Herontwerp van Easydus

Voorbereiden op de toekomst

Jordi Sletterink

Eindhoven, juni 2012

Datum voltooid: 6-6-2012
Auteur: Jordi Sletterink
Studentennummer: 2176707
Docentbegeleider: Dhr. M. Krielen
SLB'er: Dhr. T. Broumels
Opleiding: ICT & Software
Afstudeerperiode: 2011/2012
Status: Final
Naam bedrijf: Balieservice
Adres: Stuiverstraat 144, 5611 TC, Eindhoven
Bedrijfsbegeleider: Hans Zuidam
Datum uitgifte afstudeerverslag: 6-6-2012
Getekend voor gezien door bedrijfsbegeleider:
Datum:

6 juni 2012 Eindhoven Hans Zuidam

I. Voorwoord

Dit verslag beschrijft mijn afstudeerstage bij het bedrijf Balieservice, waar ik onderzoek heb gedaan naar het te herontwerpen Easydus-systeem. Voordat ik met de stage begon werkte ik al een half tot een jaar bij Balieservice. Mijn voornaamste taak was het toevoegen van functionaliteiten aan Easydus en het bieden van ondersteuning. Toen ik hoorde dat ze mij een afstudeerstage wilden aanbieden en hoorde wat de stage zou gaan inhouden, was ik al meteen geïnteresseerd hierin. Ik heb uiteindelijkveel geleerd door deze stage, met name over messaging. Daarom wil ik Balieservice ook bedanken voor het beschikbaar stellen van deze stageplaats, en de bijbehorende begeleiding. Daarbij wil ik alle betrokken personen bedanken, maar met name:

- Hans Zuidam, bedrijfsbegeleider en technische ondersteuning
- Carla Otten, voor de procesbegeleiding
- Marcus Krielen, voor de begeleiding vanuit school

II. Inhoudsopgave

I. Voorwoord 2
II. Inhoudsopgave 3
III. Samenvatting 5
IV. Summary 6
1 Inleiding 7
2 Het Bedrijf 8
2.1 Balieservice 8
2.2 Easydus 8
2.3 De netwerkorganisatie Easydus 9
2.3.1 Technisch 9
2.3.2 Accountmanagement 9
3 De Opdracht 10
3.1 Context 10
3.2 Probleemomschrijving 10
3.3 Doelstelling 10
4 Het huidige systeem 11
4.1 Aanpak 11
4.2 Gebruikte technieken 11
4.3 Klassen 11
4.4 Database 11
4.5 GUI 12
4.6 Proces-/eventflow 14
5 Event-handling en interne communicatie 15
5.1 Messaging 15
5.1.1 Patterns 15
5.1.2 Request/Reply 16
5.1.3 Message broker 16
5.2 Events 16
5.2.1 Definitie 16
5.2.2 Granulariteit 16
5.3 Bestaande messaging systemen 17
5.4 Voorbeeld 17
5.4.1 Systeemomschrijving 17

5.4.2 Modules 17
5.4.3 Messages en events 19
6 De Modules 21
6.1 Opdeling 21
6.2 Het Messaging systeem/De Broker 22
6.3 Talen van de modules 23
6.4 Opdeling klanten en projecten 23
7 Conclusies en aanbevelingen 24
V. Verklarende woordenlijst 25
VI. Evaluatie 26
VII. Literatuurlijst 27
VIII. Bijlagen 28
A. PID 28
B. Communicatieplan 29
C. Onderzoek huidig systeem 30

III. Samenvatting

Easydus is gestart in 2009, aanvankelijk onder de handelsnaam Inschrijfbalie, maar per 1 januari 2012 uit oogpunt van internationale expansie gewijzigd naar Easydus. Easydus is een softwaretool voor het projectmatig geïntegreerd inventariseren, beheren en communiceren van data. Gedurende de totale ontwikkeling is steeds het uitgangspunt geweest een breed scala aan functionaliteiten te ontwikkelen met een maximum gemak voor de gebruikers, zonder noodzakelijke hulp of inzet van IT specialisten.

Omdat de klantenkring, en dus het gebruik van Easydus, de laatste tijd hard is gegroeid, begin het systeem tegen zijn grenzen aan te lopen. Om dit te voorkomen is de stage opdracht bedacht om een advies uit te brengen over hoe het systeem het beste opgesplitst kan worden in verschillende modules, om zo het systeem beter schaalbaar te maken en voor te bereiden op de toekomst.

Als eerste is er uitgezocht hoe het systeem precies in elkaar zit, en wat op dit moment de volgorde van gebeurtenissen is binnen het systeem. Ook is de samenhang bekeken tussen de verschillende functionaliteiten en delen van het systeem. Daarna is er onderzocht hoe dit omgezet kan worden naar een modulair systeem. Het blijkt dat dit het beste kan gebeuren via een Messaging systeem, welke alle gebeurtenissen rondstuurt naar alle modules (bijvoorbeeld een inschrijving). Hierbij is er goed in kaart gebracht welke vormen van messaging en event handling bestaan, om zo de beste te kunnen kiezen voor het nieuwe Easydus systeem.

Ook is er, aan de hand van de functionaliteiten van het huidige systeem, een advies gegeven over de opsplitsing van het systeem. In dit advies wordt verteld in welke modules het systeem het beste opgedeeld kan worden, en welke functionaliteiten iedere module heeft. Als laatste is er ook een advies gegeven over de programmeertalen waarin de modules het beste geschreven kunnen worden. Dit bleek Java te zijn.

Aangezien de message broker het hart van het systeem is (hier gaan tenslotte alle berichten doorheen), is er naar verschillende brokers en messaging systeem gekeken. Van alle brokers/messaging systemen bleek RabbitMQ het beste overeen te komen met de wensen van het nieuwe systeem, en het beste mee te kunnen gaan met de toekomst. Ook bleek RabbitMQ erg goed schaalbaar te zijn, en ondersteuning te bieden voor vrijwel iedere programmeertaal.

IV. Summary

Easydus started in 2009, initially under the trade name Inschrijfbalie, but as of the 1st of January 2012, changed to Easydus because of international expansion. Easydus is a software tool for the project-based collecting, managing and communicating of data. During the entire development period, the main concept was to develop a broad range of functionalities with maximum convenience for the users, without the need of assistance from a IT specialist.

Because the clientele, and therefore the use of Easydus, has grown exponentially in the past few years, the system is started to hit its limits. To prevent this, the assignment came up to do research and come up with an advise on how to divide the system in separate modules, thus making the system more scalable and to prepare for the future.

Firstly, the current system has to be examined to find out how it works, and what the sequence of events is. The coherence between the different functionalities and parts of the system also has to be examined. After this research, it has to be examined how the system can be converted to a modular system. It appears that the best way to do this is with a messaging system, that distributes all events to the other modules. During this research, all sorts of messaging and event patters are examined, so the best (combination) can be chosen for the new Easydus system. There has also been given an advice about the split-up of the system, based on the current functionalities of Easydus. This advice describes in what modules the system can be split-up in the best, and what functionalities every module has. Finally there is also an advice about the programming language(s) the new system can be best written in. This was found to be Java.

Since the message broker is the heart of the system (all messages pass through it), there were various brokers and messaging system examined. Of all brokers/messaging systems, RabbitMQ showed the best match with the needs of the new system, and the best way to deal with future expansion. RabbitMQ also turned out to be very scalable, and there are libraries available for virtually every programming language.

1 Inleiding

Het bedrijf Balieservice, gevestigd te Eindhoven, heeft sinds 2009 het product Easydus in ontwikkeling. Easydus is een systeem dat klanten instaat stelt op eenvoudige wijze inschrijfformulieren voor een tal van activiteiten aan te maken. Inschrijvers kunnen vervolgens via de backend worden beheerd(denk hierbij aan factureren, emailcommunicatie, enz.). Omdat de klantenkring die gebruik maakt van Easydus steeds groter werd, begon het systeem uit zijn voegen te barsten. Om dit op te lossen zullen de functionaliteiten van Easydus opgesplitst moeten worden in verschillende modules, zodat deze bij een grote werklast geen hinder van elkaar zullen ondervinden.

De stage opdracht bestaat uit het opzetten van een aanbeveling over hoe Easydus het beste opgesplitst kan worden, en met welke technieken. Hoe tot deze aanbeveling is gekomen komt naar voren in deze scriptie. In hoofdstuk 2 wordt uitgebreide informatie gegeven over het bedrijf, in hoofdstuk 3 staat alle informatie over de opdracht, hoofdstuk 4 legt uit hoe het huidige systeem in elkaar zit, hoofdstuk 5 bevat informatie over event-handling en de interne communicatie, hoofdstuk 6 bevat een advies over de verschillende functionaliteiten en modules en ten slotte staat in hoofdstuk 7 de conclusie/aanbeveling.

2 Het Bedrijf

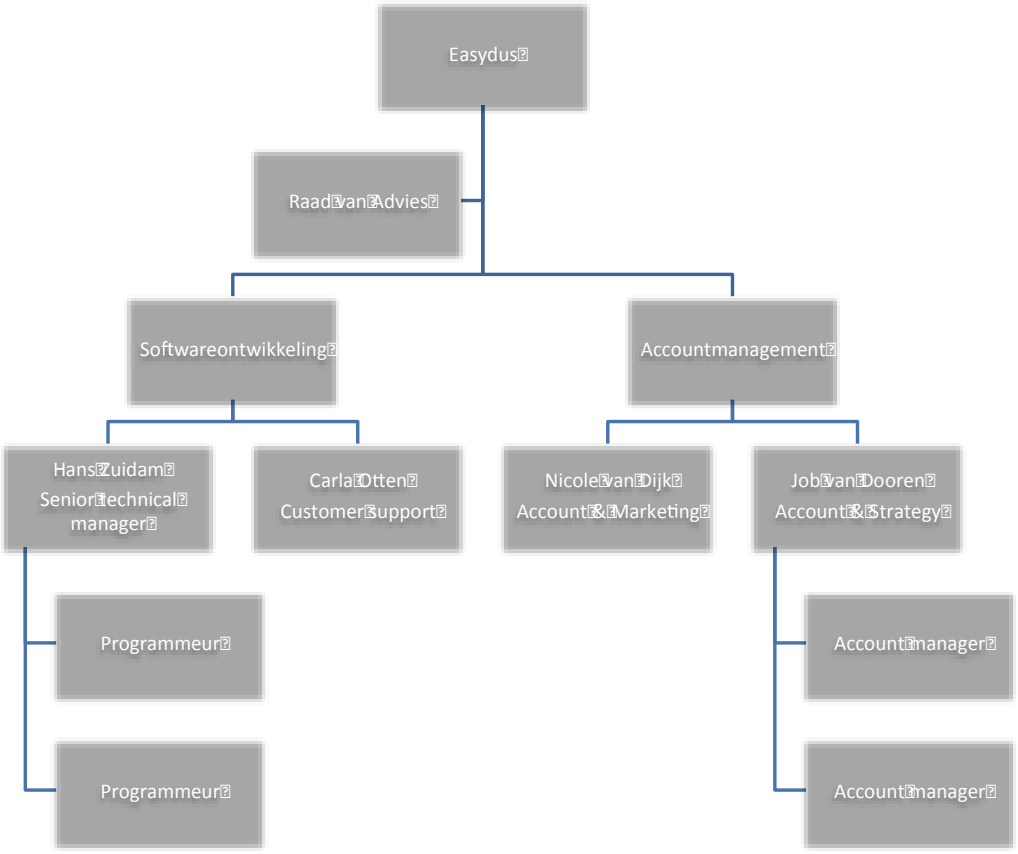
2.1 Balieservice

Balieservice is opgericht in 2007 door Carla Otten en Nicole van Dijk. Binnen Balieservice zijn diverse activiteiten ontwikkeld in de loop der jaren, voornamelijk gelegen in het adviseren bij en realiseren van websites en andere communicatietoepassingen voor organisaties. Meer en meer is de focus gelegd op de ontwikkeling en verkoop van de web-based applicatie Easydus, dat nu wordt aangestuurd door de netwerkorganisatie Easydus.

2.2 Easydus

Easydus is gestart in 2009, aanvankelijk onder de handelsnaam *Inschrijfbalie*, maar per 1 januari 2012 uit oogpunt van internationale expansie gewijzigd naar Easydus.
Easydus is een software tool voor het projectmatig geïntegreerd inventariseren, beheren en communiceren van data. Gedurende de totale ontwikkeling is steeds het uitgangspunt geweest een breed scala aan functionaliteiten te ontwikkelen met een maximum gemak voor de gebruikers, zonder noodzakelijke hulp of inzet van IT specialisten.

De verschillende doelgroepen voor het pakket zijn bedrijven, professionele dienstverleners, beroeps- en brancheorganisaties, scholen, verenigingen en goede doelenorganisaties. De klantenlijst van Easydus bedient inmiddels uiteenlopende organisaties als het Landelijk Bureau D66, Vereniging van Arbeidsrecht Advocaten Nederland, Nederlandse Vereniging Letselschadeadvocaten, Huygenscollege, Eindhovensche Golf, Holla Advocaten, Novius en Piet Hein Eek.



2.3 De netwerkorganisatie Easydus

Vanaf de eerste start is Easydus te beschouwen geweest als een netwerkorganisatie, samenwerkend met andere organisaties en personen al naar gelang de aard van de uit te voeren projecten.

Partners zijn gevonden voor zowel het strategisch verder uitzetten van dit product alsook voor het technisch verder ontwikkelen en het concreet verkopen daarvan.

2.3.1 Technisch

Aan de technische kant zijn Hans Zuidam en Carla Otten eindverantwoordelijk.

2.3.1.1 Hans Zuidam

Hans Zuidam studeerde in 1983 af op richting Technical Computer Science aan de HTS in Breda, en was daarna onder andere als senior software engineer werkzaam voor TU/e, Philips, GLI, Brand Innovators en Origin. Hij is thans eigenaar van Hans Zuidam Advies, en vervult in die hoedanigheid opdrachten voor bedrijven als NXP, TU/e en Phillips. Hans Zuidam is de stagebegeleider aan de technisch inhoudelijke kant.

2.3.1.2 Carla Otten

Carla Otten werd na haar opleiding tot directiesecretaresse Basic programmeur, richtte WITAN presentaties op, een bedrijf in (bedrijfs)presentaties, website ontwikkeling en hosting, en richtte in 2007 Balieservice op. Carla is stagebegeleider aan de proceskant.

2.3.1.3 Netwerk

In het netwerkmodel participeren in de raad van advies aan de technische kant Prof. Ralph Otten, werkzaam bij de faculteit elektronische systemen TU/e, met onder andere als specialiteit software, algoritmen en besturingssystemen. Daarnaast Jan Smeelen, eigenaar van Webfontein, werkzaam als Senior PHP programmeur.

2.3.2 Accountmanagement

Aan de accountkant zijn Nicole van Dijk en Job van Dooren de eindverantwoordelijken.

2.3.2.1 Nicole van Dijk

Nicole van Dijk volgde haar opleiding bij Schoevers, behaalde in 1980 haar HBO diploma, en in 1998 haar post HBO opleiding Bedrijfskundig Management aan het Instituut Career and Development. Zij was eigenaar van Acta, consultant bij Triceps en startte in 2007 Balieservice.

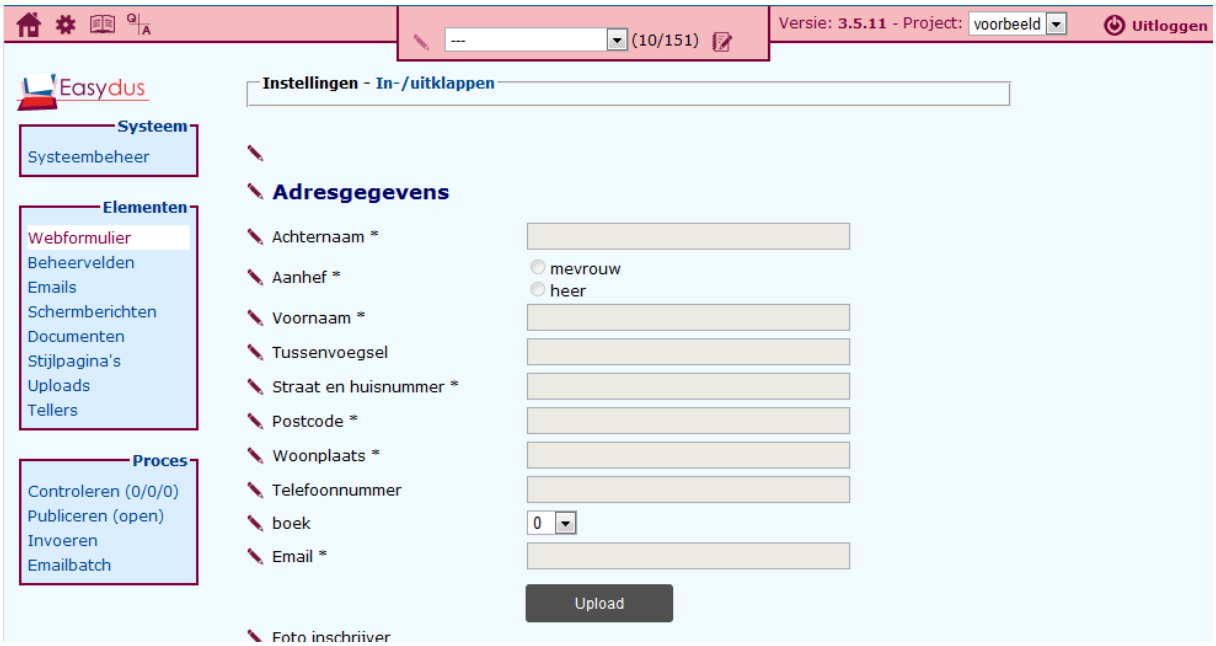
2.3.2.2 Job van Dooren

Job van Dooren behaalde zijn bul bedrijfseconomie aan de RUG in 1982, vervulde leidinggevende posities als marketeer, accountmanager en directeur binnen Unilever, Friesland en Continental Bakeries. Hij richtte in 2002 BrandWatch mee op, en investeert in verschillende bedrijven.

3 De Opdracht

3.1 Context

Het Easydus systeem is een Web (LAMP) gebaseerd systeem dat klanten in staat stelt op eenvoudige wijze inschrijfformulieren voor een tal van activiteiten aan te maken. Inschrijvers kunnen vervolgens worden beheerd in termen van inplanning voor b.v. workshops, facturering, etc. Zogenaamde back-end functies laat klanten toe inschrijvers te beheren, facturatie te beheren, communicatie met inschrijvers te verzorgen, etc.



3.2 Probleemomschrijving

Het huidige systeem voldoet, maar is niet modulair genoeg. Een groeiende klantenkring zorgt ook voor een groeiend wensenpakket. Het blijkt in toenemende mate dat implementatie van deze wensen meer moeite kost dan nodig zou moeten zijn.

3.3 Doelstelling

Het doel van de opdracht is een nieuwe systeemarchitectuur op te zetten en aan te tonen dat deze efficiënt te implementeren en te gebruiken is. Hiertoe wordt voorzien dat diverse functionele onderdelen van het systeem worden afgesplitst in zelfstandige processen. Deze onderdelen (modules) communiceren dan onderling volgens een of meerdere nader te bepalen protocollen. De communicatie zal doormiddel van het uitwisselen van gebeurtenissen (events) zijn.

Tevens moet de wijze waarop deze events in het systeem worden afgehandeld opnieuw worden ontworpen. Bij events moet worden gedacht aan zaken als het afhandelen van een inschrijving, het genereren van een factuur, het verzenden van e-mail, etc. Door deze events meer generiek te maken en een meer centrale rol te laten spelen, wordt het eenvoudiger het gedrag van het systeem aan specifieke klanten/wensen aan te passen.

4 Het huidige systeem

4.1 Aanpak

Om het nieuwe systeem te kunnen ontwerpen en te onderzoeken hoe de event-handling gaat gebeuren, moet er natuurlijk eerst uitgezocht worden hoe het bestaande systeem precies werkt. Om dit te doen heb ik gesproken met de programmeur die het systeem heeft ontworpen en gebouwd (Martijn van Eijndhoven) en ik heb de bestaande code doorgespit.

Als eerste is er gekeken welke klassen er zijn en hoe ze werken, daarna is de database geanalyseerd, daarna is gekeken hoe de GUI is opgedeeld, en als laatste is er uitgezocht wat de 'stroom' van events en acties is.

Hieronder wordt kort beschreven hoe het systeem is opgebouwd, de uitgebreide omschrijving kan gevonden worden in het onderzoeksdocument bij de bijlagen.

4.2 Gebruikte technieken

Het huidige systeem is in zijn geheel opgebouwd in PHP, zoveel mogelijk volgens het MVC-ontwerp. De databases staan op een MySQL server en de site zelf draait op een Apache-server. Apache en MySQL draaien op dit moment samen op één server, die draait op Linux (Debian 6.0). Er wordt dagelijks en per uur een back-up gemaakt, zodat de kans op lange uitval na een storing nihil is. Binnen de site zelf wordt ook nog Javascript en een beetje Ajax gebruikt, bijvoorbeeld bij de overzichten.

4.3 Klassen

Alle (systeem-)klassen staan in de map `Classes`, met uitzondering van zogenaamde third-party pakketten die gebruikt worden zoals de PHPMailer.

In deze map bevinden zich de mappen `formElements` en `external`. De map `formElements` bevat alle elementen die in het webformulier voor kunnen komen(bijvoorbeeld de `TextQuestion`, of de `MultipleChoiceQuestion`), en in de map `external` staan alle (in-house ontwikkelde) klassen die geen directe samenhang met het systeem hebben(zoals de UUID-generator of randomizer).

Alle klassen die in de map `Classes` staan zijn afgeleid van de klasse `DBObject2`. Dit is een abstracte klasse die alle basislogica bevat voor het opslaan en ophalen van data uit de database. Het enige dat in klassen die gebruik maken van `DBObject2` gedaan hoeft te worden, is het instellen welke tabel en kolommen gebruikt worden en of een van de velden een verwijzing is naar een andere klasse (zo heeft de klasse `EmailTemplate` bijvoorbeeld een veld waarin een object van de klasse `StyleSheet` wordt opgeslagen).

4.4 Database

Er zijn in totaal drie verschillende 'typen' databases:

De systeemdatabase

Dit is de database waarin alle klanten staan, met bijbehorende gegevens zoals de versie waar de klant op staat, secureID (een unieke willekeurige reeks letters en cijfers), enz. Deze database is systeembreed, er bestaat er maar één van.

De klantdatabase

In deze database worden alle inhoudelijke gegevens van een klant opgeslagen. Bijvoorbeeld alle gebruikers die in mogen loggen in deze klant met bijbehorende rechten, en welke projecten deze klant heeft. Iedere klant heeft één database van dit type.

De projectdatabase

In deze database worden alle project gerelateerde gegevens opgeslagen, zoals bijvoorbeeld de schermberichten, emails, inschrijvingen, formulieren, enz. Ieder project heeft één database van dit type en vrijwel iedere klasse heeft een eigen tabel in deze database.

4.5 GUI

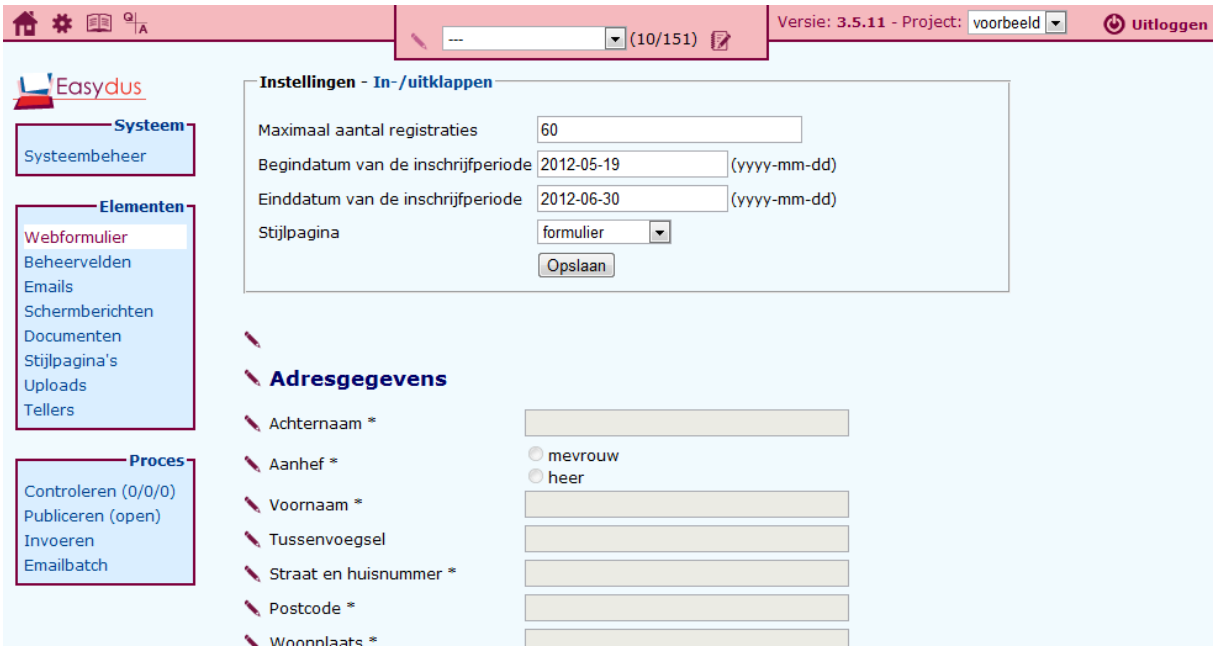
De GUI is opgedeeld in 3 verschillende delen.

Superadmin-GUI

Dit is de GUI die gebruikt wordt voor het aanmaken en beheren van klanten. Vanuit deze GUI kunnen klanten ook ge-up- of -downgrade worden naar andere versies van het systeem.

Admin-GUI

Dit is de GUI waarin een klant alle projecten en inschrijvingen beheert. In deze GUI worden alle formulieren, emailtemplates, schermberichten, factuurtemplates, enz. aangemaakt.



Enduser-GUI

Deze GUI bevat alles wat de eindgebruiker (inschrijver) te zien krijgt. Denk hierbij aan het inschrijfformulier, de schermberichten en de betalingspagina's.

4.6 Proces-/eventflow

Het systeem is enigszins event gebaseerd opgezet. Er zijn verschillende handelingen die de uiteindelijke inschrijver kan doen (inschrijven, betalen, enz.) en aan de hand daarvan gebeuren er bepaalde dingen in het systeem (emails versturen, facturen genereren, enz.).

Het systeem bevat op dit moment de volgende ‘flows’:

- Een CSV-bestand importeren
- Het maken van een inschrijving
- Een betaling doen

In deze ‘flows’ worden de volgende ‘subflows’ gebruikt:

- Het weergeven van een schermbericht
- Het sturen van een email

Al deze ‘flows’ worden uitgebreid uitgelegd in het onderzoeksdocument bij de bijlages.

5 Event-handling en interne communicatie

In dit hoofdstuk wordt beschreven wat er allemaal onderzocht is met betrekking tot de event-handling en interne communicatie voor het nieuwe systeem.

5.1 Messaging

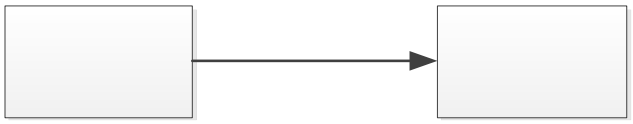
Om te zorgen dat alle delen van het nieuwe systeem op de hoogte zijn van alle events, wordt er gebruikt gemaakt van messaging. Er zijn verschillende methodes hoe messaging toegepast kan worden, deze zullen in dit hoofdstuk besproken worden.

5.1.1 Patterns

De meest voorkomende message patterns zijn Unicast, Multicast en Broadcast.

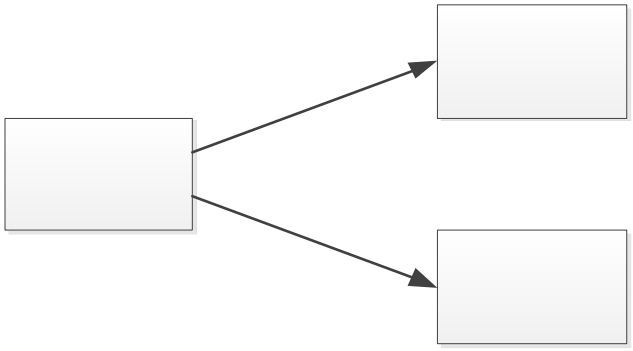
Unicast

Unicast is het sturen van een bericht aan een specifieke ontvanger. Ook wel point-to-point communicatie genoemd.



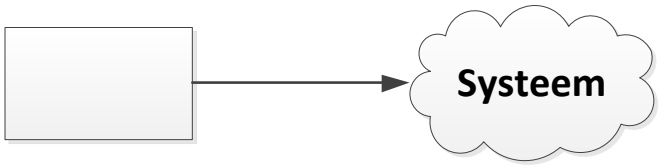
Multicast

Multicast is vrijwel hetzelfde als unicast, alleen wordt een bericht aan meerdere specifieke ontvangers gestuurd.



Broadcast

Bij broadcasting wordt een bericht aan iedereen in het systeem gestuurd, in plaats van specifieke ontvanger(s). Iedere ontvanger bepaalt hierbij zelf of er wat moet gebeuren bij ieder bericht.



5.1.2 Request/Reply

Om een data-request via messaging te kunnen doen, zal er een bericht heen en weer moeten gaan. Eerst de request met de vraag naar de data en dan de reply terug met de data zelf.

Hier zijn verschillende methoden voor bedacht:

Synchroon

Bij synchrone messaging verstuurt de verzender de request, en blokkeert dan totdat het antwoord is ontvangen. Dit kan zeer nadelig zijn als de snelheid bij de verzendende module van belang is, en de ontvangende module veel tijd nodig heeft om zijn ding te doen.

RPC (Remote Procedure Call) is een voorbeeld van synchrone messaging. Bij een RPC aanroep wordt een methode van een externe module aangeroepen, en gewacht tot die module klaar is. Het voordeel hiervan is, is dat je na een RPC aanroep zeker weet dat je een antwoord hebt ontvangen, aangezien de module niks doet tot het antwoord aangekomen is.

Asynchroon

Bij asynchrone messaging verstuurt de verzender een request, en gaat dan verder met waar het mee bezig was. Het antwoord hoeft hierbij niet snel teruggestuurd te worden, en de volgorde waarin de antwoorden terugkomen maakt ook niet uit.

Bij een systeem waarbij messaging als event-distributie wordt gebruikt, is dit de handigste methode om te gebruiken, samen met het multicast/broadcast pattern. Een event zal dan als bericht verstuurd worden aan alle modules die luisteren binnen het systeem, en iedere module bepaald dan voor zich of hij er wat mee wilt doen of niet.

5.1.3 Message broker

De message broker is de module die verantwoordelijk is voor het rondsturen van alle messages in het systeem. De broker zorgt ervoor dat alle berichten bij de geadresseerde modules aankomen, en broadcast-messages bij iedereen aankomen. Omdat alle berichten via de broker gaan, is het belangrijk dat deze snel genoeg is om de grote hoeveelheid berichten binnen het systeem te kunnen verwerken. Het grote voordeel bij het gebruik van een broker is dat niet ieder component zelf de routerings beslissingen hoeft te nemen, maar dat dit centraal geregeld wordt.

5.2 Events

5.2.1 Definitie

Een event is kort gezegd een gebeurtenis (in het systeem). Deze gebeurtenis kan veroorzaakt zijn door een gebruiker (extern: bijvoorbeeld een inschrijving), of door het systeem zelf (intern: bijvoorbeeld als een email verzonden is). Het is belangrijk om bij het ontwerpen van het nieuwe Easydus-systeem goed te definiëren welke events er allemaal zijn, omdat daar het hele systeem op gebaseerd is.

5.2.2 Granulariteit

Events bestaan op verschillende ‘diepten’ binnen het systeem. Zo zou op een zeer laag niveau iedere muisbeweging en iedere toetsaanslag een event kunnen zijn, en op een hoog niveau een voltooide inschrijving of de aanmaak van een project. Het is belangrijk om bij het ontwerp van het nieuwe

systeem goed te bepalen tot in welke granulariteit de events gaan, zodat het systeem niet overspoeld wordt met (onnodige) berichten, maar wel zo diep dat de berichten nut blijven hebben.

5.3 Bestaande messaging systemen

Er zijn vele verschillende soorten messaging systemen, waarvan er een paar voor één specifieke taal zijn gemaakt, en een paar die zo open mogelijk zijn (sommige zelfs open source). Hieronder staan er een paar.

MSMQ (Microsoft Message Queueing)
Zoals de naam al aangeeft is dit het message-protocol van Microsoft. Deze taal wordt standaard ondersteund in alle .NET-talen. Echter is het wel noodzakelijk een Windows Server te draaien, gezien dat de enige broker is voor dit protocol.

RabbitMQ (een AMQP – Advanced Message Queueing Protocol - implementatie)
RabbitMQ is een open-source implementatie van het tevens open AMQP messaging protocol, geschreven in de taal Erlang. RabbitMQ kan op verschillende platformen kan draaien(Windows, Linux, Mac OS X, en de cloud (Amazon EC2)). Tevens zijn de libraries om met een RabbitMQ-broker te kunnen communiceren in vele verschillende talen beschikbaar.

JMS (Java Message Service)
Dit is een message-broker/-service gemaakt in en voor Java. Er zijn wel verschillende implementaties van deze broker die andere talen ondersteund, maar die worden niet 'native' ondersteund. Verder lijkt het protocol van dit messaging-systeem heel erg op AMQP. Er zijn dan ook modules om berichten over te zetten voor een AMQP-systeem.

5.4 Voorbeeld

In dit hoofdstuk wordt een voorbeeld gegeven van de 'flow' in het nieuwe systeem bij het verzenden van een email na een inschrijving.

5.4.1 Systeemomschrijving

Zodra een inschrijving voltooid is, wordt er een email verzonden naar de inschrijver.

Aan deze email zit een bijlage; dit is een document gemaakt in het systeem. Deze zal worden geconverteerd naar een PDF-bestand.

Op zowel de email als de bijlage wordt substitutie gedaan. Hiermee wordt bedoeld dat er variabelen in de email en het document voorkomen, die worden ingevuld met de gegevens van de inschrijver(bijvoorbeeld de naam of het adres).

5.4.2 Modules

Om deze taken uit te voeren zijn er verschillende modules nodig waarnaar de events gestuurd worden. Deze modules zijn:

Enduser-GUI
Dit is de interface naar de eindgebruiker toe. In dit geval het inschrijfformulier dat de inschrijver invult. Deze module zorgt tevens voor de initiële event.

Workflow manager
Deze module bepaald wat er allemaal gebeurt na een event. Als er is ingesteld in het project dat er na een inschrijving een email verstuurd moet worden, zorgt deze module ervoor dat die email verstuurd gaat worden.

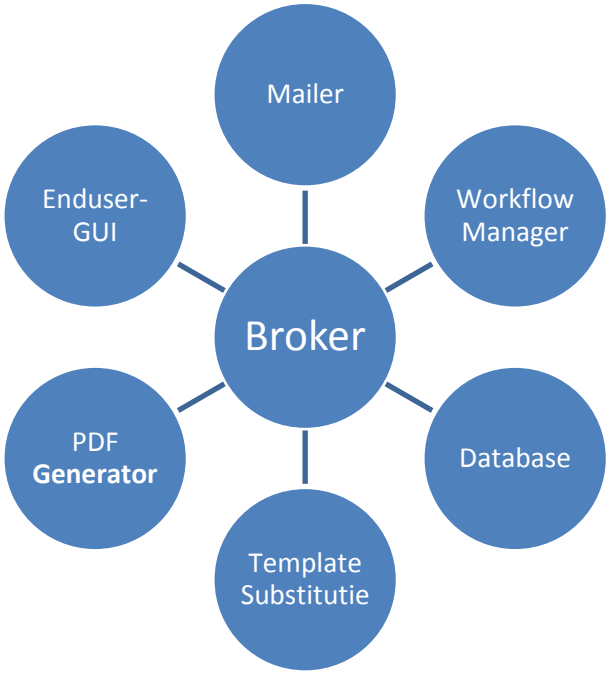
Mailer
Dit is de module die verantwoordelijk is voor het verzenden van de email (inclusief de bijlage).

Database
In deze module worden vrijwel alle projectgegevens opgeslagen. Denk hierbij aan de Inschrijvingen, EmailTemplates en schermberichten.

Template substitutie
Deze module haalt alle templates en inschrijvergegevens op, en doet de template substitutie (alle variabelen worden ingevuld met de inschrijvergegevens).

Message Broker
De message broker is een van de belangrijkste modules in het systeem. Deze zorgt ervoor dat alle berichten op de goede plaats aankomen.

PDF Generator
De PDF generator zet tekst of HTML om naar een PDF-bestand, om bijvoorbeeld met een email mee te zenden.



5.4.3 Messages en events

Het rondgaan van de events en acties kan op verschillende manieren gebeuren. Hier worden er twee beschreven.

Methode 1

Deze eerste methode gebruikt vooral broadcast messages, en iedere module doet aan de hand van deze messages een bepaald aantal dingen, of helemaal niks.

Functionele omschrijving:

1. Zodra de inschrijver de gegevens opslaat, verzend de **Enduser-GUI** een event `inschrijving` met alle inschrijvergegevens naar het hele systeem (**broadcast**). De **Database** slaat de gegevens op.
2. De **Workflow manager** zend (aan de hand van de instellingen van het project) een **broadcast** message, met de instructie om een email te verzenden, met de ID's van de `EmailTemplate`, `DocumentTemplate` en `Inschrijver` van de email. De **Mailer** bereid nu het verzenden van de email voor.
3. De **Database**-module zend nu de `EmailTemplate`, `DocumentTemplate` en `Inschrijving` naar de **Template substitutie** (**broadcast**).
4. **Template Substitutie** zend nu de gesubstitueerde templates naar de **Mailer** (**broadcast**).
5. De **Mailer** zend het Document (bijlage) naar de **PDF generator** middels **request/reply**, en krijgt de PDF terug. Nu wordt de email verzonden.

Als deze methode gebruikt wordt, gaan de volgende zes berichten rond:

- Van **Enduser-GUI** naar iedereen (**broadcast**), met de inschrijvergegevens
- Van de **Workflow Manager** naar iedereen (**broadcast**), met de gegevens van de email.
- Van de **Database** naar iedereen (**broadcast**), met de inschrijver, EmailTemplate en DocumentTemplate.
- Van de **TemplateSubstitutie** naar de **Mailer** (**unicast**), met de email en bijlage zelf.
- Van de **Mailer** naar de **PDF generator** (**request/reply**), met de bijlage(tekst).
- Van de **PDF generator** naar de **Mailer** (**request/reply**), met het PDF-bestand.

Methode 2

In plaats van broadcast-messages, kan er ook meer met unicast en request/reply worden gewerkt.

Functionele omschrijving:

1. Zodra de inschrijver de gegevens opslaat, verzend de **Enduser-GUI** een event `inschrijving` met alle inschrijvergegevens naar het hele systeem (**broadcast**). De **Database** slaat de gegevens op.
2. De **Workflow Manager** zend de ID's van de `EmailTemplate`, `DocumentTemplate` en Inschrijver naar de **Mailer** (**unicast**).
3. De **Mailer** zend twee berichten (**request/reply**) naar de **Template Substitutie**. Een met het ID van de inschrijver en de `EmailTemplate`, en een met het ID van de inschrijver en het `DocumentTemplate`.

4. De **Template Substitutie**-module zend twee berichten naar de **Database** (**request/reply**), eentje voor de data van de `EmailTemplate` en de inschrijver, en eentje voor de `DocumentTemplate` en de inschrijver. Dit kunnen er ook 4 zijn, als de inschrijver en templates los worden gevraagd aan de database.
5. Vervolgens worden de templates gesubstitueerd, en teruggezonden naar de **Mailer.**
6. De **Mailer** zend het Document (bijlage) naar de **PDF generator** middels **request/reply**, en krijgt de PDF terug. Nu wordt de email verzonden.

Bij gebruik van deze methode gaan de volgende dertien berichten rond:

- Van **Enduser-GUI** naar iedereen (**broadcast**), met de inschrijvergegevens,
- Van de **Workflow Manager** naar de **Mailer**, met de gegevens van de email.
- Van de **Mailer** naar de **Template Substitutie**, met de email en inschrijver.
- Van de **Mailer** naar de **Template Substitutie**, met het document en de inschrijver.
- Van de **Template Substitutie** naar de **Database**, met de vraag voor de gegevens van de email en de inschrijver.
- Van de **Template Substitutie** naar de **Database**, met de vraag voor de gegevens van het document en de inschrijver.
- Het antwoord van de **Database** aan de **Template Substitutie** met de email en inschrijver.
- Het antwoord van de **Database** aan de **Template Substitutie** met het document en de inschrijver.
- Het antwoord van de **Template Substitutie** aan de **Mailer** met de email.
- Het antwoord van de **Template Substitutie** aan de **Miler** met het document.
- Van de **Mailer** naar de **PDF generator**, met de bijlage (tekst).
- Van de **PDF generator** naar de **Mailer**, met het PDF-bestand.

Als gekeken wordt naar het aantal berichten, is methode 1 veel efficiënter dan methode 2. Het grote voordeel van een broadcast-georiënteerd messaging-systeem is dat de modules die de broadcasts ontvangen, al kunnen beginnen met het ophalen of klaarzetten van gegevens voordat die daadwerkelijk benodigd is. Dit is in bovenstaande methode 1 goed te zien; de mailer bereidt het verzenden van de email voor, voordat de gegevens zijn aangekomen. Ook zend de database de EmailTemplate en Inschrijvergegevens al naar de Template Substitutie zonder dat eerst gevraagd is naar de data.

6 De Modules

In dit hoofdstuk wordt bekeken hoe het systeem het beste opgesplitst kan worden.

6.1 Opdeling

Gekeken naar de functionaliteiten in het huidige Easydus-systeem, kan het systeem het beste opgedeeld worden in de volgende modules:

Message Broker

Dit is het belangrijkste onderdeel van het systeem, hier gaan alle berichten doorheen. Het is dus van belang dat de broker snel genoeg is om vele klanten aan te kunnen, rekening houdend met de groei van Easydus.

Workflow manager

Zoals de naam al zegt is de workflow manager verantwoordelijk voor de flow van een project. Hiermee wordt bedoeld dat als bijvoorbeeld een inschrijving-event wordt verstuurd, deze module ervoor zorgt dat de bijbehorende emails worden verzonden, of facturen gegenereerd.

Superadmin GUI

Via deze GUI kunnen systeembeheerders klantbeheer uitvoeren zoals het aanmaken van nieuwe klanten in het systeem, of het overzetten van klanten op andere versies van het systeem. Ook kan via deze module naar het verbruik van klanten (aantal inschrijvingen, verzonden emails, enz.) gekeken worden zodat deze gefactureerd kunnen worden.

Admin GUI

Deze module bevat de GUI die klanten op dit moment voornamelijk gebruiken. Hierin worden alle emails, documenten, facturen, formulieren, enz. aangemaakt. Ook wordt via deze GUI de Workflow Manager ingesteld (welke email wanneer weg moet gaan, enz.).

Enduser GUI

Dit is de GUI waar de inschrijver op terecht komt. Deze laat het formulier zien en verzend het bericht met de inschrijvergegevens. Ook laat deze module de schermberichten en betaalpagina's zien.

Database

In deze module worden alle gegevens met betrekking tot de inschrijvingen opgeslagen. Dit zijn bijvoorbeeld de inschrijvingen zelf, maar ook de verschillende templates, logs, verzonden emails, enz. Het is aan te raden de opslag van de templates in deze module te doen in plaats van de module waar ze bij horen (bijv. de EmailTemplates in de Mailer-module), omdat deze module in de toekomst op een aparte server kan gaan draaien die volledig is geoptimaliseerd voor data-opslag.

Deze module kan ook als tussenlaag dienen als een deel van de inschrijvergegevens uit de database van een klant komen, het hoeft dus niet de database zelf te zijn.

PDF Generator

Deze module zet tekst (meestal HTML) om naar een PDF bestand. Deze zou in de toekomst ook om kunnen zetten naar andere bestandsformaten.

Mailer
Deze module is verantwoordelijk voor het verzenden van alle emails. Het is aan te raden om de mailfunctionaliteit altijd in een aparte module te houden, zodat de rest van het systeem er geen hinder van ondervind als een klant bijvoorbeeld een miljoen emails stuurt. Net als de Databasemodule is dit tevens een tussenlaag voor externe mailers, zodat een klant ervoor zou kunnen kiezen een gespecialiseerde mailer zoals bijvoorbeeld MailChimp te gebruiken.

Logger/Eventgenerator (debug)
Deze module is er om alle gebeurtenissen binnen het systeem te volgen. Dit is essentieel bij het oplossen van bugs, en een handige tool bij het verhelpen van problemen bij klanten. Met deze module kunnen tevens handmatig events worden verstuurd, om te testen of alles naar behoren werkt.

Eventuele andere modules kunnen later altijd aan het systeem gekoppeld worden. Hierbij kan gedacht worden aan een REST-api voor externe applicaties, of een Admin-GUI voor mobiele apparaten.

6.2 Het Messaging systeem/De Broker
Omdat de broker het belangrijkste onderdeel is van het systeem, is het belangrijk dat er een betrouwbare en schaalbare broker wordt gekozen. Hier volgen er een paar met hun grootste voor- en nadelen.

MSMQ (Microsoft Message Queueing)
Zoals de naam al zegt is dit het message-protocol van Microsoft. Deze taal wordt standaard ondersteund in alle .NET-talen. Echter is het wel noodzakelijk een Windows Server te draaien, gezien dat de enige broker is voor dit protocol. Documentatie is goed beschikbaar.

JMS (Java Message Service)
Dit een message-broker/-service gemaakt in en voor Java. Er zijn wel verschillende implementaties van deze broker die andere talen, maar die worden niet ‘native’ ondersteund. Verder lijkt het protocol van dit messaging-systeem heel erg op AMQP. Er zijn dan ook modules om berichten over te zetten voor een AMQP-systeem. Een groot voordeel is dat er veel over te vinden is op internet, en er goede documentatie beschikbaar is.

RabbitMQ (een AMQP – Advanced Message Queueing Protocol - implementatie)
RabbitMQ is een open-source implementatie van het tevens open AMQP messaging protocol, geschreven in de taal Erlang. RabbitMQ kan op verschillende platformen draaien(Windows, Linux, Mac OS X, en de cloud (Amazon EC2)). Tevens zijn de libraries om met een RabbitMQ-broker te kunnen communiceren in erg veel talen beschikbaar; Java/JVM**,** Ruby, Python, .NET(C#/ASP), PHP, Perl, C/C++, Erlang, Lisp, Haskell en Ocaml.

Het is aan te raden om als broker voor het Easydus-systeem RabbitMQ te gaan gebruiken. Dit vanwege de volgende redenen:

- Het implementeert de gewenste functionaliteit
- Met is lichtgewicht, het neemt niet veel resources van de server in beslag
- Libraries voor veel programmeertalen beschikbaar
- Wordt veel toegepast in mission-critical systemen (betrouwbaarheid uit ervaring)
- Open Source, dus eventueel aan te passen aan specifieke eisen
- Het is gratis

6.3 Talen van de modules

Aangezien de libraries van RabbitMQ in vrijwel alle talen beschikbaar zijn, maakt het op dat gebied niet uit in welke talen deze geschreven gaan worden. Wat zwaarder meeweegt in deze keuze is de functionaliteit van de module. Het is aan te raden om het hele systeem in de eerste instantie in Java te bouwen, wegens de schaalbaarheid. Omdat Java applicaties in een virtuele machine draaien, kunnen deze makkelijk overgezet worden op een grotere server, of zelfs in de cloud gaan draaien zonder dat er speciale aanpassingen gemaakt hoeven te worden aan de applicatie. Er zou ook onderzocht kunnen worden in welke taal wat het snelste gaat, en de modules daarop aan te passen. Als bijvoorbeeld het verzenden van emails het snelste in C++ gaat, zou de mailer daar het beste in gebouwd kunnen worden.

6.4 Opdeling klanten en projecten

Het is van belang dat er bij het ontwerpen van het nieuwe systeem aan wordt gedacht dat er van sommige modules instanties moeten kunnen bestaan die specifiek voor één klant zijn. Als een klant aangeeft dat hij duizenden emails per dag denkt gaat te verzenden, is het handig om voor deze klant een aparte mailer te draaien, zodat de kans verdwijnt dat andere klanten hier hinder van ondervinden. Hetzelfde geldt voor alle veelgebruikte modules, zoals de Template Substitutie en de Databasemodule.

7 Conclusies en aanbevelingen

Gekeken naar de huidige opzet van Easydus, valt het systeem goed op te delen in verschillende modules. De communicatie tussen deze modules kan het beste via events gebeuren, omdat het huidige systeem ook redelijk eventgebaseerd is. Met andere woorden, op dit moment gebeurt er pas wat binnen het systeem zodra de inschrijver, of een andere externe bron, iets met het systeem doet. Denk hierbij aan het doen van een inschrijving, het doen van een betaling, enz. Om alle modules op de hoogte te houden van al deze events, is het aanbevolen een messaging systeem te gebruiken. Zo zal, zodra een inschrijving gebeurd, een broadcast-message worden verzonden, die iedere module laat weten dat er een inschrijving gedaan is. Iedere module doet op zijn beurt iets met deze message, of negeert hem(zo zal de database de inschrijving opslaan, terwijl de PDF generator misschien niks hoeft te doen). In tegenstelling tot de broadcast-messages zullen er ook messages gestuurd worden volgens het request/reply patroon; de mailer zal via dit patroon een document naar de PDF generator sturen, omdat de mailer pas verder kan zodra de PDF zelf is gegenereerd.

Bij het opsplitsen van de modules is er gekeken naar de functionaliteiten van Easydus. Daaruit is gebleken dat het systeem het beste in de volgende modules kan worden opgesplitst:

- Message Broker
- Workflow manager
- Superadmin GUI
- Admin GUI
- Enduser GUI
- Database
- PDF Generator
- Mailer
- Logger/Eventgenerator(debug)

Het is bij het opsplitsen erg belangrijk dat er een goede Message Broker wordt gekozen. Dit is namelijk het hart van het systeem, alle berichten worden door de broker in goed banen geleid. De aanbeveling voor de broker is om gebruik te maken van RabbitMQ, dit is een snelle, gratis en open source message broker waarvan uitgebreide documentatie beschikbaar is en er zijn libraries voor vele verschillende talen beschikbaar. Ook heeft RabbitMQ zich bewezen doordat het in gebruik is bij vele zogenaamde Mission-Critical systemen (bijvoorbeeld aandelenhandel).

De modules kunnen het beste in Java worden geschreven, omdat dit op vrijwel alle platformen draait, en goed schaalbaar is. Later kan er gekeken worden om specifieke modules in een andere taal te schrijven, om de efficiëntie nog verder te verhogen. Denk hierbij aan de database, deze zou op een server kunnen gaan draaien die speciaal ingericht is voor het opslaan en snel kunnen benaderen van grote hoeveelheden data.

V. Verklarende woordenlijst

Woord	Betekenis
AJAX	Een methode om data op een website te vernieuwen, zonder dat de hele pagina opnieuw geladen hoeft te worden
Apache	Een veelgebruikte, open source webserver
CSV	Comma Seperated Values, een manier om rijen data in een bestand op te slaan, waarbij alle waarden door een komma zijn gescheiden
event	Een gebeurtenis binnen een applicatie. Dit kan van alles zijn, van een muisklik tot een toetsaanslag
GUI	Graphical User Interface, een GUI is wat de gebruiker van een systeem ziet om interactie te kunnen hebben met dat systeem
HTML	Hypertext Markup Language, een taal die wordt gebruikt om de indeling en lay-out van een webpagina vast te leggen
Java	Een veelgebruikte programmeertaal, die op vrijwel ieder platform draait
LAMP	Linux Apache MySQL and PHP, dit is een veelgebruikte configuratie om een webserver in te richten. Alle elementen zijn open source en gratis te gebruiken
Messaging	Het verzenden van een bericht tussen twee computersystemen
MVC	Model View Controller, een ontwerpmethode voor applicaties, waarbij de interface gescheiden wordt van de logica en de beheerelementen
MySQL	Een gratis te gebruiken, open source database applicatie
Open Source	Dit betekend dat de broncode van de betreffende applicatie vrij toegankelijk is. De applicatie kan dus naar wens aangepast worden
PHP	PHP Hypertext Processor, een veelgebruikte open source programmeertaal. Deze taal wordt veel gebruikt om websites mee te maken, omdat het goed samenwerkt met Apache en MySQL

VI. Evaluatie

Mijn stage bij Inschrijfbalie heb ik als zeer interessant en leerzaam ervaren. Dit komt vooral omdat ik technieken zoals messaging wel kende, maar nog nooit op deze manier toegepast heb. Ook is het leuk om een systeem waar ik een tijdje aan heb gewerkt, helemaal opnieuw te mogen ontwerpen met de toekomst en groei in gedachte.

Ook heb ik geleerd dat het erg moeilijk is om in te schatten hoeveel tijd bepaalde dingen duren, en om überhaupt in te schatten wat er onderzocht moet worden. Zo wijkt het PID ook flink af met de onderzoeken die daadwerkelijk gedaan zijn, en ben ik niet toegekomen tot het technische ontwerp van het nieuwe systeem. Toch denk ik dat het advies dat naar voren komt in deze scriptie zeer nuttig is voor het bedrijf, omdat aan de hand van deze adviezen toch een vrij duidelijk ontwerp naar voren komt (het grootste deel van het 'voorwerk' is immers gedaan).

Verder vond ik het ook erg leuk om het huidige systeem te onderzoeken en te praten met de ontwerpers en bedenkers van het systeem, om zo de gedachte erachter te ontdekken, en deze over te dragen op het nieuwe ontwerp.

VII. Literatuurlijst

http://www.rabbitmq.com/

http://stackoverflow.com

http://www.cs.wustl.edu/~schmidt/POSA/POSA2/event-patterns.html

'Enterprise Integration Patterns' van Gregor Hohpe en Bobby Woolf (ISBN: 978-0321200686)

Wikipedia

VIII. Bijlagen

A. PID

B. Communicatieplan

C. Onderzoek huidig systeem