Afstudeerverslag

Datum: 4 juni 2010

Versie: 1

Bedrijf
E-mark
Wilhelminastraat 9
2011 VH Haarlem
023- 5518899

Afstudeerder
Michel Dickhoff (60092)
Lokhorst 14
2402 PN Alphen aan den Rijn
michel.dickhoff@hotmail.com
06-25572263

Bedrijfsmentor
Boudewijn Somer
boudewijn@e-mark.nl

Examinatoren
Dhr A. Andrioli
A.Andrioli@hhs.nl
Dhr J. van Peski
J.vanPeski@hhs.nl

Opdrachtgever
Alex Vernikov
alex@e-mark.nl

Samenvatting

Het bedrijf E-mark houdt zich bezig met het ontwikkelen van webapplicaties. Ook levert E-mark support aan klanten als deze vragen hebben over producten.

Een product van E-mark is de Multilogue. De Multilogue kan gezien worden als een zeer uitgebreid forum. De Multilogue die geprogrammeerd is door E-mark is in PHP gedaan. De performance is door de vele toevoegingen door de jaren heen minder geworden. Daarvoor is besloten de Multilogue te herbouwen met als doel een snellere performance. Voor de duidelijkheid, met performance wordt bedoeld de tijd die het kost om pagina's te laden en handelingen uit te voeren. De nieuwe applicatie gaat in ASP.NET met behulp van C# geprogrammeerd worden.

Als ontwikkelmethodiek is RUP gekozen. Het programmeren is opgedeeld in twee iteraties. Aan de hand van de MoSCoW methode is bepaald welke functies er moeten in komen (must haves) en welke erin zouden moeten komen (should haves). De must en should haves zijn geïmplementeerd.

Omdat E-mark al besloten had om de Mulitlogue daadwerkelijk te herbouwen is de inceptiefase kort doorlopen. Er is een plan van aanpak opgesteld met daarin de opdrachtomschrijving en risico's. Groot risico is dat de ASP.NET applicatie niet sneller is dan de PHP variant.

Eisen voor de nieuwe applicatie zijn verkregen door het analyseren van de oude Multilogue en het houden van interviews. Aan de hand van deze eisen zijn er twee databases opgezet. Één voor de Multilogue zelf en één voor de administrator. Voor de implementatie wordt het ASP.NET MVC model toegepast in combinatie met LINQ voor het ophalen van databasegegevens.

Na implementatie is de applicatie getest. Tijdens implementatie zijn er delen van code getest (eenheidstest), er is getest op invalide GET parameters, input vanuit de applicatie (black box testen) en de performance is getest door middel van de Red Gate ANTS profiler.

Tijdens de ontwikkeling van de applicatie zijn er problemen naar voren gekomen zoals het ordenen van reacties en het tonen van mensen die online zijn.

Voor het vervolg van het project kan er nog gedacht worden aan:

1) Compiled queries/Stored procedures
2) Caching
3) Filestream
4) SQL Server indexes
5) Scheduled jobs
6) Webhooks
7) E-mark mail

Voorwoord

Dit afstudeerverslag is geschreven naar aanleiding van mijn afstudeerstage bij E-mark.
Ik heb aan de opdracht gewerkt gedurende de periode 08-02-2010 tot en met 04-06-2010.
Het afstudeerverslag richt zich op het herbouwen van een bestaande applicatie van E-mark waarvan de performance niet meer optimaal is.

De reden dat ik mijn afstudeerstage bij E-mark heb gekozen is omdat het project mij aansprak. Daarnaast sprak de informele bedrijfssfeer mij aan. Verder bood deze stageplaats de mogelijkheid om mij verder te verdiepen in ASP.NET in combinatie met C#.

Ik wil E-mark graag bedanken voor de stageplaats die zij mij hebben aangeboden.
Verder wil ik Boudewijn Somer bedanken voor de begeleiding tijdens de afstudeerstage, Nils Corver voor het ondersteunen van de ontwikkeling en Mehmet Altay voor het verschaffen van informatie over de oude Multilogue.

Michel Dickhoff, 06-04-2010

Inhoudsopgave

Externe bijlagen:

Afstudeerplan, plan van aanpak, analysedocument, functioneel ontwerp, technisch ontwerp, iteratieplan, code, testverslag, stored procedures. Deze zijn geleverd op dvd.

Inleiding

De werkzaamheden die de afstudeerder bij E-mark heeft uitgevoerd stonden in het kader van het afstuderen in het vierde jaar van de opleiding technische informatica aan de Haagse Hogeschool in Delft. Het afstuderen heeft als doel dat de afstudeerder aantoont dat deze op het niveau zit van een beginnend HBO Technisch informaticus.

Dit afstudeerverslag geeft de lezer antwoord op de vraag: "Aan wat voor opdracht is er gewerkt tijdens de afstudeerstage en hoe zijn ontstane problemen opgelost?"

Om van start te gaan wordt er eerst beschreven wat de context van E-mark is en wat de bedrijfsactiviteiten van E-mark zijn (hoofdstuk 1).

De opdracht die door E-mark is opgedragen aan de afstudeerder wordt beschreven in hoofdstuk 2**.** In dit hoofdstuk zijn tevens de op te leveren producten benoemd.
Bij de opdracht heeft de afstudeerder een werkwijze gekozen die past bij de uit te voeren opdracht (hoofdstuk 3***).*** Hier is ook de planning genomen. De werkwijze die het meest geschikt bevonden is voor dit project is RUP.

Hoofdstuk 4 beschrijft de inceptiefase. Hierin is beschreven of het nut heeft om het project uit te voeren of niet.

Hoofdstuk 5 beschrijft de elaboratiefase. Dit hoofdstuk beschrijft de eisen voor de te ontwikkelen applicatie en de systeemarchitectuur. Ook wordt hier uitgelegd hoe de eisen verkregen zijn en hoe de systeemarchitectuur tot stand is gekomen.

De constructiefase beschrijft een aantal interessante componenten die geprogrammeerd zijn. Deze zijn verdeeld in twee iteraties. Hoofdstuk 6 beschrijft deze.

De transitiefase heeft als doel dat het gerealiseerde product overgedragen wordt aan de belanghebbende partij (in dit geval E-mark). Verder is hier de evaluatie opgenomen en de tests zijn beschreven. Hoofdstuk 7 bevat deze informatie.

Hoofdstuk 8 beschrijft de ondervonden problemen. Dit zijn voornamelijk problemen die ondervonden zijn tijdens de implementatie.

Hoofdstuk 9 beschrijft de aanbevelingen/performance verbeteringen die de afstudeerder gedaan heeft of nog gedaan kunnen worden.

1 E-mark

Het hoofdstuk "E-mark" beschrijft het bedrijf waar de afstudeerder aan zijn opdracht heeft gewerkt.

Het bedrijf E-mark houdt zich bezig met het ontwikkelen van webapplicaties. Ook levert E-mark support aan klanten als deze vragen hebben over producten.
Het merendeel van de applicaties die ontwikkelt zijn door E-mark, zijn geprogrammeerd in PHP. Vanaf heden heeft E-mark de ambitie ook thuis te zijn op het gebied van ASP.NET applicatieontwikkeling.

De opdrachtgever is Alex Vernikov. Deze is werkzaam als manager van de support en projectmanagement afdeling.

E-mark is verdeeld in drie onderdelen, te noemen Mail, Services en Labs.
De afdeling Mail houdt zich bezig met e-mail marketing. Zo heeft E-Mark een direct mail applicatie waarmee grote hoeveelheden mails simultaan verstuurd kunnen worden. Deze mailmodule is één van de coreproducten van E-mark. Bedrijven gebruiken de applicatie onder andere om grote hoeveelheden campagnemails en nieuwsletters te versturen.
De afdeling Services houdt zich bezig met zoekmachine marketing, website ontwikkeling en sms marketing. Op de Labs afdeling worden creatieve en innovatieve ICT oplossingen bedacht.
De afstudeerder heeft gewerkt op de afdeling services.

Bij E-mark werken ongeveer 25 mensen. Ongeveer acht mensen programmeren applicaties. De overige mensen beheren lopende projecten of geven support aan klanten.

Naast de direct mail applicatie is ook de Multilogue ontwikkeld door E-mark. De Multilogue is tevens een hoofdproduct van E-mark. In het kort biedt de Multilogue de mogelijkheid om reacties te plaatsen op binnengekomen vragen en op deze wijze te discussiëren met andere gebruikers.

De Multilogue onderscheidt zich van normale forums door de hieronder genoemde mogelijkheden:

1) het ondersteunen van video's
2) meertaligheid
3) gescheiden gebruikerssysteem met verschillende rechten.

Voor meer informatie wordt verwezen naar het analysedocument. Het analysedocument is als een externe bijlage opgenomen.

2 Herbouwen Multilogue

Hoofdstuk “Herbouwen Multilogue” geeft inzicht betreffende de uit te voeren opdracht van de afstudeerder. De probleemstelling en opdrachtbeschrijving met het beoogde doel worden uitgelegd evenals de op te leveren producten.

2.1 Probleemstelling

De huidige Multilogue, eveneens door E-mark ontwikkelt, ondervindt performanceproblemen. Met performanceproblemen wordt de tijd bedoeld die benodigd is om een pagina te tonen aan de gebruiker en om handelingen uit te voeren (zoals profielgegevens opslaan). Deze applicatie is geprogrammeerd in PHP. De nieuw te ontwikkelen Multilogue dient geprogrammeerd te worden in ASP.NET.

De bestaande Multilogue is wat traag in het laden van bepaalde pagina’s (bijvoorbeeld waar reacties geplaatst zijn). Voor gebruikers van de Multilogue is dit onwenselijk omdat dit een frustrerende werking kan hebben. Voor E-mark is dit een probleem aangezien zij met de huidige trage applicatie externe opdrachten en dus omzet kunnen derven.

De performancewinst die behaald wordt door de applicatie te schrijven naar ASP.NET zit niet in de techniek/taal ASP.NET. De reeds bestaande Multilogue is namelijk inefficiënt gecodeerd, ook de klassenstructuur is niet optimaal. De reden dat de code niet optimaal is heeft als grondslag het feit dat er door verschillende mensen geprogrammeerd is zonder de noodzakelijke bijbehorende documentatieopbouw. Gecombineerd met het feit dat E-Mark meer ASP.NET applicaties wil ontwikkelen maakt de keuze om de maatapplicatie te herbouwen verantwoord en is één op één zetting niet mogelijk.
Als kenmerkend voorbeeld gaf E-Mark aan dat bij de laatste installatie van de huidige applicatie voor één van haar opdrachtgevers een cluster van 8 servers benodigd was om de applicatie goed te laten functioneren.
Door de hiermee samenhangende hoge kosten is het gewenst dat de vernieuwde ASP.NET applicatie minder zwaar gaat worden.

Om de performance te verbeteren kan ook de Database die gebruikt wordt voor de huidige applicatie geoptimaliseerd/herbouwd worden. Denk hierbij aan de verschillende normaalsvormen die overbodige data verminderen. Per forum hoort een database die ongeveer 70 tabellen heeft met meerdere Foreign Keys en Primary Keys. Daarnaast is er nog een hoofddatabase die alle andere forumdatabases bevat. In totaal zijn er nu 400.000 entries in de database.

2.2 Doelstelling

De afstudeerder legt de functionaliteit van de oude maatapplicatie vast. Nadat de basisfunctionaliteit is gerealiseerd met ASP.NET wordt er gekeken welke toevoegingen erbij kunnen komen. Bij toevoegingen moet gedacht worden aan functionaliteit die ten gunste komt van de klant of technische aspecten.

Daarnaast is de afstudeerder verantwoordelijk voor het opleveren van zowel het functioneel en technisch ontwerp van de te maken applicatie. Ook produceert de afstudeerder een deel van

de code. Doordat het project te groot van omvang is om door één persoon te laten uitvoeren wordt het project uitgevoerd in teamverband. Het analysedocument, het functioneel ontwerp en het technisch ontwerp produceert de afstudeerder zelf.
Het testen van software omvat het black box testen, unit testen en het testen van de performance. Bij black box worden gegevens in het systeem ingevoerd waarbij gekeken wordt hoe het systeem hierop reageert.
Performance testen wordt gedaan door gebruik te maken van een profiler. Het unit testen is stukjes code apart testen.

Ook gaat er gebruik gemaakt worden van profiling met als doel de code zoveel mogelijk te optimaliseren.

Het doel van de te realiseren Multilogue is om de oude Multilogue te herbouwen in ASP.NET waarbij de code en klassenstructuur wel optimaal geschreven is en tevens sneller is dan de PHP variant van de Multilogue.

2.3 Op te leveren producten

De producten die opgeleverd worden zijn:

- Analyse document van de huidige PHP applicatie
- Functioneel ontwerp (eisen voor ASP.NET applicatie)
- Technisch ontwerp (UML modellen)
- Unit tests
- code
- Aanbevelingsdocument voor ASP.NET applicatie (beschreven in scriptie)

3 Rup

Hoofdstuk 3, “RUP” beschrijft de gekozen methode en de argumentaties voor deze methode. Op basis van de gekozen methode is een globale planning gemaakt met daarnaast een planning voor iteraties.

3.1 Methode

Binnen E-mark is er geen vaste methode om projecten te doorlopen.
De juiste methode wordt verantwoord door middel van een procesanalyse.

Probleem:

Het kiezen van een geschikte ontwikkelmethode voor een juiste projectaanpak.

Analyse probleem:

De afstudeerder heeft het boek “Systeemanalyse en -ontwerp” hiervoor geraadpleegd.
Dit boek beschrijft verschillende methodieken met de voor – en nadelen.

Oplossingen:

Mogelijke methodieken die gebruikt kunnen worden:

- Watervalmethode
- Prototypemethode
- Parallelle ontwikkeling
- Iteratief

Keuze:

Er is gekozen voor het iteratief ontwikkelen. Hieronder wordt verder uitgelegd waarom deze methodiek gekozen is.

De afstudeerder heeft de watervalmethode niet gekozen als methodiek omdat de Multilogue een te omvangrijk project is om deze via de hierboven genoemde methode te voltooien. Een groot nadeel van de watervalmethode is dat als eisen veranderen, de watervalmethode hiervoor niet geschikt is. Er moeten dan stappen teruggenomen worden (als voorbeeld van implementatie naar ontwerp).

Bij de prototypemethode wordt de analyse, ontwerp en implementatie gelijktijdig uitgevoerd. Het voordeel van deze methodiek is dat opdrachtgevers snel een idee krijgen hoe het product eruit komt te zien. Deze prototypen worden “quick & dirty” geschreven. In dit geval past deze methodiek niet bij de opdracht aangezien de doelstelling niet het zo snel mogelijk tonen van een prototype is maar het verbeteren van de performance van de Multilogue.

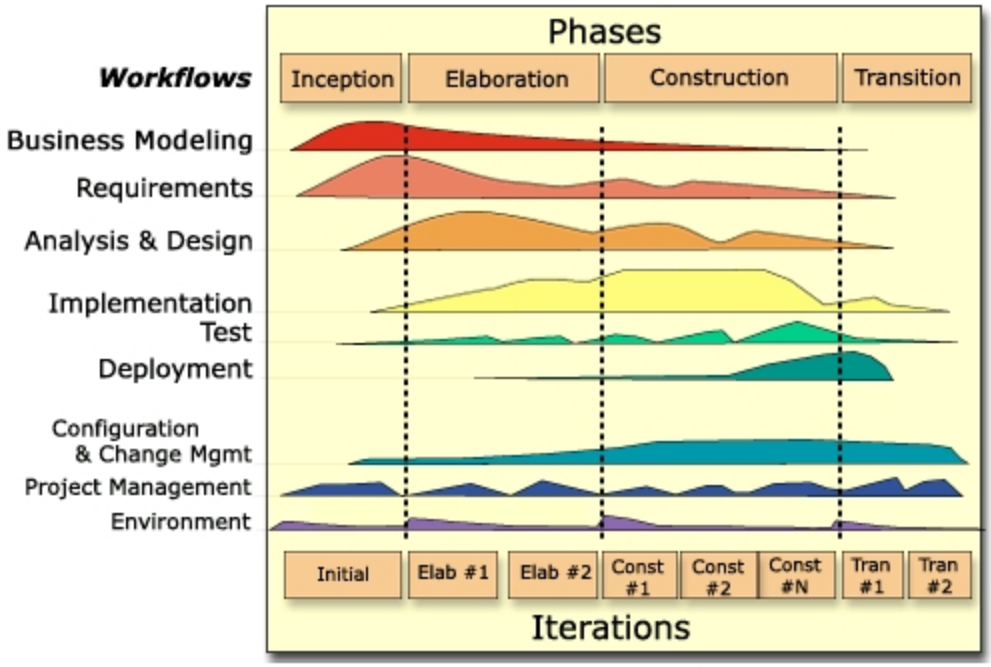
Parallelle ontwikkeling biedt geen oplossing aangezien deze methode het ontwerp onderverdeeld in een aantal subontwerpen. De afstudeerder is verantwoordelijk voor het gehele ontwerp. Ook is deze ontwikkelmethode minder geschikt omdat onderdelen zelden onafhankelijk van elkaar zijn en er derhalve vertraging ontstaat. Bovendien is integratie van de subprojecten lastig.

Doordat de applicatie groot is heeft de afstudeerder gekozen voor het iteratief/incrementeel ontwikkelen. Hierbij wordt het systeem in deeltjes ontworpen wat het ontwikkelproces ten goede komt en dus ook eenvoudiger maakt. Als ontwikkelproces wordt RUP gebruikt. RUP is gebaseerd op een aantal best practices zoals het iteratief/incrementeel ontwikkelen van software.
RUP heeft een viertal fases. Dit zijn inceptiefase, elaboratiefase, constructiefase en transitiefase (afbeelding 1). De fases waar nadruk op gelegd wordt zijn de elaboration- en constructionfase. De inceptiefase wordt niet actief doorgelopen in de zin van het controleren of het project zin of geen zin heeft.

Wat betreft acties wordt er nadruk gelegd op:

- Requirements
- Analysis en Design
- Implementation
- Deel test

Voor meer toelichting wordt er verwezen naar het plan van aanpak.



Afbeelding 1 – Rup fases

Uitwerking:
Zoals hierboven beschreven staat bestaat RUP uit 4 fasen. Hier wordt beschreven wat precies behandeld wordt in welke fase.

Het doel van de inceptiefase is het bepalen of een project uitgevoerd moet worden of niet (dus een go/no go). De inceptiefase levert het plan van aanpak op. Hierin zijn de projectomschrijving, planning en risicofactoren beschreven. Normaliter worden ook de daarmee samenhangende kosten hierin opgenomen. In dit geval zijn de kosten niet vermeld aangezien deze niet aan de afstudeerder beschikbaar zijn gesteld.
Het plan van aanpak is als externe bijlage opgenomen. De milestone van de inceptiefase heet Lifecycle Objective Milestone.

In de elaboratiefase worden de eisen voor de applicatie gespecificeerd (functioneel ontwerp) en de systeemarchitectuur ontworpen (technisch ontwerp). Voor het bepalen van de noodzakelijke eisen ten aanzien van het nieuw te ontwikkelen Multilogue dient eerste de oude Multilogue geanalyseerd worden. Dit heeft het analysedocument tot stand gebracht.

Verder wordt in de elaboratiefase use-case modellen gemaakt en de planning en risicofactoren bijgewerkt. De milestone van de elaboratiefase heet Lifecycle Architecture Milestone.

In de constructiefase worden de verschillende componenten geprogrammeerd. Dit gebeurd in iteraties. De milestone van de constructiefase heet Initial Operational Capability Milestone.

De transitiefase heeft als doel dat het gerealiseerde product overgedragen wordt aan de belanghebbende partij (in dit geval E-mark). In deze fase wordt het product getest en kunnen er trainingen gegeven worden. Verder worden opgedane ervaringen beschreven (evaluatie). De milestone van de transitiefase heet Product Release Milestone.

Het behalen van Milestones is bepaald in overleg met de bedrijfsmentor.

Daarnaast is RUP opgebouwd uit een aantal disciplines, te weten:

1) Business Modeling
2) Achterhalen van eisen
3) Analyse en ontwerp
4) Implementatie
5) Testen
6) Deployment
7) Projectbeheer
8) Omgeving
9) Configuration management
10) Change management

Deze zijn uitgebreid beschreven in het plan van aanpak.

Een probleem bij het gebruik van RUP is dat de afstudeerder hier niet eerder mee heeft gewerkt. Hierdoor was het noodzakelijk dat de afstudeerder zich eerst verdiepte in het RUP proces.

Evaluatie:
De keuze van RUP bleek een verstandige keus omdat naarmate het project vorderde het overzicht bewaard bleef. Ook biedt het door middel van de vier fases een goed framework waarin duidelijk is welke activiteit per fase gedaan moet worden.

3.2 Planning

De globale planning is weergegeven in tabel 1.

Datum	**Activiteit/ stand van zaken**
8 februari / 9 februari	Start plan van aanpak.
10 februari	Feedback pva verwerkt. Plan van aanpak 2e versie af. ** Lifecycle Objective Milestone** gehaald
10 februari – 12 februari	Start analyse.
12 februari	Analysedocument 1e versie af.
15 februari	Feedback analysedocument. Analysedocument 2e versie af.
17 februari	Start Functioneel ontwerp.
19 februari	Plan van aanpak 3e versie (feedback schoolbegeleider) en analysedocument 3e versie opgestuurd naar schoolbegeleider
23 februari	1e versie functioneel ontwerp af.
24 februari	Feedback functioneel ontwerp.
2 maart	Functioneel ontwerp afgerond.
4 maart	Bedrijfsbezoek
5 maart	Feedback verwerken op documenten
2 maart tot en met 9 maart	Uitloop functioneel ontwerp.
10 maart	Start technisch ontwerp.
22 maart	1e technisch ontwerp af zonder prioriteiten en iteraties
24 maart	Prioriteiten eisen bespreken
25 maart	terugkomdag
26 maart	Feedback TO. Technisch ontwerp 2e versie af. ** Lifecycle Architecture Milestone ** gehaald
29 maart	Ontwikkelplan afgerond
30 maart – 27 april	Iteratie 1
27 april – 30 april	Uitloop iteratie 1

3 mei – 17 mei	Iteratie 2 ** Initial Operational Capability Milestone** gehaald
18 mei – 30 mei	Testen, opschonen van code, scriptie schrijven
4 juni	Inleveren scriptie

Tabel 1 – Globale planning

Omdat Rup gebruikt wordt zijn er ook planningen voor de iteraties gemaakt. De planning van Iteratie 1 is weergegeven in tabel 2.

Datum	**Activiteit/ stand van zaken**
30 maart	Tabellen in SQL Server 2008 zetten, Page/Index maken (front-end)
31 maart	Inloggen front-end, inloggen back-end
1 april – 9 april	Toon profiel(front-end), Bezoeken van kamers (front-end), Plaats reacties (front-end)
12- 14 april	Kies taal (front-end), toevoegen van taal (administrator, back-end)
15 april – 23 april	Toevoegen van kamer/vragen (back-end) toevoegen van gebruikers (editor/administrator)
23 april – 27 april	Testen

Tabel 2 – Planning iteratie 1

Iteratie 2 is weergegeven in tabel 3.

Datum	**Activiteit/ stand van zaken**
27 april	Tabellen in SQL server zetten
28 april	Voeg reacties/vragen toe aan favorietenlijst (front-end)
3 mei – 5 mei	Wijzigen profiel (front-end), toon "Wie is online"(front-end)
7 mei – 13 mei	Rapporteer reacties (front-end), Reacties verwijderen/wijzigen (front-end editor)
14 mei -18 mei	Voeg formvelden toe (back-end editor), toon formvelden

Tabel 3 – Planning iteratie 2

4 Inceptiefase

Hoofdstuk 4, “Inceptiefase” beschrijft de genomen stappen in de inceptiefase.

De inceptiefase is door de afstudeerder niet volledig doorlopen. Hiermee wordt bedoeld dat de afstudeerder niet bepaald heeft of het herbouwen van de Multilogue gunstig is voor E-mark. Deze beslissing heeft E-mark zelf gemaakt. Voor E-mark is het een must om de Multilogue te herbouwen vanwege de performanceproblemen die beschreven zijn in hoofdstuk 2.

Het vaststellen van de opdrachtomschrijving, risico’s en planning is in overleg gedaan met Boudewijn Somer. In het kort is de opdrachtomschrijving het herbouwen van de bestaande Multilogue met performancewinst in gedachten.
Wegens de toepassing van RUP is er een globale planning gemaakt met daarnaast de vereiste iteratieplannen. Deze zijn te zien in hoofdstuk 3 (discipline projectbeheer). De iteratieplanningen zijn gemaakt voor de constructiefase.

Het belangrijkste risico is dat de nieuwe ASP.NET Multilogue niet de gewenste performance geeft. Dit risico is getackeld doordat E-mark een test gedraaid heeft in ASP.NET met een niet genormaliseerde database. Hierbij zijn vergelijkbare functies tegenover elkaar gezet. Hieruit bleek dat de ASP.NET functies sneller waren dan de PHP functies.

Dit is te verklaren omdat ASP.NET geoptimaliseerd is en wordt gecompileerd naar een lagere level taal (Common intermediate language, voorheen Microsoft Intermediate Language), dat betekent dat de code die ingevoerd is gereduceerd.
Als een PHP script wordt aangesproken wordt het script “on-the-fly” uitgevoerd.

Voor E-mark is de Multilogue van groot belang. De Multilogue heeft E-mark ongeveer een jaar aan productietijd gekost. Indien E-mark één klant heeft per jaar die de Multilogue in gebruik neemt zijn de kosten die gemaakt zijn voor het maken van de Multilogue gedekt.

In de inceptiefase is de discipline Business Modeling aan bod gekomen.

5 Elaboratiefase

In het begin van de afstudeerstage is de huidige Multilogue geanalyseerd door de afstudeerder.
Hoofdstuk 5, "Elaboratiefase" beschrijft globaal wat er gedaan is in de elaboratiefase.

In totaal is de elaboratiefase opgedeeld in drie delen. Dit zijn het analyseren van de oude Multilogue, het opstellen van de eisen voor de nieuwe Multilogue en het ontwerpen van de database en klassenmodellen.

Bij het opstellen van de eisen is de discipline "Achterhalen van eisen" aan bod gekomen. Het ontwerpen van de database en klassenmodellen valt onder de discipline "Analyse en ontwerp".

5.1 Analyseren oude Multilogue

Het woord Analyse dient men niet te verwarren met de RUP discipline "Analyse en ontwerp". Het doel van het analyseren van de Multilogue is gedaan om de applicatie globaal te leren kennen. Doordat de afstudeerder ook het functioneel en technisch ontwerp voor zijn rekening nam was het van belang om deze fase door te lopen teneinde de opdracht goed te kunnen voltooien.

De te ontwikkelen Multilogue in ASP.NET dient qua performance(snelheid van applicatie) beter te zijn dan de PHP voorganger. Om deze vergelijking te kunnen maken moest eerst de snelheid van de huidige Multilogue vastgesteld worden zodat een goede vergelijking met de ASP.NET variant mogelijk is.
Door het schrijven van bovengenoemd analysedocument is de stap naar het functioneel ontwerp ook kleiner geworden.

Allereerst heeft de afstudeerder een "rondleiding" gehad door Multilogue van één van de hoofdprogrammeurs. Daarna is de afstudeerder de Multilogue zelf gaan testen.

Voor uitgebreidere informatie wordt verwezen naar het analysedocument.

5.1.1 Functionele beschrijving Multilogue

De Multilogue is een maatapplicatie die het mogelijk maakt om gebruikers van de afnemende partij te laten reageren op vragen welke zijn gesteld door andere gebruikers binnen de afnemende partij. De vragen worden geplaatst in zogenoemde kamers. Dit platform is zo ontwikkeld zodat werknemers van het afnemende bedrijf, die zich in verschillende tijdzones bevinden, met elkaar kunnen discussiëren. De Multilogue geeft een gestructureerde weergave van reacties die geplaatst zijn door mensen (wat bijvoorbeeld bij mail niet mogelijk is na honderden replies).

De Multilogue verschilt van een normaal internetforum in het bekijken van video's, reacties/vragen toevoegen aan favorietenlijst, reacties categoriseren/waarderen en de meertaligheid die doorgevoerd is. Ook is er in de Multilogue een duidelijkere scheiding tussen verschillende typen gebruikers.

Er zijn vier type gebruikers te definiëren in de Multilogue. Dit zijn de “front-end” gebruiker (ook wel bezoeker genoemd), moderator, editor en administrator.
De belangrijkste functionaliteit voor de “front-end” gebruiker bestaat uit het bezichtigen van discussiekamers en daar reacties op vragen te plaatsen. Het plaatsen van reacties door de front-end user is eenrichtingsverkeer. Daarnaast heeft de “front-end” gebruiker de mogelijkheid om zijn profiel te zien en deze te wijzigen.
De moderator heeft de taak om gesprekken in goede banen te leiden en reacties te categoriseren/waarderen.

De editor maakt kamers aan met daarbij de vragen in de daarvoor bedoelde back-end. Daarnaast kan de editor polls opstellen en statistieken opvragen.
Een administrator kan enkel inloggen in de “back-end” die voor de administrator bedoeld is. De administrator zet de Multilogue op voor een client. Een administrator is altijd iemand die werkzaam is bij E-mark.

5.1.2 Prestaties Multilogue

Om de prestaties van de oude Multilogue vast te leggen is er o.a. gekeken naar de code, database transacties en de template engine. De reden hiervoor is dat het ophalen van onder andere objecten nauwelijks te optimaliseren is.
Voor het vastleggen van de tijd die de code bezig is wordt er een profiler (Xdebug) gebruikt.

De pagina’s in de front-end die interessant waren om te testen:

- De “Who is online” pagina (43 mensen online).
- Een discussiekamer met gevuld met 100 reacties.

De uitvoersnelheid was respectievelijk 400 milliseconde en 800 milliseconde. Wat bleek is dat er veel tijd verloren ging aan het ophalen van data uit de database. Ook het vullen van de template was voor beide gevallen een tijdrovende handeling. Voor de performance van andere pagina’s wordt verwezen naar het analysedocument.

5.2 Achterhalen van eisen

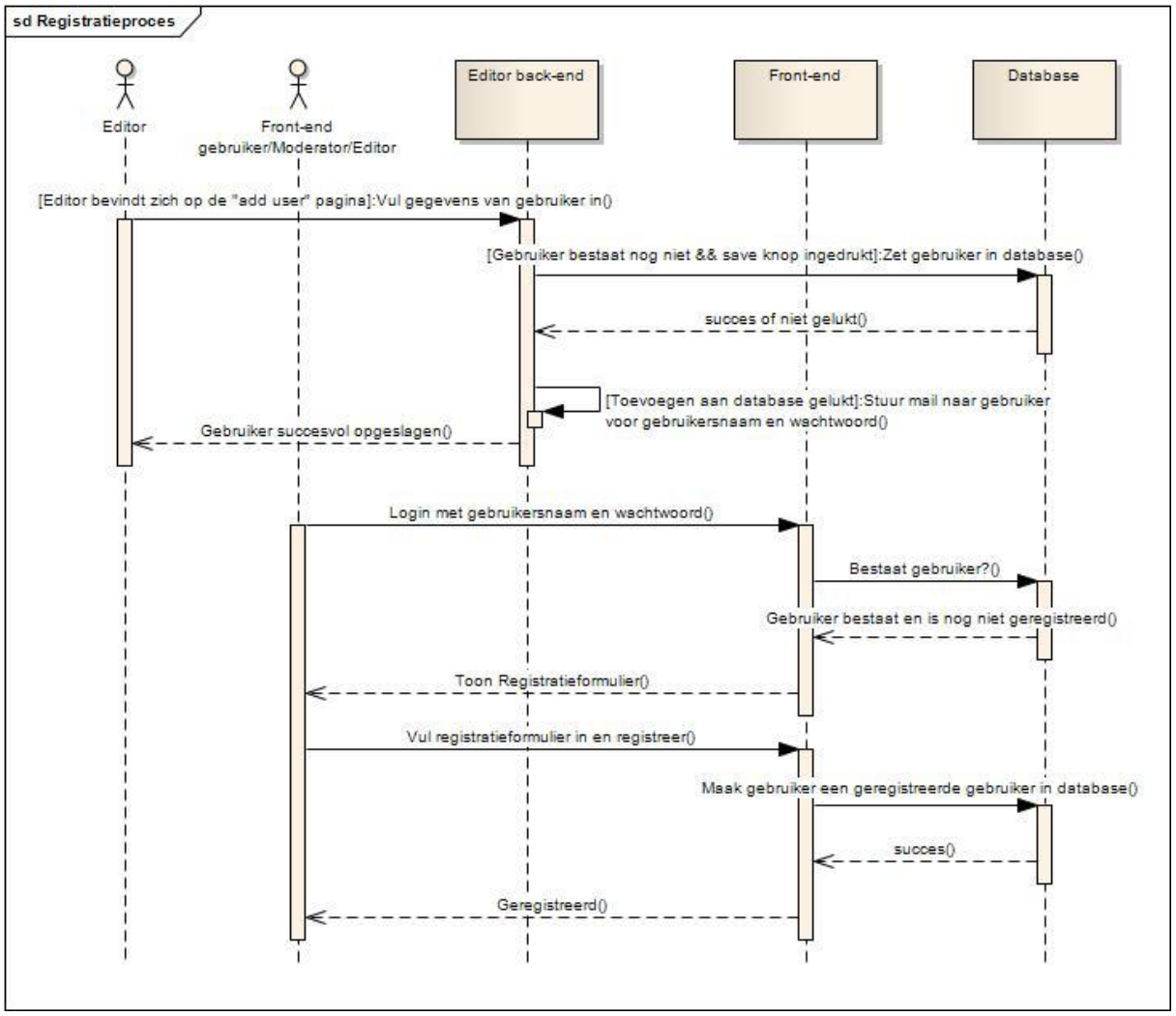
Hoofdstuk 5.2, “Eisen vaststellen” beschrijft de eisen die vastgelegd zijn waaraan het te ontwikkelen systeem moet voldoen. Deze eisen zijn uitgebreid beschreven in het functioneel ontwerp met behulp van use case scenario’s.

De te ontwikkelen applicatie bestaat uit drie componenten:

- Front-end, voor ieder type gebruiker. Hier zijn onder andere reacties te lezen en te plaatsen en de mogelijkheid om het profiel te wijzigen.
- Editor back-end, alleen toegankelijk voor editors en administrators. Editors zetten hier polls, kamers en vragen op.
- Administrator back-end, enkel toegankelijk voor administrators. Een administrator zet een Multilogue op met de juiste instellingen.

De eisen zijn achterhaald door de oude Multilogue door te nemen. Daarnaast is er meerdere malen een hoofdprogrammeur van de oude Multilogue geïnterviewd.
De belangrijkste functies per gebruiker zijn weergegeven in bijlagen 1 – 4 in use case diagrammen.
Bijlage 5 en 6 tonen de functionele en niet functionele eisen die door het interview en analyse van de Multilogue naar voren zijn gekomen.

Een belangrijk proces dat in kaart kwam tijdens een interview was het proces voor het registreren vanuit de back-end van de editor. Dit was een lastig in kaart te brengen proces aangezien deze functionaliteit behoorlijk “verstopt” zat in de Multilogue.
Zie afbeelding 2 voor het sequentiediagram van het registreerproces.



Afbeelding 2 - Registreerproces

5.3 Database ontwerp

Hoofdstuk 5.3, "Database ontwerp" beschrijft de specifieke keuzes die gemaakt zijn tijdens het ontwerpen van de database.

5.3.1 Inventarisatie database

Doordat de applicatie uit drie componenten bestaat (front-end, editor back-end en administrator back-end) is er gekozen om gebruik te maken van gescheiden databases. De front-end en back-end gegevens zijn opgenomen in één database, de administrator gegevens in een aparte database. Het voordeel hiervan is dat ook beide systemen afzonderlijk van elkaar kunnen draaien zonder dat de databasestructuur van de andere applicatie erbij zit.

Omdat het administratorgedeelte enkel voor E-mark medewerkers toegankelijk is kan deze bijvoorbeeld door E-mark zelf gehost worden.
Ruimtebeslag op de server voor de Multilogue-applicatie is derhalve minder.

Als relationeel databasebeheersysteem is er gekozen voor SQL Server 2008 van Microsoft. Dit omdat E-mark hier bekend mee is en het meegeleverd wordt met Visual Studio.

Als encryptie voor wachtwoorden is er gekozen voor SHA1. Deze is lastiger te omzeilen dan MD5 en wordt goed ondersteund.

Vaststelling welke data moet worden opgeslagen in de database wordt beschreven aan de hand van procesanalyses.

Dit hoofdstuk beschrijft drie procesanalyses:

1) Welke data moet opgeslagen worden in de database
2) Hoe bestanden op te slaan
3) In welke normaalvorm moeten de tabellen staan

Bijlagen 7 tot en met 14 geven de tabellen weer.
Het document "Technisch ontwerp" beschrijft de tabellen inhoudelijk.

ad1 Welke data moet opgeslagen worden in de database

Probleem:
Welke data moet er opgeslagen worden in de database?

Analyse probleem:
Oplossingen zijn gevonden door te overleggen met Mehmet Altay en Nils Corver, beide in dienst bij E-mark als programmeur.

Oplossingen:
1) Analyse van oude Multilogue database
2) Analyse van oude Multilogue applicatie
3) Interview met hoofdprogrammeur van oude Multilogue

Keuze:
Er is gekozen om gebruik te maken van alle bovengenoemde oplossingen. De nadruk lag echter op het analyseren van de oude Multilogue applicatie en het bijbehorende interview met de hoofdprogrammeur. Dit omdat de database van de oude Multilogue niet perfect is en daardoor misschien beïnvloed wordt door de structuur.

Uitwerking:
Gegevens zijn logisch geordend. Dat betekent dus per gebruiker/kamer/reactie enzovoort. Tijdens het samenstellen van de tabellen trad er een ander probleem op. Hoe moeten bestanden zoals plaatjes en video's opgeslagen worden. Dit is beschreven in ad2. Daarnaast is het bepalen van de juiste normaalvorm beschreven in ad3.

Evaluatie:
De keuze die uitgewerkt was bleek een goede keuze te zijn. Achteraf bleek maar één koppeltabel te missen die benodigd was voor het opslaan eigen gemaakte velden (formfields). Dit was tabel Listitem_User.

ad2 Hoe bestanden op te slaan

Probleem:
Waar moeten bestanden zoals plaatjes en video's opgeslagen worden?

Analyse probleem:
Research gedaan via internet en overleg gepleegd met Nils Corver.

Oplossingen:
1) Bestanden op slaan in de database
2) Bestanden buiten de database opslaan

Keuze:
SQL Server 2008 ondersteunt het opslaan van bestanden in een tabel. Dit kan gerealiseerd worden door gebruik te maken van het type varbinary(MAX). Het bestand wordt dan opgeslagen als een BLOB (Binary Large Object). Het opslaan van bestanden in de database is de eenvoudigste manier om het probleem op te lossen. Het is eenvoudig omdat de bestanden niet extern opgeslagen worden en daarom ook geen bestandlocaties en bestandsnamen bijgehouden hoeven te worden.

Nadelen van afbeeldingen en video's opslaan in de database zijn dat exports van de database niet meer mogelijk zijn door de enorme grootte. Ook zijn de bestanden niet gegroepeerd dus moeilijk te onderhouden. Misschien wel het grootste nadeel is dat bij een groot aantal bestanden de performance van de database aanzienlijk daalt bij het selecteren van rijen.

Om deze redenen is ervoor gekozen om bestanden buiten de database op te slaan.

Uitwerking:
De tabel File is aangemaakt met daarin een URL en bestandsnaam opgenomen. De URL verwijst naar de locatie waar het bestand opgeslagen is. Voordeel van het opnemen van een URL in de tabel en niet enkel als een configuratiewaarde is dat het op deze manier mogelijk is om bestanden over meerdere locaties/servers te verspreiden.

Naast het handmatig opslaan van bestanden en dit bijhouden in de tabel kon er ook gekozen worden voor het gebruik van het type Filestream. Het type Filestream is toegelicht in hoofdstuk 6.3 en hoofdstuk 8.3.

Evaluatie:
Na het bovengenoemde geïmplementeerd te hebben bleek het geheel goed te werken en was de performance eveneens goed.

ad3 In welke normaalvorm moeten de tabellen staan

Probleem:
Het kiezen van een goede normaalvorm zodat de tabellen overzichtelijk zijn maar met zo'n min mogelijke performanceverlies.

Analyse:
De afstudeerder heeft een aantal papers op internet gelezen over normaalvormen en het dictaat van Henk van den Bosch gebruikt voor de verschillende normaalvormen.

Oplossingen:
1) Eerste normaalvorm
2) Tweede normaalvorm
3) Derde normaalvorm
4) Boyce codd normaalvorm

Keuze:
Het is bekend dat naarmate de normaalvorm hoger wordt de performance voor het selecteren van rijen over meerdere tabellen minder wordt. Hierdoor is een zo laag mogelijke normaalvorm interessant.
Het niet normaliseren betekent dat er repeating groups aanwezig zijn. Dit komt het overzicht niet ten goede. Bij het normaliseren naar de eerste normaalvorm zijn de repeating groups niet meer aanwezig. Wel is er nog redundantie aanwezig omdat er niet gesplitst wordt in tabellen maar een combinatie van kolommen wordt gebruikt als Primary key. Dit is opgelost door het normaliseren naar de tweede normaalvorm waarbij de niet-sleutel attributen functioneel afhankelijk zijn van de volledige sleutel. Er is nog steeds redundantie aanwezig (zoals bijvoorbeeld tabel in tabel Content met een foreign key naar Page). Het is namelijk mogelijk dat er meerdere content blokken op een pagina geplaatst worden. Doordat de tabellen qua duidelijkheid overzichtelijk genoeg zijn is er besloten niet verder te normaliseren vanwege performance.

Uitwerking:
Tabellen zijn in tweede normaalvorm behouden.

Evaluatie:
Het evalueren van de normaalvorm is lastig. Er is op basis van theorie een normaalvorm gekozen. Qua duidelijkheid is de tweede normaalvorm voldoende. Het is maar de vraag als de tabbellen in eerste normaalvorm stonden of dit ook daadwerkelijk een merkbare performancewinst geeft.

5.3.2 Datatypen verantwoording

Door het gebruik van SQL Server 2008 is er zorgvuldig omgesprongen met datatypen om ruimte te besparen en de performance te verbeteren.

De elementen(rijen) in tabellen moeten uniek genummerd zijn. SQL Server 2008 biedt een UNIQUEIDENTIFIER type en integers (identity). Een UNIQUEIDENTIFIER slaat een 32 lange karakter reeks op. Zoals:
45E8F437-670D-4409-93CB-F9424A40D6EE.
Een integer die gebruikt wordt als unieke waarde is een nummer dat automatisch opgeteld kan worden.
Er is gekozen om integers te gebruiken voor unieke kolommen. Dit omdat het opslaan van een normale integer 4 bytes in beslag neemt tegenover 16 bytes van de UNIQUE IDENTIFIER. Ook is de UNIQUE IDENTIFIER niet incrementeel ophoogbaar.

De kracht van de UNIQUE IDENTIFIER is dat het echt ook uniek is (uniek in de zin van het enigste karakterreeks in de wereld). In het geval van de Multilogue is dat overbodig. Waar het wel nuttig kan zijn is bijvoorbeeld bij het samenvoegen van twee databases. Dat is hier niet het geval. Ook bij het toevoegen van een nieuwe rij in een tabel wordt de UNIQUEI DENTIFIER niet automatisch gegenereerd. Dit maakt het integertype als PRIMARY KEY tot de betere keus.

Wat betreft integertypen is er gekozen voor:

Data type	**Range**	**Opslagruimte**
Smallint	-2^15 (-32,768) tot 2^15-1 (32,767)	2 bytes
Int	2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)	4 bytes

Naast Smallint en Int is er ook Tinyint. Deze loopt tot en met 255. Hierdoor wordt echter snel de limiet bereikt en kan deze onverwachts fouten opleveren bij overschrijding.
Zo is er bij het toevoegen van talen/of filecategorieën gekozen voor een Smallint.

Daarnaast is er gekozen om data op te slaan als smalldatetime. Het verschil met datetime zit in de nauwkeurigheid. De range van smalldatetime ligt tussen 1 januari 1990 tot 6 juni 2079 en is nauwkeurig op de minuut. Hier staat tegenover dat deze waarde opgeslagen wordt als twee 2 -byte integers in tegenstelling tot datetime (twee 4-byte integers).

In de meeste gevallen is voor het opslaan van tekst gekozen voor nvarchar. De keuze bestond uit char, varchar, nchar en nvarchar.

Het verschil tussen char en varchar ligt met name in de wijze van opslag van data.
In beide gevallen moet er een lengte opgegeven worden. Als voorbeeld char(20), hier kunnen maximaal 20 karakters worden. De char(20) neemt ook altijd 20 bytes in beslag, zelfs als er data opgeslagen wordt van maar vijf karakters. De varchar(20) past zich aan de hoeveelheid karakters die de string heeft. Hierbij zijn er wel twee extra bytes benodigd voor de referentie van grootte voor SQL Server. Doordat het aantal karakters enorm kan variëren is er gekozen om een variant van varchar te gebruiken, namelijk nvarchar. Doordat de Multilogue meertalig is kunnen er tekens opgeslagen worden (niet Engelse letters) die in varchar niet goed opgeslagen kunnen worden. Nvarchar slaat de gegevens als unicode data. Hierdoor is het mogelijk om vreemde karakters op te slaan zoals “ç”. Ieder karakter kost alleen wel 2 bytes aan opslag.

5.4 Klasse ontwerp

Hoofdstuk 5.4, “Klasse ontwerp” beschrijft de keuzes die gemaakt zijn tijdens het ontwerpen van het klassenmodel. Naast de technieken die beschreven worden, is ook de globale aanpak verwoord en zijn er twee interessante voorbeelden gegeven die tijdens het ontwerp naar voren kwamen. Sequentiediagrammen zijn gegeven en beschreven in het technisch ontwerp.

5.4.1 Technieken

Er is gekozen om het ASP.NET MVC model toe te passen voor de Multilogue. Het voordeel van het ASP.NET MVC model tegenover de webforms is dat het MVC model de code scheidt van het grafische gedeelte.
Dit heeft als voordeel dat het testen van code eenvoudiger gaat. Hierdoor is Test Driven Development (TDD) mogelijk.

Een ander voordeel is dat het ASP.NET MVC volledige controle geeft over de HTML. Dit omdat templates niet door slepen in elkaar worden gezet, maar door zelf de HTML te typen.

Ook biedt het MVC model een framework dat verschillende componenten gescheiden houdt. Dit zijn de layout (views), controllers (voor het aansturen van views en data opvragen uit de modellen) en de modellen voor het opvragen van de data uit de database.

ASP.NET MVC is in beeld gekomen doordat Boudewijn Somer (bedrijfsmentor) aanraadde om er eens naar te kijken. Het MVC model is vergeleken met het traditionele ASP.NET met webforms. Webforms is het best te vergelijken met hoe een grafische Windows applicatie in elkaar gezet wordt. Dit geschiedt door objecten te verslepen en door middel van events bepaalde handelingen uit te voeren. Aan de .ASP pagina zit een C# bestand vast die events afhangt en data opvraagt uit de database. Wat dan nog nodig is het aanbrengen van een lagenstructuur. Ook het testen van code is niet optimaal aangezien alle code per pagina aangebracht wordt. Daardoor prefereert de afstudeerder het MVC model.

Daarnaast is de vraag hoe database gegevens opgehaald moeten worden. Dit kan met ADO.NET. ADO.NET levert klassen waarmee het mogelijk is om data op te halen uit databases. Alternatieven zijn het gebruik van ORM software. Een voorbeeld hiervan is LINQ, LINQ is een Microsoft .NET Framework component. LINQ wordt tegenwoordig veel gebruikt bij ASP.NET applicaties vanwege het gemak en leesbaarheid van de queries. De afstudeerder heeft dit zelf ook ondervonden en daarom krijgt LINQ de voorkeur boven het gebruik van ADO.NET.

LINQ genereert automatisch modellen (classes) van iedere tabel in de database. Elk attribuut wordt als een private variabele neergezet die een set en get toelaat. Doordat de classes als partial geïmplementeerd zijn kan eenvoudig een ander bestand aangemaakt worden die deze klasse uitbreidt met eigen functies. Voordelen hiervan zijn dat LINQ daadwerkelijk objecten maakt van de rijen van een tabel. Als hier wijzigingen plaatsvinden, houdt LINQ dat bij en indien gewenst kan er door middel van één commando deze wijzigen doorgevoerd worden. Ook een aangename functionaliteit is dat LINQ associaties overneemt uit het databasemodel. Hierbij is het eenvoudig om leesbare queries te typen wat soms niet het geval is met “normale SQL” (denk aan sub queries).

Bij het gebruik van LINQ is er overhead doordat de LINQ syntax naar syntax vertaald moet worden die de database begrijpt (T-SQL). Door van een query een compiled query te maken die static is hoeft deze query maar eenmalig geëvalueerd te worden. Daarna kost het geen overhead meer.
Ook stored procedures behoren tot de mogelijkheid met LINQ. Voordeel van stored procedures is dat hier iets meer tijdwinst behaald kan worden dan gecompileerde queries. Hier is onderzoek naar gedaan, bij een insert van 10 rows bleek dat de normale linq query 6 ms nodig had, een compiled query 2.3 ms en een stored procedure 1.7 ms.

Dit komt omdat er bij storedprocedures geen vertaalslag aan te pas komt. Er wordt enkel een functieaanroep gedaan op de database in plaats van de hele query te versturen via het netwerk. Nadeel van stored procedures is dat er een onderhoudspunt bij komt. Daarbovenop moeten mensen van E-mark stored procedures snappen en deze zelf ook kunnen onderhouden. Om deze reden worden stored procedures niet voor iedere query gebruikt.

5.4.2 MVC URL Routing

Doordat er gebruik gemaakt wordt van het ASP.NET MVC model werkt het opvragen van een pagina anders dan in andere frameworks.

In veel web frameworks (ASP, PHP…) verwijzen inkomende URLs naar bestanden die opgeslagen zijn op de harde schijf. Zo is er voor "/Products.aspx" een bestand dat diezelfde naam heeft. Hiervan wordt de code uitgevoerd en stuurt deze HTML terug naar de client.

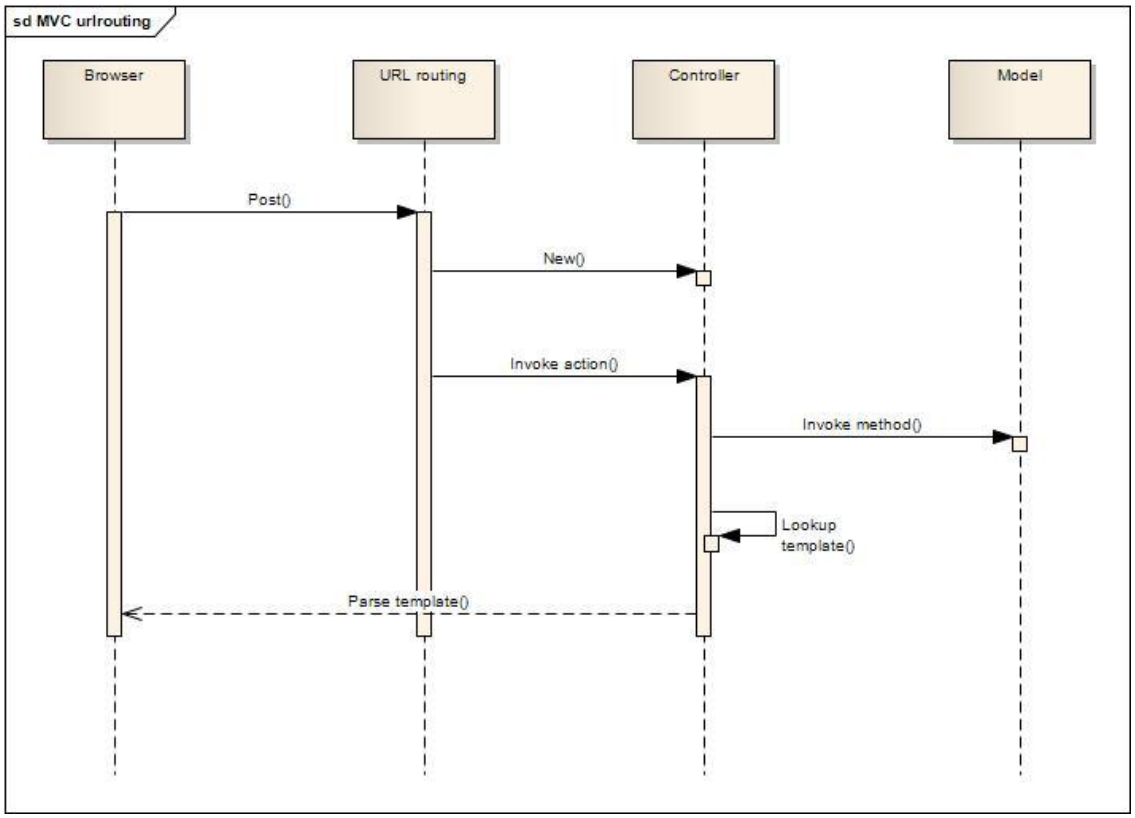
Met het ASP.NET MVC framework verwijzen de URLs niet naar bestanden op de harde schijf maar direct naar klassen . Deze klassen zijn controllers en handelen inkomende verzoeken en userinput af. Dit noemt men de URL routing.

Als een gebruiker de link /Products/Detail/product_id intypt wordt controller ProductsController met functie Detail aangesproken met parameter product_id.

In het kort zorgt de URL routing zorgt voor twee dingen:

- Bij een inkomende URL naar de applicatie wordt ervoor gezorgd dat de juiste actie van de juiste controller uitgevoerd wordt.
- Uitgaande URLs die gebruikt kunnen worden als call backs worden naar de juiste controller en actie geloodst.

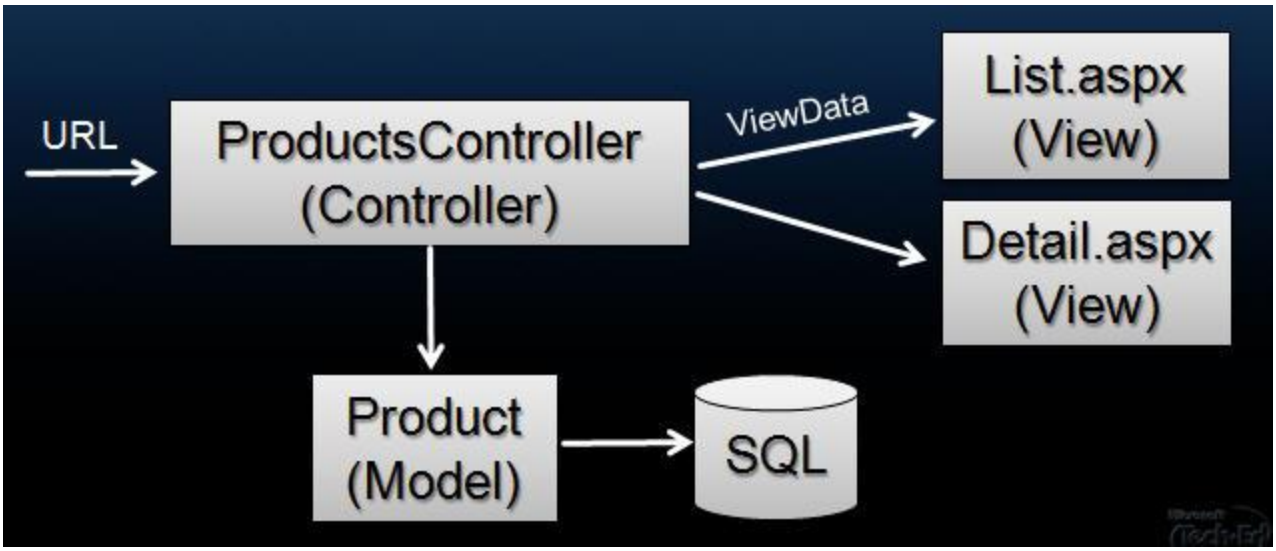
Afbeelding 3 toont een inkomende URL afhandeling.



Afbeelding 3 – inkomende URL

Wat opvalt is dat er geen view object voorkomt in het sequentiediagram. Dit komt doordat views geen objecten zijn maar enkel HTML files. Een HTML file wordt geladen met de functie "lookup template". Deze wordt terug gezonden naar de browser van de client.

Afbeelding 4 toont een minder technische afbeelding van een URL afhandeling in het MVC framework.



Afbeelding 4 – URL afhandeling

5.4.3 Aanpak ontwerp

Om tot een goed gestructureerd ontwerp te komen zijn er een aantal stappen genomen.
De eerste stap is het maken van Sitemaps. Deze zijn beschreven in bijlage 15, de term sitemap staat uitgelegd in de bijlage.
De sitemaps geven weer welke pagina's bezichtigbaar zijn voor de bezoeker. Deze zijn opgenomen als controllers.

De Controller reageert op browser calls, die vanuit handelingen van de gebruiker tot stand komen en roept daarbij de juiste actie aan. Namen van functies zijn zo eenvoudig mogelijk gehouden en in de context van de functie. Denk hierbij aan eenvoudige Add/Edit/Update namen.
Er zijn drie groepen controllers, namelijk:

- Controllers voor de front-end (bijlage 16)
- Controllers voor de editor back-end (bijlage 17)
- Controllers voor de adminstrator back-end (bijlage 18)

Deze controllers zijn beschreven in het technisch ontwerp.

Niet alle functies van controllers zijn ook verbonden aan een HTML bestand. Sommige functies voeren bepaalde handelingen uit en verwijzen vervolgens naar een andere functie van een controller. Voorbeeld hiervan is de FavoriteController. Deze controller voegt reacties of vragen toe aan een favorietenlijst en doet vervolgens een redirect naar de voorgaande actie van de desbetreffende controller (question/view, deze geeft de reacties weer van een vraag).

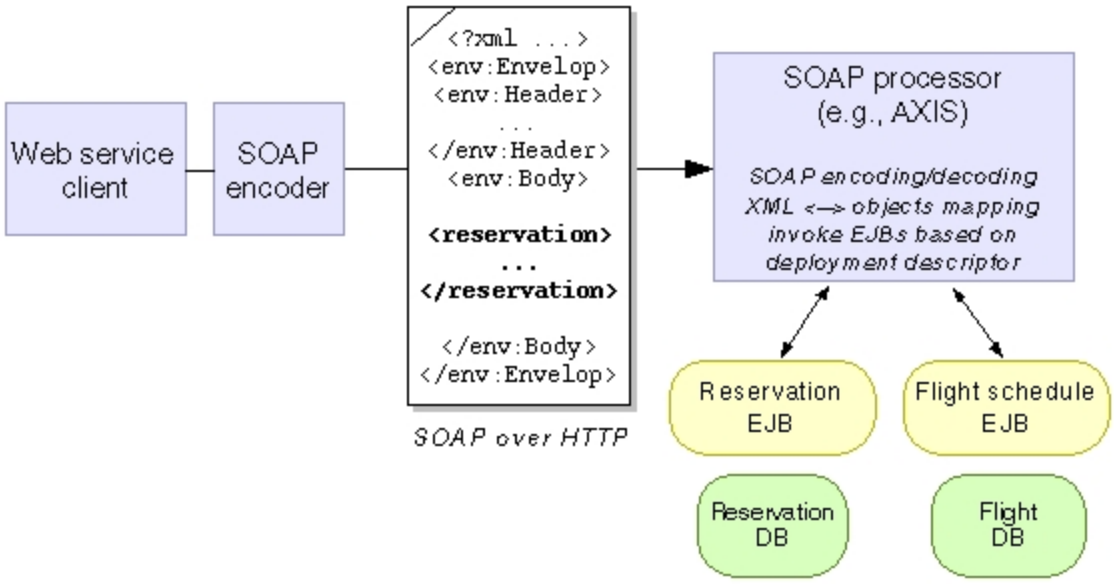
De modellen zijn onderverdeeld tesamen met de helperfuncties als een externe library. In totaal zijn er twee libraries, dit zijn de Multilogue lib (voor front-end en back-end editor) en de Administrator lib (voor administrator back-end).
In het kort gezegd zorgt de datacontext voor het genereren van de SQL queries op basis van de LINQ queries. Ook zet de datacontext rijen van data vanuit de database om in objecten(models).

De modellen zijn weergegeven in bijlage 19 (front-end/editor back-end) en bijlage 20 (administrator). De helper functies van de Multilogue zijn weergegeven in bijlage 21.
De modellen zijn beschreven in het technisch ontwerp.

5.4.4 Communicatie tussen administrator en Multilogue

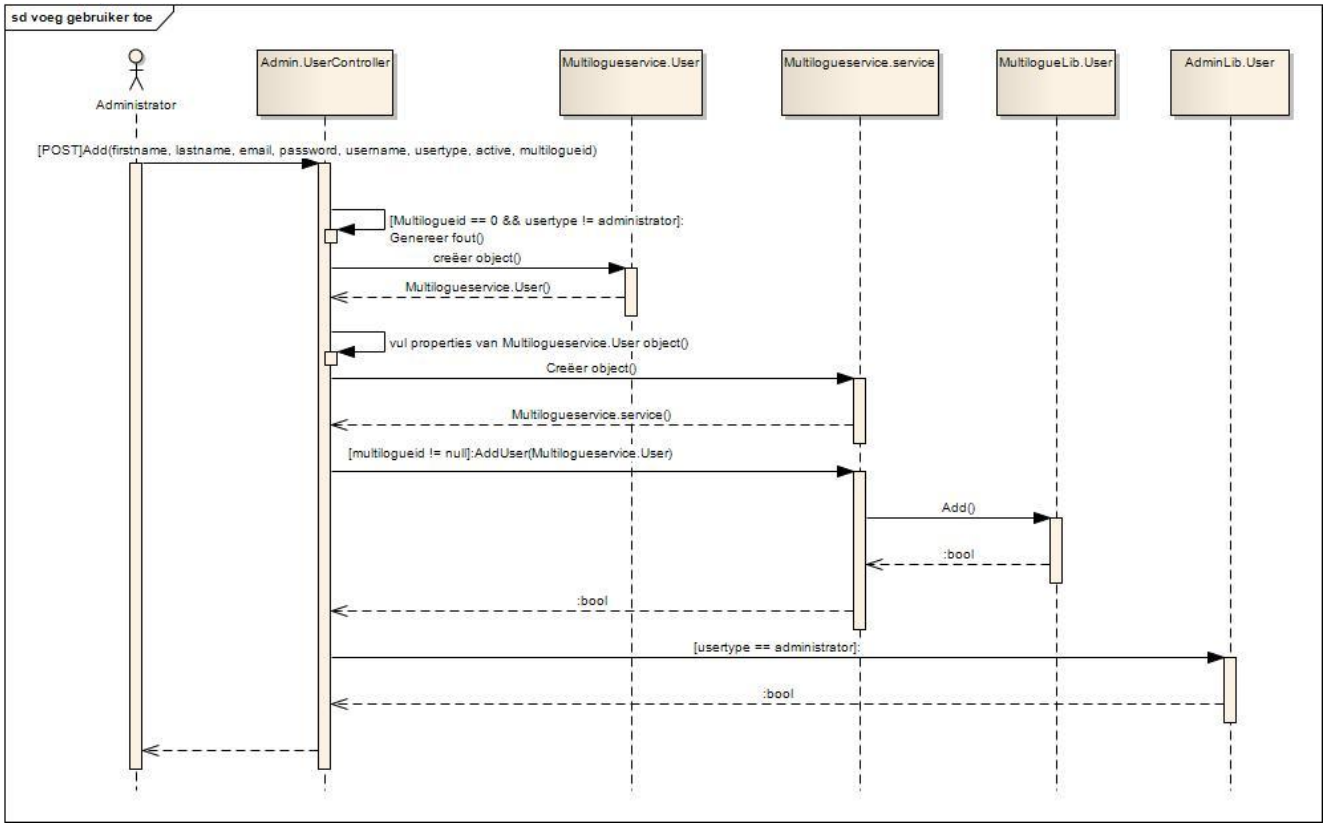
Het laten communiceren van twee ASP.NET applicaties was nieuw voor de afstudeerder. Er is onderzoek via internet gedaan naar verschillende manieren om deze functionaliteit te implementeren. De reden dat het onderzoek in de elaboratiefase is gedaan in plaats van in de constructiefase is dat het dan sneller duidelijk wordt hoe de communicatie precies verloopt.

Veel moderne applicaties gebruiken RPC. RPC is in dit geval geen optie, omdat het HTTP protocol hier niet voor ontworpen is. Het probleem met RPC is dat veel firewalls en proxy servers dit soort verkeer blokkeren.
Een andere manier voor communicatie tussen verschillende applicaties is SOAP. Deze communicatie geschiedt via HTTP. Omdat HTTP ondersteund wordt door iedere browser en server is SOAP een betere keuze dan RPC. Afbeelding 5 toont in het kort de werking van SOAP.



Afbeelding 5 –Globale Werking Soap

Een interessante functie vanuit de administrator back-end is het toevoegen van gebruikers aan een Multilogue. Zie afbeelding 6 hoe de communicatie tussen objecten verloopt om het toevoegen van een gebruiker te realiseren. Implementatie van het SOAP gedeelte is weergegeven in hoofdstuk 6.2.3.



Afbeelding 6 – Communicatie door middel van SOAP

De administrator is ingelogd en heeft de gegevens van de gebruiker ingevuld (zie bijlage 22 voor het formulier). Voor het overzicht is de vereiste voorwaarde dat de administrator ingelogd moet zijn en de datatypen van de parameters zijn weggelaten. Alle parameters zijn van het type string. De parameters firstname, lastname, email en password spreken voor zich. Parameter username is optioneel, de username kan in de toekomst gebruikt worden voor het inloggen in plaats van het gebruik van een e-mail adres (allebei in combinatie met een wachtwoord). Het usertype bevat informatie of de toe te voegen gebruiker een bezoeker, editor, moderator of administrator is. De multilogueid geeft weer aan welke Multilogue de gebruiker toegevoegd moet worden. Er kan ook gekozen worden om dit veld leeg te laten, maar dan moet de toe te voegen gebruiker van het type administrator zijn. Want enkel gebruikers met gebruikerstype administrator worden opgeslagen in de administrator back-end. Dus het toevoegen van een niet administratortype zonder het kiezen van een Multilogue genereert een fout.
Parameter active geeft aan of de gebruiker, indien deze inlogt in de front-end nog gegevens moet wijzigen (extra informatie) op de profielpagina.

Omdat de connectie via SOAP verloopt is er een speciale klasse gemaakt die verstuurd wordt naar de Multilogue applicatie. Er kon ook gekozen worden voor het doorgeven van alle variabelen. Maar op de lange termijn is dit minder flexibel als er een gegeven bij moet komen. Dat betekent dat ook de functie van de SOAP service een parameter erbij krijgt. In het geval van een object hoeft er enkel een private variabele bijgemaakt te worden.

De SOAP connectie wordt gerealiseerd met een apart project. Een zogenaamde webservice.

Als de Multilogueid niet null is, dus met andere woorden de gebruiker moet toegevoegd worden aan een Multilogue dan wordt de desbetreffende SOAP connectie aangemaakt (met juiste URL). Daarna wordt functie AddUser aangesproken. Van hieruit wordt de MultilogueLib.User class aangemaakt en functie Add aangesproken. De functie Add controleert de waardes die ingevoerd zijn. E-mail wordt gecontroleerd door middel van een reguliere expressie en andere waardes moeten langer zijn dan x aantal karakters. Deze waarde is in te stellen. De functie Add returnt een bool. De waarde is false als één van de ingevulde waardes niet voldoen aan de eisen of als de gebruiker al bestaat (e-mail bestaat al in de Multilogue database).

Na het toevoegen van de gebruiker wordt er gekeken of de gebruiker van het type administrator is. Is dit het geval en de administrator bestaat nog niet dan wordt deze toegevoegd in de administrator database. De administrator zelf ziet welke handelingen gelukt zijn en welke niet.

5.4.5 Formvelden

Een interessante functie is het aanmaken van Formvelden. Een Formveld is niets anders dan een zelf gemaakt veld die een editor aan kan maken zodat een gebruiker deze in kan vullen op zijn/haar profielpagina. Denk hierbij aan aantal huisdieren, geboortedatum of andere informatie die het bedrijf die Multilogue gebruikt wil weten.

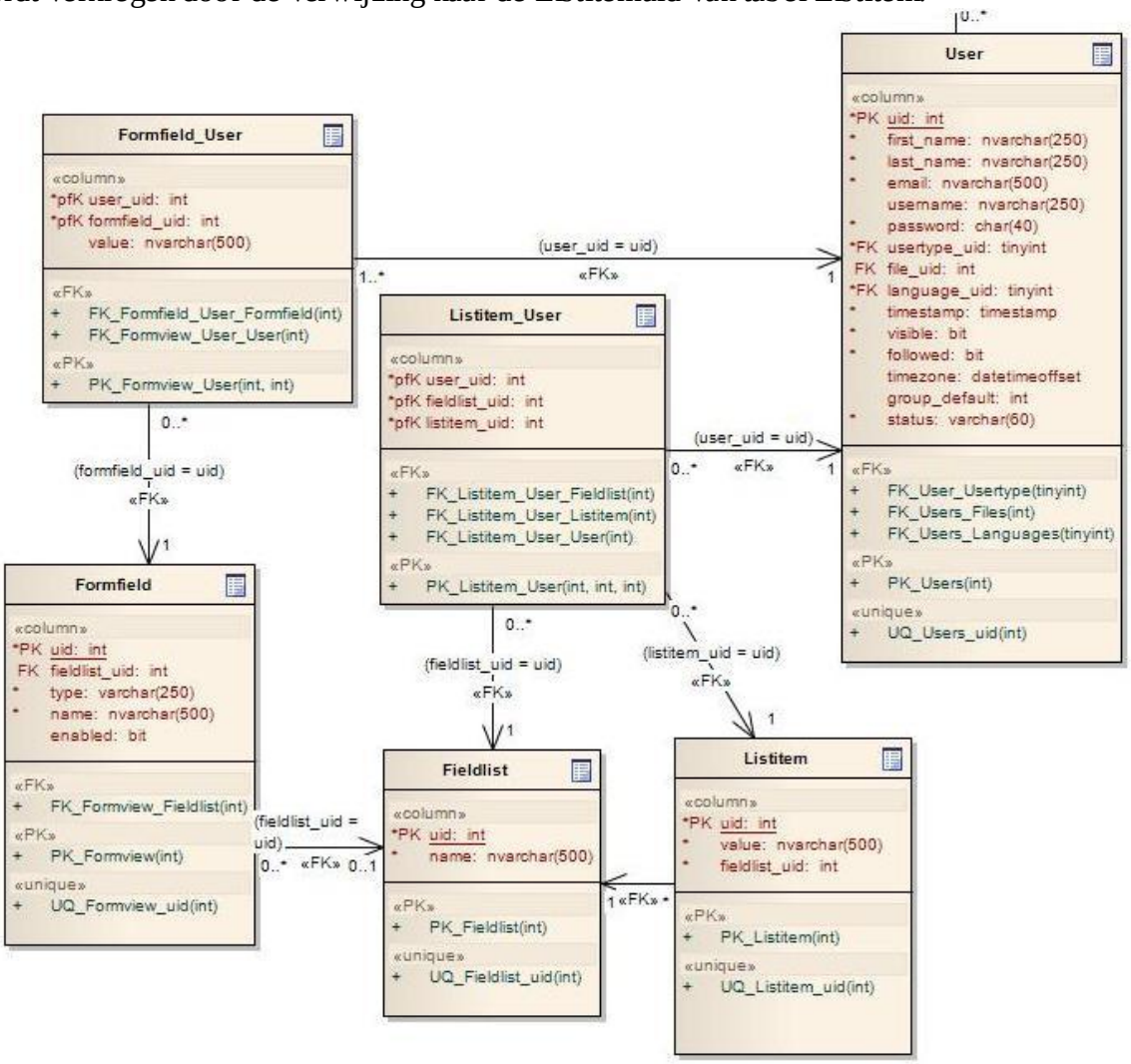
Het lastige eraan is dat er verschillende soorten formvelden zijn. Aan de hand van afbeelding 7 wordt uitgelegd hoe formvelden opgeslagen worden. In totaal zijn er vier verschillende soorten formvelden. Dit zijn tekst, dropdownbox, checkbox en radiobuttons.

Functioneel verschillen deze formvelden nogal van elkaar. Bij een tekstveld kan de gebruiker zelf een waarde invullen. Bij een dropdownbox en radiobuttons kiest de gebruiker één optie uit een lijst met opties. De dropdownbox en radiobuttons verschillen niet qua functionaliteit maar qua layout. Bij het gebruik van een checkbox zijn meerdere antwoorden mogelijk om in te vullen.

Indien een editor een formveld toevoegt wordt deze opgeslagen in tabel Formfield. Als het Formfield van het type dropdown, checkbox of radiobutton is moet er een lijst aan gekoppeld worden.

Een lijst is een soort container met één of meerdere items erin. Denk bijvoorbeeld aan een lijst genaamd maanden, met items januari, februari enzovoort. De lijst wordt opgeslagen in de tabel Fieldlist en de items in Listitem.

Een tekstwaarde van een gebruiker wordt opgeslagen in Formfield_User. Deze tabel verwijst naar het uid van tabel User en het formfielduid van tabel Formfield. Waardes van formvelden van het type radio, dropdown en checkbox worden opgeslagen in Listitem_User. Deze tabel verwijst naar het useruid van tabel User en de fieldlistuid van tabel Fieldlist. De echte waarde wordt verkregen door de verwijzing naar de Listitemuid van tabel Listitem.



Afbeelding 7 – Opslag formvelden

6 Constructiefase

Hoofdstuk 6, "Constructiefase" beschrijft het implementatiegedeelte van de afstudeeropdracht en de iteraties die geïmplementeerd zijn. De disciplines van RUP die in deze fase aan bod zijn gekomen zijn implementatie en testen.

6.1 Configuratie Visual Studio

De applicatie wordt gerealiseerd met verschillende projecten.
In de applicatie zijn de projecten MultilogueLib, AdminLib, Multilogue front-end, back-end en Admin aangemaakt.

In de LIBs zijn aanwezig:

- Helpers (voornamelijk voor het afhandelen van niet database gerelateerde functies)
- Models (voor het opvragen database gegevens)

In de Front-end, back-end en admin zijn aanwezig:

- Controllers
- Views
- Settings

Het voordeel hiervan is dat de libs eenvoudig gecompileerd kunnen worden naar een .dll en deze meegenomen kunnen worden in een project. Ook het werken aan verschillende projecten is eenvoudiger nu het verdeeld is in projecten.

6.2 Iteraties vaststellen

De iteraties zijn vastgesteld nadat alle eisen bekend waren en het ontwerp af was. Hierdoor had de afstudeerder een beter zicht op de te realiseren onderdelen en was er een betere inschatting qua tijd mogelijk. De iteraties zijn vastgesteld aan de hand van prioriteiten. De prioriteiten zijn door middel van het toepassen van de MoSCoW methode bepaald.
De prioriteiten zijn samen vastgesteld met Boudewijn Somer en Nils Corver.
Zie bijlage 23 voor een overzicht.
Er is overeengekomen om enkel de Must haves en de Should haves te implementeren.
Omdat elke iteratie circa drie weken duurt is er een iteratieplan opgesteld per iteratie. Deze is als externe bijlage opgenomen.

In het kort bestaat iteratie 1(iteratie must haves) uit het implementeren van de opgestelde Must haves. Deze iteratie is beschreven in hoofdstuk 7.3. Iteratie 2 (iteratie should haves) realiseert de should haves en staat beschreven in hoofdstuk 7.4.

6.3 Iteratie must haves

Hoofdstuk 6,3, “Iteratie must haves” beschrijft keuzes die gemaakt zijn voor het implementeren van de eerste iteratie. De Must haves zijn geïmplementeerd in deze iteratie (bijlage 23). Qua programmeerwerk worden een aantal zaken toegelicht wat de afstudeerder interessant vond om te doen.

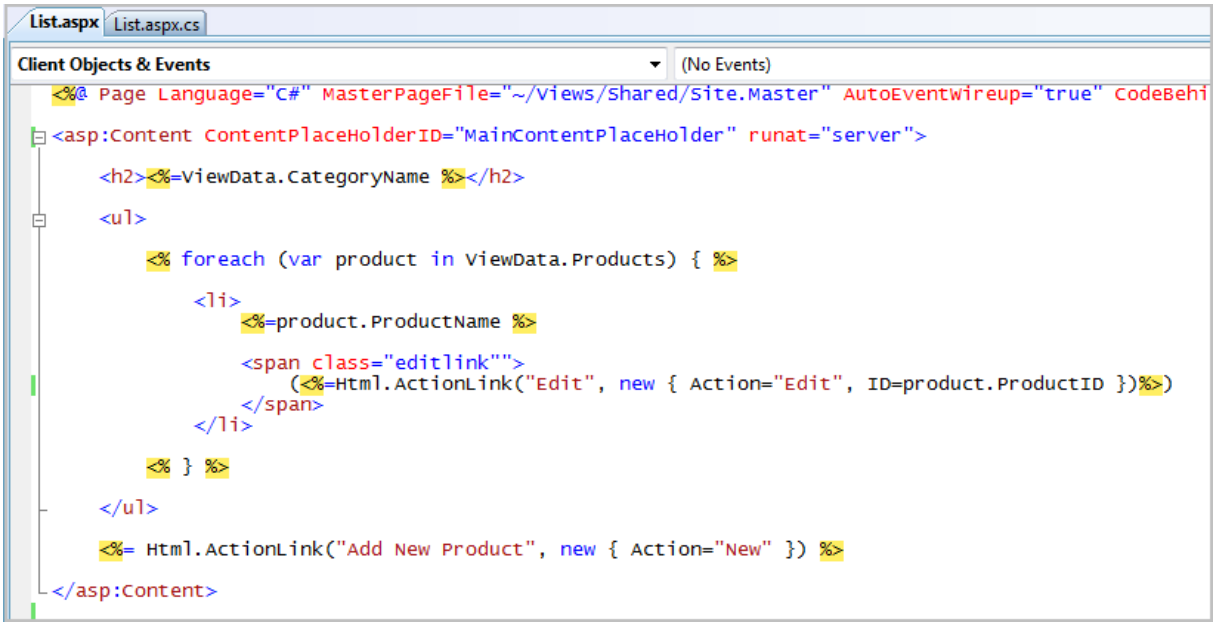
6.3.1 Analyse voor iteratie must haves

Analyse voor de eerste iteratie bestond uit het onderzoeken van:

1) Template engine
2) Prioriteiten

ad1 Template engine

Zonder template engine wordt er in ASP.NET code gebruikt tussen ASP blokken (<% … %> om bijvoorbeeld een “foreach” lus uit te voeren die data ophaalt. Zie afbeelding 8.



Afbeelding 8 – voorbeeld foreach lus

Het idee van het MVC model is juist het scheiden van code en lay-out. Dat is zonder een template of view engine zeker niet het geval. Het scheiden van code en lay-out geeft naast het eenvoudig testen ook het voordeel dat een designer de HTML kan opmaken. Over het algemeen zijn designers gespecialiseerd in het opmaken van HTML in combinatie met CSS. Zij hebben geen ervaring met code. De template engine is in overleg gekozen met behulp van een designer.

Dat betekent dat als er code opgenomen is in de HTML de moeilijkheidsgraad toe neemt voor een designer om HTML op te maken. Ook is er de mogelijkheid dat code beschadigd wordt die zij deze zelf niet kunnen repareren. Vanuit dit standpunt is een template engine benodigd.

Er zijn drie template engines onderzocht. Dit zijn StringTemplate, Spark en de template engine die binnen E-mark is ontwikkeld. Er is met name gekeken wat het eenvoudigst is om te begrijpen voor een HTML designer.

Spark gebruikt een XML manier om if statements en loops te realiseren (afbeelding 9). Naarmate de template ingewikkelder wordt, wordt de template ook lastiger te begrijpen.

```
<viewdata products="IEnumerable[[Product]]"/>
<ul if="products.Any()">
  <li each="var p in products">${p.Name}</li>
</ul>
<else>
  <p>No products available</p>
</else>
```

Afbeelding 9 – spark

Ook StringTemplate is niet geheel eenvoudig te lezen (Afbeelding 10).

```
<div>

        Employee Detail:
        $if(IsSuperUser)$
                <a href="/allowEdit">Edit</a>
        $else
                Editing not allowed
        $endif$
</div>
```

Afbeelding 10 – StringTemplate

Zowel Spark en StringTemplate zijn voor een designer lastig aan te passen omdat deze templates niet 100% HTML zijn. Er zit nog steeds programmeerlogica in die de designer niet snapt. Een bijkomend nadeel is dat deze templates niet gevalideerd kunnen worden door een HTML validator.
Deze argumenten zijn doorslaggevend geweest in het kiezen van de template engine van E-mark. Voordelen van deze template zijn:

- Leesgemak
- E-mark medewerkers zijn bekend met deze template engine
- Geschikt voor HTML validator
- Code van template engine is aan te passen

In afbeelding 11 wordt een voorbeeld gegeven van de template die binnen E-mark gemaakt is.

```
<!-- IMPORT Shared/mijnHeader -->

<div>
<!-- LIST -->
 <h1>Header</h1>
   <!-- ITEM -->
       <p>value is: {variable}</p>
   <!-- /ITEM -->
<!-- /LIST -->

</div>

<!-- IMPORT Shared/mijnFooter -->
```
Afbeelding 11 – E-mark template engine

Hierbij betekent IMPORT het ophalen van een externe HTML bestand. Waardes tussen "{}" zoals {variabele} worden gevuld vanuit de code. Hierbij is content dat tussen Item staat herhaaldelijk te gebruiken. De List is de container van één of meerdere item blokken.

ad2 Prioriteiten

Aangezien de afstudeerder technische informatica als achtergrond heeft is het maken van de grafische schil minder interessant en relevant. Daardoor wordt enkel de minimale HTML gemaakt en geen aandacht besteed aan CSS. Dit neemt E-mark later voor zijn rekening. De afstudeerder focust zich op de code.

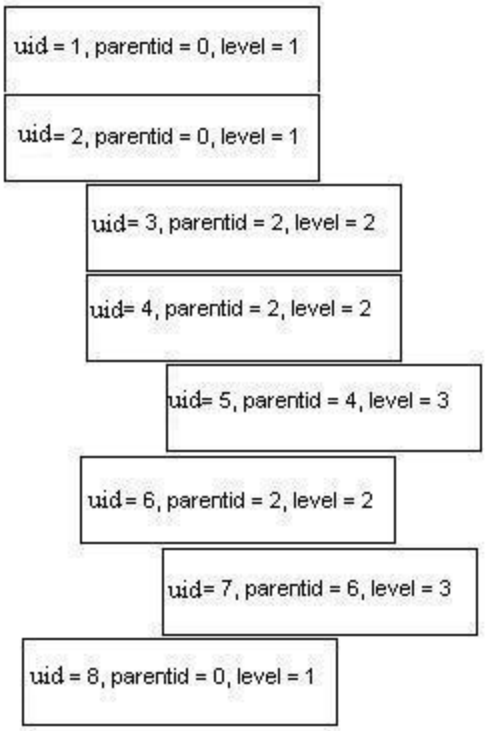
6.3.2 Tonen van geplaatste reacties

Een belangrijk onderdeel van de Multilogue is het plaatsen en lezen van reacties. Dit is het hart van de Multilogue. De PHP variant van de Multilogue haalde niet meer de snelheid die gewenst was. De snelheid van het laden van de pagina met 30 reacties bedroeg 800 milliseconde. De hiervoor genoemde tijd betreft enkel het uitvoeren van de code.

Belangrijk bij de nieuwe Multilogue is dat deze aantoont dat de reacties onder deze 800 milliseconde zit. Anders is er geen performancewinst geboekt.

Voor het totaalbeeld wordt er aangegeven waar reacties zich bevinden. Een editor kan vanuit de back-end een kamer aanmaken. Een kamer wordt aangemaakt om daarin vragen te stellen. Gebruikers kunnen reacties plaatsen op vragen en op reacties van gebruikers.

Dat betekent dus dat er een soort parent-child structuur is als er een reactie geplaatst is op een andere reactie.
Zie afbeelding 12 voor een voorbeeld van een aantal reacties die geplaatst zijn.



Afbeelding 12 – Mogelijke structuur reacties

Hierbij is de structuur te zien hoe reacties getoond moeten worden. De reacties worden in de tabel Reaction opgeslagen. Hierbij is de uid de primary key. De uid wordt opgehoogd bij elke nieuwe reactie die toe wordt gevoegd aan de tabel. De parentid verwijst naar de uid van een reactie die parent is. Het level geeft aan op welk level een reactie zich bevindt. Level 1 is het hoogste level. Als er een reactie geplaatst wordt op een level 1 reactie is het level van de nieuwe reactie 2. Als er een reactie geplaatst wordt op een level 2 wordt de nieuwe reactie level 3 en enzovoort. Er zit geen limiet op het aantal levels en het aantal reacties dat op een andere reactie geplaatst kunnen worden.

Bijlage 24 toont het resultaat van een aantal reacties. Het juist tonen van de reacties was moeilijker dan aanvankelijk gedacht werd. Daarom is het tonen van geplaatste reacties opgenomen in hoofdstuk 8 “Ondervonden problemen”. Daarin staat de verdere uitwerking.

6.3.3 Communicatie administrator/Multilogue

Het administratorgedeelte dient uiteindelijk voor het opzetten van een Multilogue. Functies zoals het toevoegen van talen en gebruikers (type editor) aan een Multilogue instantie zijn opgenomen in het Administrator gedeelte. Hierdoor functioneert de back-end van de administrator als een managementsysteem die de verscheidene Multilogues voor verschillende klanten beheert.

Omdat een Multilogue instantie en het administratorgedeelte aparte systemen zijn en op andere locaties gehost kan worden moest er een manier gevonden worden om deze systemen met elkaar te laten communiceren.

Hier wordt SOAP voor gebruikt zoals in hoofdstuk 6 beschreven is.
Om dit te realiseren is er een Webservice project aangemaakt. Deze kan als reference opgenomen worden in een ander project en vervolgens als een functie aangeroepen worden.

```
Multilogueservice.service soapConnection = new
Admin.Multilogueservice.service();

bool succesSoap = soapConnection.AddUser(soapObjectUser);
```

Hierbij is soapObjectUser een object van een klasse die gebruikt wordt de communicatie. Zo zijn er niet meer parameters benodigd als er meer informatie nodig is. De SOAP encoder zorgt ervoor dat de XML goed geplaatst wordt. Vanuit de webservice wordt de gebruiker toegevoegd in de Multilogue mits er geen velden leeg gelaten zijn en de gebruiker nog niet bestaat. De SOAP XML staat in bijlage 25.

6.3.4 Evaluatie iteratie must haves

Aan het begin van de eerste iteratie was het voornamelijk uitzoeken hoe precies de controllers, views, models en helpers gebruikt gingen worden. Uiteindelijk is er voor gekozen om models te gebruiken indien er database transacties benodigd zijn, helpers te gebruiken voor alle niet database functies (denk bijvoorbeeld aan het opbouwen van de template). Dit is zo gekozen door de afstudeerder. Omdat models in dezelfde namespace zitten als de datacontext van LINQ is dit een logsiche keus. E-mark heeft deze methode goed gekeurd.

De controllers functioneren voor het afhangen van foutieve data en het controleren van de workflow.
Daarnaast is het programmeren op een zo efficiënt mogelijke manier gedaan zonder de code onoverzichtelijk te maken.
Alle Must haves zijn geïmplementeerd en werken zoals het hoort te werken kijkend naar de oude Multilogue. Interessant is de snelheid van het laden van de pagina's.
Vergeleken met de PHP variant laden pagina's sneller. Echte cijfers worden gegeven met het tonen van reacties. De cijfers zijn bepaald aan de hand van een profiler.

Er is vergeleken met een vraag waarin 100 reacties staan. De oude Multilogue had hier ongeveer 800 ms voor nodig. De nieuwe Multilogue doet hier 240 milliseconde over.

6.4 Iteratie should haves

Hoofdstuk 6.4, “Iteratie should haves” beschrijft de keuzes die gemaakt zijn voor het implementeren van de tweede iteratie. De Should haves zijn geïmplementeerd in deze iteratie (bijlage 23).

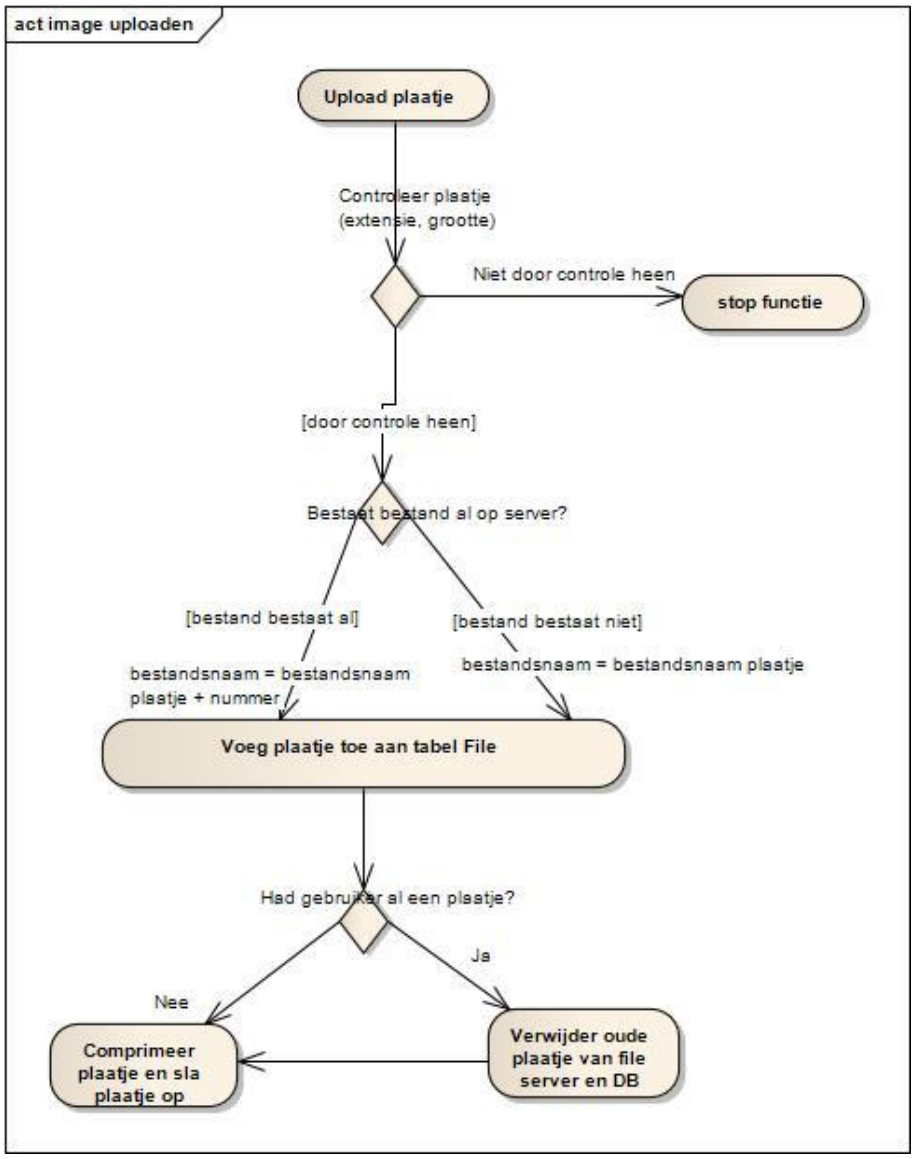
6.4.1 Analyse voor iteratie should haves

Het voornaamste struikelblok voor iteratie 2 was hoe het opslaan van bestanden afgehandeld moest worden. Bestanden opslaan in de database was geen optie aangezien dit de performance negatief kon beïnvloeden. Er is gekeken naar het datatype FILESTREAM dat geïntroduceerd werd met SQL Server 2008. FILESTREAM gebruikt een varbinary kolom. Het voordeel hiervan is dat het bestand extern opgeslagen wordt maar de integriteit behouden wordt. Dat betekent dat als het bestand verwijderd wordt de database ook hiervan op de hoogte is. De FILESTREAM is efficiënt als het om het streamen van bestanden gaat. Helaas is de ondersteuning vanuit .NET nihil. Dit komt doordat FILESTREAM een nieuw datatype is. Wel is de code gegeven om de FILESTREAM werkend te krijgen (bijlage 26).

Omdat het datatype FILESTREAM niet ondersteund wordt in .NET is er gekozen om deze methode nog niet toe te passen. In plaats hiervan is er gekozen om in de tabel File (bijlage 7) de bestandsnaam en url op te geven. Deze url is configureerbaar gemaakt in het web.config bestand. Mocht er ondersteuning komen voor het FILESTREAM datatype (wat komt) kan er gekozen worden om de FILESTREAM te gebruiken.

6.4.2 Uploaden image

Het uploaden van een image is mogelijk voor de bezoeker bij het wijzigen van het profiel. Zie afbeelding 13 voor de afhandeling van het uploaden van een plaatje.



Afbeelding 13- Uploaden van een plaatje

Er is gekozen om plaatjes te verwijderen als een gebruiker een ander plaatje kiest zodat de ruimte die de bestanden innemen beperkt blijft. Tevens worden de plaatjes verkleint tot 90 bij 90 pixels waardoor de ruimte die in beslag genomen wordt door bestanden hier eveneens beperkt blijft. De werking is beschreven in de afbeelding.

6.4.3 Toon “wie is online”

Een belangrijk onderdeel van de Multilogue is het tonen van mensen die online zijn (bijlage 29). Deze functie werd veel gebruikt in de PHP variant en bleek een zware pagina te zijn met veel gebruikers online. Zoals in bijlage 8 is te zien wordt er in tabel Session bijgehouden welke gebruiker online is en welke offline zijn. Ook het tijdstip dat ze voor het laatst online waren wordt bijgehouden (kolom updated). Het tonen van de online gebruikers beperkt zich tot 20 gebruikers per pagina. Het juist implementeren van de “Wie is online” pagina wordt uitgebreider beschreven in hoofdstuk 8.

Als er op de gebruikersnaam wordt geklikt worden de gegevens van de gebruiker getoond. Ook formvelden zijn hierin meegenomen.
De formvelden zijn ook te zien in het wijzigen van de pagina en tonen van het profiel. Zie bijlage 27. Hierbij zijn “bijnaam”, “favoriete dag”, “ben je vaak boos” en “hoe zou je je willen voelen” formvelden die toegevoegd zijn door de editor. De code voor het koppelen van waardes van formvelden aan een gebruiker is gegeven in bijlage 28.

6.4.4 Evaluatie Iteratie should haves

De should haves zijn geprogrammeerd op het instellen van een infobericht en het toevoegen van een Multiloguebericht na. Alles werkt naar behoren. De reden dat deze twee functies eruit zijn gelaten is omdat de formvelden voorrang hadden bij de implementatie. Daarna is er gekozen om meer tijd aan de scriptie te besteden en te testen van de applicatie.

De tijd voor het tonen van de mensen online middels de oude Multilogue bedroeg 800 milliseconden. De ASP.NET applicatie met hetzelfde aantal mensen online (47) heeft als resultaat 270 milliseconden.

7 Transitie

Hoofdstuk 7, “Transitie” beschrijft hoe het product overgedragen wordt en hoe er getest is. Ook wordt de evaluatie hier behandeld. Hierbij komen de disciplines testen en deployment aan bod.

7.1 Overdracht

Met Boudewijn Somer is afgesproken om het geheel als een zip op te leveren.
Het volgende wordt opgeleverd:

- Plan van aanpak
- Analysedocument
- Functioneel ontwerp
- Technisch ontwerp
- Iteratieplan
- Code (product)

7.2 Testen

Hoofdstuk 7.2, “Testen” beschrijft de tests die gepland stonden voor de ASP.NET applicatie.

Deze tests waren:

1. Performance test vergeleken met PHP variant van Multilogue
2. Invalide get parameters
3. Unit testing van model functies
4. HTML validatie
5. Eenheids testen
6. Blackbox testen

Er moet gezegd worden dat het testen niet zo uitvoerig gedaan is als van te voren gepland is. Dit kwam doordat na het tussentijds assessment bleek dat er nog veel feedback toegepast moest worden aan het afstudeerverslag. Hierdoor is het unit testen achterwege gelaten, deze zijn goed opgevangen door het Blackbox testen. De andere tests zijn wel uitgevoerd maar verder niet uitgebreid gedocumenteerd vanwege tijdgebrek.

De performance tests zijn gedaan met behulp van een profiler. Dit was ANTS.
Het testen van invalide get parameters beslaat het testen van urls met daarin waardes van variabelen. Zoals www.url.com/products/add/4 . De 4 is hier een get variabele. De applicatie mag niet crashen als een gebruiker hier een niet geldige waarde invult. Denk hierbij aan een string of de waarde leeglaten.

HTML validatie is gedaan door een standaard HTML validator op het internet (validator.w3c.org).
De eenheidstests (deeltjes van functionaliteit) zijn getest tijdens het implementeren.

7.3 Competenties

Hoofdstuk 7.3, "Competenties" beschrijft de competenties die beschreven zijn in de afstudeeropdracht. Dit hoofdstuk beschrijft waarom de competenties aangetoond zijn.
De competenties die gekozen zijn door de afstudeerder en relevant zijn aan de opdracht zijn:

1 Kiezen van een ontwikkelstrategie en ontwikkelmethodiek (A4)
2 Achterhalen van behoeften van belanghebbenden (A3)/Opstellen van systeemeisen (A5)
3 Ontwerpen van een technisch informatie systeem (C8)
4 Realiseren van software (D16)
5 Samenwerken in een project-team (G2)

Aangetoonde competenties:

ad1 Kiezen van een ontwikkelstrategie en ontwikkelmethodiek

Het kiezen van de ontwikkelstrategie en ontwikkelmethodiek is beschreven in het plan van aanpak, hoofdstuk 3 met bijbehorende motivatie. De gekozen ontwikkelmethodiek is RUP.

Ad2 Achterhalen van behoeften van belanghebbenden / Opstellen van systeemeisen

Systeemeisen zijn beschreven in het functioneel ontwerp. Behoeften van belanghebbenden zijn achterhaald door de oude Multilogue te analyseren. Hierbij is het analysedocument opgesteld. Eisen zijn verdeeld in functionele en niet functionele eisen. Overige behoeften die niet duidelijk naar voren kwamen bij het analyseren van de applicatie zijn verkregen via programmeurs die de oude Multilogue gemaakt hebben.

Ad3 Ontwerpen van een technisch informatie systeem

Het ontwerp van het informatiesysteem is gegeven en verantwoord in het technisch ontwerp met behulp van de modelleringstaal UML.

Ad4 Realiseren van software

De code is meegeleverd als externe bijlage. Er is gebruik gemaakt van de rijkheid van de taal, syntactisch juist geprogrammeerd en broncode is netjes georganiseerd.

Ad5 Samenwerken in een project-team

Het merendeel van het project heeft de afstudeerder alleen aan het project gewerkt. Er is samengewerkt in de trend van het afnemen van interviews en het oplossen van vraagstukken.

7.4 Evaluatie

Hoofdstuk 7.4, “Evaluatie” beschrijft de punten die de afstudeerder goed vond gaan en de dingen die tegen vielen.

Wat ging goed:

- Het analyseren van de eisen verliep soepel, mede door juiste vragen in interviews en het goed analyseren van de oude Multilogue
- Het technisch ontwerp. Doordat er eerst verdiept is in het ASP.NET MVC framework
- Indeling van libs en front-end, editor en administrator. Dit bleek heel erg praktisch.
- Performance van ASP.NET applicatie is beter dan oude applicatie. Code werkt
- Het managen en plannen van het project. Enkel uitgelopen op het maken van formvelden.
- Documentatie is van goed niveau.
- Communicatie met medewerkers van E-mark.

Wat viel tegen:

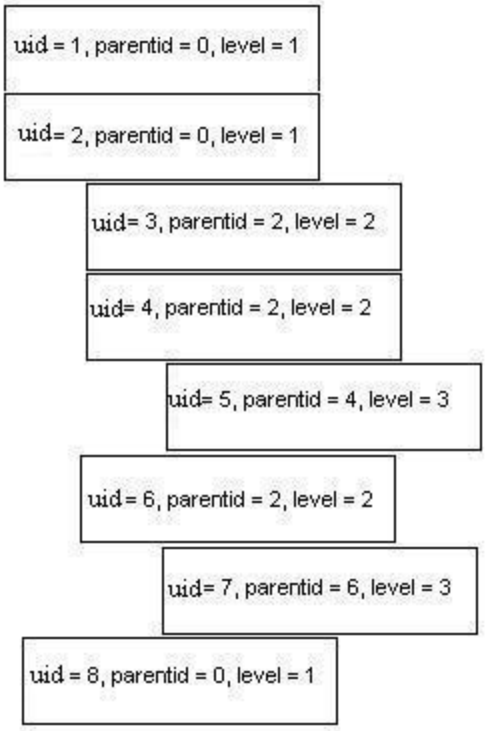
- Het zoeken van een goede profiler van ASP.NET pagina’s zonder .aspx pagina’s.
- ASP.NET MVC framework leren, deze bleek complexer te zijn dan aanvankelijk gedacht werd.
- Applicatie Multilogue was groter dan aanvankelijk gedacht werd

8 Ondervonden problemen

Hoofdstuk 8, "Ondervonden problemen" beschrijft de problemen die onverwachts optraden tijdens de afstudeerstage. Met problemen wordt bedoeld een bepaalde implementatie die langer dan gepland duurde.

8.1 Reacties ordenen

Probleem:
Gegeven de volgende situatie (afbeelding 14):



Afbeelding 14 – Reacties

Hierbij is de uid de primary key. De uid wordt opgehoogd bij elke nieuwe reactie die toe wordt gevoegd aan de tabel. De parentid verwijst naar de uid van een reactie die parent is. Het level geeft aan op welk level een reactie zich bevindt. Level 1 is het hoogste level. Als er een reactie geplaatst wordt op een level 1 reactie is het level van de nieuwe reactie 2. Als er een reactie geplaatst wordt op een level 2 wordt de nieuwe reactie level 3 en enzovoort. Er zit geen limiet op het aantal levels en het aantal reacties dat op een andere reactie geplaatst kunnen worden.

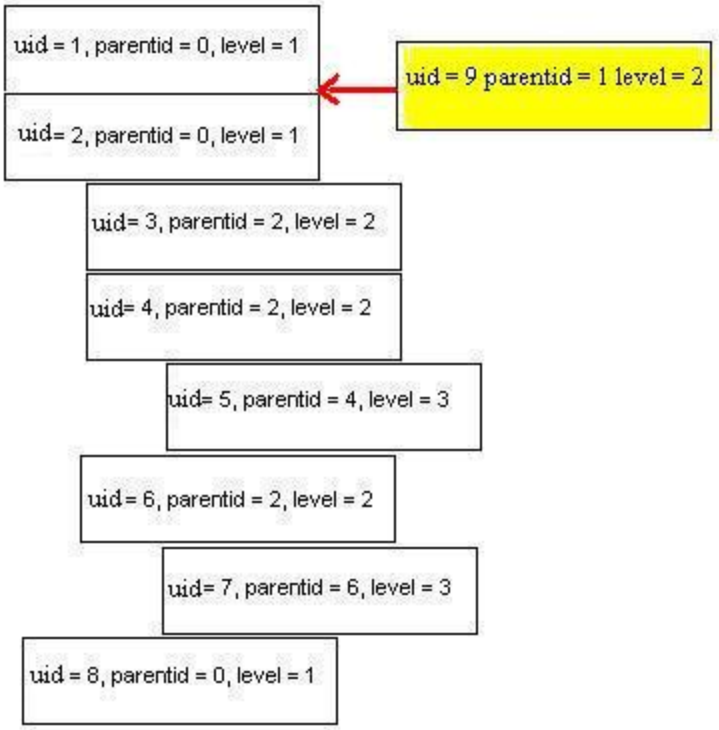
De ideale volgorde van reacties is eerste de nieuwste of oudste level 1 reactie met alle kinderen van die reactie erna. Vervolgens komt de tweede level 1 reactie met bijbehorende kinderen.

Dit geeft de volgende ideale structuur :

Level 1 reactie geplaatst op 2 januari, uid = 1, parentid = 0, level = 1
Level2 reactie geplaatst op 4 januari, uid = 2, parentid = 1, level = 2
Level2 reactie geplaatst op 5 januari, uid = 3, parentid = 2, level = 2
Level 3 reactie geplaatst op 7 januari, uid = 4, parentid = 3, level = 3

Level 1 reactie geplaatst op 3 januari, uid = 5, parentid = 0, level = 1
Level2 reactie geplaatst op 10 januari, uid = 6, parentid = 5, level = 2
Level2 reactie geplaatst op 12 januari, uid = 7, parentid = 5, level = 2

Terugkijkend naar afbeelding 14. Daar is de ideale situatie geschetst. Stel iemand reageert op een reactie. De volgende situatie doet zich voor (afbeelding 15):



Afbeelding 15- reactie wordt toegevoegd.

Nu volgt de volgende vraag. Hoe moet er geordend worden. Dit kan niet op het veld uid aangezien op elk willekeurige plek een nieuwe reactie geplaatst kan worden. De uid is een auto incrementele kolom dus dat werkt niet. Ook met SQL queries is dit niet mogelijk. Een ORDER BY date (afgezien van descending/ascending) toont niet de gewenste structuur namelijk:

Level 1 reactie geplaatst op 2 januari, uid = 1, parentid = 0, level = 1
Level 1 reactie geplaatst op 3 januari, uid = 5, parentid = 0, level = 1
Level2 reactie geplaatst op 4 januari, uid = 2, parentid = 1, level = 2
Level2 reactie geplaatst op 5 januari, uid = 3, parentid = 2, level = 2
Level 3 reactie geplaatst op 7 januari, uid = 4, parentid = 3, level = 3
Level2 reactie geplaatst op 10 januari, uid = 6, parentid = 5, level = 2
Level2 reactie geplaatst op 12 januari, uid = 7, parentid = 5, level = 2

Een ORDER BY level geeft het volgende resultaat

Level 1 reactie geplaatst op 2 januari, uid = 1, parentid = 0, level = 1
Level 1 reactie geplaatst op 3 januari, uid = 5, parentid = 0, level = 1
Level2 reactie geplaatst op 4 januari, uid = 2, parentid = 1, level = 2
Level2 reactie geplaatst op 5 januari, uid = 3, parentid = 2, level = 2
Level2 reactie geplaatst op 10 januari, uid = 6, parentid = 5, level = 2
Level2 reactie geplaatst op 12 januari, uid = 7, parentid = 5, level = 2
Level 3 reactie geplaatst op 7 januari, uid = 4, parentid = 3, level = 3

Een combinatie van ORDER By date en level werkt derhalve niet. Dit omdat de tweede orderby clause (level) pas toegepast wordt als er niet gesorteerd kan worden op date (betekent dat waardes gelijk zijn).

Analyse:
Eerst is er gekeken hoe dit functioneel opgelost kon worden. Dat betekent dat er niet gekeken is naar performance (snelheid van functie). Na onderzoek gedaan te hebben via internet bleek dat geordende lijsten goed te doen zijn met recursieve functies.

Keuzes:
Toepassen van een recursieve functie.

Uitwerking:

Zie afbeelding 16 en 17 voor de code van de recursieve functie.

```
/// <summary>
/// generate a list for reactions
/// </summary>
/// <param name="parentuid"></param>
/// <param name="level"></param>
/// <param name="partialList"></param>
public static void GenerateList(int questionuid, List<Reaction> ordenedList)
{
    List<Reaction> level1reactions = new List<Reaction>();
    using (MultilogueDataContext multilogueContext = new MultilogueDataContext())
    {
        level1reactions = (from reaction in multilogueContext.Reactions
                           where reaction.level == 1 && reaction.question_uid == questionuid
                           orderby reaction.created descending
                           select reaction).ToList();
    }
    foreach (Reaction level1Reaction in level1reactions)
    {
        level1Reaction.Sort(level1Reaction.uid, level1Reaction.level, ordenedList);
    }
}
```

Afbeelding 16 – Functie GenerateList

```
/// <summary>
/// Sort child reactions
/// </summary>
/// <param name="rootuid">id of a level 1 reaction</param>
/// <param name="level">level of a reaction</param>
/// <param name="partialList">list where children of a level 1 reaction are added</param>
public void Sort(int rootuid, int level, List<Reaction> partialList)
{
    using (MultilogueDataContext multilogueContext = new MultilogueDataContext())
    {
        if (level == 1)
        {
            partialList.Add(this);
        }
        //retrieve reactions with parentuid
        var result = (from reaction in multilogueContext.Reactions
                      where reaction.parent_uid == rootuid
                      select reaction);

        if (result.Count() != 0)
        {
            foreach (var reac in result)
            {
                if (!partialList.Contains((Reaction)reac))
                {
                    partialList.Add((Reaction)reac);
                    Sort(reac.uid, level + 1, partialList);
                }
            }
        }
    }
}
```

Afbeelding 17 – Sort

De functie GenerateList krijgt een lege ordenedList mee. In deze lijst komen uiteindelijk alle reacties te staan op de goede volgorde. GenerateList vergaart allereerst alle level 1 reacties. Voor elke level 1 reactie wordt de functie Sort aangeroepen. Deze functie is recursief, de functie verzamelt alle child reacties en voegt deze toe aan de partialList. Dit wordt gedaan totdat er geen child reacties meer over zijn. De List die gebruikt wordt als parameter is By Reference.
Dat betekent dat de lijst die gevuld wordt met gesorteerde reacties beschikbaar is in de controller.

Evaluatie:
Deze methode werkt functioneel, maar is niet de meest optimale. Nadat er 30 reacties geplaatst zijn, is er 30 ms nodig om de reacties in de goede volgorde te tonen (Gemeten met de Stopwatch class).

Probleem:
De recursieve functie werkt functioneel, het is echter algemeen bekend dat recursieve niet de meest efficiënte manier (qua snelheid van uitvoeren) is.

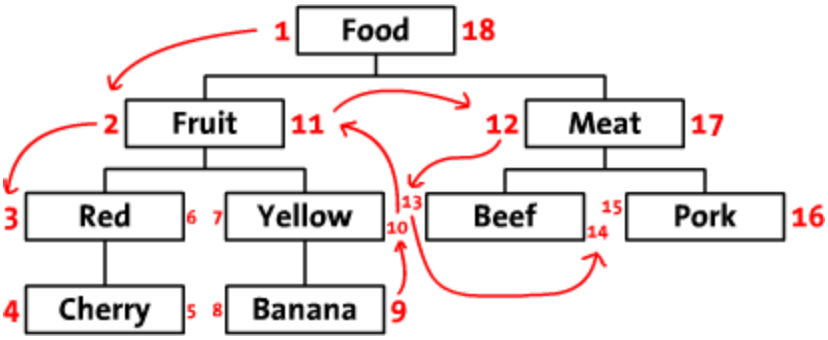
Analyse:
Research gedaan (met name internet) hoe de reacties juist geordend kunnen worden zonder gebruik te maken van recursie.

Oplossingen:

1) Datatype hierarchyid
2) Gebruiken van een balanced tree
3) Toekennen van indexes

Keuze:
Het datatype hierarchyid zit in SQL Server 2008. Deze keus bleek snel niet haalbaar. Dit omdat de hierarchyid datatype nog niet goed samenwerkt met LINQ.

Een andere mogelijkheid waaraan gedacht is, is het gebruiken van een balanced tree.
(Afbeelding 18)



Afbeelding 18 – balanced tree

Het voordeel van een balanced tree is dat het sneller elementen kan opzoeken (Big O van Log n) wat betekent dat de snelheid van het laden van reacties toeneemt. Aan de hand van de linker- en rechternummers kan er bepaald worden of een node een kind is van een parentnode. Het linker getal van de kind node is dan groter dan het linker getal van de parent node en het rechtergetal kleiner. Probleem is dat bij een insert al deze getallen (linker en rechterwaardes) geüpdatet moeten worden. Dit is duur qua performance. Een ander probleem is dat er meerdere hiërarchische bomen zijn in de database (voor iedere level 1 reactie per vraag). Dit geeft het probleem hoe de nummering van de linker en rechterwaardes toegepast moet worden. Want stel dat een level 1 reactie-boom een rechterwaarde heeft van 18. De volgende level 1 reactie-boom moet dan bij 19 beginnen. Dat betekent dat als een nieuwe reactie toegevoegd wordt aan de eerste level 1 – boom de rechterwaarde oploopt tot en met 19. Dat veroorzaakt problemen.

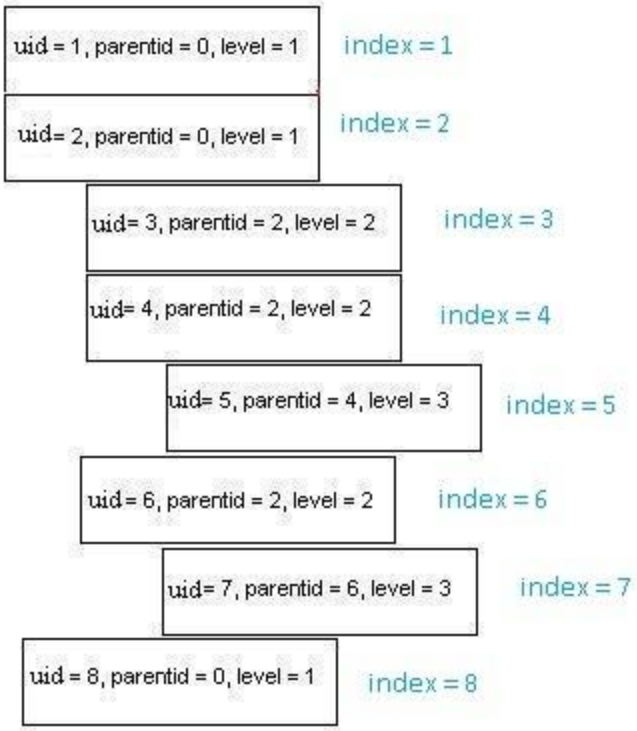
Een ander probleem is het onderscheiden van nieuwere reacties. Stel dat er gekozen wordt om uiteindelijk de nieuwere reacties eerst te laten zien en vervolgens de oude dan geeft de balance tree nog een extra probleem. Zie afbeelding 18. Als de situatie zich voordoet dat “Meat” later geplaatst is en deze voor “Fruit” geplaatst moet worden dan volstaat de volgende query niet:

```
SELECT * FROM tree WHERE lft BETWEEN 2 AND 181 ORDER BY lft ASC;
```

Hierbij is lft het linkernummer van een node.
Wat dan wenselijk is het sorteren op datum.
Maar twee ORDER BY werken niet in dit geval, omdat de tweede ORDER BY pas van toepassing als er niet gesorteerd kan worden op lft.
Om deze redenen is de balanced tree niet in gebruik genomen.

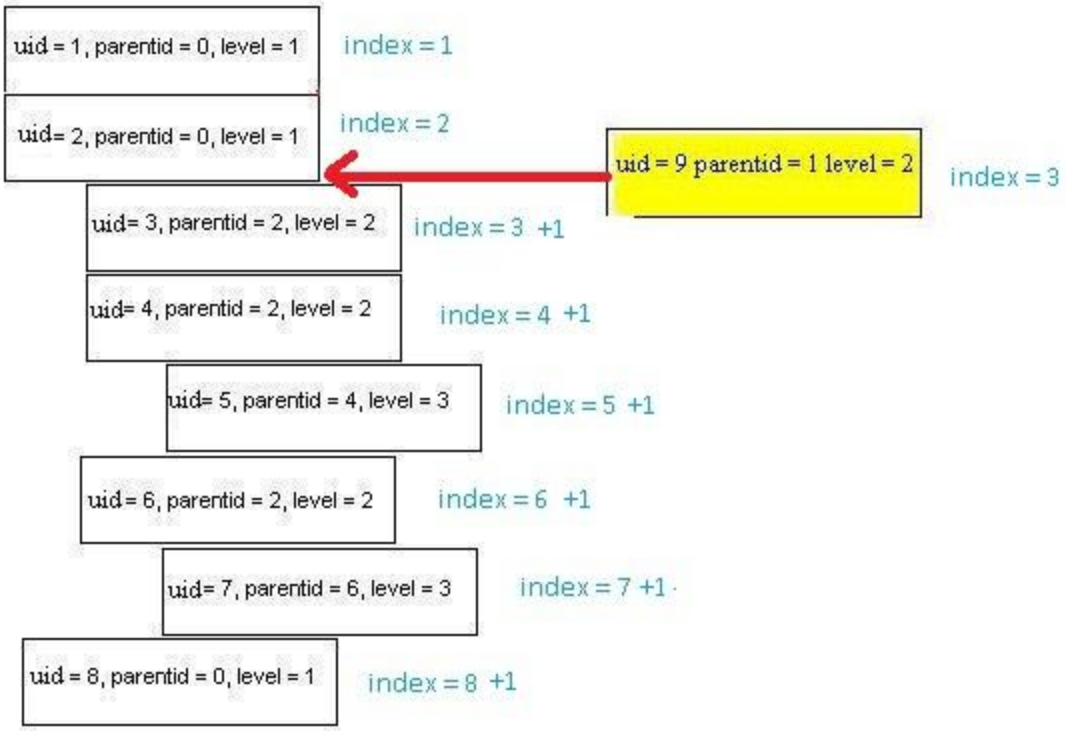
Een betere manier is het toekennen van indexes. Bij elke insert wordt er bekeken welke index er gebruikt moet worden. In het geval van een child reactie is de index van de child de parent index plus één. In het geval van een level 1 reactie is het de hoogste index plus één. Alle indexes van de reacties die in dezelfde vraag staan met een grotere index moeten met één verhoogd worden. Doordat de logica bij de insert zit hoeft er bij het ophalen van de reacties geen logica meer aan te pas te komen. Er wordt enkel gesorteerd op index. De index staat los van level, parentid of de uid.

Zie afbeelding 19 voor de standaard situatie maar nu met indexes.



Afbeelding 19 – Index reacties

Stel dat een gebruiker nu een reactie plaatst op de reactie met uid = 2. Gewenst effect is dat deze reactie boven de reactie schuift met uid = 3, omdat deze reactie nieuwer is. De hieronder opgenomen afbeelding laat zien wat het gevolg hier van is. (afbeelding 20).



Afbeelding 20 – Reactie wordt ingevoegd met nieuwe index

Hier zijn twee stored procedures voor gemaakt. Een voor het updaten van de indexes en een voor het toevoegen van de reactie in de tabel. LINQ maakt voor een stored procedure een functie aan die aangeroepen kan worden.
Doordat SQL Server de tabel lockt tijdens het inserten geeft dit geen problemen met concurrency.

Evaluatie:

Dezelfde test werd gedaan als met de recursie(het meten van de tijd voor het laden van 30 reacties). Dit duurde nu 10 ms. Dat betekent dat de manier die nu gerealiseerd is een betere performance heeft dan de recursieve uitwerking.

8.2 Updaten van offline en online bit

De multilogue heeft de mogelijkheid om te tonen welke gebruikers er online zijn. De gebruikers wordt online getoond als de gebruiker hiermee instemt op de profielpagina.

Probleem:
Het probleem in het tonen van online gebruikers zit hem in de recentheid van de lijst en ook de correctheid. Als een gebruiker via de normale manier uitlogt (dus uitloggen klikt) dan is er niets aan de hand. De sessie wordt vernietigd en de database is er van op de hoogte dat de gebruiker niet meer online is.
Stel dat de gebruiker niet uitlogt. Dat wil zeggen dat de pc in één keer uitgedaan wordt/uitvalt of dat de gebruiker de browser afsluit of het proces vernietigt van de browser.

Het probleem is dan dat de gebruiker die niet uitlogt ook niet meer uitgelogd wordt. Dat betekent dat deze altijd online staat op de "wie is online" pagina. Als veel gebruikers niet uitloggen biedt deze pagina foutieve informatie wat het nut van de pagina niet ten goede komt.

Analyse:
Probleem besproken met Boudewijn Somer en Nils Corver. Verder research gedaan op internet.

Oplossingen:

1) Javascript
2) Task Scheduler
3) Static timer

Keuze:

Een overwogen oplossing om dit op te lossen was met behulp van JavaScript. Het is mogelijk om met JavaScript een event af te hangen wanneer een gebruiker een pagina verlaat. Dit kan met de onUnload event handler. Dit lost slechts een deel van het probleem op. Namelijk alleen als de gebruiker de browser sluit zonder uit te loggen.
Indien de pc abrupt wordt afgesloten wordt de event voor onUnload niet uitgevoerd. Omdat JavaScript client-side is biedt het ook de mogelijkheid om JavaScript uit te zetten. De hiervoor genoemde oplossingen zijn derhalve niet geschikt om met JavaScript op te lossen.

Een betere manier om dit probleem op te lossen is gebruik te maken van de "SQL Server 2008 Task Scheduler". Omdat de Express versie gebruikt wordt is deze functionaliteit echter niet mogelijk. Mocht de Multilogue echt live gaan en als er gebruik gemaakt gaat worden van SQL Server 2008 dan kan deze oplossing gebruikt worden. De Task scheduler kan gezien worden als een soort cronjob van Linux.

Een andere werkende oplossing is een static timer opnemen in de Global.asax. Dit is een oplossing op applicatieniveau.

Uitwerking:
De timer wordt geset tijdens het starten van de applicatie (de client start niet de applicatie!). Hier wordt de Application_Start functie voor gebruikt. De variabele whenTaskLastRan is van het type DateTime.

```
protected void Application_Start()
        {
            whenTaskLastRan = DateTime.Now;
            AreaRegistration.RegisterAllAreas();
            RegisterRoutes(RouteTable.Routes);

        }
```

Zodra er een sessie gestart wordt (clientsessie) wordt er gekeken of er mensen online zijn waarbij het updated field ouder is dan een kwartier.

```
protected void Session_Start(object sender, EventArgs e)
        {
            M.Session.DiscoverTimeout(whenTaskLastRan);

        }

public static void DiscoverTimeout(DateTime whenTaskLastRan)
        {
            DateTime Timeout = DateTime.Now.AddMinutes(-15);
            if(whenTaskLastRan < Timeout)
            {
                using (MultilogueDataContext db = new
MultilogueDataContext())
                {
                    List<Session> usersessions =
CompiledQuery_UsersTimedOut(db).ToList();
                    if (usersessions.Count != 0)
                    {
                        foreach (Session session in usersessions)
                        {
                            if (Timeout > session.updated)
                            {
                                session.online = false;
                                db.SubmitChanges();
                            }
                        }
                    }
                }
                whenTaskLastRan = DateTime.Now;
            }

        }
```

Het updated field van tabel Session wordt geüpdatet met waarde DateTime.Now als een gebruiker van pagina switch of iets post.

Evaluatie:
De oplossing werkt. Een goed alternatief is het gebruik van de Task Scheduler in SQL Server 2008.

8.3 Testen met session, tempData

De applicatie maakt gebruik van Session voor het opslaan van de useruid en Tempdata voor foutafhandelingen en andere tijdelijke data. Het handige van het MVC model is juist de scheiding van de code en het grafisch raamwerk. Het probleem met Unit testen is dat de werkelijke applicatie niet draait. In technische termen is de pipeline van ASP.NET niet actief. Dat betekent dat Session en TempData objecten niet aangemaakt zijn en ook niet eenvoudig getest kunnen worden. Bij het testen van een Session object wordt er een null reference gegeven (dus object bestaat niet).
Dit is wel op te lossen met mocks of "fakesessions" en dergelijk maar dat kost meer tijd waardoor dit geen voordeel oplevert.
Na lang gezocht te hebben voor de beste manier van testen en in overleg met Boudewijn Somer is besloten om de modellen te unit testen. Controllers waar sessies en tempdata in verwerkt zitten worden getest door de applicatie daadwerkelijk uit te voeren met verschillende input. Dit omdat controllers niet echt logica verwerken maar voornamelijk de gebruiker loodsen naar de desbetreffende paginaa's.

8.4 Formvelden

Niet zozeer de moeilijkheidsgraad van formvelden heeft er toegeleid dat Formvelden voor een probleem zorgden. Meer omdat de tijd die gepland was voor de implementatie van de formvelden te kort was. Wat tegenviel was het weergeven van de ingevulde waardes door de gebruiker. Dit kwam doordat er meerdere typen gebruikt werden zoals checkbox en radio.

9 Performance verbeteringen/aanbevelingen

Hoofdstuk 9, Performance verbeteringen/aanbevelingen beschrijft de aanbevelingen voor het vervolg van het project. Ook worden performance verbeteringen toegelicht

De volgende aanbevelingen worden toegelicht:

1) Compiled queries/Stored procedures indien nodig
2) Output cache
3) Filestream
4) SQL Server indexes
5) Scheduled jobs
6) Webhooks
7) E-mark mail

9.1 Compiled queries/Stored procedures (performance verbetering)

Door het gebruik van LINQ is het mogelijk om naast “normale” queries deze ook voor gecompileerde queries te gebruiken. Door van een query een compiled query te maken die static is hoeft deze query maar eenmalig geëvalueerd te worden. Bij verdere aanroepen is er geen overhead meer.

Een voorbeeld van een gecompileerde LINQ query is te zien in afbeelding 14.

```
public static Func<MultilogueDataContext, IQueryable<Session>> CompiledQuery_UsersOnline =
    CompiledQuery.Compile<MultilogueDataContext, IQueryable<Session>>(
     (db)=>db.Sessions.Where(s => s.online == true).OrderBy(se => se.created));
```

Afbeelding 14 – gecompileerde query

Afbeelding 14 toont de query voor het verkrijgen van Session objecten die aangeven of een gebruiker online is. De IQueryable<Session> is het returntype. De MultilogueDataContext is een parameter die meegegeven wordt. De datacontext zorgt voor het genereren van de SQL queries op basis van de LINQ queries. Ook zet de datacontext rijen van data uit de database om in objecten(models).

In afbeelding 15 wordt weergegeven hoe een gecompileerde query aangeroepen kan worden.

```
/// <summary> ...
public static List<Session> UsersOnline()
{
    using (MultilogueDataContext db = new MultilogueDataContext())
    {
        return CompiledQuery_UsersOnline(db).ToList();
    }
}
```

Afbeelding 15 – Aanroep gecompileerde query

Gecompileerde queries geven betere performance dan niet gecompileerde queries met name als queries meerdere keren aangeroepen moeten worden.

Mocht er besloten worden dat er meer tijdwinst op het gebied van querytijd behaald moet worden kan er gebruik gemaakt van Stored Procedures. Momenteel is het Opslaan van reacties en het wijzigen van indexes gedaan met behulp van Stored Procedures. De reden hiervoor is dat deze handelingen zo snel mogelijk gedaan moeten worden.
De reden dat niet alle queries met behulp van Stored Procedures gedaan zijn is dat deze manier een extra onderhoudspunt geeft.

9.2 Caching (aanbeveling)

ASP.NET ondersteunt een drietal soorten van caching namelijk:

1) Output caching
2) Fragment caching
3) Data caching

ad1 Output caching

Output caching is de meest eenvoudige manier van cachen bij ASP.NET. Met output caching wordt de volledige pagina gecached.
De volgende regel code zorgt ervoor dat een pagina 30 seconden gecached blijft.

[OutputCache Duration="30" VaryByParam="none"]

Wat er gebeurd onder de motorkap is dat een dynamisch aangemaakte pagina bewaard wordt voordat het naar de client gezonden wordt. De HTML wordt hergebruikt zodat voor toekomstige aanvragen de code niet opnieuw uitgevoerd hoeft te worden.

In het geval van de Multilogue biedt output caching geen oplossing. Dit omdat op elke pagina de gebruiker bij naam aangesproken wordt. Dat betekent dat als een gebruiker zich aanmeld en een bepaalde pagina gecached wordt andere gebruikers deze naam dus ook zien.

ad2 Fragment caching

Fragment caching is een beetje een misleidende naam. Fragment caching is ook niet erg eenvoudig te realiseren omdat het gedeelte dat gecached dient te worden omgezet moet worden in een user-control. User controls vormen herbruikbare componenten voor pagina's.

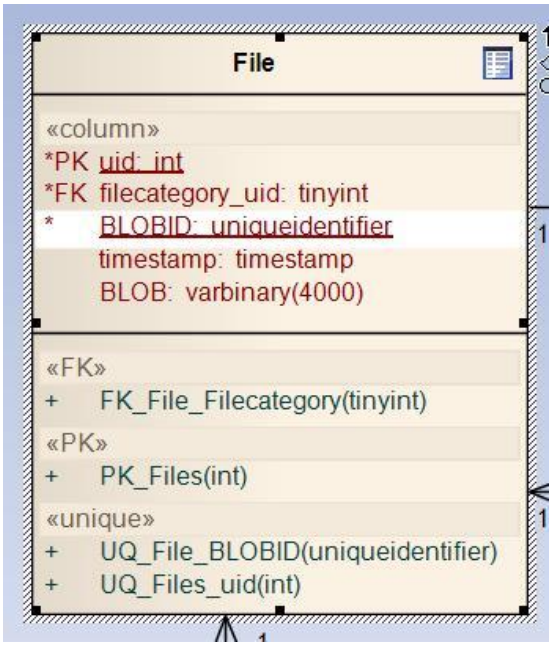
ad3 Data caching

Omdat fragment caching en output caching delen van een of de gehele pagina cached moet er nagedacht worden of dat in het geval van de Multilogue wel zo nuttig is. Omdat de applicatie een interactief geheel vormt is het niet handig om een pagina te cachen (bijvoorbeeld reacties/kamers/wie is online). Doordat de applicatie dan niet meer actueel is betekent dit dat deze vorm van caching niet geschikt is.

De data caching is de meest krachtige van de caching-opties in ASP.NET. Alles kan programmatisch gecached worden. De cache werkt als een dictionary (map in c++).
Dat betekent dat het een key, value pair is. In het geval van de Multilogue kan dit gebruikt worden om bijvoorbeeld images van reacties/wie is online gecached worden.

9.3 Filestream (aanbeveling)

Een alternatief voor het afhandelen van bestanden zoals video's en afbeeldingen is het gebruik van het datatype SQL Server FileStream.(hoofdstuk 9.1). Afbeelding 16 geeft de nieuwe tabel weer.



Afbeelding 16 – tabel File met FileStream

De BLOBID is van het type UNIQUEIDENTIFIER. Dit is benodigd als er gebruik gemaakt wordt van FileStream. Kolom BLOB slaat de verwijzing op naar het bestand als een varbinary(MAX). De 4000 staat daar omdat Enterprise architect geen MAX accepteert. Voordeel van de FileStream is dat de onderhoudbaarheid vergroot wordt. Frequente kleine updates duurt wel langer bij het gebruik van FileStream.

9.4 SQL Server indexes (performance verbetering)

Voor snellere performance op het gebied van het uitvoeren van queries is er gekeken of indexes een uitkomst bieden.

Verschillende soorten indexes zijn bekeken, namelijk:

1) Non clustered index
2) Clustered index
3) Filtered index

ad1 Non clustered index

Non clustered index wordt aangemaakt zodra er een index key gegenereerd word. De fysieke volgorde is niet hetzelfde als de index orde. Dat betekent dus dat er gebruik gemaakt wordt

van een pointer die verwijst naar de pagina en het rijnummer in de pagina. Non clustered index zijn geschikt waar rijen vaak veranderd kunnen worden.

ad2 Clustered index

Een clustered index is uniek voor een tabel. Een clustered index is een special type index dat ervoor zorgt dat de rijen van de tabel zo gesorteerd worden zoals ze ook fysiek zijn opgeslagen. Voordeel hiervan is dat het lezen sneller kan gaan dan bij een non clustered index. Toch is er niet gekozen om een clustered index toe te passen aangezien bij updates en inserts de gehele volgorde aangepast moet worden. Dit is een dure operatie. Denk bij het wijzigen van de indexen van reacties.

ad3 Filtered index

Filtered indexes kunnen gebruikt worden om kleinere delen van een tabel te gebruiken voor een query. Omdat de tabellen voornamelijk nummers gebruiken om te indexen is er niet veel snelheidswinst te behalen, aangezien het beter werkt voor strings.

9.5 Scheduled jobs (aanbeveling)

Bij het gebruik van SQL Server 2008 (niet de express versie) kan er gebruik gemaakt worden van Scheduled jobs. Dit is in hoofdstuk 12.2 beschreven.

9.6 Webhooks (aanbeveling)

Het concept van Webhooks is eenvoudig. Een webhook is een HTTP Callback, ofwel een HTTP POST dat zich voordoet wanneer een bepaalde gebeurtenis plaatsvindt. Voordeel van webhooks is dat er een event afgevuurd wordt als er een HTTP POST zich voordoet, dit is efficiënter dan pollen. Om webhooks werkend te krijgen moet de applicatie het toelaten dat gebruikers hun eigen URL registreren zodat deze aangeroepen wordt tijdens een bepaalde functie van de applicatie.

Er zijn drie algemene manieren om webhooks in een applicatie te gebruiken:

1) Push
2) Pipes
3) Plugins

ad1 Push

“Push” zorgt ervoor dat polling niet meer nodig is om te kijken of er nieuwe informatie is. Na het registreren van de webhook ontvangt de webhook de informatie zodra het er is.

ad2 Pipes

Er is sprake van een pipe als de webhook data ontvangt maar ook daadwerkelijk acties uitvoert die niet gerelateerd staan aan het event waardoor de webhook informatie verkrijgt. Denk hierbij aan een script dat een mail verstuurd of een Twitter bericht aanmaakt als een gebruiker een nieuwe foto upload.

ad3 Plugins

Plugins kunnen gebruikt worden om de applicatie uit te breiden waarbij gebruikers eigen stuk scripts koppelen aan URLs.

9.7 E-mark mail

Voor het verzenden van e-mail kan er gekozen worden om de mails (bijvoorbeeld voor het rapporteren van reacties) te versturen via E-mark mail. Dit geschiedt via een SOAP handeling.

Conclusie

De afstudeerder vindt dat het herbouwen van de Multilogue een uitdagende opdracht was. Er is tijdens het doorlopen van de opdracht veel kennis opgedaan op het gebied van ASP.NET en bijbehorende technieken (denk aan MVC model, LINQ, template engines…)

De afstudeerder is met de nieuwe applicatie (ontwikkeld tot het huidige niveau) tevreden. De applicatie is sneller dan de oude PHP Multilogue. Daarmee zijn de beoogde doelstellingen behaald.

De applicatie, ontwikkelt door de afstudeerder, heeft zondermeer de potentie om in de toekomst operationeel te worden en daarbij voor E-mark een toegevoegde waarde biedt.

Bijlage A - Bronnenlijst

De volgende websites en boeken zijn gebruikt voor het inwinnen van informatie tijdens de stage.

A.1 Internet

www.wikipedia.com
voor algemene begrippen.

http://www.asp.net/learn/mvc/#MVC_Overview
voor introductie van ASP.NET MVC.

http://weblogs.asp.net/scottgu/archive/2007/11/13/asp-net-mvc-framework-part-1.aspx
Voor het opzetten van een eenvoudige applicatie met behulp van het MVC model en LINQ. Tot en met part 6 doorgenomen.

http://weblogs.asp.net/scottgu/archive/2007/05/19/using-linq-to-sql-part-1.aspx
http://www.vikramlakhotia.com/Using_Take_and_Skip_method_in_LINQ_queries.aspx
http://msdn.microsoft.com/en-us/library/bb896297.aspx
Introductie LINQ tot en met part 4 doorgenomen en compiled queries plus stored procedures.

http://code.google.com/p/string-template-view-engine-mvc/
http://websitelogic.net/articles/mvc/stringtemplate-viewengine-asp-net-mvc/3
Voor het onderzoek naar template engine stringtemplate.

http://aleris.wordpress.com/2009/03/29/adding-spark-view-engine-to-aspnet-mvc-a-step-by-step-guide/
http://sparkviewengine.com/documentation/master-layouts
Onderzoek naar template engine sparkviewengine.

http://www.functionx.com/sqlserver/
SQL Server juist opzetten.

http://weblogs.asp.net/rosherove/archive/2009/03/20/test-review-1-nerddinner.aspx
Unit testing uitgelegd.

http://saezndaree.wordpress.com/2007/09/19/easy-sha1-encryption-in-aspnet/

Voor het encrypten van data.

http://articles.sitepoint.com/article/hierarchical-data-database/2
Hiërarchie in een tabel, hoe te tonen/opslaan.

http://searchsqlserver.techtarget.com/feature/SQL-Server-2008-data-types-Datetime-string-user-defined-and-more
http://msdn.microsoft.com/en-us/library/ms190215.aspx
http://msdn.microsoft.com/en-us/library/ms187745.aspx
Datatypen van SQL Server 2008.

http://www.sql-server-performance.com/articles/dba/Configure_Filestream_in_SQL_Server_2008_p1.aspx
http://experiencing-sql-server-2008.blogspot.com/2008/03/sql-server-2008-manage-unstructured.html
http://www.mssqltips.com/tip.asp?tip=1489
http://msdn.microsoft.com/en-us/library/cc949109.aspx
FILESTREAM datatype uitleg

http://www.w3schools.com/soap/soap_envelope.asp
Soap techniek.

http://www.mikesdotnetting.com/Article/129/Simple-task-Scheduling-using-Global.asax
Het maken van een task scheduler in de Global.asax.

http://webhooks.pbworks.com/
Concept webhooks uitgelegd.

http://www.matthidinger.com/archive/2008/02/21/asp.net-mvc-usercontrols-start-to-finish.aspx
User controls uitgelegd.

http://en.wikipedia.org/wiki/ASP.NET
Caching asp.net

http://www.sql-server-performance.com/articles/per/index_data_structures_p1.aspx
Clustered en non clustered index.

A.2 Literatuur

Literatuur heeft de volgende opmaak : “titel”, “auteur”, “uitgever”.

LINQ in action, Fabrice Margueri/Steve Eicher/Jim Wooley, Manning.
Gebruikt voor LINQ queries samen te stellen, compiled queries, stored procedures.

Systeemanalyse en –ontwerp, Alan Dennis en Barbara Haley Wixom, Academic Service.
Gebruikt voor het bepalen van de juiste ontwikkelstrategie en methodiek.

Dictaat Databases, Henk van den Bosch
Gebruikt voor database indelingen en normaalvormen

A.3 Anders

Huidige applicatie Multilogue voor het vastleggen van de functionaliteit en het testen ervan.

Bijlage B - Begrippenlijst

ADO.NET - ActiveX Data Objects for .NET, afgekort naar ADO.NET, is de opvolger van de eerdere uitgave van Microsoft ADO. ADO.NET is verbonden aan het .NET-framework en biedt, net als zijn voorganger, een programmeerinterface om (web-)applicaties in staat te stellen met willekeurige databases te communiceren.

ASP - Active Server Pages (**ASP**) is een door Microsoft ontwikkelde technologie om dynamische webpagina's en complete websites te genereren.

Blackbox testen – Blackbox testen is een bepaalde manier van testen waar input en output wordt gecontroleerd. Hoe het systeem het doet en hoe snel wordt niet veel aandacht aan besteed, het systeem wordt dus als een zwarte doos beschouwd.

BLOB - Een **BLOB** (**B**inary **L**arge **Ob**ject) is een - potentieel groot - gegevenselement in een database dat bestaat uit bytes waaraan in de database geen tekencodering of andere interpretatie is verbonden.

CSS – Cascading **S**tyle **S**heets. Wordt gebruikt om HTML pagina's op te maken.

Global.asax – Het Global.asax bestand staat ook wel bekend als de ASP.NET applicatie bestand. Dit is een optioneel bestand dat gebruikt kan worden voor applicatie en sessie events die gegenereerd worden door ASP.NET of door HTTP modules.

JavaScript - JavaScript is een programmeertaal die veel gebruikt wordt om webpagina's interactief. JavaScript is een client-side programmeertaal. Dat betekent dat de code aan de kant van de client wordt uitgevoerd.

LINQ – **L**anguage **IN**tegrated **Q**uery. LINQ is een onderdeel van het Microsoft .NET Framework. Het biedt een werkwijze aan voor een meer uniforme omgang met gegevens uit verschillende systemen, zoals relationele database, objecten XML bestanden.

MoSCoW - De **MoSCoW-methode** is een wijze van prioriteiten stellen.

- **Must have this** - deze eis *moet* in het eindresultaat terugkomen;
- **Should have this if at all possible** - deze eis is zeer gewenst, maar een vergelijkbare eigenschap is ook goed genoeg;
- **Could have this if it does not affect anything else** - deze eis mag alleen aan bod komen als er tijd genoeg is;
- **Would have** - deze eis zal nu niet aan bod komen maar kan in de toekomst interessant zijn.

MVC – Model **V**iew **C**ontroller**.**
In dit model geeft de view informatie weer. In de view zijn user interface elementen gedefinieerd, de view doet geen verwerkingen zoals berekeningen/controles. Het model definieert de representatie van de informatie waarmee de applicatie werkt. Het model legt

relaties tussen data en logica. De daadwerkelijke opslag van data wordt gedaan met een database.
De Controller verwerkt en reageert op events, die vanuit handelingen van de gebruiker tot stand komen.

ORM – Object-relational mapping. Dit is software dat ervoor zorgt dat datatypen in databases geconverteerd worden naar een objectgeoriënteerde programmeertaal.

PHP - PHP: Hypertext Preprocessor is een scripttaal, die bedoeld is om op webservers dynamische webpagina's te creëren.

POST (HTTP) – De **POST** methode wordt gebruikt als een gebruiker data verstuurd naar de server zoals het uploaden van een bestand of het verzenden van een ingevuld formulier.
In tegenstelling tot de GET methode die enkel een URL en headers stuurt naar de server, verstuurt POST ook een "body". Dit zorgt ervoor dat elk type van een willekeurige lengte verzonden kan worden naar de server.

Profiler - Een **profiler** is een programma voor het analyseren van de uitvoersnelheid en het geheugengebruik van andere programma's. Het is in de informatica een belangrijk gereedschap bij het ontwikkelen van programma's waarvan de snelheid essentieel is.

RPC - Een **remote procedure call**, is een technologie die een computerprogramma op één bepaalde computer toestaat om code uit te voeren op een andere machine zonder dat de programmeur de code expliciet hiervoor geschreven heeft.

RUP - Rational Unified Process is een iteratief softwareontwikkelingsproces.

SOAP - **S**imple **O**bject **A**ccess **P**rotocol is een computerprotocol dat wordt gebruikt voor communicatie tussen verschillende componenten.

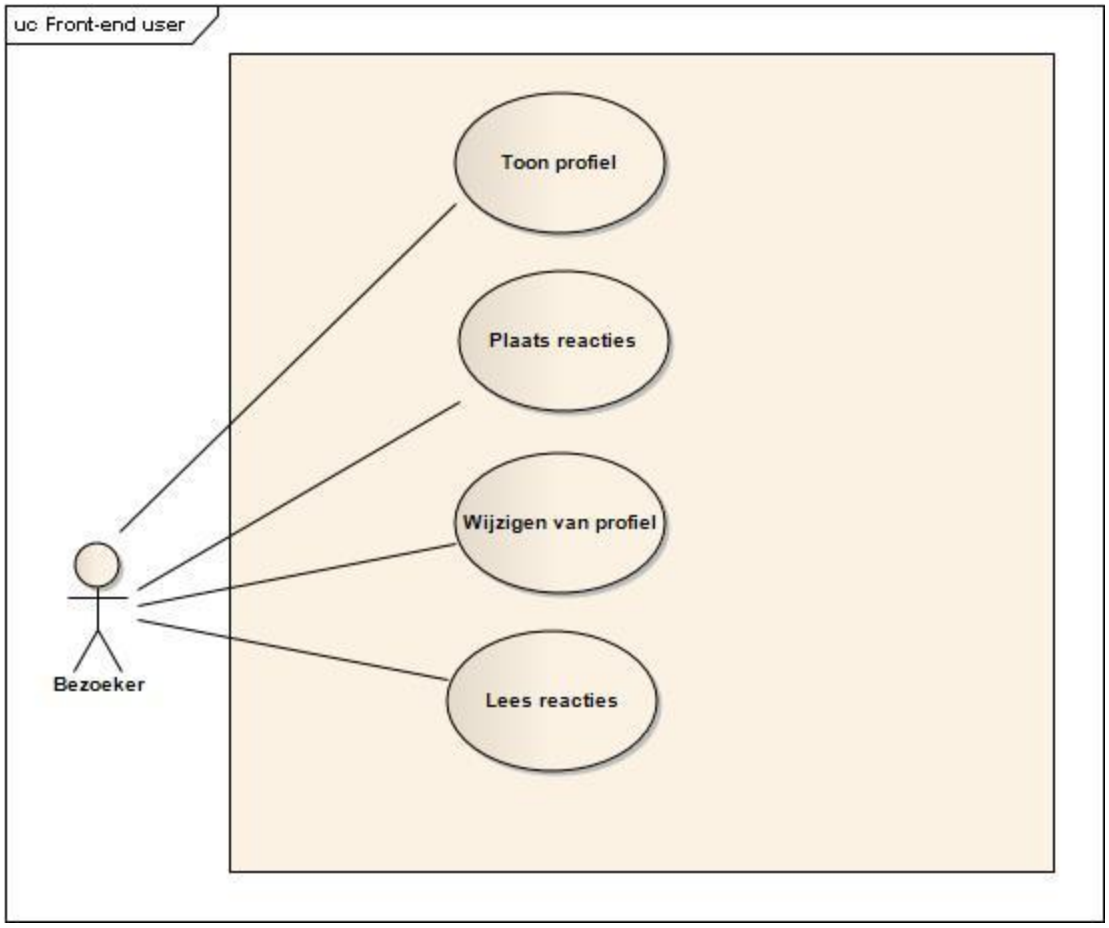
Stored procedure - Een **stored procedure** is een programma dat bewaard wordt binnen een databank. De stored procedure is aan te roepen als een functie van buitenaf of vanuit de database.

Template engine - Een **template engine** is een stuk software dat er onder andere voor zorgt dat templates(design) en code gescheiden blijft.

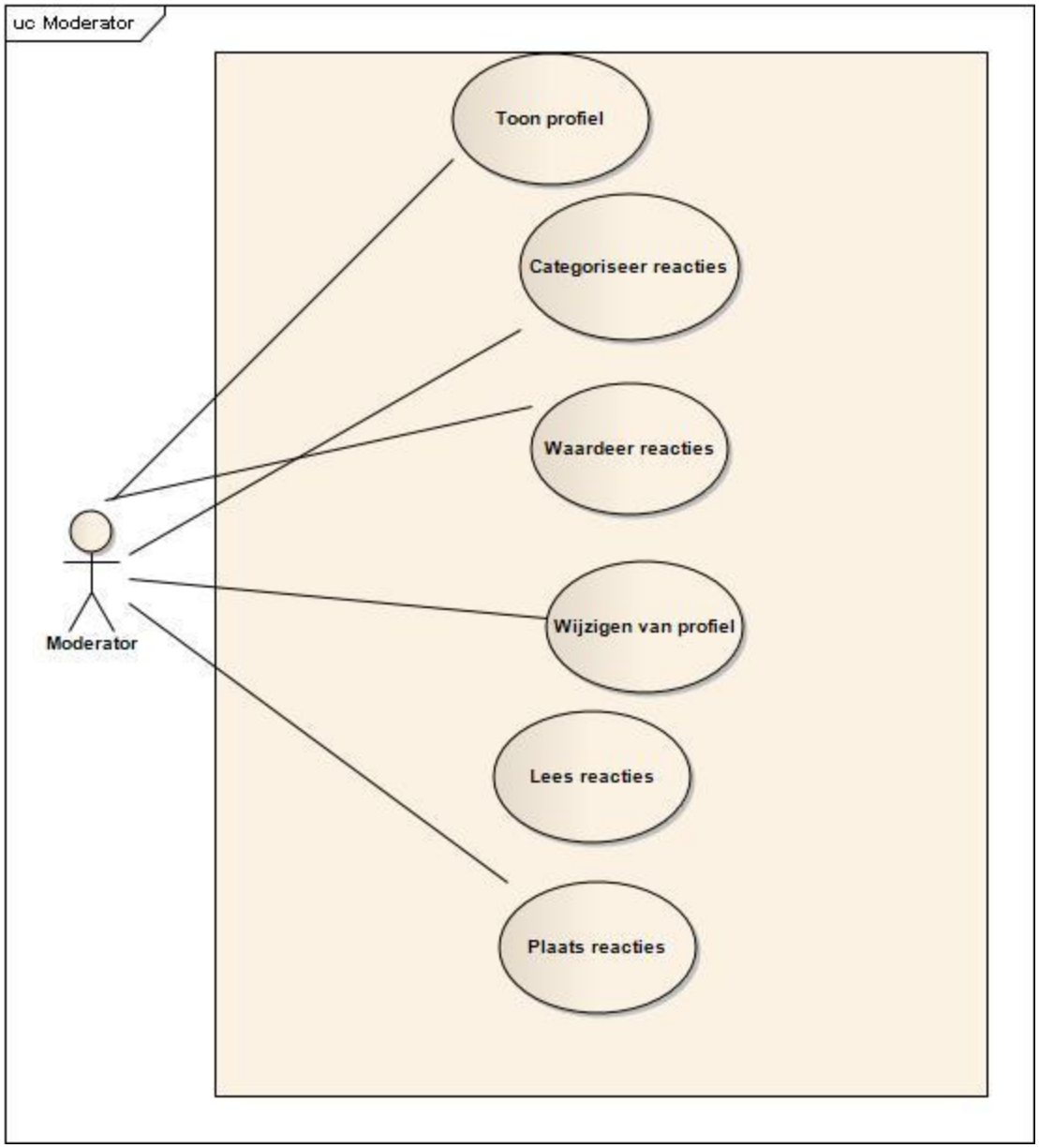
Functies die uitgevoerd kunnen worden door een template engine zijn:

- Vervangen van tekst
- Het meenemen van externe bestanden (include file)
- Loops en evaluatie

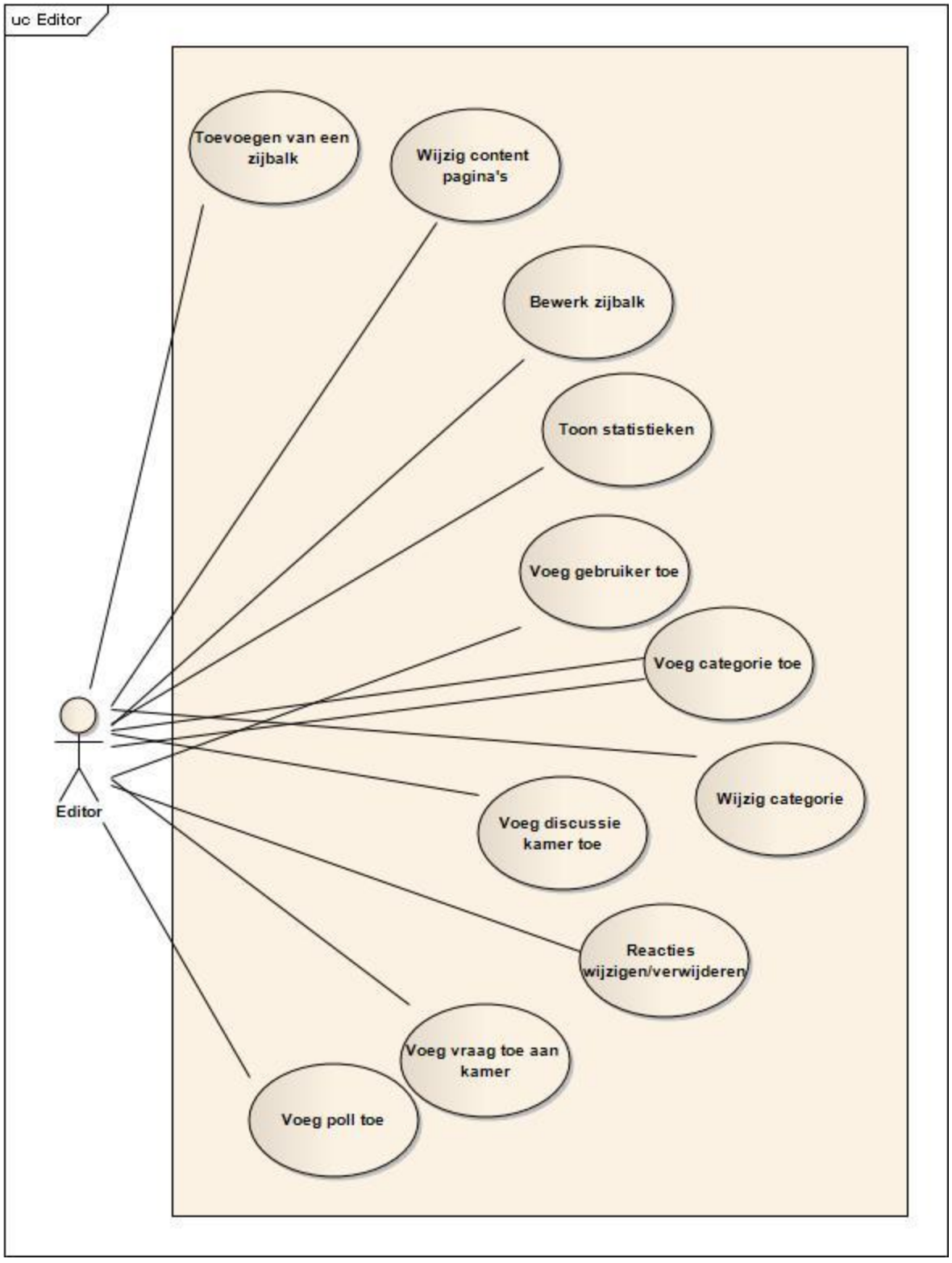
Bijlage I – Use case diagram bezoeker



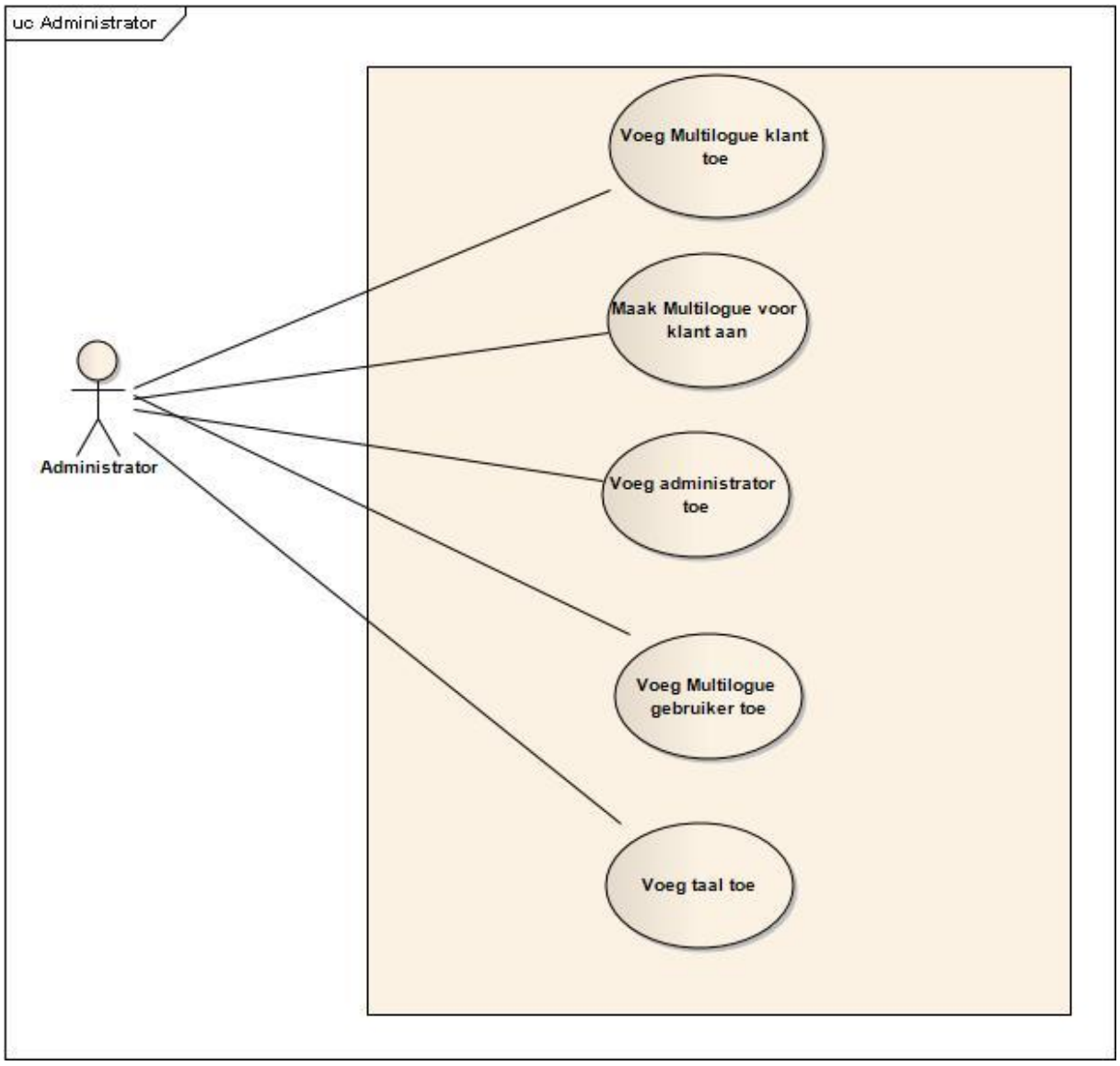
Bijlage 2 – Use case diagram moderator



Bijlage 3 – Use case diagram editor



Bijlage 4 – Use case diagram administrator



Bijlage 5 – Functionele eisen

De functionele eisen die aan het product gesteld zijn:

Gebruiker:

1) Toon profiel (*front-end*)
2) Plaats reacties (*front-end*)
3) Wijzigen van profiel (*front-end*)
4) Inloggen (*front-end*)
5) Uitloggen (*front-end*)
6) Voeg stem toe aan poll (*front-end*)
7) Kies taal (*front-end*)
8) Toon poll (*front-end*)
9) Toon "wie is online" (*front-end*)
10) Voeg reacties/vragen toe aan favorietenlijst (*front-end*)
11) Rapporteer reacties (*front-end*)
12) Bekijk video (*front-end*)
13) Bekijk laatst geschreven/gelezen reactie (*front-end*)
14) Bezoeken van kamers en lezen van reacties (*front-end*)

Moderator:

15) Categoriseer reacties (*front-end*)
16) Waardeer reacties (*front-end*)

Editor:

17) Toevoegen van een zijbalk (*editor back-end*)
18) Bewerk zijbalk (*editor back-end*)
19) Wijzig content van pagina's (*editor back-end*)
20) Toon statistieken (*editor back-end*)
21) Voeg categorie toe (*editor back-end*)
22) Wijzig categorie (*editor back-end*)
23) Voeg gebruiker toe (*editor back-end*)
24) Voeg discussiekamer toe (*editor back-end*)
25) Voeg vraag toe aan kamer (*editor back-end*)
26) Reacties verwijderen/wijzigen (*front-end*)
27) Voeg poll toe (*editor back-end*)
28) Inloggen (*editor back-end/ front-end*)
29) Uitloggen (*editor back-end/ front-end*)
30) Toon vertalingen (*editor back-end*)
31) Voeg formvelden toe (*editor back-end*)
32) Toon formvelden (*editor back-end*)
33) Voeg woorden toe aan blacklist (*editor back-end*)
34) Zoek gebruiker (*editor back-end*)
35) Instellen van infobericht (*editor back-end*)
36) Toon polls (*editor back-end*)
37) Volg gebruiker (*editor back-end*)
38) Voeg Multilogue bericht toe (*editor back-end*)

39) Voeg gebruiker toe aan groep (*editor back-end/ front-end*)
40) Maak groepen aan (*editor back-end/ front-end*)

Administrator:

41) Toon geïnstalleerde Multilogues (*administrator back-end*)
42) Voeg administrator toe (*administrator back-end*)
43) Voeg Multilogue gebruiker toe (*administrator back-end/ editor back-end*)
44) Voeg taal toe (*administrator back-end*)
45) Maak Multilogue voor klant aan (*administrator back-end*)
46) Inloggen (*editor back-end/ front-end / administrator back-end*)
47) Uitloggen (*editor back-end/ front-end / administrator back-end*)
48) Toon overzicht Multilogue klanten (*administrator back-end*)
49) Toon administrators (*administrator back-end*)

Bijlage 6 – Niet functionele eisen

De niet functionele eisen zijn opgedeeld in:
1) Operationeel
2) Prestatief
3) Beveiliging

ad1 Operationeel

Met een operationele niet functionele eis wordt de fysieke en technische omgeving bedoeld waarin het systeem gaat functioneren.

Bij de operationele eisen moet gedacht worden aan:

- Het systeem moet met de applicatie E-mark mail kunnen werken. Dit wordt gebruikt om massamail te verzenden tijdens het aanmaken van gebruikers met een csv lijst door de editor.
- De applicatie moet met Firefox en IE7 en hoger kunnen werken.
- ASP.NET server benodigd(Internet Information Server). Dit is IIS 7 SQL SERVER 2008.

ad2 Prestatief

Deze niet functionele eis kan omschreven worden als de snelheid, kracht en betrouwbaarheid van het systeem.

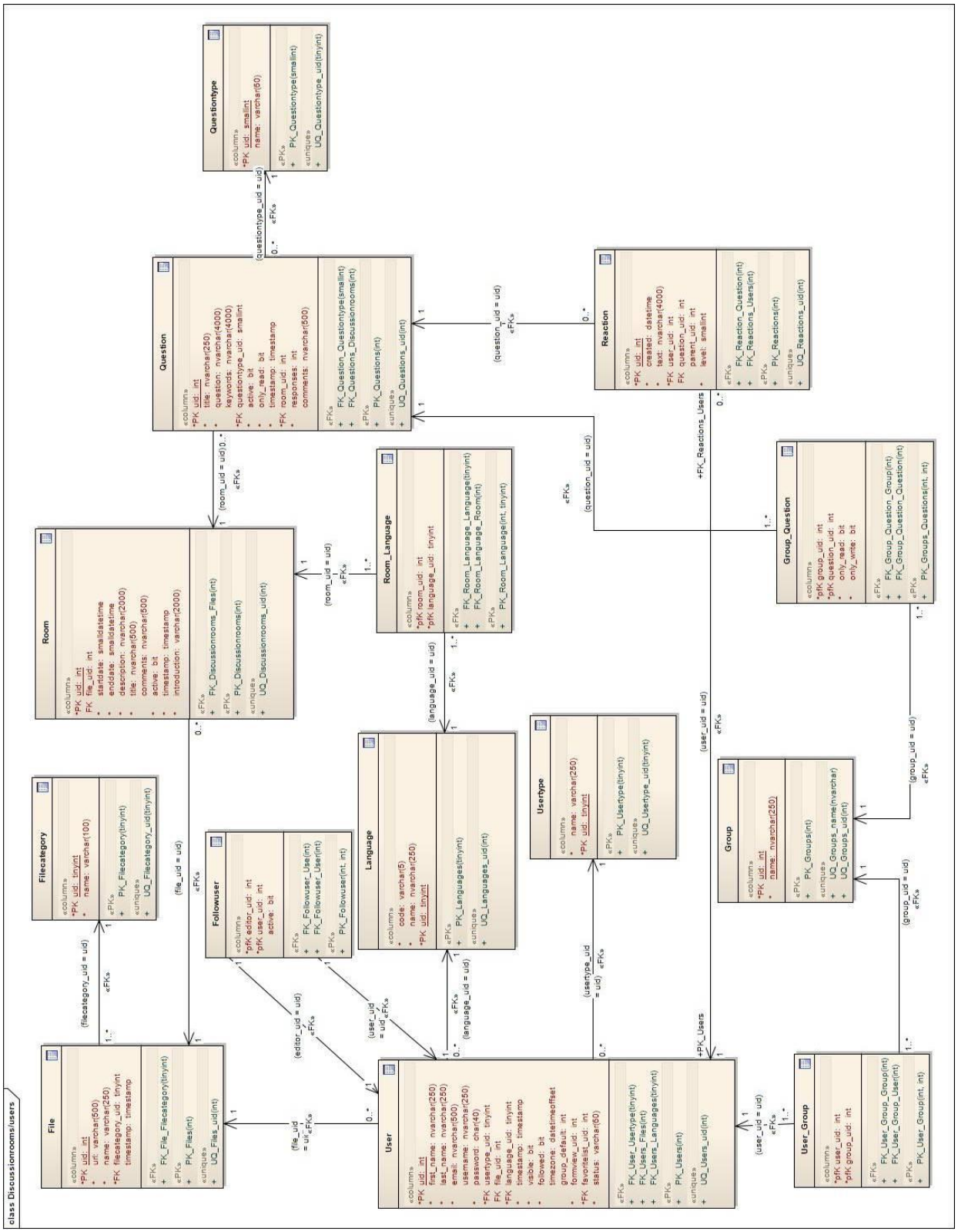
- Applicatie moet opgebouwd zijn uit modules.
- Het ontwerp maakt gebruik van een 3 TIER model. Dit zijn front, runtime (ook wel business logic genoemd) en de data acces layer.
- De ASP.NET dient sneller te zijn dan de PHP maatapplicatie.
- Installatie van applicatie moet zo min mogelijk moeite kosten. Administrator database moet gescheiden zijn van de Multilogue database. De mogelijkheid om te communiceren tussen de twee databases moet aanwezig zijn.

ad3 Beveiliging

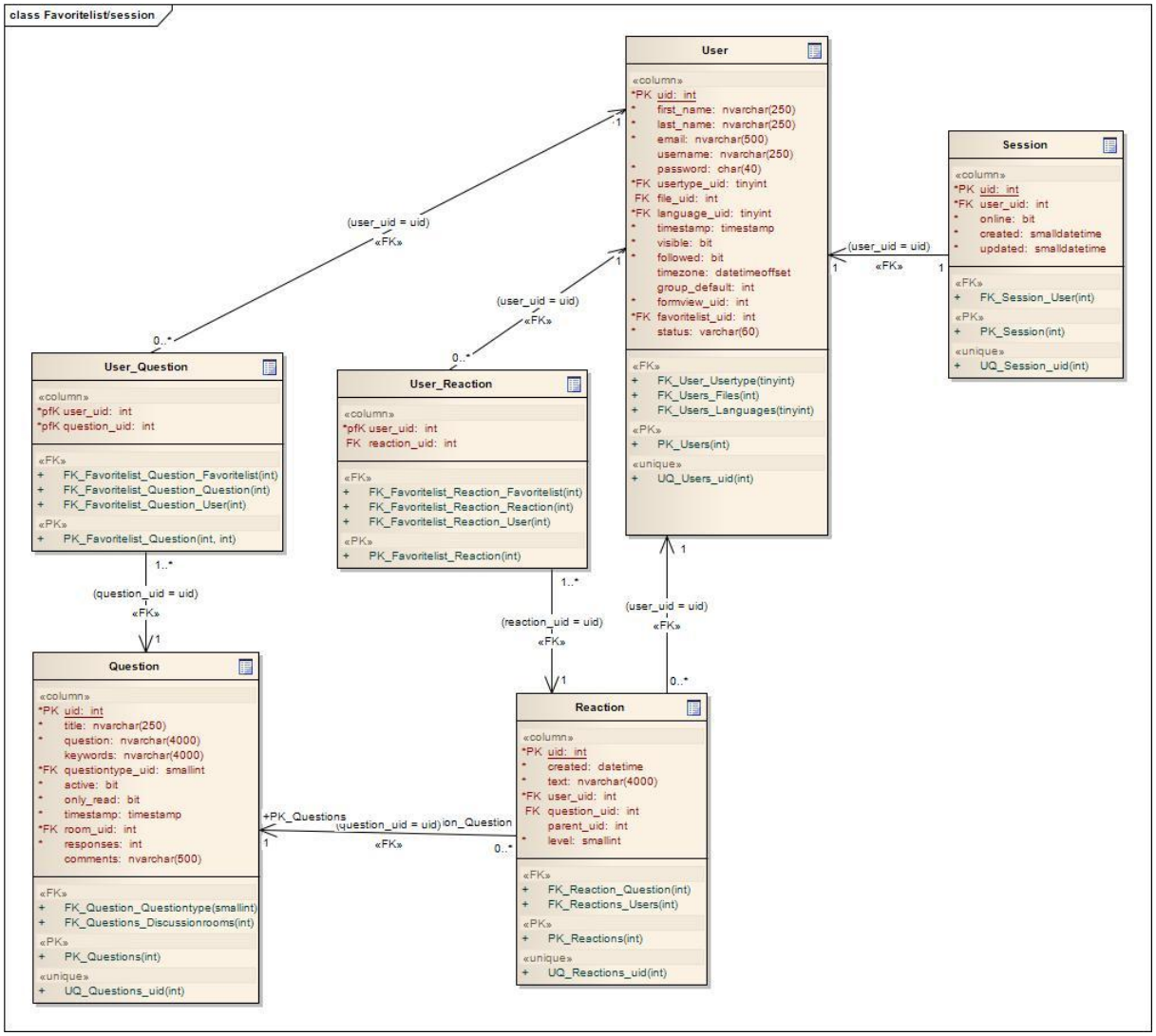
Beveiliging richt zich op bevoegde toegang tot het systeem.
Denk hierbij aan:

- Gescheiden rechten van verschillende typen gebruikers (zie use cases voor toegestane handelingen)
- Wachtwoorden zijn “ge-encrypt” opgeslagen in de database.

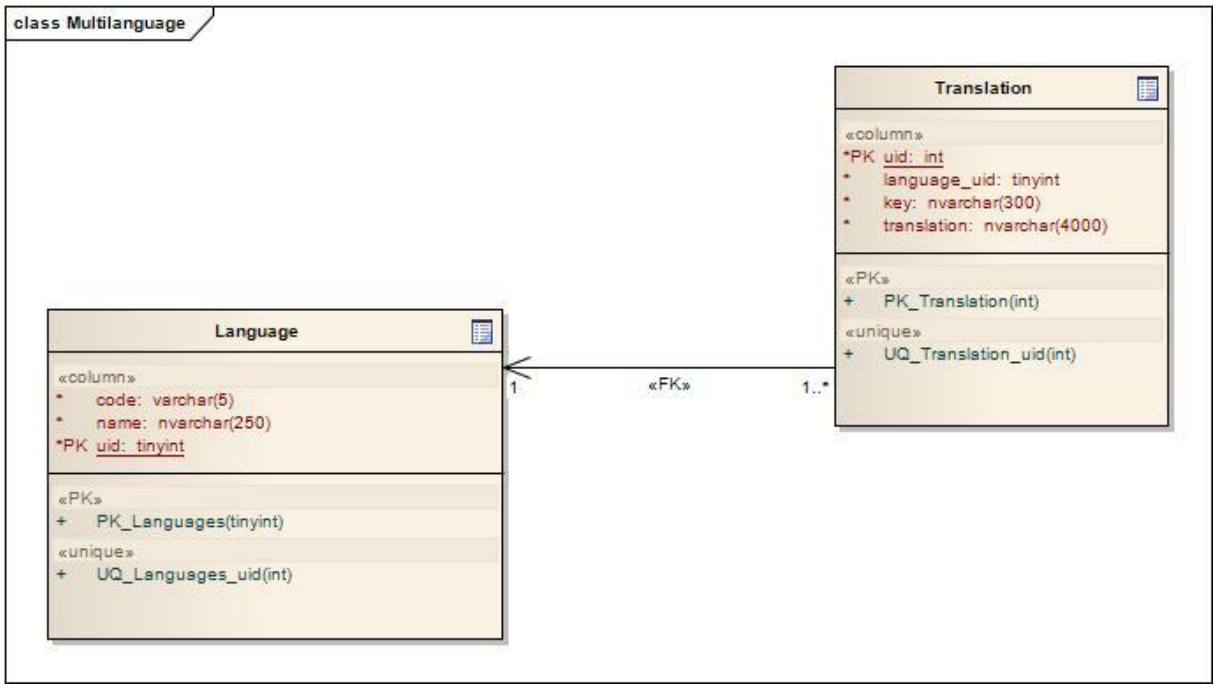
Bijlage 7 – Kamers en gebruikers (Multiloguedatabase)



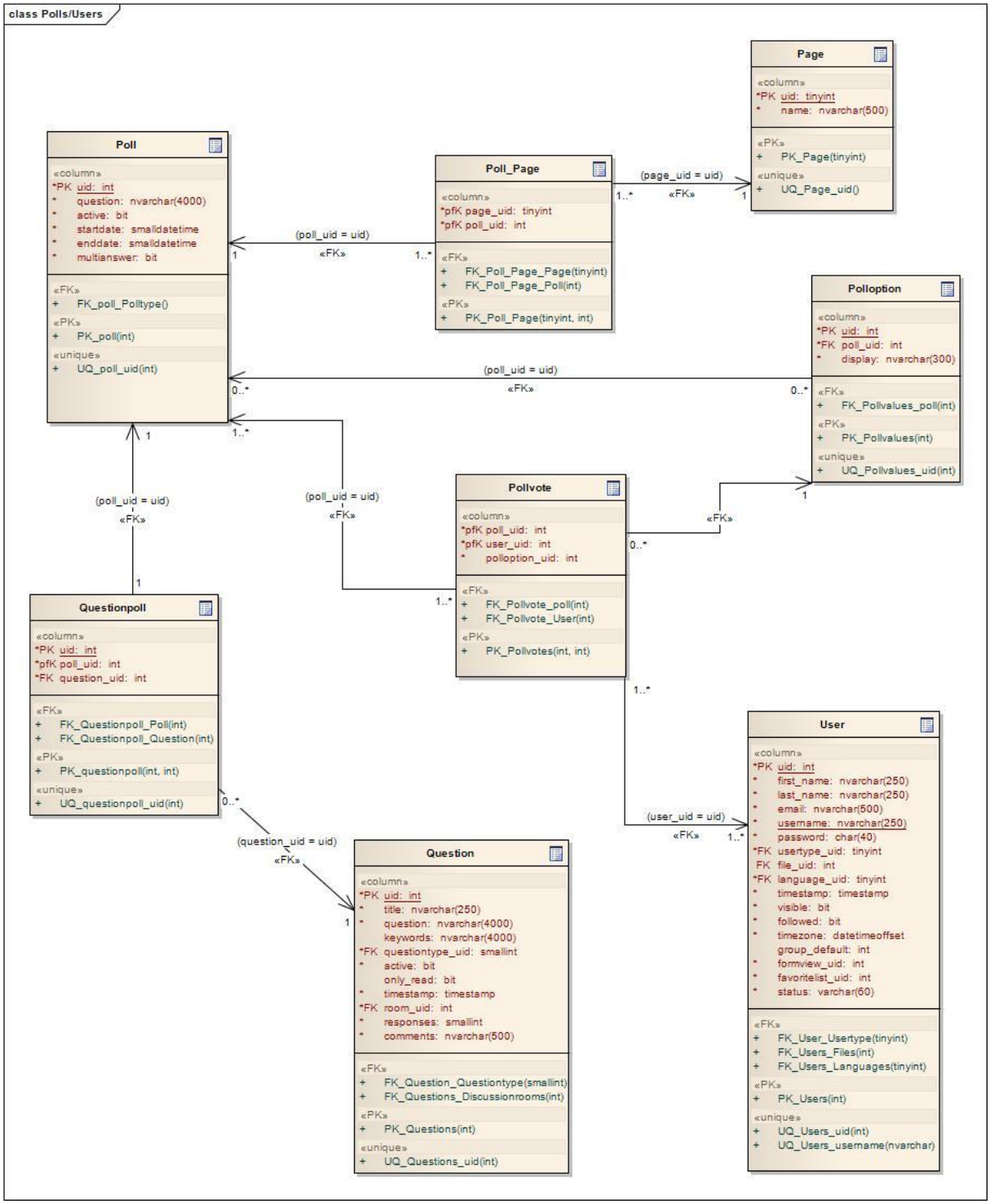
Bijlage 8 - Favorietenlijst en sessies (Multiloguedatabase)



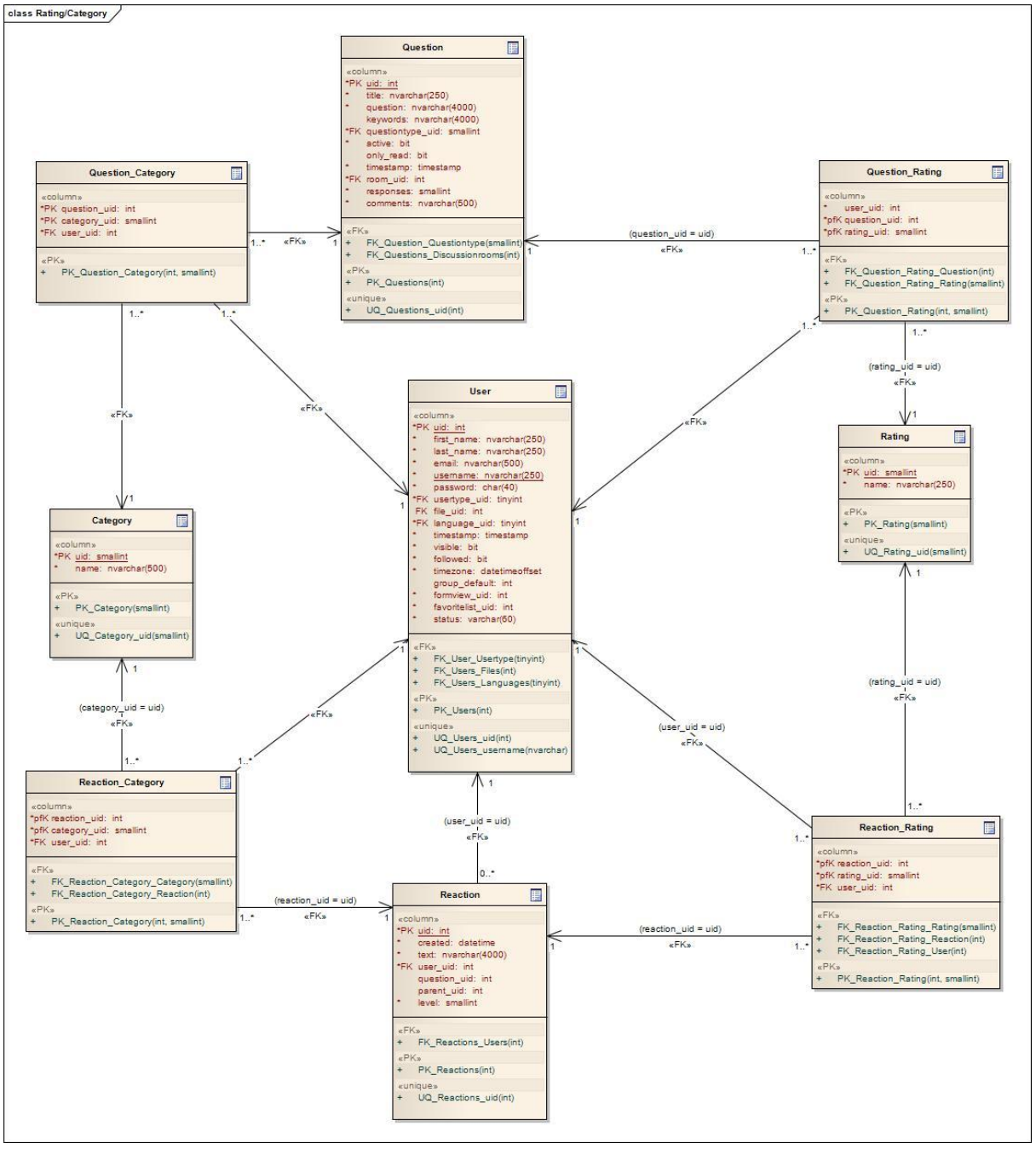
Bijlage 9 – Multilanguage (Multiloguedatabase)



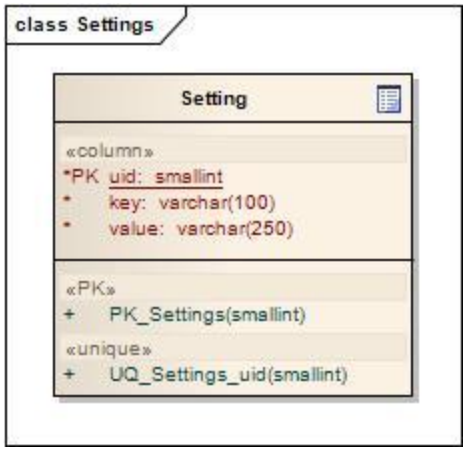
Bijlage 10 - Polls en gebruikers (Multilogue database)



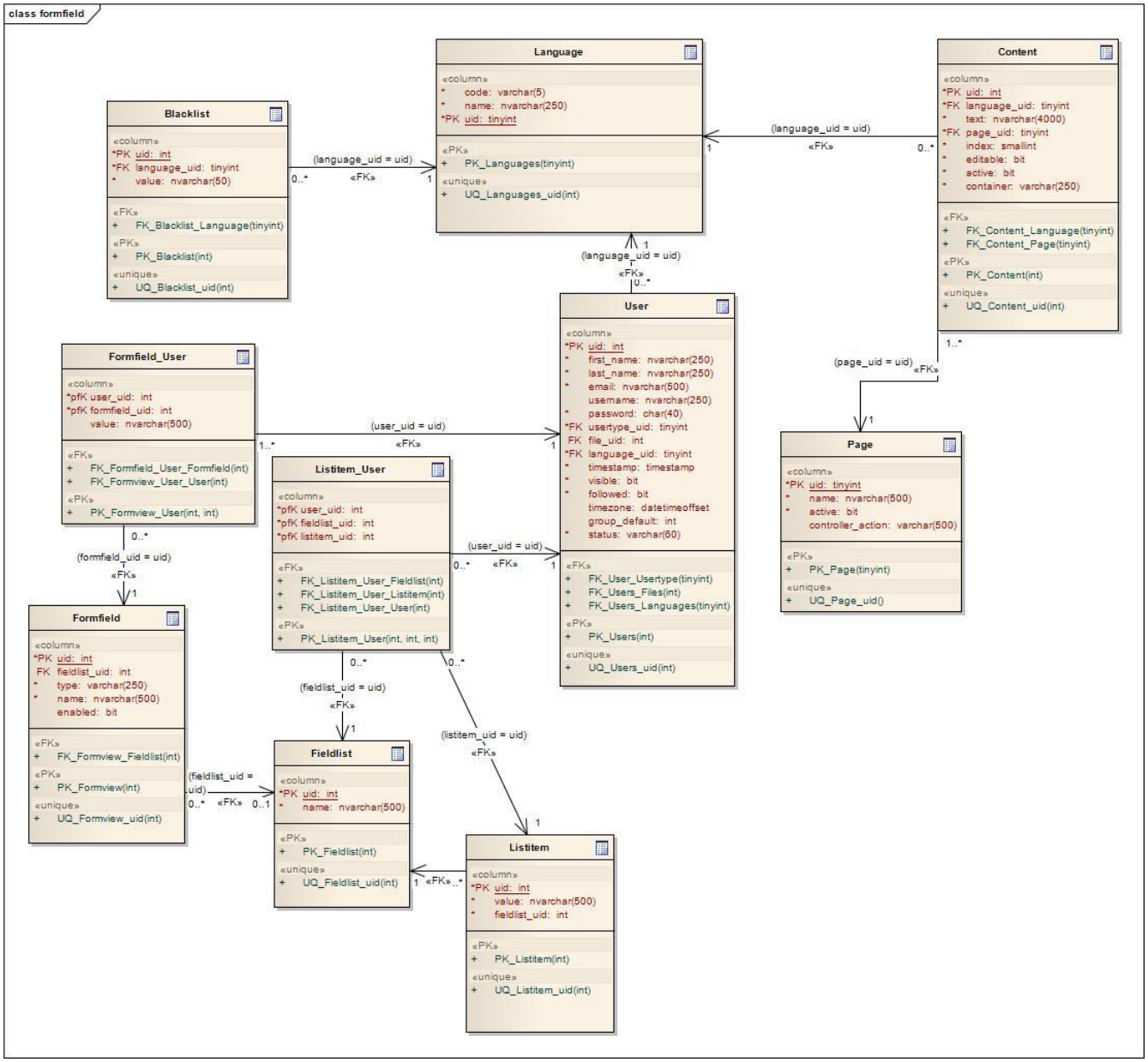
Bijlage 11 - Waarderen en categoriseren (Multiloguedatabase)



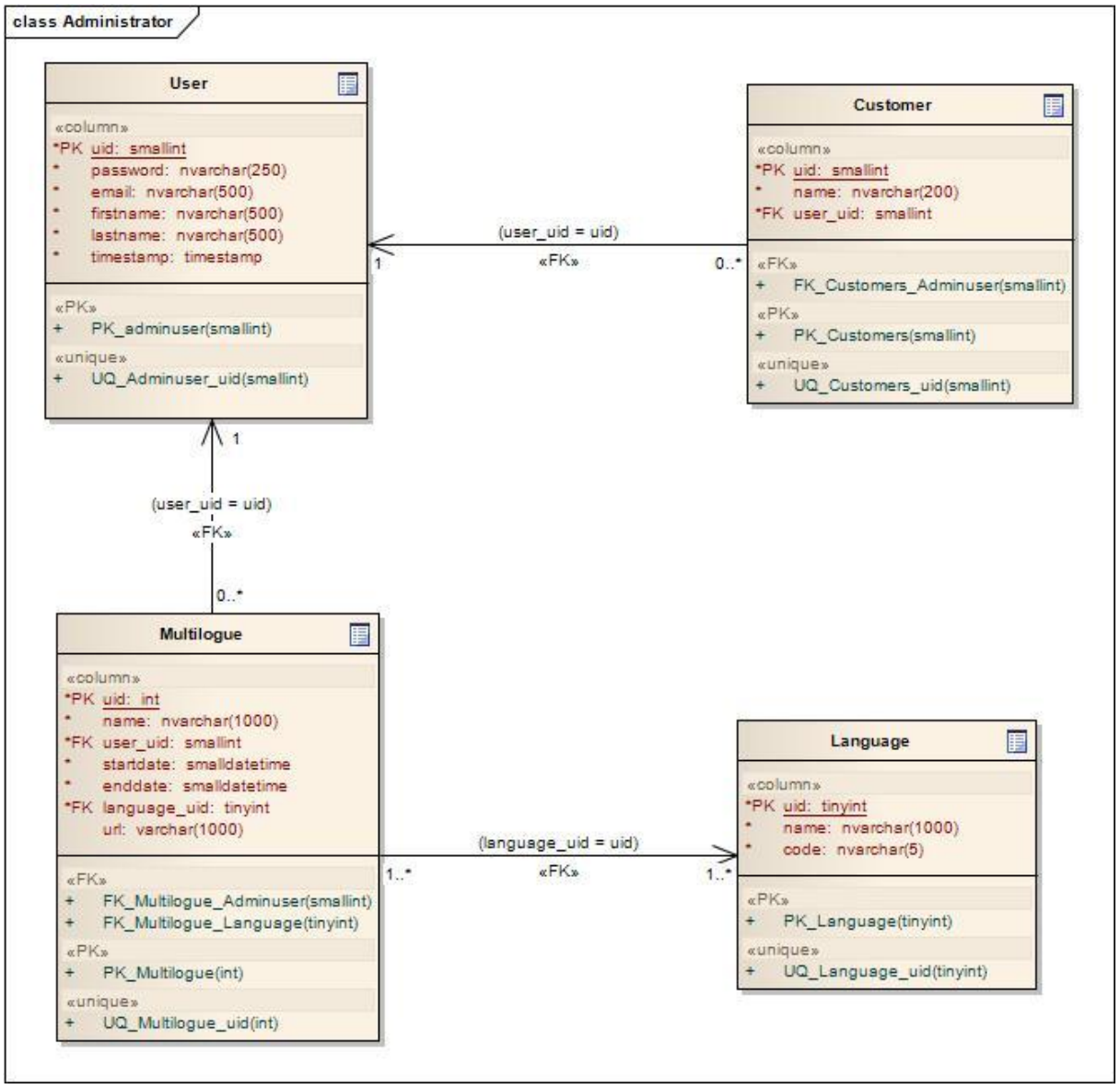
Bijlage 12 – Settings (Multiloguedatabase)



Bijlage 13 – Formfield (Multiloguedatabase)



Bijlage 14 – Administrator (Administratordatabase)



Bijlage 15 – Sitemap

Deze bijlage toont de sitemaps van de applicatie. Een sitemap is de structuur van een webapplicatie. Dat betekent de bezoekbare pagina's voor een gebruiker. Hierbij is al rekening gehouden met het MVC model. Als voorbeeld User, - login. Hierbij is User een controller genaamd UserController en Login een actie(functie).

ad1 – Front-end

De pagina's voor de front-end zijn:

User
- -Login
- -View
- -Edit
- -List

Page
- -Background
- -Help

Home
- -Index

Room
- -List
- -View

Question
- -Add
- -Edit (voor editor)
- -View

Reaction
- -Report
- -Categorize (voor moderator/editor)
- -Rate (voor moderator/editor)

Video
- -List
- -View

Poll
- -View
- -Add

ad2 – Editor Back-end

Pagina's in de back-end voor de editor zijn:

Home
- -Index

Content
- -ViewSidebars
- -AddSidebar
- -EditSidebar

- ViewPagecontent
- EditPagecontent

User
- Login
- Add
- AddUsers
- Search
- Follow
- View

Blacklist
- List
- Add

Room
- Edit
- Add
- Index

Question
- Edit
- Add

Statistic
- General
- Room
- Poll
- Responses

Category
- List
- Add
- Edit

Rating
- List
- Add
- Edit

Formfield
- List
- Add
- Edit

Fieldlist
- List
- Add
- Edit

Poll
- Add
- List
- Edit

Translation
- Edit
- View
- List

ad3 – Administrator back-end

De back-end van de administrator is gescheiden van de back-end van de editor en de front-end van de Multilogue.
Pagina's voor de administrator zijn:

Home
- -Index

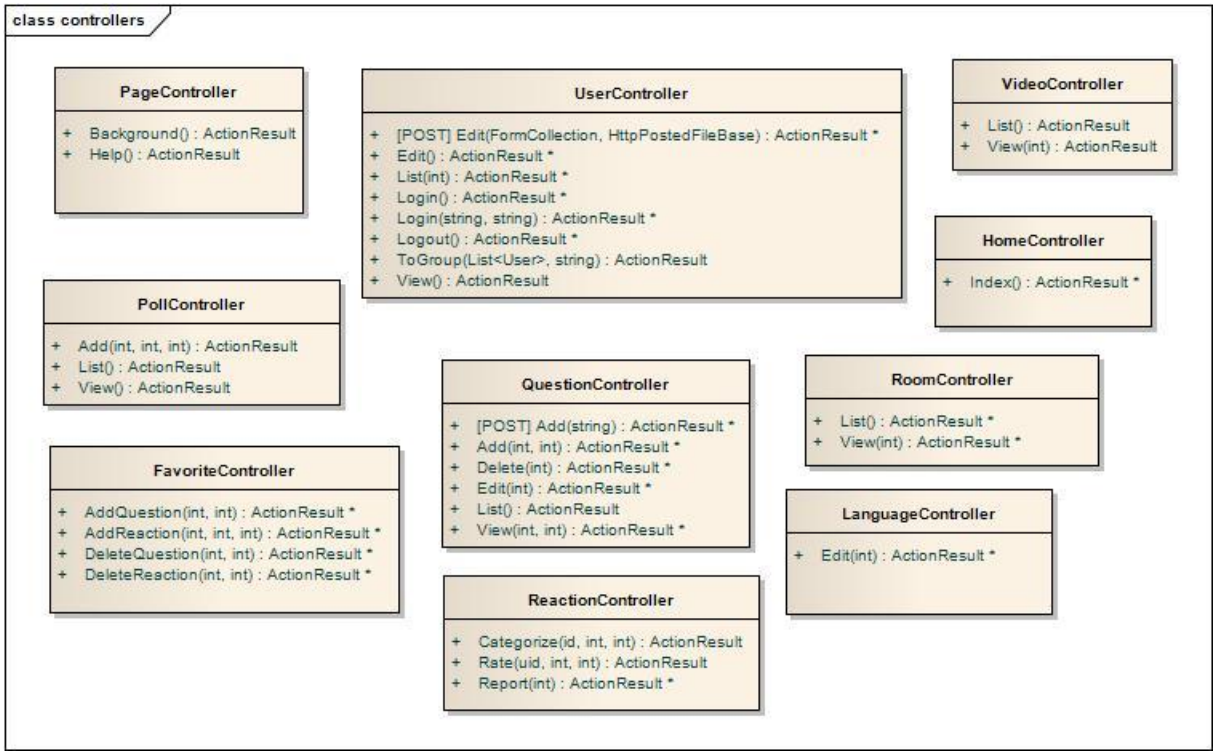
User
- -Login
- -Add
- -List

Customer
- -List
- -Add
- -Edit

Multilogue
- -List
- -Add
- -Edit

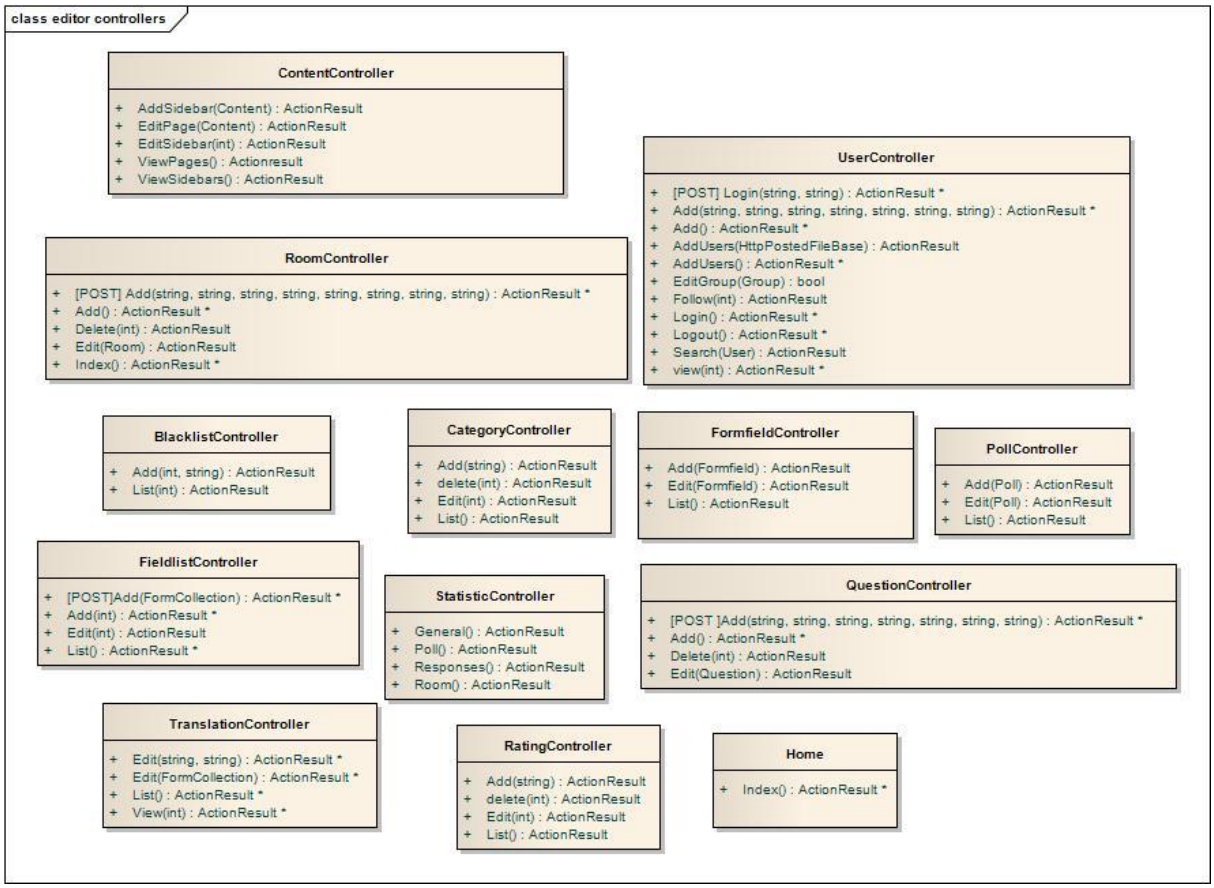
Language
- -List
- -Edit
- -Add

Bijlage 16 – Controllers front-end



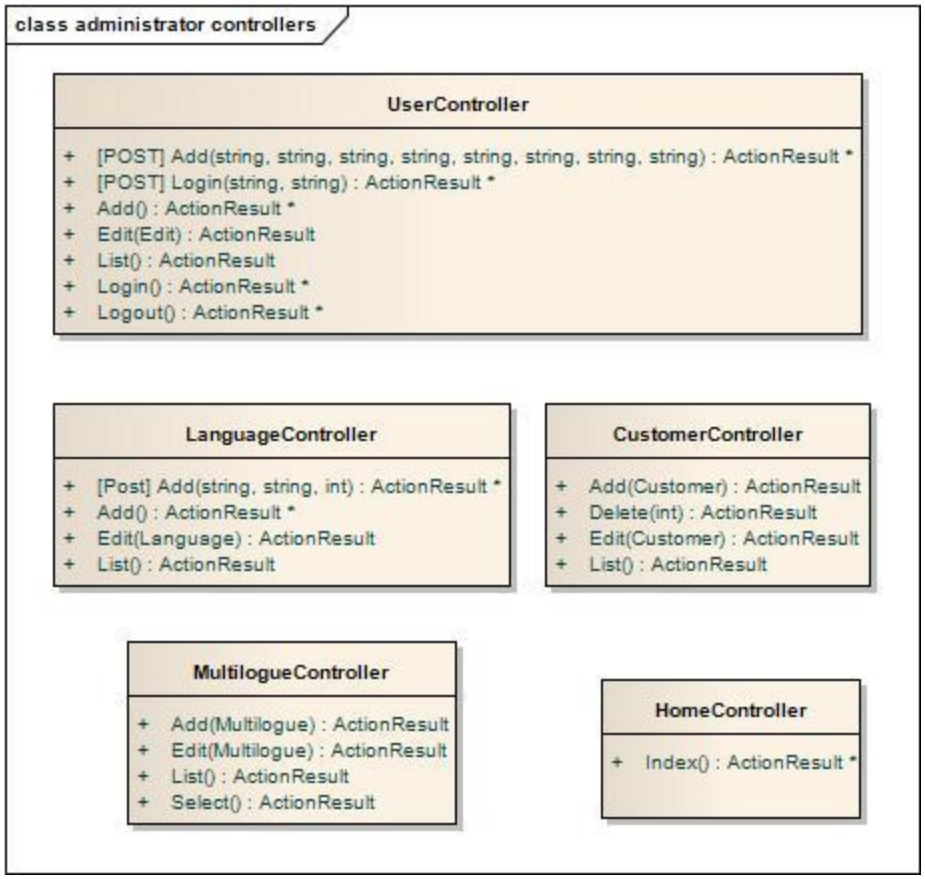
Functies met een * achter de functie zijn geïmplementeerd.

Bijlage 17 – Editor controllers



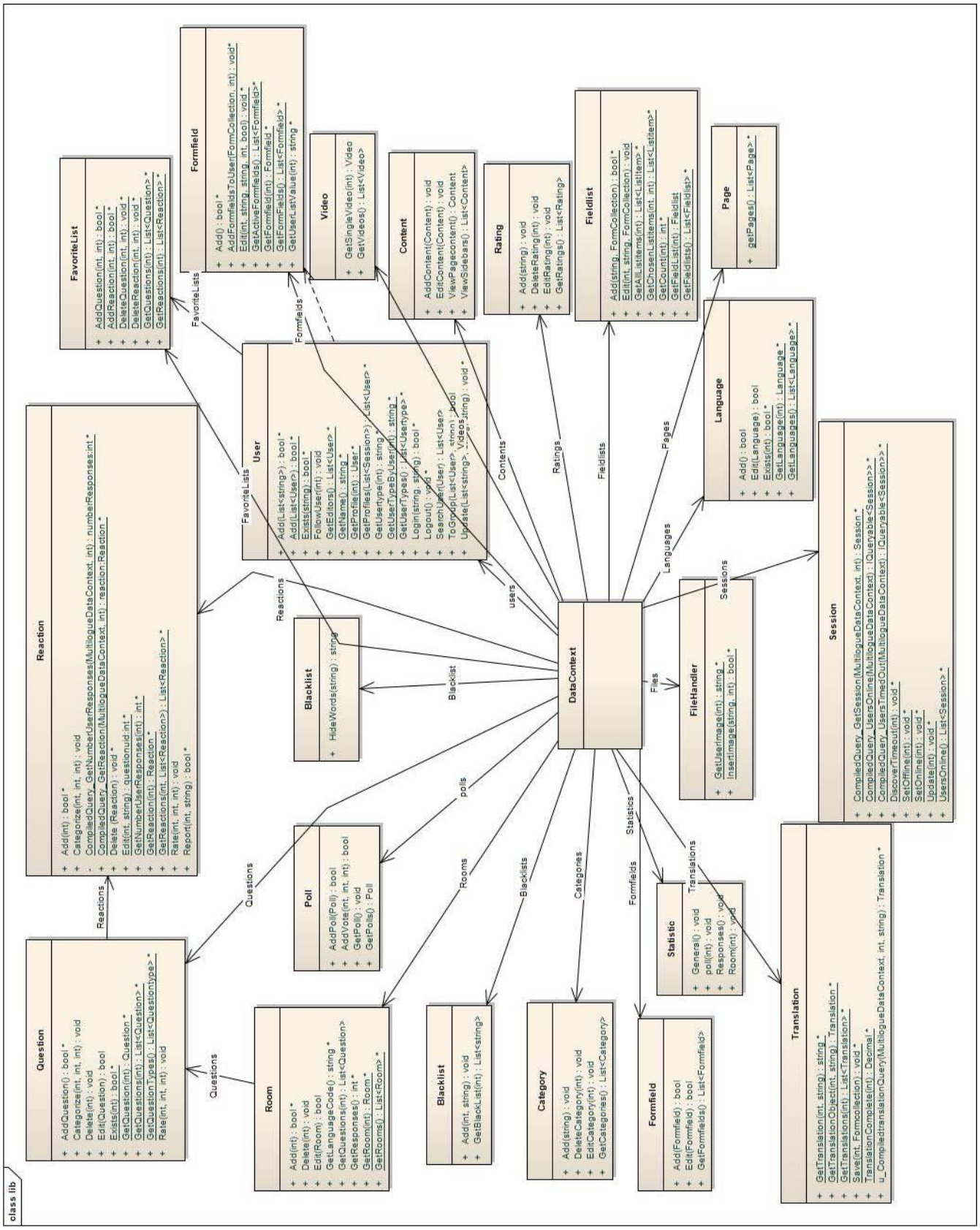
Functies met een * achter de functie zijn geïmplementeerd.

Bijlage 18 – Administrator controllers



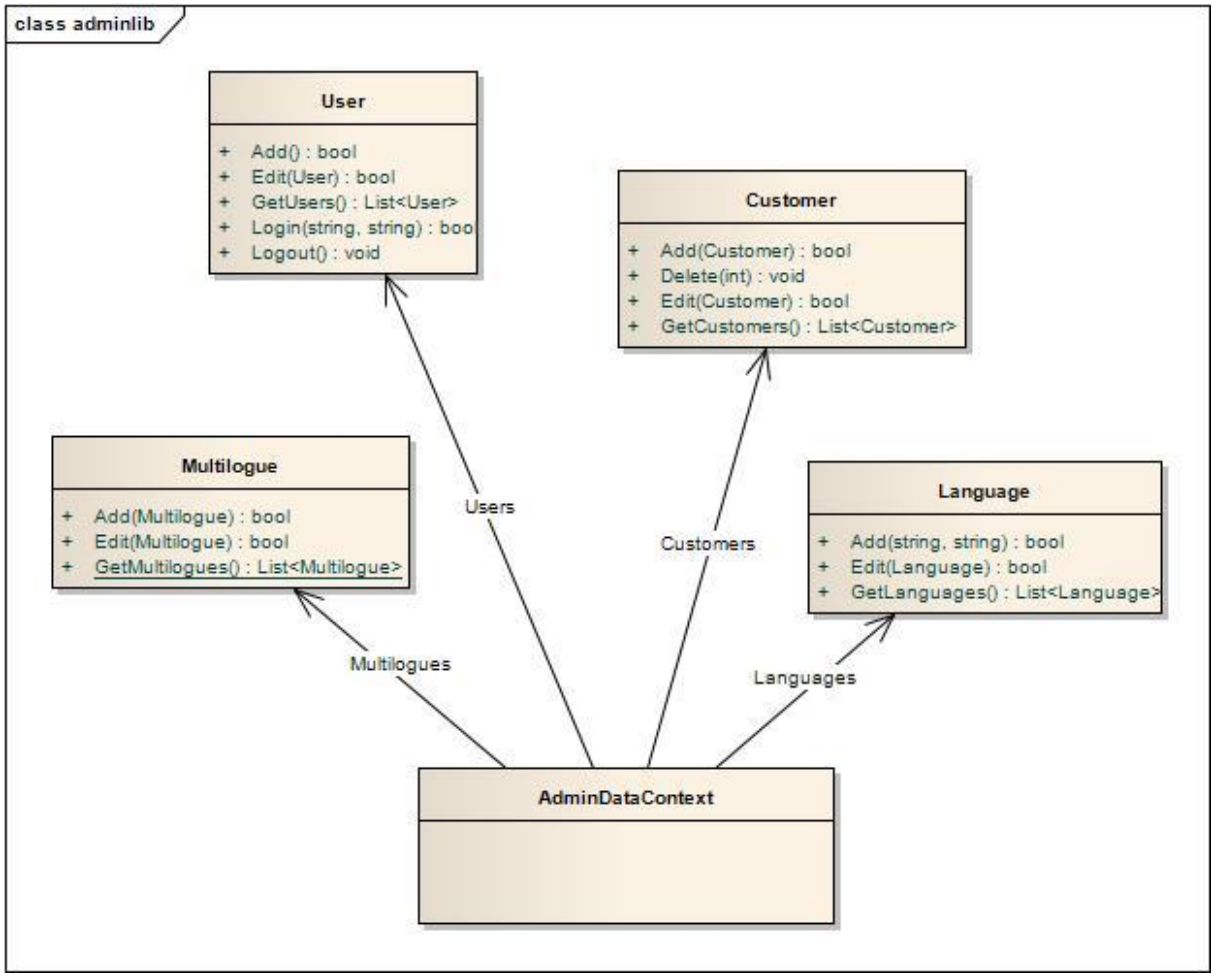
Functies met een * achter de functie zijn geïmplementeerd.

Bijlage 19 – Lib (front-end/editor back-end)

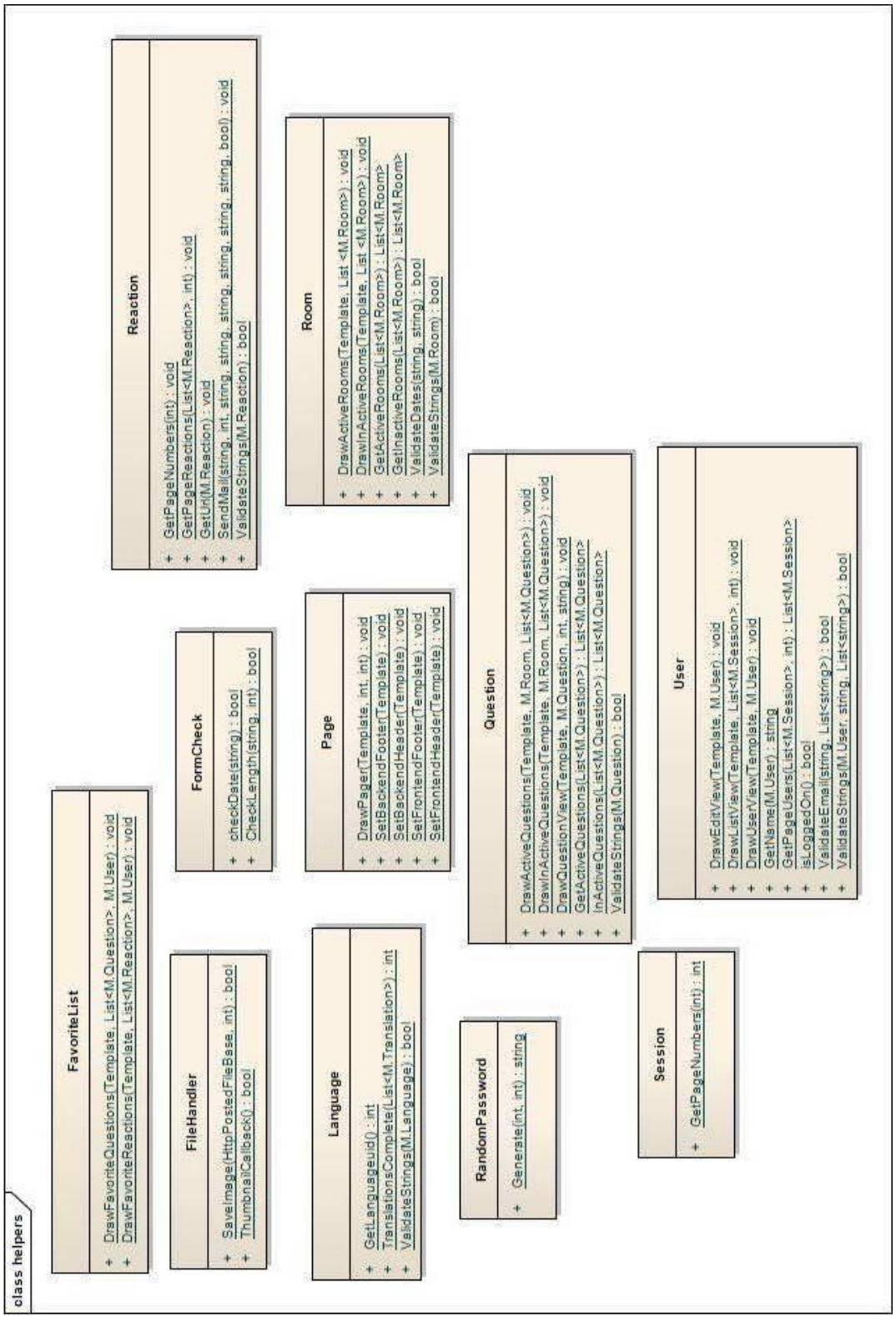


Functies met een * achter de functie zijn geïmplementeerd.

Bijlage 20 – Lib administrator



Bijlage 21 – Helpers(front-end/editor back-end)



Bijlage 22 – Toevoegen gebruiker vanuit admin

Bijlage 23 – Prioriteiten eisen

Must haves:

Eis	**Soort eis**
Toon profiel	Front-end
Plaats reacties	Front-end
Inloggen	Front-end
Uitloggen	Front-end
Kies taal	Front-end
Bezoeken van kamers en lezen van reacties	Front-end
Voeg gebruiker toe	Back-end editor
Voeg discussiekamer toe	Back-end editor
Voeg vraag toe aan kamer	Back-end editor
Inloggen	Back-end editor
Uitloggen	Back-end editor
Inloggen	Back-end administrator
Uitloggen	Back-end administrator
Voeg taal toe	Administrator/editor
Voeg Multilogue gebruiker toe	Administrator/editor

Should haves:

Eis	**Soort eis**
Wijzigen profiel	Front-end
Toon “wie is online”	Front-end
Voeg reacties/vragen toe aan favorietenlijst	Front-end
Rapporteer reacties	Front-end
Reacties verwijderen/wijzigen	Front-end editor
Voeg formvelden toe	Back-end editor
Toon formvelden	Back-end editor
Instellen van infobericht	Back-end editor
Voeg Multilogue bericht toe	Back-end editor

Could haves:

Eis	**Soort eis**
Voeg stem toe aan poll	Front-end
Toon poll	Front-end
Bekijk laatst geschreven/gelezen reactie	Front-end
Categoriseer reacties	Front-end, moderator
Waardeer reacties	Front-end, moderator
Zijbalk toevoegen	Back-end editor
Bewerk zijbalk	Back-end editor
Wijzig content van pagina's	Back-end editor
Toon statistieken	Back-end editor
Voeg categorie toe	Back-end editor
Wijzig categorie	Back-end editor
Voeg poll toe	Back-end editor
Toon vertalingen	Back-end editor
Voeg woorden toe aan blacklist	Back-end editor
Zoek gebruiker	Back-end editor
Toon polls	Back-end editor
Volg gebruiker	Back-end editor
Maak groepen aan	Back-end editor
Voeg gebruiker toe aan groep	Back-end/front-end editor
Maak Multilogue aan	Back-end administrator
Toon overzicht Multilogue klanten	Back-end administrator
Toon geïnstalleerde Multilogues	Back-end administrator

Wanna haves:

Eis	**Soort eis**
Toon administrators	Back-end administrator
Voeg administrator toe	Back-end administrator

Bijlage 24 – Tonen van reacties

Multilogue **Search**

Home Discussion Room Video My Profile Background Help
Welcome back, michel dickhoff
English Dutch French Who's online Logout

Starter question

vraag7 Add to favorite
Respond

previous 1 next


#ID:71

nilsss corver Report
28-5-2010 14:28:56 edit
hoi delete

React Add to favorite


#ID:63

michel dickhoff Report
7-5-2010 15:26:58 edit
reactie delete

React Add to favorite


#ID:62

michel dickhoff Report
7-5-2010 15:26:49 edit
sdffsa delete

Bijlage 25 – Soap XML User

```
POST /service.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <soap12:Body>
    <AddUser xmlns="http://tempuri.org/">
      <userSoapobject>
        <Firstname>string</Firstname>
        <Lastname>string</Lastname>
        <Email>string</Email>
        <Password>string</Password>
        <Username>string</Username>
        <Usertype>unsignedByte</Usertype>
        <Status>string</Status>
      </userSoapobject>
    </AddUser>
  </soap12:Body>
</soap12:Envelope>
```

Request voor functie AddUser

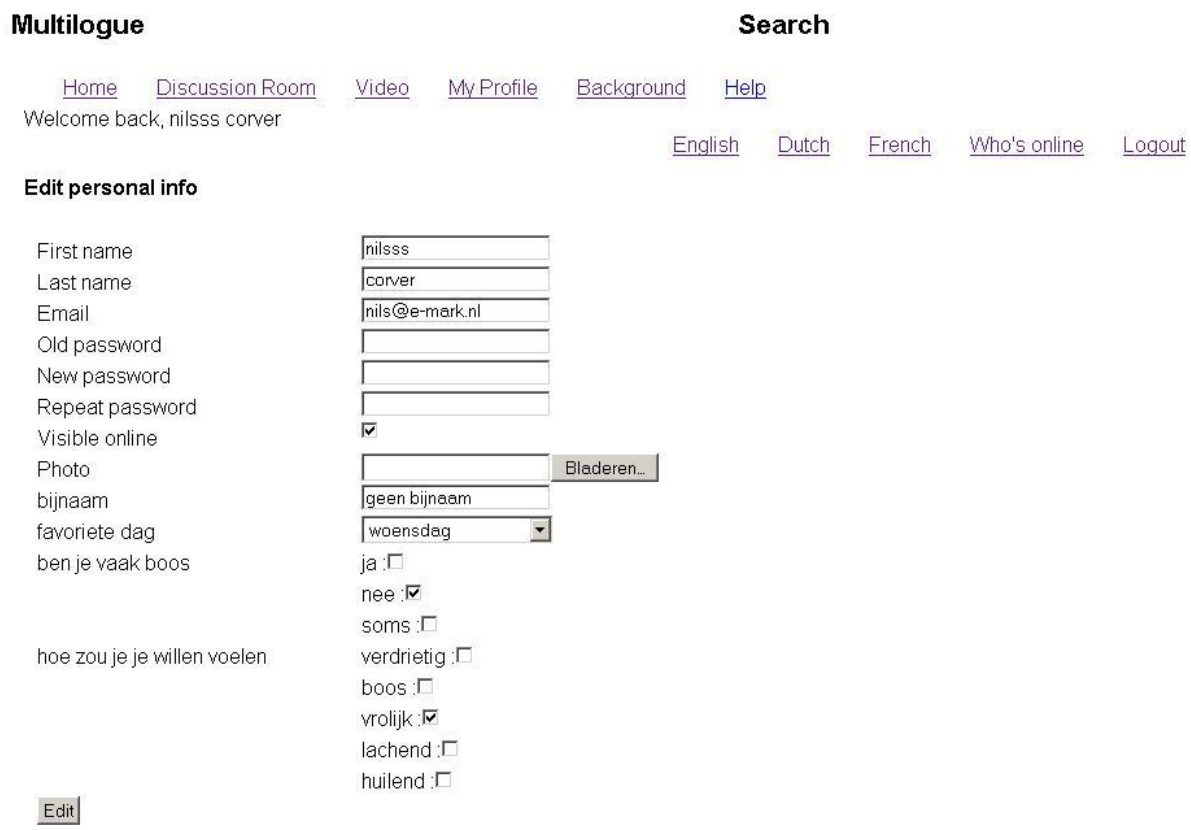
```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <soap12:Body>
    <AddUserResponse xmlns="http://tempuri.org/">
      <AddUserResult>boolean</AddUserResult>
    </AddUserResponse>
  </soap12:Body>
</soap12:Envelope>
```

Antwoord van functie AddUser

Bijlage 26 – Filestream benodigde class

```
public class NativeSqlClient
    {
        public enum DesiredAccess : uint
        {
            Read,
            Write,
            ReadWrite,
        }

        [DllImport("sqlncli10.dll", SetLastError = true, CharSet =
CharSet.Unicode)]
        private static extern SafeFileHandle OpenSqlFilestream(
          string path,
          uint access,
          uint options,
          byte[] txnToken,
          uint txnTokenLength,
          Sql64 allocationSize);

        [StructLayout(LayoutKind.Sequential)]
        private struct Sql64
        {
            public Int64 QuadPart;
            public Sql64(Int64 quadPart)
            {
                this.QuadPart = quadPart;
            }
        }

        public static SafeFileHandle GetSqlFilestreamHandle
         (string filePath, DesiredAccess access, byte[] txnToken)
        {
            SafeFileHandle handle = OpenSqlFilestream(
              filePath,
              (uint)access,
              0,
              txnToken,
              (uint)txnToken.Length,
              new Sql64(0));

            return handle;
        }

    }
```

Bijlage 27 – Edit pagina gebruiker



Bijlage 28 – koppelen van formfields aan gebruiker

```
public static void AddFormfieldsToUser(FormCollection collection, int useruid)
{
    using (MultilogueDataContext db = new MultilogueDataContext())
    {
        foreach (string formfielduid in collection.AllKeys)
        {
            class System.String
            Represents text as a series of Unicode characters.lds.FirstOrDefault(ff => ff.uid == Convert.ToInt32(formfielduid));
            if (collection[formfielduid].ToString().Length != 0)
            {
                switch (formfield.type)
                {
                    case "text":
                        Formfield_User formfieldUser = db.Formfield_Users.FirstOrDefault(f_u => f_u.formfield_uid == formfield.uid && f_u.user_uid == useruid);
                        if (formfieldUser != null) //edit row
                        {
                            formfieldUser.value = collection[formfielduid];
                        }
                        else //create new one
                        {
                            Formfield_User newFormfieldUser = new Formfield_User();
                            newFormfieldUser.user_uid = useruid;
                            newFormfieldUser.formfield_uid = formfield.uid;
                            newFormfieldUser.value = collection[formfielduid];
                            db.Formfield_Users.InsertOnSubmit(newFormfieldUser);
                        }
                        db.SubmitChanges();
                        break;

                    case "radio":
                    case "dropdown":
                        Listitem_User currentListitemUser = db.Listitem_Users.FirstOrDefault(li_u => li_u.fieldlist_uid == formfield.fieldlist_uid && li_u.user_uid == useruid);
                        if (currentListitemUser != null) //delete all currentitems
                        {
                            db.Listitem_Users.DeleteOnSubmit(currentListitemUser);
                        }
                        Listitem_User newListitemUser = new Listitem_User();
                        newListitemUser.listitem_uid = Convert.ToInt32(collection[formfielduid]);
                        newListitemUser.fieldlist_uid = (int)formfield.fieldlist_uid;
                        newListitemUser.user_uid = useruid;
                        db.Listitem_Users.InsertOnSubmit(newListitemUser);
                        db.SubmitChanges();
                        break;

                    case "checkbox":
                        string[] seperatedListitems = collection[formfielduid].Split(',');
                        List<Listitem_User> currentListitems = db.Listitem_Users.Where(li_u => li_u.fieldlist_uid == formfield.fieldlist_uid && li_u.user_uid == useruid).ToList();
                        List<Listitem_User> chosenlistitems = new List<Listitem_User>();
                        if (currentListitems != null)
                        {
                            db.Listitem_Users.DeleteAllOnSubmit(currentListitems);
                        }
                        foreach (string listitem in seperatedListitems)
                        {
                            Listitem_User newcheckboxitemUser = new Listitem_User();
                            newcheckboxitemUser.listitem_uid = Convert.ToInt32(listitem);
                            newcheckboxitemUser.fieldlist_uid = (int)formfield.fieldlist_uid;
                            newcheckboxitemUser.user_uid = useruid;
                            chosenListitems.Add(newcheckboxitemUser);
                        }

                        db.Listitem_Users.InsertAllOnSubmit(chosenListitems);
                        db.SubmitChanges();
                        break;
                }
            }
        }
    }
}
```

Bijlage 29 – Wie is online

Multilogue **Search**

Home Discussion Room Video My Profile Background Help

Welcome back, michel dickhoff

English Dutch French Who's online Logout

Who's online

There are 45 users online

previous 1 2 3 next

nilsss corver
Responses: 3

michel dickhoff
Responses: 79

boudewijn Somer
Responses: 1

tim dirkx
Responses: 0

Olivier Maaaa
Responses: 0

mehmet altay
Responses: 0

nils corv
Responses: 0

timmie timmie
Responses: 0

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2010-1.1
Startdatum: *08 – 02 – 2010*
Einddatum: 09- 07 – 2010
Inleverdatum: 4 juni 2010

Studentnummer: 60092
Achternaam: dhr Dickhoff
Voorletters: M
Roepnaam: Michel
Adres: Lokhorst 14
Postcode: 2402 PN
Woonplaats: Alphen a/d Rijn
Telefoon: – *Mobiel*: 06-25572263

Opleiding: Technische informatica
Locatie: Delft
Variant: voltijd

Naam studieloopbaanbegeleider: Adri Pronk
Naam begeleider/examinator: Tony Andrioli
Naam expert/examinator: Sjaak van Peski

Naam bedrijf: E-mark
Afdeling bedrijf:
Bezoekadres bedrijf: Wilhelminastraat 9
Postcode bezoekadres: 2011 VH
Postbusnummer:
Postcode postbusnummer:
Plaats: Haarlem
Telefoon bedrijf: 023 551 88 99
Telefax bedrijf:
Internetsite bedrijf: www.e-mark.nl

Achternaam opdrachtgever: dhr Vernikov
Voorletters opdrachtgever: A
Titulatuur opdrachtgever:
Functie opdrachtgever: ICT manager
Doorkiesnummer opdrachtgever:
Email opdrachtgever: alex.vernikov@e-mark.com

Achternaam bedrijfsmentor: dhr Somer
Voorletters bedrijfsmentor: B
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Geeft leiding aan programmeurs
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: boudewijn@e-mark.com

Doorkiesnummer afstudeerder:
Bedrijfsemail afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Opdrachtomschrijving

Titel afstudeeropdracht: Herbouw online communicatie applicatie

1. Bedrijf:

   E-Mark is een organisatie dat bestaat uit de onderdelen Mail, Services en Labs. De afdeling Mail houdt zich bezig met e-mail marketing. Zo heeft E-Mark een direct mail module waar grote hoeveelheden mails mee verstuurd kunnen worden.

   De afdeling Services houdt zich bezig met zoekmachine marketing, website ontwikkeling en sms marketing.

   Op de Labs afdeling houdt een team zich bezig met het ontwikkelen van nieuwe creatieve en innovatieve ICT oplossingen.

2. Aanleiding

   E-Mark beschouwt het ontwikkelen van webapplicaties door middel van PHP als verouderd.
   Momenteel heeft E-Mark een online applicatie die verkocht wordt.
   Deze online applicatie, die als doel heeft organisaties vanaf verschillende locaties met elkaar te laten communiceren(discussiëren), is geschreven in PHP. De applicatie wordt opgezet per opdrachtgever.
   Deze online applicatie kan gezien worden als een erg uitgebreid forum. De applicatie is moderated en men kan er alleen gebruik van maken door een uitnodiging te verkrijgen. De applicatie biedt ook meerdere forums in verschillende talen waarbij de taal van de interface los staat van de taal van het forum.

   Enkele features van deze applicatie zijn:
   - Reacties van andere mensen toevoegen aan persoonlijke (reactie)favorietenlijst
   - Biedt kamers waarbij mensen uitgenodigd kunnen worden om te discussiëren
   - Mogelijkheid om polls op te zetten
   - Het hosten van video’s
   - Local configuration (kijken waar mensen zich bevinden en op basis daarvan instellingen aanpassen)

   De moderators beheren de berichten vanuit de backend via een CMS.

   De performance is door het grote aantal gebruikers minder geworden. Momenteel gebruiken zo'n 300 000 mensen dit product. Doordat de performance minder is geworden is het lastiger om nieuwe klanten te werven. Het performance probleem is door E-Mark weliswaar visueel opgemerkt maar nog niet gerapporteerd d.m.v. verslaglegging.

3. Probleemstelling

Door de jaren heen zijn er uitbreidingen aangebracht door verschillende mensen waardoor de code *niet* efficient geschreven is. Dit komt omdat er niet voldoende documentatie was. Ook de klassenstructuur heeft hier onder moeten lijden. Dit betekent dat ook de klassenstructuur veranderd moet worden zodat de performance (snelheid van applicatie) verbeterd word. Daarnaast is ASP.NET een andere taal/technologie dan PHP.
Door bovengenoemde redenen is een één op één overzetting niet mogelijk.

De performance winst die behaald wordt door de applicatie te schrijven naar ASP.NET zit niet aan de techniek/taal ASP.NET. De maatapplicatie is zoals hierboven genoemd inefficiënt gecodeerd, ook de klassenstructuur is verre van optimaal. Gecombineerd met het feit dat E-Mark PHP als veroudert beschouwd maakt de keuze om de maatapplicatie te herbouwen verantwoord.
Als voorbeeld gaf E-Mark dat bij de laatste installatie van de applicatie voor een bedrijf een cluster van 8 servers benodigd was om de applicatie goed te laten functioneren.
Door de hoge kosten hiervan is het gewenst dat de vernieuwde ASP.NET applicatie minder zwaar gaat worden.

Om de performance te verbeteren kan ook de Database die gebruikt wordt voor de applicatie geoptimaliseerd te worden. Denk hierbij aan de verschillende normaalsvormen die overbodige data verminderen. Per forum hoort een database welke 53 tabellen heeft met meerdere Foreign Keys en Primary Keys. Daarnaast is er nog een hoofddatabase die alle andere forumdatabases bevat. In totaal zijn er nu 400.000 entries in de database.
Om de applicatie zo snel mogelijk te laten draaien moet er een goede balans gevonden worden wat betreft caching. Ook database transacties mogen de applicatie niet noemenswaardig vertragen. Het snappen van de verschillende request/respond modellen is noodzakelijk zodat hier aan gemeten kan worden.

Om de performance van de applicatie zoveel mogelijk te doen toenemen moeten de requests en responds naar en van de applicatie in kaart gebracht zodat er conclusies getrokken kunnen worden over de data die verstuurd wordt. Deze modellen worden verwerkt in het technisch ontwerp.

4. Doelstelling van de afstudeeropdracht

De afstudeerder legt de functionaliteit van de oude maatapplicatie vast. Vervolgens wordt er gekeken wat klanten toegevoegd willen zien in de nieuwe applicatie. Deze informatie wordt vastgelegd in een aanbevelingsdocument. De informatie van klanten wordt verkregen met behulp van de commerciële verkoper.

Daarnaast is de afstudeerder verantwoordelijk voor het opleveren van zowel het functioneel en technisch ontwerp van de te maken applicatie. Ook produceert de afstudeerder een deel van de code. Doordat het project te veel werk is om door één persoon te laten realiseren wordt het project uitgevoerd in teamverband.

Het testen van software wordt met behulp van stress testing, unit testing(combinatie van black box en white box) gedaan. Ook gaat er gebruik gemaakt worden van profiling. Van hieruit kan er teruggekoppeld worden naar de code. Om bepaalde handelingen eenvoudiger te maken zal de afstudeerder Auto it gebruiken.

Voor het meten van de performance wordt er voor één of meerdere performance tools gekozen. Welke dit zijn wordt in de afstudeerstage gekozen.

5. Resultaat

    - Requirements voor de te maken applicatie (aanbevelingsdocument)
    - Functioneel en technisch ontwerp
    - Code

6. Uitgangssituatie aan de hand van:

    a. Beschikbare noodzakelijke software (*indien van toepassing*)
    PHP maatapplicatie

    b. Beschikbare noodzakelijke hardware (*indien van toepassing*)
    PC

    c. Reeds opgeleverde relevante documenten

    d. Aanwezige ideeën

7. Werkzaamheden aan de hand van:

    a. Te hanteren methodieken
    De te hanteren methodieken worden tijdens het afstuderen bepaald.

    b. Uit te voeren activiteiten
        - Onderzoek naar huidige maatapplicatie
        - Specificaties van het te realiseren systeem vastleggen
        - Kiezen van geschikte ontwikkelstrategie en ontwikkelmethodiek
        - Functioneel ontwerp omzetten een in technisch ontwerp
        - Coderen
        - Onderzoek Requests/Response huidige applicatie
        - Functioneren in teamverband
        - Testen van systeem

    c. Te gebruiken technieken
    ASP.NET, HTML, CSS, C#, SQL, UML

d. Te vermelden nadrukken
    Functioneel en technisch ontwerp

8. Risico's en maatregelen

9. Op te leveren (tussen)producten

- Analyse document PHP applicatie
- Functioneel ontwerp (eisen voor ASP.NET applicatie)
- Technisch ontwerp (UML modellen)
- Unit tests
- code
- Aanbevelingsdocument voor uitbreiding ASP.NET applicatie
- In kaart brengen van Request Response modellen

Benodigde competenties

- Kiezen van een ontwikkelstrategie en ontwikkelmethodiek (A4)
- Opstellen van systeemeisen (A5)
- Ontwerpen van een technisch informatie systeem (C8)
- Realiseren van software (D16)
- Samenwerken in een project-team (G2)
- Achterhalen van behoeften van belanghebbenden (A3)

Te demonstreren competenties en wijze waarop

- Kiezen van een ontwikkelstrategie en ontwikkelmethodiek (A4)
    - Afstudeerder kiest een gepaste strategie en methodiek en verwerkt dit in de scriptie.

- Opstellen van systeemeisen (A5)
    - Het opstellen van functioneel ontwerp geeft inzicht in de systeemeisen van de te realiseren applicatie.

- Ontwerpen van een technisch informatie systeem (C8)
    - Technisch ontwerp realiseren en beschreven in formele taal. Daarnaast moet er rekening gehouden worden met de ontwikkelstrategie.

- Realiseren van software (D16)
    - Coderen in ASP.NET m.b.v. C#. Verder sluit broncode aan bij het ontwerp en is syntactisch correct. Daarnaast moet er efficient geprogrammeerd worden en een leesbare layout gekozen zijn.

- Samenwerken in een project-team (G2)
    - Goed communiceren binnen het project met andere projectleden, projectleden op de hoogte houden van voortgang en het nakomen van afspraken.

- Achterhalen van behoeften van belanghebbenden (A3)
    - Belanghebben in kaart gebracht, belangen van belanghebbenden zijn verkregen en opgesteld.

Persoonlijke verantwoording van keuze opdracht/bedrijf/competenties

Tijdens de stage heeft de afstudeerder een PHP opdracht gedaan. Het ontwikkelen van applicaties op het web beviel erg goed. Met de nieuwere ASP.NET technologie wil de afstudeerder kennis maken. Deze opdracht biedt daar de gelegenheid voor en legt ook een behoorlijke focus op het analyse en ontwikkeltraject (technisch ontwerp) wat de afstudeer aantrekt.

Op eerste zicht leek het bedrijf informeel en niet al te groot. De afstudeerder heeft dit als een pluspunt ervaren.

De competenties zijn gekozen aan de hand van de opdracht en in het kader van de opleiding technische informatica.

Nominale duur afstudeerperiode
Plusminus 20 weken

Activiteitenplanning

1-3e week: Analyse maatapplicatie, extra functionaliteiten verkrijgen (aanbevelingsdocument), opzetten van functioneel ontwerp.

4-7e week: Request/respond modellen in kaart brengen, eerste technisch ontwerp bouwen.

8-14e week: Coderen en verifieren van ontwerp (dus terugkoppelen)

15e-17e week: Testen van beschikbare code

Naam opdrachtgever:
Handtekening voor akkoord:
Datum:

Naam bedrijfsmentor:
Handtekening voor akkoord
Datum:

Naam begeleider/examinator:
Handtekening voor akkoord:
Datum:

Naam expert/examinator:
Handtekening voor akkoord:
Datum:

Plan van aanpak

8 februari 2010

Versie 3

Michel Dickhoff

Samenvatting

Het bedrijf waar de afstudeerder gaat werken is E-mark. E-Mark is een organisatie dat bestaat uit de onderdelen Mail, Services en Labs. E-mark verkoopt het product Multilogue. Momenteel is Multilogue een PHP maatapplicatie waarbij de performance door de jaren wat minder is geworden door uitbreidingen. Hierdoor is de code minder efficiënt geworden en is de klassenstructuur niet meer optimaal.

Door bovengenoemde redenen en het feit dat E-mark meer ASP.NET applicaties wil gaan opnemen als core products maakt één op één overzetting niet mogelijk. De opdracht is de maatapplicatie omzetten naar een basis ASP.NET applicatie waarbij de klassenstructuur en code wel goed gedefinieerd is. De snelheid van de nieuwe applicatie ook minder performance problemen te hebben dan de maatapplicatie.

Voor het ontwikkelen van de ASP.NET applicatie wordt RUP toegepast. Dit omdat RUP gebaseerd is op best practices en een iteratieve/incrementele ontwikkelstrategie ondersteunt. De applicatie wordt geprogrammeerd met behulp van C#.

De afstudeerder gaat in teamverband werken, maar de afstudeerder maakt het analysedocument, functioneel ontwerp en het technisch ontwerp zelf.
Wat betreft testen worden er eenheidstesten, unit tests en systeem tests gedaan.

De risicofactoren die het slagen van het project in de weg kunnen zitten bestaan uit interne en externe risicofactoren. Bij interne risicofactoren kan gedacht worden aan gegevensverlies of langdurige afwezigheid van de afstudeerder door bijvoorbeeld ziekte.
Externe risicofactoren kunnen opgevat worden als het langdurig afwezig zijn van de bedrijfsmentor.

Inhoudsopgave

Versiebeheer

Versie 1 van het plan van aanpak is nagekeken door Boudewijn Somer en Nils Corver op 10 februari 2010.
De volgende punten zijn veranderd:

- Hoofdstukverwijzingen in inleidingen van hoofdstukken zijn weggehaald voor het leesgemak.
- De delen waarbij "PHP verouderd" werd vermeld is meer genuanceerd geschreven. Dit omdat er nog een aantal programmeurs zich binnen E-Mark bevinden die PHP programmeren.
- Installatiehandleiding aan de op te leveren producten toegevoegd.
- Projectgrenzen en randvoorwaarden herschreven.
- Namen correct gespeld in hoofdstuk 4.1
- Projectbeheersing toegevoegd in hoofdstuk 3.2
- Korte beschrijving gegeven van black box en white box in hoofdstuk 2.3

Nagekeken door Tony Andrioli op 17 februari 2010. De volgende punten zijn veranderd:

- Toevoegingen nader toegelicht in hoofdstuk 2.3
- Externe risico toegevoegd (ASP.NET levert niet gewenste performance)
- Meer Methodieken onderzocht en beschreven in hoofdstuk 3
- Rollen projectleden beschreven in hoofdstuk 4
- Prince2 argumentatie aangepast

Inleiding

Dit plan van aanpak heeft tot doel inzicht te geven in het proces van het herbouwen van de Multilogue. Er wordt informatie gegeven over de aanleiding van het project. Op deze manier wordt het mogelijk voor buitenstaanders om zich een beeld te vormen van het project(***hoofdstuk* 1**).

Vervolgens worden de opdrachtgever en opdrachtnemer vermeldt. Als deze gegevens bekend zijn, wordt het probleem beschreven dat opgelost dient te worden. Ook wordt er een omschrijving van de doelstelling van de opdracht gegeven. Hierbij worden de grenzen die gesteld zijn aan de doelstelling van de opdracht en de eisen van het eindresultaat beschreven (***hoofdstuk 2***).

Om overeenstemming te krijgen over de te volgen weg naar het gewenste eindresultaat moet de gekozen ontwikkelstrategie/methodiek aan bod komen. Om een idee te krijgen welke activiteiten wanneer plaatsvinden wordt er een globale planning gemaakt. (***hoofdstuk 3***).

Het inrichten van het project gebeurt door middel van het beschrijven van de rollen die er binnen het project zijn en door wie deze vervuld worden. De hulpmiddelen die nodig zijn voor het uitvoeren van het project worden beschreven. Vervolgens kunnen de kwaliteitseisen beschreven worden waaraan het eindproduct en het tussenproduct moeten voldoen en de manier waarop dit bereikt wordt (***hoofdstuk 4***).

Er wordt een beeld gevormd van de mogelijke factoren die het slagen in de weg kunnen staan. Tevens moet vermeld worden welke maatregelen hiertegen genomen worden. Deze factoren hebben betrekking op de opdracht, aanpak en projectinrichting (***hoofdstuk 5***).

1 Achtergrond

Hoofdstuk 1, “Achtergrond”, beschrijft het bedrijf E-mark waar de afstudeerder gaat werken evenals de aanleiding van de opdracht**.**

1.1 E-mark

E-Mark is een organisatie dat bestaat uit de onderdelen Mail, Services en Labs. De afdeling Mail houdt zich bezig met e-mail marketing. Zo heeft E-Mark een direct mail module waar grote hoeveelheden mails mee verstuurd kunnen worden.

De afdeling Services houdt zich bezig met zoekmachine marketing, website ontwikkeling en sms marketing. Op de Labs afdeling houdt een team zich bezig met het ontwikkelen van nieuwe creatieve en innovatieve ICT oplossingen.

1.2 Aanleiding van opdracht

E-mark heeft een product ontwikkelt genaamd Mulitlogue. E-mark besteed de verkoop van dit product uit. Mutlilogue is een product dat gezien kan worden als een groot forum. Dit forum is moderated, dat betekent dat gedurende een aantal dagen de reacties op onderwerpen ge-edit kunnen worden door derden.

Enkele features van de multilogue zijn:

- Reacties van andere mensen toevoegen aan persoonlijke (reactie)favorietenlijst
- Biedt kamers waarbij mensen uitgenodigd kunnen worden om te discussiëren
- Mogelijkheid om polls op te zetten
- Het hosten van video’s
- Local configuration (kijken waar mensen zich bevinden en op basis daarvan instellingen aanpassen)

Multilogue bestaat uit een frontend en twee backends. Zo is er een backend voor editors, de moderators bekijken of de reacties van mensen niet uit de land lopen. De editors kunnen ook nieuwe discussie onderwerpen aanmaken. De andere backend is voor administrators. De administrator is iemand van E-mark die de applicatie opzet voor een bedrijf.

Nu is het zo dat de applicatie die nu geschreven is in PHP door de jaren heen veelvoudig verandert is waardoor de applicatie niet optimaal meer presteert. Door de vele gebruikers zijn er veel servers nodig wat de kosten omhoog duwt. Ook is het moeilijk om de applicatie uit te breiden omdat de klassenstructuur en code niet meer optimaal is. Daarbij wilt E-mark ook meer applicaties gaan programmeren met ASP.NET.

2 Opdrachtomschrijving

Hoofdstuk 2, “Opdrachtomschrijving” geeft inzicht in de uit te voeren opdracht van de afstudeerder. Zo komen de opdrachtgever en opdrachtnemer aan bod.
De probleemstelling en opdrachtbeschrijving met bijbehorend doel worden uitgelegd. De afstudeerder levert producten op naarmate het project vordert. Om verwarring te voorkomen worden de projectgrenzen en randvoorwaarden beschreven.

2.1 Opdrachtgever en opdrachtnemer

De opdrachtgever is in feite E-mark zelf. Alex Vernikov vertaalt de rol van de opdrachtgever.
De opdrachtnemer is de afstudeerder Michel Dickhoff.

2.2 Probleemstelling

De aanleiding van het project is te lezen in hoofdstuk 1.2.

De performance winst die behaald wordt door de applicatie te schrijven naar ASP.NET zit niet in de techniek/taal ASP.NET. De maatapplicatie is zoals beschreven in hoofdstuk 1.2 inefficiënt gecodeerd, ook de klassenstructuur is verre van optimaal. Gecombineerd met het feit dat E-Mark meer ASP.NET applicaties wil ontwikkelen maakt de keuze om de maatapplicatie te herbouwen verantwoord en is één op één zetting niet mogelijk.
Als voorbeeld gaf E-Mark dat bij de laatste installatie van de applicatie voor een bedrijf een cluster van 8 servers benodigd was om de applicatie goed te laten functioneren.
Door de hoge kosten hiervan is het gewenst dat de vernieuwde ASP.NET applicatie minder zwaar gaat worden.

Om de performance te verbeteren kan ook de Database die gebruikt wordt voor de applicatie geoptimaliseerd/herbouwd worden. Denk hierbij aan de verschillende normaalsvormen die overbodige data verminderen. Per forum hoort een database die ongeveer 70 tabellen heeft met meerdere Foreign Keys en Primary Keys. Daarnaast is er nog een hoofddatabase die alle andere forumdatabases bevat. In totaal zijn er nu 400.000 entries in de database.

2.3 Doelstelling en opdrachtbeschrijving

De afstudeerder legt de functionaliteit van de oude maatapplicatie vast. Nadat de basisfunctionaliteit is gerealiseerd met ASP.NET wordt er gekeken welke toevoegingen erbij kunnen komen. Bij toevoegingen moet gedacht worden aan functionaliteit die ten gunste komt van de klant. Dit zijn functionaliteitwensen van klanten die zij graag toegevoegd willen hebben in de Multilogue. Hierbij moet wel in de gaten gehouden worden dat de performance door deze uitbreidingen niet achteruit ernstig achteruit gaat.
Deze informatie wordt vastgelegd in een aanbevelingsdocument. De informatie van klanten wordt verkregen met behulp van de commerciële verkoper.

Daarnaast is de afstudeerder verantwoordelijk voor het opleveren van zowel het functioneel en technisch ontwerp van de te maken applicatie. Ook produceert de afstudeerder een deel van de code. Doordat het project te veel werk is om door één persoon te laten realiseren wordt het

project uitgevoerd in teamverband. Het analysedocument, het functioneel ontwerp en het technisch ontwerp produceert de afstudeerder zelf.

Het testen van software wordt met behulp van stress testing, unit testing(combinatie van black box en white box) gedaan. Bij black box worden gegevens in het systeem ingevoerd waarbij gekeken wordt hoe het systeem reageert. De verwachte uitkomst wordt van tevoren vastgelegd. Bij white box testing wordt er naar de code gekeken en worden paden gevolgd van variabelen.

Ook gaat er gebruik gemaakt worden van profiling met als doel de code zoveel mogelijk te optimaliseren. Van hieruit kan er teruggekoppeld worden naar de code.

Het doel van E-mark is om een nieuwe applicatie te ontwikkelen in ASP.NET waarbij de code en klassenstructuur wel optimaal geschreven is en tevens sneller is dan de PHP maatapplicatie.

2.4 Op te leveren producten

De producten die opgeleverd worden zijn:

- Analyse document PHP applicatie
- Functioneel ontwerp (eisen voor ASP.NET applicatie)
- Technisch ontwerp (UML modellen)
- Unit tests
- code
- Installatiehandleiding
- Aanbevelingsdocument voor uitbreiding ASP.NET applicatie

2.5 Projectgrenzen en randvoorwaarden

Voor het slagen van het project is het belangrijk om het project af te bakenen.
Hierbij komen twee dingen aan de orde, namelijk hoe ver het project gaat en hoe breed het project is.
Bij het herbouwen van de Multilogue zijn de volgende data vastgelegd:

- Start: 8 februari 2010
- Eind: 4 juni 2010
- Afstudeerder besteed 40 uur per week aan het project

De deadlines zijn die nu vast staan zijn:

- 23 februari 2010 – Analyse document afgerond
- 10 maart 2010 – functioneel ontwerp document afgerond
- 14 april 2010 – technisch ontwerp document afgerond
- 4 juni 2010 – afstudeerverslag af

3 Aanpak

Hoofdstuk 3, “aanpak” beschrijft een globale aanpak van de opdracht die de afstudeer gaat toepassen. De methoden en technieken worden beschreven. De werkzaamheden van de afstudeerder worden in kaart gebracht.
De standaarden die de afstudeerder toepast worden beschreven. Dit hoofdstuk geeft ook een globale planning van de te nemen stappen.

3.1 Methoden en technieken

Er wordt gebruik gemaakt van UML om diagrammen te realiseren. Deze techniek is gekozen omdat de afstudeerder deze techniek beheerst.De MoSCoW methode wordt gebruikt voor het definiëren van de prioriteiten nadat het functioneel ontwerp afgerond is.

De afstudeerder heeft de watervalmethode niet gekozen als methodiek omdat de Multilogue een te groot project is om het project via de watervalmethode te voltooien. Hierbij is het vergeten van eisen een grote aanwezige factor.

Naast de watervalmethode kan er nog gedacht worden aan het werken met Prototypen. Bij de prototypemethodologie wordt de analyse, ontwerp en implementatie tegelijk uitgevoerd. Het voordeel van deze methodiek is dat gebruikers snel een idee krijgen hoe het product eruit komt te zien. Deze prototypen worden “quick & dirty” geschreven. In dit geval past zo’n methodiek niet bij de opdracht. Namelijk de gebruikers weten wat ze kunnen met de Multilogue en het tegelijkertijd uitvoeren van de analyse, ontwerp en implementatie werkt hier niet gunstig. Het doel is niet om het product zo snel mogelijk te demonstreren en slechte code op te leveren maar juist de performance te verbeteren door middel van de code optimaal te schrijven.

Ook parallelle ontwikkeling biedt geen oplossing aangezien deze methode het ontwerp splitst in een aantal subontwerpen. De afstudeerder is verantwoordelijk voor het gehele ontwerp. Ook is deze ontwikkelmethode minder geschikt omdat onderdelen zelden onafhankelijk van elkaar zijn en er zo mogelijk vertraging komt. Bovendien is integratie van de subprojecten lastig.

De afstudeerder kiest voor iteratief/incrementeel ontwikkelen. Hierbij wordt het systeem in deeltjes ontworpen wat het ontwikkelproces eenvoudiger maakt. Als ontwikkelmethodiek wordt RUP gebruikt. RUP is gebaseerd op een aantal best practices zoals het iteratief/incrementeel ontwikkelen van software. De reden dat RUP gekozen is, is omdat RUP bewezen werkt.

RUP heeft een viertal fases. Dit zijn inceptiefase, elaboratiefase, constructiefase en transitiefase. Het doel van de inceptiefase is het geven van een go/no go voor het project. In

dit geval is deze fase niet van toepassing aangezien de ASP.NET applicatie er moet komen, dus de go is al gegeven.

De elaboratiefase levert het analysedocument, functioneel document en het technisch ontwerp op. Daarna wordt de software gebouwd die in de elaboratiefase is gedefinieerd, dit is de constructiefase. De afstudeerder beperkt de transitiefase tot het tonen van het product waarbij feedback gegeven kan worden.

Bij RUP horen een aantal disciplines die aan bod komen gedurende het project. Deze zijn verdeeld in twee hoofdgroepen, namelijk ontwikkeling en ondersteuning. Dit zijn:

1) Business Modeling
2) Achterhalen van requirements
3) Analyse en ontwerp
4) Implementatie
5) Testen
6) Deployment
7) Projectbeheer
8) Omgeving
9) Configuration management
10) Change management

ad1 Business Modeling (ontwikkeling)

Deze discipline laat de afstudeerder achterwege omdat het niet van belang is voor het project om de communicatie te verbeteren tussen IT en business in E-mark. Businessdoelen van E-mark zijn het behalen van meer klanten en geld door verkoop van de Multilogue.

ad2 Achterhalen van requirements (ontwikkeling)

Deze discipline wordt wel aandacht aan besteed door de afstudeerder, wat hieruit voort komt zijn use cases, aanvullende functionele en technische specificaties. Deze discipline start in de elaboratiefase en wordt afgerond in de constructiefase.

ad3 Analyse en ontwerp (ontwikkeling)

Deze discipline wordt benaderd in de elaboratiefase en loopt door in de constructiefase. Bij het beëindigen van deze discipline is er een technisch ontwerp ontwikkeld die voldoet aan de specificaties die opgelegd zijn in ad2.

ad4 Implementatie (ontwikkeling)

Implementeren van subsystemen en het testen van de ontwikkelde componenten in de constructiefase.

ad5 Testen (ontwikkeling)

De tests vinden plaats in verschillende fasen van het project zodat fouten in een vroeg stadium ontdekt worden. Meeste tests vinden plaats in de constructiefase.

ad6 Deployment (ondersteuning)

Deze discipline laat de afstudeerder achterwege omdat het niet de taak is van de afstudeerder om het product te installeren hulp bieden aan gebruikers van het systeem.

ad7 Projectbeheer (ondersteuning)

Het beheren van het project gebeurt op twee niveaus. Een globale planning en een planning van de specifieke iteratie.

ad8 Omgeving (ondersteuning)

De omgeving is de variabele die het plan die uitgestippeld was kan verstoren. De mensen die hier betrekking tot hebben beperkt zich tot de opdrachtgever en bedrijfsmentor en het projectteam.

ad9 Configuration management (ondersteuning)

Alle gemaakte onderdelen van een project, zoals documenten en modellen, moeten met behulp van versiebeheer onderhouden worden. Dit doet de afstudeerder door middel van SVN.

ad10 Change management (ondersteuning)

Verzoeken voor veranderingen die betrekking hebben op stukken die de afstudeerder geleverd heeft worden verwerkt door de afstudeerder zelf.

3.2 Projectbeheersing

Voor projectbeheersing kan er gedacht worden aan het gebruik van een andere methodiek. PRINCE2 richt zich totaal op het managen van een project. Doordat PRINCE2 erg document-driven is kan PRINCE2 niet in zijn geheel gebruikt worden met alle bijbehorende documentaties. Hiervoor is de tijd van 17 weken te kort. De afstudeerder wilt zich ook meer richten op het ontwikkelen dan het managen van het project.

PRINCE2 in zijn compleetheid bestaat uit:

1) Starting up a project
2) Initiating a project
3) Directing a project
4) Planning
5) Controlling a stage
6) Managing stage boundaries

7) Managing product delivery
8) Closing a project

Er is gekozen om een aantal elementen van PRINCE2 eruit te pakken en deze globaal toe te passen zonder PRINCE2 te gebruiken.

“Starting up a project” bekijkt het nut van het starten van een project. Dat de Multilogue opnieuw gebouwd moet worden is vastgesteld vanuit E-mark. Hierdoor is deze fase niet benodigd.
De stappen “Initiating a project” en “Closing a project” zijn verplicht bij het gebruik van PRINCE2. In deze fase worden resultaten, planning en verantwoordelijkheden vastgelegd. De fase “Initiating a project” is verwezenlijkt door het schrijven van het plan van aanpak.
De stap “Close a project” wordt opgevangen door het beschrijven van de elementen die gerealiseerd zijn plus de te volgen stappen. Dit gaat beschreven worden in het technisch ontwerp.

Planning, risico’s en kwaliteitsbeheersing wordt opgevangen door het plannen via de RUP discipline projectbeheer, ofwel plannen op twee niveaus (globaal en per iteratie). Kwaliteitsbeheersing wordt gerealiseerd door terugkoppeling naar het functioneel ontwerp/technisch ontwerp. Dat betekent “kan het systeem wat er beschreven is en is dit op de manier geïmplementeerd zoals beschreven is?”. De documenten worden gecontroleerd door Nils Corver en Boudewijn Somer.

De afstudeerder werkt grotendeels alleen aan het project. Taakverdelingen komen er niet aan te pas. Van een managementgroep/stuurgroep is geen sprake. Hierdoor is de afstudeerder van mening dat PRINCE2 een overhead betreft tijd creëert.

3.3 Werkzaamheden

De werkzaamheden die nu gepland zijn:

- Onderzoek naar huidige maatapplicatie
- Specificaties van het te realiseren systeem vastleggen
- Kiezen van geschikte ontwikkelstrategie en ontwikkelmethodiek
- Functioneel ontwerp omzetten een in technisch ontwerp
- Coderen
- Functioneren in teamverband
- Testen van systeem

3.4 Standaarden

Er gaat gebruik gemaakt worden van microsoft visual studio 2008. ASP.NET in combinatie met C# wordt gebruikt om de software te realiseren.

3.5 Planning

De globale planning die relevant is aan het project:

Datum	**Activiteit/ stand van zaken**
8 februari / 9 februari	PVA maken
10 februari	Start analyse
17 februari	Globale analyse document af (eerste versie)
18 februari	Feedback op analyse document
23 februari	Globale analyse document afgerond
24 februari	Start functioneel ontwerp
3 maart	Document functioneel ontwerp af (eerste versie)
4 maart	Feeback op functioneel ontwerp
10 maart	Functioneel ontwerp afgerond
10 maart tot en met 17 maart	Uitloop functioneel ontwerp
17 maart	Start technisch ontwerp
31 maart	Document technisch ontwerp af (eerste versie)
5 april	Feedback technisch ontwerp
14 april	Technisch ontwerp afgerond
14 april tot en met 24 april	Uitloopt technisch ontwerp
24 april - verder	Programmeren in iteraties met terugkoppeling naar ontwerp en meer analyse
4 juni	Afstudeerverslag inleveren

Alle data vinden plaats in 2010. Na het functioneel ontwerp wordt er een meer gedetailleerde planning gegeven voor het technisch ontwerp en het programmeren. Nadat de globale analyse klaar is komt het functioneel ontwerp tot stand waarbij de functionaliteit beschreven is die in de te programmeren applicatie komt. Daarna wordt er een meer gedetailleerde planning gemaakt waarbij de iteraties beschreven worden.

Andere belangrijke data vanuit school zijn:

Datum	Activiteit
22 februari tot en met 5 maart	Bedrijfsbezoek schoolbegeleider
22 maart tot en met 2 april	Tussentijds verslag inleveren
5 april tot en met 16 april	Bespreken concept afstudeerdossier
10 tot en met 21 mei	Tussentijds assesment

4 Projectinrichting

Hoofdstuk 4, "Projectinrichting", beschrijft de indeling van het projectteam.
Daarnaast beschrijft het hoofdstuk hoe de afstudeerder aan de informatie komt die benodigd is voor het project. De eisen die later vastgesteld worden dienen daadwerkelijk ook nageleefd te worden. Verder beschrijft dit hoofdstuk de kwaliteitsborg.

4.1 Projectindeling

Het projectteam bestaat uit:

Naam	Rol
Mehmet	Ondersteuning voor werkwijze Multilogue
Nils	Ondersteuning documentatie/ontwerp/programmeren
Tim	Ondersteuning design/programmeren
Petra	Ondersteuning design
Michel	Maken van fo, to, basiscode
Boudewijn	Ondersteuning documentatie/ontwerp/programmeren

Afstudeerder Michel richt zich op de elaboratiefase en constructiefase en levert het analysedocument, functioneel ontwerp en het technisch ontwerp op. Daarnaast zorgt Michel voor een deel van de code. Deze code wordt tevens getest op zowel robuustheid en performance in vergelijking met de bestaande Multilogue.

Bovengenoemde namen zijn alle programmeurs met als uitzondering Petra. De projectleden ondersteunen de afstudeerder in de contructiefase.

4.2 Informatievoorziening

De afstudeerder is gedurende de hele week aanwezig op de locatie van E-mark.
Hierdoor is de informatie die benodigd is eenvoudig te verkrijgen door middel van het vragen van de desbetreffende personen. Mocht de persoon die benodigd is er niet zijn kan er door middel van mail informatie verkregen worden.

4.3 Kwaliteitsborg

De afstudeerder dient de specifieke eisen nog vast te leggen. Dit gebeurd in het “Functioneel ontwerp”. Het functioneel ontwerp volgt uit het analysedocument en wordt nagekeken door Nils Corver en Boudewijn Somer. Op deze manier wordt de kwaliteit gewaarborgd. Wel kan vermeld worden hoe de afstudeerder deze eisen die vastgelegd zijn voor het product ook daadwerkelijk handhaaft.

De afstudeerder heeft een aantal tests voor ogen:

1) Eenheidstest
2) Unit test
3) Systeemtest

ad1 Eenheidstest

De eenheidstests zijn gericht op een eenheid van een programma met een specifieke functie die kan worden getest.

De eenheidstests vinden plaats tijdens het schrijven van de code door de programmeur zelf, die verantwoordelijk was voor dit gedeelte. Voordat het samenvoegen van start kan gaan moeten de aparte fases nogmaals getest worden door andere programmeurs.

ad2 Unit test

Deze test wordt uitgevoerd nadat alle functies hun individuele eenheidstests hebben doorstaan. Bij de integratietest ligt dus de nadruk op de verbinding tussen de afzonderlijke functies en hetgeen dat de gezamenlijke functies moeten doen. Hierbij wordt ook performance getest.

ad3 Systeemtest

Na het voltooien van de eenheidtest en de unit test wordt de systeemtest uitgevoerd, deze dient ter controle of alle modules en programma’s zonder problemen samenwerken. Deze systeemtest lijkt op de integratietest, maar heeft een veel breder bereik aangezien alle modules hierbij betrokken zijn.

5 Risicofactoren

Hoofdstuk 5, “Risicofactoren”, geeft een zo’n compleet mogelijke lijst van risicofactoren, met de bijbehorende maatregelen. De risicofactoren zijn gekozen op basis van “waarde van waarschijnlijkheid en impact”. Dit betekent dat risicofactoren die een kleine impact hebben op het niet slagen van het project of factoren die nauwelijks realistisch te noemen zijn, achterwege gelaten worden.

Risicofactoren die te algemeen zijn zoals ziekte, afwezigheid begeleider en verlies van gegevens zijn niet opgenomen.

Risicofactoren die tot dusverre bekent zijn:

1) ASP.NET in combinatie met C# levert niet de gewenste performance

ad1 ASP.NET in combinatie met C# levert niet de gewenste performance

Een belangrijk risico is dat de herbouwde applicatie toch niet de gewenste performance haalt in vergelijking met de PHP maatapplicatie. Dit risico is getackeld doordat E-mark een test gedraaid heeft in ASP.NET met een niet genormaliseerde database. Hierbij zijn vergelijkbare functies tegenover elkaar gezet. Hieruit bleek dat de ASP.NET functies sneller waren dan de PHP functies.

Daarentegen wordt met het realiseren van de applicatie uitbreidbaarheid gegarandeerd.

Analysedocument

15 februari 2010

Versie 3

Michel Dickhoff

Samenvatting

Dit document is geschreven zodat derden inzicht krijgen wat de Multilogue is en hoe de Multilogue presteert.

De Multilogue is een maatapplicatie die het mogelijk maakt om gebruikers te laten reageren op vragen in kamers welke gesteld zijn door het bedrijf (de kopende partij). Dit platform is ontwikkeld zodat werknemers in het bedrijf die zich in verschillende tijdzones verkeren met elkaar kunnen discussiëren. De Multilogue geeft een gestructureerde weergave van reacties die geplaatst zijn door mensen (wat bijvoorbeeld bij mail niet mogelijk is na honderden replies).
De Multilogue verschilt van een normaal internetforum in het bekijken van video's, reacties/vragen toevoegen aan favorietenlijst, reacties categoriseren/waarderen en de Multilanguage die doorgevoerd is. Ook is er in de Multilogue een duidelijkere scheiding tussen verschillende typen gebruikers (front-end user kan bijvoorbeeld geen vraag starten).

Er zijn vier type gebruikers te definiëren in de Multilogue. Dit zijn de "front-end" gebruiker, moderator, editor en administrator.
De belangrijkste functionaliteit voor de "front-end" gebruiker bestaat uit het bezichtigen van discussiekamers en daar reacties op vragen te plaatsen. Het plaatsen van reacties door de front-end user is eenrichtingsverkeer. Daarnaast heeft de "front-end" gebruiker de mogelijkheid om zijn profiel te zien en deze te wijzigen.
De moderator heeft de taak om gesprekken in goede banen te leiden en om reacties te categoriseren/waarderen.

De editor maakt discussiekamers aan met daarbij de vragen. Daarnaast kan de editor polls opstellen en statistieken opvragen.
Een administrator kan enkel inloggen in de "back-end" die voor de administrator bedoeld is. De administrator zet de Multilogue op voor een client. Een administrator is altijd iemand die werkzaam is bij E-mark.

Voor het testen van de snelheid van de Multilogue zijn eerst de langzamere pagina's gekozen door gebruik te maken van Firebug. Daarna is de code bekeken van deze pagina's en wordt er door middel van profileren bepaald welke delen op de pagina meer tijd in beslag nemen. Als profiler is er gekozen voor Xdebug.

Inhoudsopgave

Versiebeheer

Versie 1 van het analysedocument is nagekeken door Nils Corver op 12 februari 2010. De volgende punten zijn gewijzigd:

- Doel van document toegevoegd aan samenvatting
- Uitgelegd waarom E-mark Mail wordt gebruikt in hoofdstuk 1.4
- Engelse benamingen naar Nederlands vertaald
- Bijlage V juist uigelijnd
- APC toegevoegd aan conclusie
- Globaal aantal zinnen herschreven

Inleiding

Het analysedocument beschrijft globaal wat de mogelijkheden van de Multilogue zijn. Het doel van het document is een beeld te schetsen voor derden wat de Multilogue is. Om dit te realiseren wordt er per type gebruiker een uitleg gegeven wat de mogelijkheden zijn. Dit gebeurd in grote lijnen, details worden er uitgelaten.

Hoofdstuk 1 beschrijft dit door middel van tekst. ***Bijlages I*** t/m ***IV*** geven dit grafisch weer door middel van UML(Unified Modeling Language, manier van het systeem weergeven) diagrammen. De UML diagrammen geven het systeem weer als een zwarte doos waarmee de gebruiker de gewenste activiteiten verricht.

Hoofdstuk 2 geeft de huidige snelheid weer van de Multilogue. Bij snelheid moet gedacht worden aan het laden van pagina's. Bepaalde pagina's worden uitgekozen die langzamer zijn dan het gemiddelde. Hierbij worden functies getest en de snelheid hiervan. Ook de technieken van het meten worden uitgelegd.

1 Multilogue functionaliteit

Hoofdstuk 1, "Multilogue functionaliteit", beschrijft de huidige maatapplicatie(Multilogue) in de zin van wat er momenteel mogelijk is.
De belangrijkste functionaliteit van de Multilogue is het opzetten van discussie kamers. In een discussie kamer kunnen er meerdere vragen gesteld worden waarbij mensen op kunnen reageren. Er wordt onderscheid gemaakt in actieve en non actieve discussie kamers.

In Multilogue wordt er onderscheid gemaakt in vier type gebruikers. Dit zijn de:

1) "front-end" gebruiker
2) Moderator
3) Editor
4) Administrator

Per type gebruiker worden de mogelijkheden vastgelegd. Grafisch is dit weergegeven in UML use case diagrammen in de bijlagen. De use-case diagrammen geven alleen de hoofddoelen weer van de typen gebruikers zodat de diagrammen duidelijk blijven.

1.1 "front-end" gebruiker

Het hoofddoel van dit type gebruiker is het plaatsen/lezen van reacties en het bijhouden van het profiel.

Een "front-end" gebruiker wordt aangemaakt door een editor. De "front-end" gebruiker dient aanvullende informatie te geven via een registratieformulier. Deze gebruiker ziet enkel de actieve discussiekamers.

De "front-end" gebruiker kan op het gebied van reacties, reacties plaatsen en reacties rapporteren (in geval van misbruik). Daarbij is het mogelijk om een vraag of reacties toe te voegen aan een persoonlijke favorietenlijst.
Video's kunnen door de "front-end" gebruiker bekeken worden die door de editors goedgekeurd zijn. Gebruikers kunnen via een andere applicatie (.NET applicatie) video's insturen.
De applicatie maakt gebruik van een Flash Media Server om de video's op te nemen. Deze .NET applicatie stuurt de video vervolgens door naar de PHP maatapplicatie waarbij de video pas te zien is als de video goedgekeurd is door een editor. Verder is het mogelijk om het profiel te bekijken en te wijzigen plus het bekijken van profielen van mensen die online zijn.

Multilogue geeft ook de mogelijkheid aan "front-end" gebruikers om mee te stemmen in polls.

Grafische weergave is weergegeven in **bijlage I**.

1.2 Moderator

De moderator heeft de taak om gesprekken in goede banen te leiden. Dat betekent dat de moderator ervoor moet zorgen dat de reacties niet afwijken van het onderwerp. Dit kan de moderator doen door zelf reacties te plaatsen. Naast het categoriseren en waarderen van reacties heeft deze moderator geen extra functionaliteit ten opzichte van de front-end user.

Het diagram is weergegeven in ***bijlage II***.

1.3 Editor

In het kort is een editor het type gebruiker dat in de "back-end" van de Multilogue discussie kamers, polls en gebruikers aanmaakt (kan zowel met moderator of editor rechten). Een editor kan gezien worden als een moderator++. De editor heeft de mogelijkheid om in de "back-end" en "front-end" inloggen.

Het toevoegen van gebruikers kan op twee verschillende manieren, namelijk met een csv lijst of per persoon. Daarnaast bestaat de mogelijkheid om een gebruiker te zoeken op verschillende criteria en de mogelijkheid om een gebruiker te volgen.

De editor kan zowel generale polls als polls bij vragen opzetten. Een generale poll komt op pagina's terecht in een zijbalk. Een poll bij een vraag is gekoppeld aan de vraag in de discussie kamer. Naast het toevoegen van polls kan de editor content veranderen van pagina's, zijbalken wijzigen/toevoegen, velden toevoegen aan registratieformulieren en het wijzigen van de bloklijst voor ongepaste woorden.

De statistieken worden weergegeven in de vorm van generale informatie (denk hierbij aan aantal vragen, discussie kamers, meest actieve gebruikers…) Daarnaast toont het de statistieken van polls en waarderingen. De rating wordt bijvoorbeeld gedefinieerd als positief of negatief.

De editor heeft ook de mogelijkheid om bijlagen toe te voegen. De bijlagen kunnen files, images en video's zijn.

Het UML diagram is weergegeven in ***bijlage III***.

1.4 Administrator

De administrator zet een Multilogue op voor een bedrijf. De administrator komt bij E-mark vandaan. De administrator heeft de optie om een klant(let op een Multilogue klant, geen "front-end" gebruiker) toe te voegen. Dus een nieuw bedrijf waar de Multilogue voor geïnstalleerd moet worden. Ook is het mogelijk een overzicht te generen van de bestaande klanten.

Nadat er een nieuwe klant is aangemaakt kan een nieuwe Multilogue te creëren. Hier wordt de klant gekozen, de naam van de Multilogue, de start- en einddatum wanneer de Multilogue actief moet zijn en de taal. Nadat er een Multilogue gecreëerd is kan de administrator gebruikers aanmaken voor deze Multilogue met de keuze of het account een editor wordt of niet. Bij het maken van de nieuwe Multilogue moet de naam van de database opgegeven en de manier van inloggen gekozen(email of gebruikername). Daarnaast is de E-Mark Mail module gekoppeld door middel van een gebruikersnaam zodat die applicatie mail kan sturen naar gebruikers van het desbetreffende bedrijf. De redenen dat er een externe applicatie(E-Mark mail) gebruikt wordt zijn onder andere:

- Garantie dat mail aankomt
- Sjablonen zijn aanwezig in E-Mark mail
- Geschikt voor grote hoeveelheden mail te versturen
- "Datamining"

Een administrator kan ook andere administrators toevoegen en het overzicht bekijken van bestaande administrators. Als administrator is er ook de mogelijkheid om een query op alle databases te doen, dit kan gebruikt worden voor structuur wijzigingen van de database.
De administrator kan extra talen toevoegen. Deze talen kunnen gekozen worden bij het aanmaken van een nieuwe Multilogue. Deze taal wordt in de "front-end" gekozen als een "front-end" gebruiker nog geen taal heeft gekozen in zijn profiel.

Grafisch is dit weergegeven in ***bijlage IV***.

2 Snelheid Multilogue

Het hoofdstuk “Snelheid Multilogue ” beschrijft de performance van de Multilogue. Bij performance moet gedacht worden aan de tijd dat het duurt om pagina’s te laden.
Om de snelheid te testen worden eerst de langzamere pagina’s gekozen die op snelheid getest gaan worden.
Hiervoor wordt gebruikt gemaakt van Firebug. Firebug is een extensie voor Firefox dat het mogelijk maakt om websites te debuggen, wijzigen(lokaal) en het controleren van websites. Door middel van de net stat optie in Firebug is het mogelijk om de tijd te zien hoe snel een pagina laadt plus de objecten die geladen wordt.

Doordat E-mark het hosten uitbesteed en er fysiek voldoende kracht aanwezig is (denk aan bandbreedte) wordt enkel de efficiëntie van de code getest. De brandbreedte wordt doorgerekend in de prijs van de Multilogue. Hierbij wordt vanuit gegaan van pieken (aantal gebruikers in een uur, grootte van pagina dat over het netwerk verstuurd wordt).

Nadat de tragere pagina’s gekozen zijn wordt door middel van profileren gekeken welke handelingen en code meer tijd in beslag nemen. Hier is Xdebug voor gebruikt in combinatie met Wincachegrind/Kcachegrind. Xdebug is een PHP extensie dat debuggen en profileren mogelijk maakt. Profiling is het bekijken van de code-efficiëntie en hoe snel deze doorlopen wordt. Wincachegrind en Kcachegrind zijn voor het visualiseren van de data. Hierbij is Kcachegrind in staat om diagrammen te produceren.

Pagina´s zijn gekozen aan de hand van de tijd die een pagina erover doet om te laden of pagina´s die interessant zijn om te bekijken.

2.1 “front-end”

Voor de “front-end” zijn dit:

1) Hoofdpagina zonder ingelogd te zijn
2) Discussiekamer met een vraag waarbij het aantal reacties meer is dan 100
3) De “who is online” pagina

De algemene laadtijd voor de “front-end” ligt rond de één seconde. Dit is als er één gebruiker gebruik maakt van de applicatie.

ad1 Hoofdpagina zonder ingelogd te zijn

De hoofdpagina is gekozen om de tijd weer te geven als er geen data opgehaald wordt. De grafiek is afgebeeld in ***bijlage V***. Onderstaande tekst refereert naar wat er in de grafiek staat.
De totale tijd aan code die uitgevoerd wordt is 153 milliseconde.
De “main” spreekt voor zich. De meeste tijd wordt bestaan aan het grafisch opbouwen van de pagina (buildView functie). Naast het grafisch opbouwen van de pagina wordt er nog tijd besteed aan inladen van de configuratiewaardes die vastgelegd zijn in het bestand setup.inc.php. Denk hierbij aan databaseconfiguratie en het selecteren van sjablonen.

Daarnaast wordt de taal bepaald en de bijbehorende vertalingen geladen.

De buildView bestaat uit het laden van het sjabloon (Template->Template) en het strippen van de “template” blokken en variabelen (mathBlocks).
Voordat Caller::processCall aangeroepen wordt, wordt de URL gestript op de ‘/’. Voor de slash staat de klasse en na de slash de functie die aangeroepen moet worden van de desbetreffende klasse. In het blok Caller::processCall wordt er een instantie aangemaakt van de klasse. In het Login->execute blok die van uit processCall komt is “Login” de klasse en execute de methode die aangeroepen wordt. In dit blok wordt de functie ook daadwerkelijk uitgevoerd. Vervolgens wordt de data in de sjabloon gezet.

ad2 Discussiekamer

Het schema is weergegeven in ***bijlage VI***. De code wordt uitgevoerd in 800 milliseconde. Wat opvalt is de hoeveelheid tijd besteed wordt aan het uitlezen en vullen van de sjabloon. Dit is in totaal 34% van de totale tijd.

Bij de functies starterQuestion en renderResponses gaat ook tijd verloren. Hierbij kan wellicht tijdwinst behaald worden.
Doordat de Multilogue database driven is moet er ook gekeken worden naar database queries (***bijlage VII***). Hieruit is te zien dat de totale tijd voor queries maar negen procent bedraagt. De voornaamste besparing zit hier dus in de sjabloon.

ad3 Who is online

Deze pagina is getest met 46 gebruikers online. De tijd bedraagt dan 400 milliseconde. Wat opvalt is dat de querytijd maar liefst 22% van de laadtijd in beslag neemt. Mochten het aantal gebruikers erg hoog liggen dan kan dit oplopen. Eventuele winst kan hier behaald worden. Daarnaast kan misschien de functie waarbij de overview de query uitvoert geoptimaliseerd worden.

2.2 Editor

Voor de editor “back-end” zijn dit:

1) Weergeven van de video’s onder bijlagen
2) Het weergeven statistieken (weergeven aan aantal gebruikers en aantal reacties)
3) Statistieken van polls

ad1 Weergeven van video’s

De code duurt 500 milliseconde om uitgevoerd te worden waarvan 50% aan database queries verloren gaat.

ad2 Weergeven statistieken

De code is uitgevoerd in 100 ms. Hier is minder snelheidwinst te winnen. Het grootste gedeelte bestaat uit het matchen van het sjabloon (26%).

ad3 Statistieken polls

Voornaamste tijd zit hier in het laden van googlechart. Voor het tonen van de modellen op deze manier wordt het lastig om een alternatief te verzinnen die sneller is. De code is klaar in 200 ms.

Gemiddelde laadtijd van de editor”back-end” bedraagt zo’n 350 milliseconde. Bovenstaande pagina’s hebben een laadtijd(dus niet enkel code) boven de één seconde. De statistiekpagina’s zijn wat trager, dit komt met name door het generen van de grafieken. De laadtijd van de pagina die het aantal reacties per dag toont bedraagt ~15 seconde.

2.3 Administrator

Voor de administrator “back-end” is er gekozen voor de:

1) Overzicht van Multilogues

ad1 Overzicht Multilogues

De code heeft 575 milliseconde nodig om uitgevoerd te worden. Verassend is dat het vullen van de sjabloon ongeveer 50%(!) van de tijd in beslag neemt. 18% van de totale tijd wordt besteed aan het uitvoeren van queries.

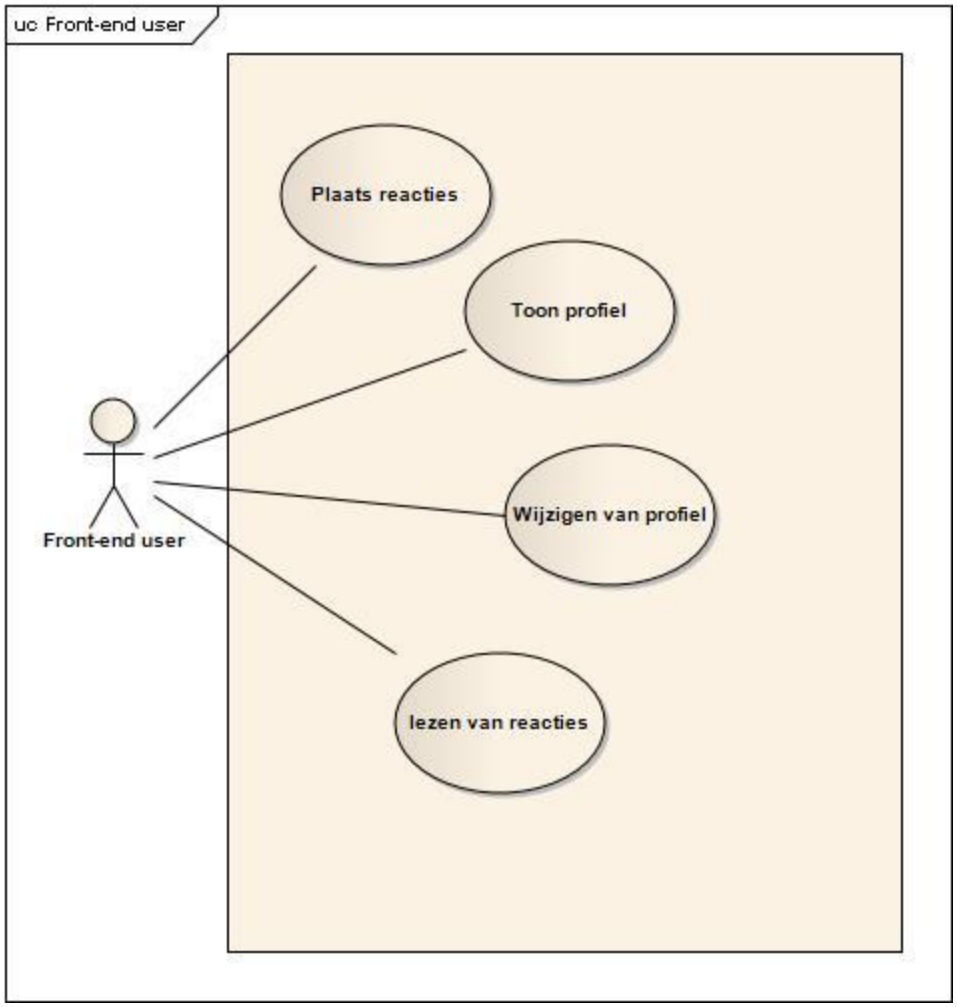
De administrator “back-end” voelt vrij snel aan. Deze “back-end” heeft dan ook niet veel toeters en bellen in grafisch opzicht. Gemiddelde laadtijd bedraagt 80 milliseconde maar er is ook niet veel data om te verwerken.

Conclusie

Globaal kan er gezegd worden dat de profilering van verschillende pagina's uiteenlopen. Bij de "front-end" zal er waarschijnlijk meer snelheidswinst te halen zijn dan bij de editor "back-end" en administrator "back-end". De zwaardere onderdelen in de Multilogue zijn voornamelijk queries en de template engine. Maar dat betekent niet dat de functies niet geoptimaliseerd kunnen worden.

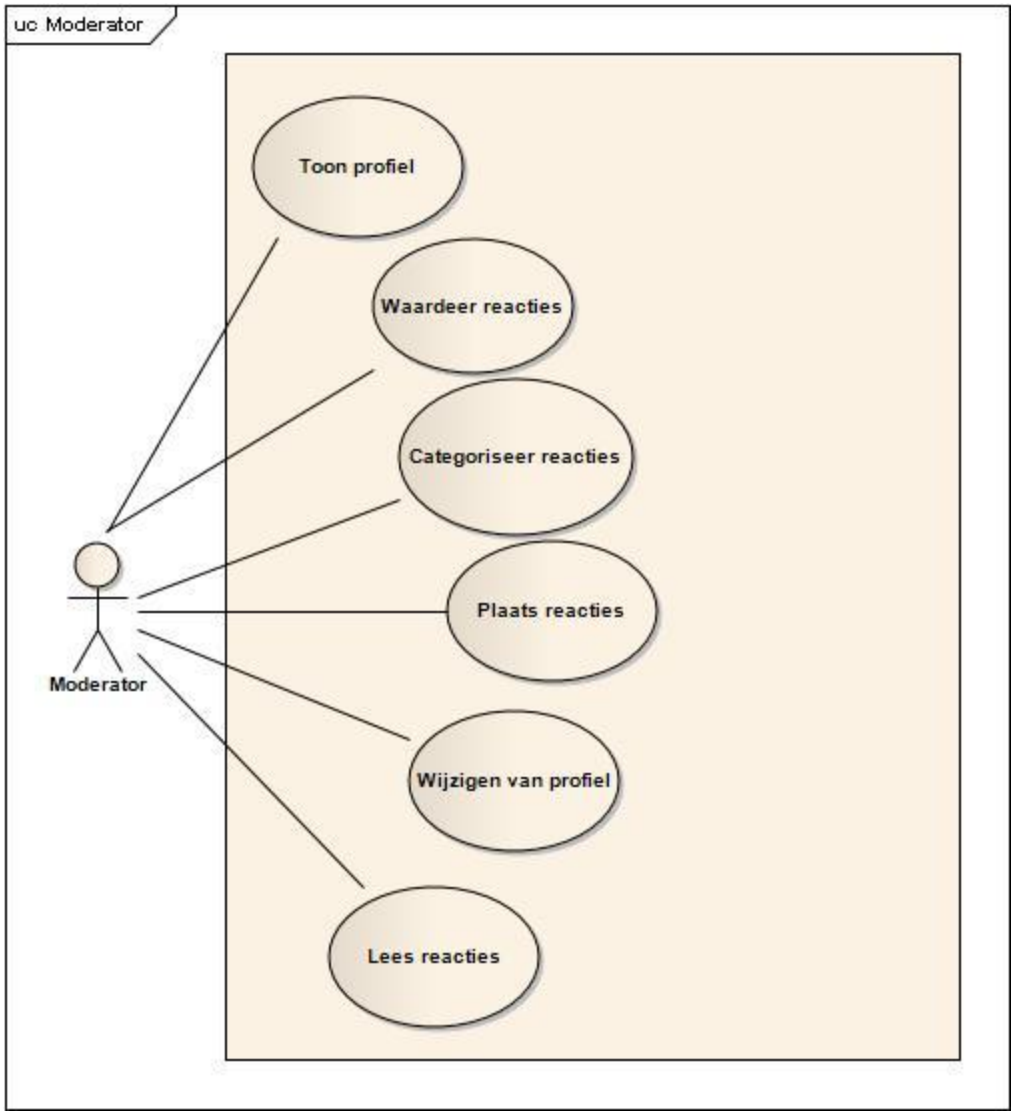
De nieuwe applicatie wordt geschreven in ASP.NET in combinatie met C#.
ASP.NET verschilt veel van PHP. ASP.NET is geoptimaliseerd en wordt gecompileerd naar een lagere level taal (Common intermediate language, voorheen Microsoft Intermediate Language), dat betekent dat de code die ingevoerd is gereduceerd.
Als een PHP script wordt aangesproken wordt het script "on-the-fly" gecompileerd en uitgevoerd. De Multilogue maakt gebruikt van de APC(Alternatieve PHP cache) module. De APC zorgt ervoor dat PHP pagina's gecompileerd opgeslagen worden in het geheugen. Ondanks dat er gebruik gemaakt is van deze module kan er snelheidswinst behaald worden door de queries te optimaliseren en gebruik te maken van een snelle template engine.

Bijlage I – Usecase “front-end” gebruiker



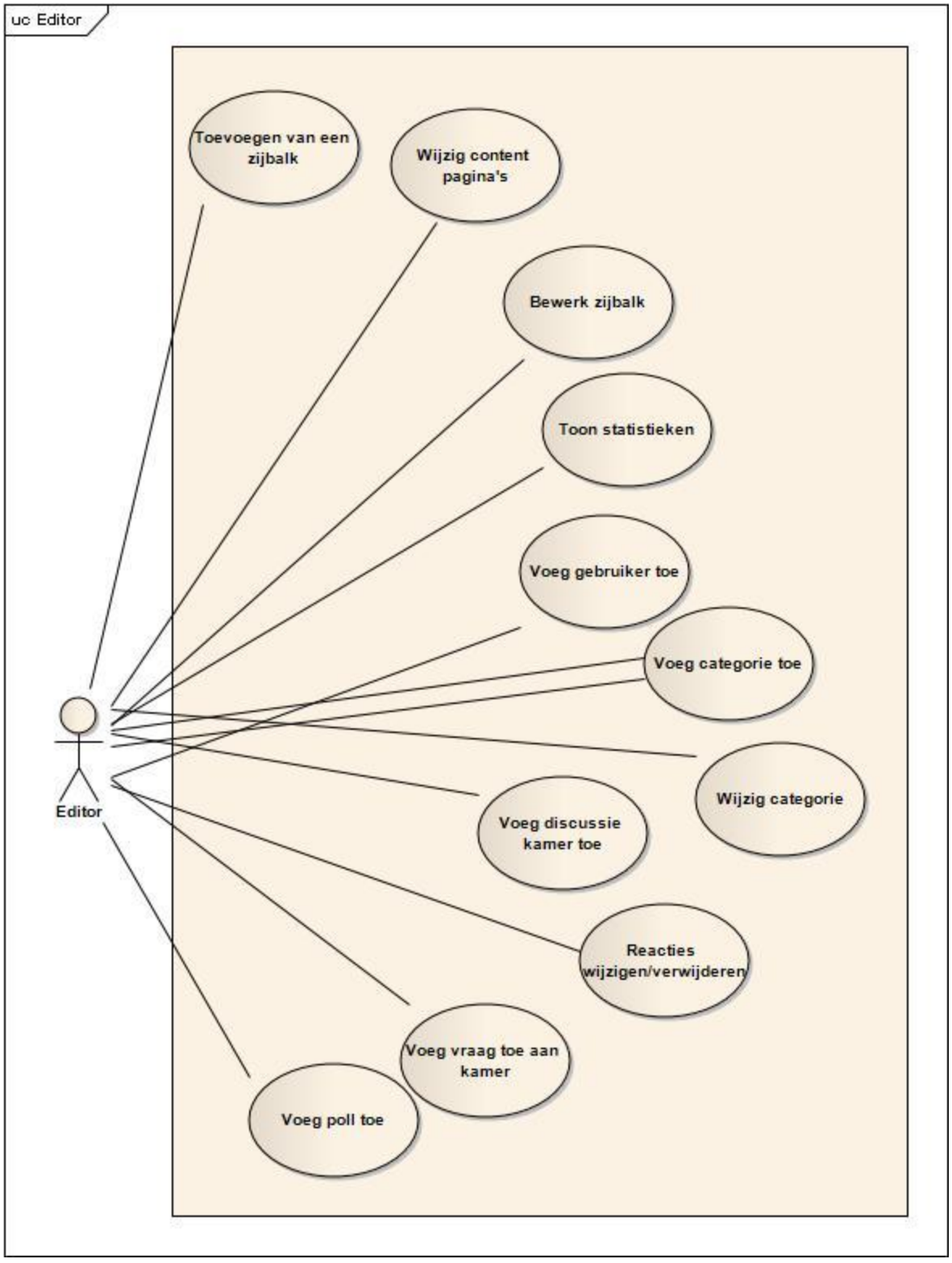
Belangrijke activiteiten van front-end user

Bijlage II – Usecase moderator



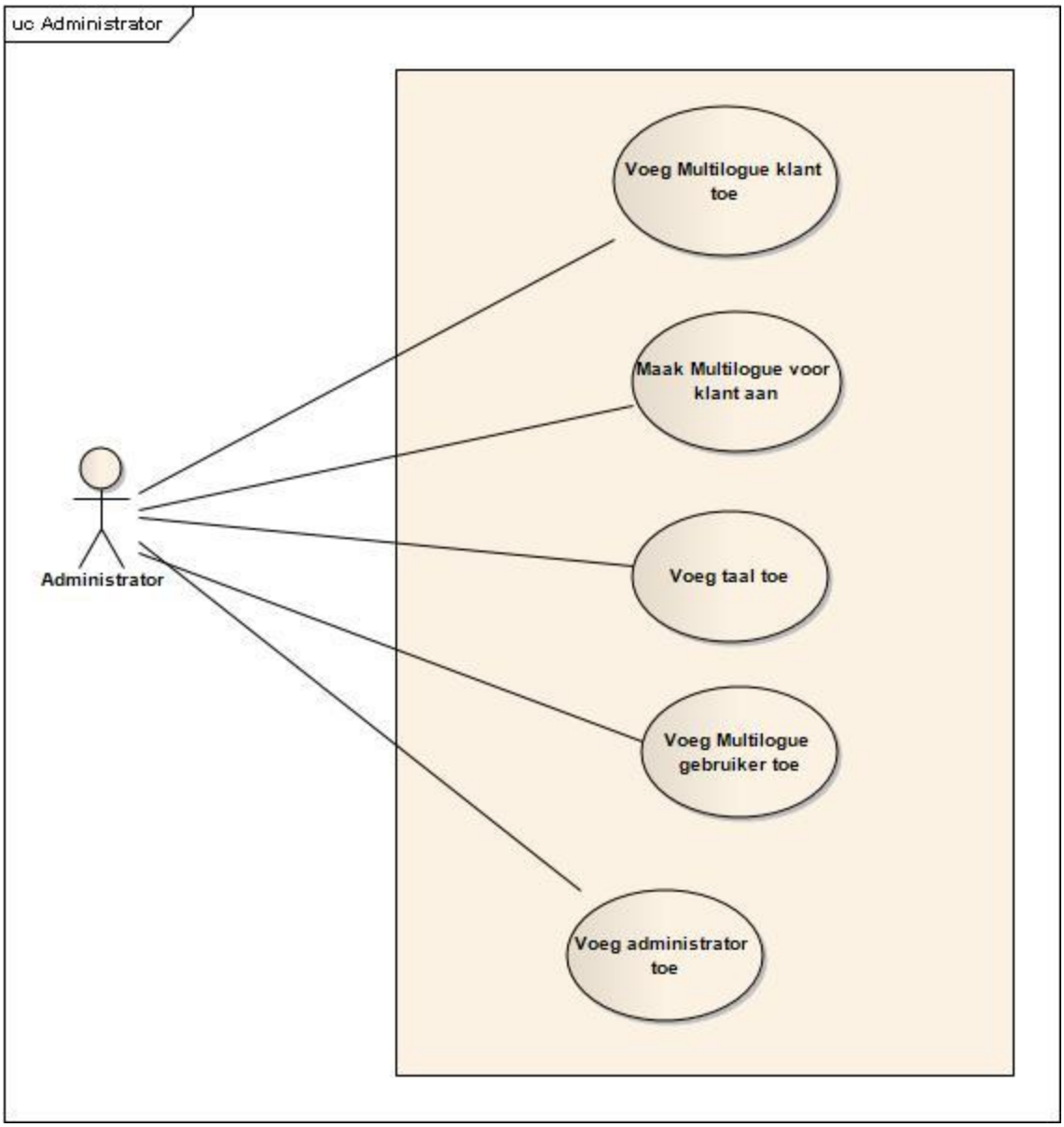
Belangrijke activiteiten moderator

Bijlage III – Use case Editor



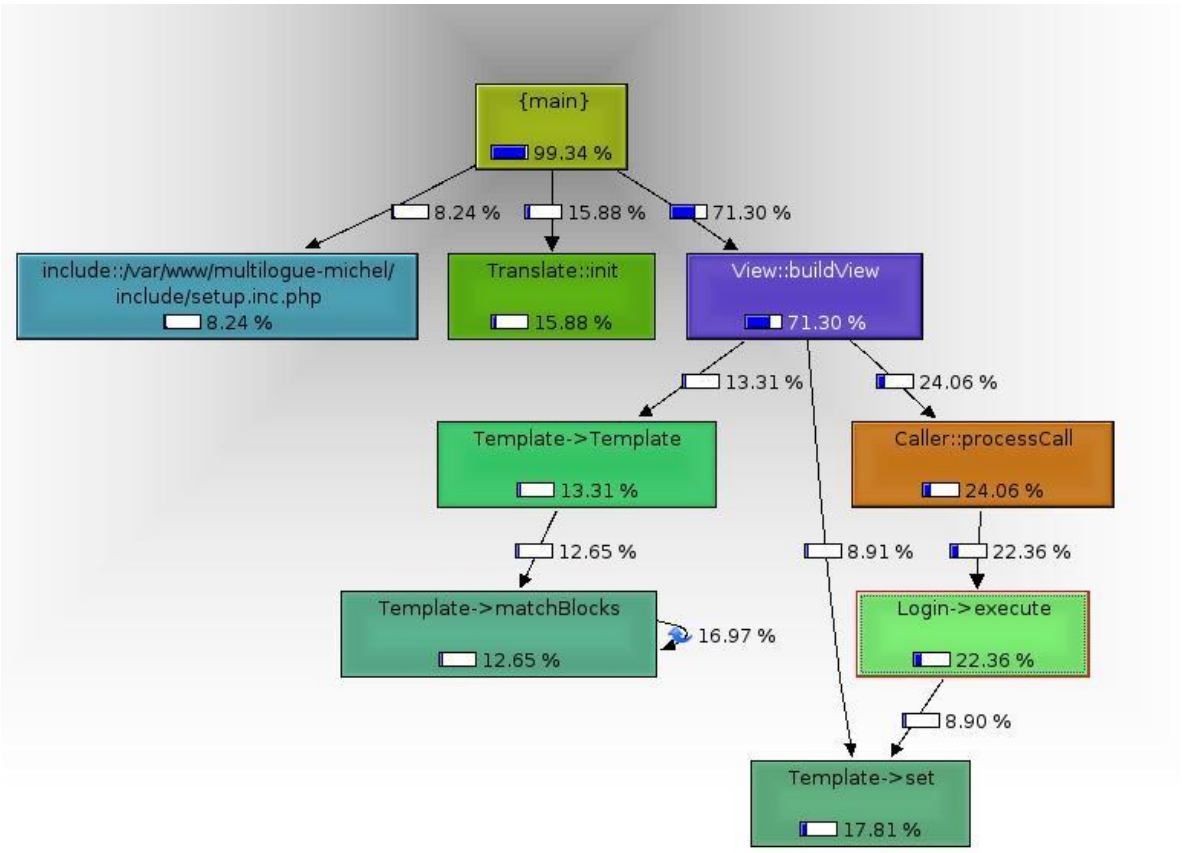
Belangrijke activiteiten editor

Bijlage IV – Usecase administrator



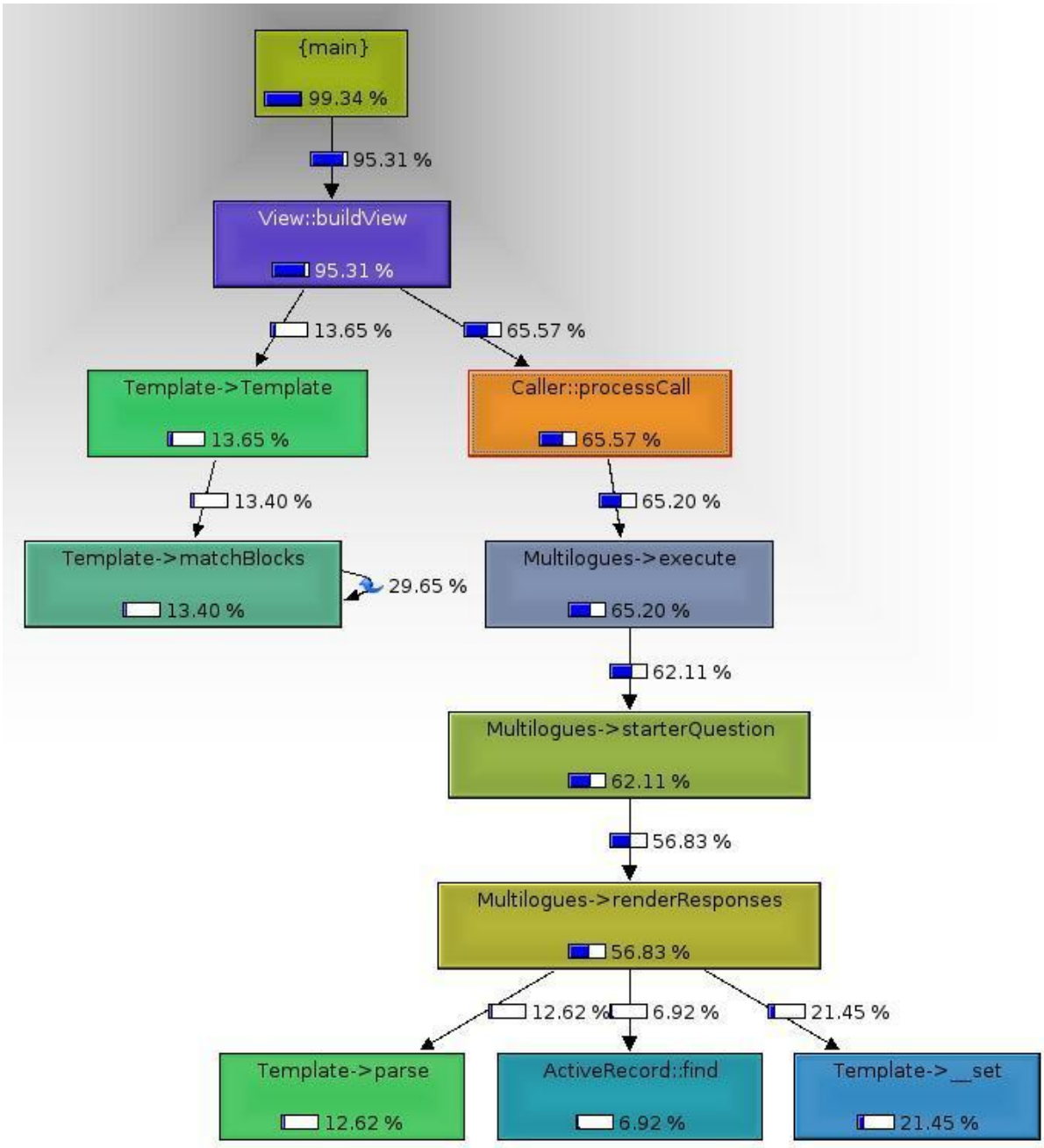
Belangrijke activiteiten Administrator

Bijlage V – Profilering hoofdpagina



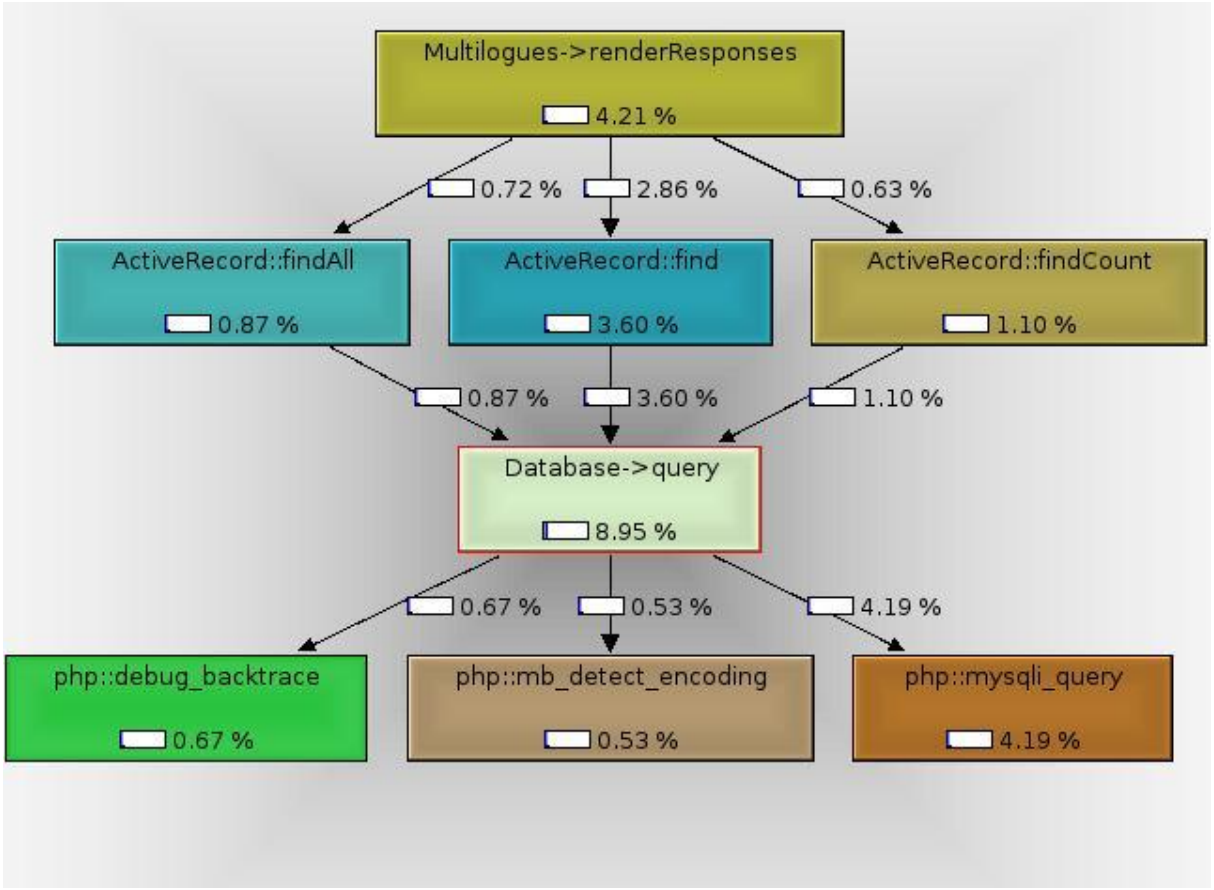
Profilering van de hoofdpagina in de "front-end"

Bijlage VI – Discussiekamer



Profileren van een discussiekamer die geladen is

Bijlage VII – Query discussiekamer



Profilering van query van een discussiekamer

Functioneel ontwerp

17 februari 2010

Versie 2

Michel Dickhoff

Samenvatting

De applicatie die ontwikkeld gaat worden dient voor het communiceren tussen gebruikers van bedrijven door middel van het plaatsen van reacties. De applicatie maakt onderscheid in vier verschillende typen gebruikers, namelijk de normale gebruiker, moderator, editor en administrator.

In grote lijnen plaatst de normale gebruiker reacties, kan stemmen op polls en kan zijn profiel wijzigen. De editor zorgt ervoor dat er kamers aanwezig zijn waarin vragen opgesteld zijn. Ook polls worden door de editor verzorgt. Een moderator heeft dezelfde mogelijkheden als een normale gebruiker. Enkel de moderator zorgt dat de reacties van gebruikers niet afwijken van het onderwerp. Dit kan door middel van aansturen. De administrator verzorgt het opzetten van de applicatie.

Naast de functionaliteit van het systeem moet er gedacht worden aan niet functionele eisen. De niet functionele eisen betreffen met name veiligheid en performance. Het is gewenst dat de PHP maatapplicatie overtroffen wordt op gebied van snelheid en uitbreidbaarheid.

Qua lay-out kan er gekozen worden om de huidige stijl van de Multilogue te behouden. De front-end moet aantrekkelijk zijn voor de gebruiker. De back-end van de editor moet zakelijk overkomen met een nette lay-out. De administrator back-end kan qua lay-out eenvoudiger aangezien enkel medewerkers van E-mark deze back-end te zien krijgen.

Voor de duidelijkheid zijn er enkele workflows beschreven. Deze zijn weergegeven in diagrammen. De workflows zijn zo eenvoudig mogelijk gehouden zodat een niet technische informaticus het verloop ook snapt. De volgende workflows zijn beschreven: registratieproces, reactie rapporteren, aanmaken van een discussiekamer met vraag en het opzetten van de applicatie door de administrator.

Als onderdeel van projectmanagement zijn de risico's en planning bijgewerkt. De analysefase en het schrijven van het functioneel ontwerp verliep spoediger dan aanvankelijk gedacht werd bij het schrijven van het plan van aanpak.

Inhoudsopgave

Versiebeheer

Op 25 februari nagekeken door Nils Corver:

1) Term formvield uitgelegd.
2) IE7 ondersteuning in plaats van IE6.

Inleiding

De fase waarin het project zich momenteel bevindt(17 februari) is de elaboratiefase. Het analysedocument is reeds opgeleverd. De komende documenten die opgeleverd gaan worden in deze zelfde fase zijn het functioneel en technisch ontwerp.

Het document richt zich met name op het “wat gaat het systeem doen” en “wat moet het systeem kunnen”. De reden dat het functioneel ontwerp geschreven is, is dat er op deze manier inzicht verkregen wordt in deze twee bovengenoemde aspecten. Dit document kan door eenieder gelezen worden die geïnteresseerd is.

De scope van het product dat beschreven is in het functioneel ontwerp beperkt zich tot het beschrijven van de functies die het systeem moet gaan bieden voor de “front-end” user, moderator, editor en administrator. De functies zijn beschreven door visuele analyse die gedaan is in de analysefase. De code is niet doorgenomen, omdat dit tijdrovend is en visuele analyse voldoende is om de functionaliteit vast te leggen van de te ontwikkelen applicatie. Bovendien gaat de applicatie geschreven worden in ASP.NET. Dat betekent dat oude code van de Multilogue niet hergebruikt kan worden doordat deze in PHP geschreven is.

Het document richt zich op de software die ontwikkelt gaat worden en niet op de hardware die benodigd is.

De prioriteiten worden later toegekend door de MoSCoW methode toe te passen. Deze worden later beschreven in een daarvoor bedoeld document.

Het document borduurt verder aan de hand van het analysedocument. Het wordt aangeraden het functioneel ontwerp pas te lezen nadat het analysedocument doorgenomen is. Dit document geeft geen technische aspecten zoals klassendiagrammen en database-informatie. Die informatie wordt beschreven in het technisch ontwerp.

Hoofdstuk 1 beschrijft in grote lijnen wat de applicatie moet gaan doen. Hoofdstuk 1 beschrijft tevens de gebruikers die de applicatie gaan gebruiken.
Hoofdstuk 2 beschrijft de functionele eisen waaraan het te maken systeem moet voldoen. Use cases en scenario’s zijn beschreven in ***hoofdstuk 3***.
Nadat de functionele eisen opgehelderd zijn worden de niet functionele eisen toegelicht in ***hoofdstuk 4***.
Hoofdstuk 5 dient om een beeld te krijgen hoe de applicatie er grafisch gaat uitzien. Hier worden mockups gegeven van de front-end en back-ends van zowel de editor en administrator. ***Hoofdstuk 6*** beschrijft de workflow van enkele procedures in het systeem. Deze zijn eenvoudig weergegeven in diagrammen.

Een onderdeel van projectmanagement is het bijhouden van risico’s en planning. Dit is beschreven in ***hoofdstuk 7***.

1 Globale beschrijving

Hoofdstuk 1, Globale beschrijving, beschrijft geen specifieke eisen maar geeft een algemeen beeld van de karakteristiek van de te realiseren applicatie. Daarnaast kaart het de type gebruikers aan die deze applicatie gaan gebruiken.

1.1 Productperspectief

Het product dat ontwikkeld gaat worden is een vervanging voor de maatapplicatie Multilogue (zie het analysedocument voor informatie over Multilogue).

1.2 Functies van het product

In hele grote lijnen moet de applicatie de volgende functionaliteit bevatten:

- Er kunnen reacties geplaatst worden als antwoord op vragen die gesteld zijn in discussiekamers.
- Nieuwe kamers en vragen moeten toegevoegd kunnen worden.
- Statistieken kunnen opgevraagd worden. Denk hierbij aan het aantal reacties per dag.
- Multilogue moet opgezet kunnen worden.

1.3 Gebruikers

De gebruikers bestaan uit:

Type	**Verantwoordelijkheid/Activiteiten**
"Normale"(front-end) gebruiker	Reacties plaatsen, deelnemen aan polls, bekijken van video's.
Moderator	Categoriseren en waarderen van reacties. Zorgt ervoor dat reacties die geplaatst zijn door gebruikers niet afwijken van het onderwerp
Editor	Toevoegen van moderators en normale gebruikers. De editor heeft de mogelijkheid om statistieken te bekijken. Voegt ook discussie kamers/vragen/polls toe.
Administrator	Opzetten van de Multilogue

2 Functionele eisen

Hoofdstuk 2, Functionele eisen, beschrijft de functionele eisen waaraan de software moet voldoen. Een functionele eis heeft direct betrekking op een proces dat het systeem moet uitvoeren of informatie die het systeem moet bevatten. De eisen zijn opgedeeld per type gebruiker. Deze functionele eisen kunnen weerspiegelt worden in use cases met bijhorende use case scenario's. Dit is gedaan in hoofdstuk 3. Omdat de nieuwe applicatie een overzetting is van de Multilogue zijn de functionele eisen merendeels worden overgenomen vanuit het analysedocument. Nadat de basisfunctionaliteit geprogrammeerd is worden aanvullende eisen opgesteld die ten gunste zijn voor de gebruiker.

2.1 "Normale" gebruiker

Met een normale gebruiker wordt een gebruiker bedoeld die enkel in de "front-end" kan inloggen en geen moderatorrechten heeft. In deze paragraaf wordt "normale" gebruiker vanaf nu gebruiker genoemd.

Voor dit type gebruiker moet de te maken applicatie de volgende functies bieden:

- De applicatie moet gebruikersgegevens bewaren (voornaam, achternaam, geboortedatum, geslacht, bedrijf). Deze gegevens moeten zichtbaar en te wijzigen zijn voor de gebruiker.
- De applicatie geeft de mogelijkheid om in en uit te loggen. Dit gebeurd aan de hand van een e-mail adres of gebruikersnaam en wachtwoord.
- De applicatie biedt de mogelijkheid voor de gebruiker om mee te stemmen in polls.
- De gebruiker moet reacties kunnen plaatsen als antwoord op vragen die gesteld zijn in de discussiekamers. Ook moet de gebruiker een reactie op een reactie kunnen geven. Gebruikers kunnen alleen actieve discussiekamers bezichtigen.
- De applicatie is "te lezen" in verschillende talen. De gebruiker kiest de taal.
- De gebruiker moet het aantal gebruikers zien die op dat huidige tijdstip online zijn. De profielen zijn zichtbaar voor de gebruiker indien andere gebruikers dat toelaten.
- De gebruiker moet reacties en vragen kunnen toevoegen aan een persoonlijke favorietenlijst.
- De applicatie biedt video's aan die bekeken kunnen worden door de gebruiker.
- De gebruiker kan overige informatie zoals laats geschreven/gelezen berichten en hoogst gewaardeerde reacties/vragen zien.
- Een gebruiker kan een reactie rapporteren in geval van misbruik.

2.2 Moderator

Zoals in het analysedocument beschreven is, is een moderator een gebruiker die gesprekken in goede banen moet begeleiden. De functionele eisen die in hoofdstuk 2.1 voor de normale gebruiker zijn beschreven gelden ook voor de moderator.

Aanvullende opties voor de moderator zijn:

- De applicatie maakt het mogelijk voor moderators om reacties te categoriseren en te waarderen.

2.3 Editor

De applicatie biedt de volgende functies voor de editor:

- Een editor kan ieder type gebruiker aanmaken (met uitzondering van de administrator).
- De editor kan inloggen met een gebruikersnaam en wachtwoord.
- Een editor kan reacties die geplaatst zijn door “normale” gebruikers of moderators wijzigen en verwijderen in de front-end.
- Een editor moet discussie kamers, vragen, polls en sidebars kunnen wijzigen en toevoegen.
- Een editor kan een formfield (een formfield is een informatieveld dat toegevoegd kan worden voor de gebruiker zoals aantal huisdieren), categorie en waarderingen(waardes van waardering) toevoegen.
- Een editor kan een gebruiker zoeken en volgen.
- De applicatie geeft de mogelijkheid om inhoud te veranderen op pagina’s en de gelegenheid om welkomstberichten in te stellen.
- Een blacklist kan gevuld worden waar vulgaire woorden in opgenomen zijn zodat deze woorden geblokt worden bij het plaatsen van een reactie.
- De editor kan bijlagen uploaden.

2.4 Administrator

De applicatie moet de volgende functies bieden aan de administrator:

- De administrator moet nieuwe “Multilogue” klanten kunnen toevoegen en een overzicht van deze klanten genereren.
- De applicatie geeft de mogelijkheid om in en uit te loggen. Dit gebeurd aan de hand van een gebruikersnaam en wachtwoord.
- Een administrator kan nieuwe gebruikers toevoegen en deze bekijken. Type gebruikers die een administrator kan aanmaken zijn administrators, editors en gebruikers zonder moderator/editor rechten.
- De administrator kan talen toevoegen die vervolgens in de front-end gebruikt kunnen worden.
- De administrator kan rechtstreeks queries uitvoeren op de database.

3 Use case scenario's

Het hoofdstuk Use case scenario's beschrijft de scenario's die van toepassing zijn op de functionele eisen. De use cases zijn overgenomen uit het analysedocument. De use case diagrammen zijn zo ingedeeld dat enkel de einddoelen van de desbetreffende gebruiker er opstaan. De rest van de activiteiten zijn bijproducten. Hier is voor gekozen zodat de diagrammen overzichtelijk blijven.

Dit omdat de functionaliteit in eerste instantie hetzelfde blijft als de nu bestaande Multilogue. De focus ligt op performance en het beter inrichten van klassen zodat uitbreiding van de applicatie beter beheersbaar is. De scenario's zijn opgedeeld per type gebruiker.

De use case scenario's zijn beschreven met de aanname dat de applicatie compleet(dus ook database) en werkend is. In sommige gevallen is er een kolom specifiek. Deze kolom geeft een beter idee wat er met de use case behaald wordt. Deze kolom is enkel getoond als deze informatie bekend en nuttig is. Gedetailleerde beschrijvingen (data dat opgeslagen moet worden in de database bijvoorbeeld) worden gegeven in het technisch ontwerp.

3.1 Normale gebruiker

In ***bijlage I*** is het use case diagram weergegeven. Onderstaande tekst refereert naar dit diagram.

Use cases:

Naam	Toon profiel.
Aannamen	Gebruiker is ingelogd.
Beschrijving	Na de desbetreffende link geklikt te hebben wordt de informatie getoond die ingevuld is door de gebruiker. De volledigheid hangt af van wat de gebruiker ingevuld heeft.
Specifiek	Er wordt getoond: • Voornaam • Achternaam • E-mail • Geboortedatum • Online zichtbaar of niet • Geslacht • Formvields • Aantal uur aanwezig op bedrijf • Bedrijf
Uitzonderingen	Geen.
Resultaat	Profiel wordt getoond.

Naam	Plaats reacties.
Aannamen	Gebruiker is ingelogd. Actieve kamers aanwezig.
Beschrijving	Zodra er een poging gedaan wordt om de reactie te plaatsen wordt er gekeken of er daadwerkelijk een bericht getypt is. Is dit niet het geval dan treedt uitzondering [Geen bericht] op. Als er wel een bericht getypt is wordt de reactie geplaatst.
Uitzonderingen	[Geen bericht] Het venster waar het bericht getypt kan worden wordt gesloten en er is geen reactie geplaatst.
Resultaat	Reactie geplaatst.

Naam	Wijzigen van profiel.
Aannamen	Gebruiker is ingelogd. Profielgegevens zijn ingevoerd.
Beschrijving	Gebruiker kan velden wijzigen van zijn/haar profiel. Als een verplicht veld leeggemaakt wordt dan treedt uitzondering [Verplicht veld niet ingevuld] op. Indien alles juist is ingevuld wordt het opgeslagen in de database.
Specifiek	De volgende informatie is te wijzigen: • Voornaam • Achternaam • Wachtwoord • Taal • Formvields • Geboortedatum • Geslacht • Foto • Zichtbaar online • Aantal uur aanwezig op bedrijf • Bedrijf
Uitzonderingen	[Verplicht veld niet ingevuld] Gebruiker krijgt melding dat er nog een veld ontbreekt die ingevuld moet worden.
Resultaat	Profielgegevens zijn gewijzigd.

Bijproducten:

Naam	Inloggen.
Aannamen	Gebruiker bevindt zich op de “inlogpagina”. Of gebruiker is geredirect door het bezoeken van een beveiligde pagina zonder ingelogd te zijn.
Beschrijving	Gebruiker vult gebruikersnaam of e-mail (keuze wordt gemaakt tijdens het opzetten van Multilogue) in combinatie met een wachtwoord. De applicatie bekijkt of de gebruiker bekend is in het systeem. Zo niet dan treed uitzondering [Gebruiker niet bekend] op. Instellingen zoals taal worden geladen.
Uitzonderingen	[Gebruiker niet bekend] Een melding wordt gegeven dat de gebruiker niet

	bekend is in het systeem. Inloggen is niet mogelijk.
Resultaat	Gebruiker is ingelogd.

Naam	Uitloggen.
Aannamen	Gebruiker is ingelogd.
Beschrijving	Gebruiker drukt op de daarvoor bedoelde knop/link. Gebruiker logt uit en vernietigt de waardes die aangeven dat de gebruiker is ingelogd. De sessiedata wordt opgeslagen in de database.
Uitzonderingen	Geen.
Resultaat	Gebruiker is uitgelogd.

Naam	Voeg stem toe aan poll.
Aannamen	Gebruiker is ingelogd.
Beschrijving	De gebruiker kan een antwoord kiezen als antwoord op de vraag van de poll. Indien er geen antwoord gekozen is treed uitzonder [geen antwoord gekozen] op. Indien de gebruiker al gestemd heeft treed uitzondering [Al gestemd] op.
Uitzonderingen	[Geen antwoord gekozen] Er wordt geen actie ondernomen. [Al gestemd] Gebruiker ziet alleen stemstatistieken.
Resultaat	Stem is opgenomen en het aantal stemstatistieken wordt getoond.

Naam	Kies taal.
Aannamen	Gebruiker is ingelogd. Één of meerdere talen naast de standaardtaal aanwezig.
Beschrijving	Gebruiker klikt link/knop voor gewenste taal. Inhoud op pagina's verandert in de gewenste taal.
Uitzonderingen	Geen.
Resultaat	Taal van inhoud van pagina's is aangepast.

Naam	Toon poll.
Aannamen	Gebruiker is ingelogd. Polls aanwezig waar op gestemd is.
Beschrijving	De generieke polls zijn zichtbaar op de homepagina. Vragenpolls zijn beschikbaar door naar de desbetreffende discussiekamer te gaan.
Uitzonderingen	Geen.
Resultaat	Gebruiker ziet “lopende” polls.

Naam	Toon “wie is online”.
Aannamen	Gebruiker is ingelogd.
Beschrijving	Gebruiker klikt op de link “wie is online”. De gebruiker ziet alle profielen die online zijn. Indien er geen gebruikers online zijn treedt uitzondering [Geen gebruiker is online] op. Gebruikers die niet online gezien willen worden kunnen dat aangeven in profielgegevens. Hier treed uitzondering [Niet zichtbaar] op.
Uitzonderingen	[Geen gebruiker is online] Er worden geen profielen getoond. [Niet zichtbaar] Gebruiker die niet online gezien wilt worden wordt niet weergegeven op de “wie is online” pagina.
Resultaat	Gebruiker ziet het aantal en welke mensen online zijn.

Naam	Voeg reacties/vragen toe aan favorietenlijst.
Aannamen	Gebruiker is ingelogd. Er zijn reacties/vragen om toe te voegen.
Beschrijving	Gebruiker voegt een reactie toe aan een lijst door een knop te drukken. Deze lijst is zichtbaar profielgegevens. De favorietenlijst kan ook geleegd worden op de profielpagina.
Uitzonderingen	Geen.
Resultaat	Reactie/vraag is toegevoegd aan favorietenlijst.

Naam	Rapporteer reacties.

Aannamen	Gebruiker is ingelogd.
Beschrijving	Gebruiker drukt op de daarvoor bedoelde knop om een reactie te rapporteren. De gebruiker heeft de mogelijkheid via een tekstveld te vermelden wat er niet deugt.
Uitzonderingen	Geen.
Resultaat	Een mail is gestuurd naar een editor die het bericht kan verwijderen/wijzigen.

Naam	Bekijk video.
Aannamen	Gebruiker is ingelogd.
Beschrijving	Gebruiker gaat naar de daarvoor bedoelde videopagina. Hier staan de beschikbare video's klaar om te bekijken. Indien er geen video's zijn treed uitzondering [Geen video's] op.
Uitzonderingen	[Geen video's] Er wordt een bericht gegenereerd dat er geen video's aanwezig zijn.
Resultaat	Gebruiker kan video selecteren en bekijken.

Naam	Bekijk laatst geschreven/gelezen reactie.
Aannamen	Gebruiker is ingelogd.
Beschrijving	Indien de gebruiker een reactie plaatst op een reactie worden deze getoond op de homepagina. Indien de gebruiker nog niets gelezen/geschreven heeft treed uitzondering [Geen reactie gelezen/geschreven] op.
Uitzonderingen	[Geen reactie gelezen/geschreven] Laatst geschreven/gelezen reactie wordt niet getoond.
Resultaat	Gebruiker ziet laatst geschreven/gelezen reactie.

Naam	Voltooi registratie.
Aannamen	Gebruiker is ingelogd. Gebruiker heeft zich nog niet geregistreerd.

Beschrijving	Nadat een editor een nieuwe gebruiker heeft aangemaakt moet de gebruiker zodra deze inlogt zich registreren. Hierbij kan de gebruiker aanvullende informatie geven/wijzigen(land, geboortedatum, bedrijf…). Op deze manier kan de editor met minimale informatie(voornaam, achternaam, e-mail) een gebruiker aanmaken. Als de editor ervoor gekozen heeft tijdens het aanmaken van de gebruiker om de gebruiker actief te maken, dan treed uitzonder [Actief] op.
Uitzonderingen	[Actief] Gebruiker hoeft zich niet te registreren. Aanvullende informatie kan gegeven worden op de profielpagina.
Resultaat	Registratie is voltooid.

3.2 Moderator

Aangezien de moderator hetzelfde kan als de normale gebruiker worden de scenario's die bij de normale gebruiker beschreven zijn hier niet vermeld. Enkel aanvullende use- cases worden vermeld. Het use case diagram is te vinden in ***bijlage II***.

Naam	Categoriseer reacties.
Aannamen	Gebruiker is ingelogd. Reacties zijn geplaatst.
Beschrijving	De moderator selecteert de reactie die hij/zij wilt categoriseren. Dit geschiedt door eenvoudig vinkjes neerzetten. Editors kunnen op basis van categorie zoekopdrachten formuleren.
Uitzonderingen	Geen.
Resultaat	Reactie is gecategoriseerd.

Naam	Waardeer reacties.
Aannamen	Gebruiker is ingelogd. Reacties zijn geplaatst.
Beschrijving	Moderator kiest de reactie die gewaardeerd dient te worden. Vanuit een dropdown box kan de waardering gekozen worden. Voor de editor is deze informatie te zien.
Uitzonderingen	Geen.
Resultaat	Reactie is gewaardeerd.

3.3 Editor

In ***bijlage III*** is het use case diagram weergegeven. Onderstaande tekst refereert naar dit diagram.

Use cases:

Naam	Toevoegen van een zijbalk.
Aannamen	Editor is ingelogd. Pagina's aanwezig om zijbalk te plaatsen.
Beschrijving	De editor kiest een pagina waar de zijbalk op toegevoegd moet worden en vult de inhoud van deze zijbalk.
Uitzonderingen	Geen.
Resultaat	Zijbalk is geplaatst op gewenste pagina.

Naam	Bewerk zijbalk.
Aannamen	Editor is ingelogd. Pagina's aanwezig. Bestaande zijbalken.
Beschrijving	De editor kan titel en inhoud van de zijbalk veranderen.
Uitzonderingen	Geen.
Resultaat	Zijbalk is gewijzigd.

Naam	Wijzig pagina's.
Aannamen	Editor is ingelogd. Pagina's aanwezig.
Beschrijving	Het wijzigen van pagina's beperkt zich tot het aanpassen van inhoud op vaste plaatsen. De editor kiest de gewenste pagina en kan inhoud in verschillende talen aanpassen.
Uitzonderingen	Geen.
Resultaat	Inhoud op pagina is gewijzigd.

Naam	Toon statistieken.
Aannamen	Editor is ingelogd.
Beschrijving	Editor kan statistieken bekijken die betrekking hebben op algemeen, polls, reacties en waarderingen. Details worden beschreven in het technisch ontwerp.
Uitzonderingen	Geen.
Resultaat	Statistieken van gewenst onderdeel wordt getoond.

Naam	Voeg categorie toe.
Aannamen	Editor is ingelogd.
Beschrijving	Editor voegt categorie toe die de moderator kan toekennen aan een reactie.
Uitzonderingen	Geen.
Resultaat	Categorie is toegevoegd.

Naam	Wijzig categorie.
Aannamen	Editor is ingelogd.
Beschrijving	Categorienaam is wijzigbaar voor de editor.
Uitzonderingen	Geen.
Resultaat	Categorienaam gewijzigd.

Naam	Voeg gebruiker toe.
Aannamen	Editor is ingelogd.
Beschrijving	Editor voegt een gebruiker toe. Deze gebruiker kan ook editor of moderator rechten toegekend krijgen. Mocht de gebruiker al bestaan (controle op e-mailadres) dan treedt uitzondering [Gebruiker bestaat al] op. Het toevoegen van een gebruiker kan ook door middel van een CSV bestand.
Uitzonderingen	[Gebruiker bestaat al] Editor krijgt een melding dat de gebruiker al bestaat. Deze gebruiker wordt niet toegevoegd.

Resultaat	Gebruiker is toegevoegd.

Naam	Voeg discussiekamer toe.
Aannamen	Editor is ingelogd.
Beschrijving	Editor voegt een discussiekamer toe die aanvankelijk leeg is. In deze discussiekamer kan één of meerdere vragen gecreëerd worden.
Uitzonderingen	Geen.
Resultaat	Discussiekamer toegevoegd.

Naam	Voeg vraag toe aan kamer.
Aannamen	Editor is ingelogd. Discussiekamers aanwezig.
Beschrijving	De editor kiest een discussiekamer waar de vraag terecht moet komen. De editor kan hier de vraag neerzetten. De editor vult de titel in en de werkelijke vraag.
Uitzonderingen	Geen.
Resultaat	Vraag toegevoegd.

Naam	Reacties verwijderen/wijzigen.
Aannamen	Editor is ingelogd. Reacties aanwezig.
Beschrijving	De editor kan aan de front-end reacties verwijderen/wijzigen. De editor kiest een reactie en klikt op de delete/wijzig knop.
Uitzonderingen	Geen.
Resultaat	Reactie is verwijderd/gewijzigd.

Naam	Voeg poll toe.
Aannamen	Editor is ingelogd.

Beschrijving	De editor kan een generieke poll maken evenals een poll die gekoppeld is aan een vraag in een discussiekamer.
Uitzonderingen	Geen.
Resultaat	Poll is toegevoegd.

Bijproducten:

Naam	Inloggen.
Aannamen	Gebruiker bevindt zich op de "inlogpagina" van de editor back-end.
Beschrijving	Gebruiker vult gebruikersnaam in combinatie met een wachtwoord. De applicatie bekijkt of de gebruiker bekend is in het systeem. Zo niet dan treed uitzondering [Gebruiker niet bekend] op.
Uitzonderingen	[Gebruiker niet bekend] Een melding wordt gegeven dat de gebruiker niet bekend is in het systeem. Inloggen is niet mogelijk
Resultaat	Editor is ingelogd.

Naam	Uitloggen.
Aannamen	Editor is ingelogd.
Beschrijving	Gebruiker drukt op de daarvoor bedoelde knop/link. Editor logt uit en vernietigt de huidige sessie
Uitzonderingen	Geen.
Resultaat	Gebruiker is uitgelogd.

Naam	Toon vertalingen.
Aannamen	Editor is ingelogd. Multilogue is meertalig.
Beschrijving	Bij het tonen vertalingen moet gedacht worden aan het percentage van wat er vertaalt is qua inhoud op pagina's. De gebruiker ziet dit in percentages.
Uitzonderingen	Geen.

Resultaat	De editor ziet de hoeveelheid vertalingen die gedaan zijn.

Naam	Voeg formvelden toe.
Aannamen	Editor is ingelogd.
Beschrijving	Een formveld moet gezien worden als een veld dat ingevuld moet worden door de gebruiker als deze zich registreert. Op deze manier is het mogelijk om een veld toe te voegen en ook te kiezen wat de keuzes zijn (tekstveld, dropdown box…)
Uitzonderingen	Geen.
Resultaat	Formveld is toegevoegd.

Naam	Toon formvelden.
Aannamen	Editor is ingelogd.
Beschrijving	Toon de dynamische formvelden die toegevoegd zijn door een editor. Deze formvelden kunnen aan verschillende pagina's toegevoegd worden. Denk hieraan bij aan het zoeken van een gebruiker of een registratiepagina.
Uitzonderingen	Geen.
Resultaat	Formvelden zijn getoond.

Naam	Voeg woorden toe aan blacklist.
Aannamen	Editor is ingelogd.
Beschrijving	Woorden worden toegevoegd aan de blacklist. Deze woorden verschijnen niet aan de voorkant. Er kan gekozen worden om gebruik te maken van censuur. Dubbele woorden worden maar één keer toegevoegd.
Uitzonderingen	Geen.
Resultaat	Woord(en) toegevoegd aan blacklist.

Naam	Zoek gebruiker.
Aannamen	Editor is ingelogd. Er bestaan gebruikers.
Beschrijving	Editor kan op basis van verschillende criteria (naam, geslacht, email…) een gebruiker zoeken. Indien de gebruiker niet gevonden is treedt uitzondering [Gebruiker niet gevonden] op.
Uitzonderingen	[Gebruiker niet gevonden] Scherm blijft leeg.
Resultaat	Toont profiel van gebruiker.

Naam	Instellen van Multilogue info bericht.
Aannamen	Editor is ingelogd.
Beschrijving	Editor kan een welkomstbericht genereren aan de voorkant. Ook kan de editor een bericht instellen als de Multilogue nog niet actief is.
Uitzonderingen	Geen.
Resultaat	Multilogue info bericht is geplaatst.

Naam	Toon polls.
Aannamen	Editor is ingelogd. Polls aanwezig.
Beschrijving	De editor kan een poll bekijken en wijzigen. Bijvoorbeeld de datum dat de poll getoond moet worden en wat voor soort poll het is. (Multiple choice of één antwoord).
Uitzonderingen	Geen.
Resultaat	Polls is getoond en eventueel gewijzigd.

Naam	Volg gebruiker.
Aannamen	Editor is ingelogd. Er is een bestaand gebruiker.
Beschrijving	Nadat de editor een gebruiker heeft gevonden kan de editor ervoor kiezen om de gebruiker te volgen. De editor ziet dan waar deze gebruiker actief is.
Uitzonderingen	Geen.
Resultaat	Gebruiker wordt gevolgd.

Naam	Voltooi registratie.
Aannamen	Editor is ingelogd in de front-end. Editor heeft zich nog niet geregistreerd.
Beschrijving	Nadat een editor een nieuwe editor heeft aangemaakt moet de editor zodra deze inlogt zich registreren. Hierbij kan de editor aanvullende informatie geven/wijzigen(land, geboortedatum, bedrijf…). Op deze manier kan de editor met minimale informatie(voornaam, achternaam, e-mail) een andere editor aanmaken. Als de editor ervoor gekozen heeft tijdens het aanmaken van de editor om de editor al actief te maken, dan treed uitzonder [Actief] op.
Uitzonderingen	[Actief] Editor hoeft zich niet te registreren. Aanvullende informatie kan gegeven worden op de profielpagina.
Resultaat	Registratie is voltooid.

3.4 Administrator

In ***bijlage IV*** is het use case diagram weergegeven. Onderstaande tekst refereert naar dit diagram.

Use cases:

Naam	Voeg Multilogue klant toe.
Aannamen	Administrator ingelogd.
Beschrijving	Enkel de naam van de klant (in dit geval het bedrijf die de kopende partij is)moet ingevuld te worden. Door op toevoegen te klikken wordt de klant toegevoegd.
Uitzonderingen	Geen.
Resultaat	Klant is toegevoegd.

Naam	Voeg administrator toe.
Aannamen	Administrator ingelogd.
Beschrijving	Administrator kan een andere administrator toevoegen door de juiste informatie te verschaffen.
Specifiek	Benodigd voor administrator toe te voegen: • Gebruikersnaam • Wachtwoord • Email • Voornaam • Achternaam
Uitzonderingen	Geen.
Resultaat	Administrator toegevoegd.

Naam	Voeg multilogue gebruiker toe.
Aannamen	Administrator ingelogd.
Beschrijving	Administrator kiest het bedrijf, waarbij vervolgens de benodigde gegevens ingevoerd moeten worden om een gebruiker toe te voegen. De administrator kan kiezen of de gebruiker een editor wordt of niet.

Uitzonderingen	Geen.
Resultaat	Gebruiker voor Multilogue is aangemaakt.

Naam	Voeg taal toe.
Aannamen	Administrator ingelogd.
Beschrijving	De administrator voegt een taal toe door de taal op te geven. Bestaat de taal al dan treed uitzondering [Taal bestaat al] op.
Uitzonderingen	[Taal bestaat al] Gebruiker krijgt melding dat taal al bestaat, deze taal wordt niet toegevoegd.
Resultaat	Taal is toegevoegd en kan gekozen worden bij het toevoegen van een Multilogue.

Naam	Maak Multilogue voor klant aan.
Aannamen	Administrator ingelogd.
Beschrijving	De administrator zet een “Multilogue op” voor een klant. Hierbij wordt een nieuwe tabel opgezet die gebruikt wordt om informatie op te slaan.
Specifiek	Er wordt gevraagd: • Standaardtaal van Multilogue • Klant van Multilogue • Naam van Multilogue • Startdatum/einddatum • Naam van database • DM instellingen
Uitzonderingen	Geen.
Resultaat	Multilogue is opgezet voor een klant.

Bijproducten:

Naam	Inloggen.
Aannamen	Gebruiker bevindt zich op de “inlogpagina” van de administrator back-end.
Beschrijving	Gebruiker vult gebruikersnaam in combinatie met een wachtwoord. De applicatie bekijkt of de gebruiker bekend is in het systeem. Zo niet dan treed uitzondering [Gebruiker niet bekend] op.
Uitzonderingen	[Gebruiker niet bekend] Een melding wordt gegeven dat de gebruiker niet bekend is in het systeem. Inloggen is niet mogelijk
Resultaat	Administrator is ingelogd.

Naam	Uitloggen.
Aannamen	Administrator is ingelogd.
Beschrijving	Gebruiker drukt op de daarvoor bedoelde knop/link. Gebruiker logt uit en vernietigt de huidige sessie.
Uitzonderingen	Geen.
Resultaat	Administrator is uitgelogd.

Naam	Toon overzicht Multilogue klanten.
Aannamen	Administrator ingelogd. Klanten zijn toegevoegd.
Beschrijving	Administrator toont een overzicht van toegevoegde klanten door de desbetreffende link te klikken.
Uitzonderingen	Geen.
Resultaat	Administrator genereert een lijst van klanten.

Naam	Toon administrators.
Aannamen	Administrator ingelogd.
Beschrijving	Administrator kiest een link waarbij alle in het systeem bekende administrators worden getoond.
Specifiek	Er wordt getoond: • Gebruikersnaam • Email • Voornaam • Achternaam
Uitzonderingen	Geen.
Resultaat	Lijst van alle administrators wordt getoond die bekend zijn in het systeem.

Naam	Toon Multilogue gebruikers per bedrijf.
Aannamen	Administrator ingelogd.
Beschrijving	Administrator kiest een bedrijf waarvan alle gebruikers (normaal, moderator en editors) worden getoond.
Uitzonderingen	Geen.
Resultaat	Toont Multilogue gebruikers per bedrijf.

Naam	Voer query uit op database.
Aannamen	Administrator ingelogd.
Beschrijving	De administrator kan queries uitvoeren op de gehele database. Is enkel nodig voor datastructuur aan te passen.
Uitzonderingen	Geen.
Resultaat	Eventuele wijziging van datastructuur in database.
Naam	Toon overzicht talen.
Aannamen	Administrator ingelogd. Talen aanwezig.

Beschrijving	Gebruiker kiest “overzicht talen” waar de administrator een overzicht van de talen ziet.
Uitzonderingen	Geen.
Resultaat	Administrator ziet toegevoegde talen.

Naam	Toon geïnstalleerde Multilogues.
Aannamen	Administrator ingelogd. Geïnstalleerde Multilogues aanwezig.
Beschrijving	Gebruiker kiest voor een overview van Multilogues. Per klant kunnen er meerdere Multilogues opgezet worden.
Uitzonderingen	Geen.
Resultaat	Administrator ziet geïnstalleerde Mutlilogues.

4 Niet functionele eisen

Een niet functionele eis verwijst naar gedragseigenschappen die het systeem moet hebben, zoals goede prestaties en gebruikersvriendelijkheid. Hier is niet de vraag wat de applicatie doet, maar hoe de applicatie het doet.

De niet functionele eisen zijn opgedeeld in:

1) Operationeel
2) Prestatief
3) Beveiliging

ad1 Operationeel

Met een operationele niet functionele eis wordt de fysieke en technische omgeving bedoeld waarin het systeem gaat functioneren.

Bij de operationele eisen moet gedacht worden aan:

- Het systeem moet met de applicatie E-mark mail kunnen werken. Dit wordt gebruikt om massamail te verzenden tijdens het aanmaken van gebruikers met een csv lijst door de editor.
- De applicatie moet met Firefox en IE7 en hoger kunnen werken.
- ASP.NET server benodigd(Internet Information Server). Dit is IIS 7 SQL SERVER 2008.

ad2 Prestatief

Deze niet functionele eis kan omschreven worden als de snelheid, kracht en betrouwbaarheid van het systeem.

- Applicatie moet opgebouwd zijn uit modules.
- Het ontwerp maakt gebruik van een 3 TIER model. Dit zijn front, runtime (ook wel business logic genoemd) en de data acces layer.
- De ASP.NET dient sneller te zijn dan de PHP maatapplicatie.
- Installatie van applicatie moet zo min mogelijk moeite kosten. Administrator database moet gescheiden zijn van de Multilogue database. De mogelijkheid om te communiceren tussen de twee databases moet aanwezig zijn.

ad3 Beveiliging

Beveiliging richt zich op bevoegde toegang tot het systeem.
Denk hierbij aan:

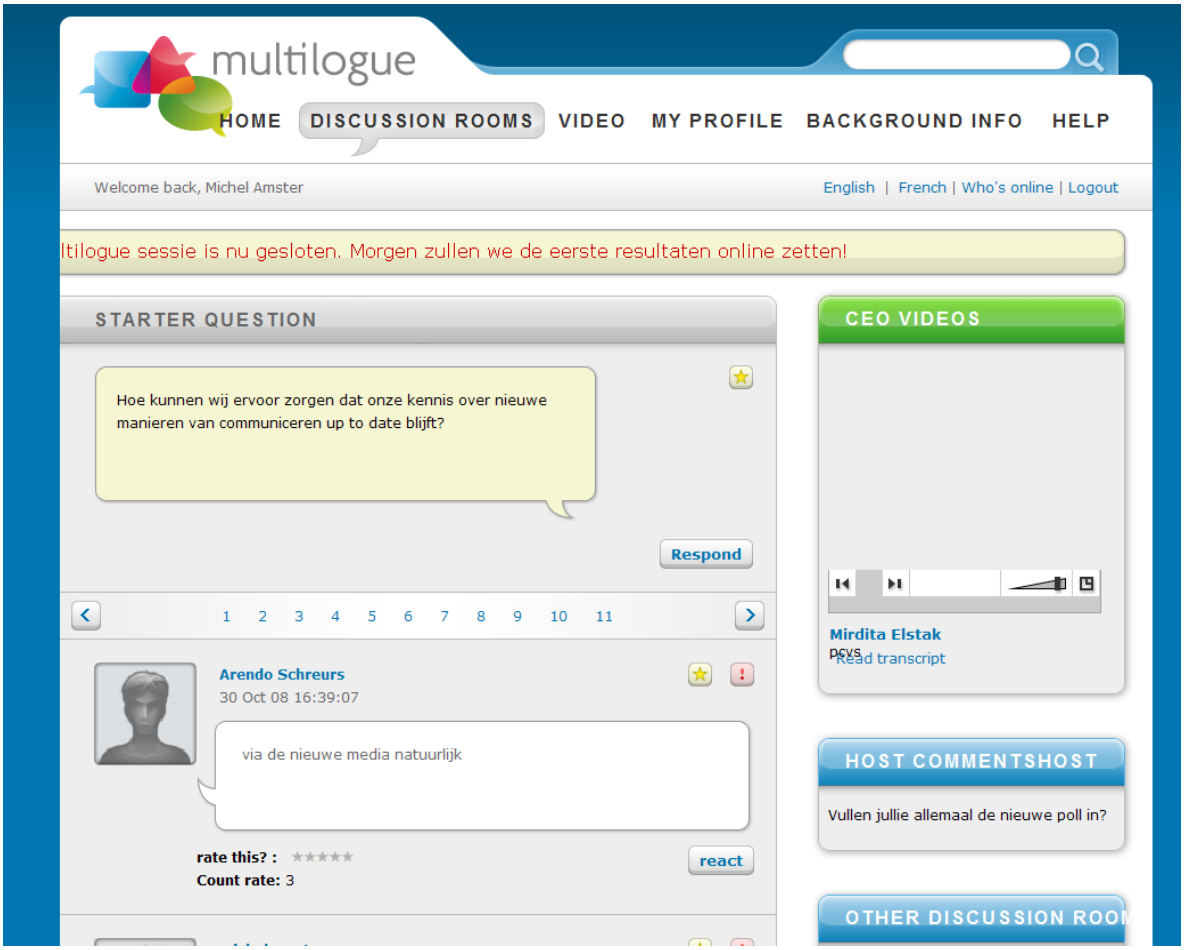
- Gescheiden rechten van verschillende typen gebruikers (zie use cases voor toegestane handelingen)
- Wachtwoorden zijn “ge-encrypt” opgeslagen in de database.

5 Mockups

Hoofdstuk 5, Mock-ups, geeft een aantal lay-out voorbeelden hoe de te realiseren applicatie er visueel uit gaat zien.
De (grafische) stijl die de huidige Multilogue nu heeft hoeft niet aangepast te worden.
Daardoor kan de stijl één op één worden overgenomen.
Dit hoofdstuk geeft niet elke pagina weer, maar is meer bedoeld om een idee te krijgen hoe het eruit gaat zien.

5.1 Front-end

Afbeelding 1/bijlage V geeft de lay-out weer als een gebruiker een discussiekamer ingaat.
Het gedeelte van de pagina waar onder andere “Home”en “Discussion rooms” staat blijft de gehele applicatie zichtbaar. Andere paginacontent wisselt afhankelijk van pagina.



Afbeelding 1 – Mockup discussiekamer.

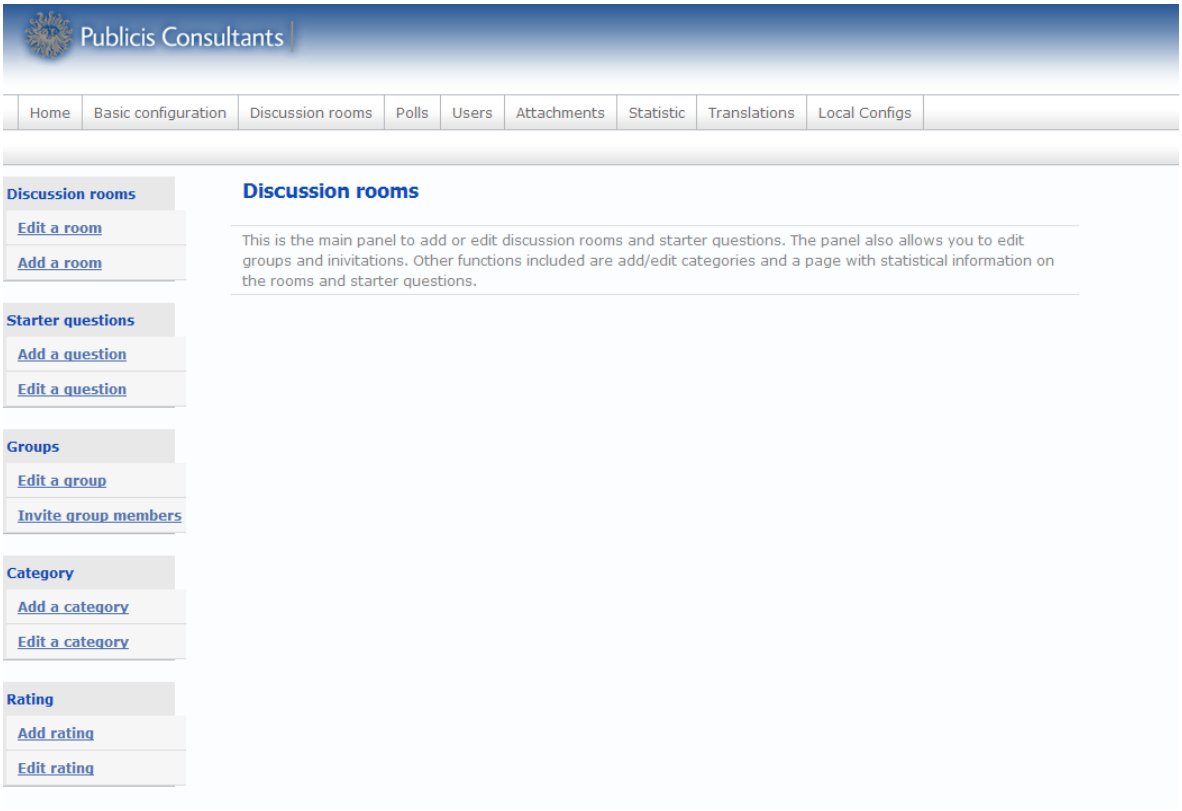
Afbeelding 2/bijlage VI geeft het profiel weer van de gebruiker met de daarbij horende informatie.

Afbeelding 2 – Mockup profiel.

De front-end moet aantrekkelijk zijn voor de gebruiker en een logische indeling hebben. Denk hierbij aan knoppen en dergelijk.

5.2 Editor back-end

Afbeelding 3/ bijlage VII geeft grafisch de mogelijkheden weer die de editor heeft op tabblad "Discussion rooms".

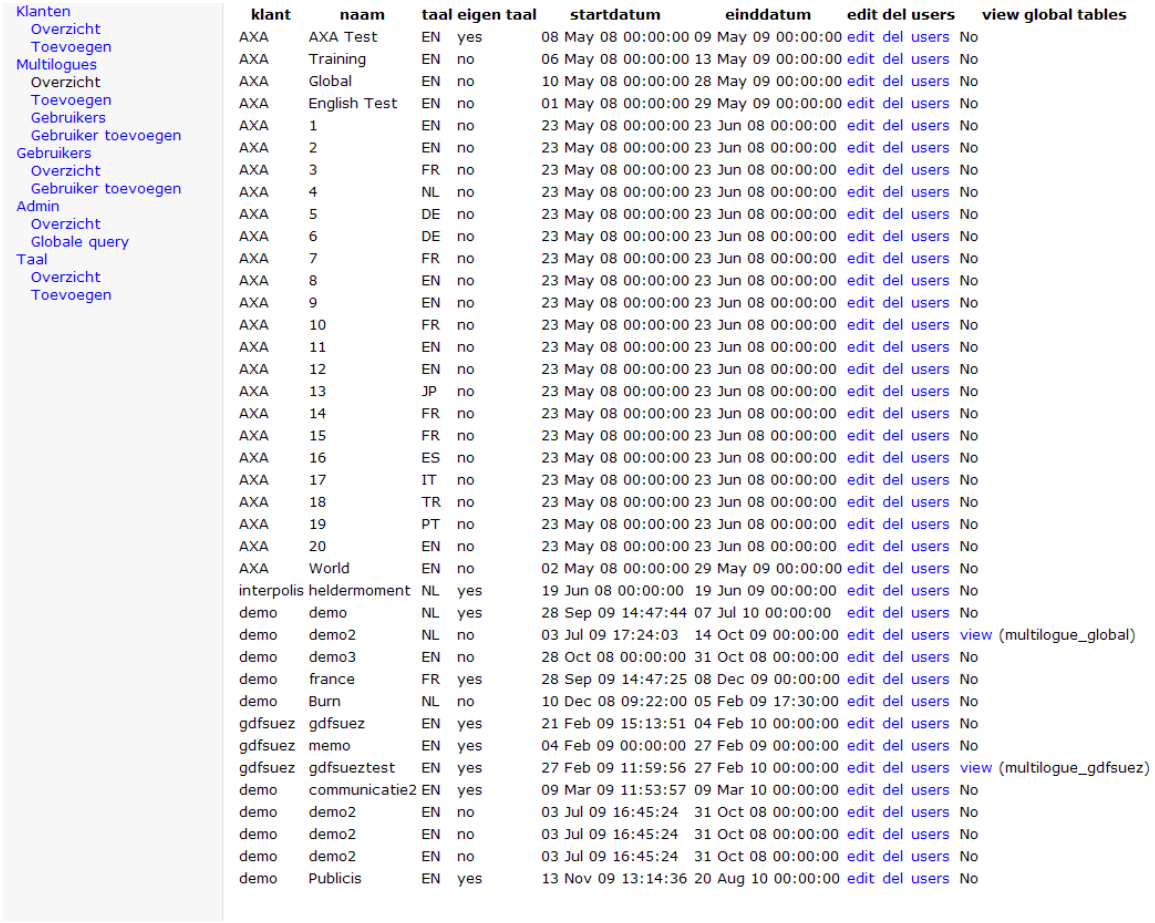


Afbeelding 3 – Mockup discussiekamer voor editor.

Bij de editor back-end is het belangrijk dat de editor eenvoudig en snel aanpassingen kan verrichten die van toepassing zijn op de front-end.

5.3 Administrator back-end

De back-end van de administrator moet functioneel werken. Aangezien de administrator een persoon uit E-mark is zijn er geen grafische hoogstandjes vereist. Afbeelding 4/ bijlage VIII geeft de back-end van de administrator weer waar een overzicht gegenereerd wordt van Multilogues.



Afbeelding 4 – Mockup administrator overzicht van Multilogues.

6 Workflow

Hoofdstuk 6, Workflow licht een aantal handelingen uit en geeft de workflows weer in eenvoudige sequentiediagrammen. Deze diagrammen zijn zo eenvoudig mogelijk gehouden om ook voor niet technische mensen het duidelijk te houden.

De workflows van de volgende processen zijn beschreven:

1) Registratieproces.
2) Aanmaken van een discussiekamer met vragen.
3) Reactie rapporteren.
4) Opzetten van Multilogue.

De [..] zijn voorwaardes waaraan voldaan moet zijn waar de actie uitgevoerd wordt. Voorbeeld:

[voorwaarde1]: voer actie uit

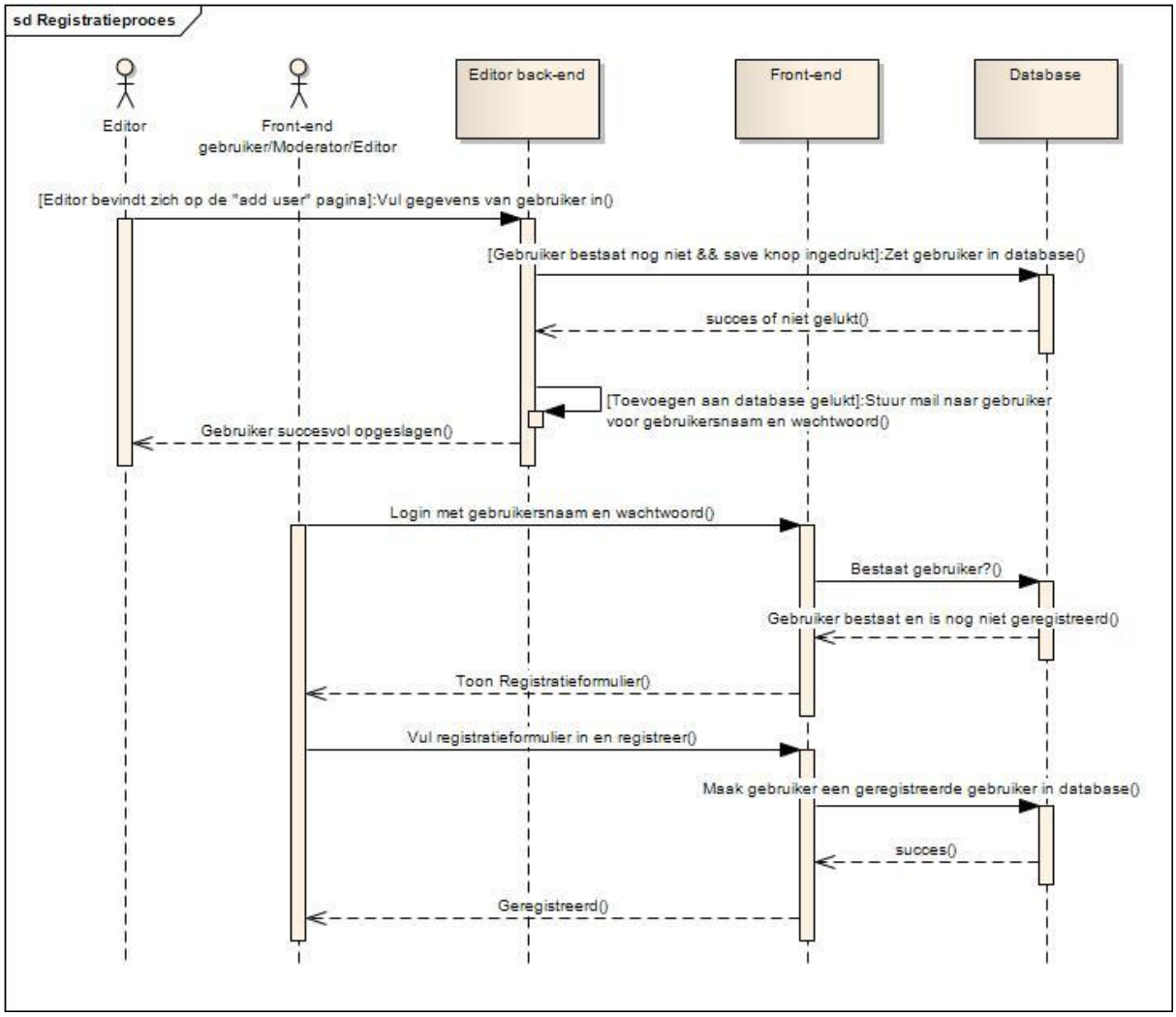
Stappen zoals inloggen worden achterwege gelaten vanwege duidelijkheid. Activiteiten zoals inloggen en andere logische stappen zijn al gedaan als dit benodigd was. De workflow die hier beschreven is, is identiek aan de workflow van de bestaande Multilogue. Deze workflow kan enigszins nog veranderen. Indien dit het geval is wordt dit opgenomen in het technisch ontwerp.

6.1 Registratieproces

Afbeelding 5/ bijlage IX beschrijft het registratieproces als één persoon wordt toegevoegd door een editor.

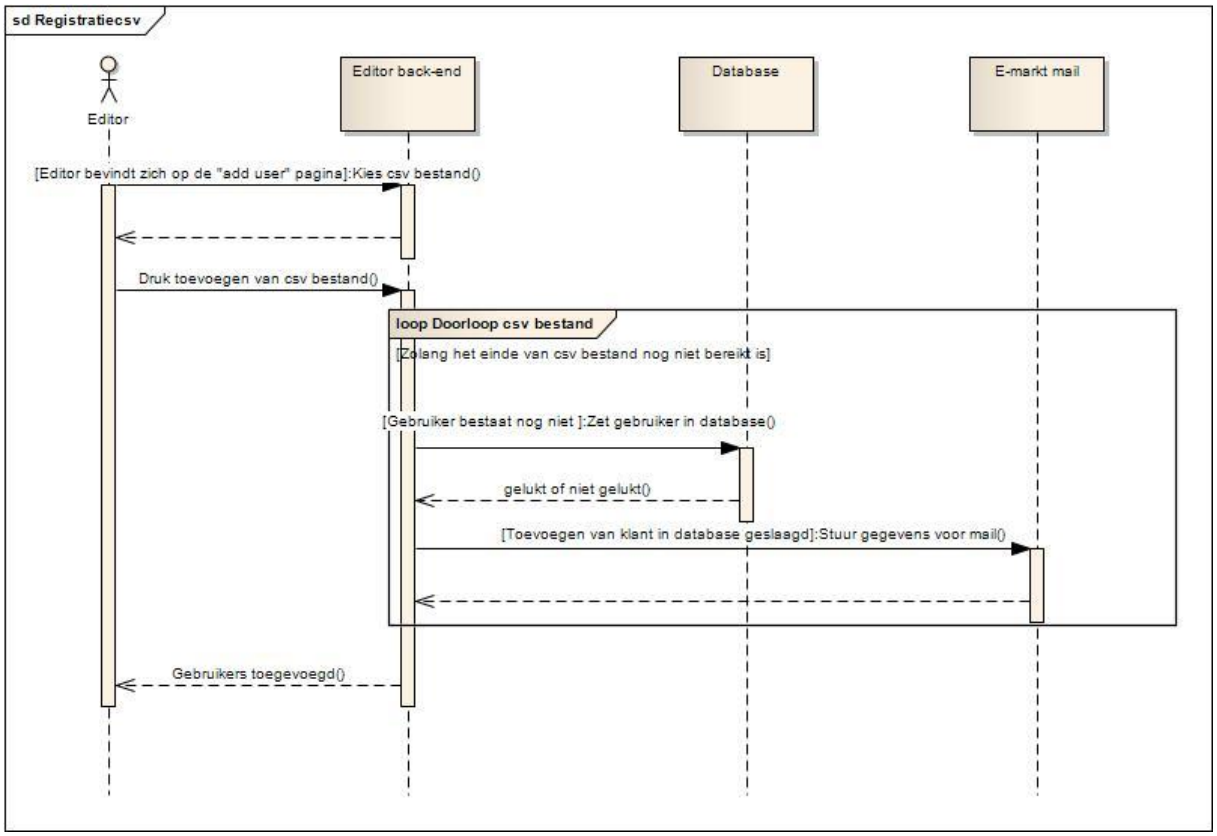
Vanuit gaand dat de editor al ingelogd is in het systeem. De editor gaat naar de daarvoor bedoelde pagina om een gebruiker aan te maken. De benodigde gegevens worden ingevoerd door de editor. Nadat de editor op “save” drukt wordt er gekeken of de gebruiker al bestaat in de database. Indien dit niet het geval is, wordt de gebruiker toegevoegd aan de database. Vanuit de back-end wordt een mail verstuurd naar de gebruiker om de gebruikersnaam en wachtwoord te mailen.

Zodra een front-end gebruiker/moderator/editor zich aanmeld in de front-end wordt er gekeken of deze gebruiker bestaat in de database. Indien deze bestaat en deze gebruiker nog niet geregistreerd is wordt het registratieformulier getoond. Na het succesvol invullen van het registratieformulier is de gebruiker geregistreerd.



Afbeelding 5 – Registratieproces.

Als er gebruikers worden toegevoegd door middel van een csv bestand geschiedt het proces anders. Enkel het opslaan van de gebruikers wordt getoond in afbeelding 6/ bijlage X. Het proces voor de front-end gebruiker/moderator/editor die zich moet registreren blijft identiek aan het proces bij het toevoegen van één gebruiker en wordt niet beschreven in dit model.



Afbeelding 6 - Registratieproces door middel van csv lijst.

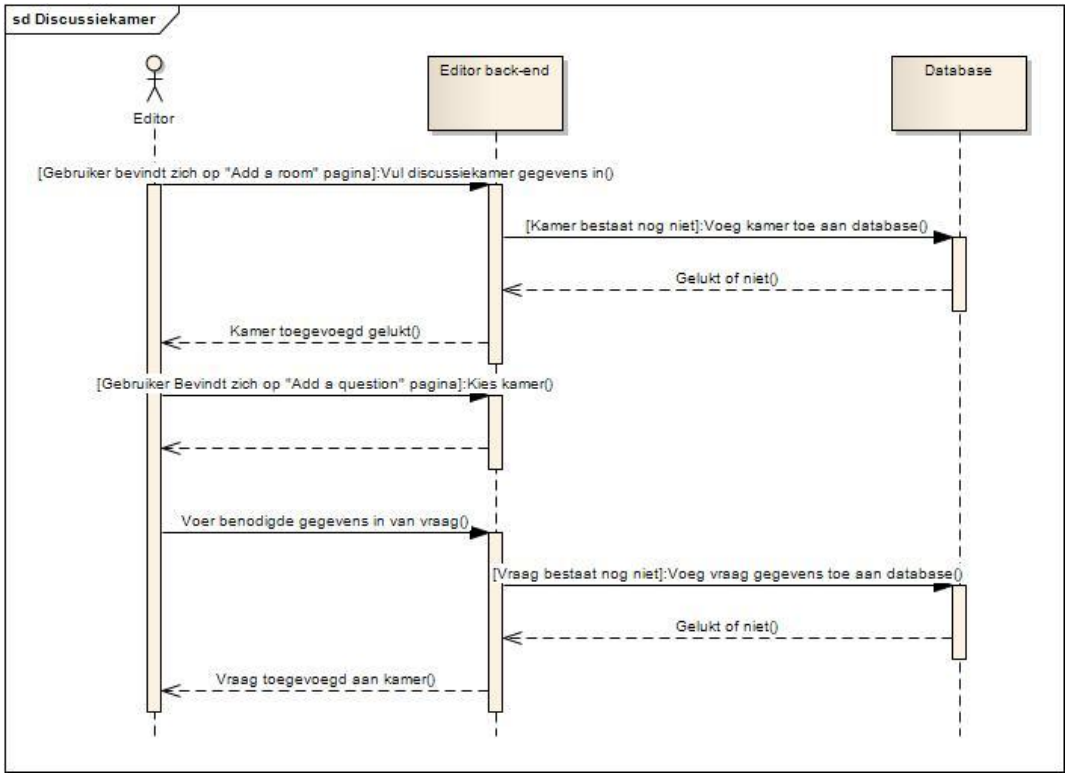
De editor kiest een csv bestand en drukt op toevoegen. De applicatie leest het csv bestand uit waar voornaam, achternaam en e-mail in opgenomen zijn. Vervolgens wordt er zolang het einde van het csv bestand niet bereikt is gebruikers toegevoegd aan de database. Dit met voorwaarde dat de gebruiker niet bestaat. Als het opslaan succesvol verlopen is wordt er met E-markt mail een mail verstuurd naar de desbetreffende persoon. Dit geschiedt door middel van SOAP(Simple Object Acces Protocol).

6.2 Aanmaken van een discussiekamer met vragen

Aanname is dat er een nieuwe kamer aangemaakt moet worden met daarin een vraag.
Eerste stap voor de editor is om de juiste pagina te selecteren en daar de gegevens in te voeren voor de discussiekamer. Deze gegevens zijn: titel, taal, start-/einddatum (dus datum actief), omschrijving en commentaar.

Om een vraag toe te voegen moet de gebruiker naar de "add a question" pagina toegaan. Hierbij selecteert de editor de kamer waar de vraag moet in komen te staan. Hierbij vult hij gegevens in zoals de titel, omschrijving en keywords van de vraag.
Nadat deze stap voltooid is de vraag en kamer te zien aan de front-end van de applicatie.

Zie afbeelding 7/bijlage XI.



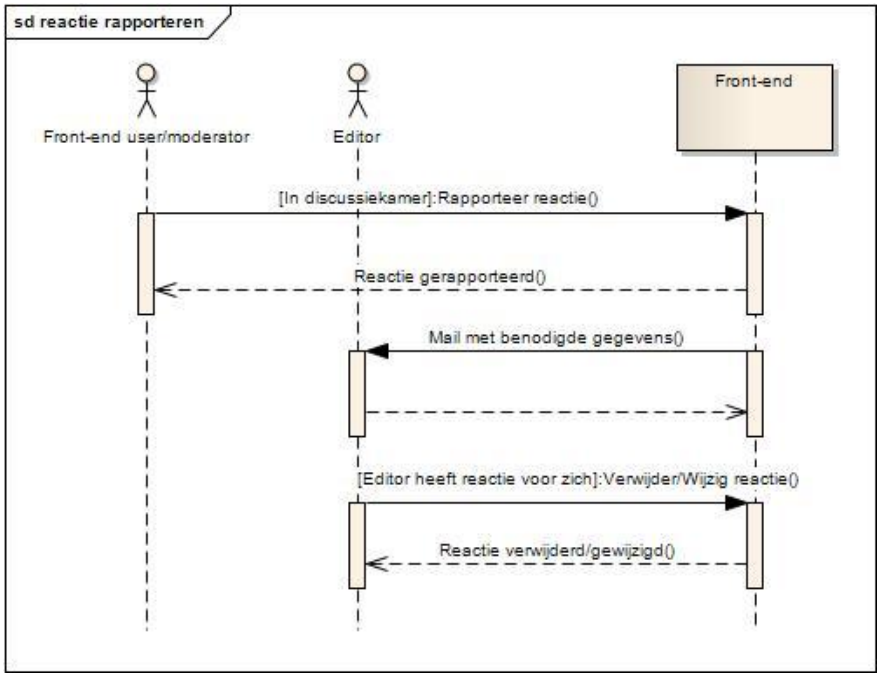
Afbeelding 7 – Discussiekamer toevoegen met vraag.

6.3 Reactie rapporteren

Bij het rapporteren van een reactie gaat er een mail naar de editor (afbeelding 8/ bijlage XII).

Zodra een moderator of een normale gebruiker een reactie rapporteert gaat er een mail naar de editor. Mailadressen worden in de applicatie ingesteld. De front-end gebruiker kan de reden intypen waarom deze reactie gerapporteerd wordt. In de mail bevindt zich de reden van rapportage en de url link die de editor naar de desbetreffende reactie brengt.

De editor kan de reactie lezen en besluiten om het te verwijderen of te wijzigen.



Afbeelding 8 – Reactie rapporteren.

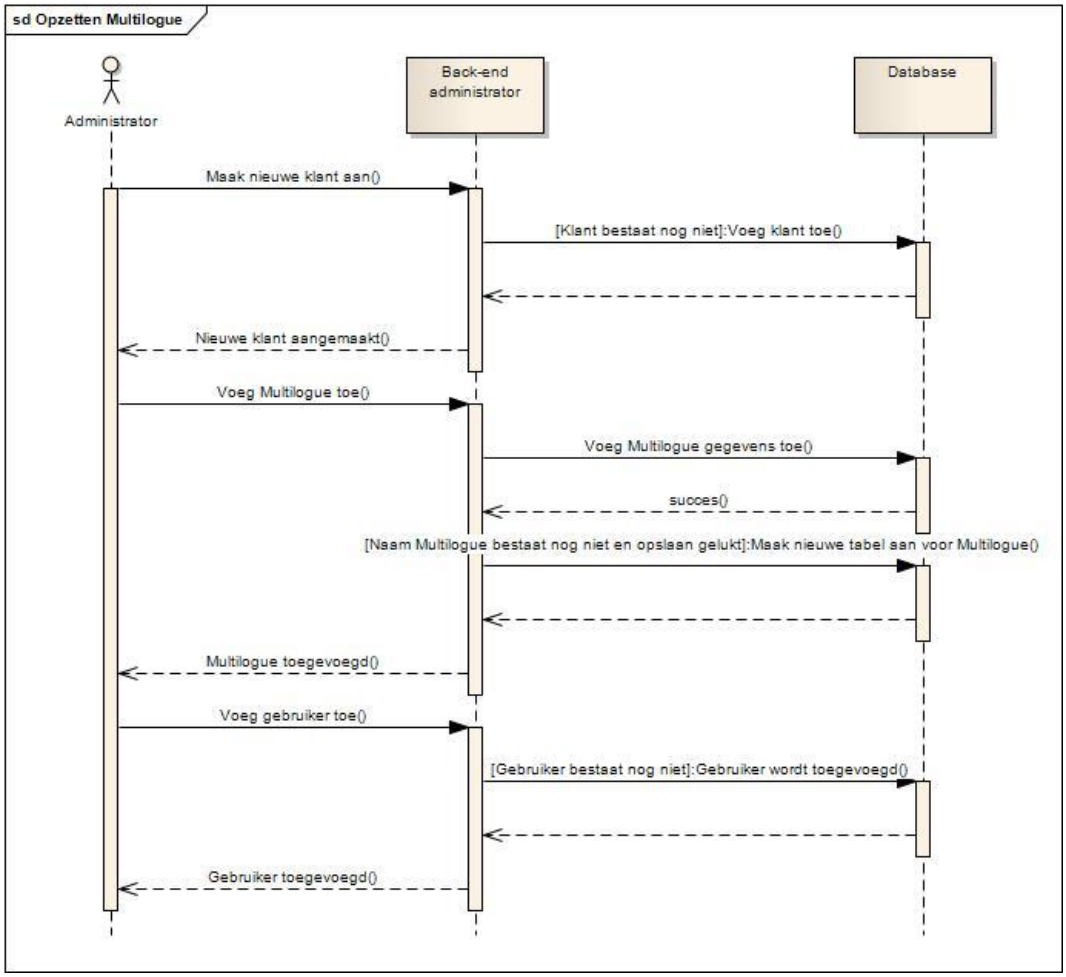
6.4 Opzetten van Multilogue

Het opzetten van de Multilogue wordt gedaan door de administrator. Zie afbeelding 9/bijlage XIII).

De administrator maakt een nieuwe klant indien dat nodig is. De klant is het bedrijf waar de Multilogue voor opgezet moet worden. Hierbij is de naam van de klant vereist. Bij het toevoegen van de Multilogue zijn onder andere Multiloguenaam, startdatum/eindatum, inlogmethode, klant, taal en databasenaam benodigd. Deze gegevens worden opgeslagen in een bestaande tabel. Indien dit succesvol verlopen is wordt er een nieuwe tabel aangemaakt voor de Multilogue.

Een zijsprong naar het toevoegen van de Multilogue, hierbij is het opgeven van een taal vereist. Indien de gewenste taal niet bestaat kan deze aangemaakt worden door de administrator. Het diagram behandelt deze stap niet.

Wat rest voor de administrator is het toevoegen van een gebruiker voor de Multilogue. Dit kan zowel een editor als een normale gebruiker zijn. Als een editor gekozen wordt dan kan de editor het toevoegen van andere gebruikers op zich nemen.



Afbeelding 9 – Opzetten Multilogue.

7 Projectmanagement

Hoofdstuk 7, Projectmanagement, beschrijft hoe de planning en risico's er nu voor staan ten opzichte van het plan van aanpak.

De risico's die in het plan van aanpak zijn hoeven niet aangepast te worden.
De planning zoals die opgesteld was in het plan van aanpak is aangepast doordat de analysefase minder lang heeft geduurd dan aanvankelijk gepland werd. Ook het functioneel ontwerp is sneller in elkaar gezet dan de planning in het plan van aanpak aangeeft.

De planning zoals die in week 1 is opgesteld is in het plan van aanpak te zien.

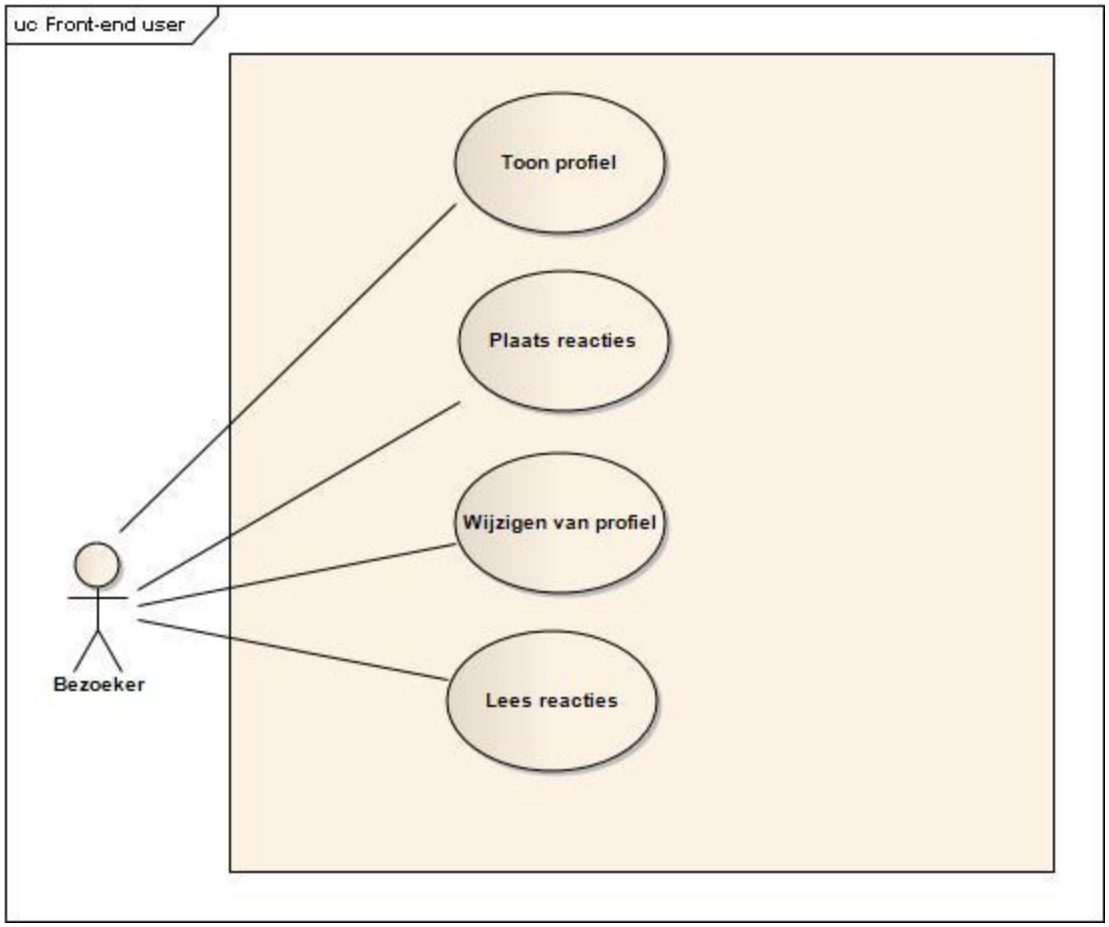
De huidige planning en gedane activiteiten. Het cursieve gedeelte is gerealiseerd.

Datum	**Activiteit/ stand van zaken**
8 februari / 9 februari	*Start plan van aanpak.*
10 februari	*Feedback pva verwerkt. Plan van aanpak af versie twee af.*
10 februari – 12 februari	*Start analyse.*
12 februari	*Analysedocument eerste versie af.*
15 februari	*Feedback analysedocument. Analysedocument versie twee af.*
17 februari	*Start Functioneel ontwerp.*
19 februari	*Plan van aanpak versie drie (feedback schoolbegeleider) en analysedocument versie drie opgestuurd naar schoolbegeleider*
23 februari **(huidige datum)**	Eerste versie functioneel ontwerp af.
24 februari	Feedback functioneel ontwerp.
2 maart	Functioneel ontwerp afgerond.
2 maart tot en met 9 maart	Uitloop functioneel ontwerp.
10 maart	Start technisch ontwerp.
26 maart	Technisch ontwerp eerste versie af.
31 maart	Feedback technisch ontwerp
10 april	Technisch ontwerp afgerond
10 april tot en met 20 april	Uitloop technisch ontwerp
20 april en verder	Programmeren in iteraties met terugkoppeling naar ontwerp en meer analyse
4 juni	Afstudeerverslag inleveren

Overige data vanuit school die van belang zijn blijven staan. Enkel de verwachte bedrijfsbezoek van de schoolbegeleider vindt niet plaats in de week van 22 februari. Dit vanwege vakantie op school.

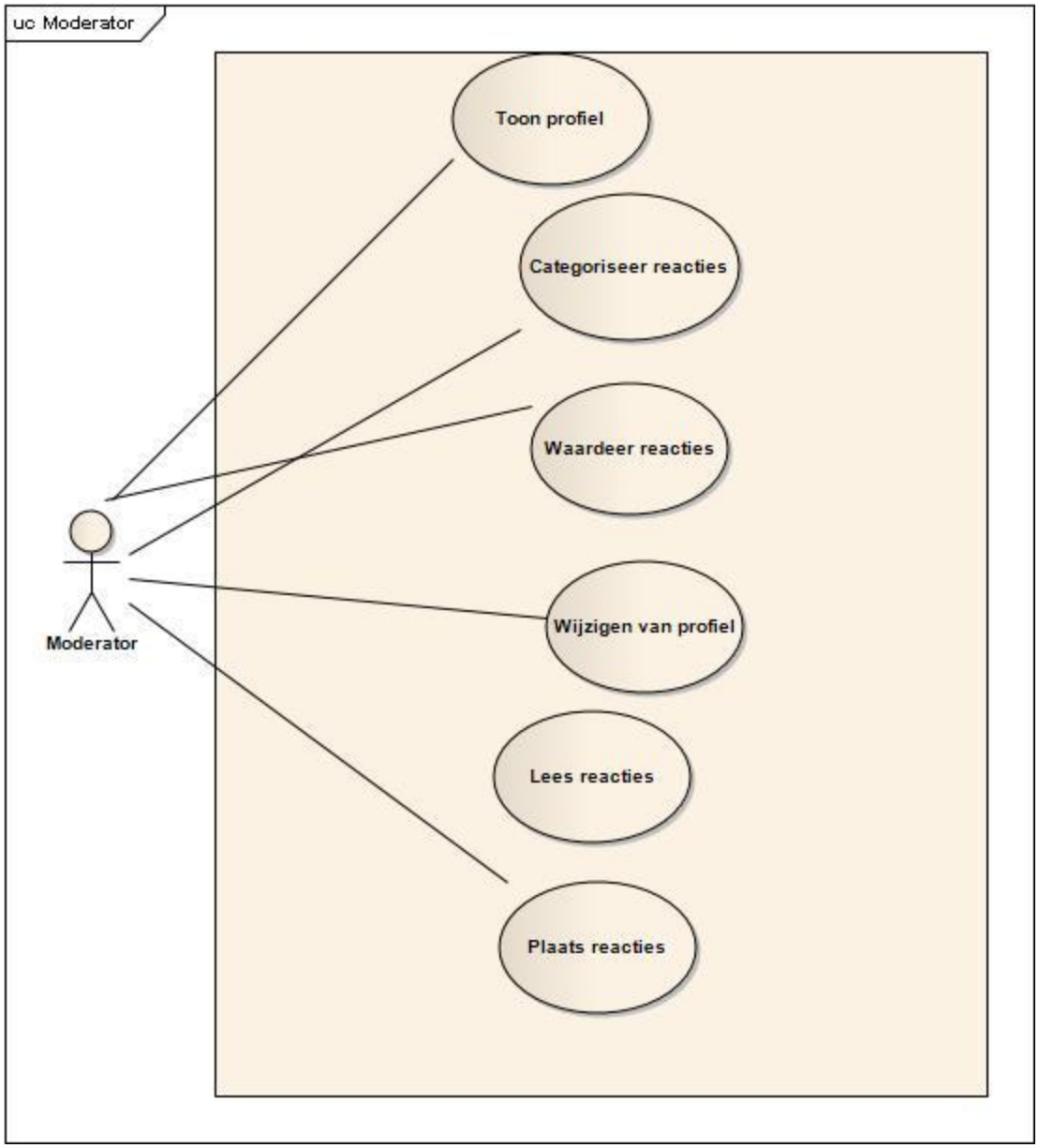
Datum	**Activiteit**
1 maart tot en met 5 maart	Bedrijfsbezoek schoolbegeleider
22 maart tot en met 2 april	Tussentijds verslag inleveren
5 april tot en met 16 april	Bespreken concept afstudeerdossier
10 tot en met 21 mei	Tussentijds assesment

Bijlage I – Normale gebruiker use case diagram



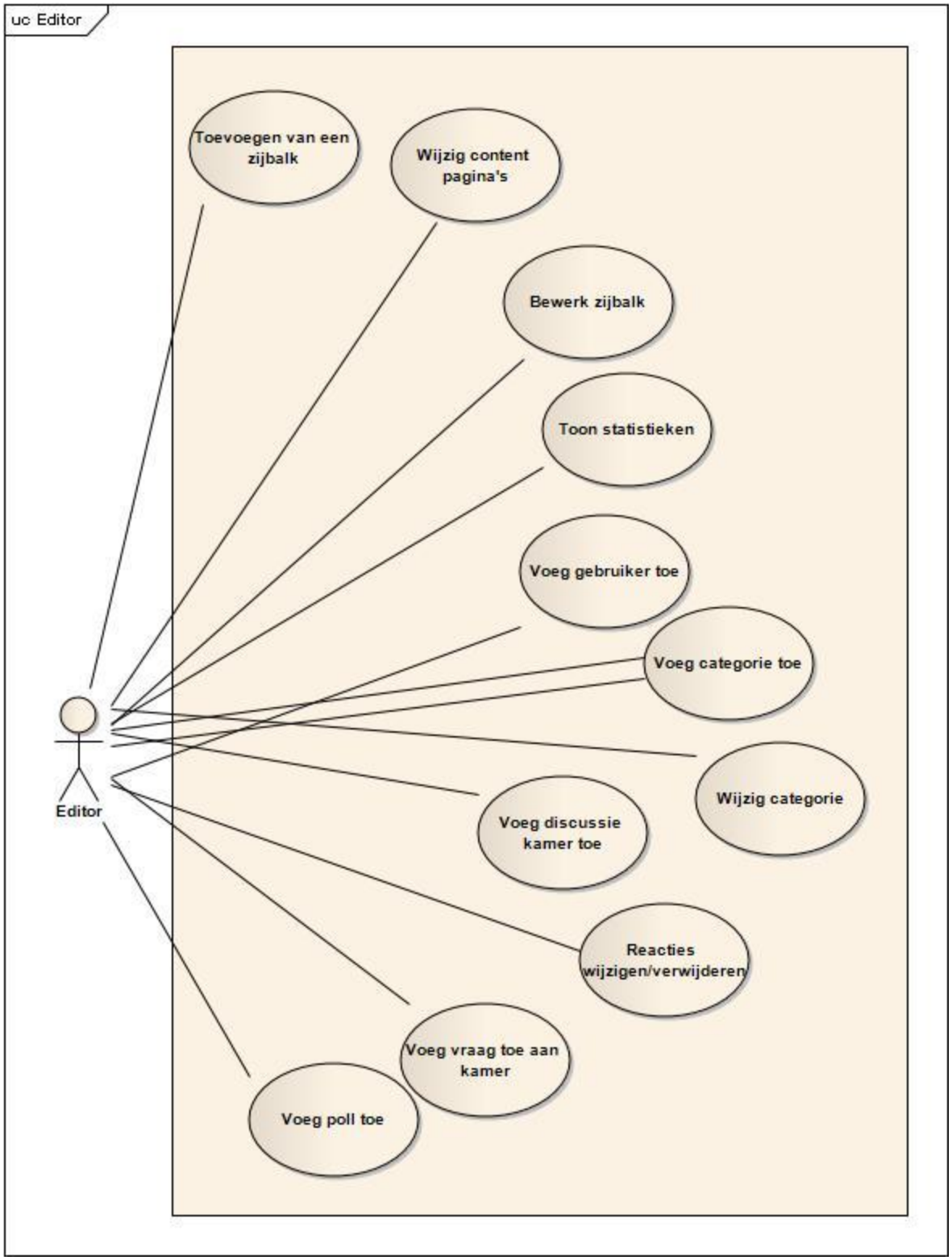
"front-end" gebruiker mogelijkheden.

Bijlage II – Moderator use case diagram



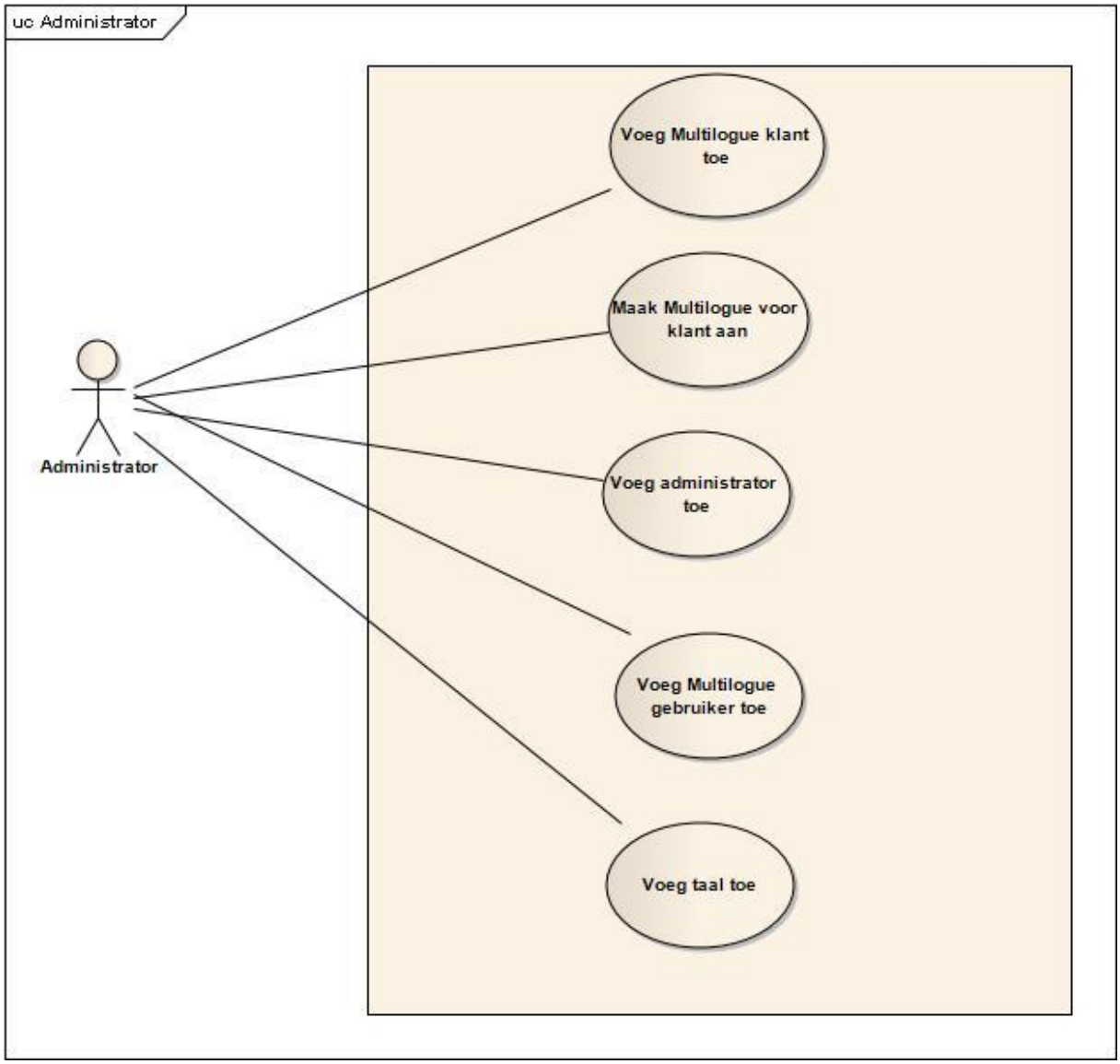
Moderator functies in de "front-end".

Bijlage III – Editor use case diagram



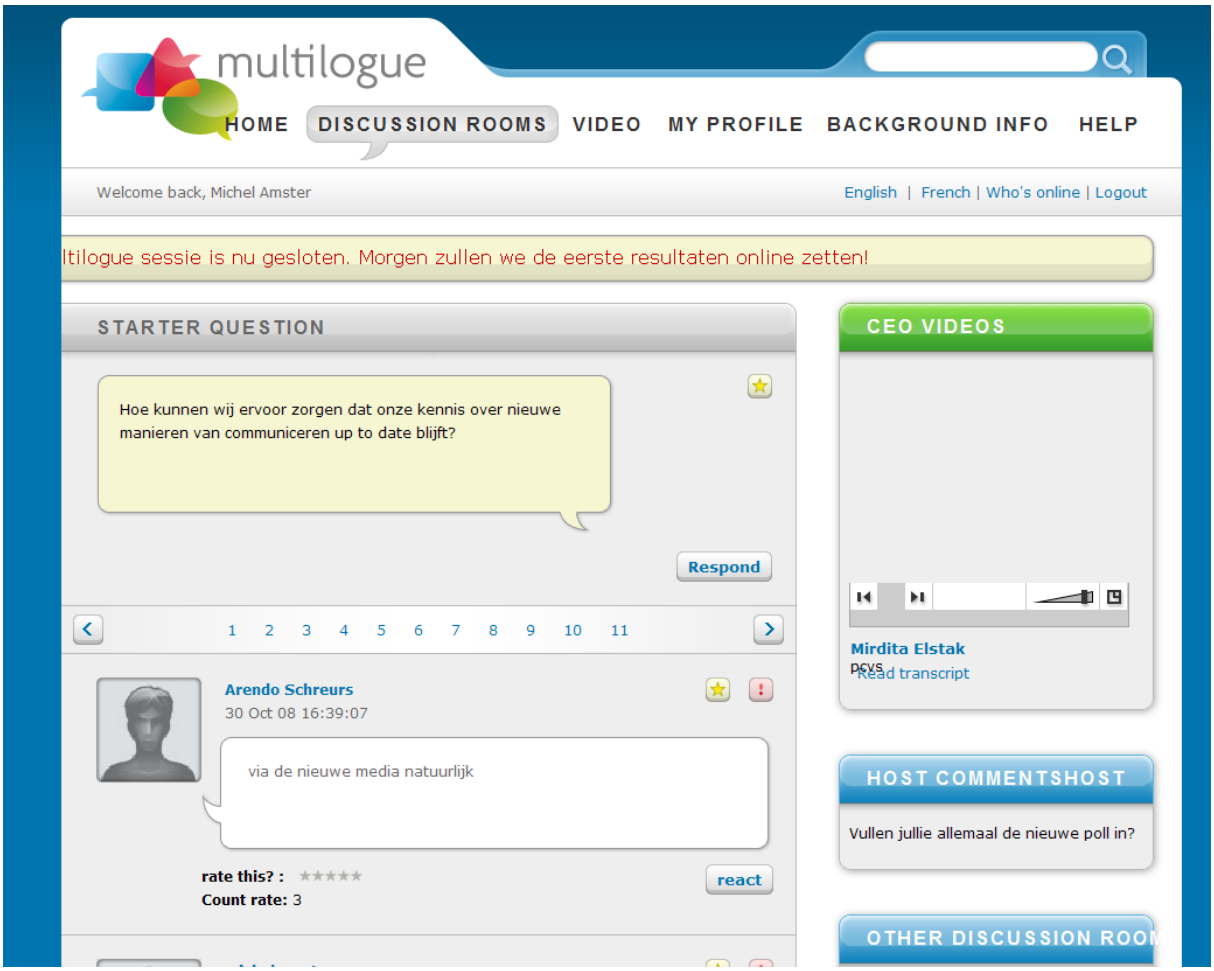
Editor functies in de back-end.

Bijlage IV – Administrator use case diagram



Administrator functies voor back-end.

Bijlage V – Mockup front-end discussiekamer

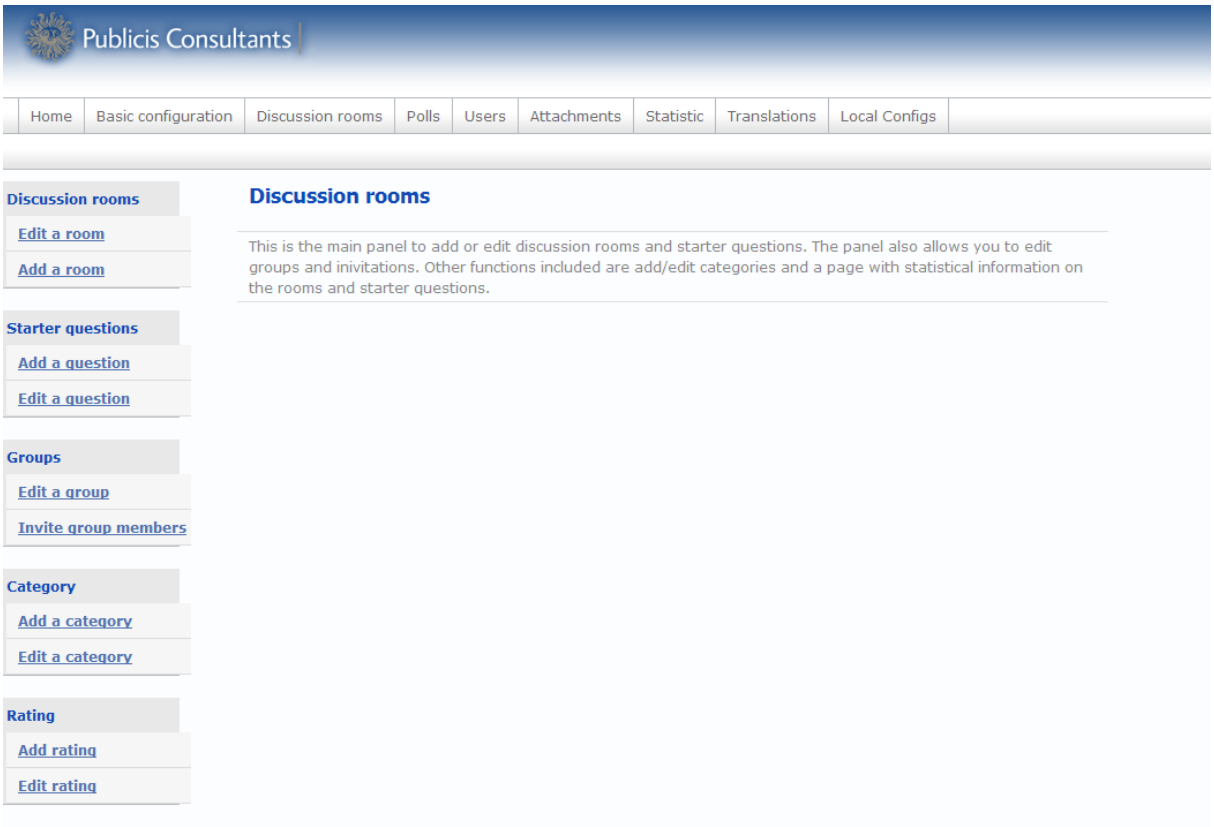


Mockup front-end discussiekamer.

Bijlage VI – Mockup front-end profiel

Mockup front-end profiel.

Bijlage VII – Mockup back-end editor



Mockup back-end editor discussie kamer mogelijkheden.

Bijlage VIII – Mockup back-end administrator

Klanten
- Overzicht
- Toevoegen

Multilogues
- Overzicht
- Toevoegen
- Gebruikers
- Gebruiker toevoegen

Gebruikers
- Overzicht
- Gebruiker toevoegen

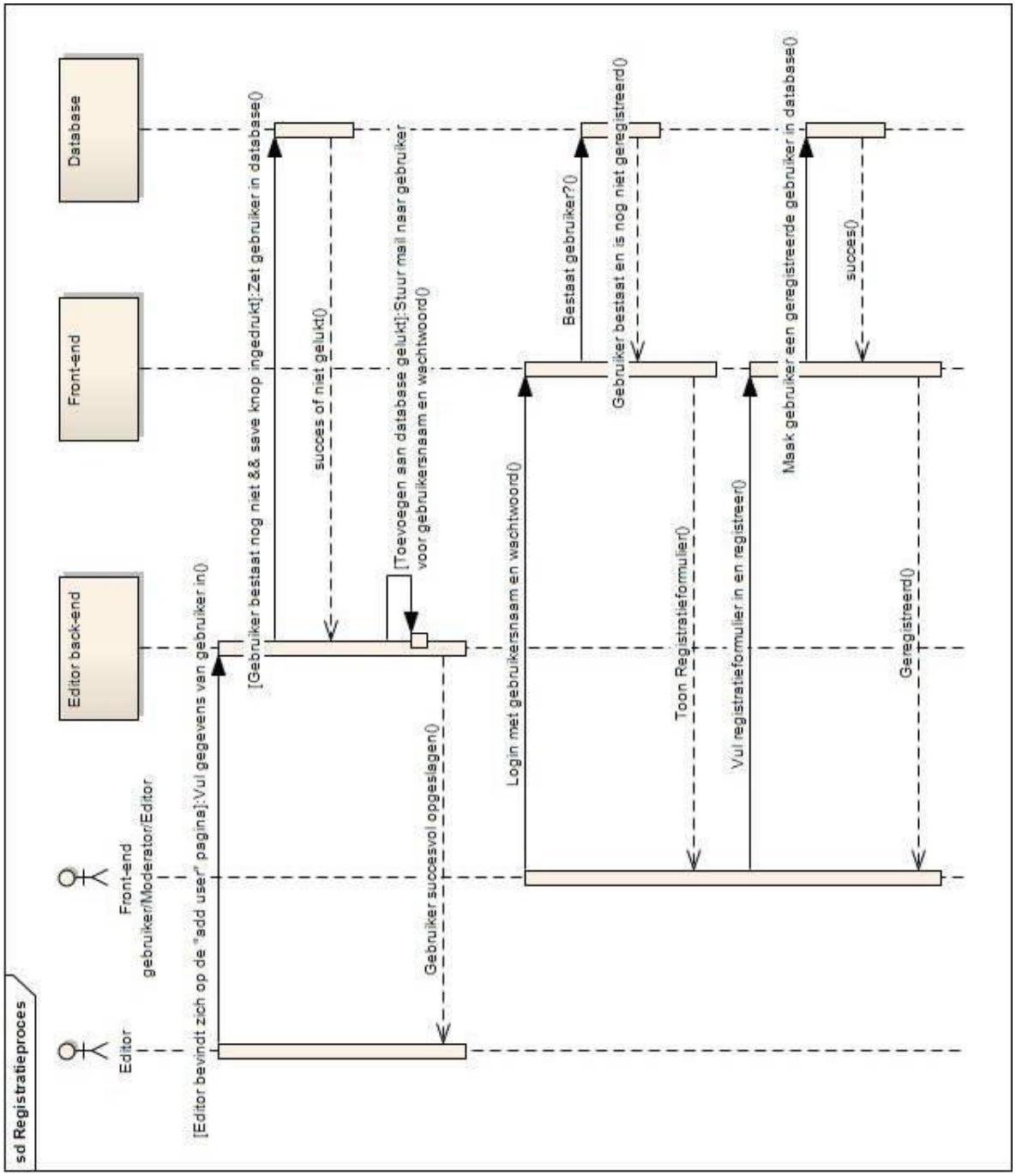
Admin
- Overzicht
- Globale query

Taal
- Overzicht
- Toevoegen

klant	naam	taal	eigen taal	startdatum	einddatum	edit	del	users	view global tables
AXA	AXA Test	EN	yes	08 May 08 00:00:00	09 May 09 00:00:00	edit	del	users	No
AXA	Training	EN	no	06 May 08 00:00:00	13 May 09 00:00:00	edit	del	users	No
AXA	Global	EN	no	10 May 08 00:00:00	28 May 09 00:00:00	edit	del	users	No
AXA	English Test	EN	no	01 May 08 00:00:00	29 May 09 00:00:00	edit	del	users	No
AXA	1	EN	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	2	EN	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	3	FR	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	4	NL	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	5	DE	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	6	DE	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	7	FR	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	8	EN	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	9	EN	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	10	FR	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	11	EN	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	12	EN	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	13	JP	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	14	FR	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	15	FR	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	16	ES	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	17	IT	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	18	TR	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	19	PT	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	20	EN	no	23 May 08 00:00:00	23 Jun 08 00:00:00	edit	del	users	No
AXA	World	EN	no	02 May 08 00:00:00	29 May 09 00:00:00	edit	del	users	No
interpolis	heldermoment	NL	yes	19 Jun 08 00:00:00	19 Jun 09 00:00:00	edit	del	users	No
demo	demo	NL	yes	28 Sep 09 14:47:44	07 Jul 10 00:00:00	edit	del	users	No
demo	demo2	NL	no	03 Jul 09 17:24:03	14 Oct 09 00:00:00	edit	del	users	view (multilogue_global)
demo	demo3	EN	no	28 Oct 08 00:00:00	31 Oct 08 00:00:00	edit	del	users	No
demo	france	FR	yes	28 Sep 09 14:47:25	08 Dec 09 00:00:00	edit	del	users	No
demo	Burn	NL	no	10 Dec 08 09:22:00	05 Feb 09 17:30:00	edit	del	users	No
gdfsuez	gdfsuez	EN	yes	21 Feb 09 15:13:51	04 Feb 10 00:00:00	edit	del	users	No
gdfsuez	memo	EN	yes	04 Feb 09 00:00:00	27 Feb 09 00:00:00	edit	del	users	No
gdfsuez	gdfsueztest	EN	yes	27 Feb 09 11:59:56	27 Feb 10 00:00:00	edit	del	users	view (multilogue_gdfsuez)
demo	communicatie2	EN	yes	09 Mar 09 11:53:57	09 Mar 10 00:00:00	edit	del	users	No
demo	demo2	EN	no	03 Jul 09 16:45:24	31 Oct 08 00:00:00	edit	del	users	No
demo	demo2	EN	no	03 Jul 09 16:45:24	31 Oct 08 00:00:00	edit	del	users	No
demo	demo2	EN	no	03 Jul 09 16:45:24	31 Oct 08 00:00:00	edit	del	users	No
demo	Publicis	EN	yes	13 Nov 09 13:14:36	20 Aug 10 00:00:00	edit	del	users	No

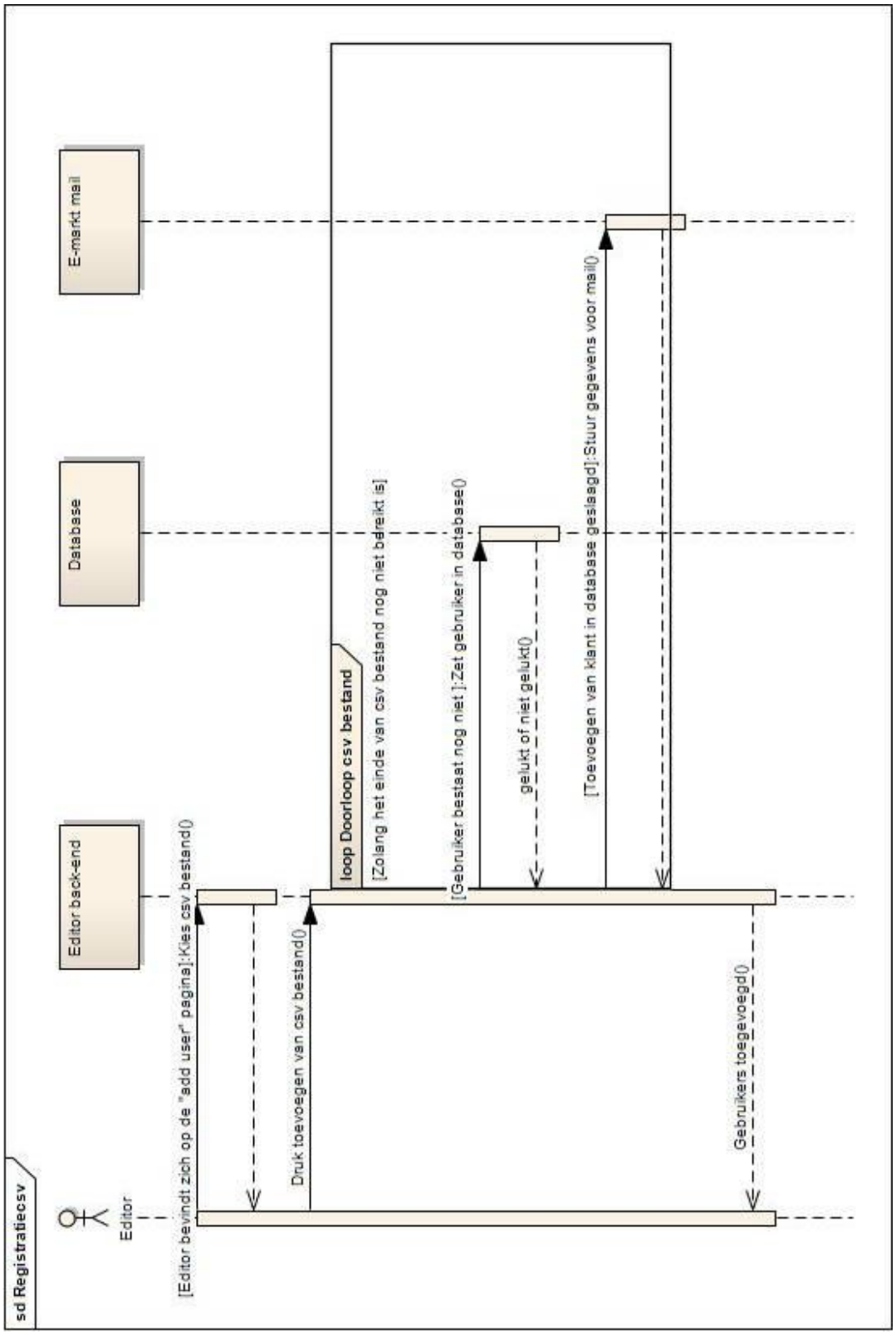
Mockup back-end administrator overzich Multilogues.

Bijlage IX – Registratieproces één persoon



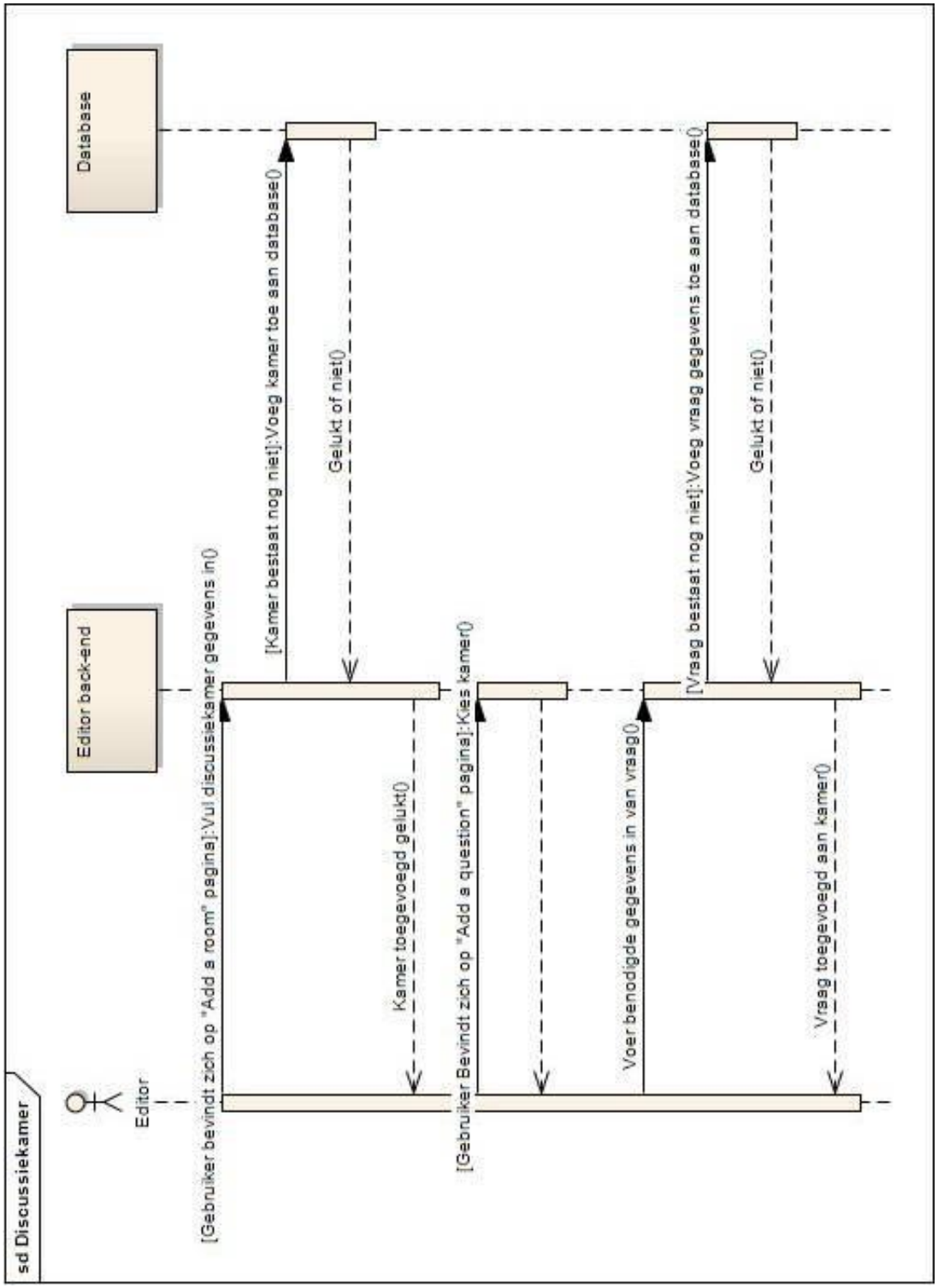
Registratieproces één persoon.

Bijlage X – Registratieproces met csv lijst



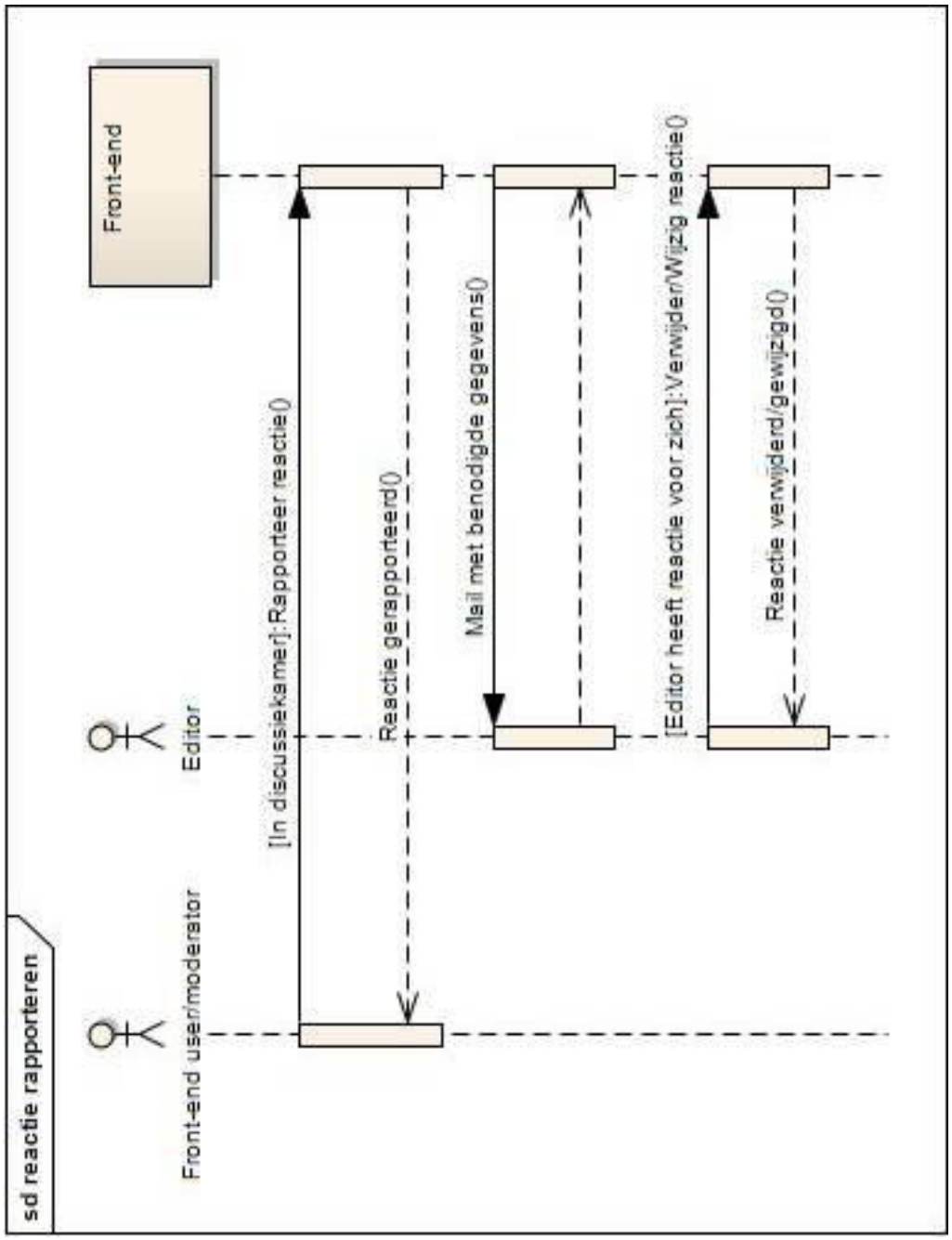
Registratieproces door middel van een csv bestand.

Bijlage XI – Discussiekamer met vraag



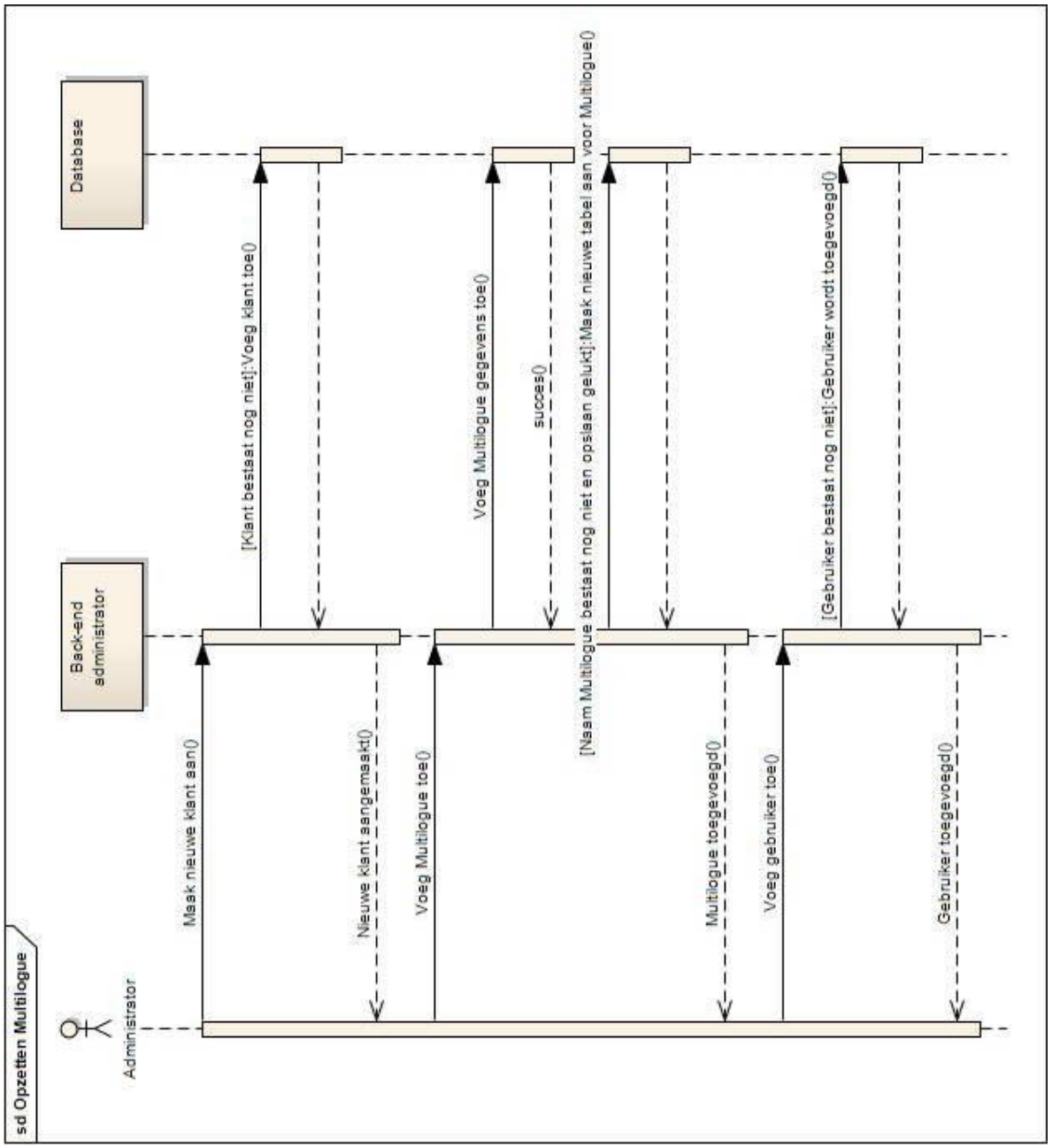
Een discussiekamer toevoegen met een vraag

Bijlage XII – Reactie rapporteren



Reactie rapporteren

Bijlage XIII – Opzetten Multilogue



Opzetten Multilogue door administrator

Technisch ontwerp

26 maart 2010

Versie 2

Michel Dickhoff

Inhoudsopgave

Versiebeheer

Op 23 maart nagekeken door Boudewijn Somer:

- Op gebied van termen minder uitleg gegeven. Het TO is immers bedoeld voor technici.

Op 26 maart nagekeken door Nils Corver:

- Namen van acties zijn korter gemaakt voor duidelijkheid.
- Argumentatie MVC ASP.NET aangepast.

Inleiding

Het document “technisch ontwerp” is geschreven zodat E-mark de applicatie verder kan ontwikkelen nadat de geplande iteraties volbracht zijn door de afstudeerder. Doordat het document geschreven is voor E-mark zijn bepaalde aspecten minder uitgebreid beschreven, omdat E-mark de huidige Multilogue gebouwd heeft en het ontwerpen van de nieuwe Multilogue in overleg is gegaan.

Dit document verwijst naar de functionele eisen uit het functioneel ontwerp. De functionele eisen zijn kort opgesomd in hoofdstuk 1 “Eisenlijst”.

De eerste stap van de applicatie die ontwikkeld gaat worden is het ontwerpen van de database. Dit ontwerp is toegelicht in hoofdstuk 2.

Omdat het een webapplicatie betreft is de architectuur van de applicatie weergeven. Met de architectuur wordt bedoeld de pagina’s van de applicatie die de gebruiker kan bezoeken. Hoofdstuk 3 geeft de architectuur (sitemap) van de applicatie weer.

Vanuit de sitemap zijn er klassen opgesteld in combinatie met het maken van sequentiediagrammen. De klassen zijn ontworpen aan de hand van de functionaliteit die de applicatie moet gaan bieden. De sequentiediagrammen beschrijven de interactie tussen de verschillende objecten.
Hoofdstuk 4 beschrijft de klassen en hoofdstuk 5 beschrijft de interactie tussen objecten.

Hoofdstuk 6 deelt de applicatie op in lagen voor duidelijkheid en het eventueel scheiden van taken.
Keuzes die gemaakt zijn, zijn beargumenteerd in hoofdstuk 7.

1 Eisenlijst

Dit hoofdstuk beschrijft de eisen zoals die opgesteld zijn in het functioneel ontwerp met kleine aanpassingen. Voor toelichting voor de eisen wordt verwezen naar het functioneel ontwerp.

De volgende veranderingen hebben plaats gevonden ten opzichte van het functioneel ontwerp:

- In gebruiker en editor eis voltooi registratie verwijderd. In feit is dit alleen het statusbit wijzigen in de User tabel. Dit onderdeel wordt gezien als het wijzigen van het profiel
- Het uitvoeren van queries in het administrator gedeelte zijn weggehaald. Bij het uitvoeren van een foutieve query kan de database geheel onbruikbaar worden.
- Toon Multilogue gebruikers per bedrijf is van administrator naar editor gegaan.

Gebruiker:

1) Toon profiel
2) Plaats reacties
3) Wijzigen van profiel
4) Inloggen
5) Uitloggen
6) Voeg stem toe aan poll
7) Kies taal
8) Toon poll
9) Toon “wie is online”
10) Voeg reacties/vragen toe aan favorietenlijst
11) Rapporteer reacties
12) Bekijk video
13) Bekijk laatst geschreven/gelezen reactie
14) Bezoeken van kamers en lezen van reacties

Moderator:

15) Categoriseer reacties
16) Waardeer reacties

Editor:

17) Toevoegen van een zijbalk
18) Bewerk zijbalk
19) Wijzig content van pagina’s
20) Toon statistieken
21) Voeg categorie toe
22) Wijzig categorie
23) Voeg gebruiker toe
24) Voeg discussiekamer toe
25) Voeg vraag toe aan kamer
26) Reacties verwijderen/wijzigen
27) Voeg poll toe

28) Inloggen
29) Uitloggen
30) Toon vertalingen
31) Voeg formvelden toe
32) Toon formvelden
33) Voeg woorden toe aan blacklist
34) Zoek gebruiker
35) Instellen van infobericht
36) Toon polls
37) Volg gebruiker
38) Voeg Multilogue bericht toe
39) Voeg gebruiker toe aan groep
40) Maak groepen aan

Administrator:

41) Toon geïnstalleerde Multilogues
42) Voeg administrator toe
43) Voeg Multilogue gebruiker toe
44) Voeg taal toe
45) Maak Multilogue voor klant aan
46) Inloggen
47) Uitloggen
48) Toon overzicht Multilogue klanten
49) Toon administrators

2 Database-indeling

Naar aanleiding van het functioneel ontwerp zijn de gegevens vastgelegd die opgeslagen moeten worden voor de applicatie. Hiervoor zijn tabellen ingericht. Dit hoofdstuk geeft globale beschrijvingen van de database tabellen.
Voor de applicatie zijn er twee databases bedacht:

1) Database voor Multilogue waar voornamelijk de bezoeker, moderator en editor gebruik van maken.
2) Database voor administrator.

De reden voor deze scheiding is dat op deze manier het administratorgedeelte apart kan draaien en geïnstalleerd worden.

De twee databases zijn in de tweede normaalvorm gezet. Dit betekent dat er geen repeating groups aanwezig zijn en de niet-sleutel attributen functioneel afhankelijk zijn van de volledige sleutel. Er is niet verder genormaliseerd omdat dit de performance vermindert door het verder splitsen van tabellen.

2.1 Database Multilogue

Doordat de database voor de Multilogue een grote hoeveelheid tabellen heeft is deze onderverdeeld in een aantal groepen van tabellen.

Deze groepen zijn:

1) Reacties en gebruikers
2) Favorietenlijst en sessies
3) “Multilanguage”
4) Polls en gebruikers
5) Waarderen en categoriseren
6) Settings
7) “Formfield”

2.1.1 Reacties en gebruikers

De tabellen voor de groep “Reacties en gebruikers” slaat de gebruikers, gebruikersgroepen en reacties op.

In bijlage 1 zijn de tabellen weergegeven. Onderstaande tekst refereert hiernaar.

De tabel User bevat gegevens van gebruikers. Een gebruiker heeft een gebruikerstype. De huidige applicatie maakt onderscheid in de typen editor, moderator en de bezoeker. Deze gebruikerstypen zijn opgeslagen in Usertype (niet functionele eis “Gescheiden rechten van verschillende typen gebruikers”). Bij het aanmaken van een nieuwe gebruiker komt de gebruiker in een standaard groep terecht. Een User heeft een status. Deze status kan ook “banned” zijn als de bezoeker zich onjuist gedragen heeft. De editor kan een bezoeker blokken.

De bezoeker kiest een taal en kan een plaatje voor zijn profiel kiezen. Het opslaan van plaatjes en video’s gebeurd niet in de database (als BLOB). De reden hiervoor is dat naar mate het aantal bestanden toeneemt de performance van de database minder wordt. Ook neemt de grootte van de tabellen enorm toe wat onhandig is (denk aan een export van de database). Wat wel opgeslagen wordt is waar het bestand te vinden is.
De tabel File bevat de URL, de naam van het bestand en het type van het bestand. Zo kan er een aparte map ingericht worden voor onder andere plaatsjes en video´s.

Reacties (tabel Reaction) van bezoekers zijn gekoppeld aan vragen (tabel Question). Vragen zijn weer gekoppeld aan kamers (tabel Room).
De structuur is dan als volgt. Een kamer heeft één of meerdere vragen. Een vraag heeft de bijbehorende reacties.
Tabel Questionttype slaat vraagtypes op. Denk hierbij aan statements of vragen.

Doordat een vraag ook gekoppeld is aan een groep is het mogelijk om een bepaalde groep mensen toegang te geven tot een vraag.

Wat verschilt ten opzichte van de huidige applicatie is dat de administrator ook in de front-end en back-end(editor) van de Multilogue kan inloggen. Dit is eenvoudiger voor bugfixes, omdat er niet een nieuw account aan gemaakt hoeft te worden. De administrator heeft alle rechten(gelijk aan editor).

Favorietenlijst en sessies

Favoriete vragen en reacties worden opgeslagen in favorietenlijst met behulp van User_Reaction en User_Question. De tabel Session houdt bij wanneer een gebruiker voor het laatst ingelogd is en of deze online is.

In bijlage 2 zijn de tabellen gegeven. De tabel User heeft een verwijzing naar de tabel Favoritelist. Reacties en vragen zijn gekoppeld door middel van koppeltabellen doordat dit veel op veel relaties zijn.

2.1.2 Multilanguage

De tabel language slaat de talen op die beschikbaar zijn voor de Multilogue. Als de taal nederlands is kan de code beschreven worden als "nld". Dit gebeurd volgens de internationale standaard ISO 639-3. De tabel Translation bevat een key met de daarbij horende vertaling. Zie bijlage 3 voor de tabellen.

2.1.3 Polls en gebruikers

Bijlage 4 geeft de tabellen weer hoe de stemmen van gebruikers op polls worden opgeslagen.

In de Multilogue wordt er onderscheid gemaakt tussen twee verschillende polls. Dit zijn generale polls die opgeslagen worden in Poll en polls gekoppeld aan vragen (Questionpoll).

Een pagina (Page) kan meerdere polls bevatten. Vandaar de koppeltabel tussen tabel Poll en tabel Page.

De tabel Polloption houdt de opties bij die gekozen kunnen worden in een poll. Pollvote slaat de votes op van gebruikers.

2.1.4 Waarderen en categoriseren

De groep tabellen voor het waarderen en categoriseren geven de dataopslag weer voor het waarderen en categoriseren van vragen en reacties (bijlage 5).

De tabellen Rating en Category slaan de namen op van de categorie- en waarderingsnamen. De reden dat deze tabellen gescheiden zijn is dat het op deze manier eenvoudig mogelijk is om zowel categorie als waardering als een plugin te installeren (dus niet inbegrepen in het basisproduct). Daarnaast verschillen category en de waardering van elkaar doordat er meerdere categorieën door een moderator toegekend kunnen worden aan een reactie en maar één waardering aan een reactie. Ook de weergave van waarderingen en categorieën is veranderbaar (denk aan cijfers/sterren).

De tabellen zijn weergegeven zodat er een beeld is hoe de tabellen eruit zien.

In de koppeltabellen is er een verwijzing naar de gebruiker zodat zichtbaar is welke moderator/editor de vraag/reactie gewaardeerd/gecategoriseerd heeft.

2.1.5 Settings

De settings slaat configuratiewaarden op van de Multilogue. De tabel is te zien in bijlage 6.

Er is gekozen om een tabel aan te maken waar alle configuratiewaardes worden opgeslagen. Kolom key is de configuratienaam en value de waarde die hiervoor geldt.

2.1.6 Formfield

Dit hoofdstuk beschrijft de wijze waarop een "formfield" opgeslagen wordt. Een formfield is een dynamisch veld dat door een editor aangemaakt kan worden. Dit veld kan vervolgens gebruikt worden in de front-end zodat gebruikers een extra veld kunnen/moeten invullen in het profiel.

Een formfield kan een enkel invoerveld zijn evenals een keuze uit een lijst (dropdown box). Indien het formfield een lijst betreft wordt dit opgeslagen in tabel Fieldlist met bijbehorende opties (opgeslagen in Listitem).

Koppeltabel Formfield_User slaat de waardes op die gebruikers invullen in het formfield. Bijlage 7 geeft de tabellen weer.

2.2 Database administrator

Hoofdstuk 1.2 beschrijft de database die gebruikt wordt voor het administrator gedeelte. De tabellen zijn gegeven in bijlage 8.

Het administrator gedeelte functioneert enkel voor het bijhouden van de Multilogues.

Tabel Customer bevat klanten die een Multilogue draaiende hebben of draaiende gehad hebben, of gaan hebben.
Tabel User moet niet verward worden met de gebruikers zoals die in de Multilogue database zijn gedefinieerd. Tabel User in de administrator context is een gebruiker van het administrator systeem en staat los van de Multilogue.
Talen kunnen toegevoegd worden die vervolgens gekozen kunnen worden bij het aanmaken van een nieuwe Multilogue.

3 Sitemaps

Hoofdstuk 2, “Sitemaps” geeft de structuur aan van de webapplicatie ofwel de pagina’s van de webapplicatie die zichtbaar zijn voor de eindgebruiker (de views van het mvc model). Wat betreft de URLs van de applicatie wordt er de volgende standaard aangehouden:

BaseURL/Controllernaam/actienaam/eventuele_parameters.

Deze methode wordt gehanteerd omdat dit de standaard opzet is voor een URL in combinatie met het MVC model.
Sitemaps zijn gegeven voor de:

1) Front-end
2) Editor back-end
3) Administrator back-end

3.1 Front-end

De pagina’s voor de front-end zijn:

User
- -Login
- -View
- -Edit
- -List

Page
- -Background
- -Help

Home
- -Index

Room
- -List
- -View

Question
- -List
- -Add
- -Edit

Reaction
- -Report
- -Categorize
- -Rate

Video
- -List
- -View

Poll
- -View
- -Add

Favorite
- -AddQuestion
- -AddReaction
- -List

3.2 Editor back-end

Pagina's in de back-end voor de editor zijn:

Home
- -Index

Content
- -ViewSidebars
- -AddSidebar
- -EditSidebar
- -ViewPagecontent
- -EditPagecontent

User
- -Login
- -Add
- -AddUsers
- -Search
- -Follow
- -View

Blacklist
- -List
- -Add

Room
- -Edit
- -Add
- -Index

Question
- -Edit
- -Add

Statistic
- -General
- -Room
- -Poll
- -Responses

Category
- -List
- -Add
- -Edit

Rating
- -List
- -Add
- -Edit

Formfield
- -List
- -Add
- -Edit

Fieldlist
- -List
- -Add
- -Edit

Poll
- -Add
- -List
- -Edit

Translation
- -Show
- -Edit
- -View

3.3 Administrator back-end

De back-end van de administrator is gescheiden van de back-end van de editor en de front-end van de Multilogue.
Pagina's voor de administrator zijn:

Home
- -Index

User
- -Login
- -Add
- -List

Customer
- -List
- -Add
- -Edit

Multilogue
- -List
- -Add
- -Edit

Language
- -List
- -Edit
- -Add

4 Klassenstructuur

Hoofdstuk 3, “Klassenstructuur” geeft de structuur van klassen weer voor de te realiseren applicatie. De structuur is verdeeld in LIBs, Front-end, Back-end Editor en Administrator. In hoofdstuk 5.5 staat hoe de applicatie gerealiseerd moet worden in Visual Studio 2008. Deze indeling wordt ook hier behouden. Dat betekent dat in het “LIB project” modellen van de tabellen worden bewaard en tevens de uitgebreide classes van de models. De front-end/back-end en administrator bezitten de controllers en views van de projecten (niet functionele eis “Applicatie moet opgebouwd zijn uit modules”).

Benamingen van functies zijn zo kort mogelijk gehouden zodat het duidelijk is wat een functie doet en makkelijk te lezen is.

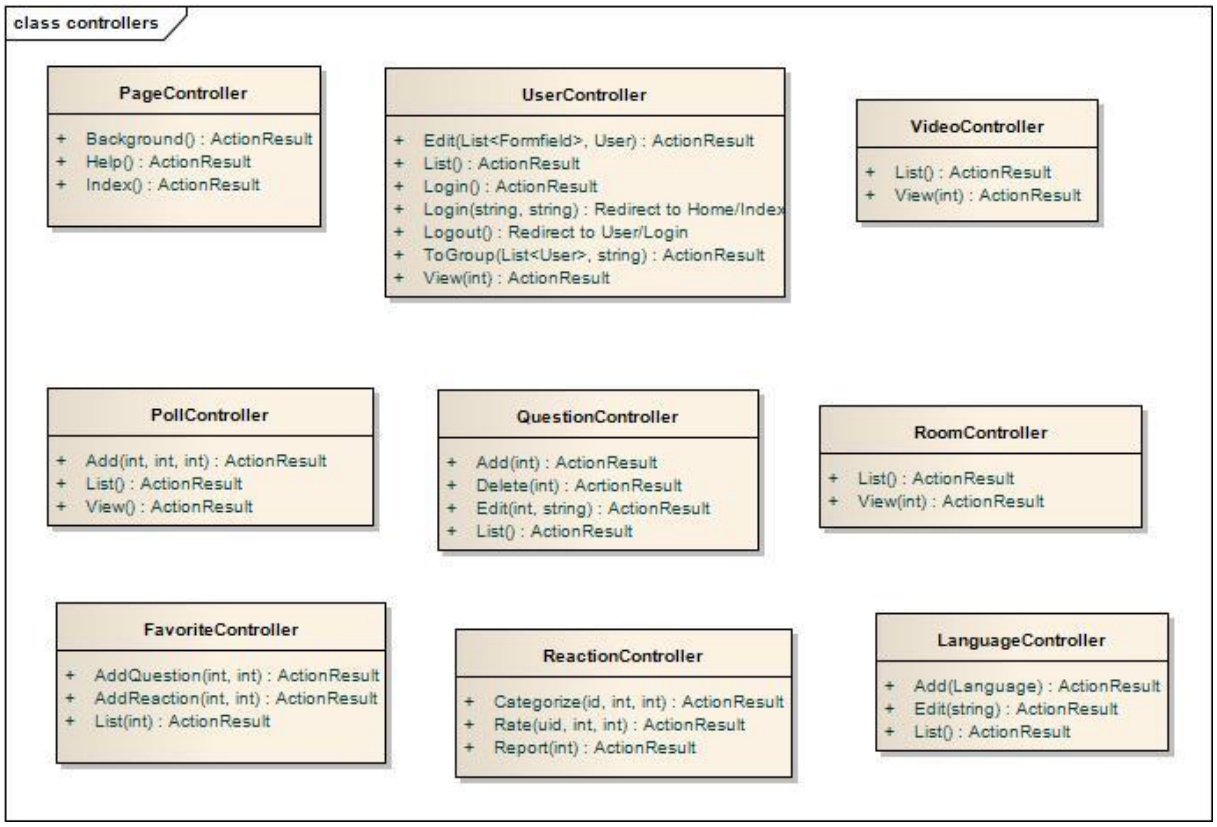
4.1 Front-end Controllers

De controllers voor de front-end zijn weergegeven in afbeelding 1.
Instanties van controllers hebben geen relaties met elkaar. Een aantal acties hebben wel een redirect naar een andere actie (na succesvol ingelogd zijn redirect naar Home/Index).

De HomeController zorgt ervoor dat de juiste handelingen uitgevoerd worden als de gebruiker de background, help en index pagina bezoekt.

De UserController zorgt voor het afhandelen van gebruikers specifieke handelingen. Zo wordt er gezorgd voor het correct inloggen, wijzigen van profiel, tonen van gebruikers die online zijn en het toevoegen van gebruikers aan groepen.

Het tonen van polls gebeurd door de PollController(eis 8). Acties op het gebied van vragen en reacties worden afgehandeld door de QuestionController en ReactionController. Reacties kunnen verwijderd worden door een editor (eis 26). Ook kan de laatst geschreven reactie opgehaald worden (eis 13).



Afbeelding 1 Front-end controllers

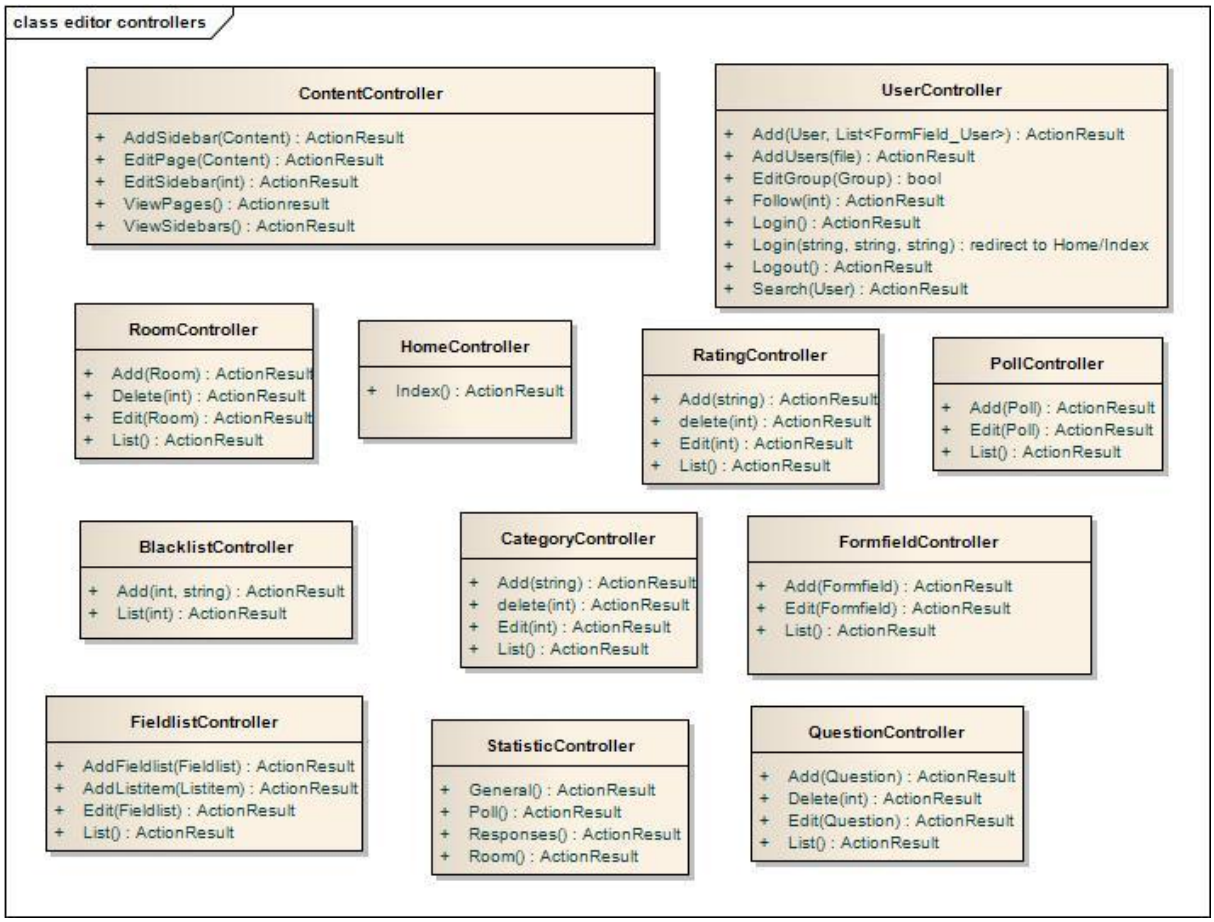
4.2 Editor Controllers

De controllers voor de editor kant zijn weergegeven in afbeelding 2.
Het volgen van de gebruiker door een editor wordt verwezenlijkt door de UserController met actie FollowUser (eis 37). Het zoeken van gebruikers gebeurd door middel van actie SearchUser (eis 34).

De ContentController zorgt voor acties zoals het toevoegen van zijbalken en het wijzigen van pagina-inhoud. Het toevoegen en het tonen van de polls gebeurd wordt gedaan door de PollController (eis 27/36).

Vertalingen van inhoud worden op de pagina's zelf weergegeven door middel van de taal te kiezen uit een dropdown box (eis 30).

Voor de statistieken is er de StatisticController(eis 20).



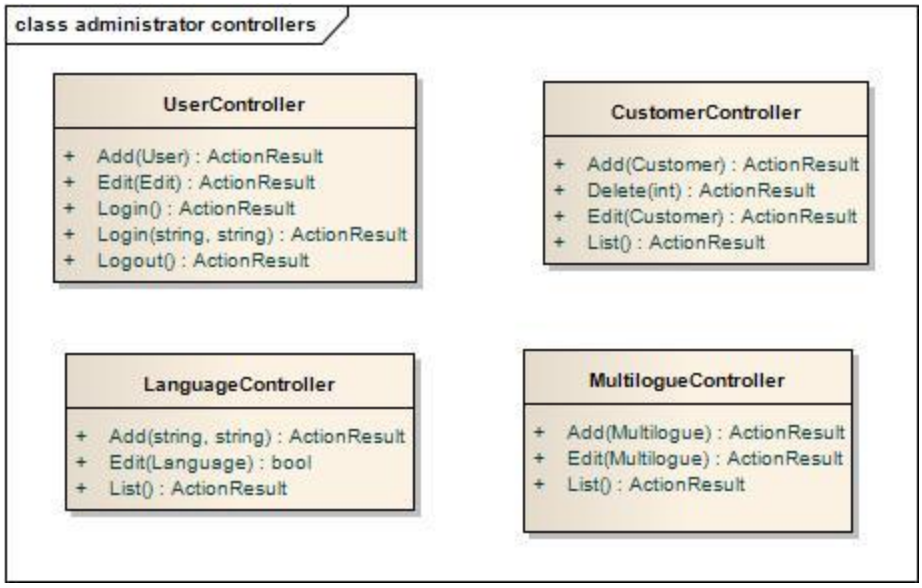
Afbeelding 2 Editor controllers

4.3 Administrator

De controllers voor de administrator kant zijn weergegeven in afbeelding 3.
De administrator is een aparte applicatie die gegevens en settings van Multilogues samenstelt. Deze settings worden opgeslagen in de Multilogue. Hoe het Administrator gedeelte precies gaat werken met Mutlilogues wordt later bepaald. Dit is zo gekozen omdat de administrator alleen Multilogues aanmaakt.

Bij het aanmaken van een Multilogue kiest de administrator de beschikbare talen voor de Multilogue of voegt een nieuwe taal toe indien deze nog niet aanwezig is(eis 44).

Naast het toevoegen van talen kunnen er ook nieuwe administrators toegevoegd worden en bekeken worden (eis 42/49). Bij het toevoegen van een nieuwe administrator wordt deze ook toegevoegd in de User tabel van de Multilogue. Dit geschiedt via SOAP.

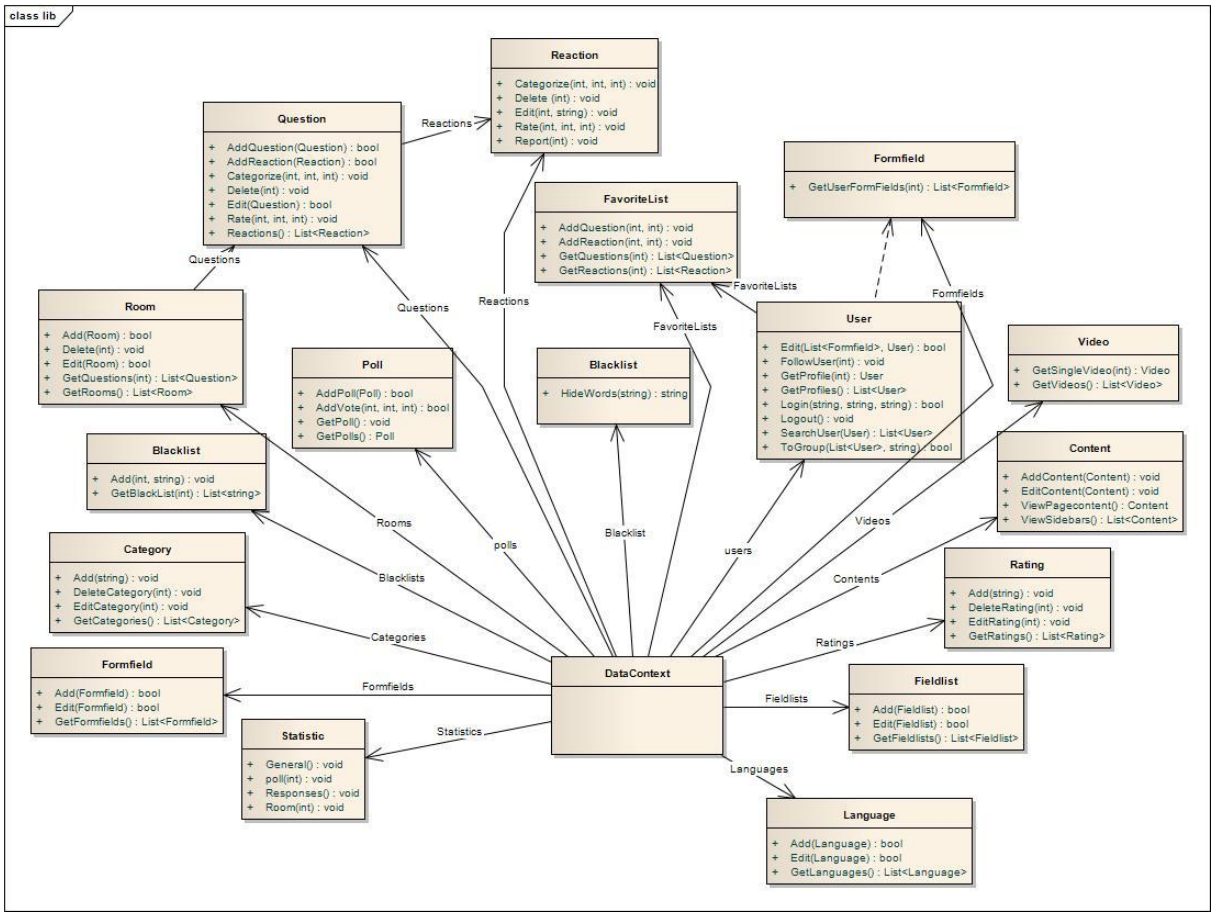


Afbeelding 3 – Administrator controllers

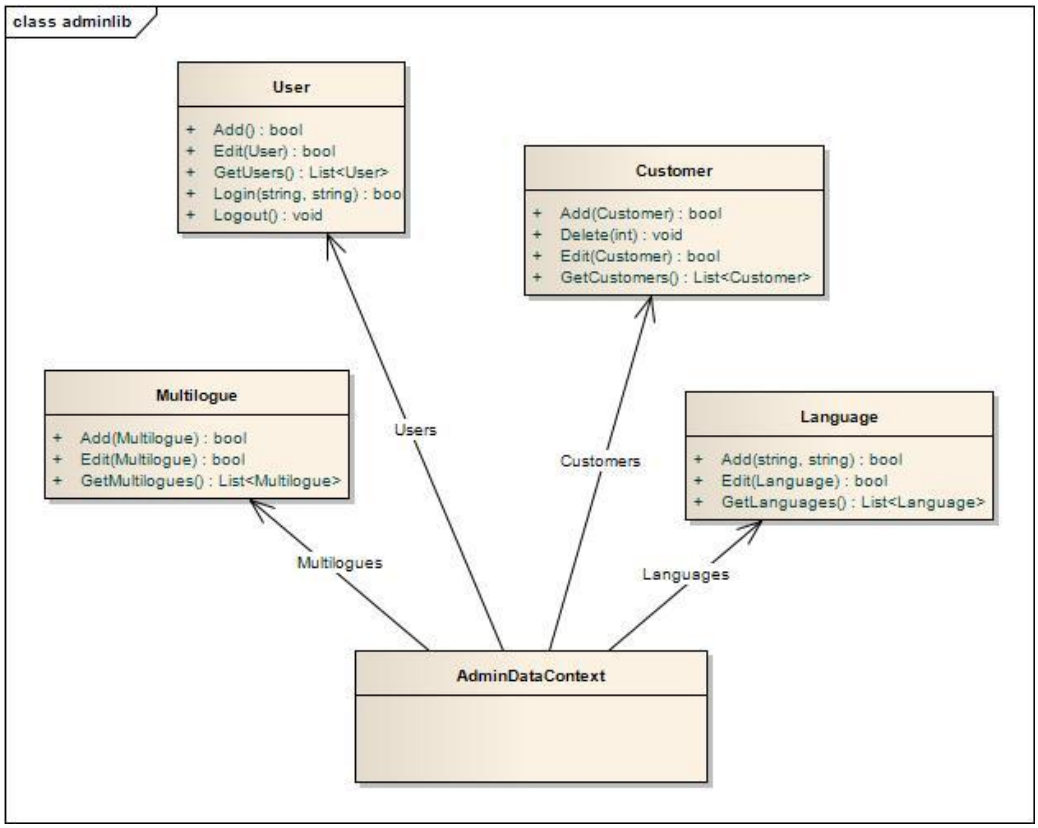
4.4 Libs

De Lib voor de Multilogue en administrator zijn weergegeven in de afbeeldingen 4 en 5 en bijlagen 9 en 10. De admin lib en Multilogue lib zijn gescheiden van elkaar. Hier worden de modellen en de classes die de model class uitbreiden in opgeslagen. Het voordeel hiervan is dat het project gecompiled kan worden naar een dll. Deze kan vervolgens gebruikt worden voor de front-end, editor en administrator.

De diagrammen geven enkel de hulpfuncties weer van de (gedeeltelijke klassen). Elke klasse bezit alle kolomattributen van de tabel als private variabelen. Deze zijn eruit gelaten voor de duidelijkheid. De DataContext wordt in de gedeeltelijke klassen aangemaakt wanneer dit nodig is.



Afbeelding 4 – Multilogue lib



Afbeelding 5 – Admin lib

5 Interactie objecten

Hoofdstuk 4, "Interactie objecten" beschrijft de interactie tussen objecten. Functionele eisen die complexer zijn, zijn toegelicht in sequentiediagrammen.
De sequentiediagrammen beperken zich tot de controller classes en uitgebreide classes van LINQ voor hulpfuncties. De (html)views die aangesproken worden vanuit de controllers worden achterwege gelaten zodat de sequentiediagrammen niet te groot worden.

Sequentiediagrammen zijn zo eenvoudig mogelijk weergegeven. Omdat de applicatie database driven is bevatten meeste diagrammen het opslaan of ophalen van data.

Sequentiediagrammen zijn gegeven van:

1) Front-end
2) Editor
3) Administrator

5.1 Front-end

Hoofdstuk 5.1, Front-end, geeft de functionele eisen weer in sequentiediagrammen die betrekking hebben op de front-end.

De functionele eisen die beschreven zijn:

1) Inloggen
2) Uitloggen
3) Plaats reactie
4) Stem in poll
5) Wijzigen van profiel
6) Kies taal
7) Wie is online
8) Voeg reactie toe aan favorietenlijst
9) Rapporteer reactie
10) Bekijk video
11) Reactie verwijderen wijzigen
12) Waardeer reactie
13) Voeg gebruiker toe aan groep

5.1.1 Inloggen

Om reacties te plaatsen en het profiel te tonen is het nodig om in te loggen (bijlage 11). Als de gebruiker nog niet ingelogd is (dus geen sessie aanwezig) wordt de gebruiker naar de loginpagina gebracht. Daar dient de gebruiker een e-mail en een wachtwoord op te geven. Het type geeft weer dat het om een front-end login gaat. Dit type wordt gebruikt bij het editor gedeelte omdat daar alleen editors toegestaan zijn.

Hierbij wordt gebruik gemaakt van "SHA1". SHA1 wordt gebruikt voor het encrypten van het wachtwoord (niet functionele eis "Wachtwoorden zijn "ge-encrypt" opgeslagen in de

database”). Zodra er een gebruiker gevonden is met het opgegeven e-mail en wachtwoord wordt de sessie gevuld met het uid van de gebruiker. De gebruiker wordt naar de index pagina gebracht nadat er geen fouten opgetreden zijn. (eis 4)

Alle onderstaande acties/eisen wordt de aanname gedaan dat de gebruiker ingelogd is.

5.1.2 Uitloggen

Als de gebruiker wilt uitloggen gebeurt dit door middel van een link te klikken. Deze link verwijst naar de actie logout van controller UserController. De sessie wordt geleegd en het online bit in de tabel Session wordt op nul gezet. De gebruiker komt weer op de loginpagina terecht. (eis 5)

Zie bijlage 12.

5.1.3 Plaats reactie

Het proces van het plaatsen van een reactie is weergegeven in bijlage 13 en 14. Dit betekent dat het selecteren van de kamer en vraag ook aan bod komt.

Voordat een gebruiker een reactie kan plaatsen moet er eerst een kamer gekozen worden. De actie List zorgt ervoor dat alle kamers getoond worden die van belang zijn. Of kamers getoond worden of niet hangt af van de taal die vergeleken wordt met de taal van de gebruiker en of de kamer nog actief is of niet. De taal van de gebruiker en de kamer moeten overeenkomen. Nadat de kamer gekozen is wordt de vraag gekozen en kan er een reactie geplaatst worden (eis 2).

5.1.4 Stem in poll

Bij het stemmen in een poll geeft de gebruiker zijn/haar mening op een vraag die gesteld is door een editor (eis 6).
Zie bijlage 15.

5.1.5 Wijzigen van profiel

Bij het wijzigen van het profiel kan de gebruiker profielgegevens naar wens aanpassen. De profielgegevens worden opgehaald door middel van de useruid die in de sessie staat . Ook de formfields worden opgehaald. Hierbij wijzigt de gebruiker een of meerdere velden waarbij vervolgens de database geüpdatet wordt (eis 1/3). Het diagram is gegeven in bijlage 16.

5.1.6 Kies taal

Het taal kiezen geschiedt door middel van op een link te klikken. De code van de taal wordt in de sessie gezet. Vervolgens kan de inhoud van de pagina’s in de juiste taal weergegeven worden door gebruik te maken van de Translation tabel (eis 7). Zie bijlage 17.

5.1.7 Wie is online

Om de gebruiker de profielen te tonen van mensen die online zijn wordt eerst de Session tabel geraadpleegd om de uids op te halen van gebruikers die online zijn. Gegevens van de gebruikers worden vervolgens opgehaald aan de hand van het uid met als voorwaarde dat zij ook online gezien willen worden (visible bit) (eis 9). Het diagram staat in bijlage 18.

5.1.8 Voeg reactie/vraag toe aan favorietenlijst

Voor het toevoegen van een reactie aan de favorietenlijst zijn de reactieid en userid benodigd. Deze worden opgeslagen in de tabel User_Reaction (bijlage 19).
Het toevoegen van een favoriete vraag geschiedt op dezelfde wijze, enkel is hier tabel User_Question benodigd in plaats van User_Reaction (eis 10).

5.1.9 Rapporteer reactie

Indien een gebruiker een reactie rapporteert wordt er een mail verzonden naar de editors met bijbehorende reactie-informatie. Waaronder een directe link waar de reactie te vinden is. De editor kan besluiten om de reactie te verwijderen of wijzigen (eis 11). Zie bijlage 20.

5.1.10 Bekijk video

Als een gebruiker een video wilt bekijken wordt eerst de gehele lijst van beschikbare video's getoond. Hierna maakt de gebruiker een keuze die vervolgens opgezocht wordt aan de hand van gegevens die in de File tabel staan (eis 12). Het sequentiediagram is gegeven in bijlage 21.

5.1.11 Reactie verwijderen/wijzigen

Het verwijderen en wijzigen van reacties doet de editor in de front-end. De editor kan de reactie wijzigen als hij of zij dit nodig vindt. In bijlage 23 is het wijzigen van een reactie in een sequentiediagram weergegeven. Voor het verwijderen wordt de actie Delete aangeroepen van ReactionController (eis 22).

5.1.12 Waardeer reactie

Het waarderen van reacties in de front-end kan gedaan worden door moderators en editors. Hierbij wordt er een cijfer of een waardering toegekend aan een reactie. In bijlage 23 is het sequentiediagram gegeven (eis 16). Het categoriseren van reactie gebeurd op dezelfde wijze (ReactieController/Categorize) en class Reaction met functie Categorize (eis 15).

5.1.13 Voeg gebruiker toe aan groep

Alleen de editor is gemachtigd om een gebruiker toe te voegen aan een groep. Een gebruiker kan elk type (normaal, moderator en editor) zijn. Het toevoegen van een groep kan nut hebben als een vraag in een kamer ook gekoppeld wordt aan een groep. Hierbij kunnen een beperkt aantal gebruikers de vraag lezen als zij zich in de groep bevinden die de vraag mag lezen. In bijlage 24 is het sequentiediagram gegeven (eis 39/40).

5.2 Editor back-end

Hoofdstuk 5.2, Editor back-end, geeft de functionele eisen weer in sequentiediagrammen die betrekking hebben op de Editor.

De functionele eisen die beschreven zijn:

1) Inloggen
2) Uitloggen
3) Gebruiker toevoegen
4) Voeg kamer toe
5) Voeg vraag toe
6) Vul blacklist
7) Pagina content toevoegen
8) Voeg categorie/waardering toe
9) Voeg formfield toe

5.2.1 Inloggen

Voor het aanmaken van nieuwe kamers en vragen moet er ingelogd worden. In de back-end kunnen enkel gebruikers van het type editor inloggen. Indien dit niet het geval is mislukt het inloggen (eis 28). Bijlage 25 geeft deze eis weer in een sequentiediagram.

Alle onderstaande acties/eisen wordt de aanname gedaan dat de editor ingelogd is.

5.2.2 Uitloggen

Het uitloggen in de back-end geschiedt op dezelfde wijze als bij het uitloggen in de front-end (eis 29). Zie bijlage 26.

5.2.3 Gebruiker toevoegen

Bij het toevoegen van een gebruiker (bijlage 27) worden de benodigde gegevens ingevoerd (zie tabel User). De status die de gebruiker automatisch krijgt toegewezen is “niet actief”. De gebruiker en de waardes van de formfields worden vervolgens in de database gezet.

Het is ook mogelijk om een csv bestand op te geven met informatie over gebruikers. De controller maakt er User objecten van die vervolgens doorgegeven worden aan een Instantie van User (eis 23).

5.2.4 Kamer toevoegen

Het toevoegen van een kamer is weergegeven in bijlage 28. De kamer kan toegewezen worden aan meerdere talen (eis 24).

5.2.5 Voeg vraag toe

Nadat een kamer gekozen is om de vraag toe te voegen wordt de vraag opgeslagen in tabel Question. Optioneel is het kiezen van een groep. Als een vraag gekoppeld wordt aan één of meerdere groepen kunnen alleen gebruikers die zich in dezelfde groep bevinden de vraag zien (eis 25). Zie bijlage 29 voor het sequentiediagram.

5.2.6 Vul blacklist

De blacklist wordt gevuld met woorden per taal die niet toegestaan zijn (eis 33). Zie bijlage 30.

5.2.7 Pagina content toevoegen

Onder pagina content worden ook zijbalken verstaan. Het verschil zit hem in het container attribuut. Deze geeft aan of het pagina content is of een zijbalk. Een belangrijk verschil is dat zijbalken toe te voegen zijn en paginacontent alleen wijzigbaar is (eis 17/18/19/35). Zie bijlage 31.

5.2.8 Voeg categorie/waardering toe

Het toevoegen van categorieën en waarderingen zorgen ervoor dat moderators in de front-end reacties kunnen categoriseren en waarderen met de ingevulde waardes. De waardering en categorienamen zijn nog wijzigbaar (eis 15/16/21/22). Zie bijlage 32.

5.2.9 Voeg formfield toe

Bij het toevoegen van een formfield(bijlage 33) kan er gekozen worden om welk type het gaat. Denk hierbij aan eenvoudig invoerveld of een lijst(Fieldlist) met opties(Listitem) (eis 31/32). Naast het toevoegen zijn ook de formfields te bekijken en aan te passen.

5.3 Administrator

Hoofdstuk 5.3, Administrator, geeft de functionele eisen weer in sequentiediagrammen die betrekking hebben op de Administrator.

De functionele eisen die beschreven zijn:

1) Inloggen
2) Voeg klant toe
3) Voeg Multilogue toe

5.3.1 Inloggen

Het inloggen geschiedt op dezelfde wijze als de back-end van de editor en de front-end. Enkel de User tabel bevindt zich in de administrator tabel (eis 46). Het uitloggen gebeurt door het aanspreken van de controller UserController en actie Logout. Actie Logout spreekt functie Logout van de uitgebreide model class User aan(eis 47). Zie bijlage 34 voor het sequentiediagram.

Alle onderstaande acties/eisen wordt de aanname gedaan dat de administrator ingelogd is.

5.3.2 Voeg klant toe

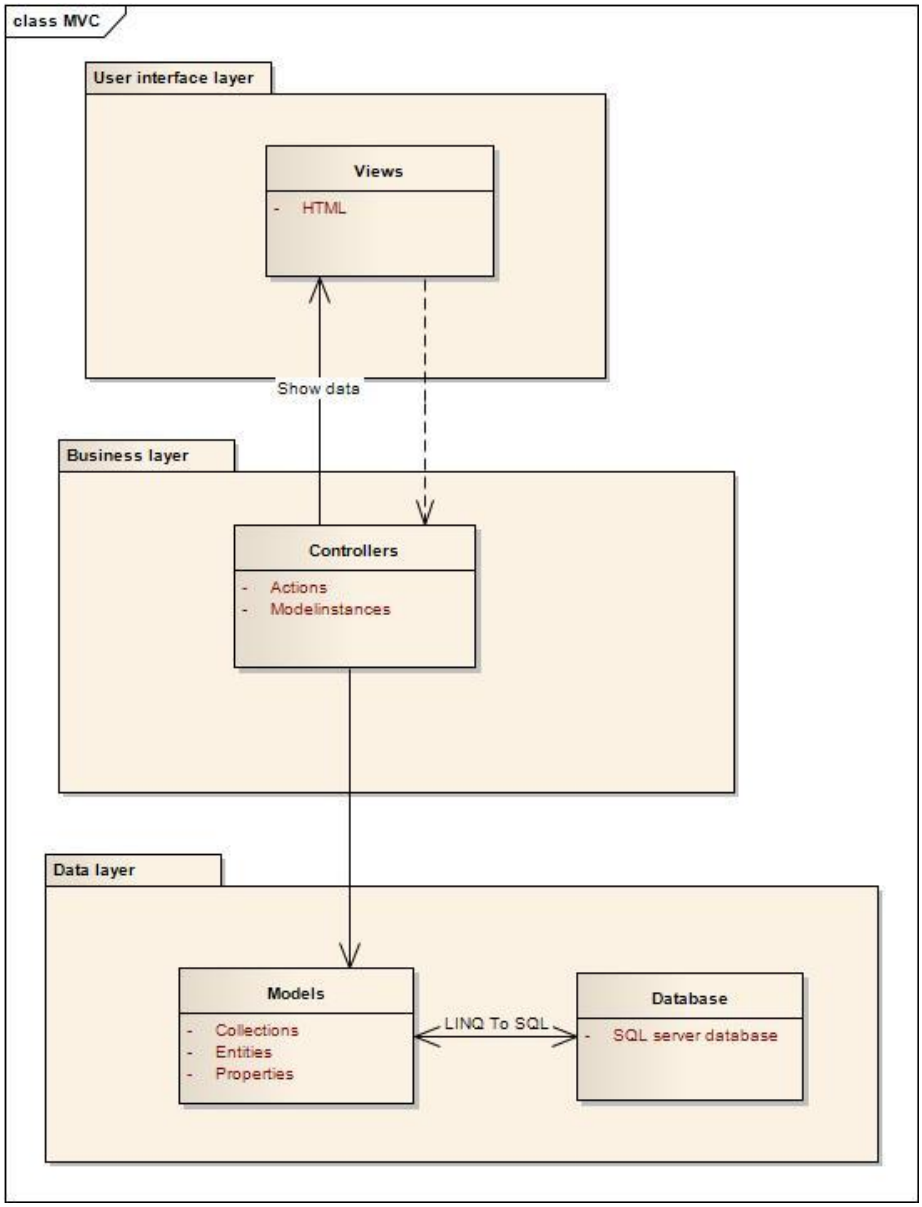
Het toevoegen van een klant (bijlage 35) is niets minder het invullen van de naam van het bedrijf waar de Multilogue voor opgezet wordt. Deze kan dan gekozen worden bij het toevoegen van de Multilogue (eis 41). Het tonen van klanten gebeurd wordt afgehandeld door de controller CustomerController met actie View() . Deze actie spreekt functie View aan class Customer (eis 48).

5.3.3 Voeg Multilogue toe

De Multilogues worden in de tabel Multilogue opgeslagen. De informatie nodig is voor het correct opslaan is de naam en de periode wanneer de Multilogue actief moet zijn. Dit is aangegeven met start- en einddatum. Verder wordt de URL gekozen en de taal gekozen (eis 43). Het tonen van de geïnstalleerde Multilogues wordt afgehandeld door Controller MultilogueController met actie View. Zie bijlage 36 voor de sequentiediagram die het toevoegen van een Multilogue beschrijft.

6 Functionele decompositie

Hoofdstuk 6, “Functionele decompositie” beschrijft de verschillende elementen van de applicatie in verschillende lagen. Doordat er gebruik gemaakt is van het MVC model kan het niet één op één worden overgezet met een lagenstructuur zoals de “three-tier architecture” (afbeelding 7) (niet functionele eis “ontwerp maakt gebruik van een 3 TIER model”).



Afbeelding 6 – MVC in lagen

7 Gebruikte faciliteiten/methodes

Hoofdstuk 5, “Gebruikte faciliteiten/methodes”, worden de gebruikte methodieken en faciliteiten beargumenteerd.
De volgende keuzes zijn beschreven :

1) ASP.NET MVC model
2) Template engine
3) LINQ
4) Microsoft SQL server 2008
5) Implementatie Visual Studio

7.1 ASP.NET MVC model

Het ASP.NET MVC model wordt onder andere gebruikt omdat het de code scheidt van de grafische schil.
Dit heeft als voordeel dat het testen van code eenvoudiger gaat. Hierdoor is Test Driven Development (TDD) mogelijk.

Een ander voordeel is dat het ASP.NET MVC volledige controle geeft over de HTML.
Doordat webforms niet door slepen in elkaar worden gezet, maar door zelf de HTML te typen.

De routing gaat als volgt, gegeven de volgende link: /Products/Detail/:product_id.
Deze redenen verantwoorden het gebruik van het MVC model.

7.2 Template engine

Zonder template engine wordt er in ASP.NET code gebruikt tussen ASP blokken (<% … %> om bijvoorbeeld een "foreach" lus uit te voeren die data ophaalt. Zie afbeelding 7.

List.aspx | List.aspx.cs

Client Objects & Events | (No Events)

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" AutoEventWireup="true" CodeBehi

<asp:Content ContentPlaceHolderID="MainContentPlaceHolder" runat="server">

    <h2><%=ViewData.CategoryName %></h2>

    <ul>

        <% foreach (var product in ViewData.Products) { %>

            <li>
                <%=product.ProductName %>

                <span class="editlink"">
                    (<%=Html.ActionLink("Edit", new { Action="Edit", ID=product.ProductID })%>)
                </span>
            </li>

        <% } %>

    </ul>

    <%= Html.ActionLink("Add New Product", new { Action="New" }) %>

</asp:Content>
```

Afbeelding 7 – voorbeeld foreach lus

Het idee van het MVC model is juist het scheiden van code en lay-out. Dat is zonder een template of view engine zeker niet het geval. Het scheiden van code en lay-out geeft naast het eenvoudig testen ook het voordeel dat een designer de HTML kan opmaken. Naarmate de code toeneemt neemt de moeilijkheidsgraad toe om HTML op te maken en het niet "beschadigen" van de code. Vanuit dit standpunt is een template engine benodigd.

Er zijn drie template engines onderzocht. Dit zijn StringTemplate, Spark en de template engine die binnen E-mark is ontwikkeld. Er is met name gekeken wat het eenvoudigst is om te begrijpen voor een HTML designer.

Spark gebruikt een XML manier om if statements en loops te realiseren (afbeelding 8). Naarmate de template ingewikkelder wordt, wordt de template ook lastiger te lezen.

```
<viewdata products="IEnumerable[[Product]]"/>
<ul if="products.Any()">
  <li each="var p in products">${p.Name}</li>
</ul>
<else>
  <p>No products available</p>
</else>
```

Afbeelding 8 – spark

Ook StringTemplate is niet geheel eenvoudig te lezen (Afbeelding 9).

```
<div>

        Employee Detail:
        $if(IsSuperUser)$
                <a href="/allowEdit">Edit</a>
        $else
                Editing not allowed
        $endif$
</div>
```

Afbeelding 9 – StringTemplate

Zowel Spark en StringTemplate zijn voor een designer lastig aan te passen. Ook een nadeel is dat deze templates niet door gevalideerd kunnen worden door een HTML validator.
Deze argumenten zijn doorslaggevend geweest in het kiezen van de template engine van E-mark. Voordelen van deze template zijn:

- Leesgemak
- E-mark is bekend met deze template engine
- Geschikt voor HTML validator
- Template engine is aan te passen

In afbeelding 10 is er een voorbeeld gegeven van de template die binnen E-mark gemaakt is.

```
<!-- IMPORT Shared/mijnHeader -->

<div>
<!-- LIST -->
 <h1>Header</h1>
   <!-- ITEM -->
       <p>value is: {variable}</p>
   <!-- /ITEM -->
<!-- /LIST -->

</div>

<!-- IMPORT Shared/mijnFooter -->
```

Afbeelding 10 – E-mark template engine

7.3 LINQ

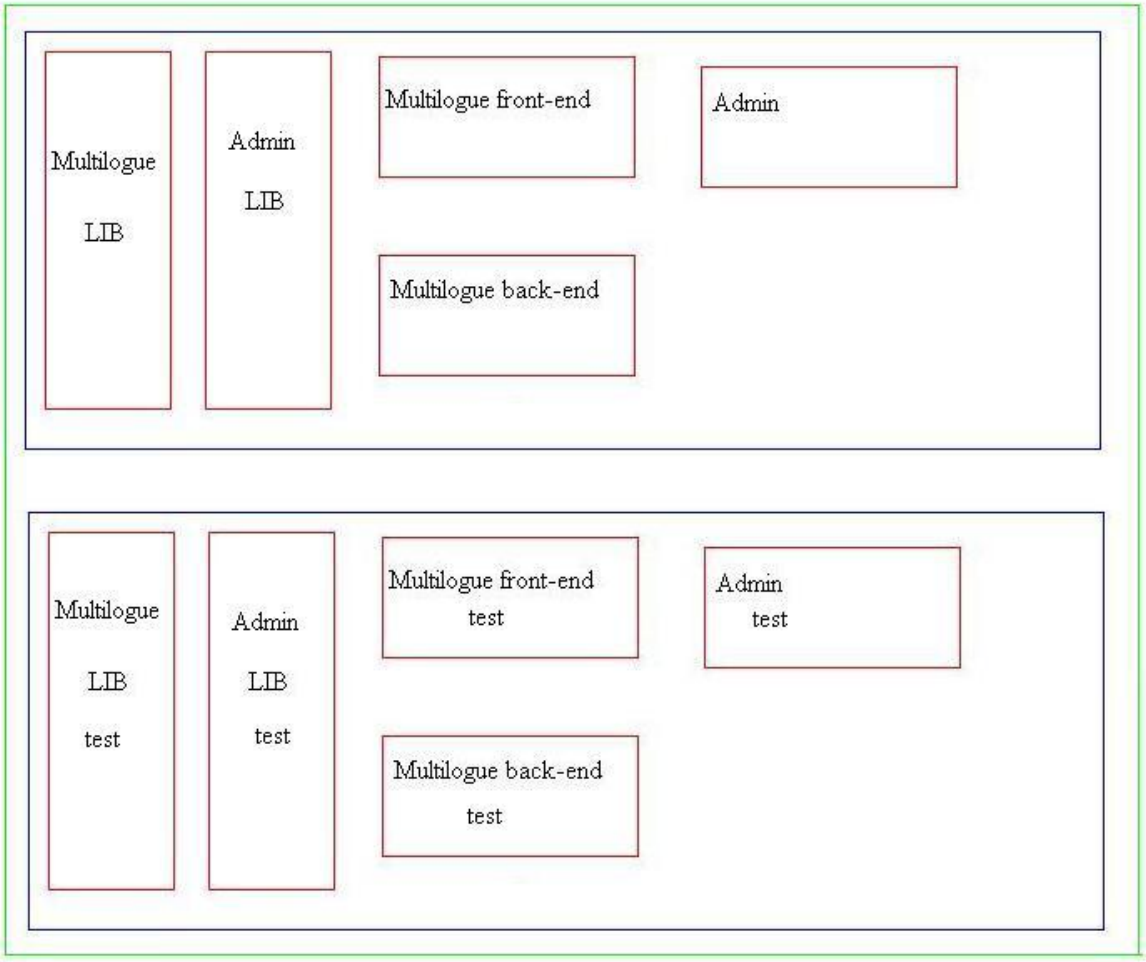
LINQ is gekozen omdat het gat tussen de programmeertaal en de database verkleint. LINQ genereert automatisch modellen (classes) van iedere tabel in de database. Elk attribuut wordt als een private variabele neergezet die een set en get toelaat. Doordat de classes als partial geïmplementeerd zijn kan eenvoudig een ander bestand aangemaakt worden die deze class uitbreid met hulpfuncties.

7.4 Microsoft SQL server 2008

SQL server 2008 is gekozen omdat er met Microsoft producten gewerkt wordt binnen E-mark als het om ASP.NET applicaties gaat.

7.5 Implementatie Visual Studio

De implementatie in Visual Studio is grafisch weergegeven in afbeelding 11.



Afbeelding 11 – Visual Studio

De groene lijn geeft de "solution" weer. Het bovenste blauwe kader is de applicatie zelf en het onderste blauwe kader geeft de testkader weer. De rode modules zijn de projecten.
Deze structuur houdt de tests gescheiden van de applicatie en het "Admin" gedeelte kan apart gestart worden.

In de applicatie zijn de projecten LIB, Multilogue front-end, back-end en Admin weergegeven.

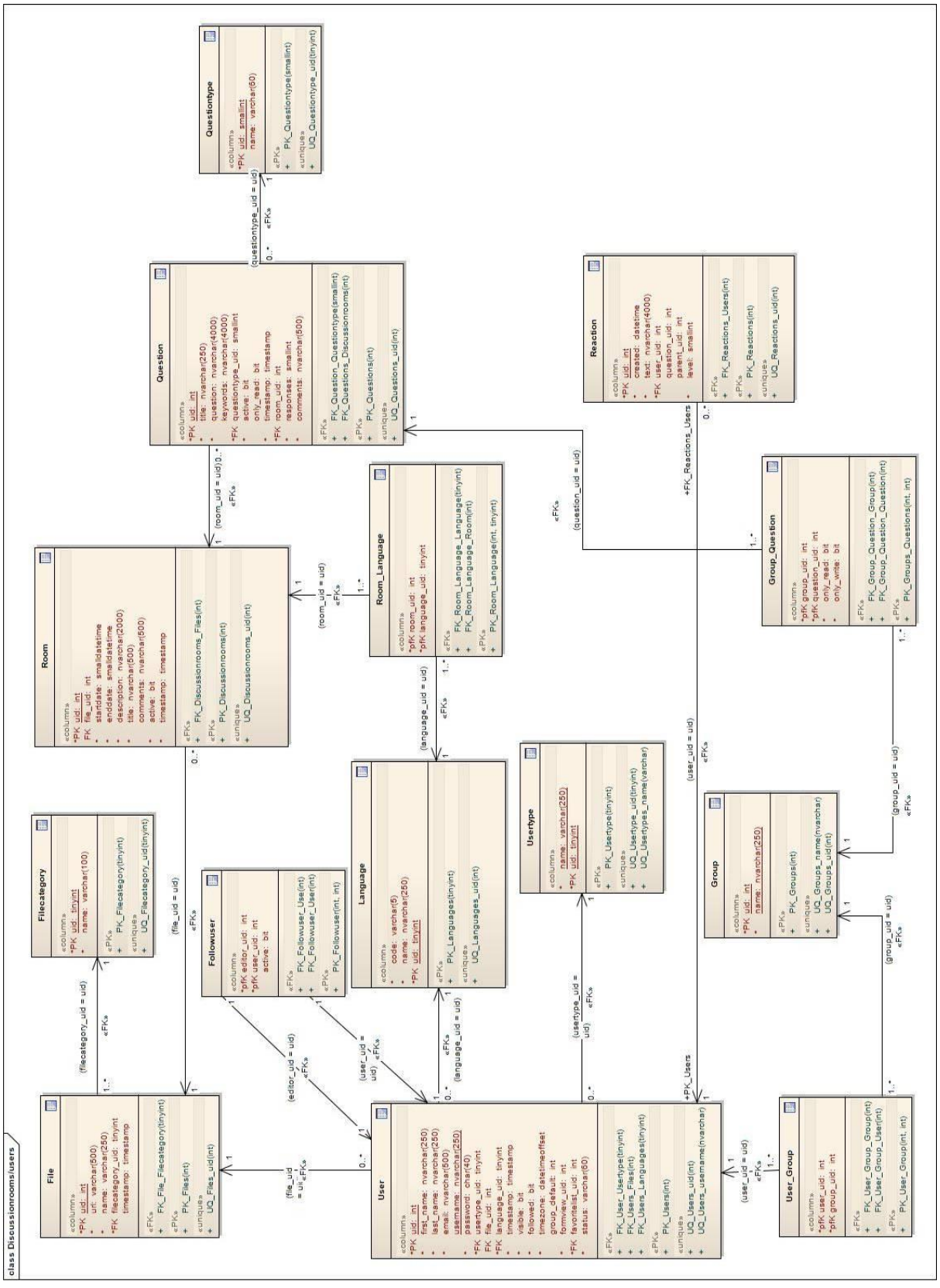
In de LIBs zijn aanwezig:

- Helpers
- Utils
- Models

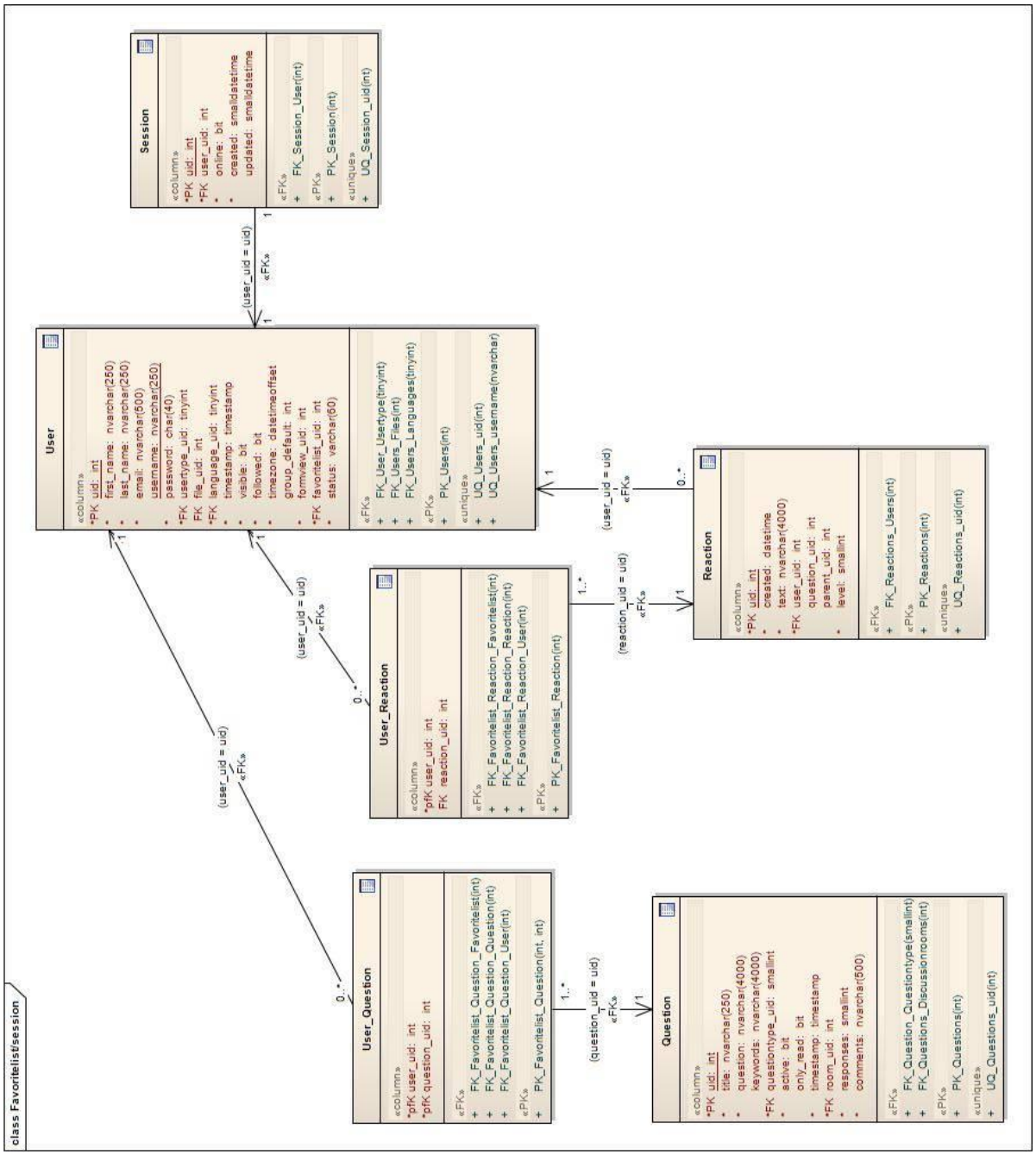
In de Front-end, back-end en admin zijn aanwezig:

- Controllers
- Views
- Settings

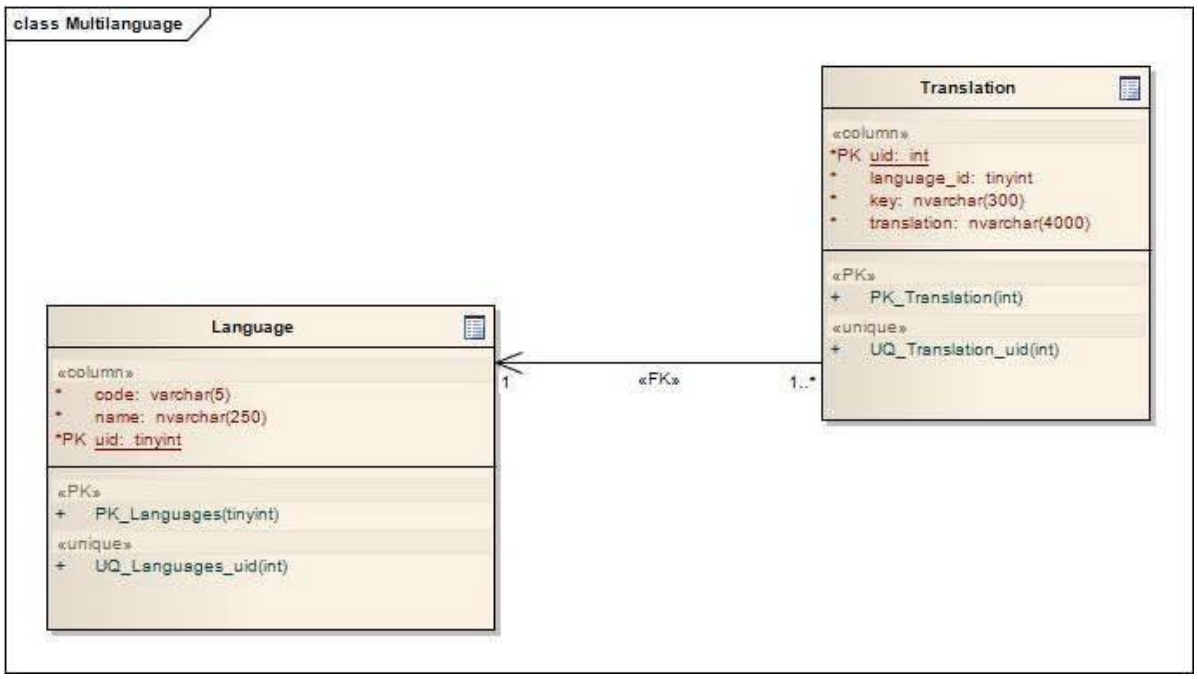
Bijlage 1 – Kamers en gebruikers (database)



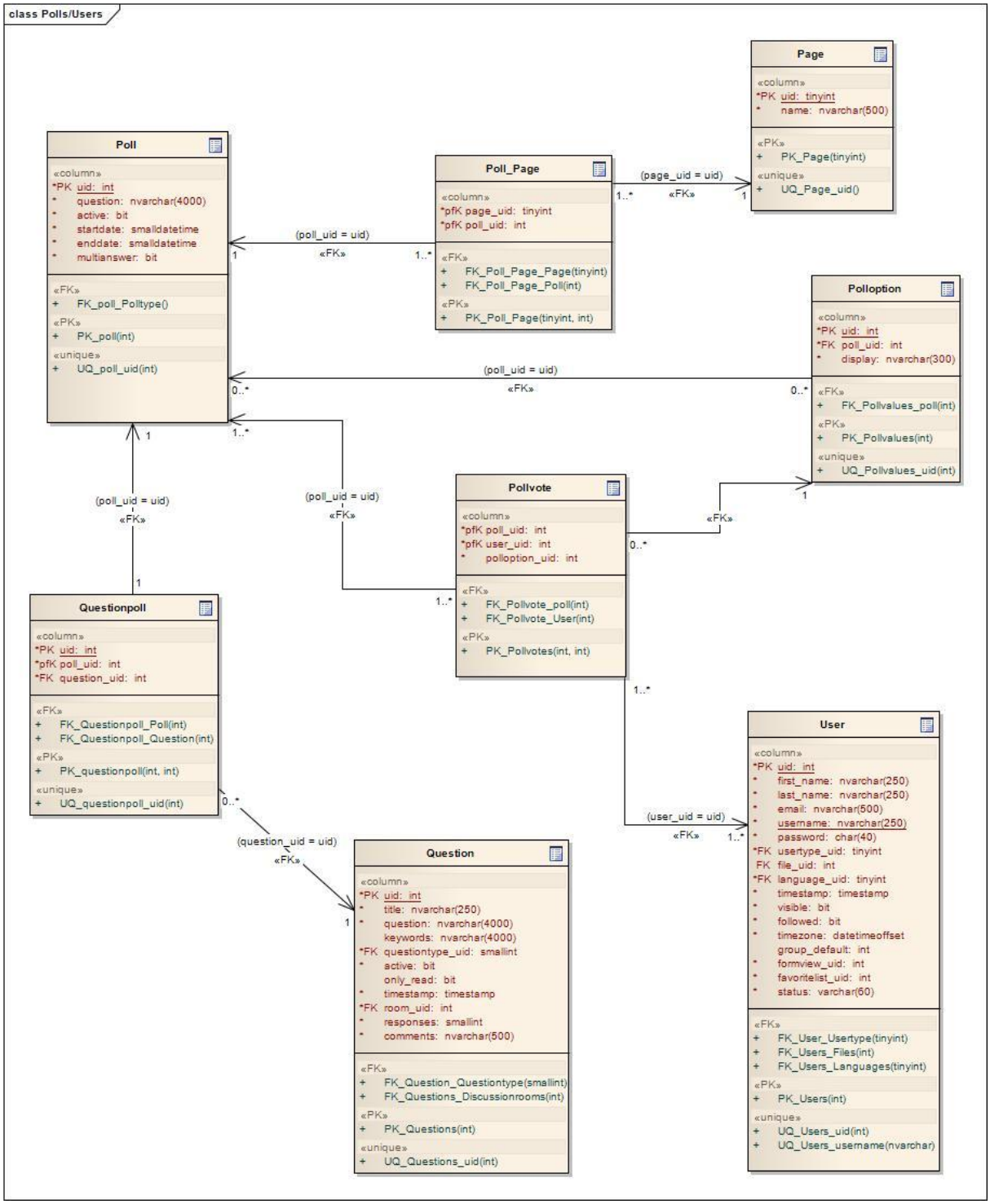
Bijlage 2 - Favorietenlijst en sessies (database)



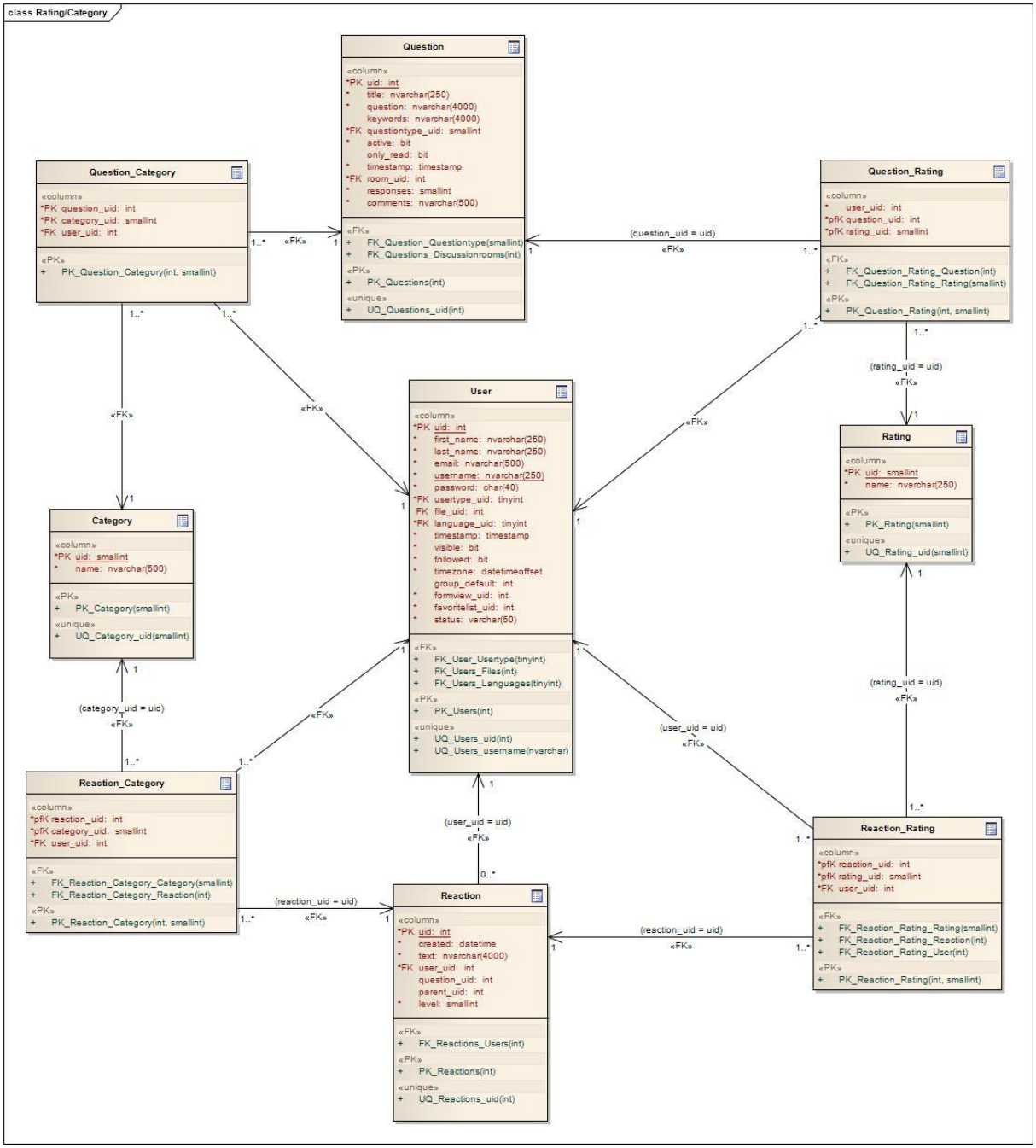
Bijlage 3 – Multilanguage (database)



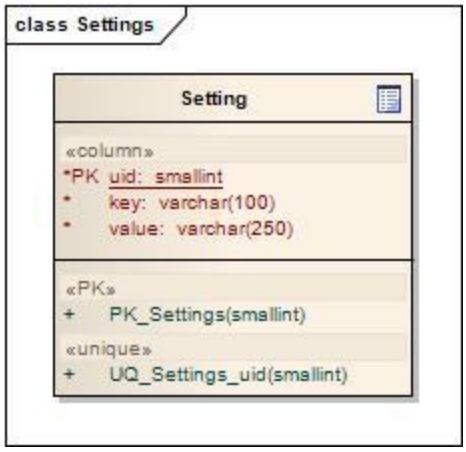
Bijlage 4 - Polls en gebruikers (database)



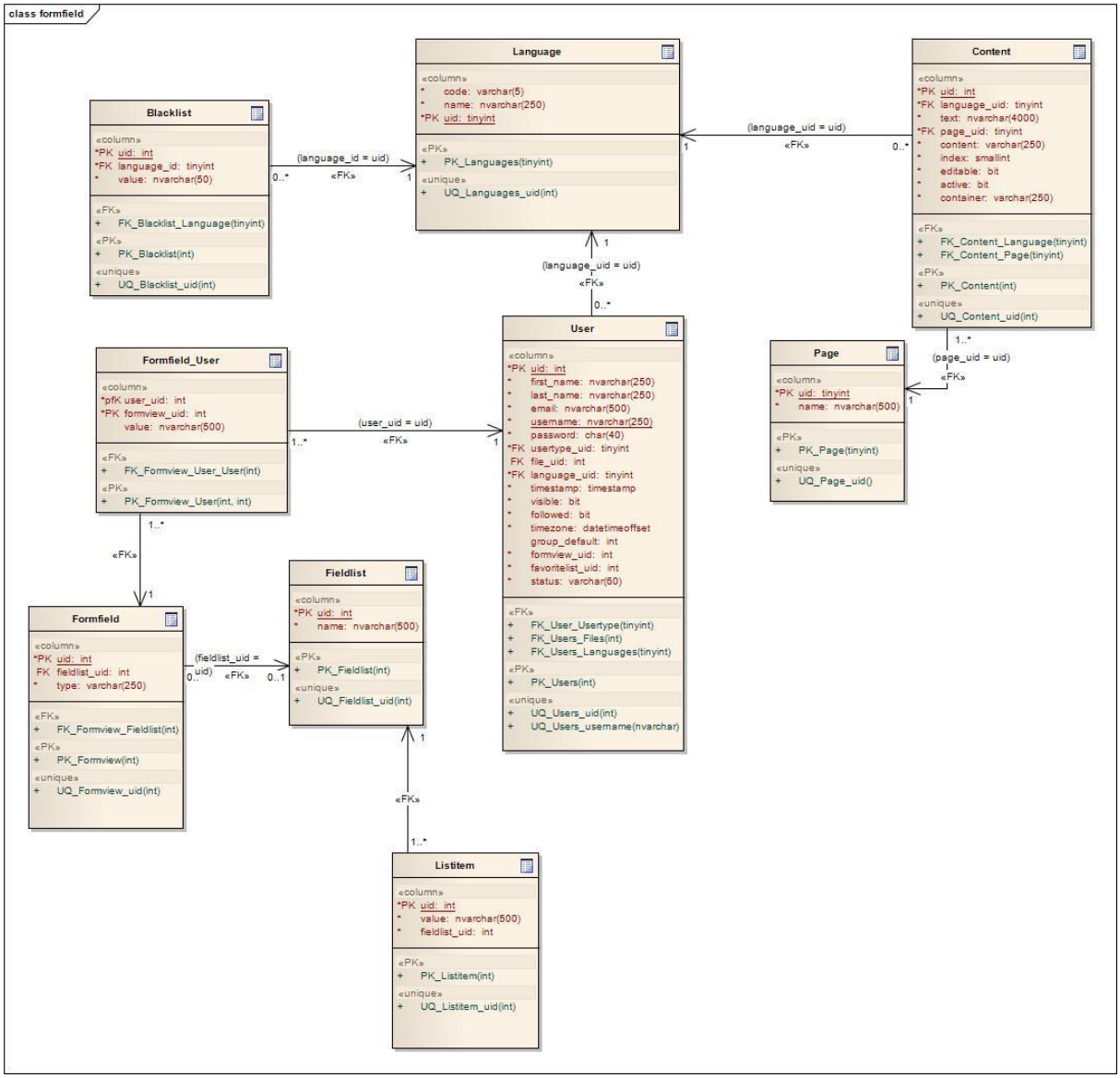
Bijlage 5 - Waarderen en categoriseren (database)



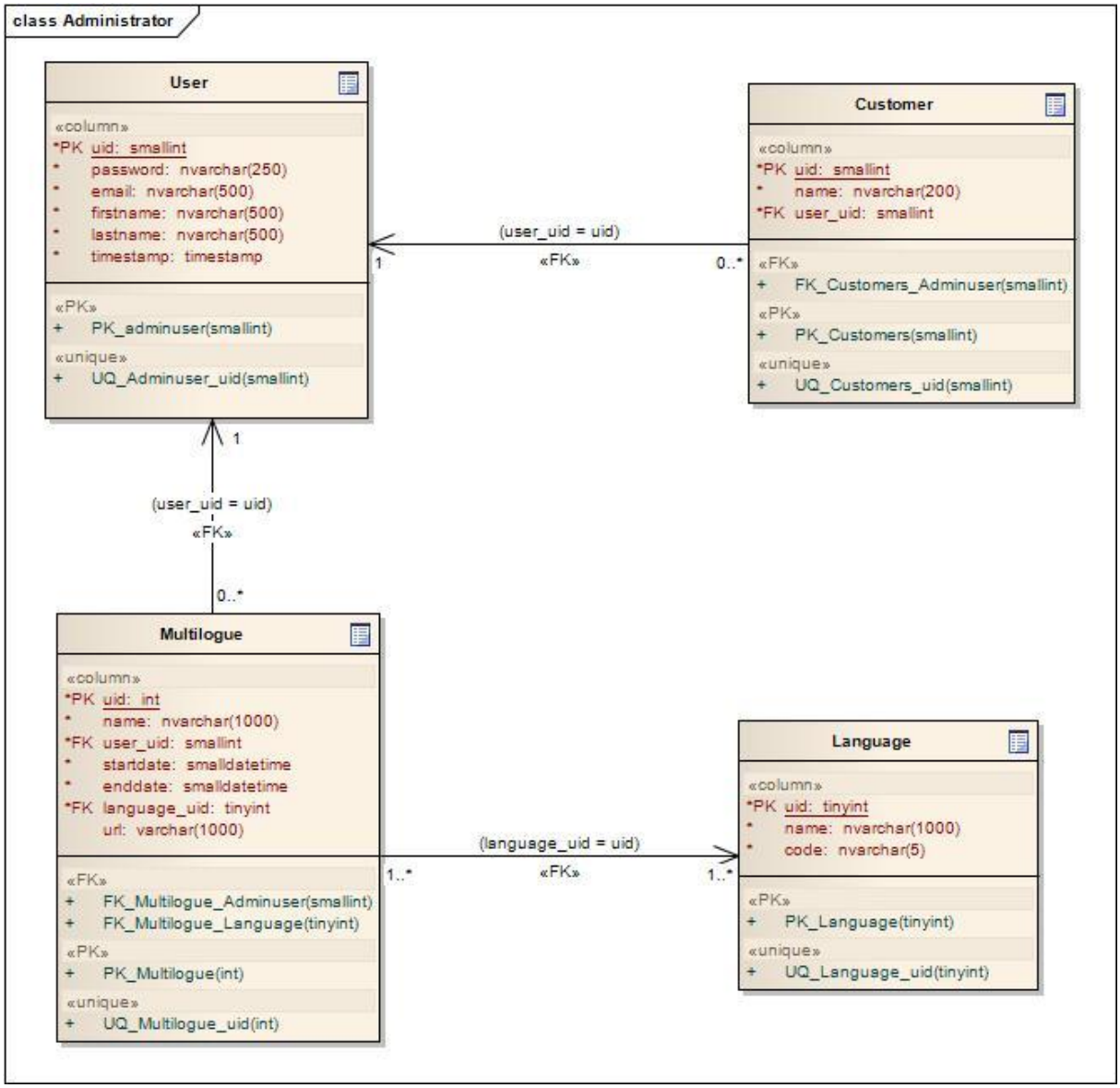
Bijlage 6 – Settings (database)



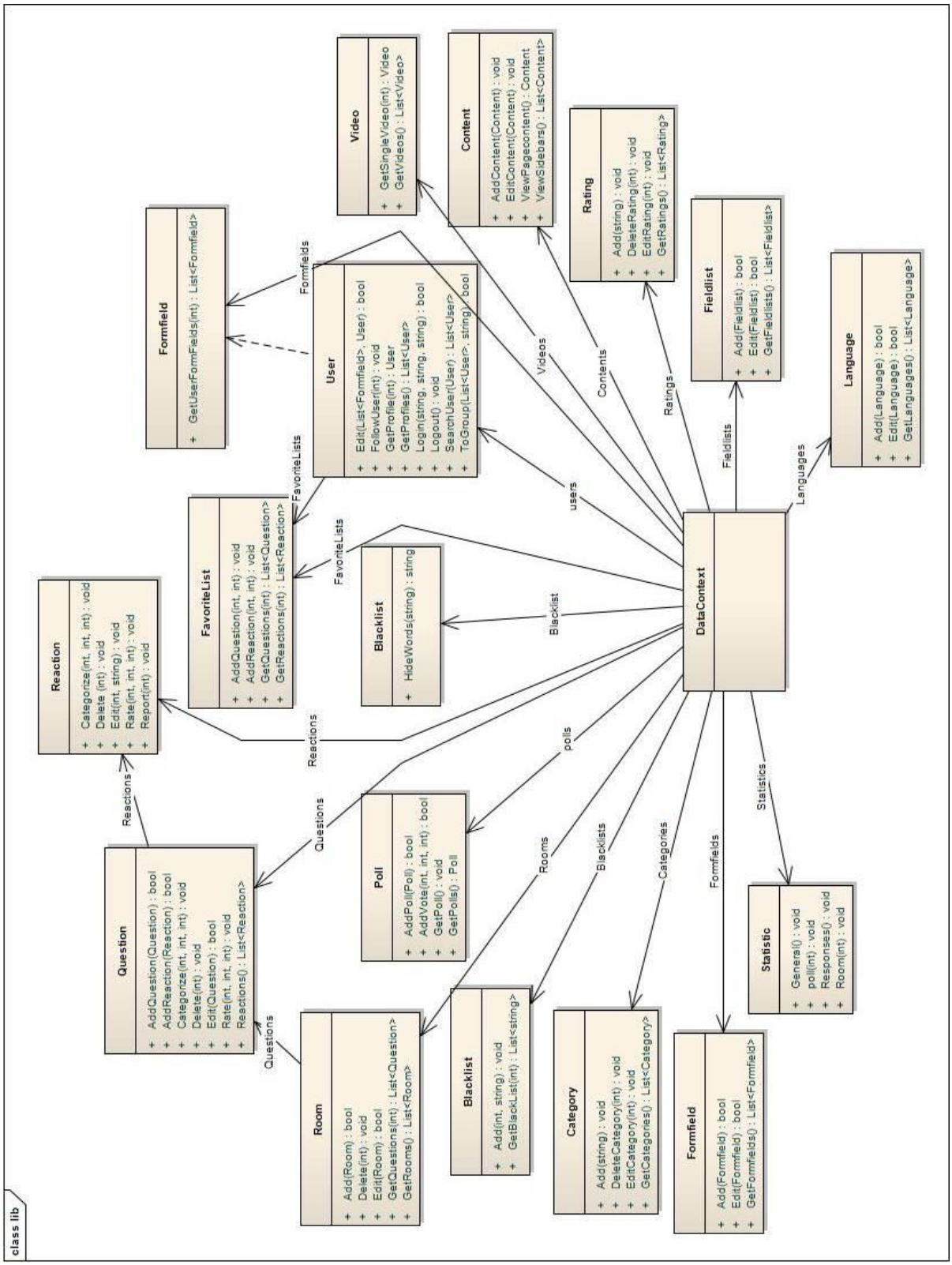
Bijlage 7 – Formfield (database)



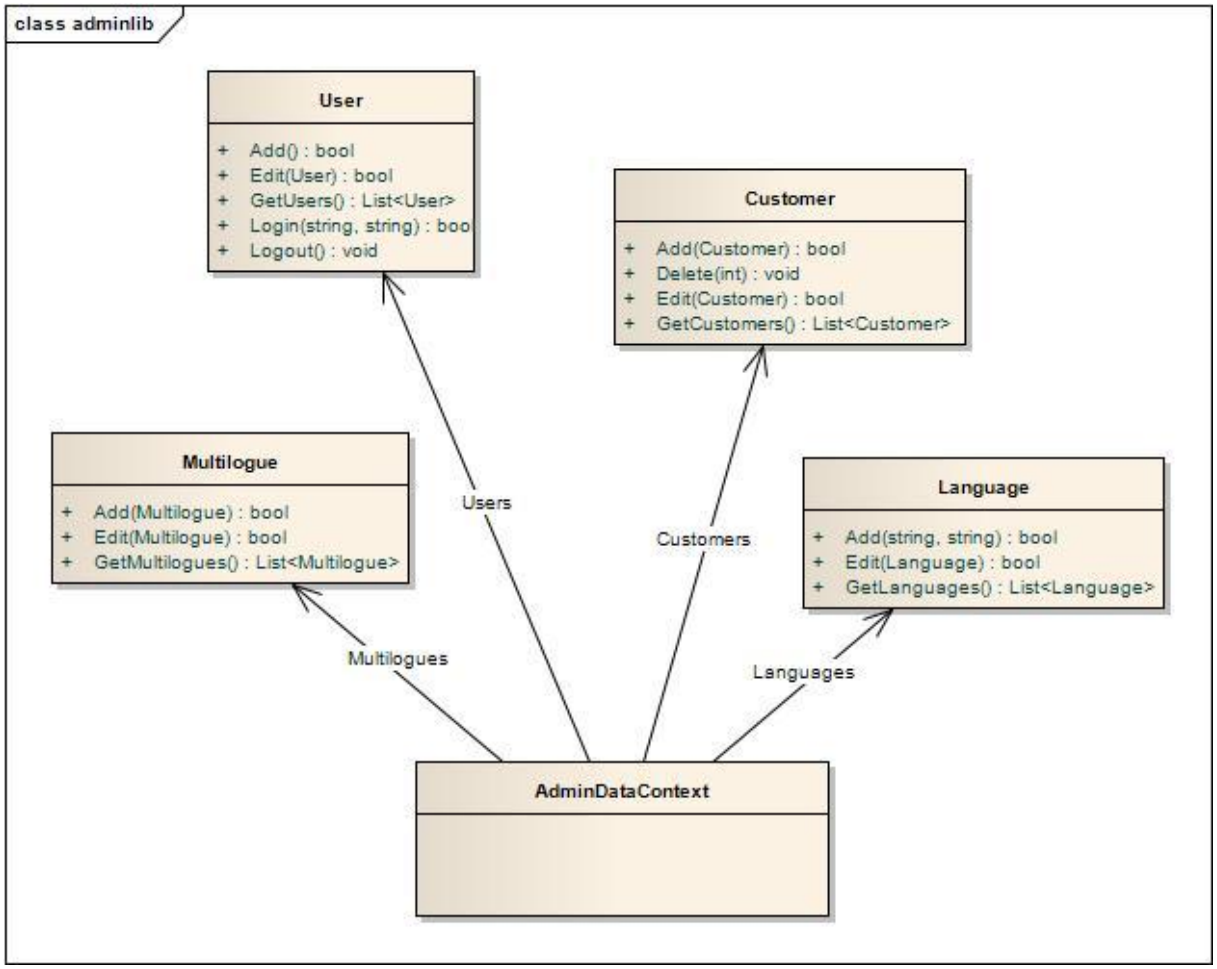
Bijlage 8 – Administrator (database)



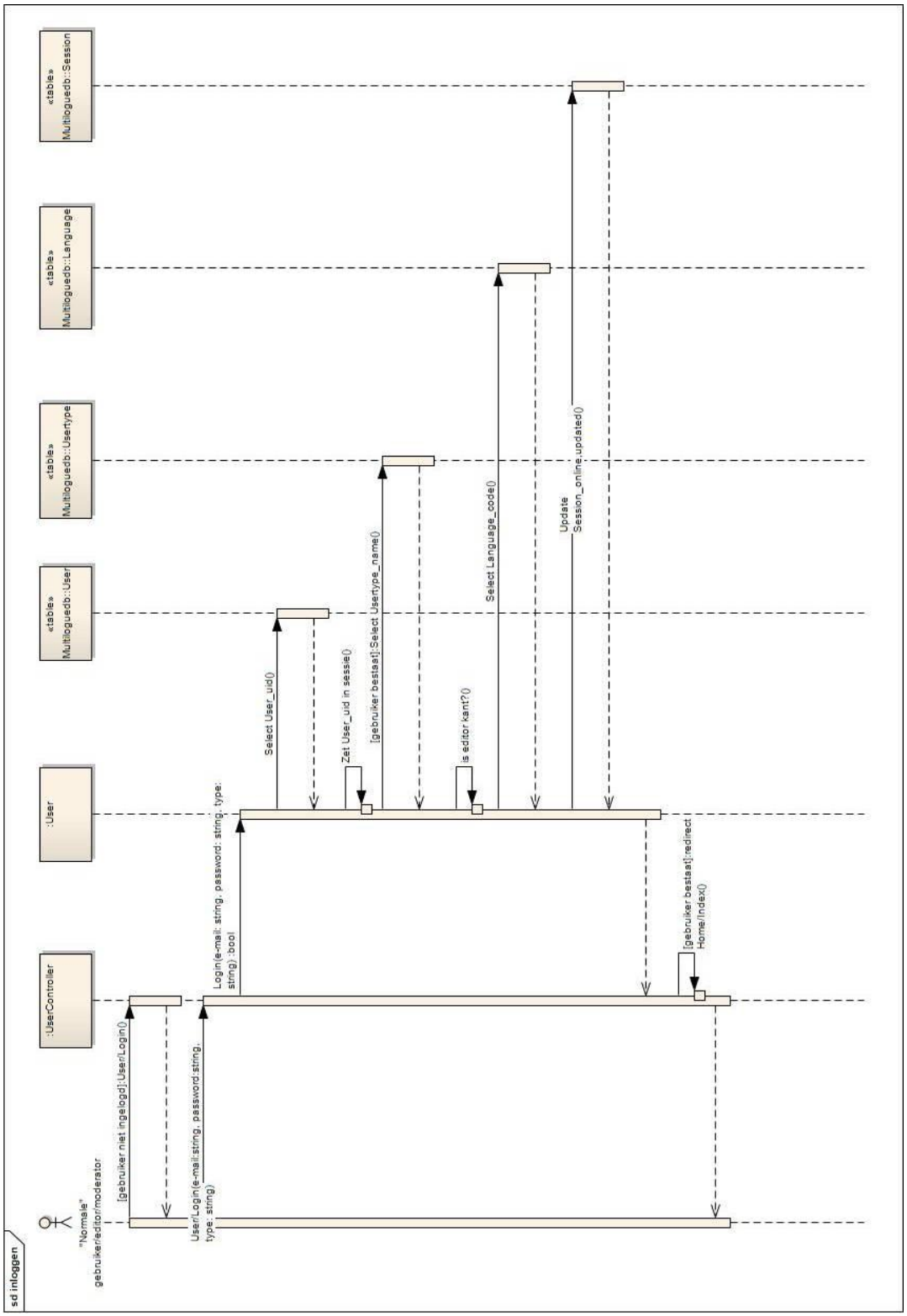
Bijlage 9 – Multilogue Lib



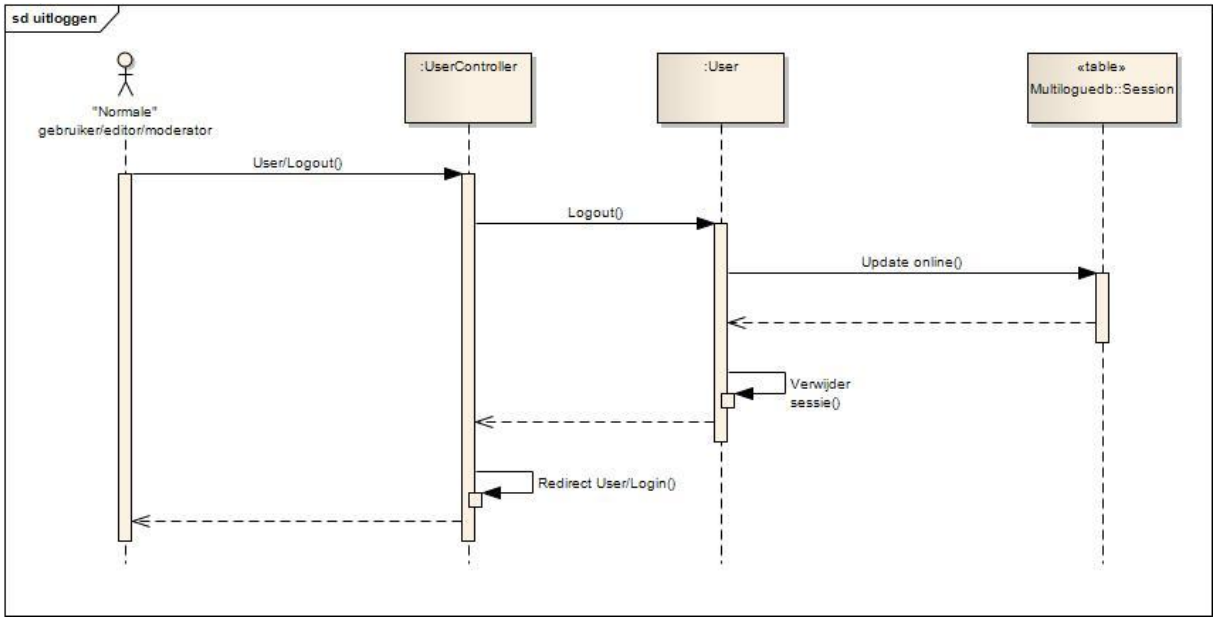
Bijlage 10 – Admin lib



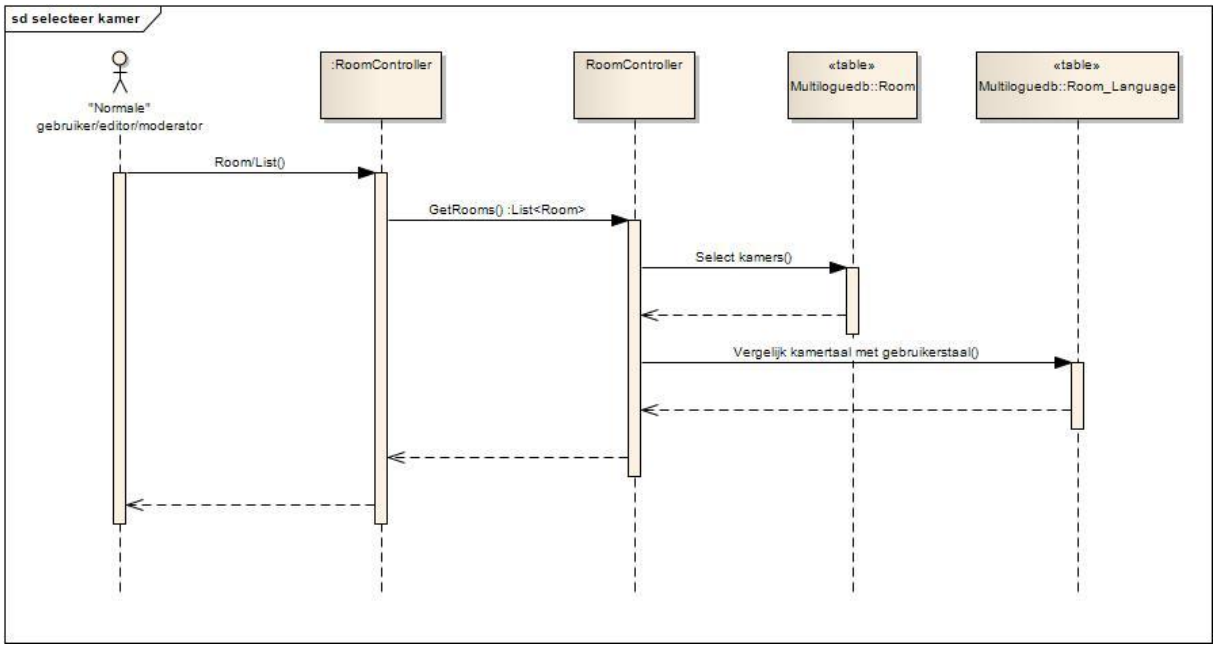
Bijlage 11 – front-end Inloggen

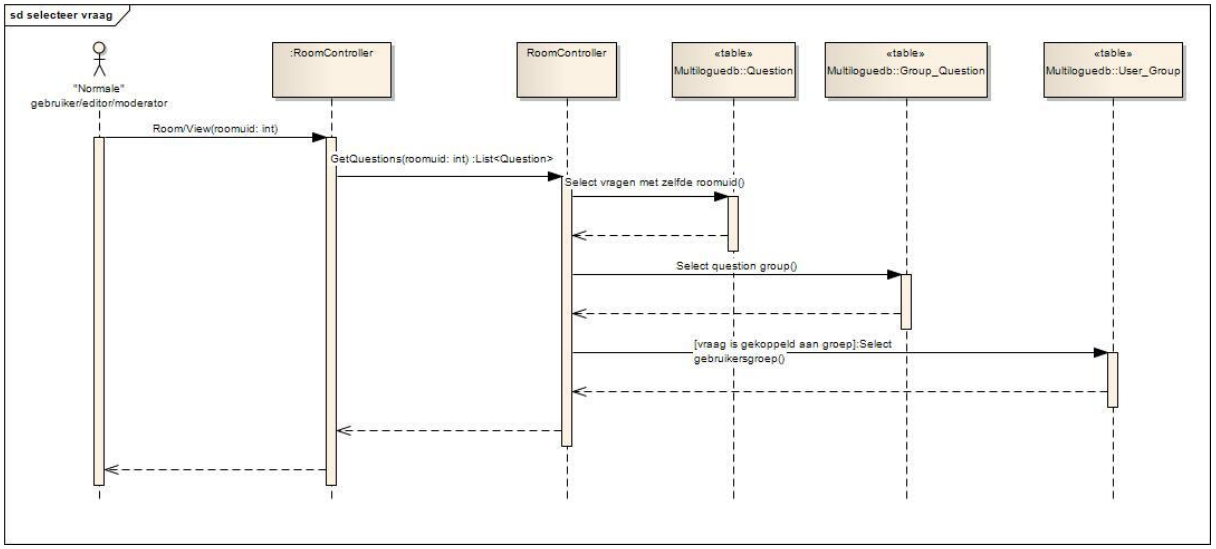


Bijlage 12 - Front-end uitloggen

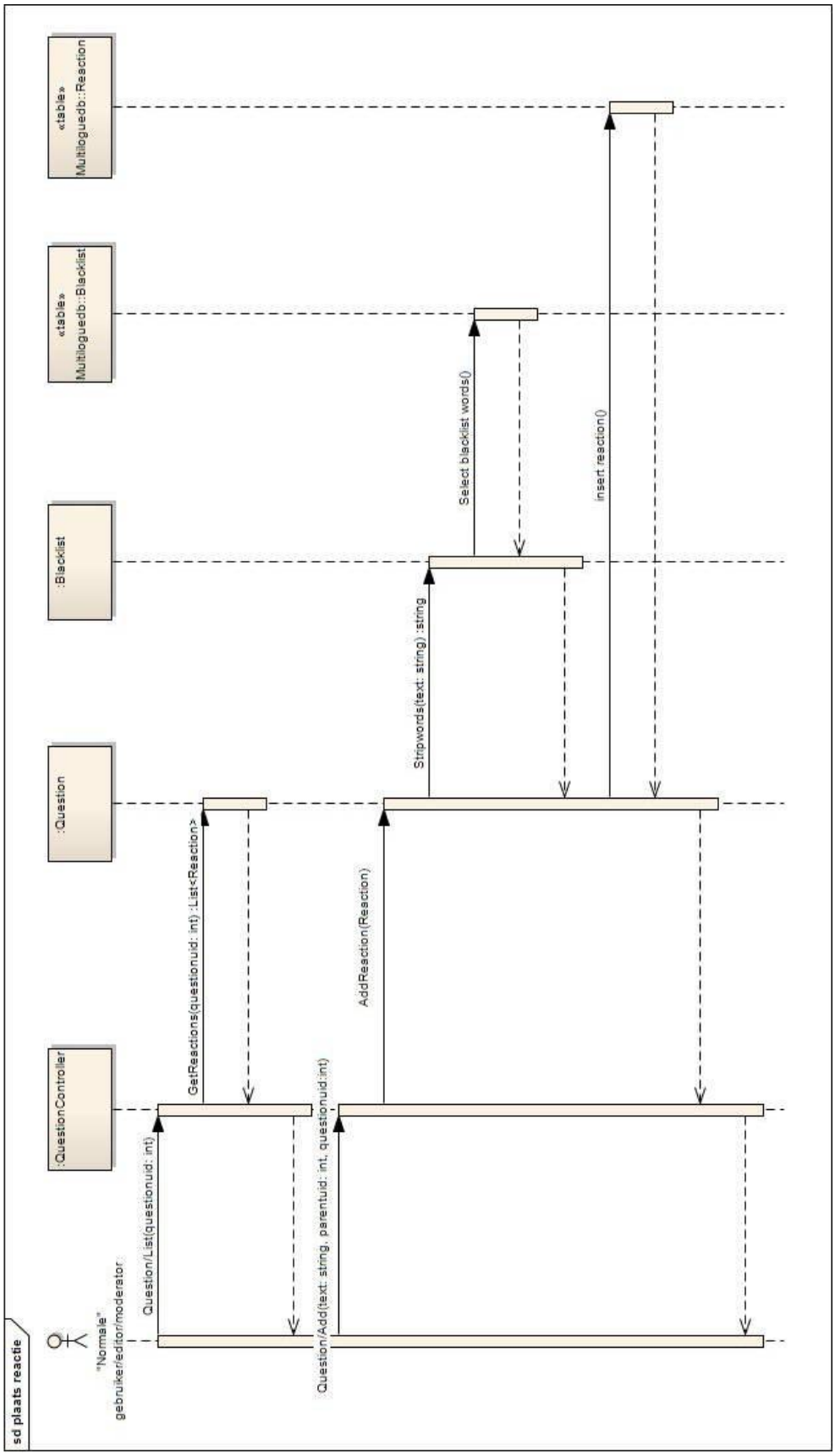


Bijlage 13 – Selecteer kamer/vraag

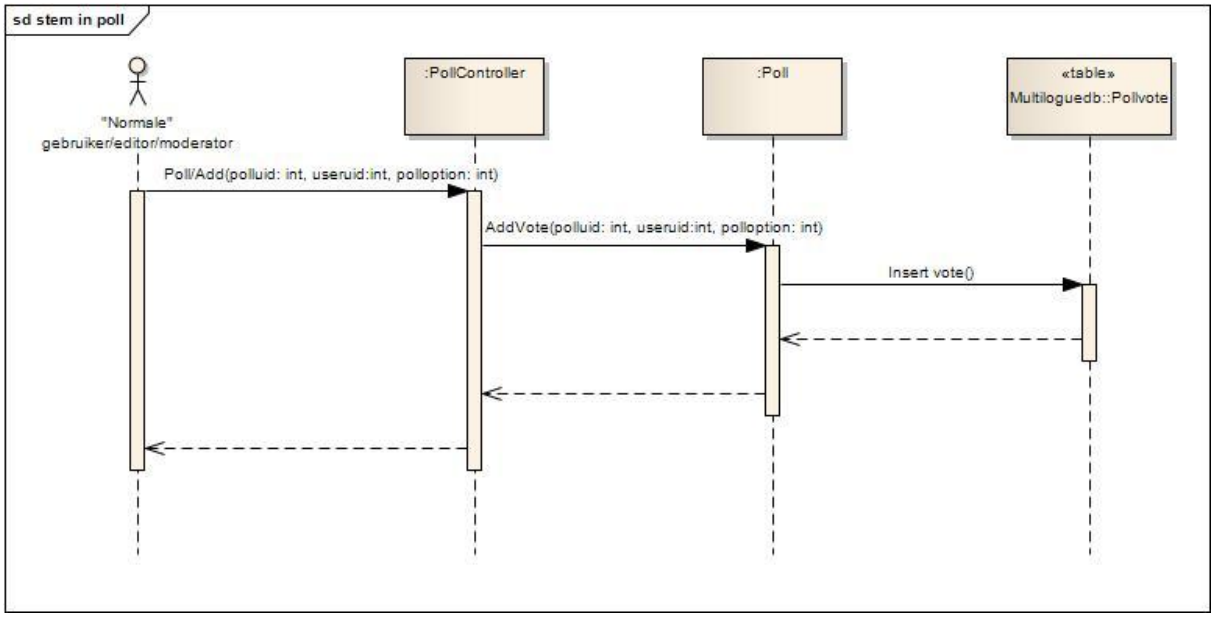




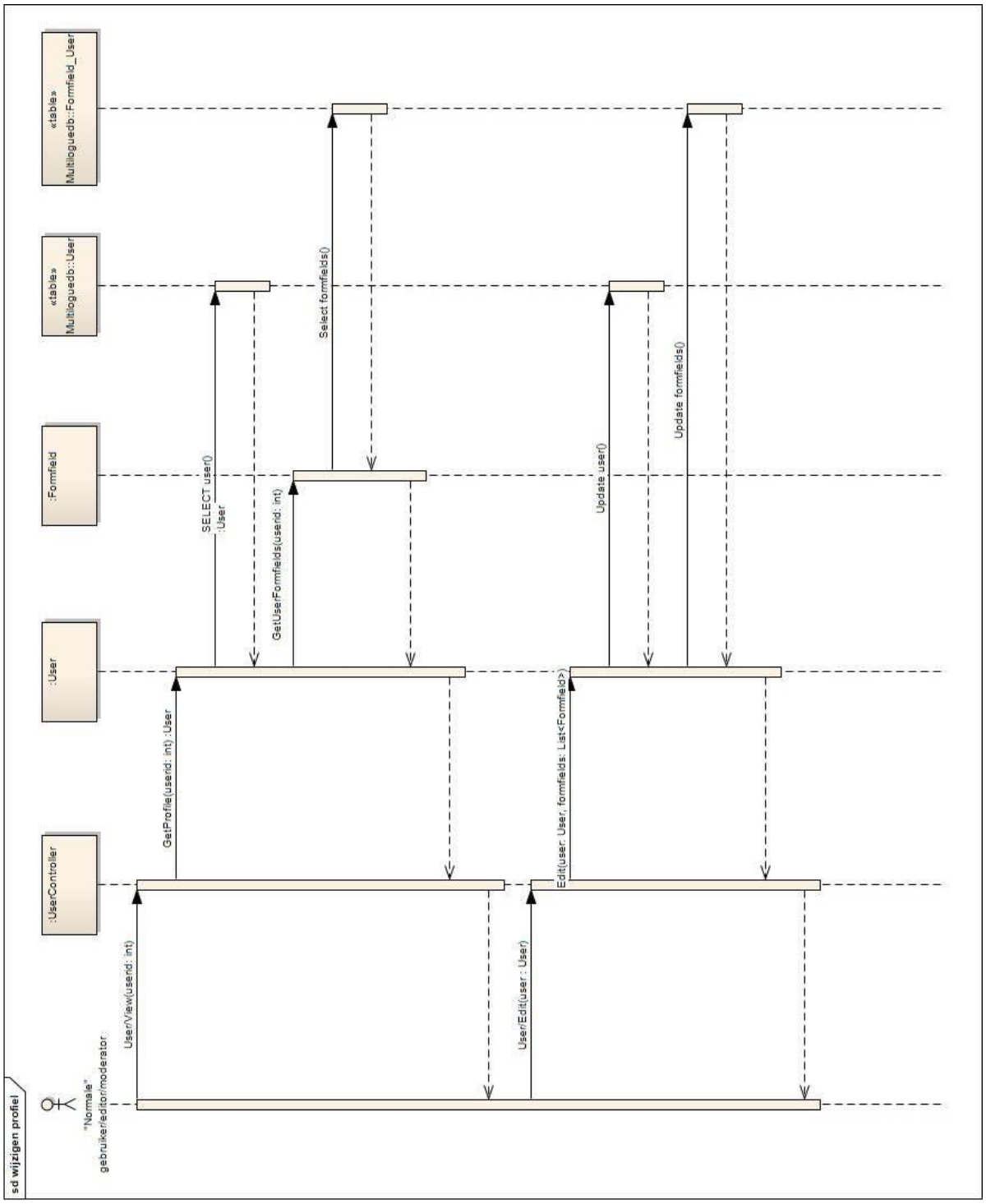
Bijlage 14 – Plaats reactie



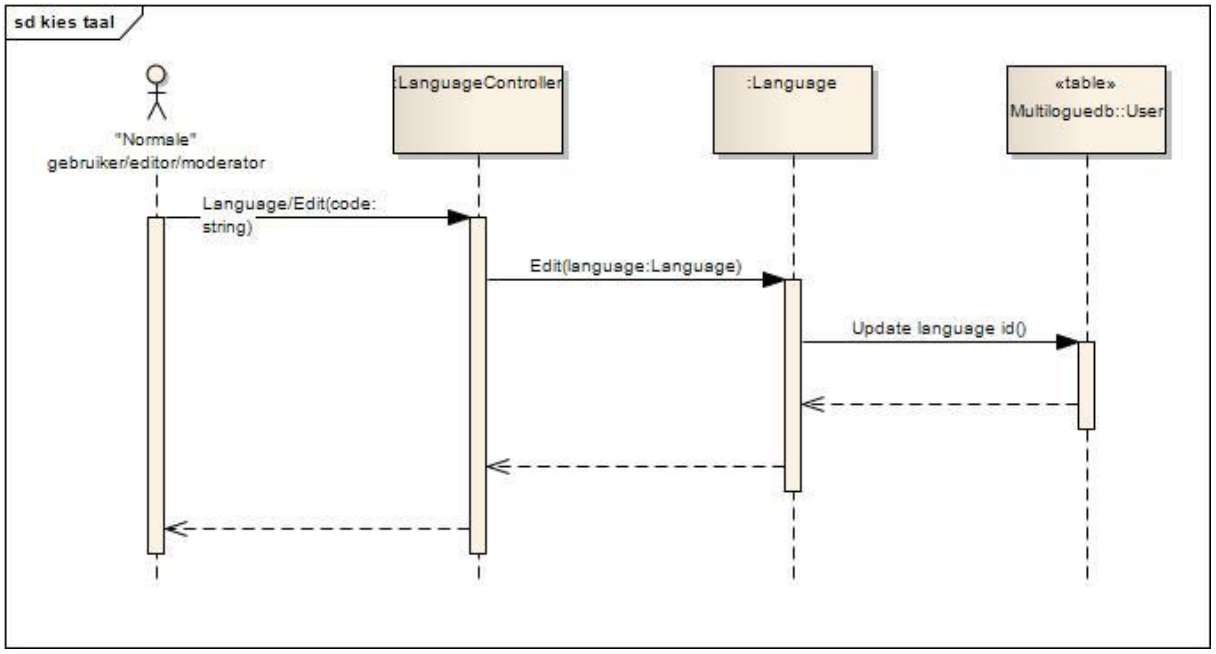
Bijlage 15 – Stem in poll



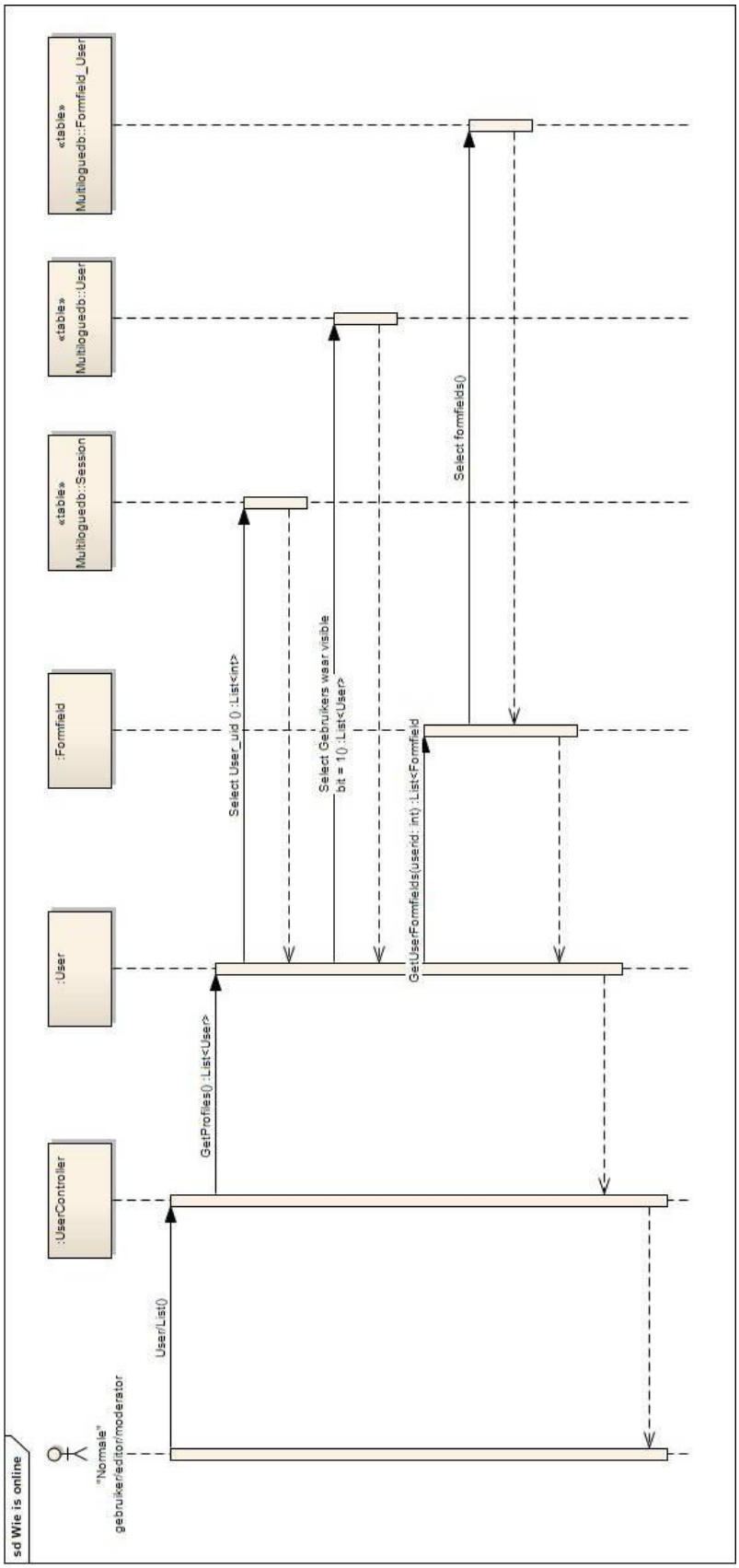
Bijlage 16 – Wijzigen profiel



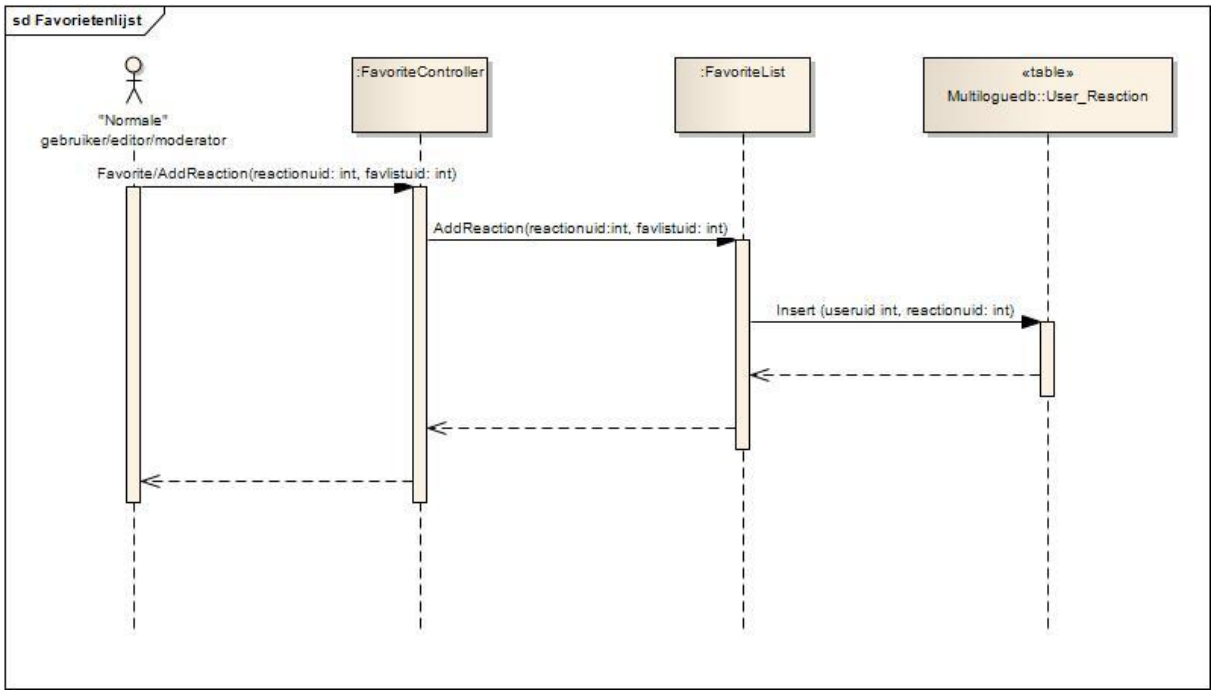
Bijlage 17 – Taal kiezen



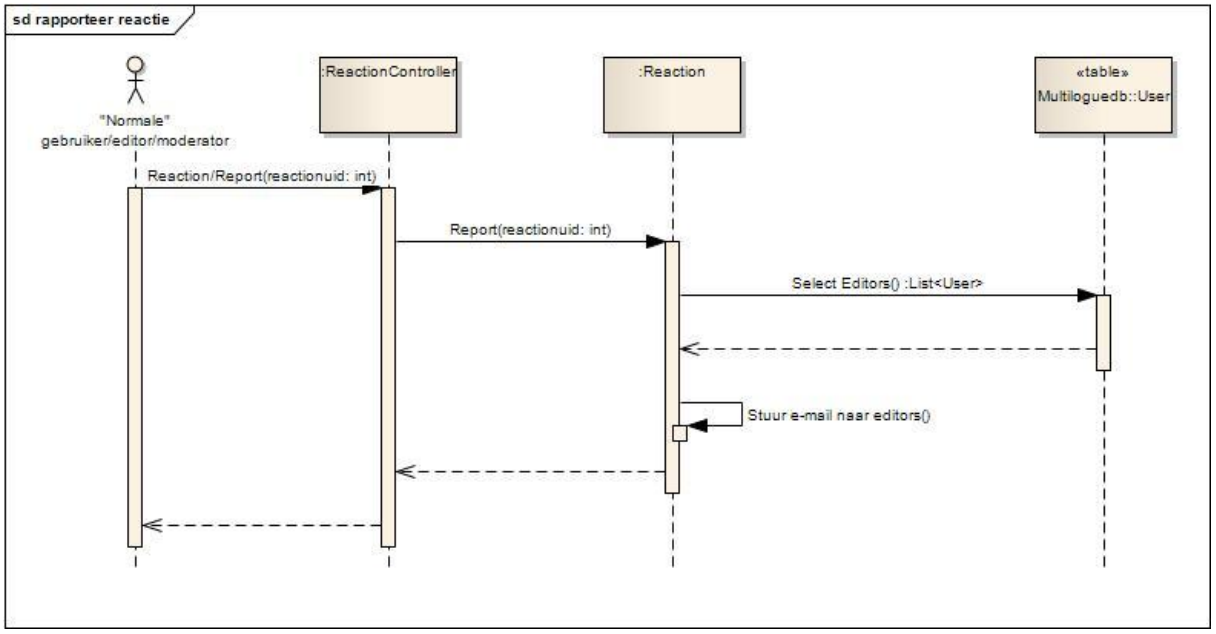
Bijlage 18 – Wie is online



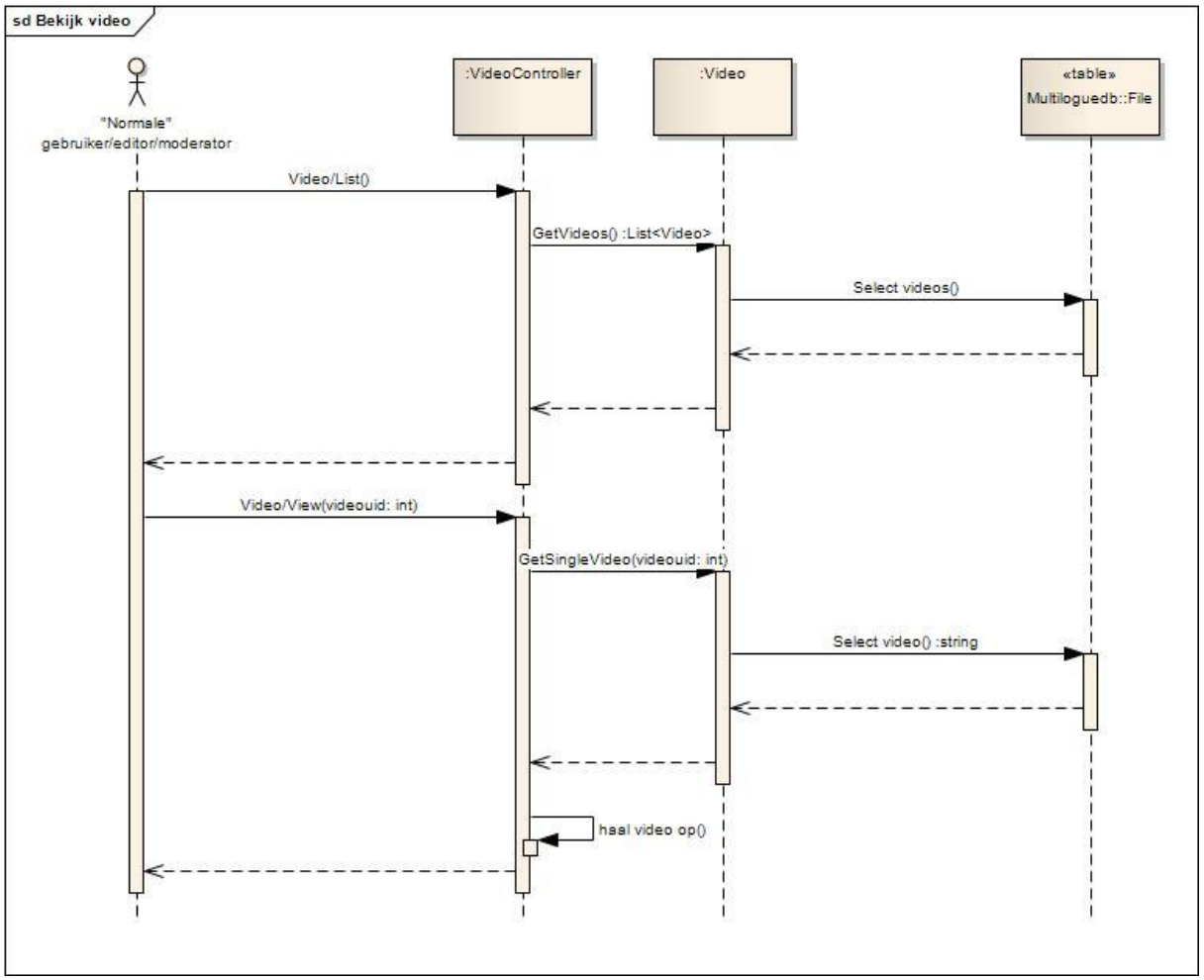
Bijlage 19 – Voeg reactie toe aan favorietenlijst



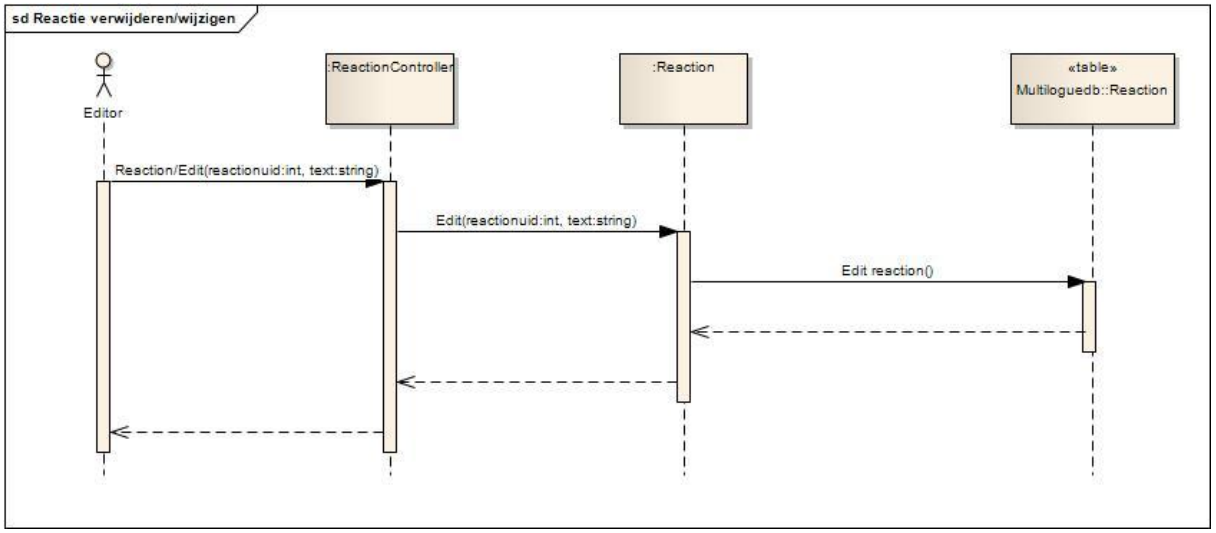
Bijlage 20 – Rapporteer reactie



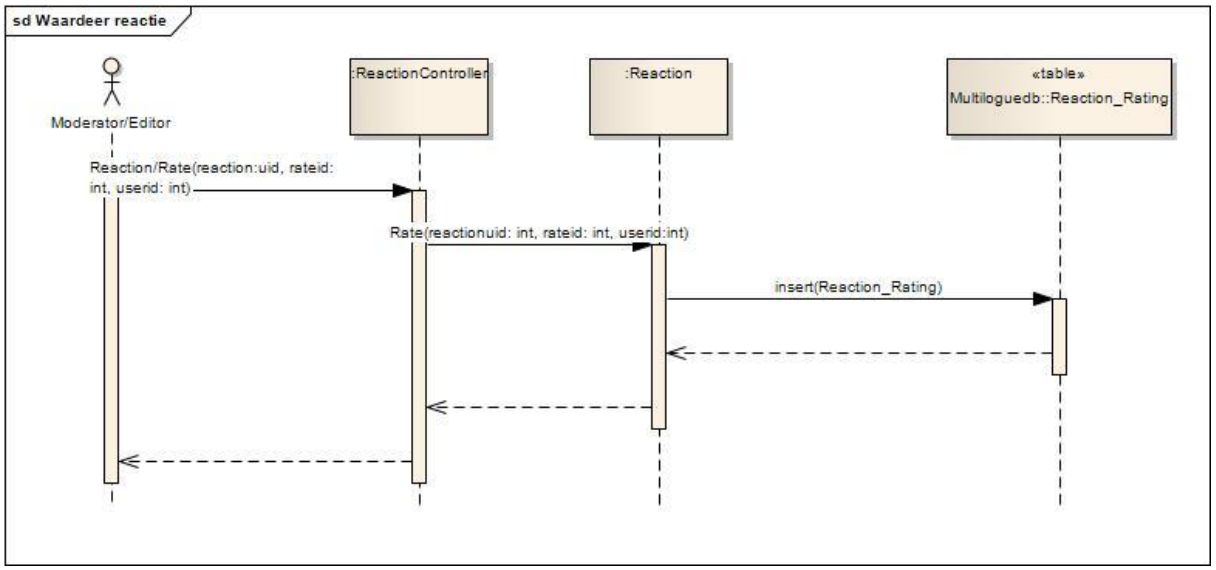
Bijlage 21 – bekijk video



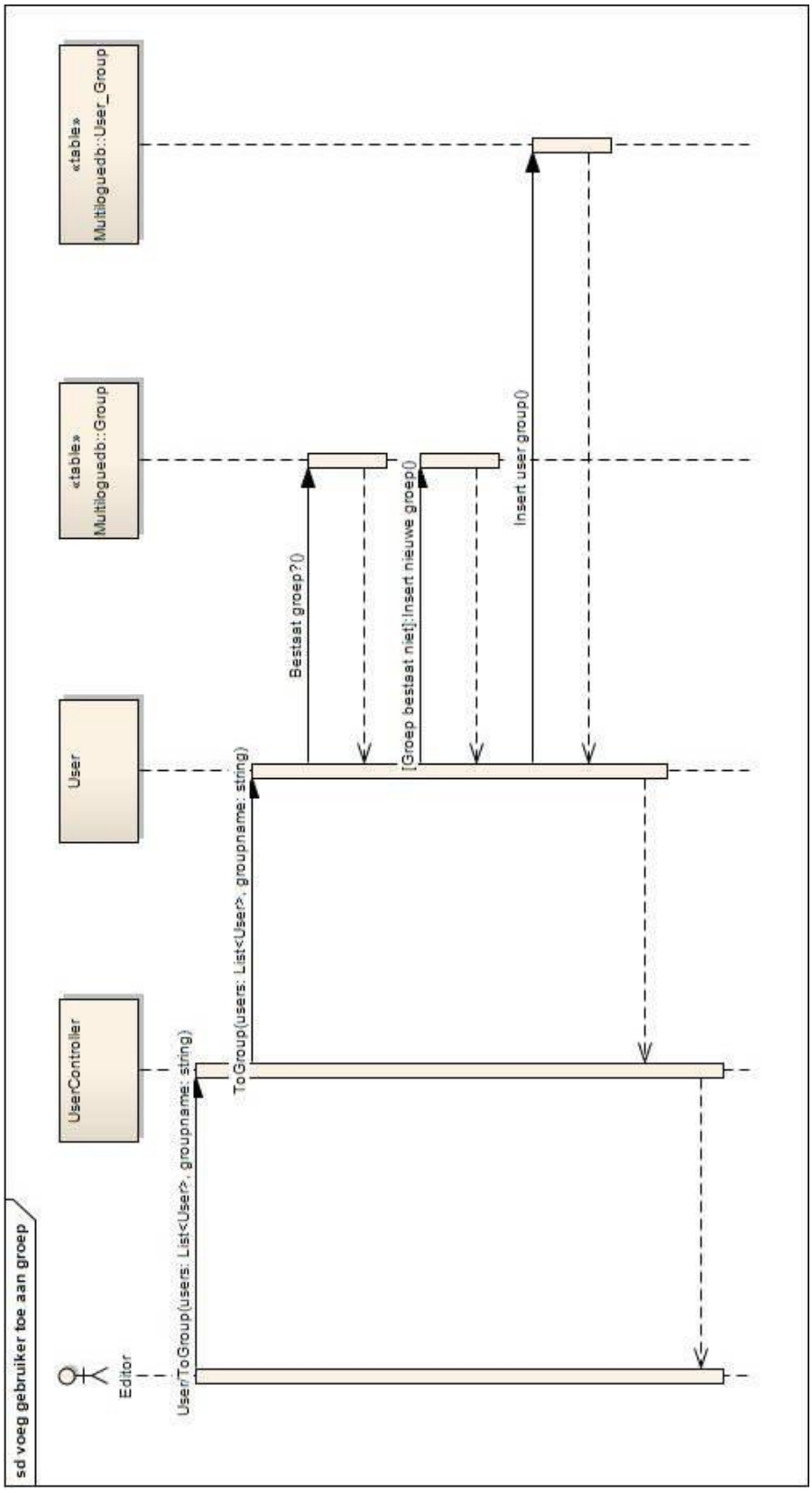
Bijlage 22 – Reacties wijzigen



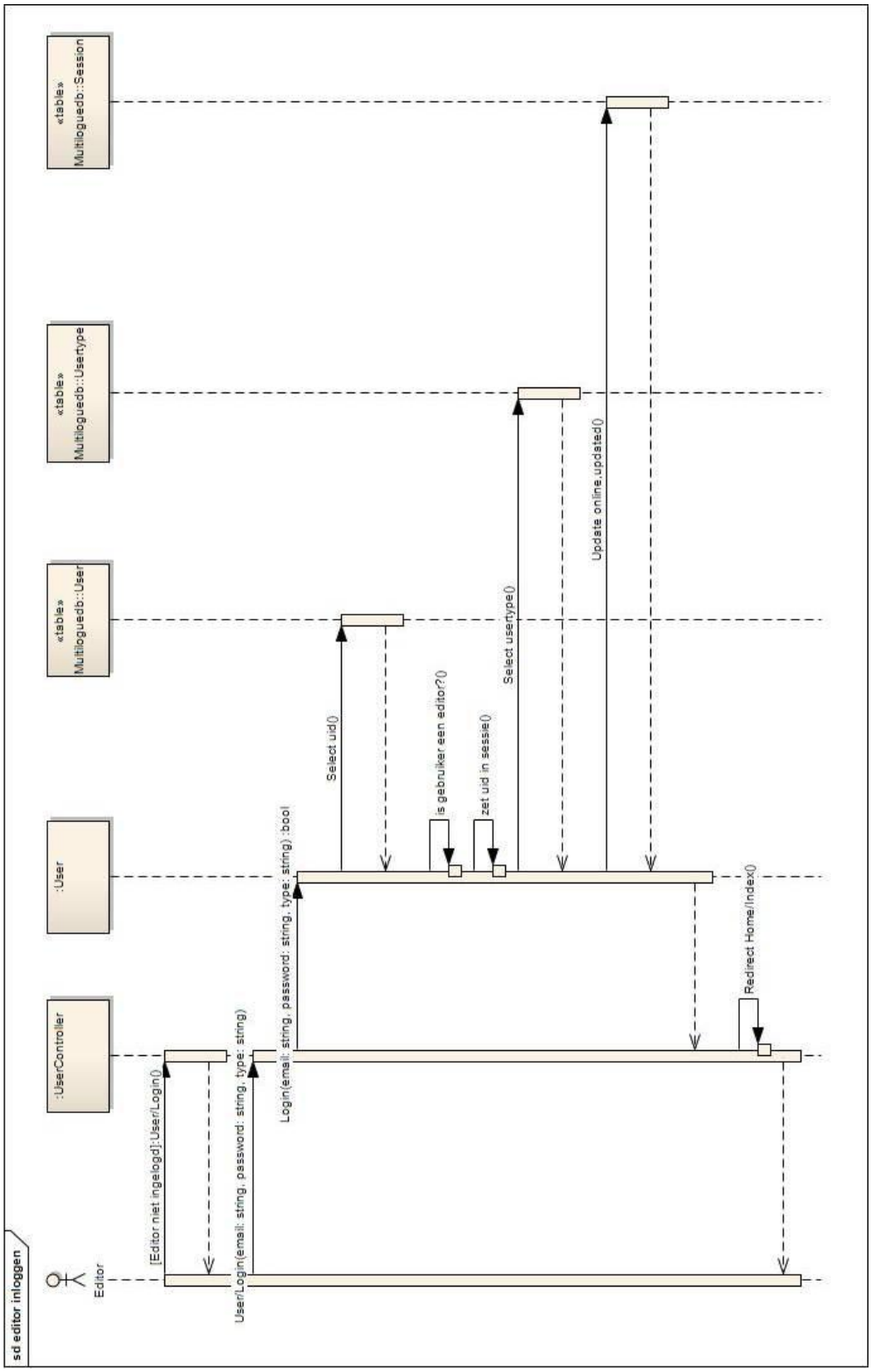
Bijlage 23 – Waardeer reactie



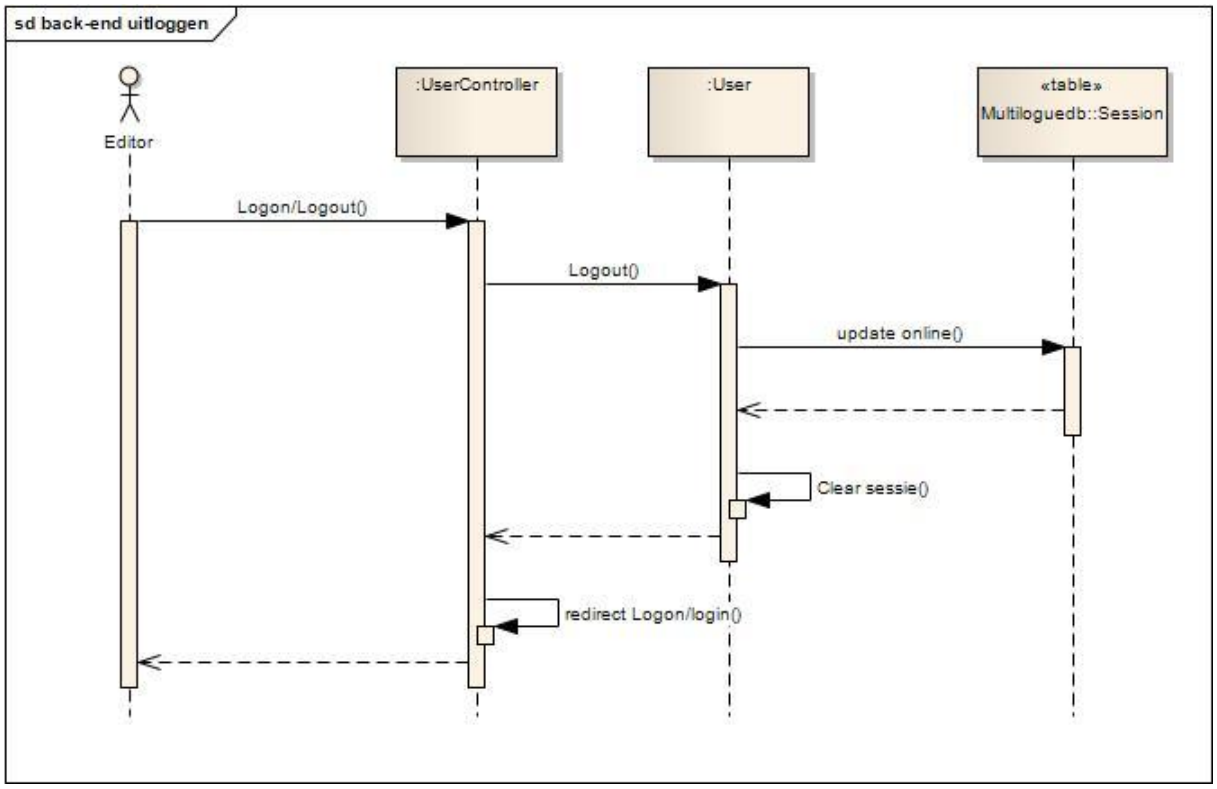
Bijlage 24 – voeg gebruiker toe aan groep



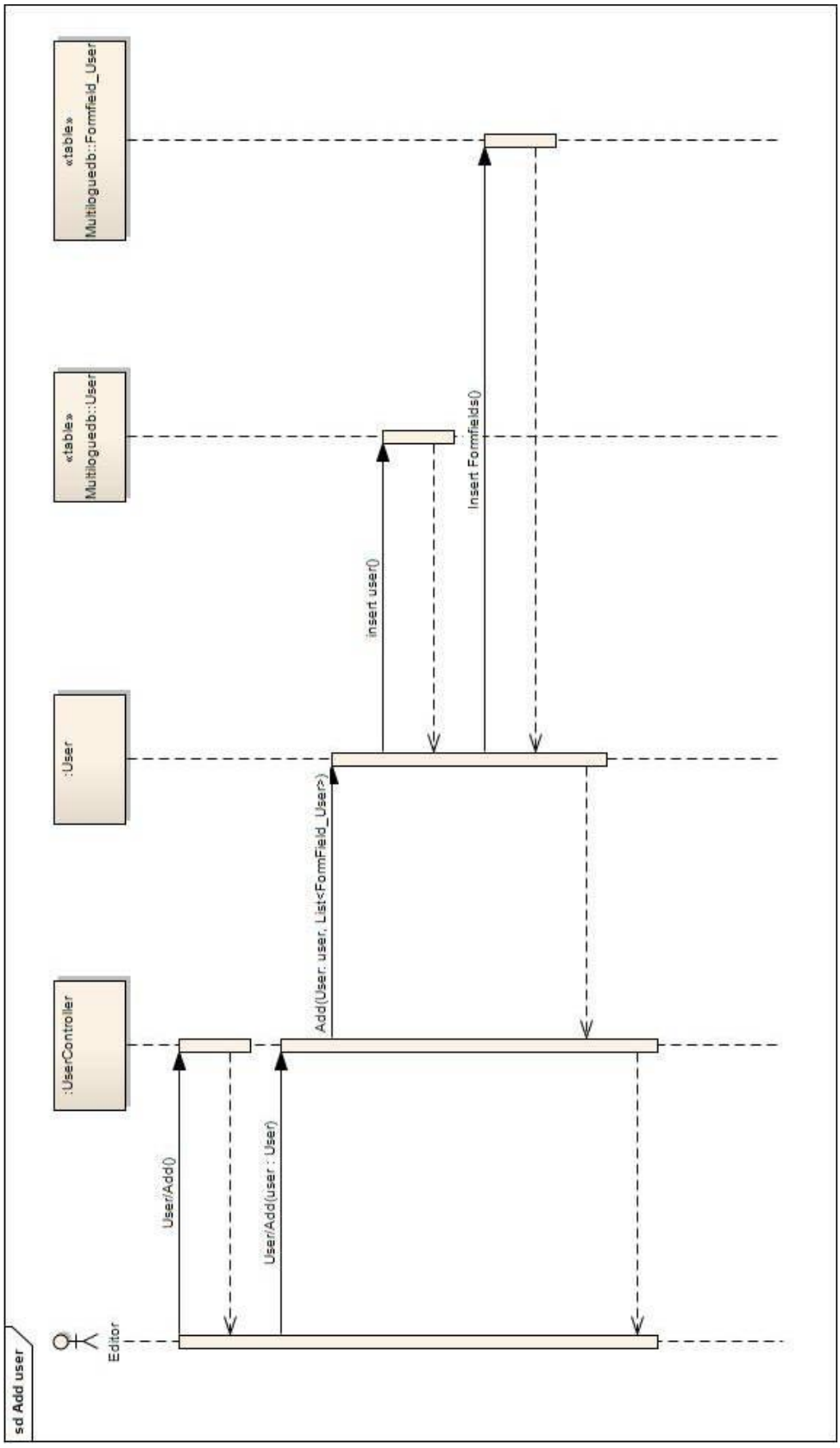
Bijlage 25 – Editor inloggen



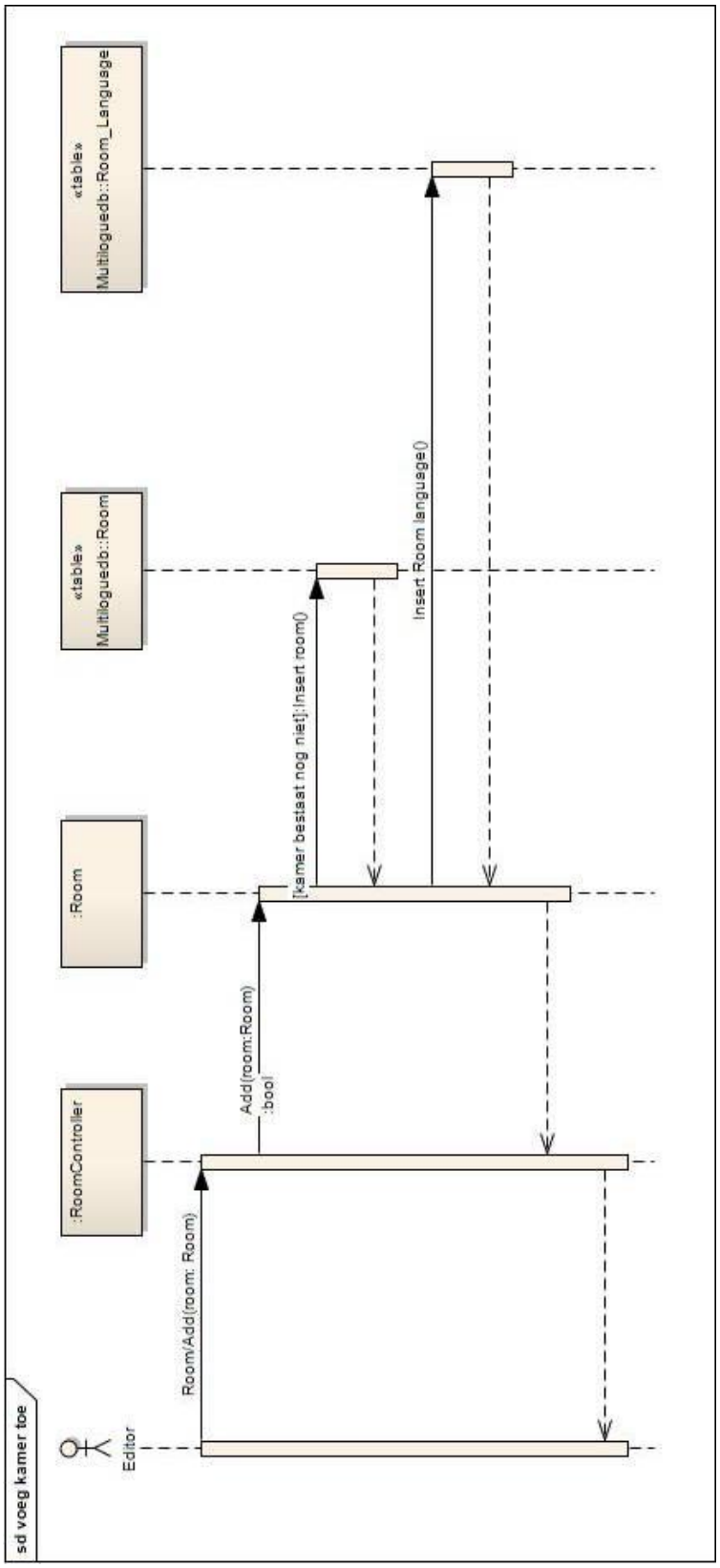
Bijlage 26 – Editor uitloggen



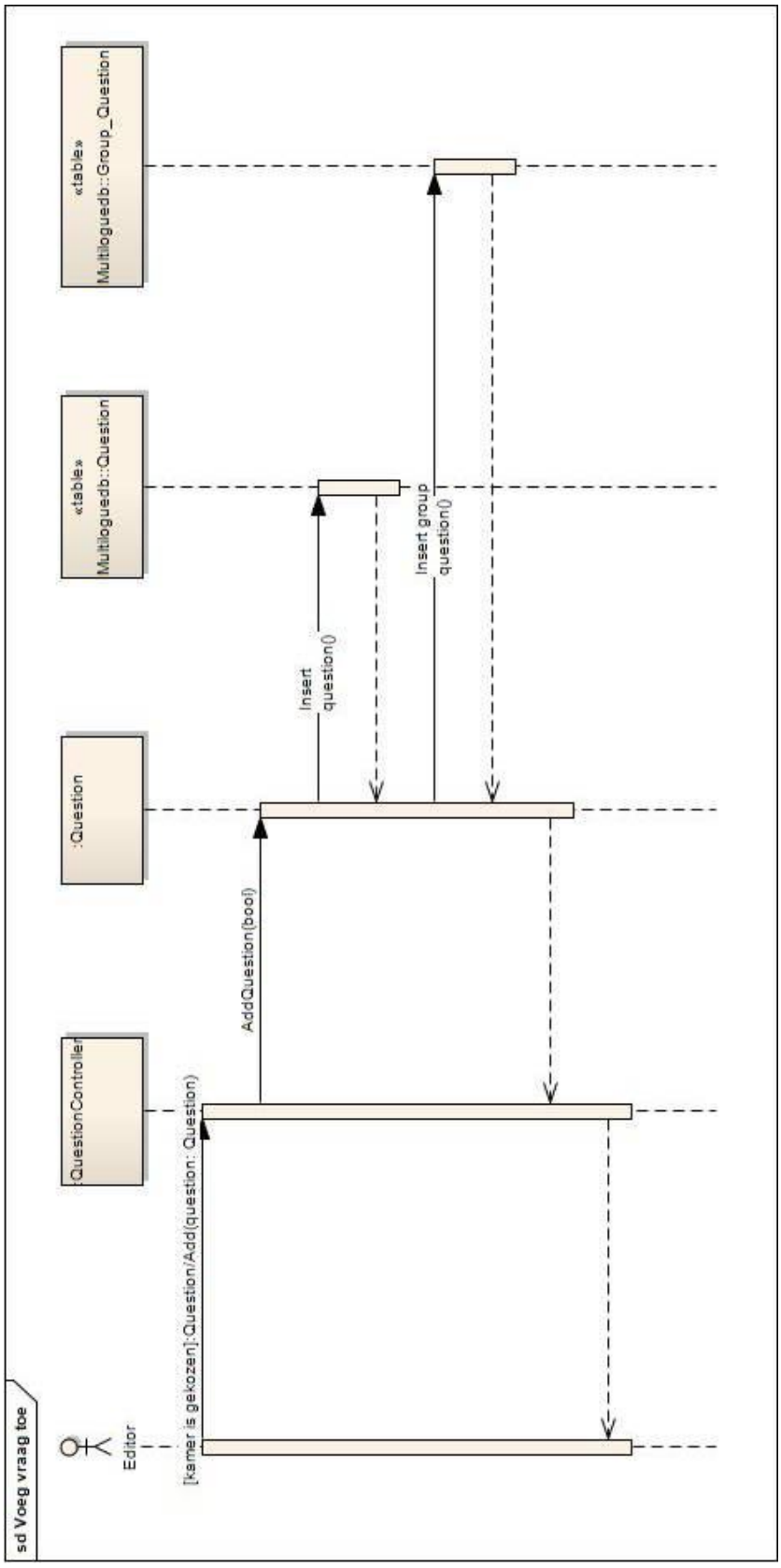
Bijlage 27 – Gebruiker toevoegen



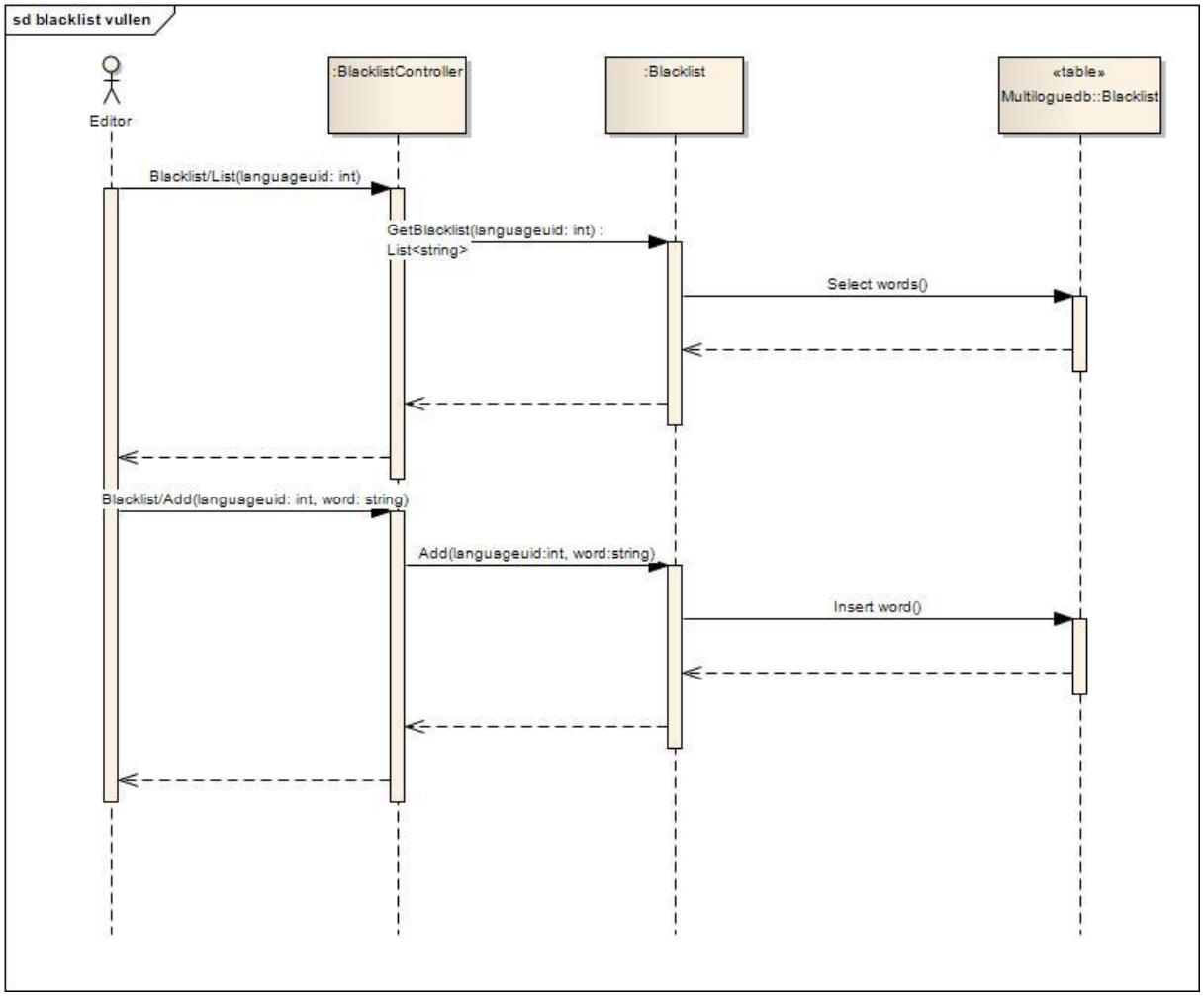
Bijlage 28 – Voeg kamer toe



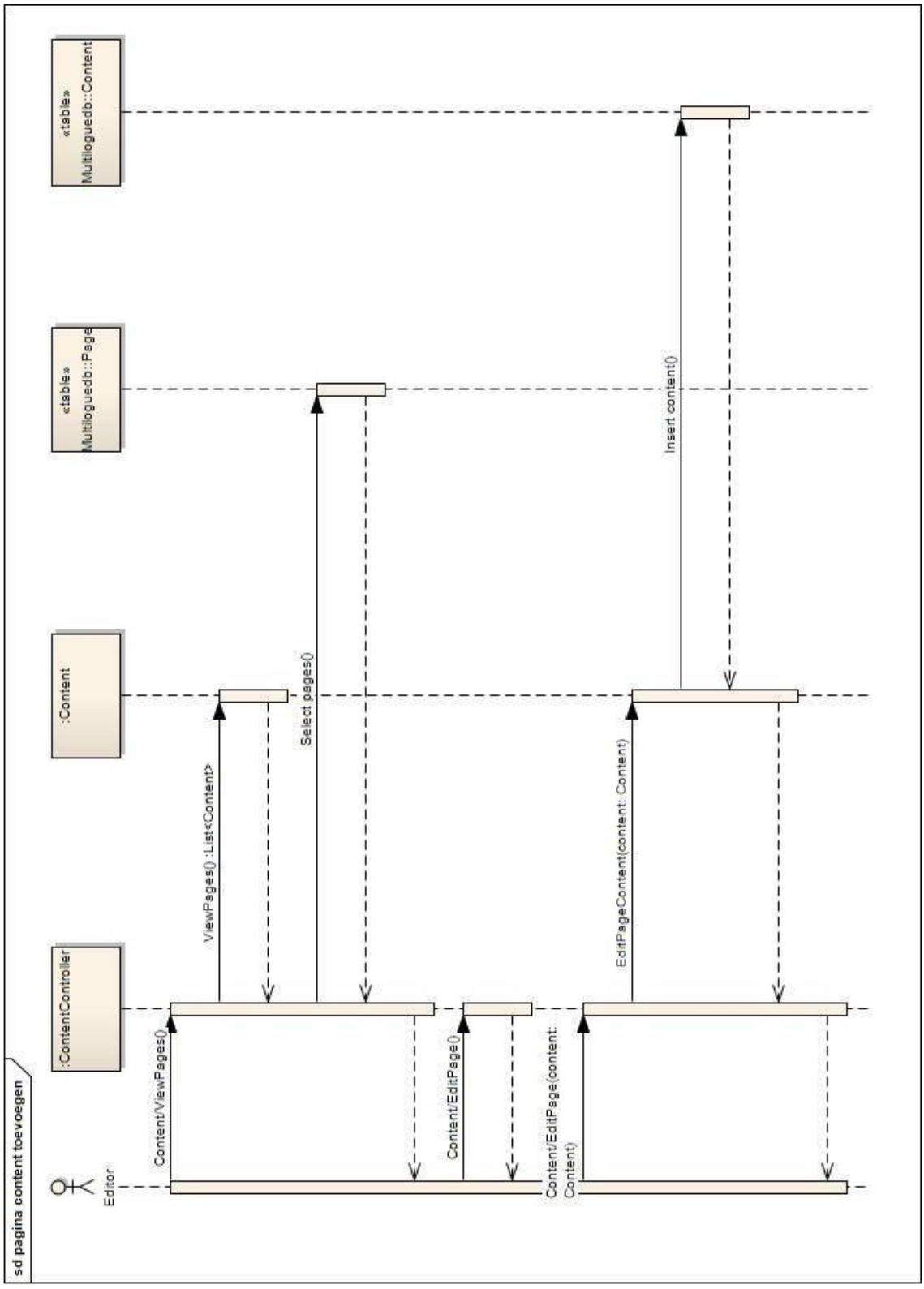
Bijlage 29 – Voeg vraag toe



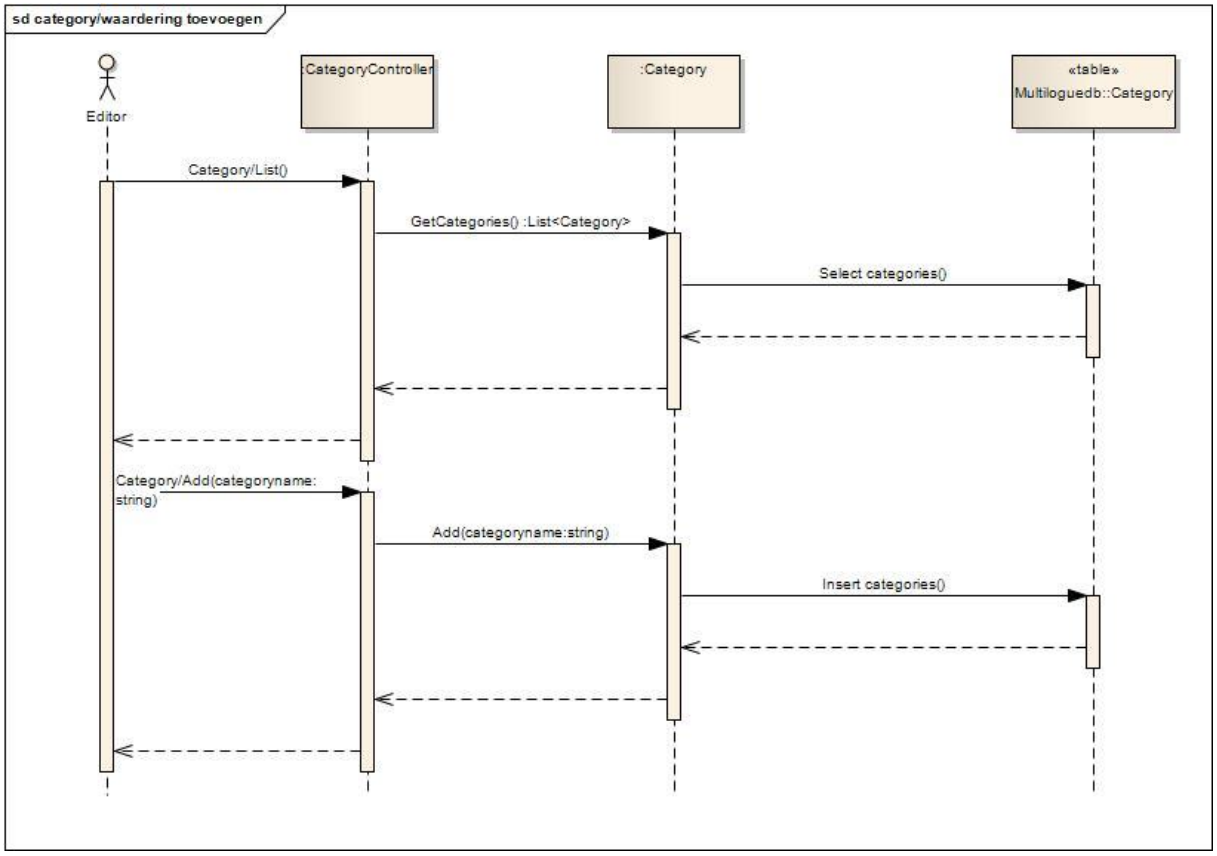
Bijlage 30 – Vul blacklist



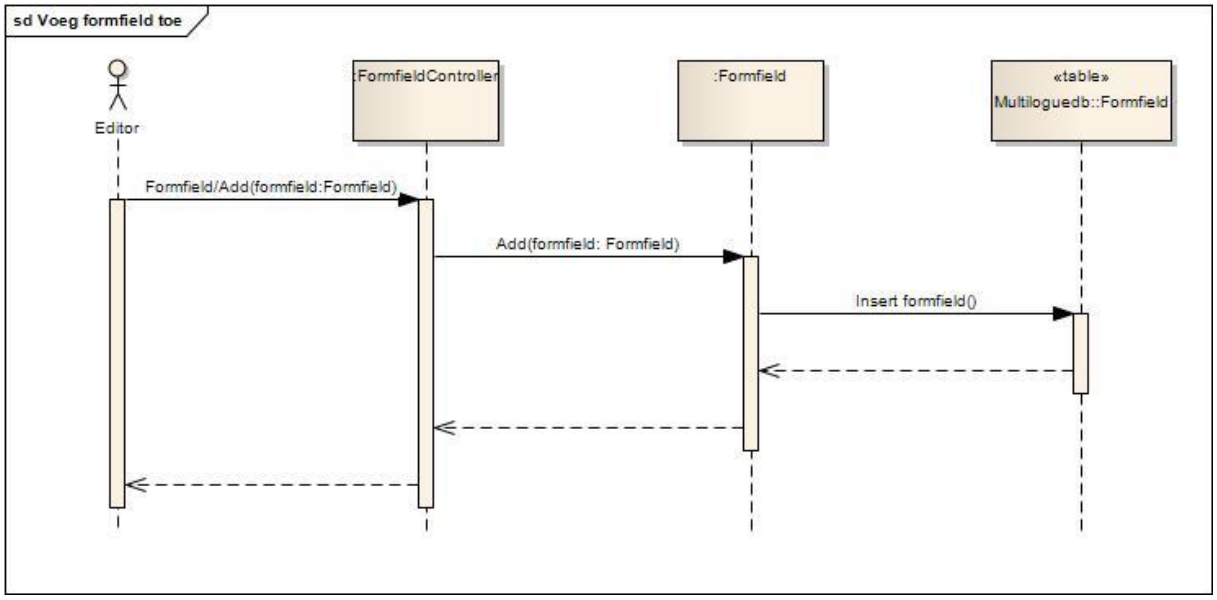
Bijlage 31 - Pagina content toevoegen



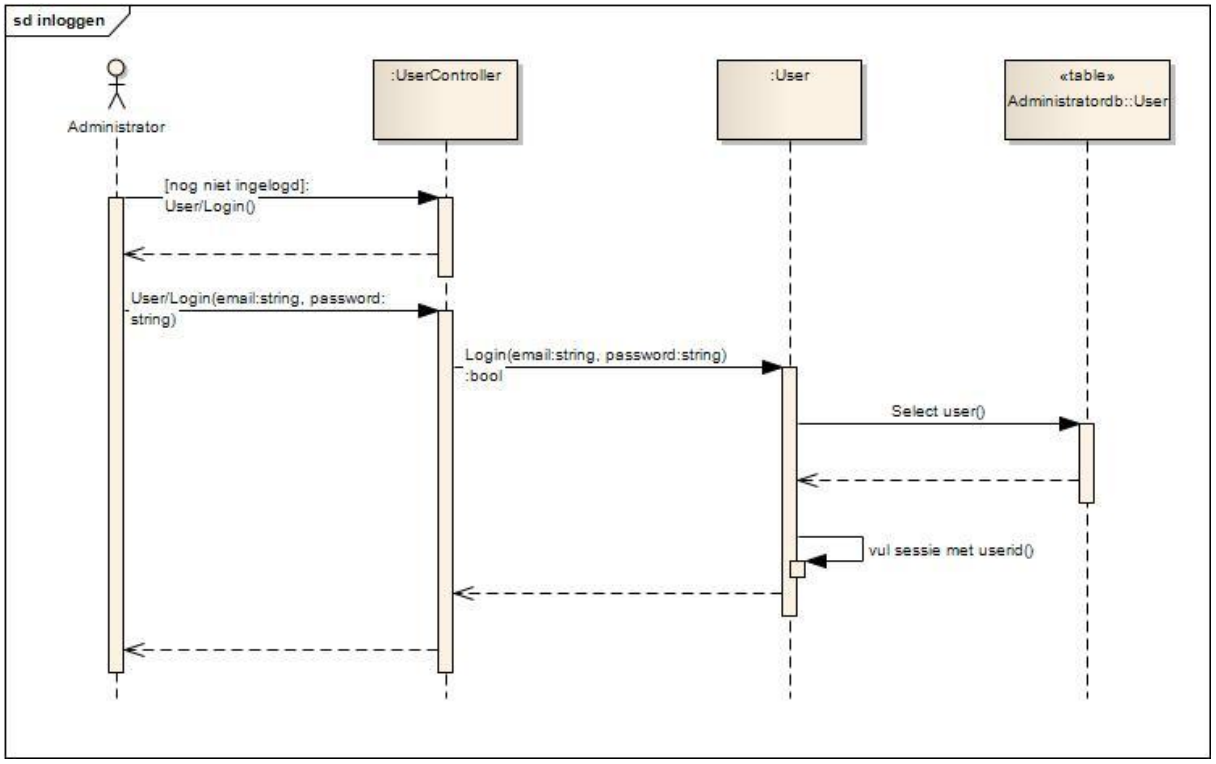
Bijlage 32 - Voeg categorie/waardering toe



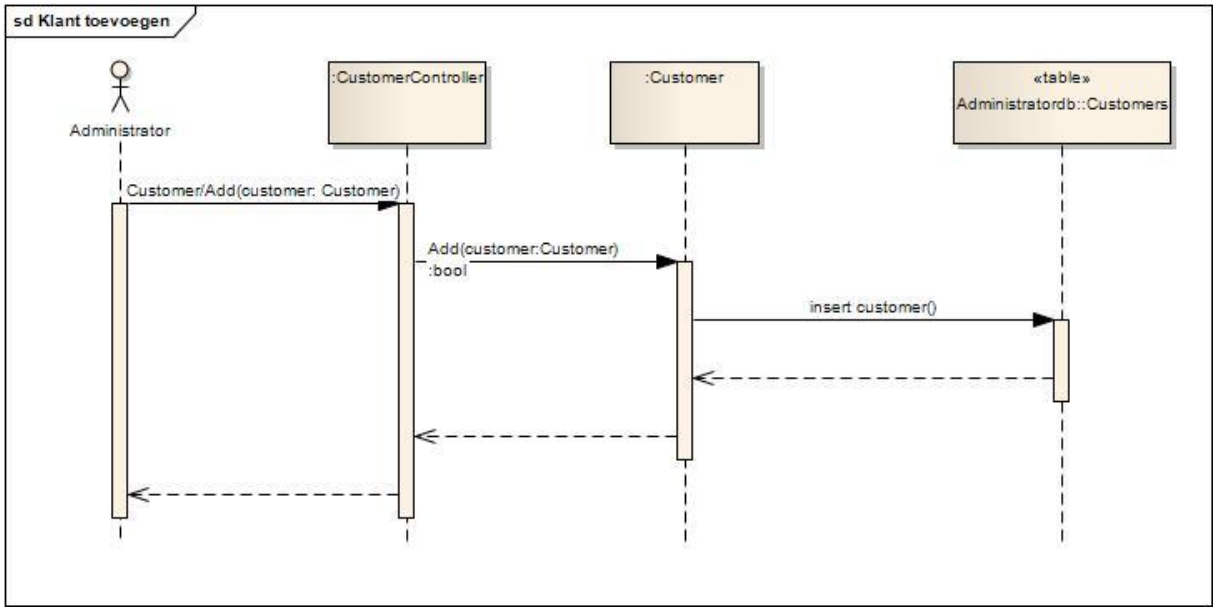
Bijlage 33 - Voeg formfield toe



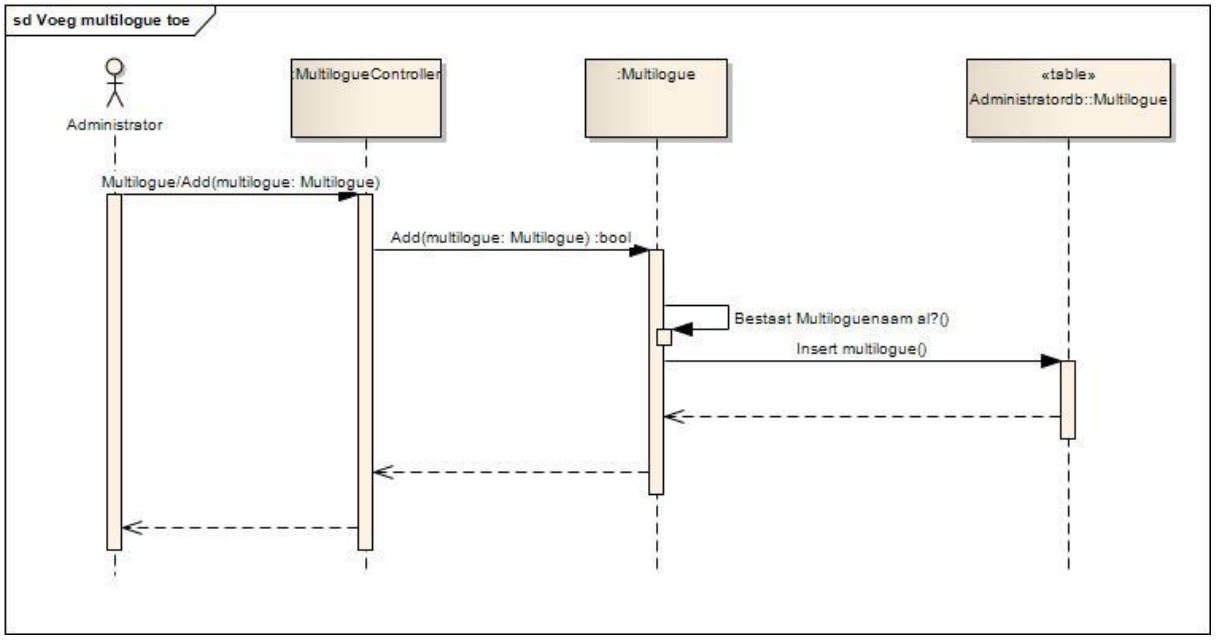
Bijlage 34 – Administrator inloggen



Bijlage 35 – Voeg klant toe



Bijlage 36 - Voeg Multilogue toe



Iteratieplan

29 maart 2010

Versie 1

Michel Dickhoff

Inhoudsopgave

Inleiding

Het Iteratieplan heeft als doel het inzicht te geven hoe het ontwikkel van de applicatie opgesplitst is in iteraties. Deze iteraties zijn beschreven zodat er verder ontwikkeld kan worden door E-mark.

Hoofdstuk 1 geeft de prioriteiten weer van de eisen. De prioriteiten zijn samengesteld door de afstudeerder, Boudewijn Somer en Nils Corver.

Aan de hand van de prioriteiten worden de iteraties beschreven in hoofdstuk 2.
Hoofdstuk 3 geeft de globale planning en de planning weer van de eerste iteratie.

1 Prioriteiten

Hoofdstuk prioriteiten legt de prioriteiten vast van de eisen die gesteld vooraf gesteld zijn aan de te realiseren applicatie. Hier wordt gebruik gemaakt van de MoSCoW methode.

1.1 Must have

De must have zijn eisen die gerealiseerd moeten worden. Zie tabel 1.

Eis	**Soort eis**
Toon profiel	Front-end
Plaats reacties	Front-end
Inloggen	Front-end
Uitloggen	Front-end
Kies taal	Front-end
Bezoeken van kamers en lezen van reacties	Front-end
Voeg gebruiker toe	Back-end editor
Voeg discussiekamer toe	Back-end editor
Voeg vraag toe aan kamer	Back-end editor
Inloggen	Back-end editor
Uitloggen	Back-end editor
Inloggen	Back-end administrator
Uitloggen	Back-end administrator
Voeg taal toe	Administrator/editor
Voeg Multilogue gebruiker toe	Administrator/editor

Tabel 1 must have eisen

1.2 Should have

De shoulds zijn de eisen die gewenst zijn (tabel 2).

Eis	**Soort eis**
Wijzigen profiel	Front-end
Toon “wie is online”	Front-end
Voeg reacties/vragen toe aan favorietenlijst	Front-end
Rapporteer reacties	Front-end
Reacties verwijderen/wijzigen	Front-end editor
Voeg formvelden toe	Back-end editor
Toon formvelden	Back-end editor
Instellen van infobericht	Back-end editor
Voeg Multilogue bericht toe	Back-end editor

Tabel 2 – should have eisen

1.3 Could have

De could have zijn eisen die aan bod komen als er genoeg tijd is (tabel 3).

Eis	**Soort eis**
Voeg stem toe aan poll	Front-end
Toon poll	Front-end
Bekijk laatst geschreven/gelezen reactie	Front-end
Categoriseer reacties	Front-end, moderator
Waardeer reacties	Front-end, moderator

Zijbalk toevoegen	Back-end editor
Bewerk zijbalk	Back-end editor
Wijzig content van pagina's	Back-end editor
Toon statistieken	Back-end editor
Voeg categorie toe	Back-end editor
Wijzig categorie	Back-end editor
Voeg poll toe	Back-end editor
Toon vertalingen	Back-end editor
Voeg woorden toe aan blacklist	Back-end editor
Zoek gebruiker	Back-end editor
Toon polls	Back-end editor
Volg gebruiker	Back-end editor
Maak groepen aan	Back-end editor
Voeg gebruiker toe aan groep	Back-end/front-end editor
Maak Multilogue aan	Back-end administrator
Toon overzicht Multilogue klanten	Back-end administrator
Toon geïnstalleerde Multilogues	Back-end administrator

Tabel 3 could have eisen

1.4 Won’t have

De won’t have zijn eisen die nu niet aan bod komen maar voor in de toekomst interessant zijn (tabel 4).

Eis	**Soort eis**
Toon administrators	Back-end administrator
Voeg administrator toe	Back-end administrator

Tabel 4 – Won’t haves

2 Iteraties

Hoofdstuk 2, “Iteraties ” legt de iteraties vast voor de te ontwikkelen applicatie.

Het ontwikkeltraject wordt opgedeeld in drie iteraties. De iteraties zijn het realiseren van de must haves, de should haves en de could haves. De iteraties zijn beschreven zodat er duidelijk is in welke stappen de applicatie tot stand moet komen.

De wanna haves worden niet beschreven doordat deze een lage prioriteit hebben.
Tijdens het implementeren wordt er getest door middel van Unit tests. Deze worden beschreven in het testdocument. De performance wordt gemeten door middel van de ANTS Profiler van Red Gate Software en ook gedocumenteerd.

2.1 Must haves

Iteratie Must haves wordt er gericht op de front-end van de Multilogue en de back-end van de editor.
In de database Multilogue worden de volgende tabellen aangemaakt:

- User
- Language
- Usertype
- File
- Filecategory
- Room
- Room_Language
- Question
- Questiontype
- Reaction
- Translation
- Session
- Pages
- Content

De controllers die voor de front-end worden geïmplementeerd met bijbehorende libs zijn:

- UserController en lib User
- QuestionController en lib Question
- RoomController en lib Room
- LanguageController en lib Language
- ReactionController en lib Reaction
- PageController

De controllers die voor de back-end worden geïmplementeerd zijn:

- HomeController
- UserController en lib User
- QuestionController en lib Question

- RoomController en lib Room

De volgende views worden in de front-end geïmplementeerd:

- User/Login, User/View
- Page/Index
- Room/List, Room/View
- Question/List, Question/Add

De volgende views worden in de back-end geïmplementeerd:

- Home/Index
- User/Add, User/AddUsers, User/Login
- Room/List, Room/Add, Room/Edit
- Question/List, Question/Add, Question/Edit
- Question/List, Question/Add

2.2 Should haves

Deze iteratie richt zich op het uitbreiden van de functionaliteit die aanwezig is in de front-end en back-end van de gebruiker.

In de database Multilogue worden de volgende tabellen aangemaakt:

- User_Question, User_Reaction
- Formfield_User, Formfield, Fieldlist, Listitem
- Content
- Page

De controllers die voor de front-end worden geïmplementeerd of bijgewerkt met bijbehorende libs zijn:

- UserController en lib User
- ReactionController en lib Reaction
- FavoriteController en lib FavoriteList
- QuestionController

De controllers die voor de back-end worden geïmplementeerd of worden bijgewerkt zijn:

- QuestionController en lib Question
- FormfieldController en lib Formfield
- FieldlistController en lib Fieldlist
- ContentController en lib Content

De volgende views worden in de front-end geïmplementeerd:

- User/Edit, User/List
- Reaction/Report

- Question/Edit

De volgende views worden in de back-end geïmplementeerd:

- Formfield/Add, Formfield/Edit, Formfield/List
- Fieldlist/Add, Fieldlist/Edit, Fieldlist/List
- Content/ViewPagecontent, Content/EditPagecontent

2.3 Could haves

Deze iteratie richt zich op het uitbreiden van de Multilogue met functies die niet de kern zijn. Ook wordt het losse administratorgedeelte ontwikkelt.

In de database Multilogue worden de volgende tabellen aangemaakt:

- Poll, Questionpoll, Poll_Page, Polloption, Pollvote
- Formfield_User, Formfield, Fieldlist, Listitem
- Rating, Category
- Reaction_Rating, Question_Rating
- Question_Category, Reaction_Category
- Blacklist
- Group, User_Group, Group_Question

In de database Administrator worden de volgende tabellen aangemaakt:

- Customer
- Multilogue
- Language
- User

De controllers in de front-end die worden bijgewerkt/toegevoegd worden:

- PollController en lib Poll
- ReactionController en lib Reaction

De controllers in de back-end van de administrator die worden bijgewerkt/toegevoegd worden:

- ContentController en lib Content
- StatisticController en lib Statistic
- BlacklistController en lib Blacklist
- UserController en lib User
- PollController en lib Poll

De controllers in de back-end van de administrator die worden bijgewerkt/toegevoegd worden:

- CustomerController en lib Customer
- MultilogueController en lib Multilogue
- LanguageController en lib Language
- UserController en lib User

De views die geïmplementeerd moeten worden in de front-end zijn:

- Poll/Add, Poll/View
- Favorite/AddQuestion, Favorite/AddReaction, Favorite/List

De views die geïmplementeerd moeten worden in de back-end van de editor zijn:

- Content/AddSidebar, Content/EditSidebar, ViewSidebars
- User/Search, User/Follow
- Blacklist/List, Blacklist/Add
- Statistic/General, Statistic/Room, Statistic/Poll, Statistic/Responses
- Category/Add, Category/List, Category/Edit
- Rating/Add, Rating /List, Rating /Edit
- Poll/Add, Poll /List, Poll /Edit

De views die geïmplementeerd moeten worden in de back-end van de administrator zijn:

- User/Login, User/Add, User/List
- Customer/Edit, Customer/Add, Customer /List
- Multilogue/Edit, Multilogue/Add, Multilogue /List
- Language/Edit, Language/Add, Language /List

3 Planning

Hoofdstuk 3, “Planning” geeft projectplanning weer en de planning voor de eerste iteratie. De afstudeerder programmeert de Must haves en de Should haves.

3.1 Globale planning

Datum	**Activiteit/ stand van zaken**
8 februari / 9 februari	*Start plan van aanpak.*
10 februari	*Feedback pva verwerkt. Plan van aanpak af versie twee af.*
10 februari – 12 februari	*Start analyse.*
12 februari	*Analysedocument eerste versie af.*
15 februari	*Feedback analysedocument. Analysedocument versie twee af.*
17 februari	*Start Functioneel ontwerp.*
19 februari	*Plan van aanpak versie drie (feedback schoolbegeleider) en analysedocument versie drie opgestuurd naar schoolbegeleider*
23 februari	*Eerste versie functioneel ontwerp af.*
24 februari	*Feedback functioneel ontwerp.*
2 maart	*Functioneel ontwerp afgerond.*
4 maart	*Bedrijfsbezoek*
5 maart	*Feedback verwerken op documenten*
2 maart tot en met 9 maart	*Uitloop functioneel ontwerp.*
10 maart	*Start technisch ontwerp.*
22 maart	*Eerste technisch ontwerp af zonder prioriteiten en iteraties*
24 maart	*Prioriteiten eisen bespreken*
25 maart	terugkomdag
26 maart	*Feedback TO. Technisch ontwerp tweede versie af.*
29 maart	Ontwikkelplan afgerond
30 maart – 23 april	Iteratie 1
26 april – 30 april	Uitloop iteratie 1
3 mei – 28 mei	Iteratie 2

31 mei – 4 juni	Uitloop iteratie 2, aanbevelingsdocument
4 juni	Afstudeerverslag inleveren

3.2 Iteratie 1

Datum	**Activiteit/ stand van zaken**
27 maart	Tabellen in SQL Server 2008 zetten, Page/Index maken (front-end)
31 maart	Inloggen front-end, inloggen back-end
1 april – 9 april	Toon profiel(front-end), Bezoeken van kamers (front-end), Plaats reacties (front-end)
12 april - 14 april	Kies taal (front-end), toevoegen van taal (administrator, back-end)
15 april – 23 april	Toevoegen van kamer/vragen (back-end) toevoegen van gebruikers (editor/administrator)
23 april	Testen en testdocument

Het plaatsen van reacties, het toevoegen van kamers en vragen is het lastigste. Daarom wordt dit gedeelte vroeg in het programmeerstadium opgenomen. 23 April worden de laatste test afgerond. Dit betekent niet dat er alleen op 23 April getest wordt. Er wordt gedurende het hele ontwikkeltraject getest. Onder getest wordt verstaan Unit tests en performance tests.

3.3 Iteratie 2

Datum	**Activiteit/ stand van zaken**
27 april	Tabellen in SQL server zetten
28 april	Voeg reacties/vragen toe aan favorietenlijst (front-end)
3 mei – 5 mei	Wijzigen profiel (front-end), toon “Wie is online”(front-end)
7 mei – 13 mei	Rapporteer reacties (front-end), Reacties verwijderen/wijzigen (front-end editor)
14 mei -19 mei	Voeg formvelden toe (back-end editor), toon formvelden
20 mei – 21 mei	Instellen van infobericht (editor back-end), voeg Multilogue bericht toe (editor back-end)
22 mei- 4 juni	Testen en optimaliseren