eenvoudig worden gevonden waar eventuele aanpassingen plaats moeten vinden. Aan de basis van de applicatie moet een degelijke, consistente database staan. Het is daarom erg belangrijk dat een flexibel datamodel wordt opgesteld. Een aanpassing in de applicatie mag niet meteen leiden tot een drastische aanpassing van het datamodel.
Het datamodel geeft weer welke tabellen zullen worden gebruikt en welke relaties daartussen bestaan. Figuur 3.2 en Figuur 3.3 bevatten het datamodel van SpaarPost. Figuur 3.3 is het datamodel van de CMA en geeft daarbij geen relaties weer tussen de tabellen.

Het datamodel geeft relevante informatie over de business objecten in de applicatie. Business objecten zijn de klassen die overeenkomen met reële zaken in de context van de SpaarPost applicatie. Belangrijkste objecten zijn de consument, het bedrijf, de nieuwsbrieven, de selecties (opt-in) en de mailing-acties. Elke nieuwsbrief behoort weer tot een categorie. Er zijn tal van functionaliteiten die indirect te maken hebben met deze objecten (zie use-cases). Business Objecten zijn een onderdeel van object oriëntatie technieken. De gekozen programmeeromgeving staat echter deze techniek niet toe (zie hoofdstuk Technische Architectuur), maar leent zich meer voor een methode die puur gebaseerd is op componenten. Om deze reden is ook geen klassemodel gerealiseerd, maar is de applicatie puur opgedeeld in componenten. Elke eerder uitgewerkte use-case is daarom hiërarchisch in te delen in grotere componenten. Dit aspect wordt in het volgende hoofdstuk behandeld.
Wat betreft de user interface is de intentie om zo vroeg mogelijk in het ontwikkelproces benodigde wijzigingen door te voeren en om zoveel mogelijk aan de wensen van de gebruiker te voldoen. Ook deze methode wordt nader toegelicht in het hoofdstuk Technische Architectuur.



Figuur 3.2 Het datamodel (1)



Figuur 3.3 Het datamodel (2)

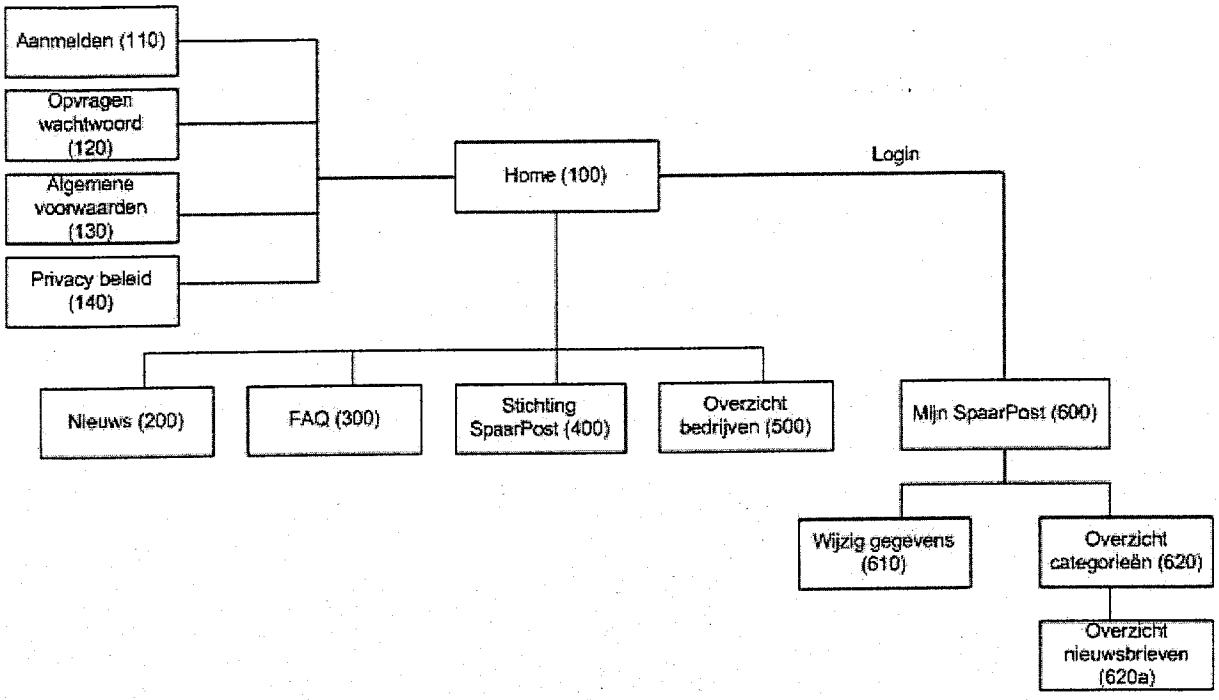
4
Functionele architectuur

4.1. Sitemap

Een belangrijk overzicht dat volgt uit de use-cases is de sitemap. De sitemap is een overzicht van de webapplicatie en laat zien hoe de gebruikers navigeren over de site. Op hiërarchische wijze worden de pagina's ingedeeld, zodat men kan zien hoeveel clicks nodig zijn om op een bepaalde pagina te komen. Relevante pagina's moeten daarbij niet te diep geworteld zijn. Door de sitemap vroeg in het project op te stellen kan het als waardevol communicatiemiddel dienen tussen stakeholders en ontwikkelteam. De stakeholders hebben een betere visie over het gehele systeem en voor de ontwikkelaars is het een noodzakelijke leidraad naar implementatie. Tijdens het creëren van de sitemap is nog niet bekend hoe de pagina's er werkelijk uit gaan zien, er wordt slechts een navigatiestructuur gegeven. Deze pagina's kunnen later nog wel worden opgesplitst of samengevoegd. Figuur 4.1 geeft de sitemap voor het consumentengedeelte van SpaarPost. Het is tevens een soort van state machine, omdat aan te geven is in welke toestand de gebruiker zich op een bepaald moment bevindt.Ook is er een duidelijke relatie tussen de use-cases en deze sitemap. Elke pagina verzorgt een stukje functionaliteit uit de use-cases.

Een voorbeeld:
Use-case: *Het selecteren van een nieuwsbrief*
Alvorens te kunnen selecteren moet de gebruiker ingelogd zijn en zich bevinden op de pagina "Mijn SpaarPost". Daar wordt gekozen voor het toevoegen van een nieuwsbrief en krijgt de gebruiker een overzicht van de categorieën. Wanneer hij een keuze maakt uit deze categorieën komt hij op de pagina waar hij nieuwsbrieven binnen deze categorie kan selecteren.

Dit voorbeeld geeft een scenario dat zich over meerdere schermen afspeelt. Er zijn echter ook scenario's die op slechts één scherm uitgevoerd kunnen worden of zelfs meerdere op één scherm.

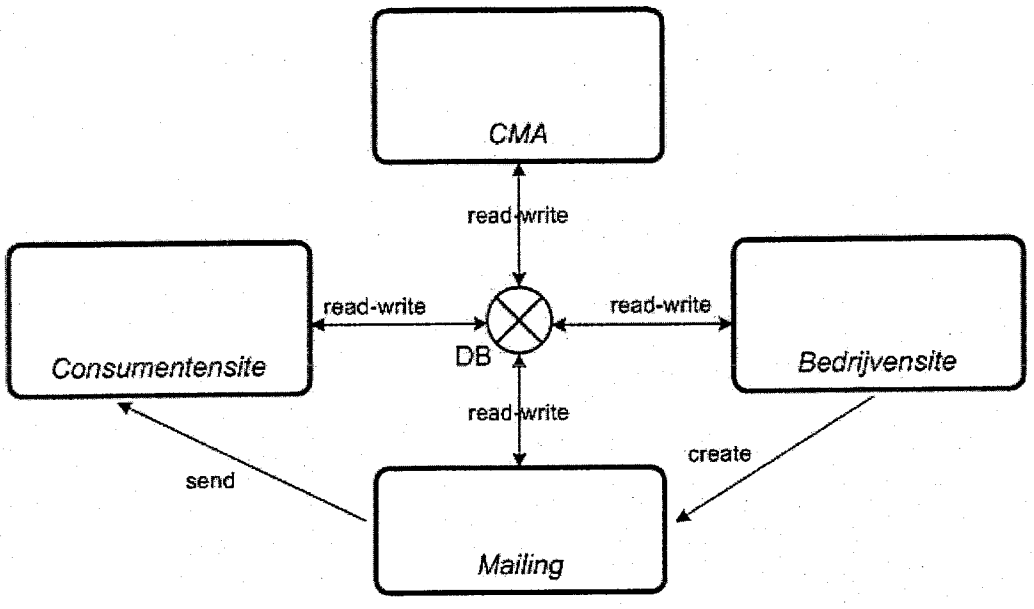


Figuur 4.1 De sitemap

Voor het bedrijvengedeelte en de CMA gebruiken we geen sitemap. De CMA is puur gebaseerd op het onderhouden van objecten. CRUD (Create, Retrieve, Update, Delete). Deze beheert bedrijven, contactpersonen, nieuwsbrieven en teksten. Het bedrijvengedeelte is een interface die als voornaamste functie voorziet in de workflow voor de nieuwsbrieven.

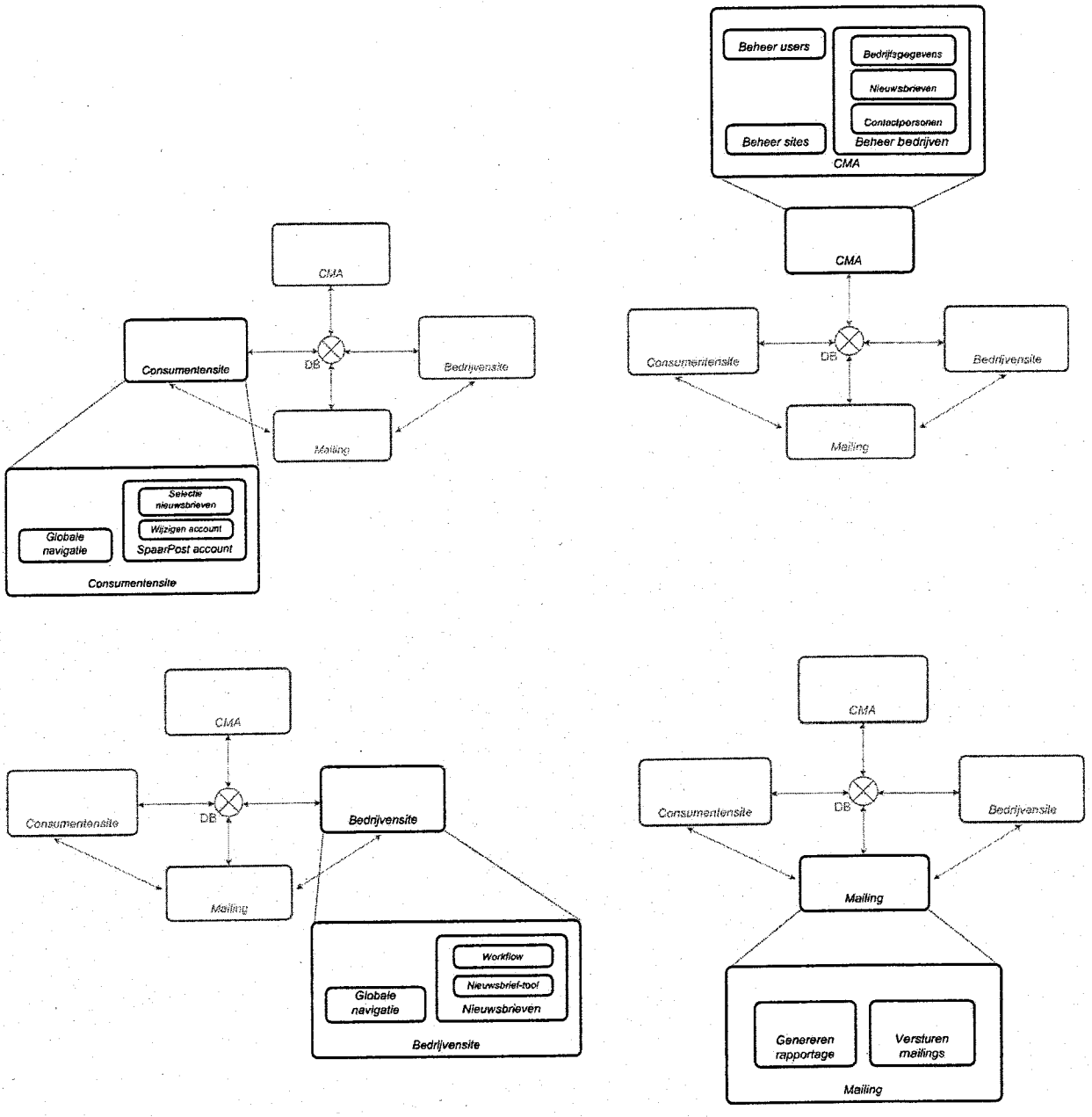
4.2. Componentverdeling

Om een goede hiërarchie toe te kennen aan de applicatie wordt deze onderverdeeld in componenten. De eerder gemaakt component-nets van de use-cases moeten daarbij elk in een overkoepelende component ondergebracht worden. Startend met deze componenten bekijken we een globaal workflow net en vervangen een plaats door een nieuw sound t-workflow net. De kleinste componenten, de use-case netten kunnen vervolgens in deze componenten worden ondergebracht. We gebruiken daarom een combinatie van top-down en bottom-up ontwerpen.
De meest relevante objecten zijn de bedrijven, consumenten, nieuwsbrieven en mailing-acties. Allereerst moeten deze objecten aangemaakt kunnen worden. Dit proces moet zoveel mogelijk geautomatiseerd worden, echter het creëren van sommige objecten is afhankelijk van menselijke beslissingen (SpaarPost). SpaarPost beslist namelijk of een bedrijf wel of niet toegevoegd kan worden en tot welke categorie dit bedrijf zijn nieuwsbrieven kan richten. Voor dit alles en het beheer van de website zelf is een Content Management Applicatie nodig (CMA). Dit is de eerste component in het systeem.
In tegenstelling tot bedrijven, moeten consumenten zichzelf wel aan kunnen melden, zonder tussenkomst van SpaarPost. De belangrijkste activiteit van een consument is daarna de selectie van nieuwsbrieven. Hiervoor is de consumentensite van SpaarPost. Voordat de consument een nieuwsbrief kan ontvangen moet er een nieuwsbrief worden opgesteld. Dit is ook meteen de hoofdactiviteit van het bedrijf. Wanneer het bedrijf klaar is met verzenden wordt de nieuwsbrief klaargezet voor verzending en moet een mailing-component voor de werkelijke verzending gaan zorgen. Schematisch is dit als volgt weer te geven.



Figuur 4.2 Componentenraamwerk

Bovenstaand schema geeft in grote lijnen weer in welke onderdelen de webapplicatie verwerkt is. Het is op deze manier erg eenvoudig om alle eerder opgestelde use-cases te verdelen over deze componenten. De indeling in componenten is echter nog een slag dieper te maken. Uit het use-case diagram is deze indeling al enigszins af te leiden. De consumentensite zal bestaan uit een component *Globale navigatie* en een component *SpaarPost account*, welke weer onder te verdelen is in *Selectie nieuwsbrieven* en *Wijzigen account*. De bedrijvensite bevat eveneens een *Globale navigatie* en een component voor het bewerken van de nieuwsbrieven. Zoals in de use-cases al is aangegeven doorloopt de nieuwsbrief een soort van workflow van schets naar gereed product. Behalve deze workflow moet er eveneens een component of tool aanwezig zijn om de nieuwsbrieven op te stellen of in te laden, te bewerken, en vervolgens weer op te slaan. (zie Figuur 4.3)



Figuur 4.3 Subcomponenten

Bijna alle componenten zijn intern weer onder te verdelen in twee deelcomponenten. Eén deelcomponent zorgt voor de communicatie met de database, het andere gedeelte zorgt voor de user interface naar de gebruiker van de component. Om al deze logica in de Petri Netten te scheiden zou te ver gaan, echter de gekozen programmeeromgeving maakt dit in de code wel gestructureerd mogelijk. Deze codestructuur wordt besproken in het volgende hoofdstuk, waarin onder anderen de ontwikkelmethode wordt uitgelegd.

4.3. Componentmodellering

Aan de vier hoofdcomponenten moeten nog use-cases worden toegewezen. In grote lijnen ligt deze structuur al vast, maar de communicatie tussen de componenten of use-cases onderling moet ook nog beschreven worden. Dit is mogelijk door de eerder opgestelde C-netten in te passen in de componenten en zo per component een compleet net te creëren.

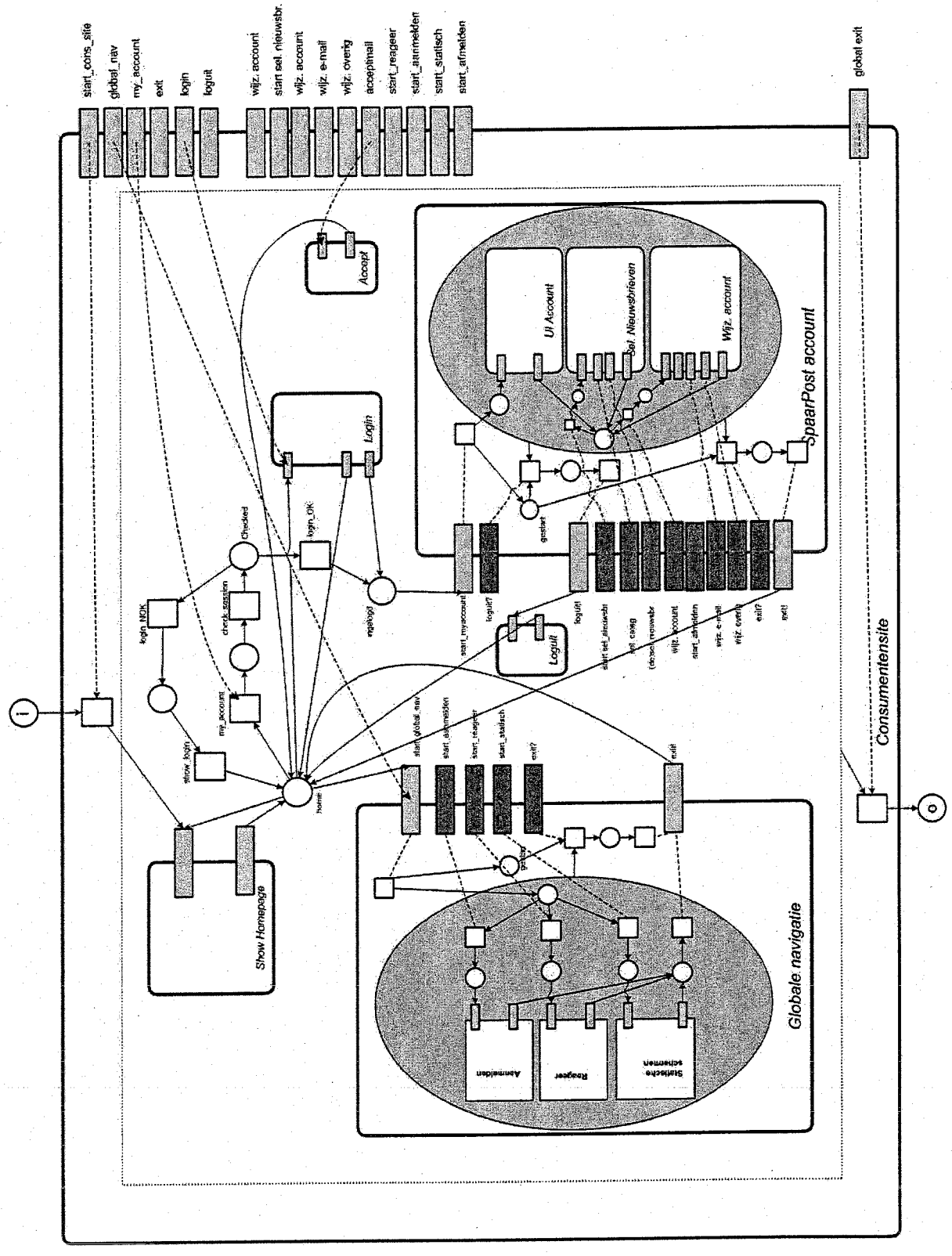
Consumentensite

De eerste component is de consumentensite. In Figuur 4.4 zie je een totaal net van deze component. De werking van deze component zal worden toegelicht.

Er zijn in een webapplicatie tal van communicaties van en naar de gebruiker. De communicatie van de gebruiker uit zich in een actie zoals deze aan de rechterkant van de component staan vermeld. Om een wirwar van lijnen te voorkomen zijn een aantal labels, welke actie van de gebruiker vragen donkerder gekleurd. Communicatie naar de gebruiker gaat altijd via scherm of een verstuurde e-mail en uit zich in de naamgeving van de betreffende transitie.

Voordat een van de interne componenten kan worden aangeroepen moet de interface gestart worden door middel van use-case *SpaarPost Globaal*. Deze use-case omvat de totale component. Wanneer een gebruiker de consumentensite start, wordt allereerst de homepage gestart. Vanuit deze component wordt de plaats *Home* bereikt. Door in te loggen is toegang te verkrijgen tot de *SpaarPost Account* waar de consument een keuze kan maken uit zijn nieuwsbrieven of zijn accountdata kan wijzigen. Behalve deze component is er de *Globale Navigatie* welke ook zonder autorisatie te bereiken is. Om de componenten te verlaten (wanneer de gebruiker kiest voor een andere site of uitlogt) wordt in het net gebruik gemaakt van een superplaats. Wanneer zich ergens in de component een token bevindt en de component is gestart (er ligt een token in *gestart*) kan de gebruiker, door middel van een exit- of loguit-actie, de component direct verlaten. Een exit-actie is een actie die elke keer vuurt wanneer de gebruiker een site verlaat.

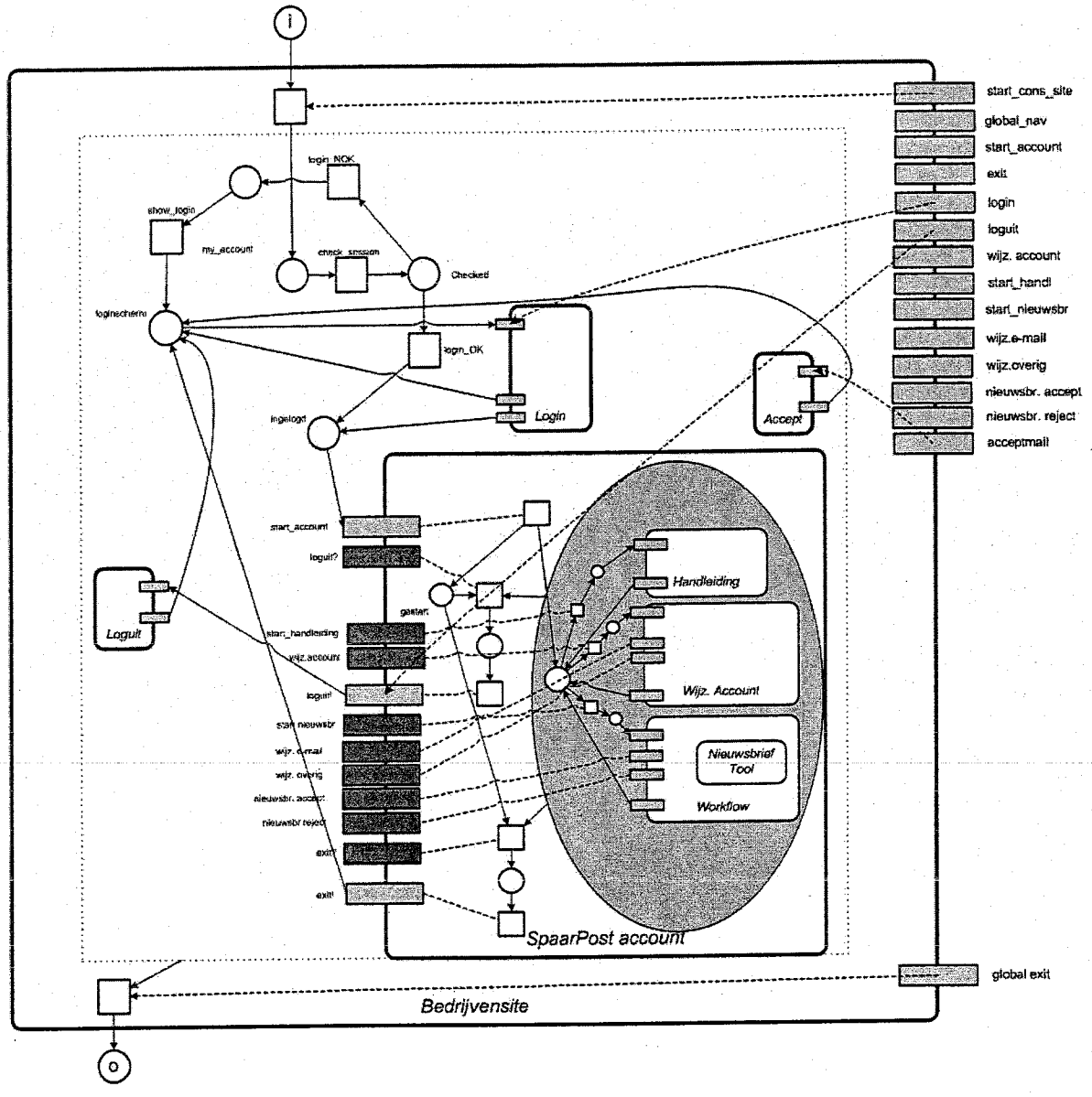
Omdat de site in zijn geheel ook verlaten moet kunnen worden is de gehele component omlijnd door een gestippeld vierkant, welke eveneens een superplaats is. De "globale exit" kan de component beëindigen.



Figuur 4.4 Consumentensite

Bedrijvensite

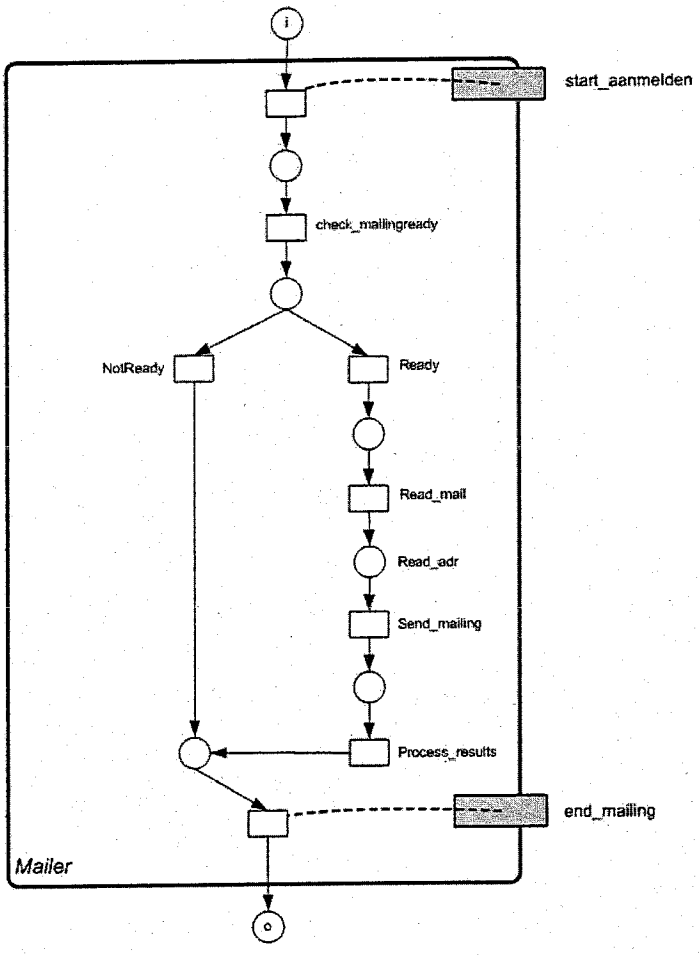
De tweede component die dieper uitgewerkt kan worden is de bedrijvensite. Deze component is wat kleiner en heeft als belangrijkste subcomponent de workflow voor het opstellen en verzenden van nieuwsbrieven. Voor toegang tot de componenten is altijd een login vereist, welke door SpaarPost wordt verstrekt. Er kan vervolgens gekozen worden uit het bekijken van de handleiding, het beheren van nieuwsbrieven (de workflow) en het wijzigen van de account. De meest interessante component is hier de workflow van nieuwsbrieven. Zolang de nieuwsbrief nog status "Draft" heeft kan deze nog bewerkt worden met de nieuwsbrief-tool. Er wordt een testmail verstuurd met code, zodat de gebruiker een draft kan goedkeuren met deze code. Wanneer deze code klopt is de nieuwsbrief klaar om verzonden te worden. Wanneer deze vervolgens met een druk op de knop verzonden wordt komt deze in de wachtrij van de mailer.



Figuur 4.5 Bedrijvensite

De mailer

De mailer is een component die periodiek moet gaan kijken of er nieuwsbrieven klaar staan in een wachtrij. Indien dit het geval is, moeten deze nieuwsbrieven worden verstuurd naar de op dat moment ingeschreven consumenten. Tevens moet er na de actie een feedback komen naar SpaarPost met de resultaten van elke mailing-actie.



Figuur 4.6 Mailer

Voor de CMA-component stellen we geen C-net op. De meest relevante modules hierin zijn:

FAQ	- Voor het beheren van de Frequently Asked Questions
Nieuws	- Voor het beheren van het nieuws en het nieuwsarchief
Categorieën	- Voor het beheren van de beschikbare categorieën en subcategorieën
Bedrijven	- Voor het beheren van de bedrijfsgegevens, contactpersonen en nieuwsbrieven

5
Technische architectuur

Dit hoofdstuk zal toelichten op welke manier alle logische componenten verwerkt zijn tot software componenten in de applicatie. Er wordt toegelicht hoe de componenten zijn ontwikkeld en waarom bepaalde keuzes gemaakt zijn. Voordat we dieper in kunnen gaan op deze keuzes zal eerst wat meer duidelijk moeten zijn betreffende de gekozen ontwikkelmethode. Daarom starten we dit hoofdstuk met een korte uitleg over de gebruikte programmeertaal en de gekozen programmeermethode.

5.1. Coldfusion

Coldfusion MX is een product van Macromedia. Het is een applicatie die draait op een webserver, ontwikkeld om een eenvoudige maar krachtige variant te creëren van andere scripttalen, zoals Perl, ASP, enzovoorts. Coldfusion is een programmeertaal die tag-based is (net als HTML) en kan op eenvoudige wijze communiceren met verschillende types databases. Het is geen objectgeoriënteerde taal maar specialiseert zich op de interactie met de gebruiker. Web applicaties kunnen zo ook eenvoudig informatie uit de database ophalen, tonen, bewerken en opslaan. Verschillende technologieën zijn mogelijk met Coldfusion. Tags maken het automatisch genereren van Javascript mogelijk en zelfs technieken als COM, Corba en Java zijn geïntegreerd.
Ook wordt rekening gehouden met schaalbaarheid, omdat Coldfusion ontwikkeld is om ook te kunnen draaien op een "cluster" van servers, om ook erg drukke pagina's voldoende snelheid te bieden.
Verder is Coldfusion een "server-side" taal. Dit betekent dat de code niet draait op de browser van de client maar op de web server, zodat scripts in Coldfusion op elke browser identiek draaien. De belangrijkste voordelen van server side scripting zijn dan weer de onafhankelijkheid van de client browser (omdat de output zuiver HTML-code is) en de uitgebreide mogelijkheden tot interactie tussen de gebruiker en de databank(en) op de *server*.

5.2. Fusebox

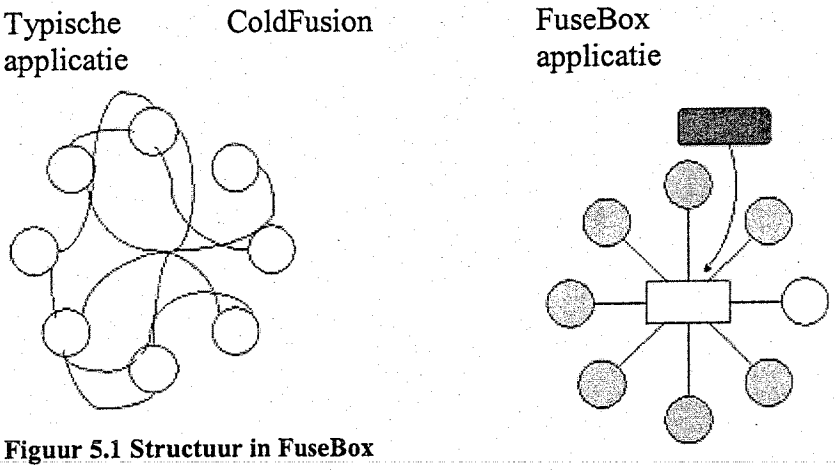
Fusebox is enerzijds een framework met helpfiles en organisatorische principes, anderzijds is het een methodologie om het software ontwikkeltraject te verbeteren. Fusebox bevat een framework met "core files" die het creëren van complexe websites overzichtelijker maakt. Instellingen, variabelen en layouts kunnen in deze files beheerd worden. Fusebox, de naam zegt het al, is een soort kast met zekeringen (fuses). Als er een zekering springt, moet de rest blijven werken. Elke zekering heeft een eigen functie en heeft daarbij geen hulp nodig van andere zekeringen. De gehele fuse box (de applicatie) bestaat uit verschillende circuits (componenten), welke elk een eigen stukje functionaliteit bevatten. Elk circuit bestaat weer uit fuseacties welke via requests in de circuits worden aangeroepen. Deze fuseacties zijn weer opgedeeld in individuele fuse files. Een fuse is een atomair stukje code, alleen bereikbaar vanuit de huidige component, dus private. Vaak wordt eerst een fuse aangeroepen die een query uitvoert, waarna een andere fuse de resultaten op het scherm plaatst. De volgende fuses zijn beschikbaar in FuseBox:

• **Display Fuses**	-	prefix: "dsp", gebruikt om informatie te laten zien aan de gebruiker.
• **Select Query Fuses**	-	prefix: "qry", gebruikt voor SELECT-queries.
• **Action Query Fuses**	-	prefix: "act", gebruikt voor INSERT, UPDATE, DELETE of andere actie-queries.
• **Location Fuses**	-	prefix: "url", gebruikt om de applicatie naar een nieuwe URL te sturen.

Om een snelle realisatie van de applicatie te bevorderen en de risico's op fouten te verkleinen wordt een dergelijke methode toegepast. Kort samengevat zijn de voordelen van FuseBox:

- Een standaard raamwerk waardoor iedereen die FuseBox kent snel van de structuur van de broncode van de applicatie op de hoogte is.
- Verhoogde modulariteit in de broncode. (maakt het hergebruik van de broncode eenvoudiger)

Deze voordelen zijn beiden doorslaggevend geweest in het ontwikkeltraject van SpaarPost. Enerzijds betekent dit dat de structuur van de applicatie snel te begrijpen is, terwijl anderzijds eenvoudig bestaande componenten ingevoegd kunnen worden.
FuseBox is een consistente manier om directories en files te rangschikken, corresponderende met objecten en acties. Ook helpt het om de flow van programmacode te beheersen.



Figuur 5.1 Structuur in FuseBox

De structuur wordt gehandhaafd door de volgende FuseBox core files:

- fbx_Circuits.cfm
- fbx_Settings.cfm
- fbx_Layouts.cfm
- fbx_Switch.cfm
- fbx_FuseboxXX_CFXX.cfm

We starten met de fbx_Circuits file. Deze houdt bij waar in de mappen-structuur zich bepaalde bestanden bevinden. Er is slechts één dergelijke file in de gehele applicatie wat

betekent dat alle locaties centraal zijn opgeslagen. Ook worden in dit bestand aliassen gemaakt, zodat mappen met een korte duidelijke naam kunnen worden aangeroepen.
Het volgende bestand, fbx_Settings, bevat alle default waarden binnen de applicatie. Er is er altijd één van in de root van de applicatie en indien in de diepere structuur een instelling overschreven moet worden kan dat door ook daar dit bestand te plaatsen en de waarden opnieuw in te stellen.
Met fbx_Layouts kan voor iedere sectie in de applicatie een andere layout worden ingesteld.
Een belangrijke maar ook eenvoudige file is fbx_Switch. Dit bestand regelt welke bestanden moeten worden geladen in een pagina, bij een bepaalde fuse-actie.
Alle instellingen in eerder genoemde files worden beheerd in de laatste file fbx_FuseBoxXX_CFXX.cfm, waarbij de eerste XX voor de versie van FuseBox staat en de tweede XX voor de versie van ColdFusion. Dit bestand is het hart van de FuseBox methode.

5.3. Implementatie

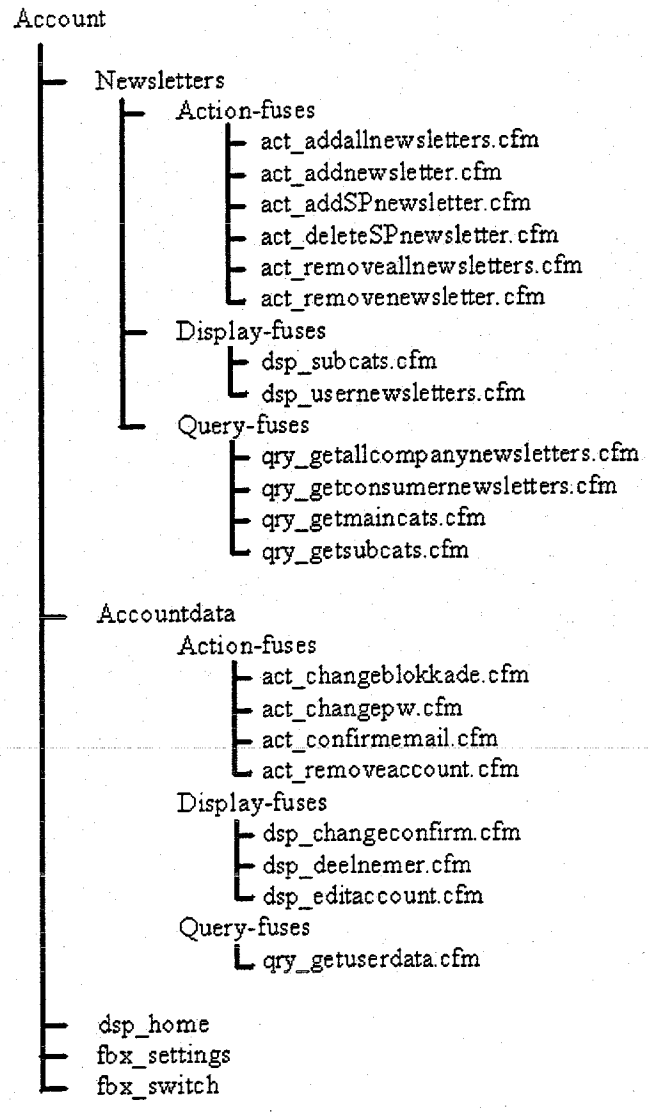
De startdatum voor de nieuwe wetgeving tegen junkmail en het beëindigen van een samenwerking met een andere partij heeft geleid tot een tijdsdruk voor het opleveren van de applicatie. Dit maakt het gebruik van bestaande componenten niet alleen gewenst, maar ook noodzakelijk.

De consumentensite is echter "from scratch"ontwikkeld. Een gebruiksvriendelijke en logische manier voor het selecteren van nieuwsbrieven was hierbij het voornaamste.
De software-componenten en data-opslag zijn hierin als volgt geordend:

Logisch	Software	Uitleg
SpaarPost Account	Account	Alle logica beschikbaar voor de ingelogde gebruiker, onderverdeeld in onderdelen voor het selecteren van nieuwsbrieven en het wijzigen van de accountgegevens
Globale Navigatie	Algemeen	Globale navigatie, statische schermen en alles waarvoor niet ingelogd hoeft te zijn
Login/Loguit	Login/Loguit	Code voor authenticatie/autorisatie
	Customtags	Tools welke voor dit project juist geconfigureerd zijn, zoals controle voor login, toegang tot pagina's e.d.
	Images	Plaatjes
	Layout	Het globale uiterlijk van de pagina's, de standaard layout
	Library	Javascripts voor de controle van formulieren en validatie aan de client-zijde

Tabel 5.1 Indeling consumentensite

Als voorbeeld gaan we de eerste component uitwerken, om aan te geven hoe de logische componenten zijn verwerkt in de software. Voor overige componenten is dezelfde strategie toegepast. De eerste en tevens meest relevante component "Account" is gestructureerd zoals in Figuur 5.1 te zien is. In de logische component "SpaarPost Account" is een onderverdeling gemaakt in enerzijds de user interface voor de startpagina, en anderzijds de componenten "Selectie Nieuwsbrieven" en "Wijzigen Account". In de root van de hoofdcomponent "Account" staan twee Fusebox-bestanden, fbx_settings en fbx_switch. Fbx_settings bevat algemene instellingen, zoals globale variabelen. Fbx_switch regelt de gehele structuur en bepaalt welke fuses aangeroepen worden bij welke pagina's. De user interface voor de startpagina haalt al direct onderdelen uit de subcomponenten "Nieuwsbrief" en "Accountdata". Voor het tonen van deze pagina zijn namelijk de huidige gebruikersgegevens nodig (qry_getuserdata) en de geselecteerde nieuwsbrieven (qry_getconsumersnewsletters). In dit geval blijkt dus dat logische component "UI Account" delen van de andere subcomponenten gebruikt. Heel duidelijk is verder de eerdere scheiding in logica tussen de display-fuses voor alle user interface logica, de action-fuses voor de wijzigingen in de database en de query-acties voor het lezen uit de database.



Figuur 5.2 Codestructuur bij component "Account"

Elke component bevat deze structuur, waardoor de programmeur altijd goed op de hoogte is van de locatie binnen de component waar hij zich bevindt en werkt op dat moment ook aan een zeer specifiek onderdeel van de applicatie.

Behalve ontwikkeling "from-scratch" is ook het configureren van bestaande componenten een hoofdactiviteit geweest bij de implementatie. Voor het bedrijvengedeelte heeft het gebruik van bestaande componenten relevante tijdswinst opgeleverd. Voor het bewerken van nieuwsbrieven was reeds een tool beschikbaar dat optimaal ingezet kon worden. Er kan op elk niveau aan een nieuwsbrief gewerkt worden. In een Word-look-a-like omgeving, of rechtstreeks in de broncode. Ook de workflow van nieuwsbrieven heeft een interface die is afgeleid van de CMA. In de CMA worden immers eveneens items toegevoegd, gewijzigd of verwijderd, net als in de workflow met de nieuwsbrieven gebeurt. Tabel 5.2 geeft de ordening binnen de bedrijvensite, waarin "Nieuwsbrieven" en "Editor" het meest interessant zijn.

Logisch	Software	Uitleg
Wijz. account	Accountdata	Alle logica wijzigen van de accountgegevens
Workflow	Nieuwsbrieven	De workflow voor de nieuwsbrieven
Nieuwsbrief tool	Editor	Component om de nieuwsbrieven aan te passen, uploaden en bewaren.
Handleiding	Handleiding	De Handleiding
Login/Loguit	Login/loguit	Code voor authenticatie/autorisatie
	Customtags	Tools welke voor dit project juist geconfigureerd zijn, zoals controle voor login, toegang tot pagina's e.d.
	Images	Plaatjes
	Layout	Het globale uiterlijk van de pagina's, de standaard layout
	Library	Javascripts voor de controle van formulieren en validatie aan de client-zijde

Tabel 5.2 Indeling bedrijvensite

Voor de CMA bestond eveneens een goed geteste, bewezen component. Deze tool biedt de mogelijkheid om database objecten te bewerken. Ook pagina's op de website zijn zo via een centrale locatie te beheren. Er is een module voor de pagina's én een module voor overige database-objecten waarvoor centraal beheer noodzakelijk is. Tabel 5.3 geeft binnen de CMA aan hoe de onderdelen zijn verdeeld.

Software	Uitleg
Algemeen	Algemene instellingen van de CMA
Help	Een helpfunctie hoe de CMA werkt
Menu	Navigatie binnen de CMA
Messages	Berichten voor gebruikers van de CMA
Modules	De verschillende modules binnen de CMA, Bedrijven, FAQ, Nieuws en Overig
Pagina's	Statische pagina's waarvan de teksten kunnen worden gewijzigd en on- en offline kan worden ingesteld
Rechten	De rechten van de gebruikers
UserManager	De gebruikers

Tabel 5.3 Indeling CMA

De mailing component is beschikbaar bij de hostingpartij. Primaire voorwaarde is dat implementatie van communicerende onderdelen, zoals de workflow voor nieuwsbrieven, conform de eisen van deze component gebeurt. De mailer is geen ColdFusion component, maar een volledig losstaand onderdeel. Daarom volgt een uitleg van de afspraken die gemaakt zijn betreffende de functies van dit onderdeel.

Via de bedrijvensite worden twee bestanden in een configureerbare directory op het systeem (buiten de document root) geplaatst. Deze bestanden hebben de namen "xxxxxxxx.mail" voor de nieuwsbrief zelf en xxxxxxxx.addresses voor de adressen waar de nieuwsbrief naar verstuurd zal moeten worden. Hierbij is xxxxxxxx een unieke identificatie. De SpaarPost mailer controleert periodiek deze directory en zal deze bestanden ontdekken en verwerken. Wanneer de mailer begint aan het verwerken van de betreffende mailing worden de bestanden naar een archive directory verplaatst. De mailer zal uiteindelijk een bericht in de directory plaatsen met als bestandsnaam xxxxxxxx.xml met daarin resultaten over de mailing-actie. De exacte inhoud/formaat van dit bestand wordt later beschreven. De mailer zal niets met de bestanden doen voordat zowel een .mail en een .addresses bestand voor de betreffende identifier aanwezig zijn. Op het moment dat dat het geval is, mag de mailer er vanuit gaan dat de bestanden volledig zijn. De mailer neemt maatregelen om te voorkomen dat in geval van storing het bericht nul- of meerdere malen naar hetzelfde adres wordt verzonden. De website zorgt ervoor dat een e-mail adres slechts éénmaal in het bestand voorkomt.
Voor exacte specificaties van de bestanden, zie Bijlage A.

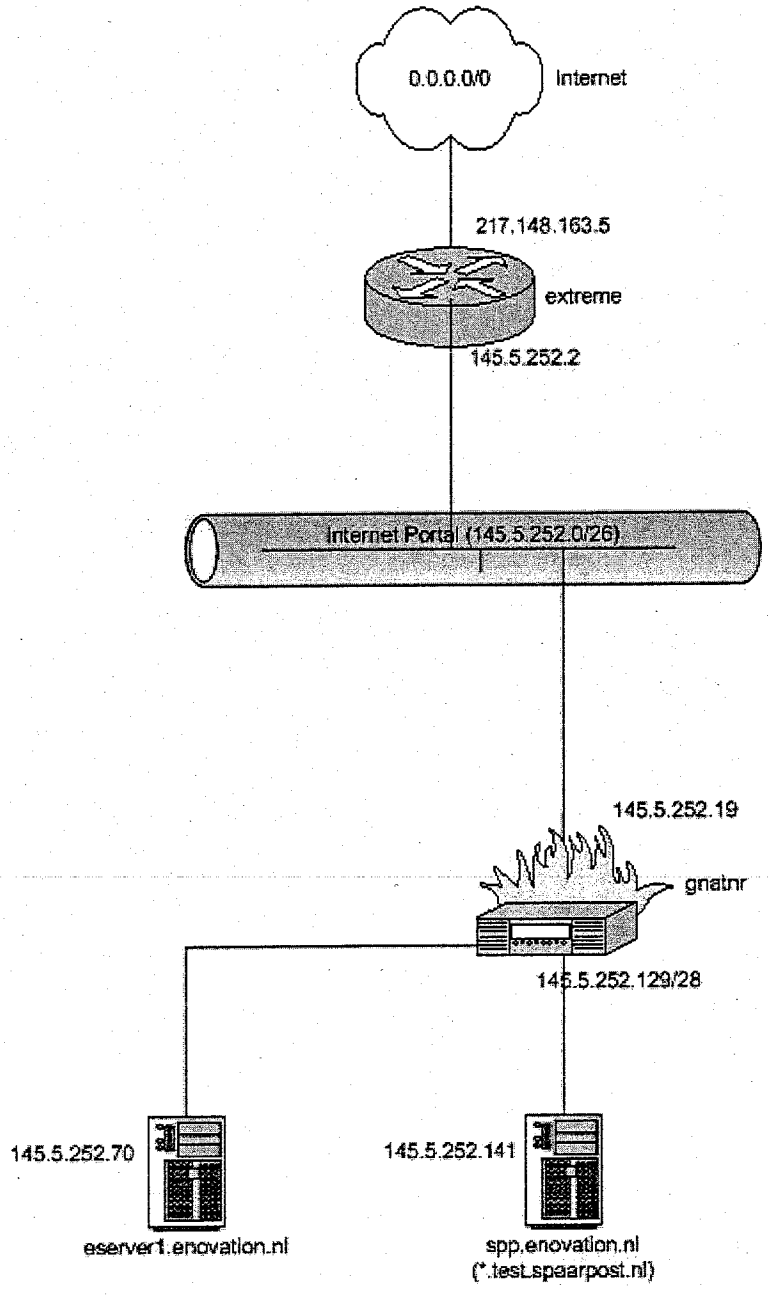
Hiermee is de technische architectuur volledig. In de indeling van de software componenten is vooral zichtbaar dat de logische componenten hier direct in terugkomen.

6
Infrastructuur en ontwikkelomgeving

Als ontwikkelomgeving maken we gebruik van Windows 2000 en Windows XP met ColdFusion MX voor Windows als programmatuur en in eerste instantie MySQL als database. Dat deze keuze flexibel is zal spoedig blijken. De hosting van de applicatie wordt verzorgd door de firma E-Novation. Omdat de hosting is gebaseerd op een Linux OS omgeving en de ontwikkeling van de applicatie op een Microsoft Windows platform is deze flexibiliteit ook een vereiste. Coldfusion draait zowel op een Windows als een Linux platform. Ook MySQL kan probleemloos omgezet worden. Een aantal zaken waarbij tijdens ontwikkeling rekening mee wordt gehouden zijn hoofdlettergevoeligheid en slashes in Linux (Zowel voor programmacode als database).
De keuze voor MySQL boven meer commerciële systemen als bijvoorbeeld Oracle is een voorlopige beslissing, echter hoeft niet definitief te zijn. Oracle heeft een uitgebreidere ondersteuning (subselecties, views, referentiële integriteit, enz.) dan MySQL. MySQL is daarentegen gratis en ondersteunt eveneens grote hoeveelheden data. Ook qua snelheid is MySQL (zeker voor de wat kleinere databases) superieur. Dit is de reden dat in de eerste instantie gekozen wordt voor MySQL. Later kan de database eventueel worden omgezet naar Oracle (hiervoor zijn tal van conversieprogramma's beschikbaar). Voor Oracle zijn immers betere tools aanwezig voor monitoring. Wat betreft de mogelijkheid om de MySQL database te converteren naar Oracle geldt het volgende:
Over het algemeen zullen er enkele kleine verschillen zijn, maar MySQL is een subset van Oracle. MySQLDump is een tool om data eenvoudig over te zetten. Verder hangt deze eenvoud ook af van de kennis van en ervaring met beide systemen. Enkele knelpunten zouden kunnen optreden met auto increment en tijd/datum velden. In Oracle bestaat in tegenstelling tot in MySQL het auto increment type niet, waardoor de programmeur een zgn. "sequence" en "trigger" moet creëren om de volgende waarde uit de tabel te lezen en te plaatsen in het nieuwe veld. In plaats van triggers zou de sequence ook direct in de insert statements geplaatst kunnen worden. Voor tijd/datum velden geldt hetzelfde. In Oracle moet bijvoorbeeld een trigger geschreven worden om de systeemdatum een veld te schrijven, terwijl in MySQL een dergelijk veld direct aangemaakt kan worden. Voor de eenvoud heeft MySQL hier dus het voordeel, wat betreft flexibiliteit is Oracle de beste. Voor beide opties valt wel iets te zeggen. Buiten deze aspecten is het relatief eenvoudig om code te converteren van MySQL naar Oracle.

Database en applicatie staan tijdens de acceptatietest nog op één server, maar zodra de pilot start zal deze gescheiden worden. Dit om de gegevens in de database nog beter te beveiligen. Onderstaand overzicht geeft de infrastructuur van de huidige situatie bij E-Novation. Applicatie/database server en mailserver zijn gescheiden. Database en applicatie kunnen eventueel ook gescheiden worden.
In hoofdstuk 2 werden bij de niet-functionele eisen onder anderen schaalbaarheid, performance en security genoemd. De keuze voor Coldfusion bevestigt nogmaals deze schaalbaarheid, want met deze serverapplicatie is het mogelijk om meerdere webservers te clusteren.

Wat betreft performance bevat Coldfusion verschillende technieken om processen te versnellen. Voorbeelden hiervan zijn caching-technieken, zoals query caching, client-side caching en server-side caching. Ook het clusteren van web servers komt de performance ten goede. Dit clusteren samen met load balancing zorgt voor een gelijkmatige verdeling van requests over meerdere servers.
Het laatste aspect is security. Deze is echter zeker niet onbelangrijk voor een applicatie als SpaarPost, waarin privacy gewaarborgd wordt en waarbij wellicht een grote publiciteit komt kijken. In de eerste plaats wordt dit aspect met behulp van de juiste hardware aangepakt. Toegang van het Internet naar de Internet portal en andersom wordt geregeld via een router. Extra beveiliging met een additionele router en strenge firewall is er van de Internet portal naar de servers. Toegang tot de applicatiecode en database is alleen mogelijk via ingestelde IP-adressen bij E-Novation en een login.



Figuur 6.1 Infrastructuur

7 Lessons Learned

Tijdens het ontwerp- en ontwikkeltraject van de SpaarPost applicatie zijn er tal van situaties ontstaan waarbij relevante keuzes gemaakt moesten worden. Vooral de afweging tussen ontwikkeling "from scratch" en het gebruik van bestaande componenten stond hierbij centraal. Bij het selecteren van bestaande componenten is het van belang dat deze voldoen aan de specificaties. Enerzijds kunnen het bewezen componenten uit het verleden zijn, anderzijds moeten het componenten zijn waar een goed ontwerp aan vast zit. Dit laatste is iets wat zeker aan de Technische Universiteit uitgebreid aan de orde komt. Een correct, consistent ontwerp.
Het ontwerp van SpaarPost heeft als basis het modelleren in Petri Netten. In dit geval is vooral het accent gelegd op de hiërarchie. Echter ook synchronisatie zou een belangrijk aspect kunnen zijn dat binnen de applicatie gemodelleerd zou kunnen worden met deze methode, door bijvoorbeeld de synchronisatie tussen de hoofdcomponenten onderling te modelleren. Ook bieden Petri Netten tal van analysemogelijkheden. Binnen de scope van dit verslag lijkt analyse wat overbodig, maar het tekent wel de uitgebreide mogelijkheden van een dergelijke ontwerptechniek.
Toch blijft het een moeilijke keuze, de keuze tussen ontwikkeling "from scratch" en ontwikkeling met bestaande componenten en lijkt het moeilijk een afweging tussen deze twee opties te maken. Feit is wel dat steeds meer omgevingen de mogelijkheid bieden om specifieke componenten te programmeren. In het geval van internet applicaties zijn dit vaak webservices, die met simpele in- en outputparameters resultaten kunnen genereren zonder iets van de interne werking van de component te weten.
Ook is dit bijvoorbeeld mogelijk met programmeertaal ColdFusion, door het gebruik van CFC's (ColdFusion Components).

Literatuur

[AHT2000] Component-based software architectures: a framework base don inheritance of behavior, W.M.P. van der Aalst, K.M. van Hee, R.A. van der Toorn, 2000

[CHS2003] Use Cases as Workflows, M. Chaudron, K.M. van Hee, L. Somers, 2003

[AGV2000] Business modelling is not process modelling, H. Akkermans, J. Gordijn, H. van Vliet, 2000

[SS2000] UML Diagrams to Object Petri Net Models: An Approach for Modeling and Analysis, J.A. Saldhana, M.Shatz, 2000

[GM1998] Unified Modeling Language, een overzicht D. Greefhorst, M. Maat, 1998

[HSVW2003] Architecture of Information Systems using the theory of Petri nets, K. van Hee, N. Sidorova, M. Voorhoeve, J. van der Woude

Bijlagen

A Enkele definities / lemma's over Petri Netten

Uit: Architecture of Information Systems using the theory of Petri nets.

Definition 0.6.4 (workflow net). *A Petri net N is a* WF-net (Workflow net) *if and only if:*

- *N has two special places: i and f. Place i is a initial place: ${}^\bullet i = \emptyset$ and f is a final place: $f^\bullet = \emptyset$.*
- *If we add a* closing transition *t to N that connects place f with i (i.e., ${}^\bullet t = \{f\}$ and $t^\bullet = \{i\}$), then the resulting Petri net is strongly connected. Later on, we call this net the* closure *of WF-net N and denote $\overline{N}$ (cf. Fig. 0.5).*

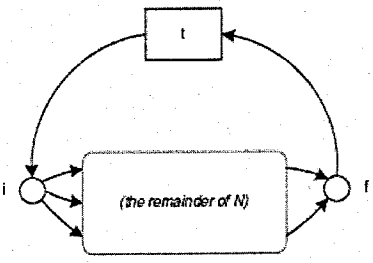


Fig. 0.5. The scheme of $\overline{N}$.

Given the definition of a workflow net, it is easy to derive the following structural properties of WF-nets:

Lemma 0.6.1. *For every WF-net $N = \langle P, T, F \rangle$*

- *the second requirement in the definition above is equivalent to the following requirement: 'Every node $x \in P \cup T$ is on a path from i to f';*
- *for any place $p \in P$ different from i, ${}^\bullet p \neq \emptyset$, i.e., i is the only initial place;*
- *for any place $p \in P$ different from f, $p^\bullet \neq \emptyset$, i.e., f is the only final place.*

Definition 0.6.5 (soundness). *A WF-net is* k-sound *iff for every marking M reachable from marking i^k, there exists a firing sequence leading from marking M to marking f^k. Formally:*

$$\forall M : (i^k \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} f^k).$$

A WF-net is sound *iff for every natural k, it is k-sound.*

Definition 0.6.6 (t-workflow net). *A Petri net N is a* tWF-net (t-workflow net) *if and only if:*

- *N has two special transitions: t_i and t_f. Transition t_i is an initial transition: ${}^\bullet t = \emptyset$. Transition t_f is a stop transition: $f^\bullet = \emptyset$.*
- *If we add a place p to N which connects transitions t_f and t_i (i.e., ${}^\bullet p = t_f$ and $p^\bullet = t_i$), then the resulting Petri net is strongly connected.*

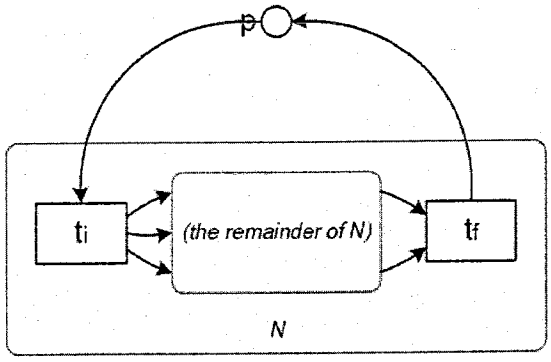


Fig. 0.7. The scheme of $\overline{N}$.

Definition 0.6.9 (SMWF). *N is a State Machine Workflow net (SMWF) iff N is a Workflow net and a state machine.*

Lemma 0.6.1 implies that for any SMWF the following holds:

$$\forall t \in T : \mid {}^\bullet t \mid = 1 \wedge \mid t^\bullet \mid = 1.$$

This immediately implies the following fact:

Lemma 0.6.8. *Any SMWF is a sound workflow net.*

Proof. First, we prove that any SMWF is 1-sound. Consider a SMWF N with the initial marking i. Any reachable marking in (N, i) consists of 1 token (lemma 0.6.7). Let p be a marking consisting of the only token in an arbitrary place p. N is strongly connected, which means that there exists a path $p, t_1, p_1, t_2, p_2, \ldots, t_n, f$ leading from p to f. It is easy to see that $t_1, \ldots, t_n$ form a firing sequence σ such that $p \xrightarrow{\sigma} f$. Hence, N is a 1-sound net.

Now, let's proof that, for any natural k, N is k-sound. Consider an arbitrary marking M reachable from i^k. Lemma 0.6.7 implies that $\mid M \mid = k$, i.e., $M = p_1 + p_2 + \ldots + p_k$. From the proof above we know that there exist firing sequences $\sigma_1, \ldots, \sigma_k$ such that $p_1 \xrightarrow{\sigma_1} f, \ldots, p_k \xrightarrow{\sigma_k} f$. Then (lemma 0.5.3) $M \xrightarrow{\sigma_1 \ldots \sigma_k} f^k$, i.e., N is k-sound whatever k is taken. Hence, N is sound. □

B Bestandsformaten

Mailbestand (*.mail)

De mailer zal de mail verzenden, waarbij de adressen alleen in de SMTP-envelope worden verstuurd (dus via RCPT TO: voorafgaand aan de DATA, volgens RFC821). In de header-informatie in de mail zelf zijn dus géén adressen aanwezig. Het To-adres wordt gelijk gesteld aan het From-adres. Het From-adres is een mailbox op de mailer machine.

Dit is een mailbestand volgens RFC822 en/of RFC2045.
Het kan hier zowel text- als MIME berichten betreffen.

Minimaal vereiste headers zijn: Subject, From, en To.
Optioneel kan een Reply-To header worden opgegeven. Dit is dan het adres waarnaar antwoorden op het bericht worden verzonden.
From en To dienen gelijk te zijn. Regels eindigen met CR/LF. De headers worden gescheiden van de message body door middel van een lege regel (CR/LF/CR/LF).

Voorbeeld plain text:

```
Subject: Nieuwe producten
Date: Mon, 16 Jun 2003 14:15:03 +0200
From: "Ruud vd Borne" <ruudvdborne@spaarpost.nl>
To: " Ruud vd Borne" <ruudvdborne@spaarpost.nl>

Wij hebben nieuwe producten, kom naar onze website op www.klant.nl
```

Voorbeeld MIME multipart/alternative:

```
Content-Type: multipart/alternative;
      boundary="----_=_NextPart_001_01C331A5.6B63D6EF"
Subject: Nieuwe producten
Date: Mon, 16 Jun 2003 14:15:03 +0200
Message-ID: <8956D84B73461E4AB9550D08F53E131E01812EF0@exchange.kantoor.office>
From: "Ruud vd Borne" <ruudvdborne@spaarpost.nl>
To: " Ruud vd Borne" <ruudvdborne@spaarpost.nl>

This is a multi-part message in MIME format.

------_=_NextPart_001_01C331A5.6B63D6EF
Content-Type: text/plain;
      charset="US-ASCII"
Content-Transfer-Encoding: quoted-printable

Wij hebben nieuwe producten, kom naar onze website op www.klant.nl
=20

------_=_NextPart_001_01C331A5.6B63D6EF
Content-Type: text/html;
      charset="US-ASCII"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>Message</TITLE>
<META http-equiv=3DContent-Type content=3D"text/html; =
charset=3Dus-ascii">
<META content=3D"MSHTML 6.00.2800.1170" name=3DGENERATOR></HEAD>
<BODY>
<DIV><SPAN class=3D880161412-13062003><FONT size=3D2>Wij hebben nieuwe =
```

```
producten,=20
kom naar onze website op <A=20
href=3D"http://www.klant.nl">www.klant.nl</A></FONT></SPAN></DIV>
<DIV><SPAN class=3D880161412-13062003></SPAN> </DIV></BODY></HTML>
=00
------_=_NextPart_001_01C331A5.6B63D6EF-
```

Adressenbestand (*.addresses)

Dit bestand bevat adressen, één adres per regel, waaraan het e-mail bericht zal worden verzonden.

Resultaatbestand (*.xml)

Dit bestand bevat het resultaat van de mailing.

<mailingresults>
    <result code="code">omschrijving</result>
</mailingresults>

Waarbij code:

0 OK
1 Adressenbestand corrupt
2 Mailbestand corrupt
3 Probleem met mailer programma

Dit bestand zal t.z.t. nog worden uitgebreid met extra resultaatcodes en –velden.