WIZZBIT
ICT DIENSTVERLENING & ADVIES

AFSTUDEERRAPPORT

EEN FACTUURPORTAL IN ONTWIKKELING AFMAKEN EN ALS CLOUDDIENST BESCHIKBAAR MAKEN

Afstuderen

Fabian Lachman

Den Haag, juni 2015

Versie 1.0

Van Gijnstraat 15 | 2288 GB | Rijswijk | T +31 070 394 32 00 | E info@wizzbit.nl | W www.wizzbit.nl
KvK 27135332 | BTW NL 822427552 B01 | IBAN NL98RABO0153705558 | BIC RABONL2U

REFERAAT

Dit verslag beschrijft het proces dat is doorlopen bij het uitvoeren van de afstudeeropdracht “Een factuurportal in ontwikkeling afmaken en als clouddienst beschikbaar maken”. De opdracht is uitgevoerd in de periode van okt 2014 tot en met juni 2015.

Descriptoren: Systeem ontwikkelen, Clouddienst, PHP, Symfony

VOORWOORD

Dit verslag is geschreven in het kader mijn afstudeeropdracht bij Wizzbit B.V. te Rijswijk. Het afstuderen is de laatste fase van mijn opleiding Informatica bij de Haagse Hogeschool. De afstudeeropdracht heeft plaats gevonden in de periode van oktober 2014 t/m juni 2015.

De opdracht bestond uit het afbouwen van een factuurportal en het ontwikkelen van een nieuwe systeem waarin multi-tenant applicaties ontwikkeld kunnen worden.

Bij deze wil ik Peter Rog, Hans de Wit en Tim Janss van Wizzbit bedanken voor de ondersteuning en feedback die ze mij hebben gegeven tijdens deze opdracht. Ook wil ik Alwine Lousberg en Paul Smit bedanken voor de leerzame feedback en begeleiding vanuit de Haagse Hogeschool.

1 INLEIDING

Dit document omschrijft het proces en werkzaamheden die zijn uitgevoerd tijdens mijn afstudeeropdracht bij het bedrijf Wizzbit.

In hoofdstuk 2 worden eerst de organisaties beschreven die betrokken zijn bij deze opdracht.

In hoofdstuk 3 wordt de opdracht beschreven.

In hoofdstuk 4 worden de werkwijze en rolverdeling binnen Wizzbit beschreven.

Deze afstudeeropdracht bestaat uit twee delen, het eerste deel van de opdracht is het afbouwen van een factuurarchief dat nu in ontwikkeling is voor een relatie van Wizzbit. In het tweede deel van de opdracht een nieuw systeem ontwikkelt waarmee applicaties zoals het factuurarchief aan meerdere klanten verkocht kan worden.

Vanaf hoofdstuk 5 wordt het eerste deel van de afstudeeropdracht beschreven.

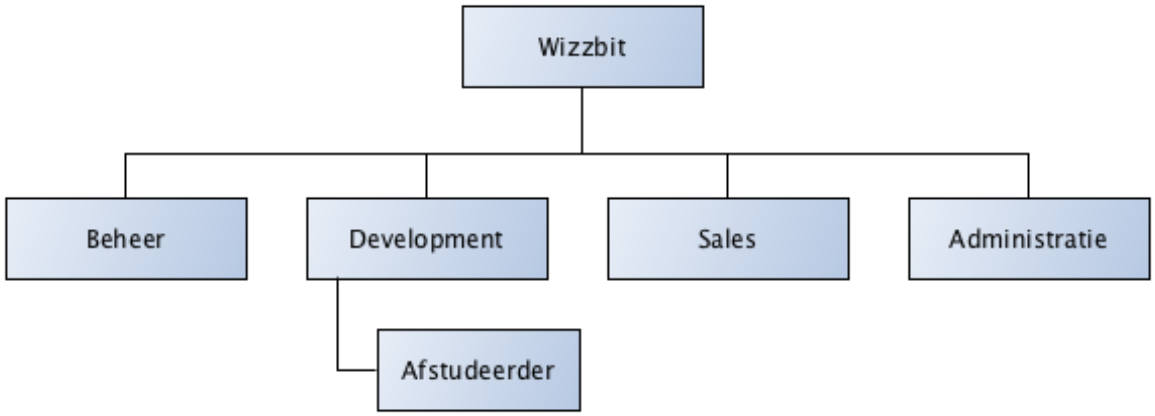
Vanaf hoofdstuk 6 wordt het tweede deel van de afstudeeropdracht beschreven.

2 ORGANISATIES

In dit hoofdstuk worden de organisaties beschreven die betrokken zijn bij deze opdracht.

2.1 WIZZBIT

Wizzbit is een ICT dienstverlener met voornamelijk klanten in het MKB. Het team van Wizzbit bestaat uit 10 personen: 1 directeur, 2 developers, 3 systeembeheerders, 1 sales medewerker, 1 administratief medewerker en 2 stagiairs. Wizzbit levert o.a. de volgende diensten en producten: Systeembeheer, helpdesk, domeinen en hosting, online back-up, VOIP bellen, website ontwikkeling en software ontwikkeling.

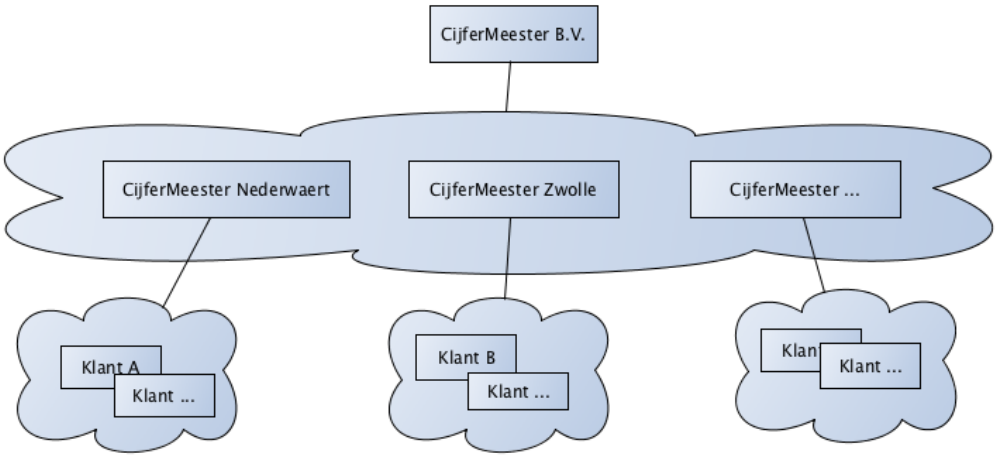


Figuur 1 Plaats van afstudeerder binnen Wizzbit

Zelf ben ik sinds 2006 werkzaam bij Wizzbit. De afstudeeropdracht wordt uitgevoerd naast en in combinatie met mijn normale werkzaamheden als softwareontwikkelaar. Deze opdracht wordt volledig door mij uitgevoerd.

2.2 CIJFERMEESTER

CijferMeester is een administratiekantoor voor boekhouding, belastingen en bedrijfsadvies. Door heel Nederland heeft CijferMeester zelfstandige kantoren die deel uit maken van de CijferMeester franchise.



Figuur 2 CijferMeester kantoren

Vanuit het hoofdkantoor biedt CijferMeester ondersteuning, trainingen en software aan alle kantoren. CijferMeester biedt zelfstandige kantoren de mogelijkheid zich aan te sluiten bij de CijferMeester franchise.

CijferMeester is een klant van Wizzbit. Wizzbit verzorgt het ICT-beheer en levert maatwerk software aan CijferMeester. Omgekeerd is Wizzbit ook een klant van CijferMeester, de facturatie van Wizzbit loopt via diensten van CijferMeester.

3 DE OPDRACHT

Wizzbit is voor CijferMeester een systeem aan het ontwikkelen voor het archiveren van factuurscans.

Aanleiding voor het ontwikkelen van dit systeem was de bijzondere situatie waar CijferMeester mee te maken heeft als overkoepelende organisatie voor de hele franchise.

3.1 PROBLEEMSTELLING

Ieder CijferMeester kantoor is zelfstandig ondernemer en heeft haar eigen klanten waarvoor de boekhouding geregeld wordt. Voor de verwerking van in- en verkoopfacturen wordt er gebruik gemaakt van een dienst (Newviews) waarbij er per verwerkte factuur een bedrag gerekend wordt. Bij lage volumes zijn de kosten te overzien, echter door het volume van de gehele franchise liepen de kosten op bij CijferMeester. Hierop is men gaan onderzoeken naar andere mogelijkheden om de prijs per scan drastisch te verlagen.

Het uitgangspunt voor CijferMeester was binnen Newviews het volgende proces bij het verwerken van de factuurscans:

1) Klant van CijferMeester scant de facturen en mailt deze naar een speciaal e-mailadres
2) De gescande facturen worden met automatische tekstherkenning digitaal beschikbaar gemaakt voor CijferMeester
3) De facturen (en bijbehorende scans) worden bewaard in een online archief (Newviews)
4) Facturen worden ingeboekt in online boekhoudpakket (Twinfield)
5) Klant kan bij iedere factuur in het online boekhoudpakket het origineel gescande document opvragen via een koppeling naar Newviews
6) Klant kan inloggen op het archief om ingeboekte facturen op te zoeken

Voor het digitaliseren van de facturen heeft CijferMeester een goedkoper alternatief gevonden (Scan Sys). Deze applicatie vervangt echter alleen stap 2 en biedt geen oplossing voor de andere stappen. Wizzbit ontwikkeld nu de CijferPortal, dit is het systeem wat een oplossing biedt voor stap 1, 3, 5 en 6. Als de CijferPortal gereed is zal er ook een migratie gedaan worden van het oude systeem naar de CijferPortal. Hierbij moeten alle factureren overgezet worden naar de CijferPortal en moet er rekening gehouden worden dat het oude systeem geen export mogelijkheden heeft.

3.2 DOELSTELLING

Deze opdracht heeft de volgende doelstellingen:

- Afbouwen CijferPortal
- Het systeem aan meerdere klanten kunnen verkopen
- Andere systemen ontwikkeld door Wizzbit ook op deze manier aan meerdere klanten kunnen verkopen

Afbouwen CijferPortal
Dit systeem wordt nu voor CijferMeester ontwikkeld maar moet uiteindelijk aan meerdere klanten verkoopbaar gemaakt kunnen worden. Omdat er met CijferMeester al afspraken gemaakt zijn wordt de CijferPortal eerst afgemaakt en daarna zal een systeem ontwikkeld worden waarin applicaties zoals de CijferPortal verkoopbaar gemaakt kunnen worden. Het afbouwen van de CijferPortal omvat het eerste gedeelte van de afstudeeropdracht.

De CijferPortal aan meerdere klanten verkopen
Wizzbit heeft meerdere applicaties ontwikkeld die steeds door één klant gebruikt worden. Wizzbit wil nu de tijd en kosten die in de ontwikkeling van een applicatie worden gestopt beter gaan benutten door de ontwikkelde applicaties aan meerdere klanten te kunnen verkopen zodat er optimaal rendement gehaald kan worden uit een ontwikkelde applicatie. In het tweede deel van de afstudeeropdracht wordt het afgebouwde systeem aangepast zodat deze aan meerdere klanten verkocht kan worden.

De kennis en ervaring die wordt opgedaan bij deze opdracht kan dan ook gebruikt worden voor de andere applicaties die door Wizzbit ontwikkeld zijn.

Focus
In mijn afstuderen wil ik voor het aantonen van mijn beroepstaken de focus vooral leggen op het tweede deel van mijn afstudeeropdracht (applicatie aan meerdere klanten verkoopbaar maken). Het afbouwen van de CijferPortal is wel een noodzaak om te kunnen beginnen met het tweede deel maar de meeste tijd en energie in het schrijven van mijn afstudeerdossier gaat in het tweede deel zitten.

4 AANPAK

In de volgende hoofdstukken worden de werkwijze binnen Wizzbit beschreven gevolgd door een planning voor de afstudeeropdracht en de producten die opgeleverd gaan worden.

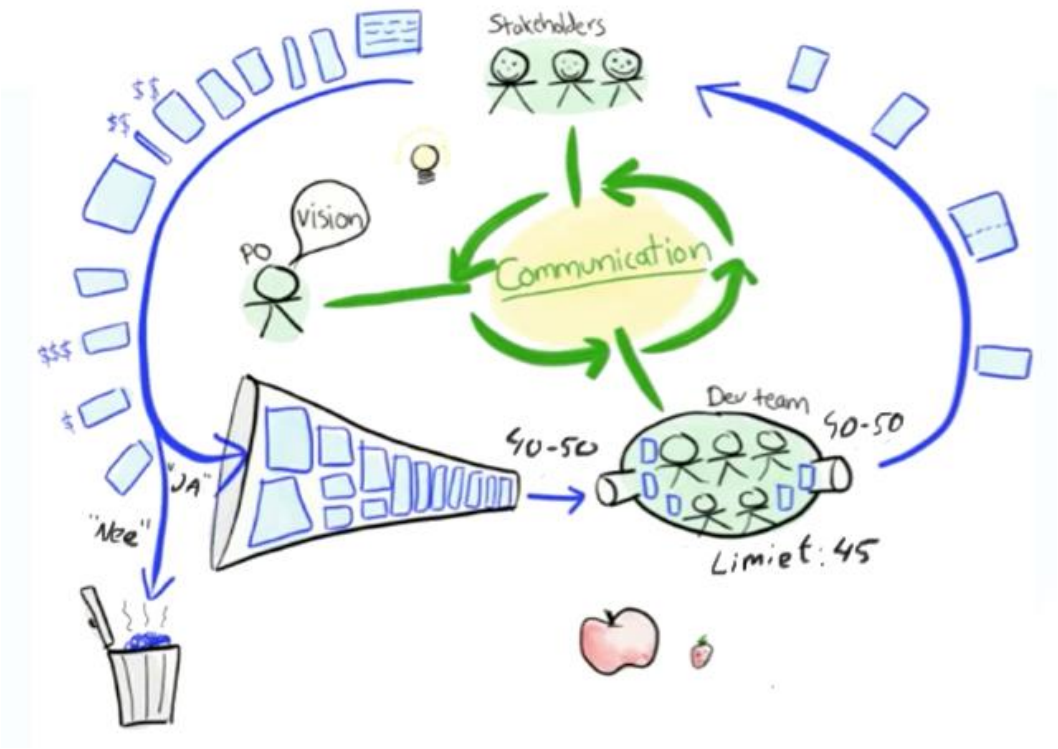
4.1 WERKWIJZE

De werkwijze binnen Wizzbit is Agile-gericht, er wordt gebruik gemaakt van een eigen vorm van SCRUM waarbij er gebruik gemaakt wordt van sprints van 1-2 weken. De klanten van Wizzbit verwachten dat Wizzbit flexibel is en deze werkwijze sluit daar goed op aan. De focus ligt op het vroeg opleveren van bruikbare software.

In de werkwijze wordt er steeds gebruik gemaakt van een planning voor de komende 3 weken en een backlog. Deze staan gevuld met taken van verschillende projecten. Iedere taak krijgt punten toegewezen aan de hand van SCRUM-pokeren. Aan het begin van iedere week worden de taken van de vorige week besproken en wordt er een telling gemaakt van de afgeronde taken (in punten), van deze puntentelling houden we het gemiddelde bij zodat we weten hoeveel er ingepland kan worden voor de komende week.

Bij het begin van een project worden in grote lijnen de wensen van de klant helder gemaakt, afhankelijk van de grootte van het project wordt er al dan niet een functioneel ontwerp gemaakt. Vervolgens worden er scrum-punten toegekend aan de werkzaamheden die gedaan moeten worden.

Rolverdeling
Het onderstaande plaatje geeft een beeld van hoe er bij Wizzbit gewerkt wordt, de rollen worden als volgt vervult: het **Dev Team** bestaat uit mijzelf en een collega-ontwikkelaar, de **Stakeholders** zijn onze klanten en de **Product Owner** (PO) is de directeur van Wizzbit.



Figuur 3 Product Owner, Dev Team en Stakeholders

In de bovenstaande schets zie je tegen de klok in een stroom van wensen die vanuit de stakeholders komen. Uiteindelijk bepaald de Product Owner in overleg met de stakeholders en het Dev Team wat wel of niet in de backlog terecht komt. Vervolgens is het Dev Team iedere sprint bezig met 40-50 “punten” aan werkzaamheden.

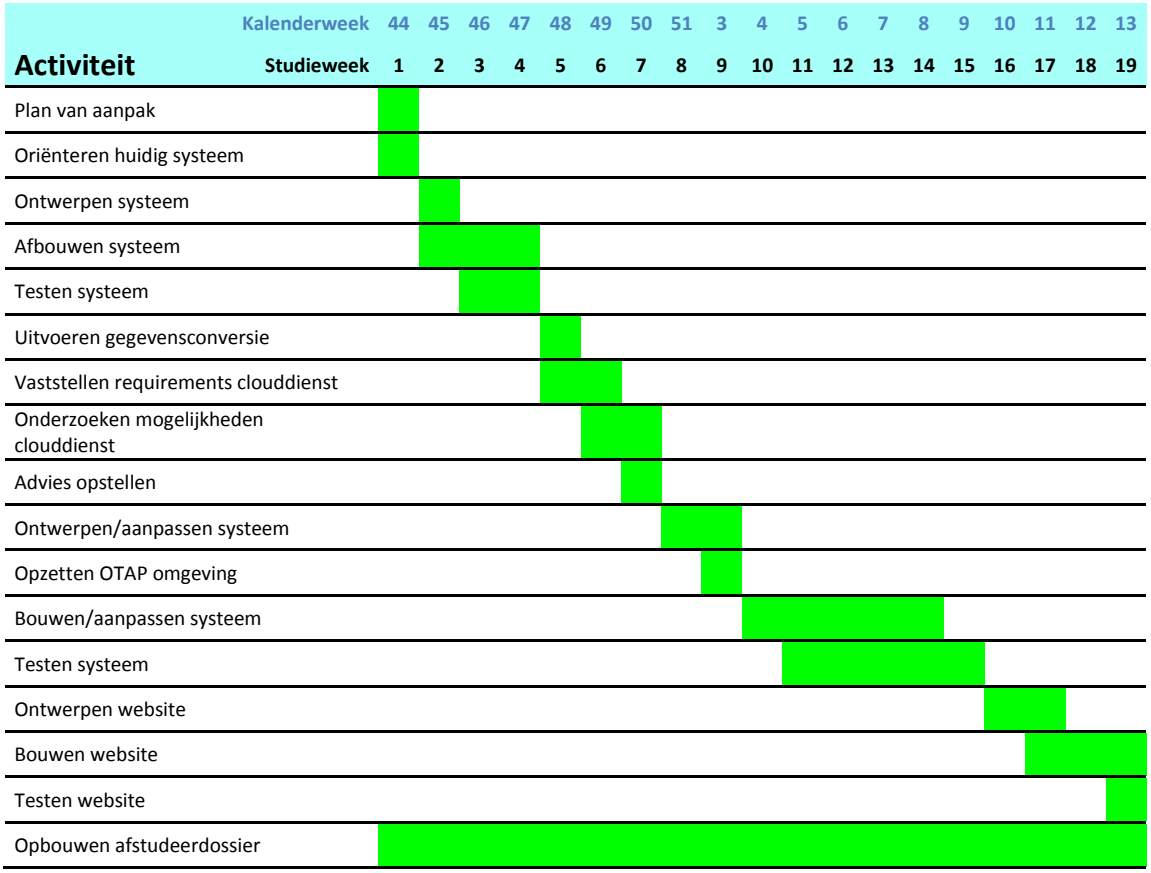
Risico gebaseerd testen
Om de tijd die we benutten voor het testen optimaal te gebruiken maken we gebruik van risico gebaseerd testen. Met deze methode wordt bepaald welke delen van het systeem meer aandacht krijgen voor het testen dan andere.

Issue tracking
Voor het bijhouden van bugs en openstaande punten maken we gebruik van YouTrack als bug tracking systeem.

UML
UML wordt gebruikt als modelleertaal.

Versiebeheer
We maken gebruik van Git als versiebeheersysteem.

4.2 PLANNING


Kalenderweek 44 45 46 47 48 49 50 51 3 4 5 6 7 8 9 10 11 12 13
Activiteit Studieweek 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Plan van aanpak
Oriënteren huidig systeem
Ontwerpen systeem
Afbouwen systeem
Testen systeem
Uitvoeren gegevensconversie
Vaststellen requirements clouddienst
Onderzoeken mogelijkheden clouddienst
Advies opstellen
Ontwerpen/aanpassen systeem
Opzetten OTAP omgeving
Bouwen/aanpassen systeem
Testen systeem
Ontwerpen website
Bouwen website
Testen website
Opbouwen afstudeerdossier

4.3 PRODUCTEN

1. **Plan van aanpak**: In dit document staat beschreven wat de opdracht inhoudt, welke activiteiten worden uitgevoerd, wat de eindproducten zijn en wat de planning is.
2. **Afgebouwde factuurportal**: Dit is het nieuwe systeem dat wordt opgeleverd wordt bestaande uit:
    a. **Een portal waar CijferMeester kantoren en klanten op kunnen inloggen.**
    b. **Een systeem voor de verwerking van opgestuurde facturen**
3. **Scraper-tool:** Een script waarmee de oude facturen en administraties uit Newviews kunnen worden overgezet naar de nieuwe factuurportal.
4. **Geconverteerde facturen:** De facturen zijn overgezet en toegankelijk in de nieuwe factuurportal.
5. **Requirementsrapport**: In dit document staat beschreven aan welke voorwaarden de clouddienst moet voldoen.
6. **Adviesrapport**: In dit document staat beschreven wat advies is dat volgt uit de oriëntatie van de huidige omgeving, onderzoek naar mogelijkheden nieuwe omgeving en het requirementsrapport.
7. **Architectual Description**: In dit document staat omschreven uit elke systemen, componenten en lagen het gehele systeem bestaat, wat de relatie is tussen de verschillende systemen en hoe deze met elkaar communiceren.
8. **OTAP omgeving:** Dit is de omgeving en de documentatie die opgezet wordt zodat de software componenten die ontwikkeld worden door de verschillende fases van de OTAP methodiek kunnen worden doorlopen.
9. **Gegevensmodel cloudomgeving/dienst:** In deze documenten staan de structuur en relaties van de gegevensobjecten beschreven.
10. **Glossary**: In dit document staan termen beschreven die gebruikt worden binnen dit project.
11. **Testplan:** In dit document staat beschreven wat er getest wordt en hoe het testtraject er uit gaat zien.
12. **Testrapport:** In dit document staat beschreven: de resultaten van de uitgevoerde testen, evaluatie van de testproces en een beschrijving van eventuele vervolgstappen die nodig zijn.
13. **Nieuw systeem als clouddienst:** Het nieuwe systeem werkend in een cloudomgeving.
14. **Nieuwe website voor de clouddienst:** Een website waarop de nieuwe clouddienst verkocht wordt.

5 SYSTEEM IN ONTWIKKELING AFBOUWEN

Het eerste deel van dit project bestaat uit het afbouwen van de CijferPortal, dit is het huidige systeem dat in ontwikkeling is. De CijferPortal bestaat uit een aantal deelsystemen, in het volgende hoofdstuk wordt de huidige situatie beschreven en welke deelsystemen er nog gebouwd moeten worden.

5.1 HUIDIGE SITUATIE

Een deel van het systeem is al gebouwd. Dit heb ik in een eerder stadium gedaan. Toen heb ik ook de requirements vastgesteld en een omschrijving van de architectuur gemaakt. Deze zijn als bijlagen beschikbaar (Bijlage 1 - Requirementsrapport.docx en Bijlage 2 - Architectual Description.docx). Delen hiervan heb ik het volgende hoofdstuk opgenomen.

De CijferPortal is nu in gebruik met 1 administratie van Wizzbit en heeft een tijdje proef gedraaid om de laatste kinkjes te verhelpen.

Dit betekent dat er nu factuurscans die verwerkt zijn in Scan Sys kunnen worden bewaard in de CijferPortal en dat klanten kunnen inloggen op de CijferPortal om ze weer in te zien.

Wat nog ontbreekt is mogelijkheid om factuurscans te kunnen mailen naar het systeem, dit zal mogelijk gemaakt worden met de CijferCatcher; een systeem die wij gaan ontwikkelen wat ervoor gaat zorgen dat factuurscans uit bijlagen in een e-mailbericht in de juiste import map van Scan Sys worden geplaatst voor verdere verwerking.

Verder moeten ook alle oude administraties die nu in Newviews staan worden overgezet naar de CijferPortal. Dit is nu met handmatige stappen gedaan voor de 1e administratie van Wizzbit. Voor dit proces moet een tool gemaakt worden waarmee dit automatisch kan op een generieke manier zodat het voor alle kantoren gebruikt kan worden. Deze tool heet de CijferSlurper.

Als laatste zijn er een aantal extra functionaliteiten nodig binnen de CijferPortal, deze worden benoemd in hoofdstuk 5.

5.2 REQUIREMENTS

Dit zijn de requirements zoals deze waren vastgesteld. In het systeem ontbreken op dit moment nog de requirements die in blauw gemarkeerd zijn.

Business requirements

Code	Requirement	Bron
B1	De klanten van CijferMeester willen factuurscans digitaal kunnen opsturen	Oriëntatie huidige situatie
B2	De klanten van CijferMeester willen factuurscans van verwerkte facturen online kunnen opvragen	Oriëntatie huidige situatie
B3	CijferMeester wil op een centrale manier met haar klanten kunnen communiceren	Oriëntatie huidige situatie

B4	CijferMeester wil kosten besparen	Oriëntatie huidige situatie

Business Rules

Code	Requirement	Bron
R1	Facturen a) Een factuur kan een inkoop- of verkoopfactuur zijn b) Een factuur behoort tot een administratie c) Een factuur heeft een boekstuknummer d) Een factuur heeft een datum e) Een factuur heeft een totaalbedrag f) Een factuur heeft een relatienaam g) Een factuur heeft een factuurnummer	Oriëntatie huidige situatie
R2	Een administratie is gekoppeld aan Twinfield met een administratiecode	Oriëntatie huidige situatie
R3	Een kantoor kan een of meerdere administratie beheren	Oriëntatie huidige situatie

Systeemrequirements

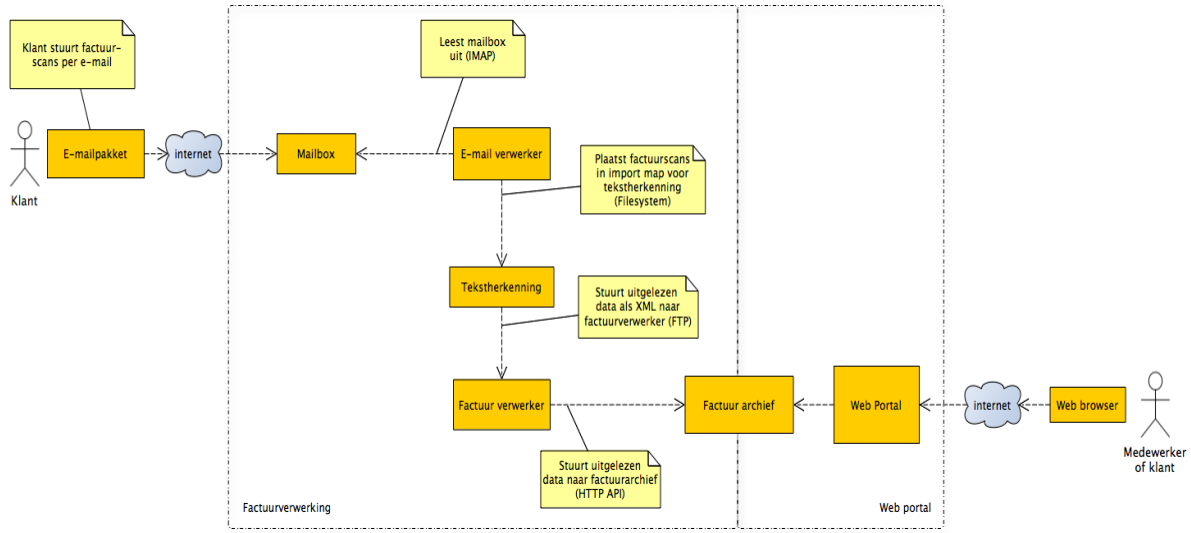
Code	Requirement	Traceerbaarheid	Bron
S1	Het systeem moet per mail opgestuurde facturen plaatsen in een import map van Scan Sys	B1	Oriëntatie huidige situatie
S2	Het systeem moet aan de hand van het e-mailadres (AAN) herkennen in welk mapje de facturen geplaatst moeten worden	B1	Oriëntatie huidige situatie
S3	Het systeem moet aan de klant die het mailtje verstuurt een ontvangstbevestiging terugsturen	B1	Oriëntatie huidige situatie
S4	Het systeem moet de facturen die verwerkt zijn door Scan Sys weer kunnen inlezen	B2	Oriëntatie huidige situatie
S5	Het systeem moet ervoor zorgen dat er directe link beschikbaar is naar een factuur zodat deze vanuit Twinfield aangeroepen kan worden	B2	Oriëntatie huidige situatie
S6	Het systeem moet voor meerdere kantoren van CijferMeester te gebruiken zijn		Oriëntatie huidige situatie

Gebruikersrequirements

Code	Requirement	Traceer-baarheid	Bron
G1	Een klant moet een factuur als bijlage kunnen opsturen naar het systeem	B1	Oriëntatie huidige situatie
G2	Een klant moet kunnen inloggen op het systeem	B2	Oriëntatie huidige situatie
G3	Een klant moet de verwerkte factureren van zijn of haar administratie kunnen opvragen	B2	Oriëntatie huidige situatie
G4	Een klant moet kunnen zoeken naar facturen	B2	Oriëntatie huidige situatie
G5	Een klant moet vanuit Twinfield een factuur kunnen	B2	Oriëntatie huidige situatie

	openen		
G6	Een medewerker moet klanten kunnen beheren onder administraties	S6	Oriëntatie huidige situatie
G7	Een medewerker van het hoofdkantoor moet kantoren kunnen beheren	S6	Oriëntatie huidige situatie
G8	Een medewerker van het hoofdkantoor moet medewerkers kunnen beheren	S6	Oriëntatie huidige situatie
G9	Een medewerker moet documenten digitaal beschikbaar kunnen maken voor een klant	B3	Oriëntatie huidige situatie
G10	Een klant moet documenten kunnen downloaden	B3	Oriëntatie huidige situatie

De onderstaande afbeelding toont het totaalplaatje zoals het systeem volgens de eerder gemaakte architectuuromschrijving eruit moet gaan zien.

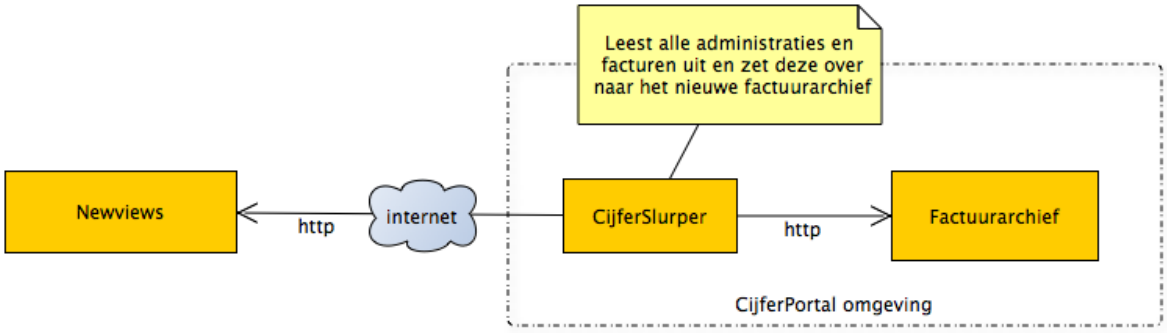


Figuur 4 De functionele elementen van het systeem

Wat op dit moment nog ontbreekt zijn de volgende elementen:

- E-mailverwerker (De CijferCatcher)
- Deel van de Web Portal (De CijferPortal)

Daarnaast moet er voor de overgang naar het nieuwe systeem een migratie gedaan worden van alle oude factuurscans in het oude systeem naar het nieuwe systeem. Het oude systeem heeft geen export functionaliteiten dus er moet een tool ontwikkeld worden waarmee dit gedaan kan worden (De CijferSlurper).



Figuur 5 De CijferSlurper tussen het oude en het nieuwe systeem

5.3 CIJFERPORTAL

Aan het begin van het afstudeertraject is een deel van de CijferPortal al gebouwd. In de huidige fase zijn er nog een aantal wensen die gerealiseerd moeten worden:

- Gebruikers moeten meerdere factuurscans tegelijk kunnen openen
- Er moet een telling gemaakt kunnen worden van het aantal verwerkte facturen per klant per maand
- Het moet mogelijk zijn om facturen ook weer te verwijderen

Deze wensen zijn erbij gekomen aan de hand van feedback op eerdere versies van het systeem.

REQUIREMENTS

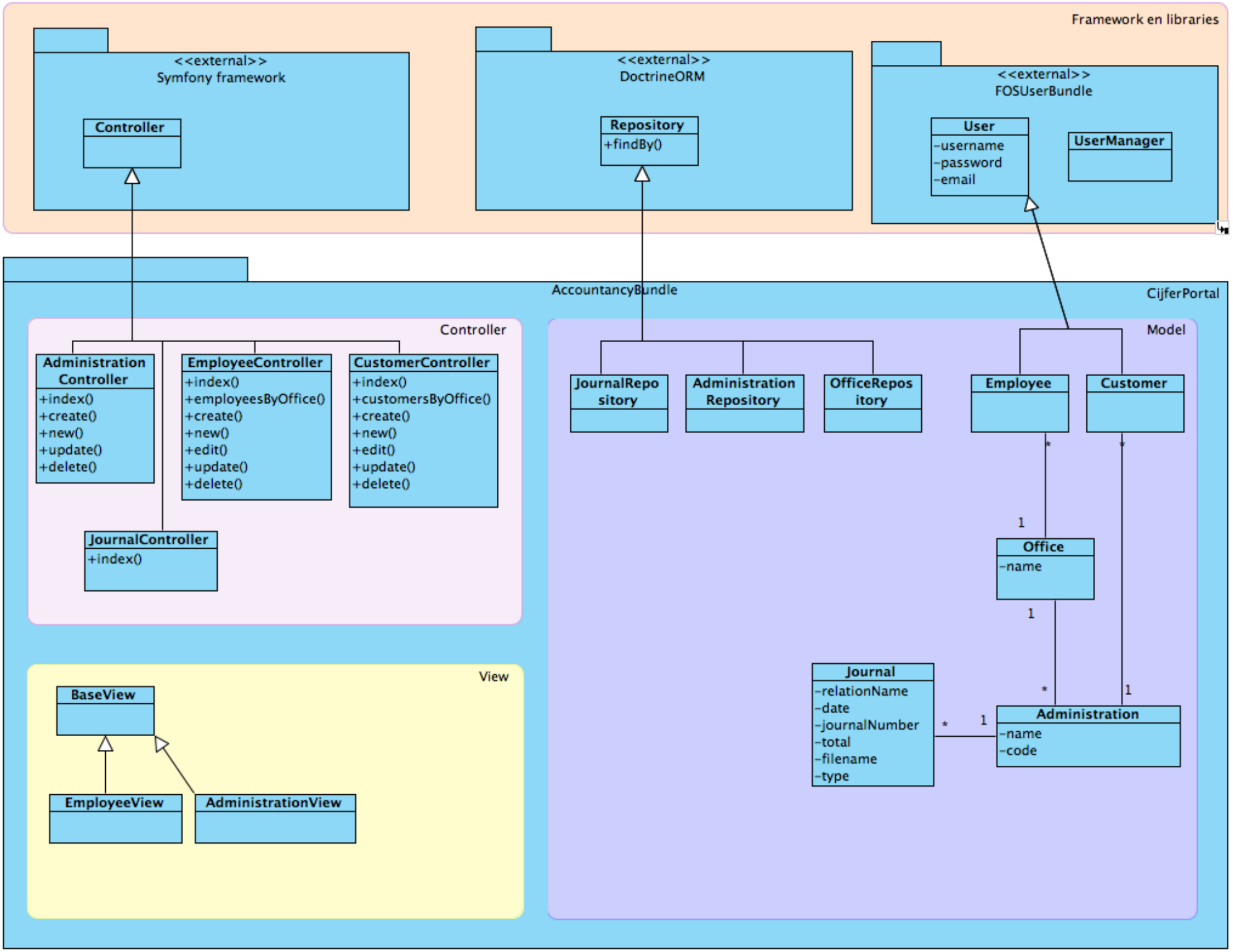
Meerdere factuurscans tegelijk openen
In het oude systeem was er een handige manier om meerdere factuurscans tegelijk te openen. De facturen werden dan samengevoegd tot 1 pdf waar je snel doorheen kon scrollen. In het nieuwe systeem ontbreekt deze functionaliteit.

Maandtelling
Het hoofdkantoor moet weten hoeveel factureren er worden verwerkt door de kantoren en heeft dit nodig in een Excel-overzicht per maand.

Verwijderde facturen
Het moet mogelijk worden om facturen te verwijderen. Wel moet in de maandtelling te zien blijven hoeveel facturen er zijn toegevoegd (inclusief verwijderd).

HUIDIG ONTWERP

De CijferPortal is al deels gebouwd. Hier zijn geen complete ontwerpen van beschikbaar. Voordat ik de nieuwe dingen ben gaan bouwen heb ik een klassendiagram gemaakt van de huidige CijferPortal waarin de belangrijkste klassen worden beschreven. Deze is op de volgende pagina te zien.



Figuur 6 Design klassendiagram CijferPortal

De CijferPortal maakt gebruik van een Model-View-Controller design pattern en is gemaakt met behulp van het Symfony framework, dit is een PHP framework.

Het Symfony framework kent vele bundels, dit zijn herbruikbare stukjes functionaliteit samengevoegd in bundel. De CijferPortal maakt gebruik van de volgende bundels:

FOSUserBundle: Deze bundel zorgt voor inlogfunctionaliteit en stelt de volgende klassen beschikbaar:

User: deze klasse is verantwoordelijk voor het opslaan van gegevens bij een gebruiker.

UserManager: deze klasse is verantwoordelijk voor het ophalen/opslaan van gebruikers.

DoctrineORM: Dit is een Object-Relational-Mapper (ORM) en zorgt voor een vertaling van entiteiten naar een relationele database.

Repository: Deze klasse is verantwoordelijk voor het opvragen van entiteiten

De volgende klassen zijn ontworpen voor de CijferPortal:

Office: deze klasse is verantwoordelijk voor het opslaan van gegevens van een kantoor

Administration: deze klasse is verantwoordelijk voor het opslaan van gegevens bij een administratie

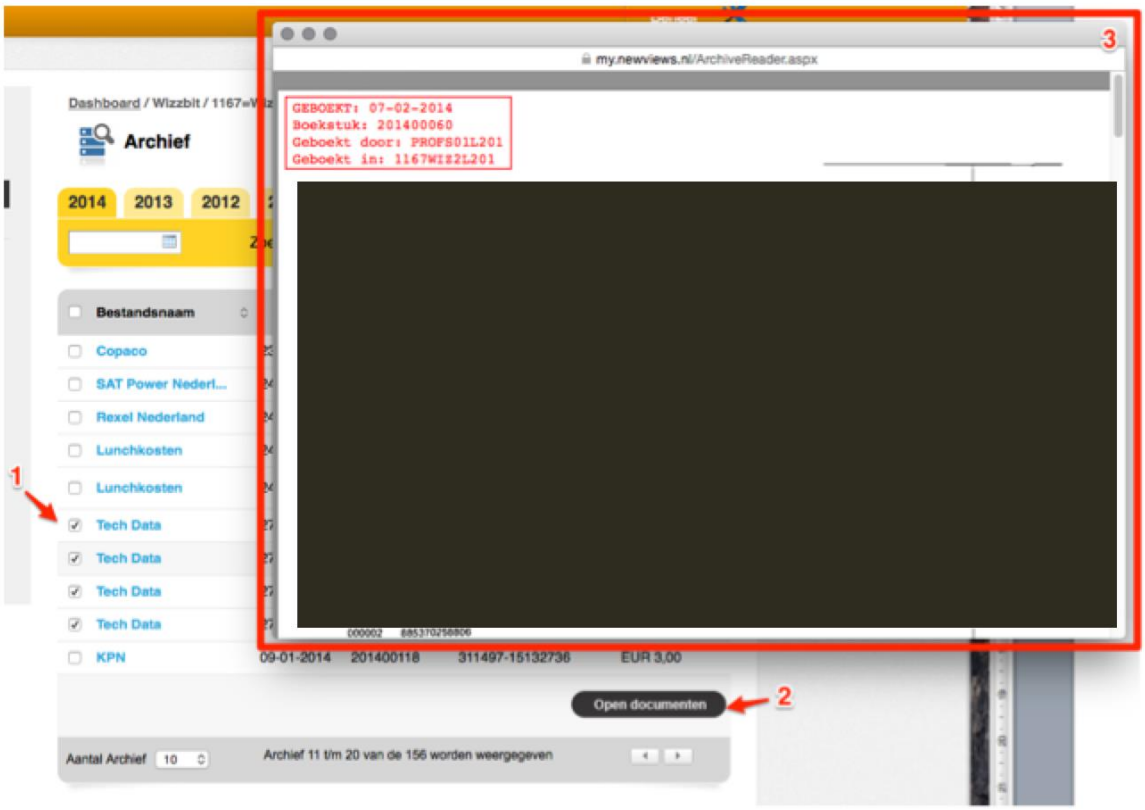
Journal: Deze klasse is verantwoordelijk voor het opslaan van gegevens bij een factuur

Employee: Deze klasse is verantwoordelijk voor het opslaan van gegevens van een medewerker

Customer: Deze klasse is verantwoordelijk voor het opslaan van gegevens van een klant

MEERDERE FACTUREN TEGELIJK OPENEN

In de onderstaande afbeelding is te zien hoe in het oude systeem meerdere facturen tegelijk konden worden geopend.



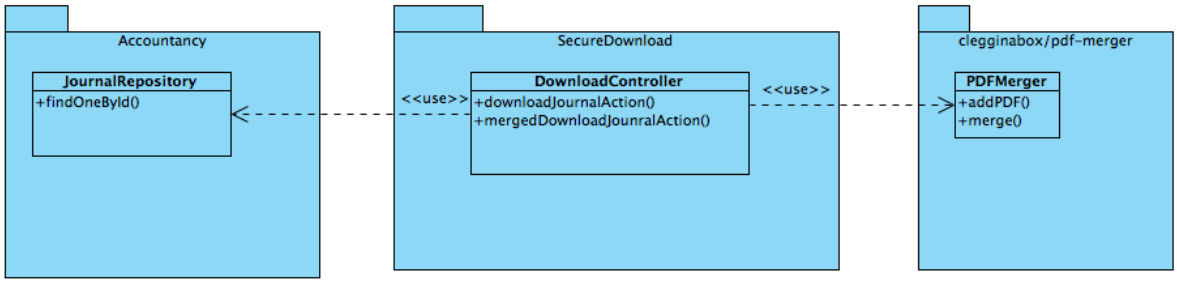
Figuur 7 Meerdere facturen aanvinken in het oude systeem

1) Vink factureren aan
2) Klik Open documenten
3) De factuurscans worden samengevoegd geopend

ONTWERP

Het downloaden van de factuurscans gebeurt nu via de DownloadController, deze klasse zorgt ervoor dat bestanden niet zomaar door iedereen gedownload kunnen worden. De DownloadController heeft 1 methode voor het downloaden van 1 factuurscan.

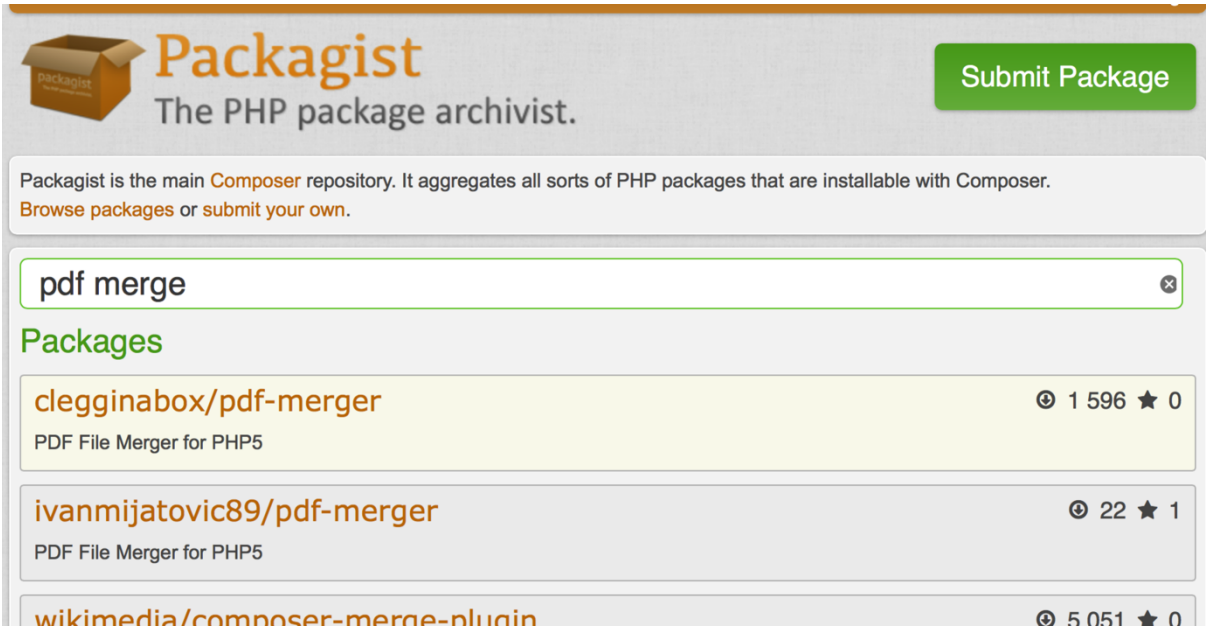
Om meerdere factuurscans als 1 samengevoegde pdf te kunnen downloaden heb ik in het ontwerp de DownloadController uitgebreid met 1 extra methode die meerdere facturen kan opvragen. Deze methode zal gebruik maken van de PDFMerger-klasse, dit is een externe library voor het samenvoegen van pdf bestanden.



Figuur 8 Klassendiagram m.b.t downloaden factuur in CijferPortal

WERKZAAMHEDEN

Voordat ik het ontwerp had aangepast ben ik eerst gaan onderzoeken of er al een bestaande oplossing was voor het samenvoegen van pdf bestanden. Daarbij heb ik het pakket clegginabox/pdf-merger gevonden.



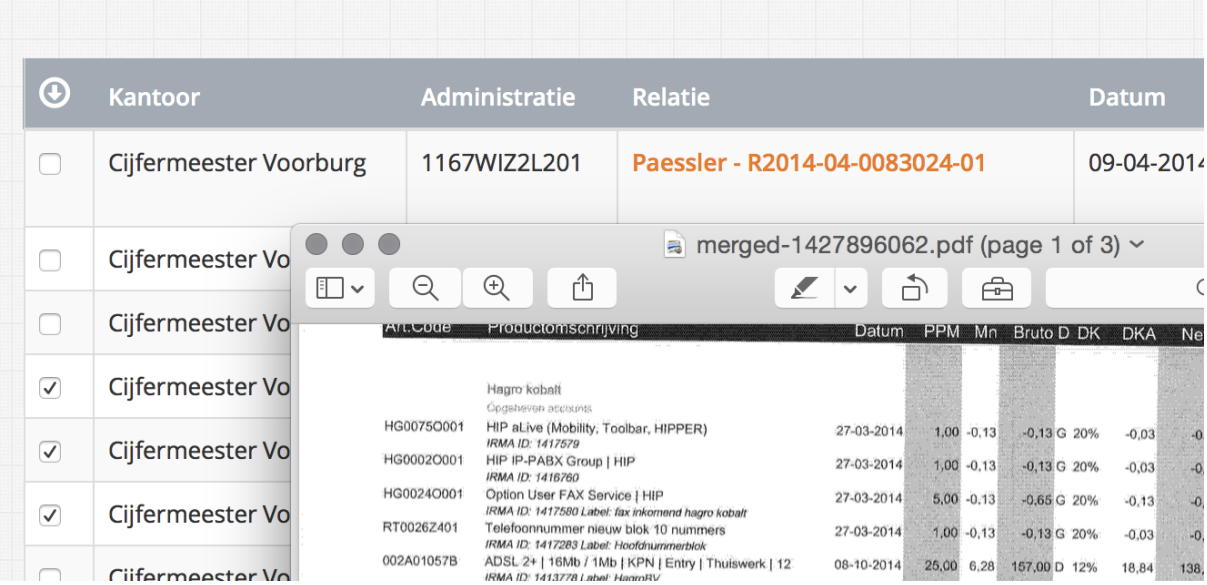
Figuur 9 Packagist is een catalogus van PHP-pakketten

Ik heb eerst onderzocht of dit pakket geschikt was, hierbij heb ik gelet op een aantal punten:

- Zijn er veel problemen met dit pakket? Zijn er veel (openstaande) bug reports.
- Is het pakket goed gedocumenteerd?
- Is het pakket veilig te gebruiken? (Broncode doorgenomen)
- Is het pakket vrij te gebruiken (licentie)?
- Is het een veelgebruikt pakket? Dit is het eerste punt waar ik op gelet heb. Als het ene pakket veel minder gebruikers heeft dan een ander dan is dat meestal omdat een (of meer) van de eerder genoemde punten niet in orde zijn.

Het pakket heb ik uiteindelijk verwerkt in het klassendiagram.

Bij het bouwen heb ik zoals beschreven in het klassendiagram DownloadController uitgebreid met 1 methode. Vervolgens heb ik in de front-end de vinkjes en een download knop toegevoegd net zoals deze ook in het oude systeem aanwezig waren.



Figuur 10 Meerdere facturen tegelijk downloaden in het nieuwe systeem

TESTEN

Testen pdf-merge pakket
Als eerste is het pdf-merge pakket getest, los van de CijferPortal om in ieder geval vast te stellen dat het pakket werkt zoals gedocumenteerd. Hierbij heb ik lokaal op mijn ontwikkelomgeving het pakket binnengehaald en een test script gemaakt die een paar pdfjes samenvoegt.

Downloaden meerdere facturen
De nieuwe functionaliteit is getest door meerdere facturen aan te vinken en dan op de download knop te klikken. Om vast te stellen dat alles goed werkt moet het resultaat net als in het oude systeem de facturen samenvoegen en één nieuw samengevoegd document tonen.

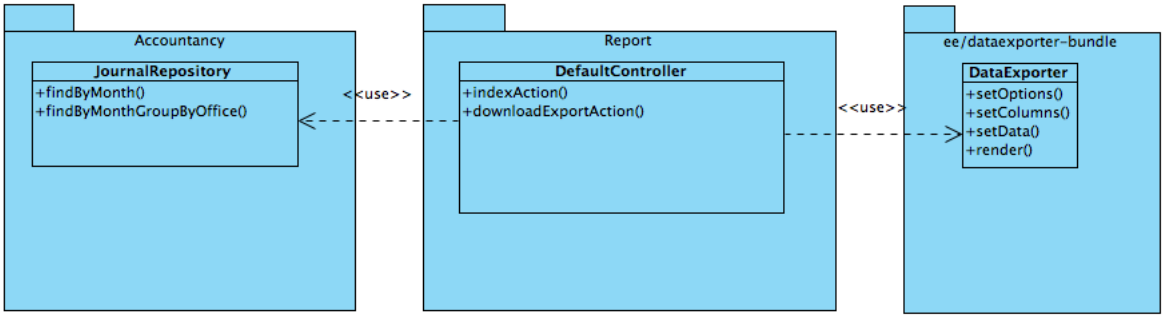
Download 1 factuur
Het downloaden van één factuur hoort bij de regressietest, hierbij is vastgesteld dat de oorspronkelijke functie waarbij één factuur los gedownload kan ook nog steeds werkt.

MAANDTELLING

Het hoofdkantoor moet weten hoeveel facturen er worden verwerkt door de kantoren en heeft dit nodig in een Excel-overzichtje per maand.

ONTWERP

Ik heb het klassendiagram uitgebreid met een Report-package, die in de toekomst eventueel kan worden uitgebreid met andere rapporten. Ook hier heb ik weer gebruik gemaakt van een externe library; de ee/dataexplorter-bundle maakt het mogelijk om gegevens naar een Exceldocument te exporteren.



TESTEN

Klopt de telling?
Om controleren of de maandtelling goed werkt heb ik het volgende getest:
Wat zijn het daadwerkelijke aantal facturen van een vorige maand op het bestandssysteem? Dit is vast te stellen door de werkelijke pdf bestanden op het bestandssysteem te tellen (Buiten de CijferPortal om). Dit getal heb ik vergeleken met het resultaat van de nieuwe maandtelling functie.

Deze test heb ik uitgevoegd met verschillende administraties en maanden. Ook heb ik een factuur toegevoegd om vast te stellen het aantal met 1 verhoogt wordt.

Acceptatietest
Bij de acceptatietest is naar voren gekomen dat de telling niet op een handige manier werd getoond, namelijk per administratie. CijferMeester moest dan nog zelf de administraties van een kantoor bij elkaar optellen. Dit is daarna aangepast naar aantallen per kantoor.

FACTUREN VERWIJDEREN

Indien er een factuurscan verkeerd in de CijferPortal staan dan wil CijferMeester de mogelijkheid hebben om deze te verwijderen.

ONTWERP

Om dit mogelijk te maken heb ik in het klassendiagram de JournalController aangepast, deze klasse is verantwoordelijk voor het afhandelen van alle acties die worden uitgevoerd op een factuur (boekstuk/journal).

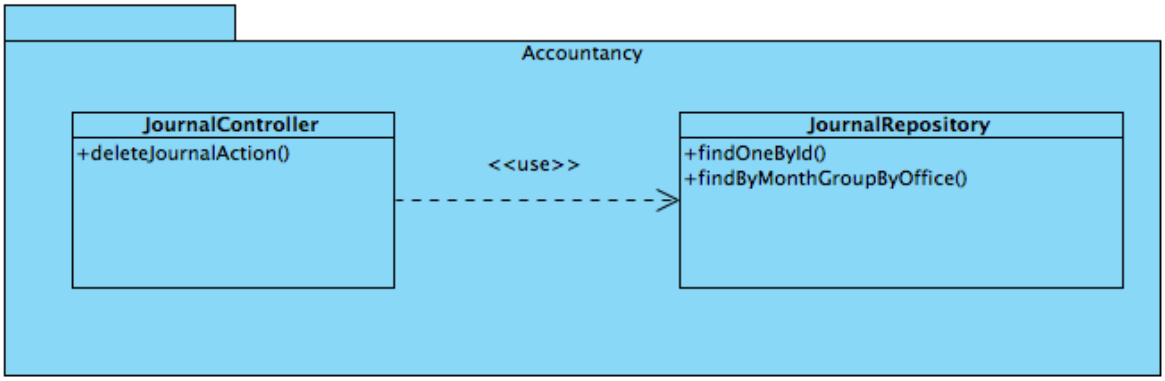
Bij het verwijderen heb ik rekening gehouden met het eventueel terughalen van verwijderde facturen. Dit was niet direct een wens van CijferMeester maar wel iets wat veel problemen achteraf kan voorkomen.

Door zelf een veld isDeleted (true/false) toe te voegen aan de Journal-klasse kan bepaald worden of een factuur verwijderd is of niet.

Voordat ik deze wijziging ben gaan doorvoeren heb ik onderzocht of er bestaande oplossingen zijn en daarbij heb ik de DoctrineExtensionsBundle gevonden. Deze bundle is een uitbreiding op het ORM waar nu gebruik van wordt gemaakt en biedt de volgende features:

- **Tree** - this extension automates the tree handling process and adds some tree specific functions on repository. (**closure**, **nestedset** or **materialized path**)
- **Translatable** - gives you a very handy solution for translating records into different languages. Easy to setup, easier to use.
- **Sluggable** - urlizes your specified fields into single unique slug
- **Timestampable** - updates date fields on create, update and even property change.
- **Blameable** - updates string or association fields on create, update and even property change with a user name resp. reference.
- **Loggable** - helps tracking changes and history of objects, also supports version management.
- **Sortable** - makes any document or entity sortable
- **Translator** - explicit way to handle translations
- **Softdeleteable** - allows to implicitly remove records
- **Uploadable** - provides file upload handling in entity fields
- **Reference Integrity** - provides reference integrity for MongoDB, supports 'nullify' and 'restrict'

In de huidige situatie heb ik alleen de **Softdeletable** feature nodig, door gebruik te maken van deze bundle hoef ik niet opnieuw het wiel uit te vinden en is het ORM ook alvast uitgebreid met andere features die goed van toepassing kunnen zijn in de toekomst (bijvoorbeeld Loggable om wijzigingen bij te houden).



Figuur 11 Betrokken klassen bij verwijderen van factuur

TESTEN

Ik heb de volgende tests uitgevoerd om vast te stellen dat het verwijderen naar wens werkt:

Factuur verwijderen
Is deze na verwijderen nog te downloaden? (directe download link)
Is de factuur niet meer in de lijst/zoekresultaten zichtbaar?

Regressietest
Deze functionaliteit heeft impact om andere delen van het systeem, dus ook eerdere tests zijn opnieuw uitgevoerd om vast te stellen dat alles nog steeds naar behoren werkt. Een belangrijke hierbij is de maandtelling. Klopt deze nog als een factuur verwijderd is?

5.4 CIJFERCATCHER

In het oude systeem hadden de kantoren van CijferMeester ieder een eigen omgeving bij Newviews waar naartoe gemaild kon worden. Met de overstap naar Scan Sys ontbreekt deze functionaliteit, bij Scan Sys komt namelijk alles binnen op 1 omgeving. De CijferCatcher moet ervoor gaan zorgen dat binnenkomende scans worden uitgesplitst per administratie.

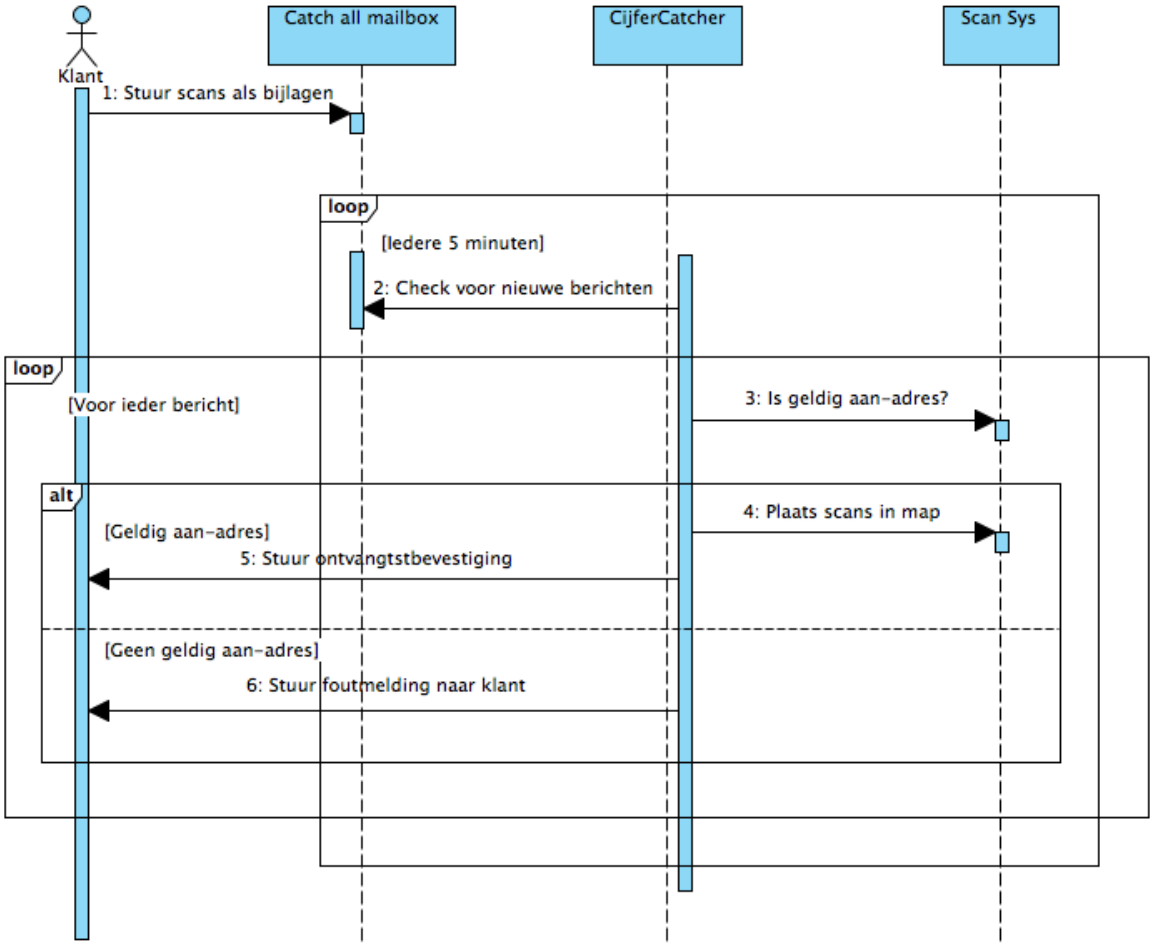
REQUIREMENTS

- S1) Het systeem moet per mail opgestuurde facturen plaatsen in een import map van Scan Sys
- S2) Het systeem moet aan de hand van het e-mailadres (AAN) herkennen in welk mapje de facturen geplaatst moeten worden
- S3) Het systeem moet aan de klant die het mailtje verstuurt een ontvangstbevestiging terugsturen

Om ervoor te zorgen dat de mailtjes in de juiste import map komen moet de CijferCatcher gaan filteren op het adres waar de mailtjes naartoe gestuurd worden.

ONTWERP

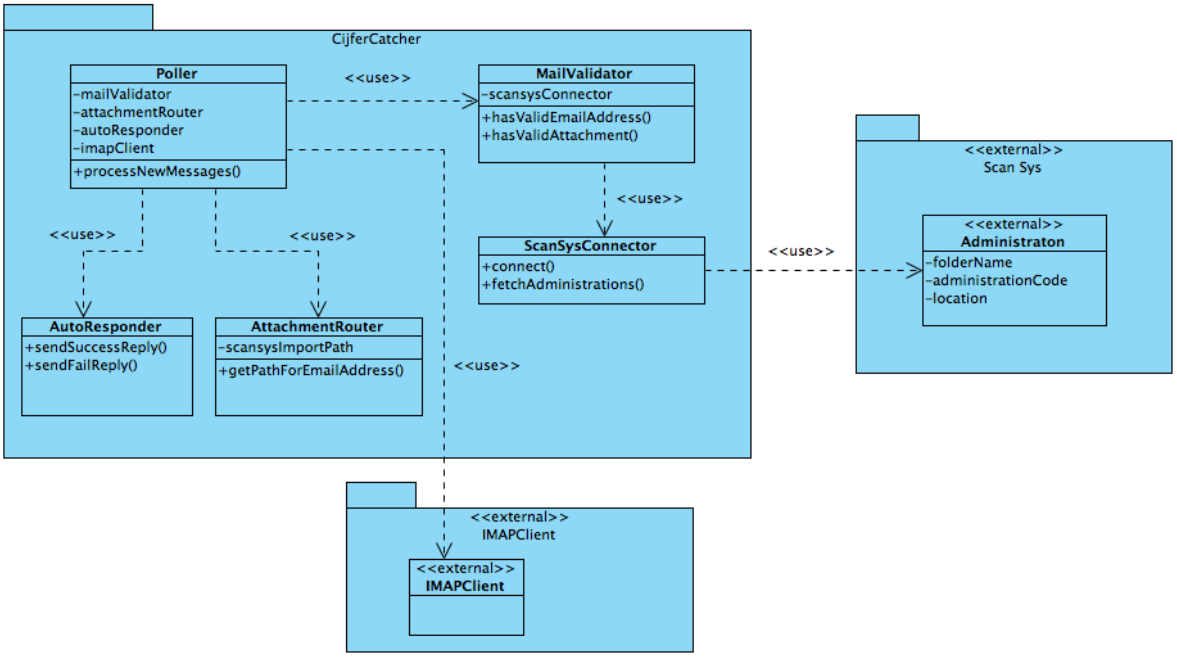
Het onderstaande sequentiediagram heb ik gemaakt om te laten zien hoe de interactie tussen de verschillende systeemdelen zal plaats vinden.



Figuur 12 Sequentiediagram CijferCatcher

KLASSENDIAGRAM

Om de structuur van de CijferCatcher vast te leggen heb ik een klassendiagram gemaakt waarin de belangrijkste klassen worden beschreven.



De verantwoordelijkheden van de klassen zijn:

Poller: Deze klasse is verantwoordelijk voor het verwerken van nieuwe e-mailberichten die zijn opgestuurd. Het verwerken zal elke 5 minuten gebeuren.

AutoResponder: Deze klasse is verantwoordelijk voor het versturen van een antwoord naar de klant (Bevestigingsmail). Dit kunnen twee antwoorden zijn: Een 'Gelukt!'-antwoord voor als de opgestuurde factuurscan goed is aangekomen, of een 'Mislukt!'-antwoord als er iets niet goed is gegaan.

AttachmentRouter: Deze klasse is verantwoordelijk voor het opbouwen van de uiteindelijke map waar de factuurscan moet worden geplaatst. Bijvoorbeeld C:\ScanSys\Import\Klant18\Inkoop. De klasse moet er dus voor zorgen dat de bijlagen in de specifieke in- of verkoopmap komt van de klant.

MailValidator: Deze klasse is verantwoordelijk voor het controleren of de mail die is opgestuurd klopt (controle op geldig AAN-adres en bijlage).

ScanSysConnector: Deze klasse is verantwoordelijk voor het uitlezen van de lijst met administraties in Scan Sys. Scan Sys heeft een tabel met administraties en bijbehorende mapnamen. De MailValidator heeft deze lijst nodig om te bepalen of een AAN-adres geldig is.

CATCH ALL MAILBOX

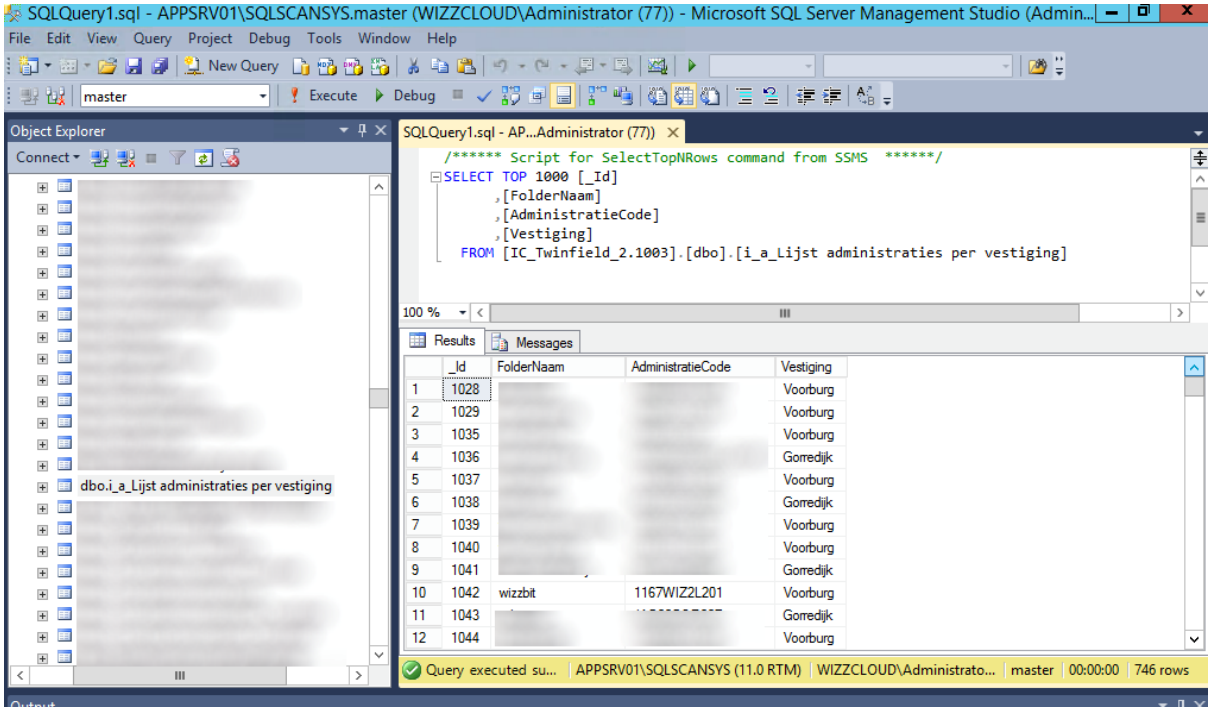
Om ervoor te zorgen dat klanten facturen kunnen sturen naar adressen zoals wizzbit-inkoop@cijfermeesterscan.nl of wizzbit-verkoop@cijfermeesterscan.nl moet er gebruik gemaakt worden van een 'catch all mailbox' dit is een mailbox waar alles op binnenkomt ongeacht wat er voor het apenstaartje in het e-mailadres. Een van de diensten die Wizzbit levert is mail, een catch all mailbox inrichten is daarom geen probleem. Zodra de mail is afgeleverd op de mailserver moet hier uiteraard nog wel wat mee gebeuren. De CijferCatcher zal de mail moeten verwerken en zorgen dat deze op de juiste plek komt.

SCAN SYS

Scan Sys is de applicatie die CijferMeester gebruikt voor het verwerken van factuurscans. In Scan Sys is er door CijferMeester een database opgebouwd met mappen en bijbehorende administratie codes. Door factuurscans in een bepaalde map te slepen (bijvoorbeeld \wizzbit\inkoop\) wordt deze door Scan Sys opgepakt in de juiste administratie.

Scan Sys slaat de map namen en bijbehorende administratie codes op in een simpele SQL tabel die wij zouden kunnen uitlezen om te gebruiken als validatie voor binnenkomende e-mails.

In de onderstaande afbeelding is te zien hoe de map namen zijn opgeslagen.



Figuur 13 Tabel met geldige map namen

5.5 CIJFERSLURPER

De CijferSlurper moet ervoor gaan zorgen dat alle facturen van het oude systeem worden overgezet naar het nieuwe systeem. Bij het overzetten is aan het begin een test gedaan met 1 administratie en veel handmatige stappen. Om uiteindelijk alle administraties van alle kantoren over te zetten zou veel tijd kosten op deze manier. Het maken van een generieke tool die automatische alles overzet zal in het begin wat meer tijd kosten maar zal zichzelf daarna terugverdienen doordat alle kantoren in het vervolg sneller kunnen worden gemigreerd.

Het aanmaken van ieder kantoor in het nieuwe systeem doet CijferMeester zelf en de kantoren maken hun administraties, medewerkers en klanten aan. De CijferSlurper zorgt voor het overzetten van alle facturen.

In het oude systeem heeft ieder kantoor een eigen omgeving met daarin hun eigen administraties en facturen.

REQUIREMENTS

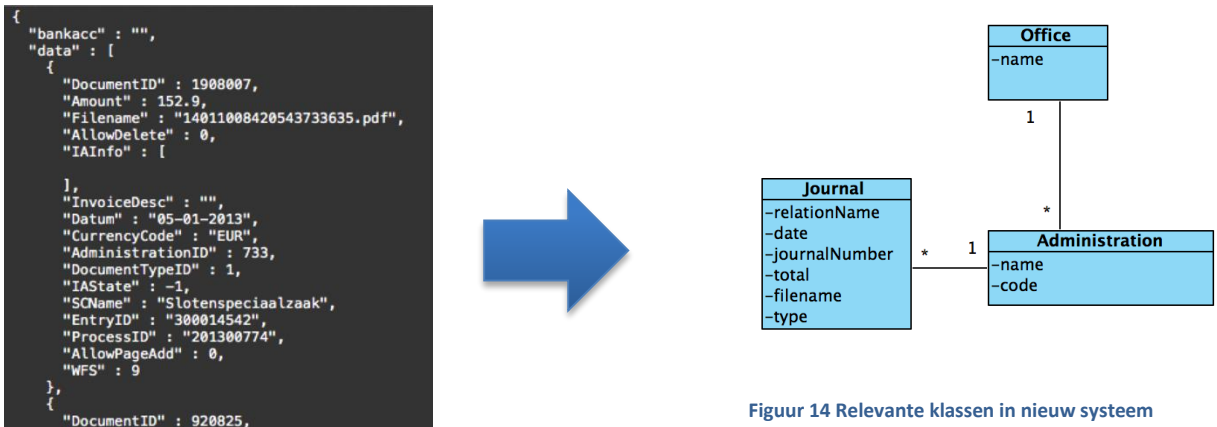
- Alle factuurscans moeten worden overgezet van het oude systeem naar de nieuwe CijferPortal
- Het overzetten moet voor alle kantoren herbruikbaar zijn

ONTWERP

Het oude systeem biedt geen exportmogelijkheden dus de facturen kunnen in de huidige situatie alleen 1 voor 1 handmatig gedownload worden. Het doel van de CijferSlurper is om dit proces te automatiseren.

Omdat de CijferSlurper voor een groot deel acties zal automatiseren die normaal gesproken via een webbrowser worden uitgevoerd leek het mij verstandig om de CijferSlurper te maken met node.js, dit is een softwareplatform waarbij applicaties geschreven worden in javascript. Dit heeft als voordeel dat dezelfde taal gebruikt wordt die nu ook gebruikt wordt binnen de browser in het oude systeem, dit maakt het makkelijk om dingen uit te lezen.

Er is geen ontwerp van het oude systeem beschikbaar.



Figuur 14 Relevante klassen in nieuw systeem

Figuur 13 Gegevens uit het oude systeem

Om de facturen goed over te zetten heb ik een data dictionairy gemaakt waarin helder wordt hoe de gegevens in het oude systeem staan en hoe deze in de CijferPortal moeten komen.

Facturen				
Oude systeem	**Nieuw systeem**	**Betekenis**	**Voorbeelddata**	**Opmerking**
SCName	relationName	Relatie	Wizzbit Paessler	
Datum	date	Factuurdatum	05-01-2013 21-09-2011	
ProcessID	journalNumber	Boekstuknummer	201300774 201100261	
Amount	total	Totaalbedrag	152.9 640.22	
Filename	filename	Bestandsnaam	14011008420543733635.pdf 120928153124129202171.pdf	
DocumentTypeId	type	Inkoop of verkoop factuur	1 21	1 = Inkoop 21 = Verkoop

Administration		
Oude systeem	**Nieuw systeem**	**Betekenis**
Id	id	Unieke administratie code
	Code	Twinfield code van administratie
Name	Naam	Naam van administratie

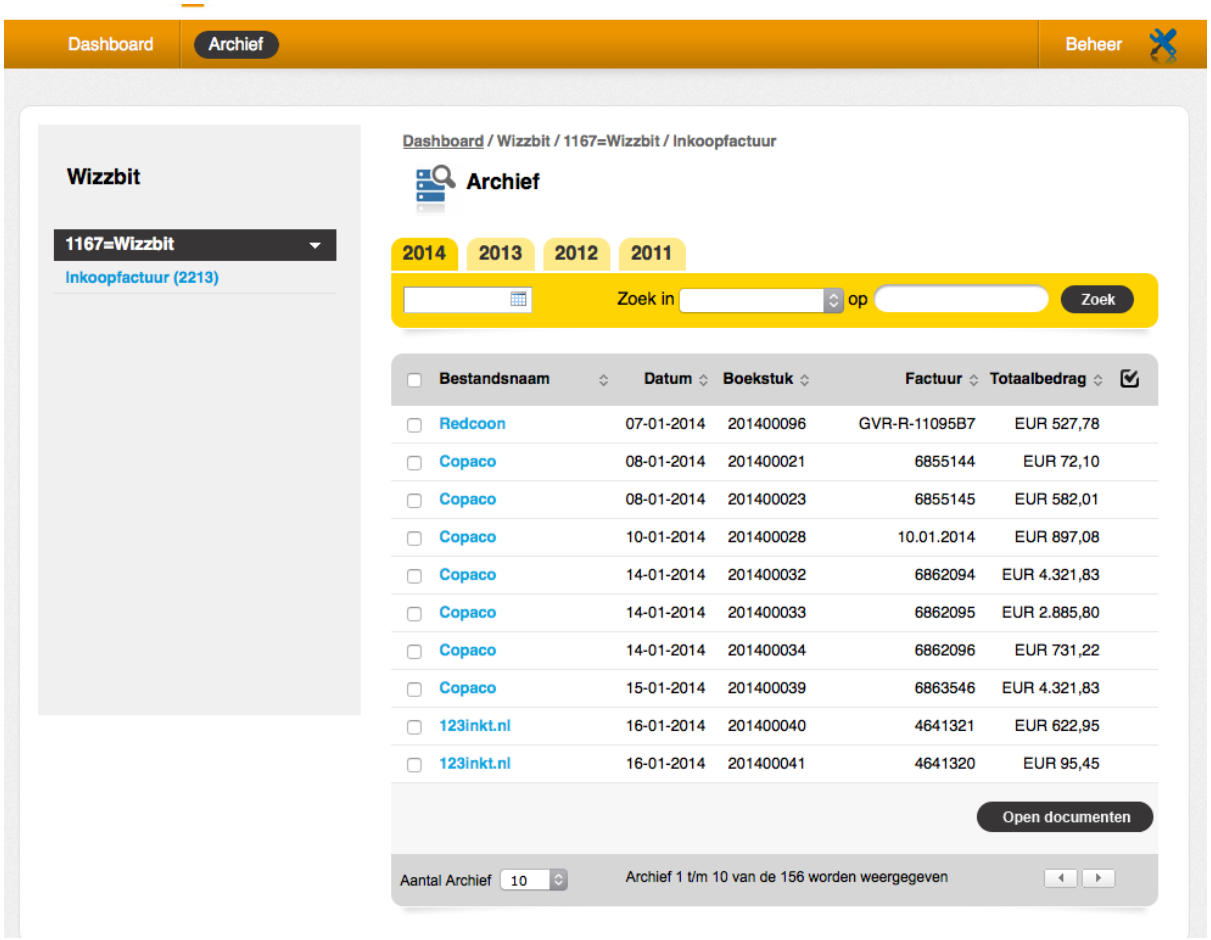
Office		
Oude systeem	**Nieuw systeem**	**Betekenis**
	Naam	Naam van kantoor

Data volledig meenemen naar nieuw systeem

Niet alle data (die wij tot onze beschikking hebben) van het oude systeem heeft een plek gekregen in het nieuwe systeem, sommige gegevens waren alleen relevant voor het oude systeem of niet nodig in het nieuwe systeem. Maar om er zeker van te zijn dat we niet onnodig data verliezen heb ik besloten de hele oude JSON structuur van een factuur uit het oude systeem ook nog eens in 1 extra veld opgeslagen in het nieuwe systeem. Zo kan er altijd achteraf nog iets mee gedaan worden mocht dit nodig zijn.

WERKZAAMHEDEN

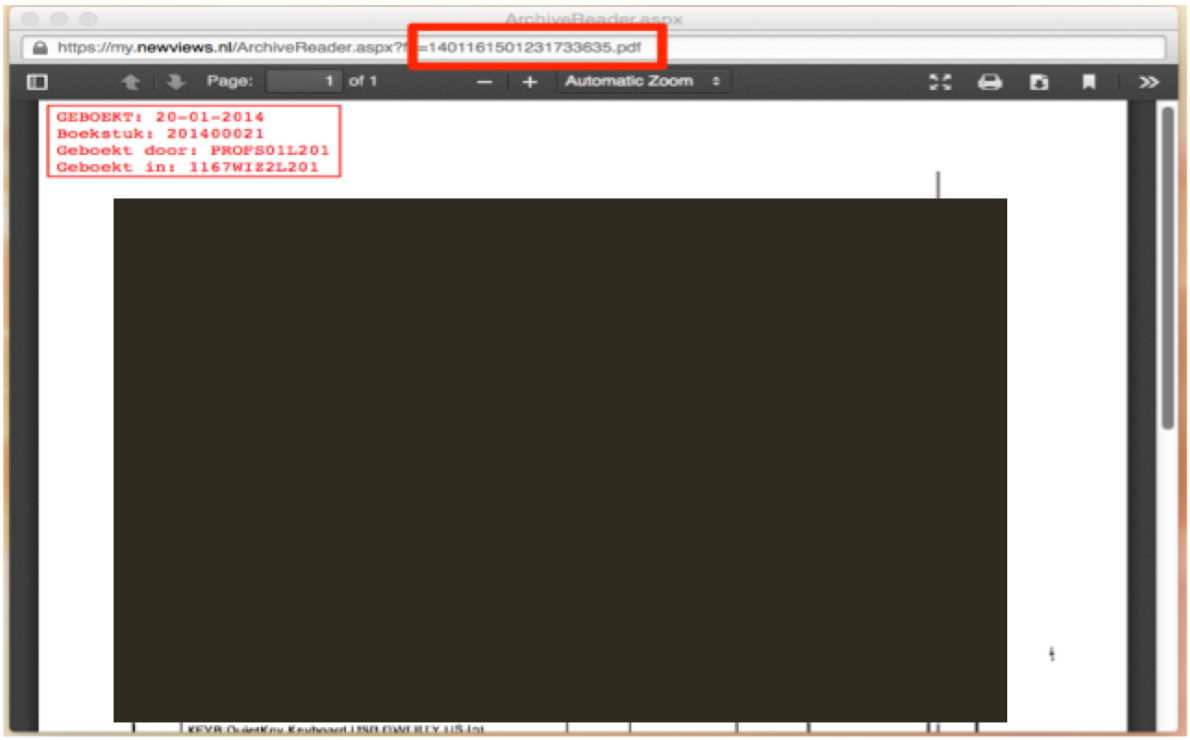
In het oude systeem worden de oude facturen bewaard. Alle facturen moeten worden overgezet naar het nieuwe systeem. Iedere factuur heeft een aantal gegevens met daarnaast ook een scan waarop de originele factuur te zien is.



Figuur 14 De facturen in het oude systeem

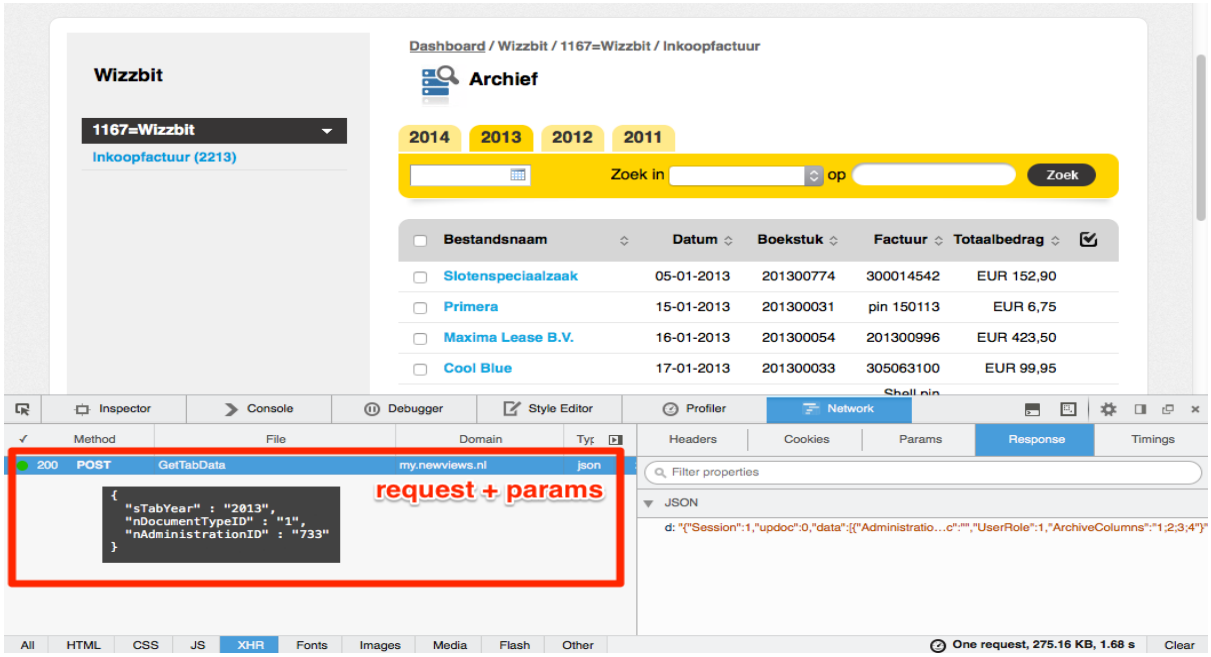
Ik heb onderzocht of het mogelijk is om de gegevens te scrapen, dit is een techniek die gebruikt wordt om gegevens van websites uit te lezen, dit is nodig omdat het oude systeem geen export mogelijkheden heeft, ook is er geen documentatie beschikbaar.

Door factuurscans te openen is gebleken dat iedere factuurscan een unieke bestandsnaam heeft.



Figuur 15 Bestandsnaam voor iedere factuurscan

Door de HTTP requests te analyseren is vastgesteld dat er voor ieder jaartabje die wordt aangeklikt een AJAX-request wordt gedaan, de server stuurt dan de gegevens van dat jaar terug.



Figuur 16 AJAX-request voor opvragen gegevens van 2013

De gegevens die de server terug stuurt bevat alle informatie die nodig is, ook de bestandsnaam van de factuurscan zit er bij, het dus mogelijk om een tool te maken die dit automatisch uitleest en over zet naar het de CijferPortal.

```
{
  "bankacc" : "",
  "data" : [
    {
      "DocumentID" : 1908007,
      "Amount" : 152.9,
      "Filename" : "14011008420543733635.pdf",
      "AllowDelete" : 0,
      "IAInfo" : [

      ],
      "InvoiceDesc" : "",
      "Datum" : "05-01-2013",
      "CurrencyCode" : "EUR",
      "AdministrationID" : 733,
      "DocumentTypeID" : 1,
      "IAState" : -1,
      "SCName" : "Slotenspeciaalzaak",
      "EntryID" : "300014542",
      "ProcessID" : "201300774",
      "AllowPageAdd" : 0,
      "WFS" : 9
    },
    {
      "DocumentID" : 920825,
      "Amount" : 6.75,
      "Filename" : "1301222115297733635.pdf",
      "AllowDelete" : 0,
      "IAInfo" : [

      ]
```

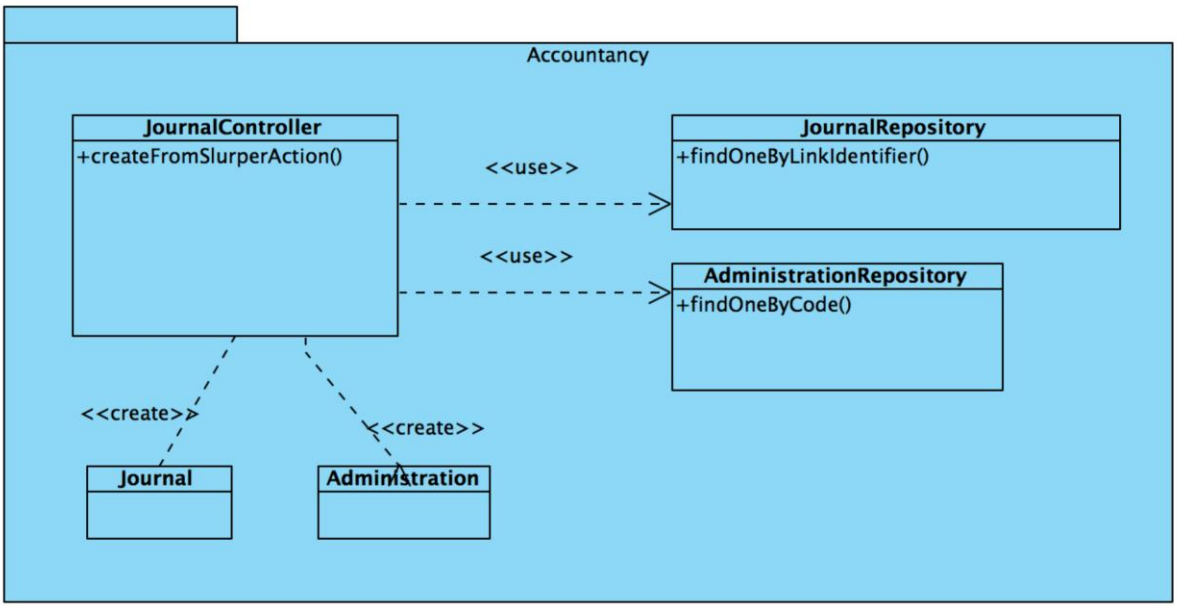

Figuur 17 Gegevens die de server terug stuurt in JSON formaat

AANPASSING CIJFERPORTAL

Het factuurarchief is uitgebreid met een 1 functie die de CijferSlurper kan aanroepen om een factuur toe te voegen:

```php
/**
 * Factuur aanmaken vanuit de de CijferSlurper
 *
 * @param Request $request
 * @return Response
 */
public function createFromSlurperAction(Request $request)
{
    $em = $this->container->get('doctrine.orm.default_entity_manager');

    // Checken of die al bestaat (op basis van linkIdentifier (unieke bestandsnaam)
    if (($request->get('filename')) && ($em->getRepository('Journal')->findOneByLinkIdentifier($request->get('filename')))) {
        return new Response($request->get('filename') . ' already exists, skipped.');
    }


    // Als de administratie nog niet is aangemaakt dan aanmaken
    $administrationEntity = $em->getRepository('Administration')->findOneByCode($request->get('administrationCode'));

    if ( ! $administrationEntity) {
        $administrationEntity = new Administration();
        $administrationEntity->setCode($request->get('administrationCode'));

        $office = $em->getRepository('Office')->findOneById($request->get('officeId'));
        $administrationEntity->setOffice($office);

        $em->persist($administrationEntity);
    }


    // Factuur aanmaken
    $journal = new Journal($administrationEntity);
    $journal->setType($request->get('type'));
    $journal->setJournalNumber($request->get('journalNumber'));
    $journal->setInvoice($request->get('invoice'));
    $journal->setDate(new \DateTime($request->get('date')));
    $journal->setRelationName($request->get('relationName'));
    $journal->setTotal($request->get('total'));
    $journal->setJournalFilename($request->get('filename'));
    $journal->setLinkIdentifier($request->get('filename'));
    $journal->setAttachment($request->files->get('file'));
    $journal->setSource($request->get('source'));

    $em->persist($journal);
    $em->flush();

    return new Response($request->get('filename') . ' added.');
}
```

Figuur 18 Functie voor aanmaken factuur vanuit de CijferSlurper



Figuur 19 Nieuw functie in het klassendiagram

MAPPING ADMINISTRATIE CODES

Iedere administratie heeft in Newviews een koppeling met Twinfield aan de hand van een Twinfield-administratiecode. Bij het uitlezen van het oude systeem was het niet mogelijk om deze code uit te lezen, dus moet er een lijst worden opgesteld van welke administraties bij welke Twinfield-administratie code hoort.

	A	B
1	**Oude code**	**Nieuwe code**
2	1241-Profsrijswijk	PRORI01L201
3	1240-Profsvoorburg	PROVO01L201
4	1170-Xpozer	1170XPO1L201
5	1167=Wizzbit	1167WIZ2L201
6	1176-Installatiebedrijf Ch. de Boers	1176BOE1L201
7	1170-Xpozer Europe B.V.	1170XPEUL201
8	1170-LeCle B.V.	1170CLE1L201
9	1170-Ivanqwish B.V.	1170IVA1L201
10	1187-dekleinekoffiebranderij	1187KOF1L201
11	1213English Breeze	1213ENG1L201
12	1254-SiSoek	1254SIS1L201
13	1088-The Online Factory	FAMI01L201
14	1289-Tiboring Constructions BV	1289CONL201
15	1290-Rosapax B.V.	1290ROS1L201
16	1291-Tiboring Holding B.V.	1291TIH1L201
17	1197-Prestige Trading BV	1197PRE1L201
18	1301-Brunello	1301BRU1L201
19	1236-Northsea Export B.V.	1236NORL201
20	1310-Siesoton	1310SIE1L201
21	1302-Profs XXL V.O.F.	1302PRO1L201

Figuur 20 Mapping oude administratiecodes, nieuwe administratiecodes

Tijdens maken/testen hebben is er steeds getest met 1 administratie (die van Wizzbit). Uiteindelijk moesten alle administraties worden overgezet. Toen dit daadwerkelijk gedaan was bleek dat er nog 1 ding ontbrak en dat was namelijk de juiste administratie code. De administratie code in het oude systeem is namelijk niet de code die in het nieuwe systeem gebruikt wordt. Wat nodig was is de Twinfield administratie code. Omdat in het begin steeds maar met 1 administratie getest was is dit over het hoofd gezien. Dit betekend dat er voor het overzetten nog 1 handmatige actie overblijft en dat is het maken van een mapping van de administraties in het oude systeem naar de nieuwe.

Om ervoor te zorgen dat het oude systeem niet te veel belast wordt ervoor gezorgd dat er bij het uitlezen van iedere factuur een korte vertraging van 1-3 seconden zit.

TESTEN

Het testen of de CijferSlurper goed alle facturen overzet is gedaan alles goed te loggen en daarna de logs te controleren op foutmeldingen. Na het overzetten van facturen zijn er verschillende steekproeven gedaan door verschillende facturen en administraties te openen. Ook is er een vergelijking gemaakt tussen de aantallen in het oude systeem en de CijferPortal. Bij het testen is gebleken dat het soms kan voorkomen dat de login sessie verloopt tijdens het overzetten. Om ervoor te zorgen dat niet alles opnieuw gedaan hoeft te worden is de CijferSlurper aangepast zodat deze verder kan gaan vanaf de laatste administratie die wel goed gelukt was voordat de login sessie was verlopen.

6 DE CIJFERPORTAL AAN MEERDERE KLANTEN VERKOPEN

Nu de CijferPortal is afgebouwd is de volgende stap het systeem als clouddienst beschikbaar maken. Hiervoor moet er eerst gekeken worden naar welke delen nou eigenlijk beschikbaar gemaakt kunnen worden. Sommige delen van het systeem zijn namelijk heel specifiek voor CijferMeester ontworpen.

Het systeem bestaat nu uit de volgende subsystemen:

- De CijferPortal – Het online archief waar de gescande facturen worden gearchiveerd
- De CijferCatcher – Het systeem wat ervoor zorgt dat de opgestuurde facturen bij de juiste CijferMeester terecht komen
- De CijferSlurper – De tool om documenten van het oude systeem over te zetten naar de CijferPortal

De eerste vraag is wat hiervan kan ook aan andere klanten verkocht worden.

Daarnaast zal er nog een systeem moeten komen waarin er voor iedere klant een eigen omgeving kan worden opgezet.

Ook zal er gekeken moeten worden in hoeverre we met dit systeem binnen onze eigen datacenter blijven of wellicht iets gaan doen met Amazon AWS of Microsoft Azure, de twee grootste cloudinfrastructuur providers op dit moment.

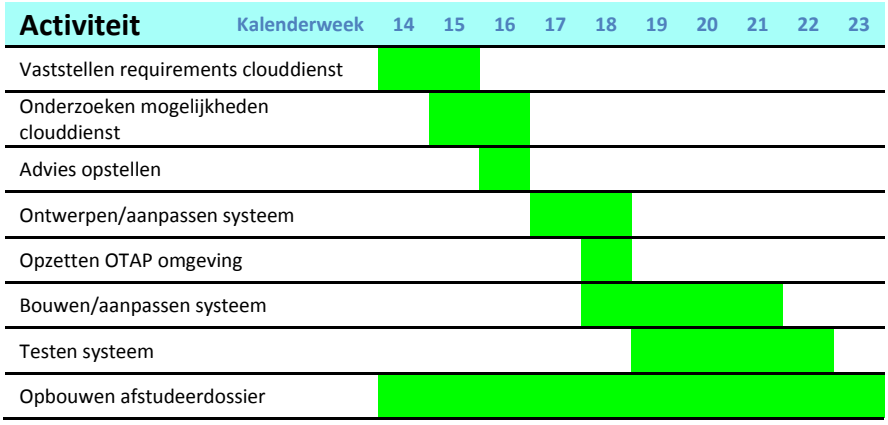
6.1 AANPAK

In hoofdstuk 4 is eerder de aanpak beschreven voor de afstudeeropdracht die eigenlijk uit twee projecten bestaat, het afbouwen van de CijferPortal en daarna het verkoopbaar maken van de CijferPortal. Hieronder wordt de aanpak beschreven zoals deze nu van toepassing is op het tweede project.

- Bijstellen planning
- Vaststellen requirements
- Onderzoek mogelijkheden
- Advies uitbrengen
- SCRUM-planning maken, Backlog opbouwen
- Sprints uitvoeren (evaluatie + planning volgende)
    - Ontwerpen systeem/functies
    - Ontwerpen tests
    - OTAP omgeving opzetten
    - Bouwen systeem/architectuur
    - Testen systeem

6.2 BIJSTELLEN PLANNING

Vanwege redenen die in hoofdstuk 9 worden beschreven was het nodig om de planning bij te stellen. Deze is nu als volgt geworden:



6.3 REQUIREMENTS

Voor het vaststellen van de requirements heb ik een brainstorm ingepland met mijn collega's (waaronder de directeur van Wizzbit, de Product Owner), in deze brainstorm hebben we het gehad over wat we nu hebben en wat we uiteindelijk willen hebben. Hieruit zijn in grote lijnen de volgende wensen en eisen gekomen:

- Het systeem moet aan meerdere klanten verkocht kunnen worden
- Het moet mogelijk zijn om verschillende soorten documenten op te slaan (meer dan alleen facturen)
- Het moet mogelijk zijn om een eigen branding / huisstijl in te stellen

Om de requirements verder uit te werken heb ik een onderzoek gedaan naar wat Wizzbit nu heeft en wat er mogelijk is.

6.4 ONDERZOEK MOGELIJKHEDEN

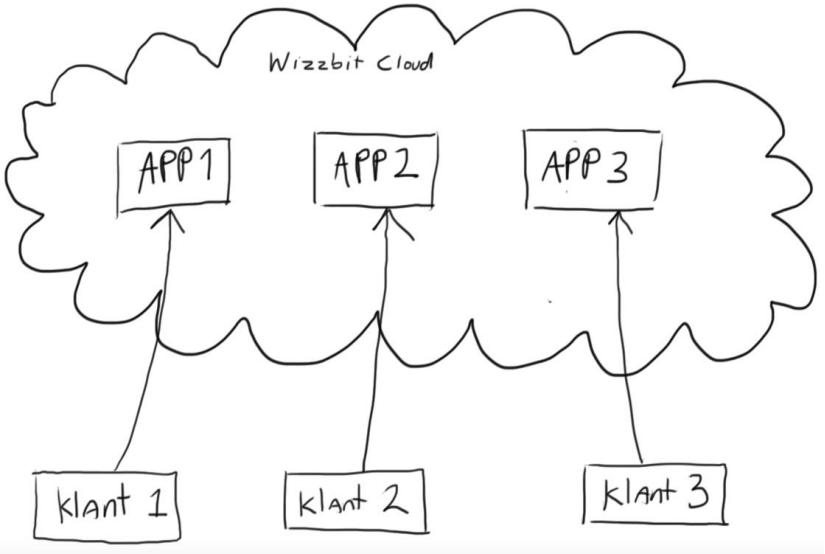
Met de requirements in gedachten heb ik onderzocht wat de mogelijkheden voor ons zijn. Hierbij heb ik gekeken naar andere diensten die ook via de cloud worden aangeboden en bekeken of dat ook bij ons past. Zo heeft bijvoorbeeld CampaignMonitor een handig systeem waarbij je na het aanmelden een eigen omgeving krijgt op een eigen subdomein en deze omgeving wordt binnen een paar tellen automatisch opgezet. Eenmaal op de omgeving ingelogd kan je als reseller je eigen klantomgevingen maken die ook binnen enkele klikken te gebruiken zijn voor de jouw klanten.

Het onderzoek heeft de volgende hoofdvragen:

- Wat hebben we nu?
- Wat is er mogelijk?

WAT HEBBEN WE NU?

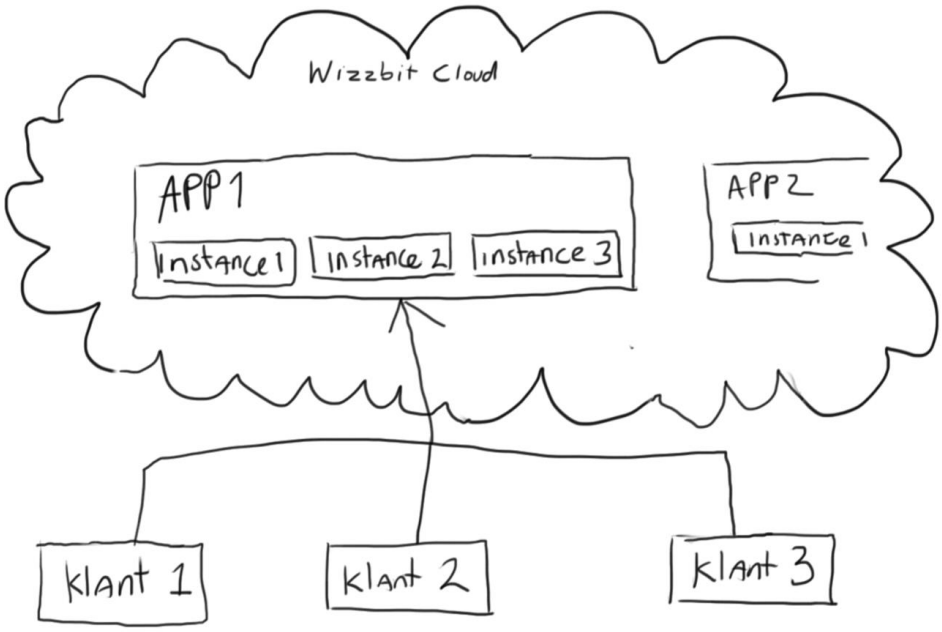
Bij het onderzoek heb ik even een stapje teruggedaan en heb ik op conceptueel niveau gekeken naar wat we aan het doen zijn. Aanleiding hiervan was de vele vergelijkingen van SaaS (Software-as-a-Service) met ASP (Application Service Provider) die ik tegen kwam bij het onderzoeken naar cloud mogelijkheden. Ik heb niet eerder stilgestaan bij onder wat voor model of begrip het valt wat Wizzbit nu doet. Hierbij heb ik geconcludeerd dat we nu webapplicaties maken en verkopen onder het single-tenant ASP-model waarbij we steeds één applicatie op maat maken voor één klant. Zo is de CijferPortal nu op maat gemaakt voor één klant en daarnaast hebben we ook een offertesysteem voor één andere klant en een ledenadministratie voor één andere klant, een genderportal voor één andere klant, een CRM voor nog één andere klant, allemaal ASP en single-tenant. Bij het verkoopbaar maken van de CijferPortal gaan we deze multi-tenant maken, als dit allemaal naar wens verloopt dan kunnen we deze truc uiteindelijk ook toepassen op onze andere single-tenant systemen (mits daar vraag voor is). Daarnaast is het nog te overwegen of we de applicaties onder SaaS model aanbieden of niet. De grootste verandering zit vooral in het veranderen van single-tenant naar multi-tenant.



Figuur 21 Single-tenant ASP webapplicaties van Wizzbit (huidige situatie)

Conclusie

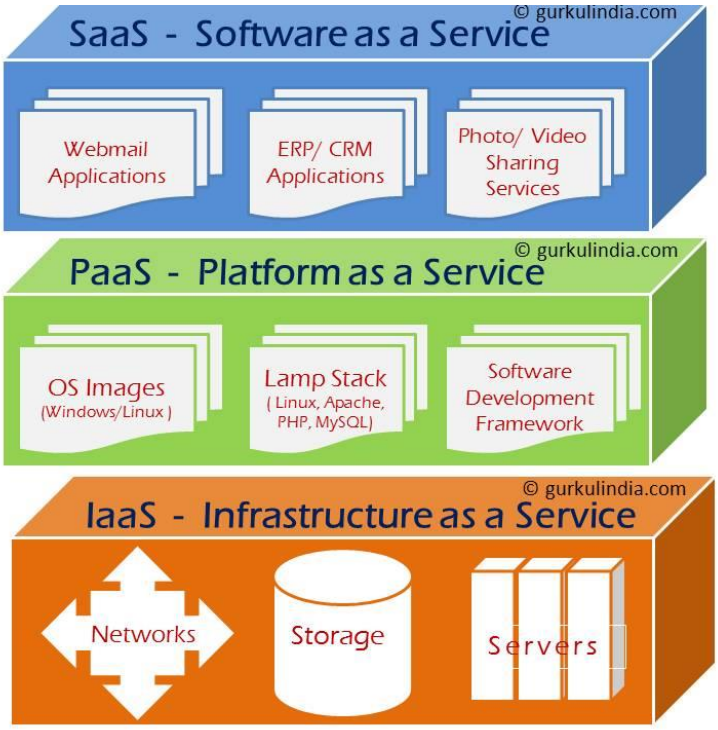
Om de CijferPortal en andere toekomstige systemen aan meerdere klanten te kunnen verkopen moeten deze systemen met multi-tenancy in gedachten ontwikkeld worden. Deze conclusie leidt tot een aantal nieuwe onderzoekspunten die verder op in dit onderdeel worden besproken.



Figuur 22 Multi-tenant architectuur (toekomst)

WAT IS ER MOGELIJK?

Op het technisch vlak zijn er een aantal dingen mogelijk. Hierbij heb ik de verschillende lagen van cloud computing erbij gepakt:



Figuur 23 Lagen cloud computing

Op dit moment host Wizzbit alles zelf, we hebben onze eigen hardware waar alles op draait op dit moment in onze eigen Wizzbit cloud. We zien dat alles steeds meer naar de grotere cloud providers zoals Amazon of Microsoft toe gaat, waarbij het mogelijk is om de onderste lagen uit de cloud computing lagen (IaaS en PaaS) als dienst af te nemen.

Wizzbit wil voorbereid zijn op een eventuele migratie naar Amazon AWS of Microsoft Azure. Dit valt op dit moment wel buiten de scope van dit project maar is wel iets waar alvast rekening mee gehouden moet worden. Om zelf ook goed geïnformeerd te zijn over de mogelijkheden in de cloud heb ik deelgenoomen aan een Microsoft Azure DevCamp en heb ik daarnaast ook een aantal trainingsvideo's bekeken van Amazon AWS zodat ik op de hoogte ben van wat er mogelijk is bij de twee grootste cloud service providers en weet waar ik rekening mee moet houden bij het ontwerpen van het systeem.

WAAR MOET REKENING MEE GEHOUDEN WORDEN BIJ HET ONTWIKKELEN VAN EEN MULTI-TENANT SYSTEEM?

Als hulpmiddel heb ik het volgende document gebruikt: Developing Multi-tenant Applications for the Cloud, 3rd Edition[1]. Alhoewel dit specifiek gericht is voor gebruik met Microsoft Azure bevat het veel belangrijke aandachtspunten die voor Wizzbit relevant zijn. Vooral het single- en multi-tenant deel wordt aandachtig besproken. De punten worden beschreven vanuit twee perspectieven, die van de tenant (klant) en die van de provider (Wizzbit). Ik heb uit het document de punten opgenomen die belangrijk zijn voor ons om rekening mee te houden, deze zijn als volgt:

The Tenant's Perspective (Uitgangspunt van de klant)

- **Isolatie.** Klanten moeten van elkaar geïsoleerd zijn, de ene klant mag niet bij de data van een andere klant kunnen komen.
- **Beschikbaarheid.** Klanten verwachten dat het systeem constant beschikbaar is, activiteiten van andere klanten mogen geen invloed hebben op de beschikbaarheid het systeem.
- **Schaalbaarheid.** Het moet voor een individuele klant mogelijk zijn om te schalen onafhankelijk van de andere klanten, de aanwezigheid of activiteiten van andere klanten mogen geen invloed hebben op de performance van het systeem.
- **Kosten.** Klanten verwachten dat de kosten voor het gebruik van een multi-tenant applicatie lager zal zijn dan het draaien van een dedicated, single-tenant applicatie. Klanten moeten op de hoogte zijn van het kostenmodel voor het gebruik van de applicatie.
- **Aanpasbaarheid.** Een individuele klant wil misschien eigen logo's, kleuren of misschien zelfs eigen functies kunnen toevoegen.
- **Naleving van wetgeving.** Een klant moet wellicht er van verzekerd zijn dat het systeem voldoet aan de wetgeving voor bijvoorbeeld het opslaan van persoonlijke gegevens of het verwerken van data.

The Provider's Perspective (Uitgangspunt van Wizzbit)

- **Voldoen aan de requirements en doelen van de klanten.** Wizzbit moet ervoor zorgen dat het systeem voldoet aan de verwachtingen van de klanten. Dit kan eventueel in een formele SLA opgesteld worden.
- **Winstgevendheid.** Omzet van het systeem moet voldoende zijn voor zowel de kosten voor in stand houden van het systeem en het terugverdienen van de geïnvesteerde kosten voor het bouwen van het systeem.
- **Facturatie.** Wizzbit moet klanten kunnen factureren. Dit kan een vast bedrag per maand zijn maar dit zou ook een bedrag op basis van gebruik van het systeem kunnen zijn, dit betekent dat het mogelijk moet zijn om te monitoren hoe of hoeveel een klant gebruik maakt van het systeem.

[1] Developing Multi-tenant Applications for the Cloud, 3rd Edition

- **Meerdere service levels.** Wizzbit kan de wens hebben om verschillende service levels aan te bieden. Bijvoorbeeld een standard of premium abonnement. Deze verschillende abonnementen kunnen andere functies, limitaties of andere eigenschappen hebben.
- **Provisionen.** Wizzbit moet nieuwe klanten kunnen provisionen in het systeem. Bij een klein aantal klanten kan dit een handmatig proces zijn. Bij grote aantallen is het meestal nodig om dit te automatiseren met zelf-service provisionen.
- **Onderhoudbaarheid.** Wizzbit moet het systeem kunnen upgraden terwijl meerdere klanten er gebruik van maken.
- **Monitoring.** Wizzbit moet de mogelijkheid hebben om het systeem te monitoren om problemen te identificeren en troubleshooten.
- **Automatisering.** Naast automatisch provisionen kan het ook verstandig zijn om andere taken te automatiseren, zoals bijvoorbeeld het dynamisch schalen van de beschikbare resources wanneer dit nodig is.

ISOLATIE

Om er voor te zorgen dat de klantomgevingen van elkaar geïsoleerd worden heb ik een aantal mogelijkheden bekeken.

Isolatie door middel van programmeercode
Voor dingen die te maken hebben met de veiligheid en isolatie van het systeem kan er beter gebruik gemaakt worden van bewezen modules, waar door veel ogen naar gekeken is. Zelf schrijven geeft een groter risico op lekken. Wizzbit maakt nu gebruik van een bewezen LoginBundle, dus het inlogdeel is veilig, maar de isolatie na het inloggen zou dan zelf geschreven moeten worden. Volledig zelf schrijven is een risico, een fout of lek in code zou kunnen leiden tot lekken tussen 2 omgevingen. Daarom lijkt het mij niet verstandig om op codeniveau de omgevingen te scheiden.

Isolatie op bestandssysteem en database niveau
Dit is hoe Wizzbit op dit moment voor onze verschillende single-tenant klanten de applicaties isoleert. Hierbij heeft iedere klant een eigen database heeft en een eigen webroot op het bestandssysteem.

Isolatie op server niveau
Dit is nog veiliger maar waarschijnlijk overbodig, als een van de klanten groeit zouden we deze eventueel op een eigen server kunnen zetten, maar dit zou dan meer zijn voor de performance en dan voor de veiligheid.

Isolatie van de back-up
Back-up en herstel wordt verzorgd door Wizzbit, de back-up is alleen toegankelijk voor Wizzbit en niet voor de klanten, er is hier dus geen noodzaak om een scheiding te maken per back-up, wel moet het mogelijk om een back-up van de ene klant los te kunnen herstellen van de andere klant. De scheiding moet er dus wel zijn maar niet op toegangsniveau.

Conclusie
Het is verstandig om dezelfde isolatie methode te gebruiken die nu al in gebruik is voor de bestaande applicaties. Dit brengt de minste onbekende risico's en is relatief eenvoudig toe te passen binnen de huidige infrastructuur van Wizzbit.

SCHAALBAARHEID

Waar vooral rekening mee gehouden moet worden in dit systeem is de opslagcapaciteit. Dit zal waarschijnlijk de eerste bottleneck worden omdat het systeem vooral een opslag is van documenten. Er zijn geen CPU of geheugen intensieve taken die uitgevoerd worden.

Conclusie
Met een File Abstraction Layer (FAL) zou eventueel de opslag flexibel per klant kunnen worden aangepast naar een andere schijf, koppeling of cloudopslag.

FACTURATIE

Om te kunnen factureren op basis van gebruik zijn de volgende manieren mogelijk:

- Meten van aantal facturen (documenten).
- Meten van het aantal actieve gebruikers
- Meten van het aantal downloads

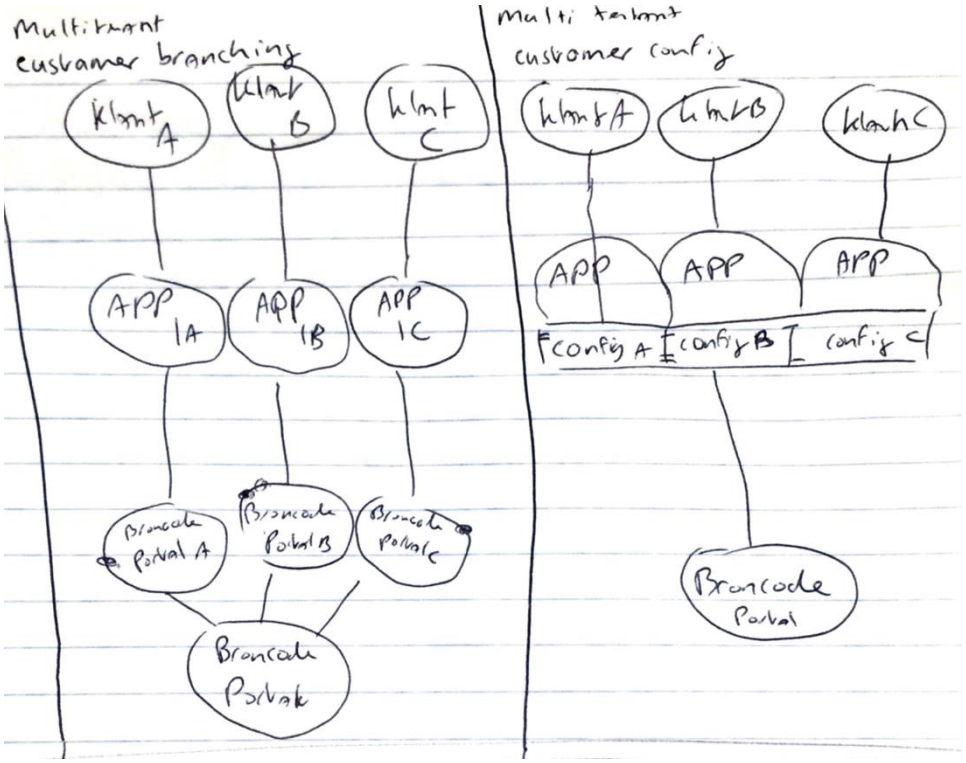
Conclusie
Voor iedere manier zijn voor- en nadelen te bedenken en voor toekomstige modules zal er waarschijnlijk iets totaal anders gemeten moeten worden. Wat van belang is dat er gemeten kan worden.

ONDERHOUDBAARHEID

Om er voor te zorgen dat het systeem goed te onderhouden is moet er ook nagedacht worden over hoe het versiebeheer eruit gaat zien. Op dit moment hebben we één repository per applicatie met branches voor verschillende doeleinden.

Figuur 24 Eén repository per klant/app

Een mogelijk opzet is om per klant een eigen branch te gebruiken. Ik heb gezocht naar wat nog meer mogelijk is en ben erachter gekomen dat het eigenlijk niet verstandig is om branches per klant te hebben omdat dit moeilijk te onderhouden is. Een goede opzet is om één gedeelde broncode te hebben voor alle klanten waar alle features in zitten. Doormiddel van configuratie per klant kan het systeem nog steeds per klant op maat gemaakt worden. Dit voorkomt conflicten en complexiteit veroorzaakt door branches die te veel uit elkaar lopen.



Figuur 25 Broncode: branch per klant of configuratie per klant

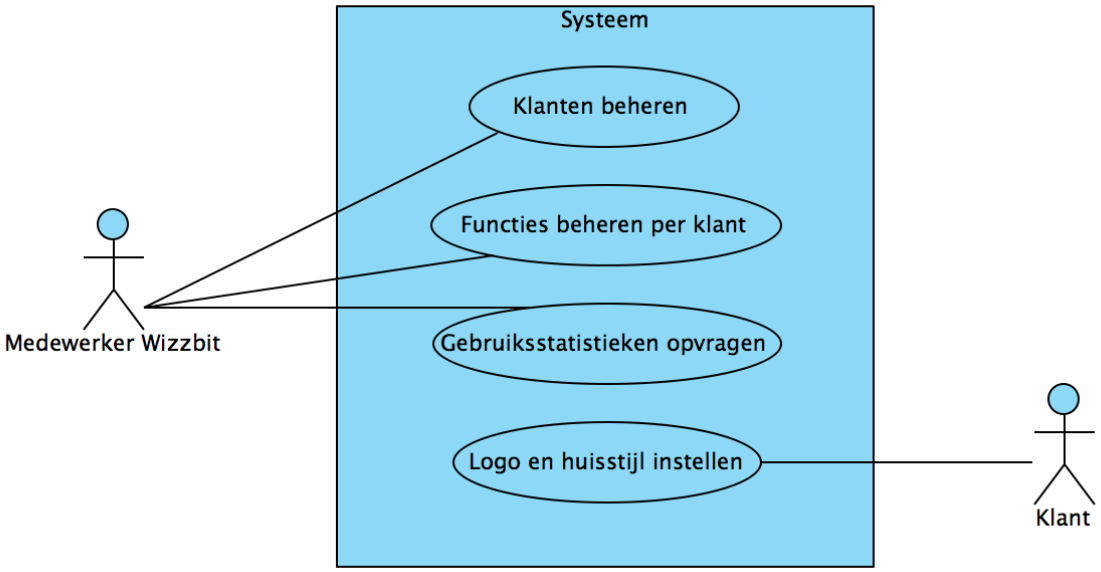
Conclusie

Het systeem moet zodanig generiek en configureerbaar gemaakt worden zodat deze in één code repository kan worden onderhouden.

6.5 REQUIREMENTS UITGEWERKT

Met het globale requirements en het onderzoek heb ik de requirements verder kunnen uitwerken.

USE CASE DIAGRAM



Figuur 26 Use case diagram

BUSINESS REQUIREMENTS

Code	Requirement	Prioriteit	Bron
B1	De software moet door meerdere klanten gebruikt kunnen worden	Must	Brainstorm
B2	De software moet efficiënt onderhouden kunnen worden	Must	Brainstorm
B3	De software moet uitgebreid kunnen worden met nieuwe functies	Must	Brainstorm
B4	Het moet inzichtelijk zijn hoe de software gebruikt wordt	Should	Brainstorm

SYSTEEMREQUIREMENTS

Functionele requirements

Code	Requirement	Prioriteit	Traceerbaarheid
S1	De basis van het systeem waar iedere klant op inlogt moet generiek zijn	Must	B2
S2	Het moet mogelijk zijn om verschillende soorten documenten op te slaan	Must	B1, B2
S3	Het moet mogelijk zijn een eigen branding of huisstijl in te stellen	Should	B1
S5	Voor Wizzbit moet het mogelijk zijn om klanten te beheren	Must	B1
S6	Voor Wizzbit moet het mogelijk te zien welke klanten welke functies hoeveel gebruiken	Should	B4
S7	Het moet mogelijk zijn om module aan of uit te zetten per klant	Must	B1, B2, B3
S8	Het systeem moet kunnen meten hoe vaak of hoeveel een functie in het systeem wordt gebruikt	Should	B4

Niet-functionele requirements

Code	Requirement	Prioriteit	Traceerbaarheid
NF1	Het moet mogelijk zijn om binnen een uur een klantomgeving op te zetten	Should	
NF2	De klantomgevingen moeten veilig van elkaar gescheiden zijn	Must	B1

GEBRUIKERSREQUIREMENTS

Code	Requirement	Prioriteit	Traceerbaarheid
G1	Medewerkers van Wizzbit moeten klanten kunnen aanmaken/beheren	Must	B1, S5
G2	Medewerkers van Wizzbit moeten functies voor een klant kunnen aan of uitzetten	Must	B1, S5, B3
G3	Klanten moeten een logo en kleurenstijl kunnen instellen voor hun omgeving	Should	S3
G4	Een medewerker van Wizzbit moet per klant gebruiksstatistieken kunnen opvragen	Should	B4

USE CASE LIJST

Naam	**Klanten beheren**
ID	UC1
Bron	G1
Samenvatting	Een medewerker van Wizzbit kan een klantomgeving aanmaken
Primaire actoren	Medewerker Wizzbit
Precondities	Medewerker heeft toegang tot het systeem
Primair scenario	Klant toevoegen/wijzigen 1. Medewerker opent klanten lijst 2. Medewerker krijgt lijst met bestaande klanten te zien 3. Medewerker klikt op Klant toevoegen 4. Medewerker vult gegevens in 5. Medewerker klikt op Opslaan
Alternatieve scenario's	Klant verwijderen/deactiveren (Alt 1) 1. Medewerker opent klanten lijst 2. Medewerker krijgt lijst met bestaande klanten te zien 3. Medewerker klikt op Klant deactiveren 4. Medewerker klikt op Opslaan
Postcondities	• Klant is opgeslagen in het systeem • Alt1: Klant en omgeving zijn gedeactiveerd

Naam	**Functies beheren per klant**
ID	UC2
Bron	G2
Samenvatting	Een medewerker van Wizzbit kan functie voor een klant instellen
Primaire actoren	Medewerker Wizzbit
Precondities	Medewerker heeft toegang tot het systeem
Primair scenario	Functies instellen 1. Medewerker opent klanten lijst 2. Medewerker krijgt lijst met bestaande klanten te zien 3. Medewerker klikt op Klant aanpassen 4. Medewerker zet functie aan of uit en vult eventueel opties in 5. Medewerker klikt op Opslaan
Postcondities	• Klant kan ingestelde functies gebruiken

Naam	**Gebruiksstatistieken opvragen**
ID	UC3
Bron	G4
Samenvatting	Een medewerker van Wizzbit kan de gebruiksstatistieken van een klant opvragen
Primaire actoren	Medewerker Wizzbit
Precondities	Medewerker heeft toegang tot het systeem
Primair scenario	1. Medewerker opent klantenlijst 2. Medewerker krijgt lijst met bestaande klanten te zien 3. Medewerker klikt op Gebruiksstatistieken 4. Medewerker krijgt de statistieken te zien

Naam	**Logo en huisstijl instellen**
ID	UC4
Bron	G3
Samenvatting	Een klant stuurt een ingescande factuur als bijlage naar een e-mailadres.
Primaire actoren	Klant
Secundaire actoren	
Precondities	Klant heeft toegang tot de omgeving
Primair scenario	Klant vult eigen kleurcodes en logo in 1. Klant klikt op Systeem -> Instellingen 2. Klant kan kleurcodes invullen en een logo uploaden 3. Klant drukt op opslaan
Alternatieve scenario's	Logo en huisstijl resetten 1. Klant klikt op Systeem -> Instellingen 2. Klant drukt op 'Kleurcodes en logo terug naar standaardinstellingen zetten'.
Postcondities	• De omgeving van de klant krijg andere kleuren en logo. • Alt 1: De omgeving van de klant krijg weer het standaard logo en kleurstelling.

REQUIREMENTS OP PRIORITEIT

S1	De basis van het systeem waar iedere klant op inlogt moet generiek zijn	Must	B2
S2	Het moet mogelijk zijn om verschillende soorten documenten op te slaan	Must	B1, B2
S7	Het moet mogelijk zijn om module aan of uit te zetten per klant	Must	B1, B2, B3
NF2	De klantomgevingen moeten veilig van elkaar gescheiden zijn	Must	B1
S5	Voor Wizzbit moet het mogelijk zijn om klanten te beheren	Must	B1
S6	Voor Wizzbit moet het mogelijk te zien welke klanten welke functies hoeveel gebruiken	Should	B4
S8	Het systeem moet kunnen meten hoe vaak of hoeveel een functie in het systeem wordt gebruikt	Should	B4
NF1	Het moet mogelijk zijn om snel/binnen een uur? Een klantomgeving op te zetten	Should	
S3	Het moet mogelijk zijn een eigen branding of huisstijl in te stellen	Should	B1

De requirements heb ik geprioriteerd met de MoSCoW methode (Swart, 2010).

6.6 ADVIES

Om de wensen van Wizzbit te realiseren zijn er een aantal veranderingen nodig. De veranderingen zijn omvat in een advies. De hoofdzaken uit het advies zijn hieronder te vinden.

1) De huidige CijferPortal moet omgezet worden naar systeem bruikbaar voor meerdere klanten, dit betekent dat het volgende gedaan moet worden:
    a. Omzetten van single- naar multi-tenant systeem
    b. Generiek maken (niet alleen facturen maar ook andere type documenten zoals vrachtdocumenten of krantenknipsels)

2) Er zal een generieke basisomgeving moeten komen waar alle klanten op kunnen inloggen, iedere klant kan toegang hebben tot verschillende modules. De CijferPortal zal een module worden die voor iedere klant aangezet kan worden, bij de één voor facturen en bij de ander voor bijvoorbeeld vrachtdocumenten. De generieke basisomgeving zorgt er voor dat er maar één systeem onderhouden hoeft te worden. In de huidige situatie zijn er bijvoorbeeld meerdere applicaties die allemaal hun eigen loginsysteem hebben, de één met een 'Wachtwoord vergeten?' functie en de ander niet. Toevoegen van functionaliteit moet in de huidige situatie voor ieder systeem apart en opnieuw. Het generiek maken van functionaliteit scheelt tijd en kosten in onderhoud van het systeem.

3) Er moet een beheertool komen voor het beheren van klanten in een multi-tenant applicatie. In deze beheertool kan Wizzbit klantomgevingen aanmaken en functies aan of uit zetten en configureren.

Ook zal de ontwikkeling van een nieuwe applicatie een andere aanpak krijgen, op dit moment gebeurd er in het kort:

Wens van de klant → Bedenk oplossing op maat → Implementeer oplossing voor klant

Hierbij wordt er geen rekening gehouden met potentieel meerdere klanten. In het vervolg zal het proces veranderen naar:

Wens van de klant → Bedenk oplossing op maat -> Maak oplossing generiek → Implementeer oplossing voor alle klanten → Configureer oplossing op maat voor klant A

Omdat er al op korte termijn potentiele klanten zijn is er gekeken naar het minimale wat nodig is om een werkend systeem in de lucht te hebben. Met dit in gedachten is er een onderscheid gemaakt van dingen die op korte termijn gerealiseerd kunnen worden en dingen die later gedaan kunnen worden.

Korte termijn:
Het maken van de beheertool waarmee klantomgevingen kunnen worden opgezet. In de eerste versie is het nog niet nodig om de beheertool een gebruiksvriendelijke interface te geven. Dat betekent wel dat er in het begin voor het aanmaken en beheren van klantomgevingen wat technische kennis en documentatie nodig is.

Het omzetten van de huidige (single-tenant) CijferPortal naar een generieke documenten module geschikt voor multi-tenant gebruik die naast facturen ook gebruikt kan worden voor andere type documenten.

Het systeem zal in eerste versie nog in het datacenter van Wizzbit draaien binnen de bestaande infrastructuur. Wel moet het systeem zodanig gebouwd worden dat deze in een later stadium overgezet kan worden naar een

externe cloud provider zoals Amazon Web Services of Microsoft Azure. Er is op korte termijn nog te weinig kennis en ervaring opgedaan om het systeem gelijk al in een externe cloud te draaien. Wel zijn er een aantal medewerkers van Wizzbit op workshops en trainingen geweest en is er geëxperimenteerd met Microsoft Azure en Amazon AWS. Wizzbit weet dus waar rekening mee gehouden moet worden bij het maken van de eerste versie van het systeem.

Lange termijn:
Het maken van een grafische interface voor de beheertool zodat de klantomgevingen door medewerkers van Wizzbit kunnen worden beheerd zonder technische kennis.

Andere single-tenant systemen die ontwikkeld zijn door Wizzbit multi-tenant en generiek maken zodat ze gebruikt kunnen worden als module die voor iedere klant aan of uit gezet kan worden.

Overzetten naar externe cloud provider. Nadat er meer kennis en ervaring is opgedaan kan het systeem worden overgezet naar de cloud.

6.7 SCRUM PLANNING / BACKLOG OPBOUWEN

De requirements zijn verwerkt in een backlog en vervolgens verdeeld in 3 sprints van 2 weken, hierbij hebben de items uit de backlog een logische samenhang gekregen:

1e Sprint: Generieke klantomgevingen
Opzetten architectuur geïsoleerde klantomgevingen

2e Sprint: Generiek documentenarchief
De CijferPortal beschikbaar maken als generieke documentarchief module

3e Sprint: Beheertool
Klantomgevingen aanmaken en beheren

Bij het indelen van de sprints heb ik aan iedere taak scrumpunten gegeven. Hierbij werd al meteen duidelijk dat de eerste sprint waarschijnlijk niet volledig haalbaar zou zijn. Uit ervaring is ongeveer 25 punten per week haalbaar dus ik heb de indeling aangepast naar een planning die realistisch en haalbaar is

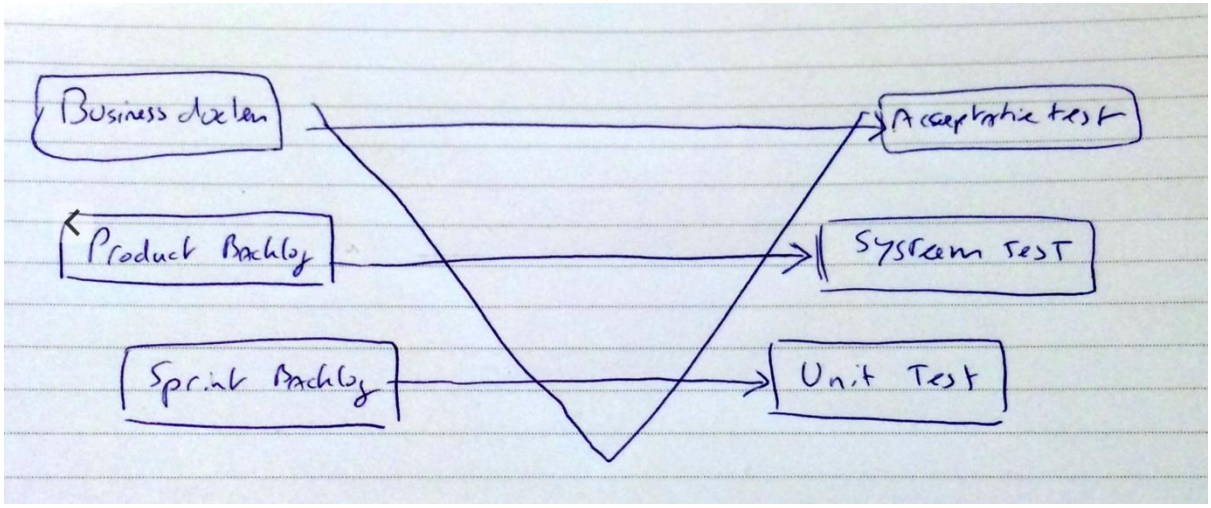
6.8 DE EERSTE SPRINT: GENERIEKE KLANTOMGEVINGEN

De eerste sprint heeft als doel de architectuur op te zetten om geïsoleerde klantomgevingen te creëren.

Item	Requirement ID	Prio	Taak	Scrumpunten
Sprint 1: Generieke klantomgevingen **Doel: Opzetten architectuur geïsoleerde klantomgevingen**				**59**
De basis van het systeem waar iedere klant op inlogt moet generiek zijn.	S1	Must	Testontwerp	13
Het moet mogelijk zijn om module aan of uit te zetten per klant	S7	Must	Ontwerpen architectuur	13
De klantomgevingen moeten veilig van elkaar gescheiden zijn	NF2	Must	Ontwerpen systeem	13
Het systeem moet gebruik maken van open standaarden	NF3	Must	Bouwen systeem	13
			Testen	5
			Documenteren	2

TESTPLAN

Er is begonnen met het maken van een Testplan, hierin is omgeschreven hoe het testtraject eruit zal zien en met wat voor methodes er getest gaat worden. Omdat het oorspronkelijke V-model bedoelt is voor een watervalmethode heb ik een aangepaste V-model beschreven die beter aansluit op het SCRUM-proces.



Figuur 27 Agile V-model

RISICO GEBASEERD TESTEN

Om te bepalen aan welke delen van het systeem de meeste aandacht moet worden besteed bij het testen heb ik gebruik gemaakt van risico gebaseerd testen. Hierbij wordt de scope en diepgang van de functies die getest wordt bepaald door het risico wat Wizzbit loopt als de functies niet goed werken.

Ik heb een lijst opgesteld met de functies en daarna het relatief belang bepaalt door de opdrachtgever punten toe te laten wijzen aan de risicogebieden. Het resultaat hiervan staat in de onderstaande tabel.

Risicogebied	**Relatief belang**
Generieke multi-tenant klantomgeving	9
Documentenmodule voor verschillende soorten documenten	5
Modules aan/uit zetten per klant	3
Klanten beheren	3
Eigen branding of huisstijl instellen	1
Meten functie gebruik	1
Rapportage functiegebruik	1

SYSTEEMTESTS

De testcases voor de systeemtests worden verzameld in een Excel document, dit zijn functionele blackbox tests die uitgevoerd worden om vast te stellen dat het systeem voldoet aan de requirements uit de product backlog.

Testcase	Requirement	Titel	Omschrijving	Pre	Stappen	Test input	Verwacht resultaat
T1	S1	Klantomgevingen	Vaststellen dat het mogelijk is om voor iedere klant een eigen klantomgeving aan te maken		1) Login op TreeMaker console 2) Voer uit: php app/console palmae:create 3) Beantwoord vragen met testdata 4) Herhaal stap 2 en 3 voor iedere alias	alias: palm18 e-mail: fabian@wizzbit.nl wachtwoord: testwachtwoord alias: palm19 e-mail: flachman@wizzbit.nl wachtwoord: testwachtwoord2 alias: palm20 e-mail: testuser@wizzbit.nl wachtwoord: testwachtwoord3	Klantomgeving is aangemaakt en het is mogelijk om in te loggen op palm18.web118.wizzbit.net, palm19.web118.wizzbit.net en palm20.web118.wizzbit.net
T2	NF2	Isolatie	Vaststellen dat de klantomgevingen veilig van elkaar geisoleerd zijn		1) Broncode/architectuur review door expert		1) Expert geeft OK
T3A	S7	Modules aan	Vaststellen dat het mogelijk is per	T1	1) Login op TreeManager console		Bij palm18 is alleen de documenten module geactiveerd

Figuur 28 Systeemtests

De volledige lijst met systeemtests is terug te vinden in de bijlagen.

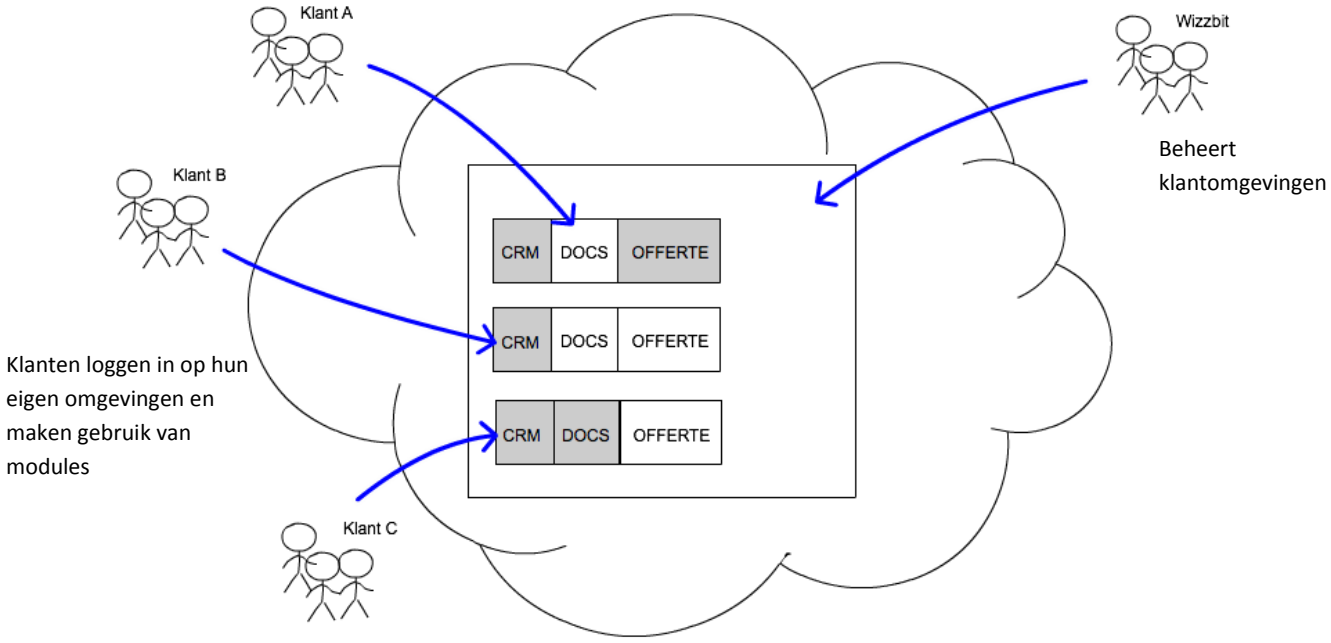
UNIT TESTS

Bij het maken van de unit tests wordt gebruik gemaakt van Test Driven Development (TDD), hierbij worden de volgende stappen uitgevoerd:

1. Test maken
2. (Alle) tests draaien en controleren of de nieuwe test faalt
3. Code schrijven
4. Tests draaien en kijken of deze slagen
5. Code opschonen (Refactoren) en testen (stap 3 en 4 opnieuw)

ONTWERPEN ARCHITECTUUR

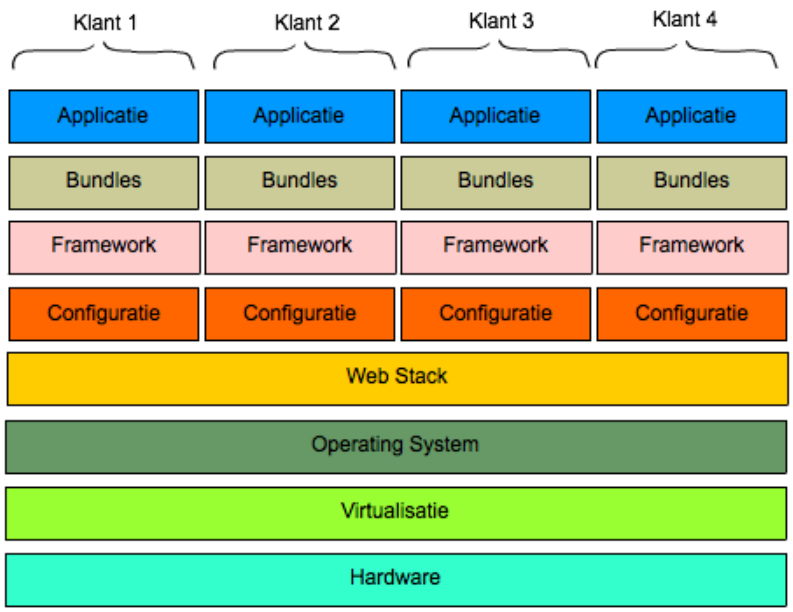
Bij het ontwerpen van de architectuur van het nieuwe systeem heb ik eerst op functioneel niveau een overzicht gemaakt van het te ontwerpen systeem om duidelijk te maken wie wat moet kunnen doen in het systeem.



Figuur 29 Iedere klant heeft zijn eigen omgeving met verschillende modules

Huidige situatie

Vervolgens heb ik in kaart gebracht hoe het huidige situatie in elkaar zit bij de applicaties die Wizzbit nu ontwikkeld:



Figuur 30 Architectuur huidige situatie

Om deze afbeelding goed te begrijpen is het handig om concrete voorbeelden erbij te hebben:

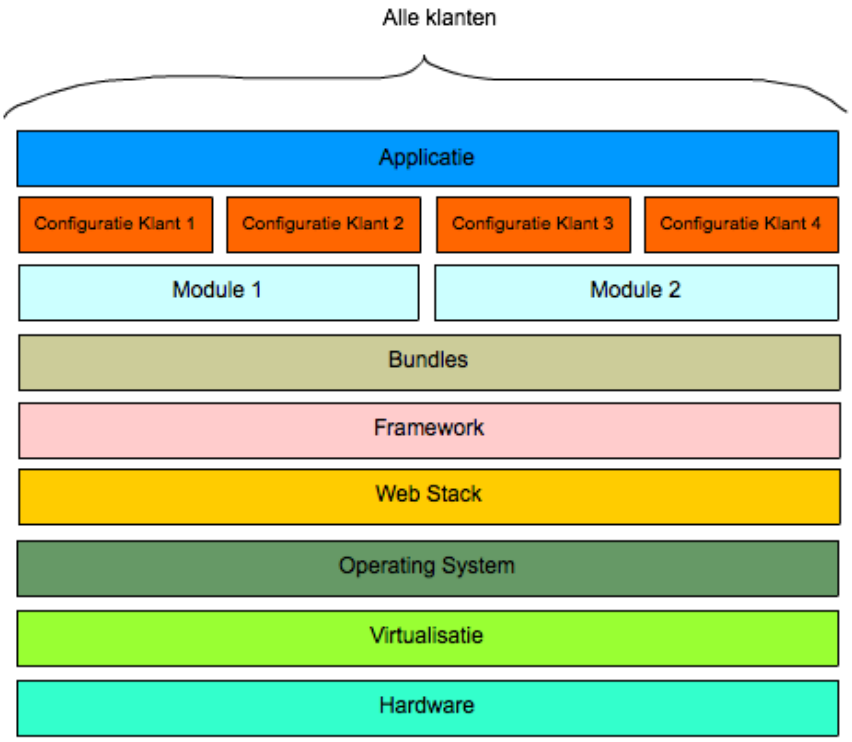
	Klant 1	**Klant 2**	**Klant 3**
Applicatie	CijferPortal	CRM	Offerte Tool
Bundles	Login 1.1, DataExporter 1.4, ORM 2.3, Pdf-Merger 0.8, ..	Login 1.2, DataExporter 1.5, ORM 2.6, Geo 2.1, CreateSend 1.0, …	Login 1.3, DataExporter 1.5, ORM 2.6, …
Framework	Symfony 2.3	Symfony 2.6	Symfony 2.6

In de tabel is te zien hoe de verschillende applicaties gebouwd zijn met behulp van het Symfony framework en verschillende bundles, de CijferPortal is daar één van. Er zijn verschillende versies van Symfony in gebruik en ook verschillende versies van dezelfde bundle, zo heeft klant 1 Login versie 1.1 en klant 2 heeft versie 1.2. Voor iedere klant en applicatie moeten de bundles en frameworks individueel onderhouden worden, updates en uitbreidingen gebeuren per klant los van elkaar. Als er een beveiligingsupdate voor een bundle moet worden gedaan dan moet dat voor iedere klant opnieuw.

Ik heb ervoor gekozen om het huidige systeem op deze manier uit te beelden omdat op deze manier duidelijk gemaakt kan worden waar nu veel tijd en onderhoud in gaat zitten bij het bouwen en onderhouden van de huidige applicaties. Ieder blok uit het figuur 30 heeft namelijk onderhoud nodig en het is zichtbaar dat er veel dubbele blokken zijn.

Nieuwe situatie

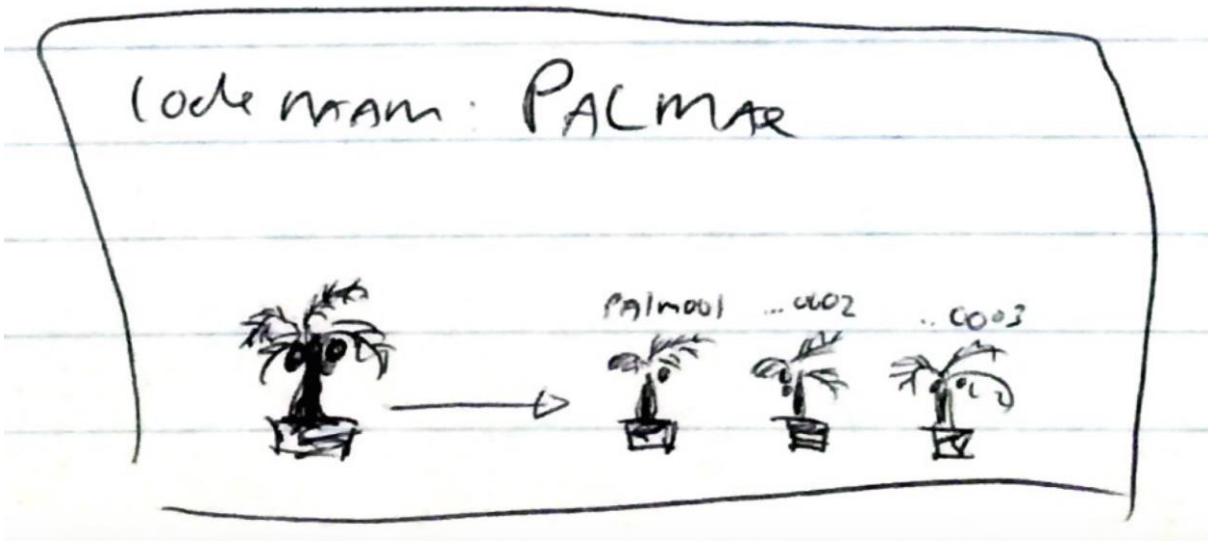
In de nieuwe situatie zullen alle klanten op dezelfde applicatie inloggen en er is alleen nog onderhoud nodig op één applicatie. Wat in de oude situatie onder verschillende applicaties valt (bijvoorbeeld CRM of CijferPortal) zal in het nieuwe systeem als generieke module beschikbaar worden en alle klanten kunnen gebruik maken van deze modules.



Figuur 31 Architectuur nieuw systeem

NAAM: PALMAE

Om het nieuwe systeem goed te kunnen omschrijven heb nagedacht over hoe de verschillende (deel)systemen genoemd kunnen worden. Hierbij heb ik geprobeerd een passende analogie te vinden en ben ik uitgekomen op **Palmae**. Palmae staat voor Palmenfamilie. In het nieuwe systeem zou je iedere klantomgeving kunnen zien als een palmboom die afstamt van dezelfde palmenfamilie (de generieke klantomgeving). De ene palmboom heeft meer kokosnoten (modules) dan de andere. Het opzetten en beheren van de klantomgevingen gebeurd via een beheertool, dit is de **TreeManager**.

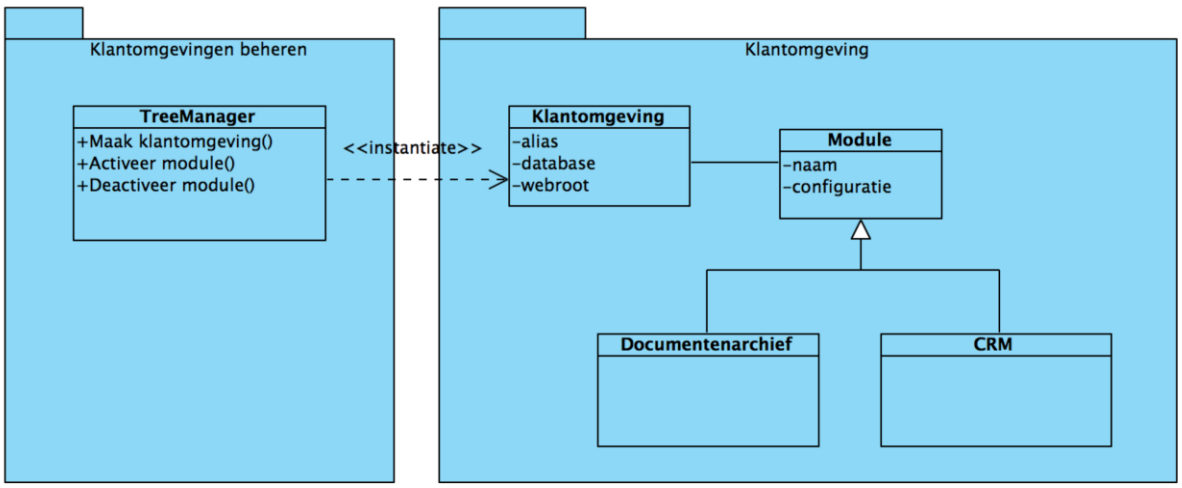


Figuur 32 Klantomgevingen (Palmen) stammen af van de generieke klantomgeving (Palmae)

KLASSENDIAGRAM PALMAE

In de eerste sprint ligt de focus op het opzetten van geïsoleerde klantomgevingen waarin modules aan en uit gezet kunnen worden. Uiteindelijk zal dan bijvoorbeeld de CijferPortal één van de modules zijn maar dat zal niet in deze sprint gebeuren. De CijferPortal moet namelijk eerst generiek gemaakt worden en dat zal in de tweede sprint gebeuren. In de eerste sprint wordt de architectuur opgezet waarmee het mogelijk wordt om modules te hebben.

In het onderstaande klassendiagram is te zien hoe ik bedacht heb om dit op te zetten. De TreeManager is verantwoordelijk voor het aanmaken van de klantomgevingen. Iedere klantomgeving heeft een eigen alias en is waarmee deze uiteindelijk te bereiken zal zijn op een subdomein, bijvoorbeeld: **klant1**.palmae.net of **klant2**.palmae.net. Iedere klantomgeving heeft zijn eigen database en webroot op het filesysteem waarmee deze geïsoleerd wordt van de andere klantomgevingen. Iedere module kan geconfigureerd worden met eigen configuratie opties.



Figuur 33 Klantomgevingen beheren met de TreeManager

OTAP OMGEVING

Een onderdeel van het opzetten van de architectuur voor Palmae is het opzetten van de Ontwikkel-, Test-, Acceptatie- en Productieomgeving.

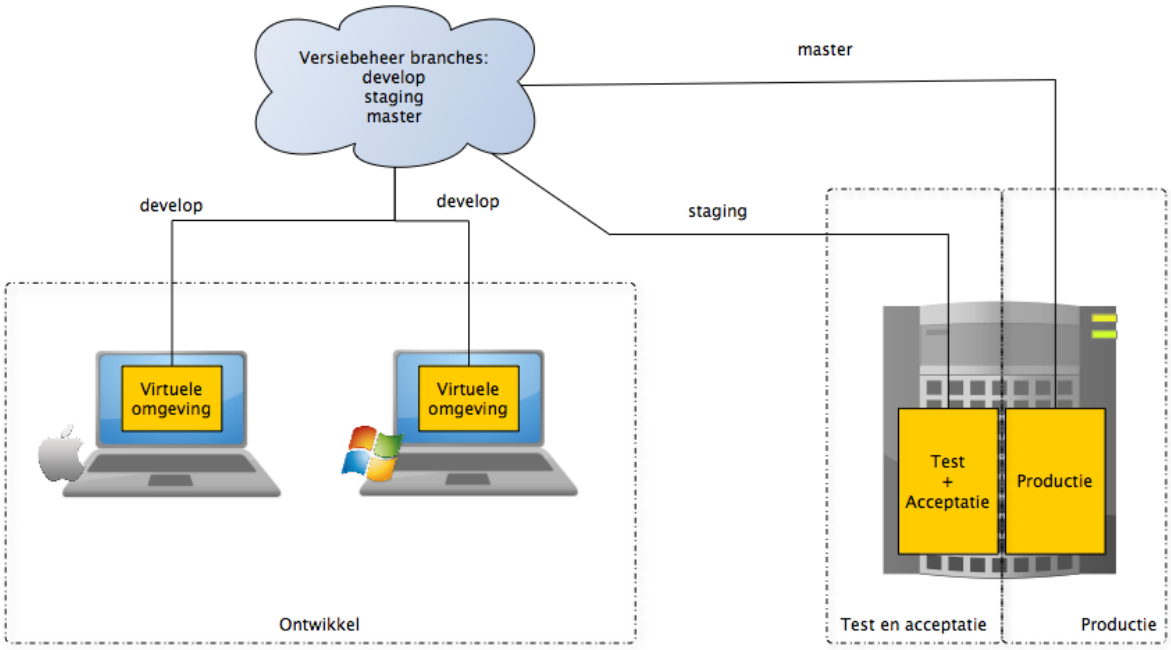
In de ontwikkelomgeving moeten ontwikkelaars het systeem kunnen ontwikkelen. Belangrijk is hierbij dat de ontwikkelomgeving zo veel mogelijk overeenkomt met de productieomgeving en dat ontwikkelaars vrij zijn om te experimenteren zonder dat de productieomgeving wordt aangetast.

In de test- en acceptatieomgeving kan nieuwe functionaliteit getest worden voordat deze aan de klanten beschikbaar wordt gemaakt.

In de productieomgeving draait de applicatie die daadwerkelijk door de klanten gebruikt wordt.

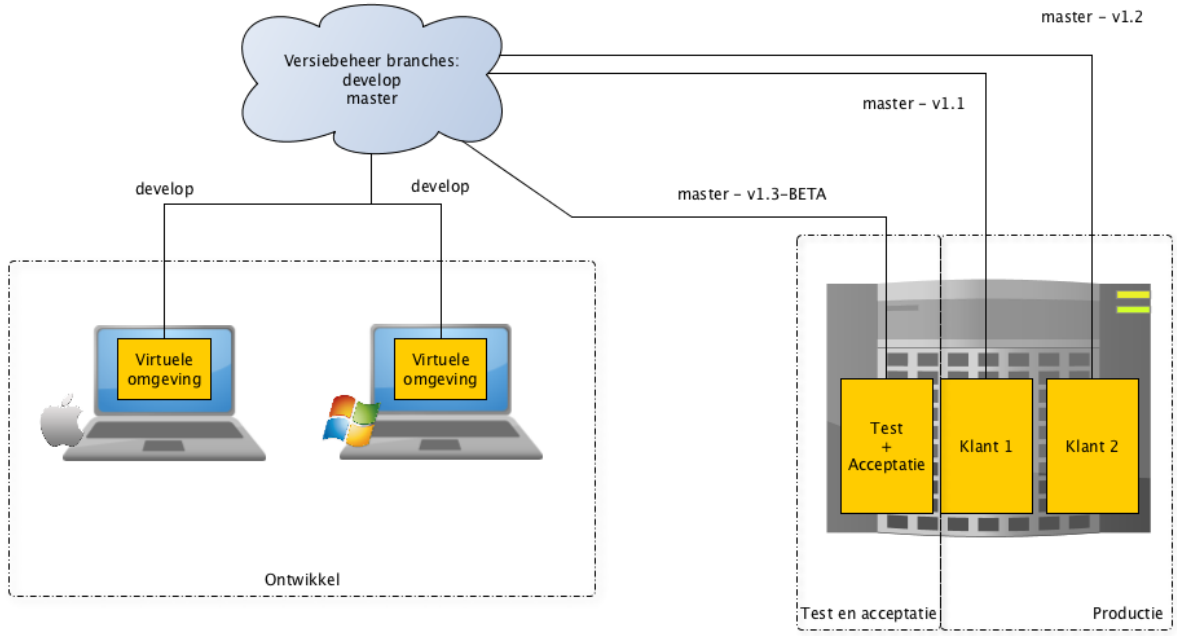
De ontwikkelomgeving: Palmae-Devbox
Er zijn verschillende manieren om een ontwikkelomgeving op te zetten
- Lokale (x)AMP
- Virtual image
- Semi-virtual vagrant- of docker-based
- Losse ontwikkelserver

Vagrant is de juiste balans tussen een ontwikkelomgeving die voor alle ontwikkelaars gelijk is en geen conflicten geeft met andere projecten. Daarnaast is het voor de ontwikkelaars prettig dat ze hun eigen OS en programma's kunnen gebruiken. Wizzbit maakt op dit moment gebruik van deze methode voor de huidige applicaties.



Figuur 34 OTAP omgeving huidige systemen (bijvoorbeeld CijferPortal)

De architectuur van Palmae is zodanig opgezet dat de klantomgevingen van elkaar geïsoleerd zijn en op verschillende versies kunnen draaien. Een test/acceptatie omgeving is dus niet meer dan een klantomgeving van Wizzbit die op een bepaalde (test)versie draait. Door iedere versie een tag mee te geven in het versiebeheersysteem (Git) kan onderscheid gemaakt worden tussen verschillende versies. In figuur 35 is te zien dat de mogelijkheid tot geïsoleerde klantomgevingen met eigen versies het gebruik van de staging branch overbodig heeft gemaakt.

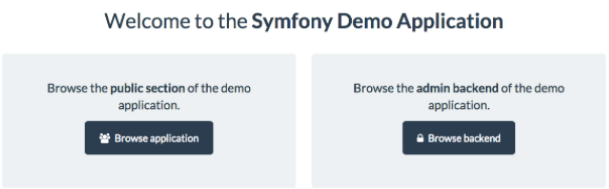


Figuur 35 OTAP omgeving Palmae

Een voordeel van verschillende versies per omgeving is dat sommige features eerst op een beperkt aantal klanten kan worden “getest”, als een nieuwe feature goed bevalt kan deze beschikbaar worden gemaakt aan alle klanten.

ONTWERPEN EN BOUWEN GENERIEKE KLANTOMGEVING

Bij het maken van de generieke klantomgeving ben ik begonnen met een nieuwe frisse Symfony installatie gebaseerd op de Symfony Demo Application distributie. Dit is een implementatie van de Symfony Best Practices.

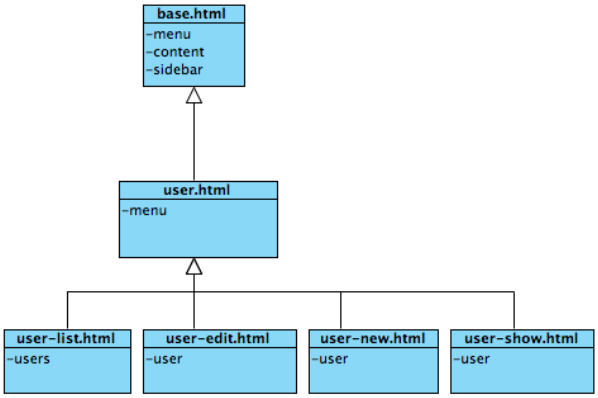


Vervolgens heb ik alles wat niet nodig is eruit gehaald (demo componenten) en ben ik begonnen met het implementeren van delen die op iedere klantomgeving nodig zijn (basis layout, login en gebruik van modules).

BASIS LAYOUT

De basis layout is opgemaakt in HTML en maakt gebruik van het Bootstrap framework. Bootstrap is een veelgebruikt HTML, CSS en javascript framework waarmee de opbouw en componenten (menu's, knoppen etc.) van een pagina los gedefinieerd kunnen worden van het uiterlijk. Hierdoor is het mogelijk om met Bootstrap-thema's makkelijk de uitstraling van een applicatie of pagina te veranderen.

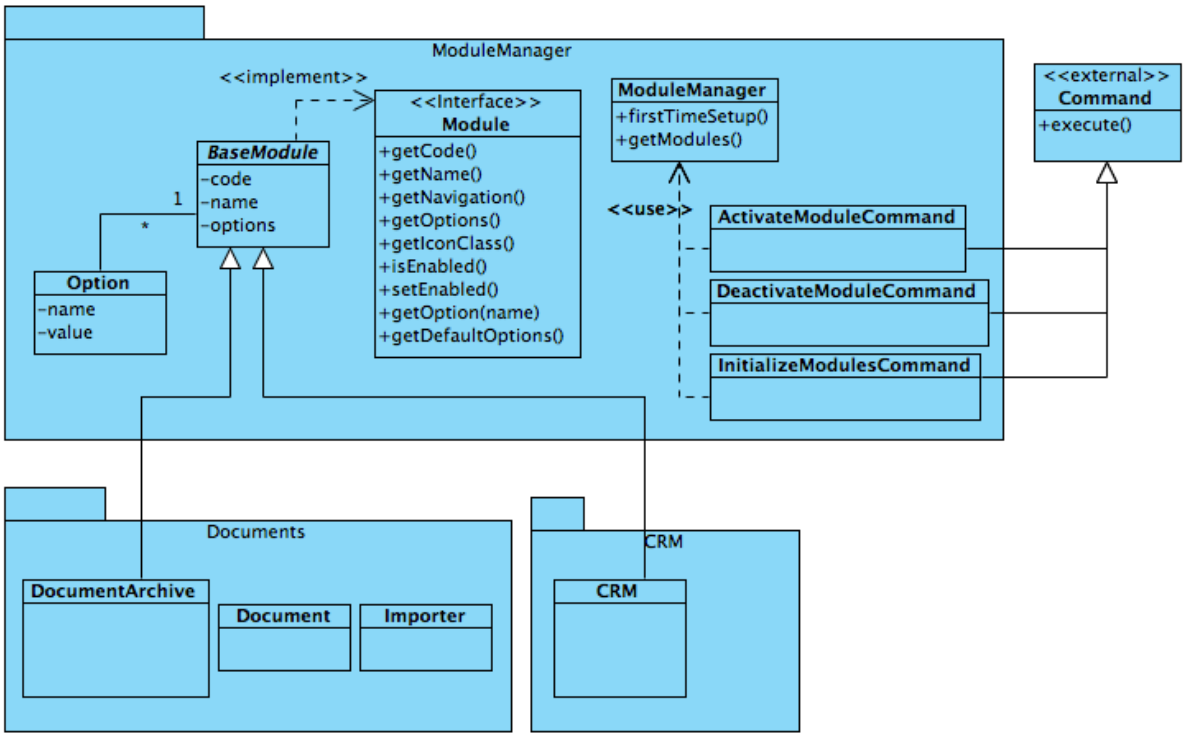
Naast het Bootstrap framework wordt er ook gebruik gemaakt van de Twig template engine. De Twig template engine maakt een hoop zaken eenvoudiger en maakt ook dingen mogelijk zoals het gebruik maken van overerving in templates.



Figuur 36 Overerving van HTML-templates

MODULES

Het moet mogelijk zijn om verschillende modules te hebben en iedere module moet te configureren zijn met eigen opties. De eerste module zal de Documenten module zijn (de CijferPortal generiek gemaakt). Iedere module is configureerbaar zijn met zijn eigen opties, het is niet bekend welke modules en opties er allemaal nog meer in de toekomst zullen zijn dus ik heb het zodanig ontworpen dat iedere module zijn eigen opties kan hebben. In het onderstaande klassendiagram is te zien hoe ik alle klassen die te maken hebben met het beheer van modules heb verwerkt in een ModuleManager package.



Figuur 37 Klassendiagram Palmae modulebeheer

De klassen hebben de volgende verantwoordelijkheden:

BaseModule: Dit is een abstracte klasse die als basis gebruikt wordt voor iedere module die in de toekomst ontwikkeld gaat worden. Deze klasse is verantwoordelijk voor het opslaan van opties die bij een module horen.

In het klassendiagram is een DocumentArchive- en een CRM-klasse als voorbeeld gemaakt die gebruik maken van de BaseModule.

Option: Deze klasse is verantwoordelijk voor het opslaan van een configuratie optie bij een module.

ModuleManager: De ModuleManager is verantwoordelijk het initialiseren van de modules en het beschikbaar maken van de modules aan de rest van de applicatie.

Activate-/**Deactivate**- en **InitializeCommand**: Deze klassen maken het mogelijk om via de command line modules aan/uit te zetten of te initialiseren. Deze klassen maken gebruik van **Command** klasse die met het Symfony framework meekomt. Deze commando's zullen uiteindelijk gebruikt worden door de **TreeManager.**

TREEMANAGER

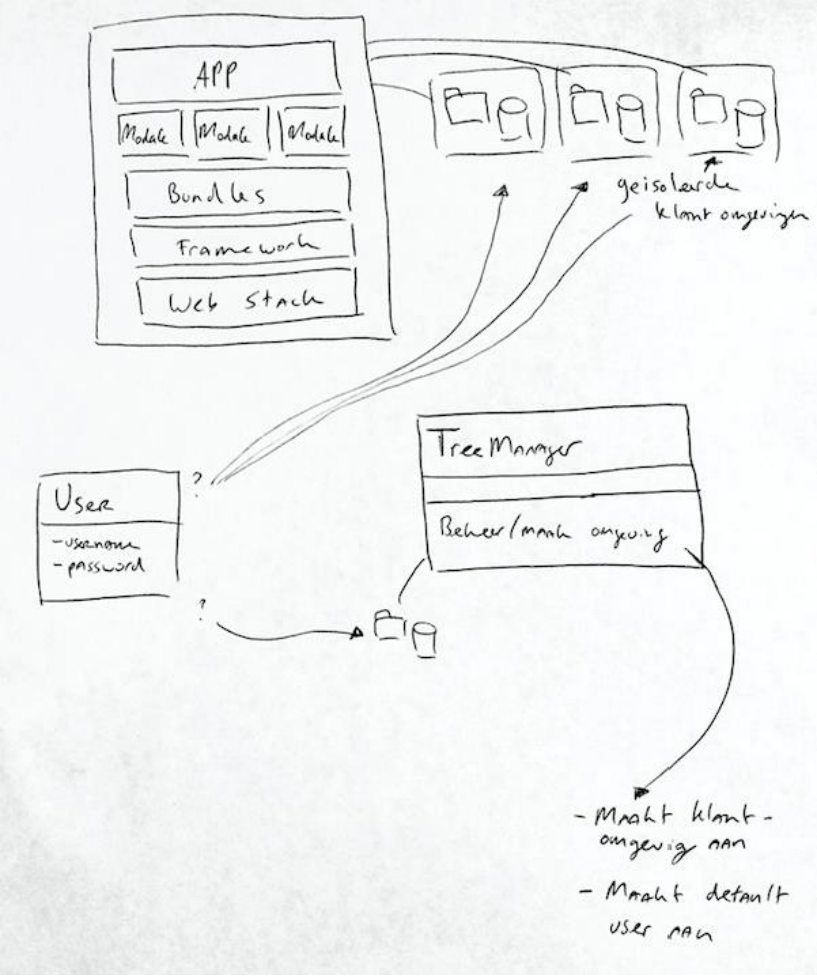
Bij de eerste versie van de TreeManager moet het mogelijk zijn om snel een klantomgeving op te zetten. Dit hoeft nog niet op een gebruiksvriendelijke manier met een grafische interface te zijn. De TreeManager moet dus de stappen die nu nog handmatig moeten uitgevoerd automatiseren. De stappen zijn nu:

1. Nieuwe database aanmaken
2. Nieuwe database user maken
3. Nieuwe database user rechten geven aan nieuwe database
4. Nieuwe webruimte aanmaken
5. Palmae generieke klantomgeving installeren en configureren
6. Nieuwe klant gebruikersnaam en wachtwoord instellen waarmee de klant op de omgeving kan inloggen

Om dit mogelijk te maken is er gebruik gemaakt van een aantal scripts die deze stappen automatiseren.

Plaatsing User-entiteit

Een vraagstuk dat naar voren is gekomen bij het ontwerpen van de TreeManager was in welk deel de klant entiteit moet kunnen “leven” in de klantomgeving of in de TreeManager?

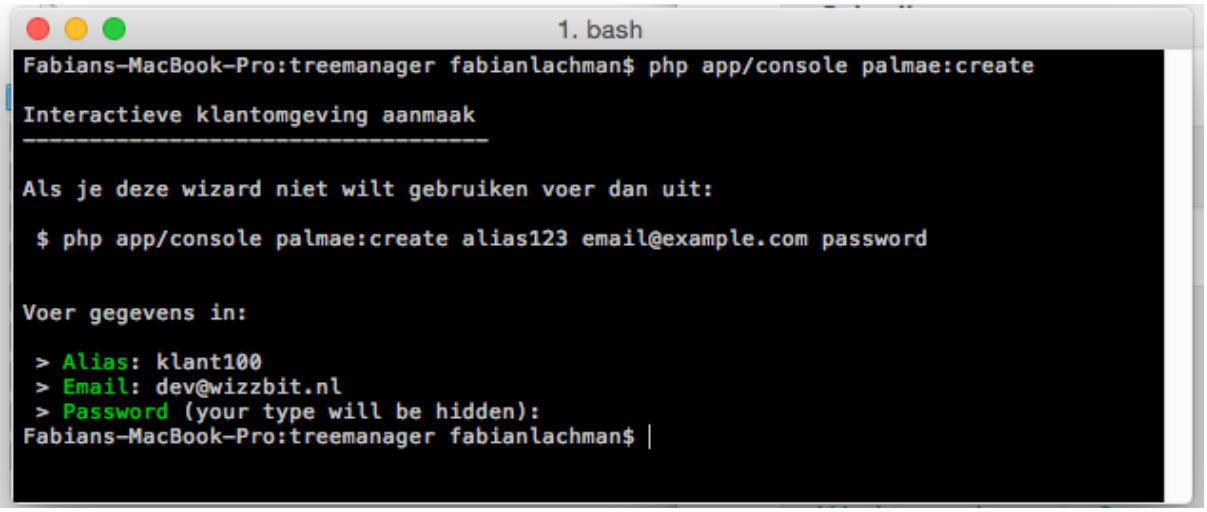


Figuur 38 Schets van waar de User-entiteit geplaatst kan worden

Uiteindelijk heb ik ervoor gekozen om de User-entiteit in de klantomgeving te verwerken. Ik heb hiervoor gekozen omdat dit veiliger is, als één klantomgeving gehackt wordt dan zijn niet meteen alle overige klantomgevingen ook gecompromitteerd. Als alle users zouden worden opgeslagen bij de TreeManager dan is dat één centraal punt met een hoop gevoelige informatie. Een andere reden is het beter kunnen onderhouden van het geheel, wanneer de TreeManager klaar is dan is het de bedoeling dat hier weinig onderhoud, updates of uitbreidingen aan nodig is. Door de User-entiteit in de klantomgevingen te plaatsen kan deze makkelijk meegroeien met nieuwe functionaliteit die wordt toegevoegd aan de klantomgevingen zonder dat daar ook nog de TreeManager voor moet worden aangepast.

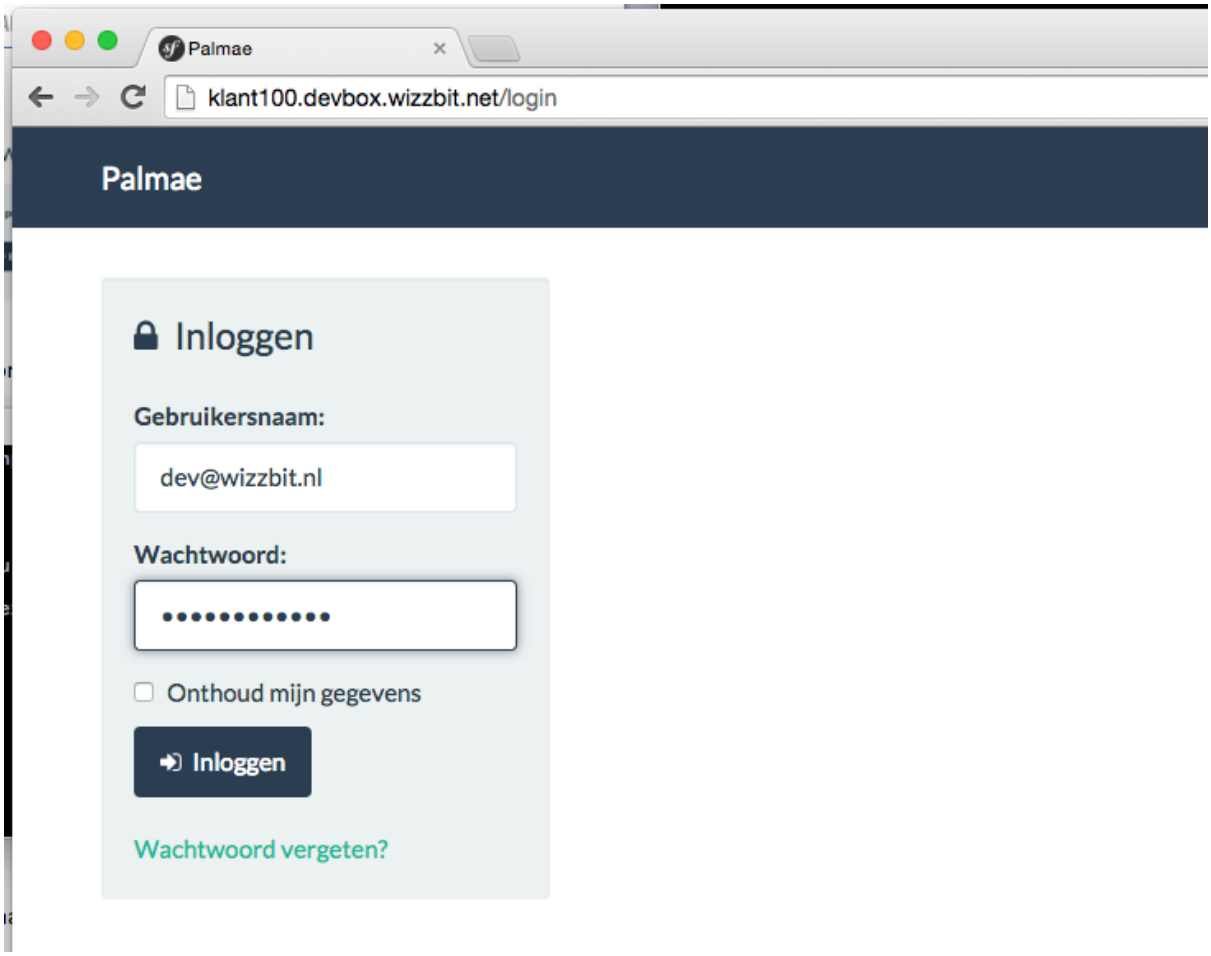
PRODUCT DEMO

Tegen het einde van de sprint heb ik een demo gegeven aan het team van Wizzbit. In deze demo heb ik laten zien hoe het met de TreeManager mogelijk is om in een paar tellen een nieuwe Palmae klantomgeving aan te maken. Dit is nu mogelijk door één commando uit te voeren en vervolgens drie vragen te beantwoorden.

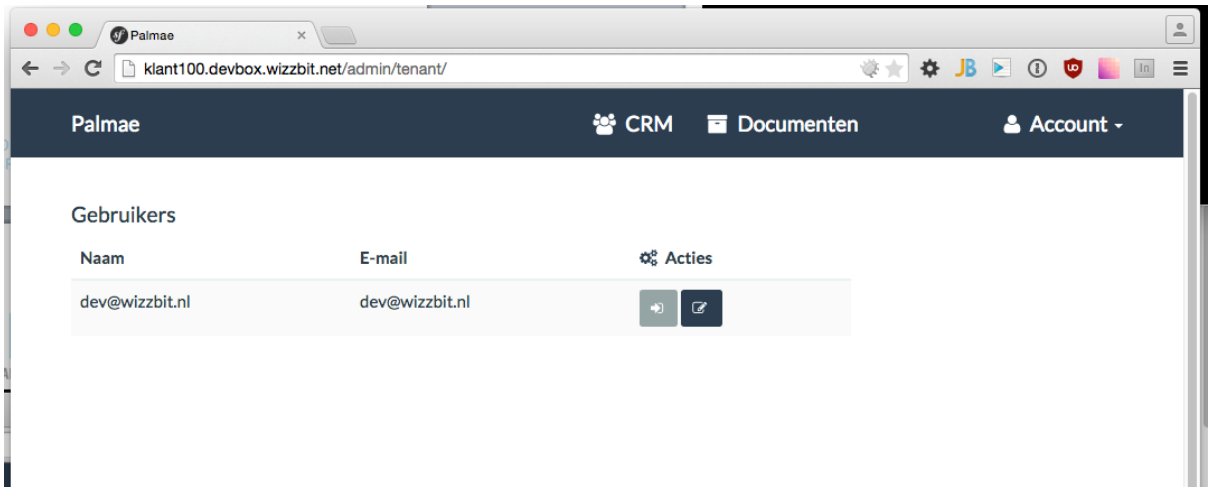


Figuur 39 Een klantomgeving aanmaken met de TreeManager

Na het beantwoorden van de drie vragen is de omgeving aangemaakt en kan er ingelogd worden:



Figuur 40 Inloggen op de aangemaakte klantomgeving



Figuur 41 Verschillende modules zijn beschikbaar in de klantomgeving (CRM, Documenten)

6.9 DE TWEEDE SPRINT: DE CIJFERPORTAL ALS GENERIEKE MODULE

De tweede sprint heeft als doel de huidige CijferPortal om te zetten naar een generieke documentenarchief module zodat deze naast facturen ook andere documenten kan opslaan.

Sprint 2: Generieke documentenarchief **Doel: De factuurportal beschikbaar maken als generieke documentenarchief module**				**49**
Het moet mogelijk zijn om verschillende soorten documenten op te slaan (In plaats van een facturen module, een documenten module, voor opslag van meer dan alleen factureren)	S2	Must	Test ontwerpen	10
			Documentenarchief module ontwerpen	13
			Documentenarchief module bouwen	13
			Testen	8
			Documenteren	5

TESTONTWERP

Om aan het einde van de sprint vast te kunnen stellen dat het doel van de sprint gehaald is zijn de volgende systeemtests geschreven:

Testcase	Requirement	Titel	Omschrijving
T5	S2	Documentarchief - Meerdere type documenten	Vaststellen dat het documentarchief als factuurarchief gebruikt kan worden
			Vaststellen dat het documentarchief als HRM-archief gebruikt kan worden
T6	S2	Documentarchief - Import	Vaststellen dat documenten geïmporteerd kunnen worden
T7	S2	Documentarchief - Lijst/Zoeken	Vaststellen dat het mogelijk is om te zoeken door het factuurarchief
			Vaststellen dat het mogelijk is om te zoeken door het HRM-archief

Figuur 42 Testcases voor de tweede sprint

ONTWERPEN EN BOUWEN GENERIEKE DOCUMENTEN MODULE

Om de CijferPortal om te zetten naar een generieke module moeten de volgende vragen beantwoord worden:

- In wat voor situaties kan de CijferPortal nog meer gebruikt worden?
- Wat zijn de overeenkomsten in deze situaties?
- Hoe kan het generiek gemaakt worden?

De CijferPortal zorgt ervoor dat gescande facturen kunnen worden gearchiveerd nadat deze door Scan Sys zijn uitgelezen. De CijferPortal is dus een add-on op Scan Sys en met de bovenste vragen in gedachten ben ik gaan kijken naar wat voor type documenten Scan Sys nog meer kan verwerken.

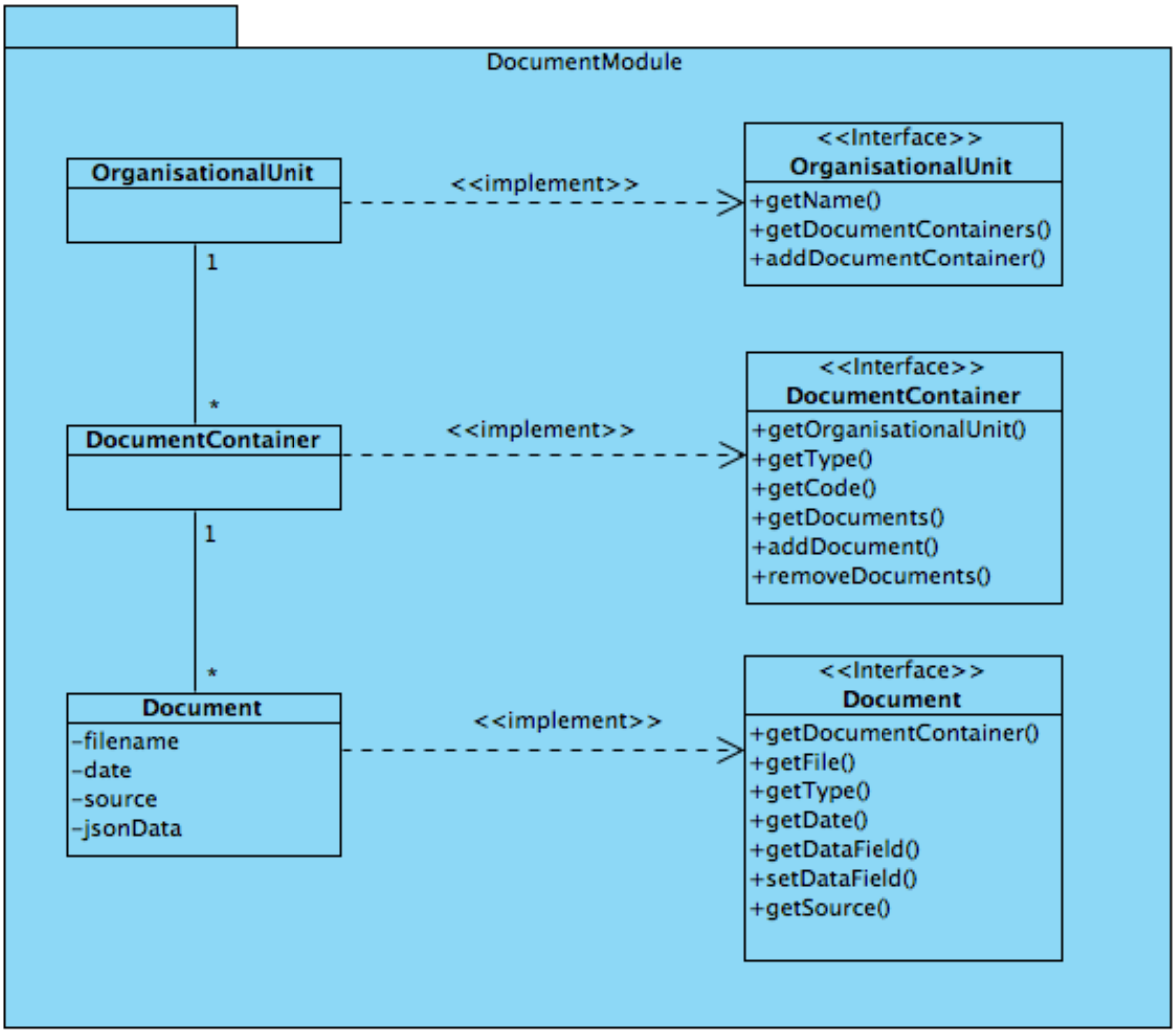
Verschillende type documenten
Er is uitgezocht met wat voor type documenten Scan Sys naast facturen ook overweg kan, dit zijn: HRM-documenten, transportdocumenten en notariële documenten.

Overeenkomsten
Voor ieder type document heb ik onderzocht hoe dit zou kunnen worden gebouwd en daarbij ben ik erachter gekomen dat er in iedere situatie steeds 3 lagen zijn die steeds terugkomen. Zo heb je bijvoorbeeld bij de facturen in de CijferPortal:
Kantoor -> Administratie -> Factuur

Bij HRM documenten heb je:
Afdeling -> Personeelsdossier -> Document

Generieke oplossing

Door ervoor te zorgen dat er 3 lagen mogelijk zijn is het mogelijk om ieder type document op te slaan. Ik heb daarvoor de volgende structuur gebruikt:



Figuur 43 Generieke oplossing voor verschillende type documenten

Informatie bij een document opslaan

Om ervoor te zorgen dat alle informatie bij een document kan worden opgeslagen heb ik naast de velden die altijd voor komen zoals datum ook een extra methode toegevoegd waarmee flexibel velden kunnen worden aangemaakt en opgevraagd zonder dat er daarvoor de klassen moeten worden aangepast.

```
30      /**
31       * Gegevens bij een document opvragen, een key kan bijvoorbeeld zijn:
32       * total, journalNumber, invoice
33       *
34       * Welke keys hangt af van het type document
35       *
36       * @param $key
37       * @return string
38       */
39      public function getDataField($key);
```

```php
    /**
     *
     * Gegevens bij een document opslaan, een key kan bijvoorbeeld zijn:
     * total, journalNumber, invoice
     *
     * Welke keys hangt af van het type document
     *
     * @param $key string
     * @param $value string
     * @return mixed
     */
    public function setDataField($key, $value);
```

Op database niveau worden deze velden opgeslagen in een JSON-structuur in een jsonData-kolom. Dankzij het ORM kan dit worden meegeven als annotatie in de betreffende klasse:

```php
    /**
     * @var string
     *
     * @ORM\Column(name="json_data", type="text", nullable=true)
     */
    private $jsonData;
```

Het ORM vertaald dit dan uiteindelijk naar het volgende relationeel representatiemodel:

```
CREATE TABLE `document` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `filename` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `date` date NOT NULL,
  `date_added` datetime NOT NULL,
  `source` longtext COLLATE utf8_unicode_ci,
  `json_data` longtext COLLATE utf8_unicode_ci,
  `documentContainer_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `IDX_D8698A767804A92E` (`documentContainer_id`),
  CONSTRAINT `FK_D8698A767804A92E` FOREIGN KEY (`documentCon
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COLLAT
```

Figuur 44 Flexibel JSON-veld voor alle overige velden

Het gebruik van deze methode is niet altijd een oplossing, velden die in dit JSON-veld zijn opgeslagen zijn namelijk minder goed te gebruiken in SQL-query's. Bij het overwegen of deze methode hier toepasbaar is, is er gekeken naar wat voor query's er uitgevoerd moeten kunnen worden en welke velden deze nodig hebben. Het enige wat nodig is zijn jaar (datum) en type, deze velden zijn generiek genoeg om voor alle type documenten te gebruiken en kunnen dus als losse velden worden opgeslagen.

Tegenwoorden hebben steeds meer databases ondersteuning om JSON als native veld op te slaan (in plaats van een TEXT-veld), dit zorgt voor een meer efficiënte opslag en de mogelijkheid tot sneller opvragen van gegevens. Er is gekeken of dit nu al te gebruiken is maar dit wordt pas ondersteund in MySQL 5.7 en hier is nog geen stabiele release van. Er zijn wel andere databases zoals bijvoorbeeld Postgress waar dit al ondersteund

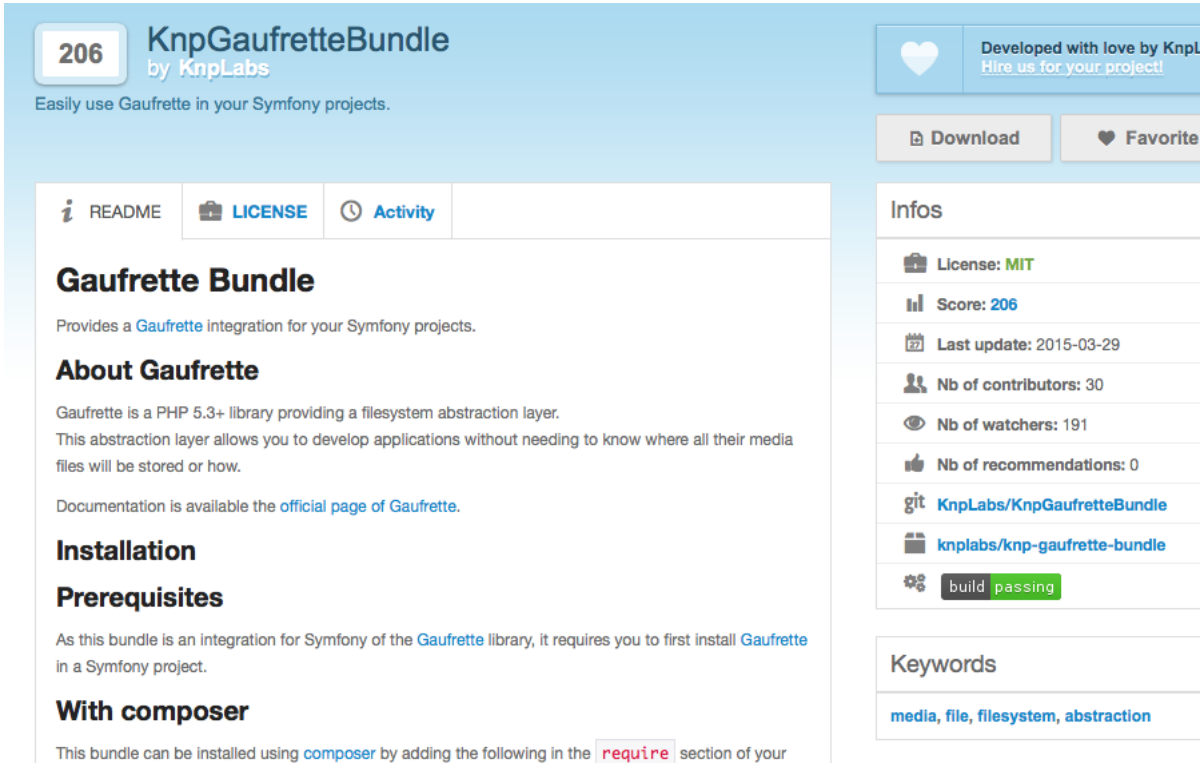
wordt maar hiervoor zou eerst een MySQL vs Postgress onderzoek gedaan moeten worden en zou het team van Wizzbit naast het onderhouden van MySQL servers ook Postgress erbij moeten doen. Dit weegt niet op tegen het (voorlopig) opslaan als TEKST. In de toekomst kan dit altijd alsnog worden omgezet bij een upgrade naar MySQL 5.7.

Flexibele opslag

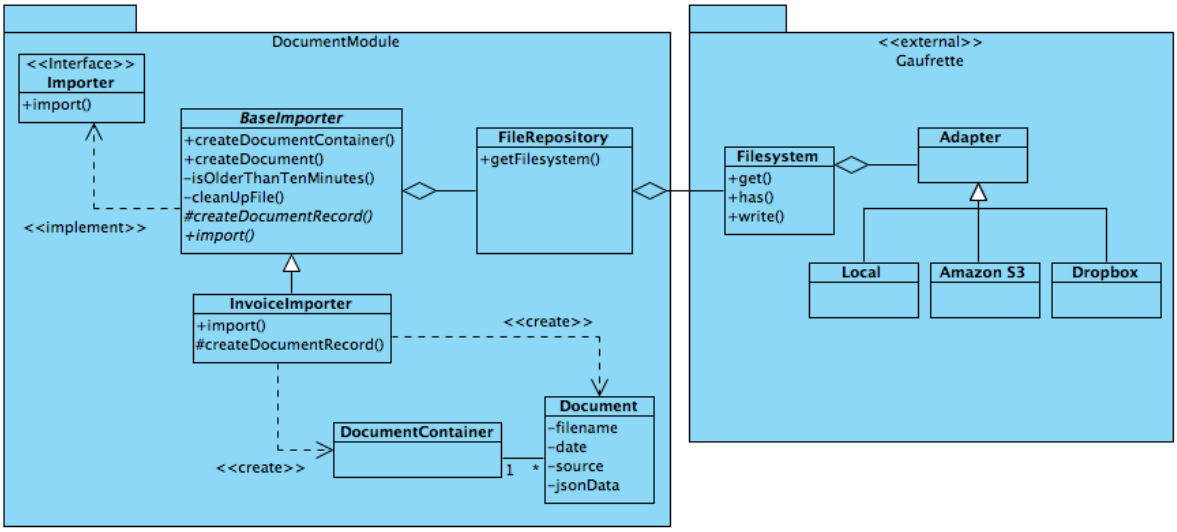
Om ervoor te zorgen dat de hoeveelheid beschikbare opslag voor documenten flexibel kan groeien is er gezocht naar een File Abstraction Layer (FAL). Hier is dezelfde methodiek toegepast als eerder in het traject en zijn ook hier deze stappen doorlopen om een geschikt pakket te vinden.

- Zijn er veel problemen met dit pakket?
- Is het pakket goed gedocumenteerd?
- Is het pakket veilig te gebruiken?
- Is het pakket vrij te gebruiken?
- Is het een veelgebruikt pakket?

Ik ben uitgekomen op het pakket Gaufrette, dit pakket maakt het mogelijk je bestanden lokaal op te slaan maar ook om te schakelen naar bijvoorbeeld externe diensten zoals Dropbox, Amazon S3 of Azure Blob Storage.



Figuur 45 Pagina met informatie over KnpGaufretteBundle



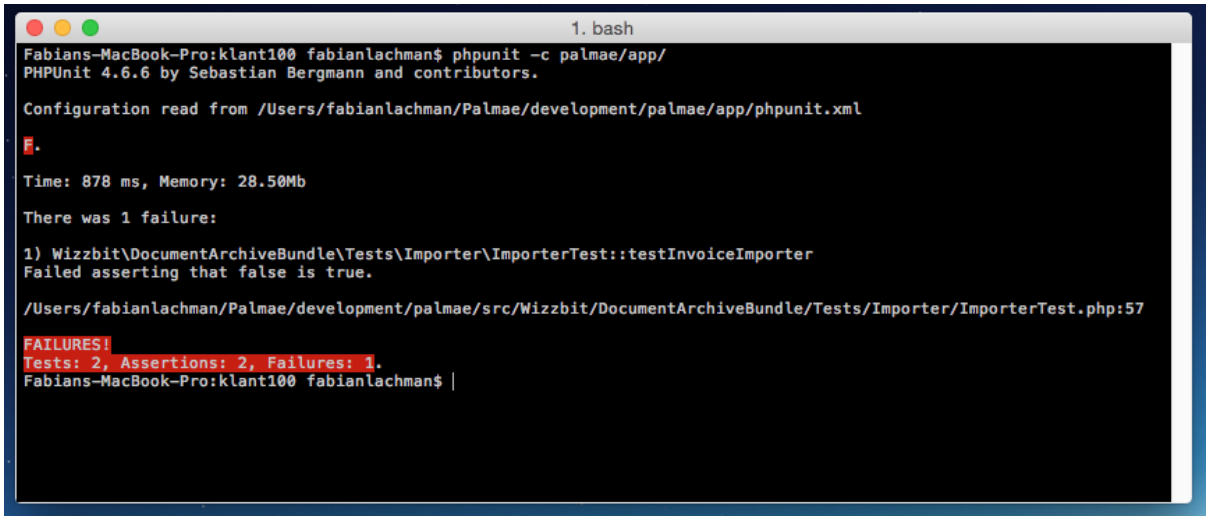
Figuur 46 Importer en gebruik File Abstraction Layer

Import functie

Om de importfunctie generiek te maken heb ik gebruik gemaakt van het template pattern waarbij een deel van de functionaliteit die nodig is bij het importeren van documenten is verwerkt in een abstracte klasse (BaseImporter), voor het importeren van facturen wordt dan een InvoiceImporter gebruikt die de BaseImporter aanvult met dingen specifiek voor facturen. Op dezelfde manier kan bijvoorbeeld ook een HrmImporter gebruikt worden voor HRM-documenten.

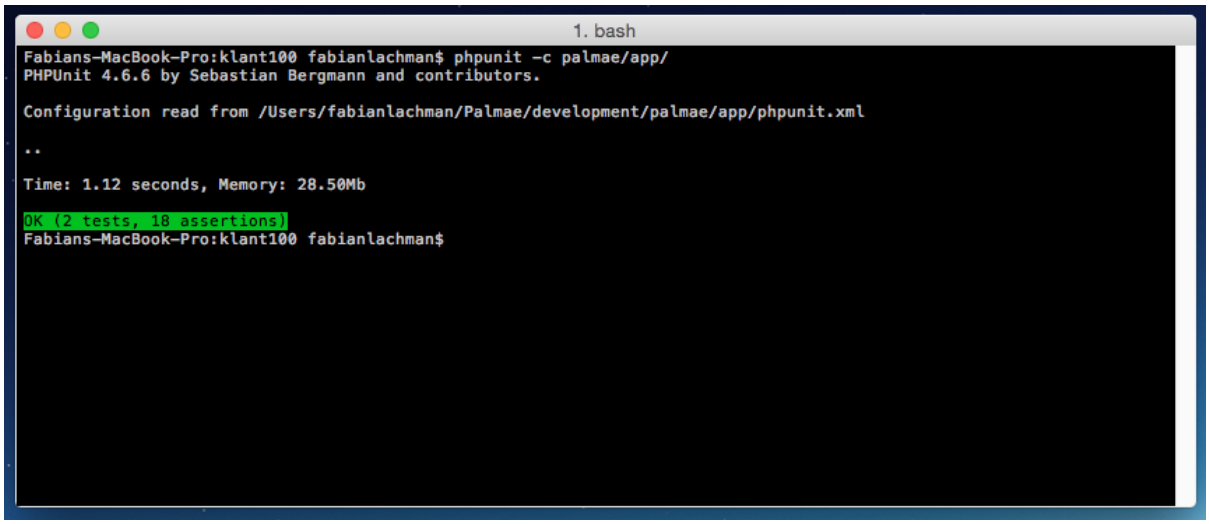
Testen

Voordat er is gestart met het bouwen van de import functie is er eerst een unit test geschreven waarmee de importfunctie getest kan worden. De test zet een paar voorbeeld documenten in de import map, voert de import functie uit en controleert daarna of bestanden zijn geïmporteerd.



Figuur 47 Unit testen; eerst een falende test geschreven

Na het schrijven van de test is de code geschreven om de functie werkend te maken en is de test slagend gemaakt.



Figuur 48 Slagende test

Vervolgens is de code opgeschoond (refactoren) zonder dat er zorgen gemaakt hoeft te worden dat er iets breekt omdat er steeds weer wordt gecontroleerd of de test nog steeds slaagt.

7 EVALUATIE

In dit hoofdstuk evalueer ik het proces dat doorlopen is tijdens de afstudeeropdracht. Daarnaast evalueer ik ook de producten die tijdens opdracht tot stand zijn gekomen.

7.1 PROCESEVALUATIE

Algemeen
Terugkijkend over de hele afstudeerperiode zou het denk ik beter geweest zijn als de afstudeeropdracht niet uit twee delen bestond, bij de afstudeeropdracht had ik de focus willen leggen op het tweede deel (Palmae) maar omdat het eerste deel bestond uit het afbouwen van de CijferPortal en dus halverwege een bestaand project was heb ik moeite gehad om een goed beginpunt te vinden voor het afstuderen. Ik ben hierop blijven haken bij het beschrijven van het eerste deel en heb ik moeite gehad met dingen op papier te zetten, zodanig dat ik veel ben gaan uitstellen en uiteindelijk zelfs met een persoonlijk dilemma zat of ik nog wel door moest gaan of niet. Op dat moment heb ik een aantal gesprekken gehad op de HHS en met mijn werkgever en uiteindelijk besloten om door te gaan. Ik heb hiervoor wel een uitstel nodig gehad voor het inleveren van het afstudeerdossier en heb afspraken gemaakt om dingen anders aan te pakken:

- Meer mijn collega's betrekken voor input en feedback
- Een aparte afstudeerruimte gemaakt om volledig te kunnen focussen en niet afgeleid te worden met de dagelijkse werkzaamheden

Daarnaast ben ik vaak met de fiets in plaats van met de auto naar het werk gegaan. Ik heb een baan waarbij ik de hele dag stil zit en ik sport niet. Het boek Spark: The Revolutionary New Science of Exercise and the Brain (Eric Hagerman & John J. Ratey, 2008) heeft mij overtuigd hoe belangrijk beweging is voor je mentale gezondheid. Dit heeft mij geholpen om beter in mijn vel te zitten.

Afwijking afstudeerplan
In het begin was de omschrijving van het tweede deel van de afstudeeropdracht: Het systeem als clouddienst beschikbaar maken. Dit was eigenlijk geen goede omschrijving omdat de wens was om de door Wizzbit ontwikkelde systemen verkoopbaar te maken. Het systeem (CijferPortal) als clouddienst beschikbaar maken maakt die wens mogelijk maar legt te veel nadruk op de CijferPortal als clouddienst. Tijdens het analyseren van de requirements en het uitgevoerde onderzoek is naar voren gekomen dat de nadruk veel meer lag op het ontwikkelen van een multi-tenant klantomgeving (Palmae) die als basis kan dienen voor ontwikkelde applicaties.

SCRUM
Het afbouwen van de CijferPortal is gedaan op de wijze zoals beschreven in hoofdstuk 4.1. Dit is handig voor alle verschillende projecten en werkzaamheden binnen Wizzbit maar minder handig om te beschrijven voor mijn afstudeeropdracht door alle andere werkzaamheden die erdoorheen lopen. Om het beter te kunnen beschrijven heb ik vanaf het tweede deel van de afstudeeropdracht (Palmae) het project uit de bestaande werkwijze van Wizzbit getrokken en als apart project los aangepakt en beschreven met een eigen SCRUM-planning en punten toekenning los van de dagelijkse werkzaamheden.

Requirements
Ik ben tevreden over de opgestelde requirements, vooral omdat het in het begin nog niet helemaal helder was wat er eigenlijk moest gebeuren om de CijferPortal verkoopbaar te maken.

Testen
In deze opdracht heb ik een klein stukje risico gebaseerd testen toegepast waarbij vooral de nadruk is gelegd op erachter komen waar de meeste aandacht aan moet worden besteed bij het testen.

UML
Vergeleken met de andere projecten bij Wizzbit heb ik redelijk veel gebruik gemaakt van UML. Achteraf gezien denk ik wel dat ik misschien soms veel details heb weggelaten in de documenten omdat veel dingen binnen Wizzbit mondeling besproken en toegelicht kunnen worden.

7.2 PRODUCTEVALUATIE

In dit hoofdstuk evalueer ik de opgeleverde producten.

CIJFERPORTAL

Over de CijferPortal ben ik tevreden, het systeem wordt volop gebruikt door CijferMeester.

CIJFERCATCHER

De CijferCatcher is nu in gebruik en is een werkend onderdeel van het gehele systeem, ik ben er tevreden over maar er zijn nog wel een paar dingen die verbeterd kunnen worden:

- Het herkennen van sommige bijlagen gaat soms niet helemaal goed
- Mail terugkoppeling kan wat uitgebreidere informatie tonen als een mail niet verwerkt kan worden, dit kan tijd besparen omdat nu de logs erbij gepakt moeten worden om te achterhalen waarom een mail niet verwerkt kon worden

Deze items staan nu op de backlog en kunnen in een volgende versie worden opgelost.

Voor de rest werkt het systeem naar behoren, de factuurscans worden door CijferMeesters en klanten gemaild naar het systeem.

CIJFERSLURPER

De CijferSlurper werkt naar behoren, om de zoveel tijd heeft CijferMeester een nieuw kantoor die klaar is om over te gaan naar het nieuwe systeem. We krijgen dan een bericht van CijferMeester om dit nieuwe kantoor over te zetten. Dankzij de CijferSlurper kan nu met weinig omkijken op een gestructureerde manier het factuurarchief worden overgezet voor ieder kantoor.

GEMIGREERDE FACTUREN

Op dit moment zijn er 57038 facturen verdeeld over 237 administraties en 6 kantoren gemigreerd naar het nieuwe systeem.

PALMAE OTAP OMGEVING

Met de OTAP omgeving ben ik tevreden, vooral de manier waarop nu ook makkelijk verschillende versies naast elkaar tegelijk kunnen draaien is een verbetering ten opzichte van de oude situatie.

PALMAE

Palmae is eigenlijk nooit af omdat er altijd nieuwe modules bij gebouwd kunnen worden maar het deel wat nu af is ben ik tevreden over. Het snel kunnen opzetten van een nieuwe klantomgeving scheelt ons een hoop tijd en onderhouden van één code repository is een stuk efficiënter dan voorheen.

RAPPORTEN EN DOCUMENTEN

De opgeleverde rapporten en documenten zijn de punten waar ik nog het meest in kan verbeteren. Ik heb misschien mijn focus te veel gelegd op het maken van werkende software.

7.3 BEWIJZEN BEROEPSTAKEN

In dit hoofdstuk omschrijf ik hoe de beroepstaken die ik heb gekozen zijn vervuld.

UITVOEREN ANALYSE DOOR DEFINITIE VAN REQUIREMENTS

Om de requirements van Palmae vast te stellen heb ik meerdere gesprekken gevoerd, het begon met hele globale eisen en wensen en aan de hand van overleg en onderzoek heb ik dit uitgewerkt in een requirementsrapport. De requirements heb ik verdeeld in business-, systeem en gebruikersrequirements en vervolgens met de MoSCoW-methode een geprioriteerd. De geprioriteerde lijst heb ik later gebruikt om te bepalen waar het eerst aan gewerkt moet worden bij het indelen van de sprints.

OPSTELLEN GEGEVENSMODEL VOOR EEN DATABASE

Ik heb met verschillende klassendiagrammen het gegevensmodel opgesteld. Voor de Documenten module in Palmae heb ik laten hoe zien de gegevens op een generieke manier kunnen worden opgeslagen.

UITVOEREN GEGEVENSCONVERSIE

Voor het afbouwen van de CijferPortal moest er een gegevensconversie gedaan worden. Het oude systeem had meer dan 60 klantadministraties die allemaal overgezet moesten worden. Het oude systeem had geen exportfunctie dus moest er een scraper worden ontwikkeld die de oude documenten kan uitlezen en overzetten.

Er was geen documentatie van het oude systeem beschikbaar dus er moest een goede analyse gedaan worden van de gegevensstructuur. Om de oude gegevens te vertalen naar het nieuwe systeem heb ik een data dictionairy gemaakt.

ONTWERPEN SOFTWAREARCHITECTUUR

Ik heb de architectuur van de CijferPortal en bijhorende deelsystemen beschreven, daarnaast heb ik voor het nieuwe systeem (Palmae) een de architectuur beschreven die het mogelijk maakt om multi-tenant applicaties te bouwen. Hierbij heb ik rekening moeten houden met uitbreidbaarheid en flexibel kunnen onderhouden van de applicaties.

BOUWEN APPLICATIE

Er zijn een aantal verschillende applicaties gebouwd waarbij gebruik is gemaakt van verschillende talen, frameworks en technieken.

De CijferPortal is gebouwd in PHP en hierbij is gebruik gemaakt van het Symfony PHP-framework, hierbij is er gebruik gemaakt van Best Practices zoals deze worden geadviseerd in de Symfony-gemeenschap.

Voor het testen is er gebruik gemaakt van PHPUnit, een framework voor het toepassen van Unit-tests in PHP.

Er is gebruik gemaakt van verschillende package managers: Composer voor PHP-pakketten, NPM voor Node.js pakketten.

Voor versie beheer is er gebruik gemaakt van Git, ik heb hier gebruik gemaakt van verschillende branches en tags.

Verschillende standaarden die gebruikt zijn: XML voor het uitlezen van data bij de factuurscans, JSON voor het uitlezen van factuurgegevens uit het oude systeem.

8 GLOSSARY

Term	Engels	Omschrijving
ASP	ASP	Application Service Provider
Boekstuk	Journal	Een boekstuk. Alle facturen werden in het oude systeem opgeslagen als boekstuk.
CijferCatcher	CijferCatcher	Deel van het systeem dat zorgt voor de verwerking van opgestuurde factuurscans
CijferPortal	CijferPortal	Deel van het systeem waar klanten en medewerkers op inloggen om facturen in te zien
CijferSlurper	CijferSlurper	Migratiescript voor het overzetten van data van het oude systeem naar het nieuwe systeem
DevBox	DevBox	Virtuele ontwikkelomgeving voor Palmae
Factuur	Invoice	Een factuur die door klanten naar CijferMeester wordt verstuurd ter verwerking
File Abstration Layer	File Abstraction Layer	Een laag waarmee een client opslagruimte kan aanroepen zonder dat de client weet waar of hoe deze opslag draait
Hoofdkantoor	Headquarters	Hoofdkantoor van CijferMeester
Inkoopfactuur	Purchase Invoice	Een type factuur
Kantoor	Office	Een van de (boekhoud)kantoren van CijferMeester
Klant	Customer	Een klant van CijferMeester
Klantomgeving	Customer Environment	Een geïsoleerde omgeving waar een klant op kan inloggen
Medewerker kantoor	Employee	Een medewerker van CijferMeester
Module	Module	Een verzameling van aan elkaar gerelateerde functionaliteiten, samengevoegd tot één module die geactiveerd kan worden per klant in Palmae
Multi-tenant	Multi-tenant	Een multi-tenant systeem kan door meerdere klanten gebruikt worden
Newviews	Newviews	Het oude systeem waar de facturen in werden verwerkt