Afstudeerverslag

Bouw van het koppelsysteem Condor

Auteur	**Lennart Kramer**
Student Nr.	**07042264**
Opleiding	**Informatica**
Instituut	**Haagse Hogeschool Zoetermeer**
Bedrijf	**42 B.V.**
Contact	**lennartkramer1988@gmail.com**

Voorwoord

Mijn dank gaat uit naar alle begeleiders en docenten die mij tijdens de studie hebben geholpen. Voor mijn afstudeerstage wil ik de opdrachtgever Eric Meijer bedanken voor het aanbieden van de opdracht. Mijn begeleider binnen het bedrijf Robert Bor voor de begeleiding en Marvin Koot als eindgebruiker voor zijn input en de goede samenwerking.

Zoetermeer, 8 januari 2015

Lennart Kramer

Inhoudsopgave

1. Inleiding

Dit verslag is opgebouwd gedurende de looptijd van mijn afstudeerstage voor de opleiding HBO Informatica aan de Haagse Hogeschool te Zoetermeer. Het verslag is bedoeld voor iedereen die geïnteresseerd is in het verloop van de stage maar kan mogelijk lastig te lezen zijn voor personen die het vakgebied niet kennen.

De afstudeerstage zal worden gelopen bij het bedrijf 42 B.V. dat sinds 2003 Java applicaties ontwikkelt en onderhoud. Alle interne applicaties bij 42 gebruiken vogelnamen en een groot deel daarvan wordt in de zogenaamde voliere gehuisvest. Dat houdt in dat deze applicaties dezelfde database gebruiken voor gedeelde informatie. Voor mijn applicatie heb ik de vogelnaam Condor gekozen.

Om te zorgen dat niet alle applicaties los direct met de database communiceren en meer controle hierover te houden is de applicatie Postduif geschreven. Postduif is een applicatie die een REST API ontsluit waarmee de andere systemen communiceren om bij de database te kunnen en waarmee Condor ook zal communiceren.

REST API: Dit is een web service API die gebruik maakt van de standaard HTTP calls GET, PUT, POST en DELETE en kan verschillende intermedia data types accepteren zoals JSON en XML.

Termen zullen tussen de regels door in blokken zoals hierboven worden uitgelegd en kunnen worden overgeslagen als de term bekend is. Onderstaande leeswijzer geeft aan hoe het verslag gelezen kan worden maar om het hele proces te bevatten kan het best alle hoofdstukken in volgorde worden gelezen.

Leeswijzer

In het tweede hoofdstuk wordt informatie over het stage bedrijf gegeven zodat er een beter beeld ontstaat over de organisatie waarbij de afstudeeropdracht is uitgevoerd. Een nadere specificatie van deze afstudeeropdracht is te vinden in hoofdstuk 3. Al het verrichtte werk voorafgaand aan de eerste sprint wordt toegelicht in hoofdstuk 4 onder de naam sprint 0.

Hoofdstuk 5 t/m 11 gaan ieder over een enkele sprint en vormen samen de 7 sprints waarin de applicatie zoals deze in de opdracht is gespecificeerd is ontwikkeld. Binnen iedere sprint wordt steeds aangegeven waarom in die sprint is gewerkt aan de onderdelen van de sprint. Daarna volgt steeds een ontwerp van de onderdelen van de sprint. Gevolgd door een omschrijving van de bouw van de onderdelen en op welke manier deze getest zijn. Iedere sprint wordt afgesloten met een reflectie over de sprint.

Hoofdstuk 12 bevat een productreflectie waarin het eindproduct nader wordt bekeken. Hierin wordt bijvoorbeeld aangegeven wat de huidige status is van de applicatie en wanneer deze in productie genomen zal worden. Hoofdstuk 13 geeft aan hoe de 7 sprints hebben geleid tot het voldoen aan de in de opdrachtomschrijving gekozen beroepstaken.

2. Informatie over het bedrijf

Om een goed beeld te krijgen van de omgeving waarin de afstudeerstage verricht wordt is het belangrijk om het bedrijf waarin dit gebeurd nader toe te lichten. Dit hoofdstuk bevat een korte geschiedenis van het bedrijf. De persoonlijke doelstelling van de CEO, de bedrijfsdoelstelling en visie.

Daarnaast is het handig om mijn positie binnen de organisatie te weten. Dit wordt tekstueel toegelicht en getoond door middel van een organogram. Het hoofdstuk wordt afgesloten met een uitleg over de ontwikkelmethodiek bij 42. Projecten zowel intern als extern gebruiken bij 42 dezelfde technieken, methodieken en ontwikkelstraat.

Geschiedenis

42 BV is in 2003 opgericht door Eric Meijer. In 2005 is besloten dat 42 BV en Kensas samen verder zouden gaan onder een nieuwe holding genaamd M.A.D. Dit staat voor Mice and Dolphins, de twee meest intelligente diersoorten op aarde volgens the Hitchhiker's Guide To the Galaxy, het boek waar eveneens de bedrijfsnaam 42 BV afkomstig van is. Kensas is opgegaan in 42 BV.

Tegenwoordig is 42 BV uitgegroeid tot een bedrijf met meer dan veertig medewerkers. De meeste medewerkers zitten in het hoofdkantoor in Zoetermeer of zijn gedetacheerd bij een klant.

Doelstelling en Visie

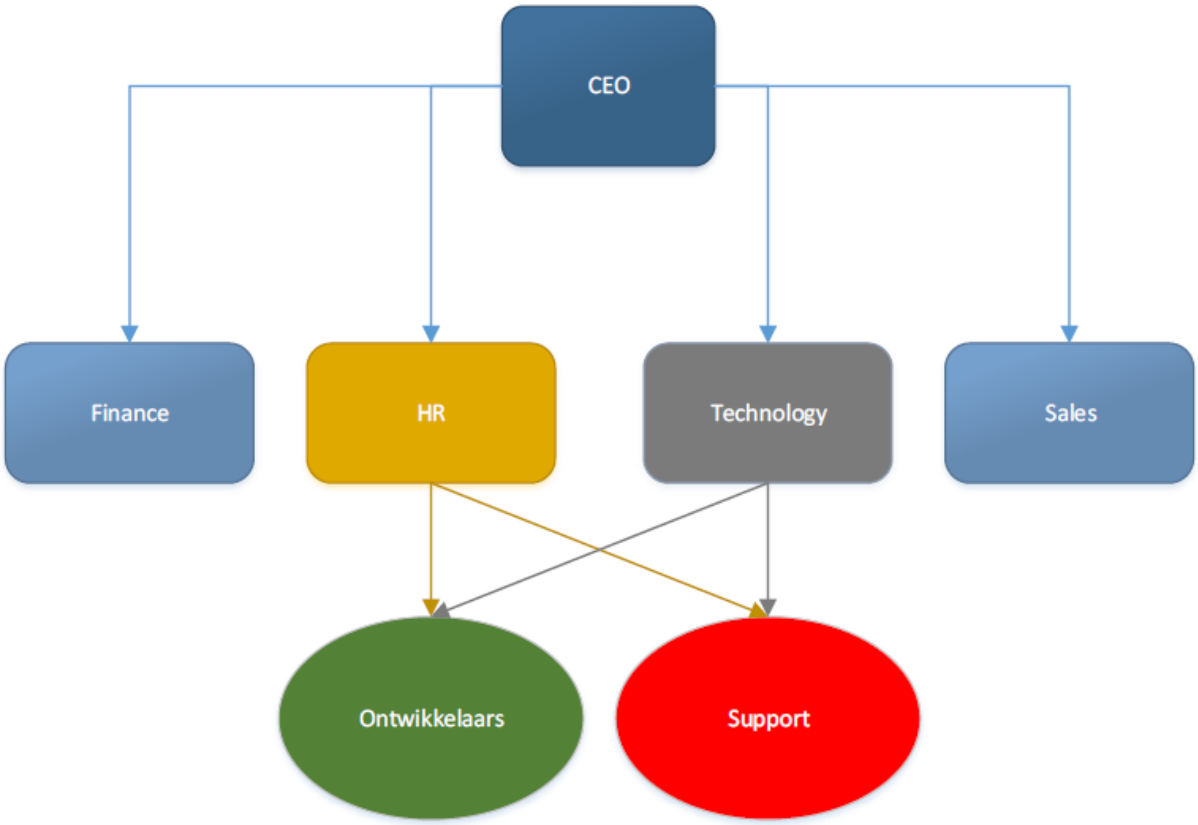
De persoonlijke doelstelling van Eric, is dat 42 BV een leuk bedrijf is waar mensen prettig met elkaar samenwerken en ook een informele sfeer hangt. Hierbij is het ook belangrijk dat er veel wordt samengewerkt.

De doelstelling op zakelijk vlak is dat het bedrijf nog mag groeien qua werknemers. Nu zijn er iets meer dan veertig werknemers, maar volgens Eric zou 42 BV het beste functioneren als er tussen de 55 en 60 werknemers zijn. Met deze hoeveelheid medewerkers kent iedereen elkaar nog en is er geen extra management laag nodig.

42 staat voor het ontwikkelen van maatwerk software waarmee klanten hun werkprocessen en bedrijfsresultaten kunnen verbeteren. Hierbij worden vooral applicaties ontwikkeld in Java. 42 Richt zich hierbij alleen op de zakelijke markt. Hierbij is er ook geen voorkeur voor een bepaalde branche wat terug te zien is in de variërende klantenbasis. Enkele van deze klanten zijn: Ahold, ASR, Belastingdienst, CBG Nidera en ANWB.

Structuur

De doelstelling komt duidelijk terug in de structuur van de organisatie. Het is een platte organisatie waarin veel onderling wordt samen gewerkt. De afdeling HR verzorgt de arbeidscontracten, werk evaluaties en werving van ontwikkelaars en support medewerkers. De technologie afdeling waar Robert Bor als CTO de leiding over heeft bepaald de gebruikte technieken, geeft advies over architectuurvraagstukken en verricht code reviews om de kwaliteit van de applicaties te waarborgen.



Figuur 1: Organogram 42 B.V.

Tussen de ontwikkelaars en support is ook een samenwerking, zo verzorgt support de accounts en rechten, beheert deze de testomgeving tijdens het afstudeerproject en de applicatie na afloop hiervan. Als Informatica afstudeerder val ik binnen de groep ontwikkelaars.

Ontwikkelstraat van 42

De projecten bij 42 zowel intern en extern maken gebruik van de ontwikkelmethodiek SCRUM en maken gebruik van dezelfde ontwikkelstraat. Hiermee kan iedereen binnen het bedrijf de vooruitgang zien van alle projecten. Opdrachtgever kan zo status van alle projecten zien. Centraal binnen de ontwikkelstraat van 42 staat de Atlassian suite waarvan meerdere producten worden gebruikt.

Atlassian suite: Softwarepakket van het Australische bedrijf Atlassian waarmee software projecten gerealiseerd worden. De onderdelen van de suite die 42 gebruikt en waarom zal later worden gespecificeerd.

Wat er allemaal wordt gebruikt van deze Atlassian suite, hoe dat gebeurd en waarvoor zal tijdens de loop van dit verslag besproken worden.

3. De opdracht gespecificeerd

In dit hoofdstuk wordt de opdracht nader omschreven en een beter beeld gegeven wat de huidige situatie is en welke problemen er met het project worden opgelost. Daarna wordt de gewenste situatie en doelstelling geformuleerd waarmee het uiteindelijk gewenste resultaat beter duidelijk wordt.

Huidige situatie / probleemstelling

42 realiseert langlopende projecten waarvoor een uren- en projectadministratie wordt gevoerd, waarvoor een aantal systemen wordt gebruikt:

- De financiële zaken worden vastgelegd in een boekhoudpakket (Twinfield).
- Medewerkers registreren gewerkte uren in het urenregistratiesysteem.
- De projectadministratie bevat afspraken met betrekking tot de financiën.
- De al verzonden of conceptfacturen worden bijgehouden in het facturatiesysteem.
- Excel sheet met lopende periode en forecast.
- Handmatig toevoegen van de maandelijkse journaalposten.

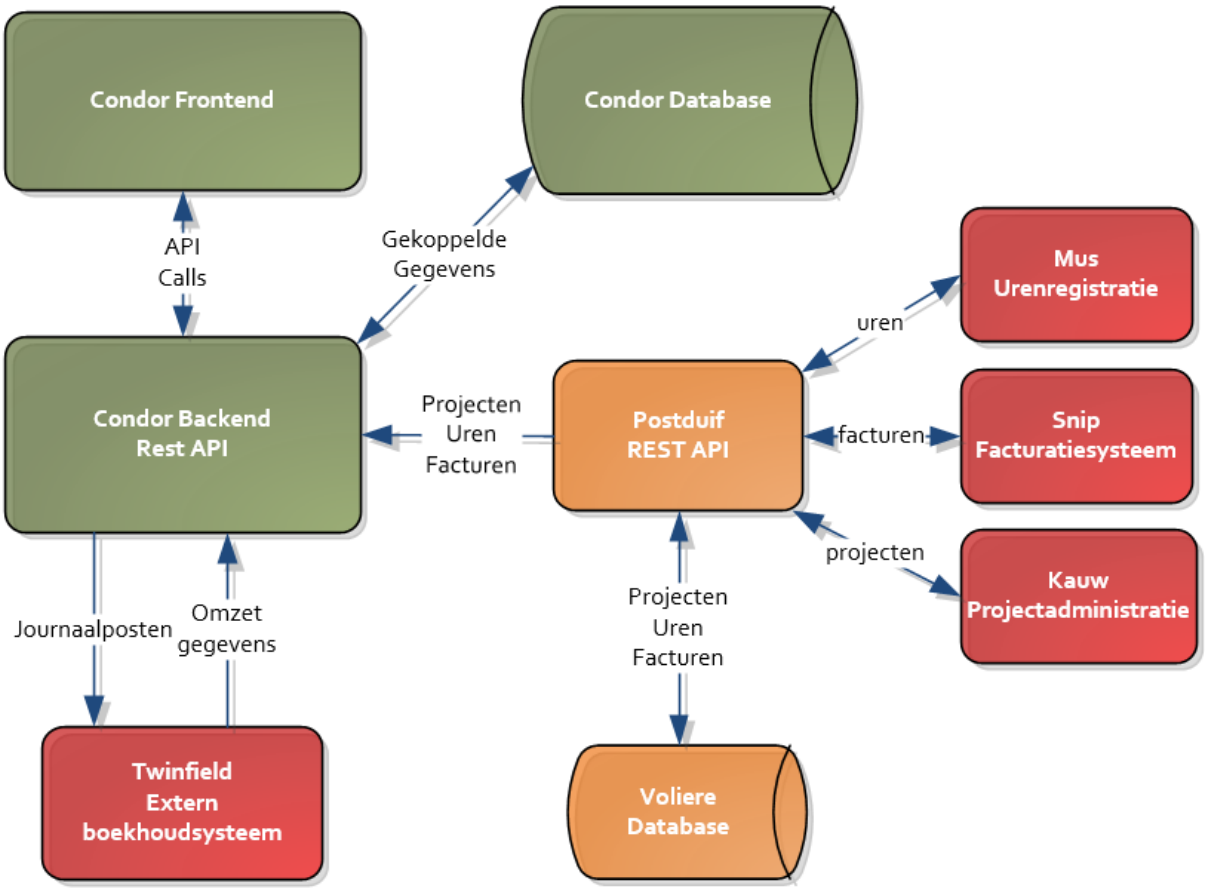
De integratie tussen de verschillende systemen is beperkt. Het is nu mogelijk om vanuit SNIP (facturering) naar Twinfield (boekhouding) factuurinformatie over te zetten. Wat nog niet kan is het uitlezen van informatie en het koppelen van gegevens om inzicht te krijgen in de lopende periode en de forecast.

Gewenste situatie / doelstelling

De doelstelling is om een betere integratie te krijgen tussen het boekhoudpakket, urenregistratie, project en facturatie systeem. Waarbij een duidelijk overzicht van de lopende periode en een forecast worden ontwikkeld. Daarnaast moet ook het boekhoudkundig proces makkelijker worden gemaakt door het automatiseren van het inboeken van correctie journaalposten.

Na realisatie van het project zal er meer inzicht zijn in de huidige status van een project met betrekking tot uren en kosten. Een inzichtelijk dashboard zal de huidige Excel sheet vervangen en meer informatie bieden m.b.t. gemaakte omzet en een omzet forecast bieden. Het handmatig werk van het boekhoudpakket zal zijn afgenomen en meer geautomatiseerd.

Het figuur op de volgende pagina geeft weer hoe het systeem in de omgeving zal staan. Het betreft een verder uitgewerkte versie van de schets in Bijlage A: Afstudeerplan hoofdstuk 2.4.



Figuur 2: Context diagram Condor

De groene onderdelen zijn de delen die nog niet bestaan en tijdens de afstudeerstage ontwikkeld zullen worden.

Oranje onderdelen zijn al bestaande delen die mogen worden aangepast indien dit nodig is. Er mogen dus bijvoorbeeld toevoegingen aan de Postduif API gedaan worden of een extra view in de voliere database toegevoegd.

Met de rode onderdelen wordt direct of indirect gecommuniceerd maar kunnen niet worden aangepast. Belangrijkste is hiervoor de Twinfield XML SOAP API, dit heeft ten gevolge voor het te bouwen systeem dat er XML documenten moeten worden opgebouwd en uitgelezen.

XML SOAP API: Dit is een API gebaseerd op het Simple Object Access Protocol dat wordt gebruikt om gestructureerde informatie te delen. SOAP maakt gebruik van de XML structuur voor het versturen en ontvangen van data.

De Condor backend is zelf een REST API waarmee andere systemen ook weer mee moeten kunnen communiceren. De frontend staat geheel los van de backend en voert alleen maar API calls uit om gegevens op te halen en te versturen van en naar de backend.

De gekoppelde gegevens uit Postduif en Twinfield worden samen met eventuele correcties of concept correcties opgeslagen in een eigen database. Deze opslag vind plaats om dataverkeer naar de externe systemen te minimaliseren. En hierdoor ook de vertraging die daardoor ontstaat weg te nemen.

Ontwikkelmethodiek

Sprints binnen 42 duren standaard twee weken en zal ook voor dit project gehanteerd worden. Echter omdat het hier gaat om een eenmansproject zullen niet alle onderdelen van Scrum worden toegepast. Er zal geen dagelijkse sprint meeting worden gehouden.

Aan het einde van elke Sprint zal er een sprint review meeting gehouden worden met de opdrachtgever, eindgebruiker (product-owner) en stage begeleider(scrum-master). In deze meeting wordt een presentatie gehouden over wat er tijdens de Sprint is gedaan, een demo gegeven van het huidige product, worden onduidelijkheden besproken en worden de requirements bijgewerkt indien nodig.

Na deze meeting volgt ook direct de sprint planning meeting. Hierbij zal de planning voor de volgende sprint worden besproken. De taken die hieruit voorkomen worden in de sprint backlog bijgehouden door middel van het programma Jira.

JIRA: Dit is een issue management systeem en onderdeel van het Atlassian pakket dat door 42 wordt gebruikt om per project de product backlog bij te houden. Jira bevat tevens een grafiek vergelijkbaar met een burndown chart.

Op te leveren producten

De volgende producten worden tijdens het project ontwikkeld en de definitieve versies zullen aan het einde worden opgeleverd.

- Plan van Aanpak
- Requirement document
- Broncode in GIT
- Unit Testen
- Database
- Documentatie op Confluence.

Het plan van aanpak en de eerste versie van het requirement document zullen worden geschreven voordat de sprints beginnen. De broncode wordt tijdens de sprints geschreven op basis van de globale planning vanuit het plan van aanpak en eventuele wensen uit de sprint meetings. Voor deze code zullen unit tests worden geschreven over de gehele periode van het project heen.

De database zal in PostgreSQL worden gemaakt. PostgreSQL is op dit moment de bedrijfsstandaard, voorheen was dit MySQL maar doordat dit werd overgenomen door Oracle en door dreigende aanscherping van gebruikerslicentie is de overstap naar PostgreSQL gemaakt.

PostgreSQL: Dit is een open source relationeel database management systeem dat de sterke punten van twee andere database management systemen probeert te combineren (MySQL en MS SQL Server).

Alle documentatie zoals database modellen, installatie handleiding voor development omgeving en deployment handleiding wordt op de bedrijf wiki Confluence gezet.

Confluence: Dit is een onderdeel van het Atlassian pakket dat door 42 wordt gebruikt om alle documentatie in te bewaren. Kort gezegd is dit een Enterprise wiki systeem.

4. Sprint 0

Tijdens de voorbereidingsfase wordt het requirements document gestart. In dit document worden de user stories en use cases beschreven. Daarnaast wordt er een plan van aanpak geschreven voor eventuele risico's, afbakening van de scope en het maken van een globale planning. Deze globale planning is meer een ontwikkelvisie waarvan tijdens de sprints nog afgeweken kan worden. Tot slot wordt tijdens sprint 0 besproken hoe het project is opgezet.

4.1. Schrijven van de user stories

De user stories worden geschreven om de wensen van de eindgebruikers op te nemen. Op basis van de opdrachtomschrijving konden de eerste user stories worden geschreven. Om deze user stories te verifiëren en nog gemiste stories te ontdekken is er een kick off meeting met de opdrachtgever en eindgebruiker. Deze kick off meeting wordt voorbereid door de opdracht nog eens goed door te lezen en vragen die hieruit voorkomen op te stellen.

Zo was het nog niet helemaal duidelijk wat er allemaal in de omzet Excel sheet stond die het systeem moest vervangen. Ook was het niet duidelijk welke handelingen verricht werden tijdens het bijhouden van deze sheet. Al deze handelingen zouden waarschijnlijk wel belangrijke user stories worden. Zo lang niet alle functionaliteit wordt overgenomen kan de sheet niet worden weg gedaan.

Ook was er de wens voor een dashboard maar nog niet duidelijk wat ze met dit dashboard wilden kunnen. Daarnaast stond er in de opdracht dat het boekhoudkundig proces makkelijker moest worden door het automatisch boeken. Welke boekingen dit waren was nog niet duidelijk. Om goede user stories te schrijven moest het allemaal dus specifieker worden.

#	Omschrijving	Prioriteit
1.	Als een gebruiker wil ik de omzet bekijken gesorteerd op factuurnummer per maand.	Hoog
2.	Als een administrateur wil ik concept correcties toevoegen.	Hoog
3.	Als een administrateur wil ik concept correcties wijzigen.	Hoog
4.	Als een administrateur wil ik concept correcties verwijderen.	Hoog
5.	Als een administrateur wil ik concept correcties definitief maken door ze in te boeken.	Hoog
6.	Als een administrateur wil ik concept facturen alvast kunnen boeken zodat deze in de omzet worden opgenomen voordat de factuur is verstuurd.	Midden
7.	Als een administrateur wil ik geboekte concept facturen kunnen matchen met een definitieve factuur of het bedrag laten vervallen.	Midden
8.	Als een gebruiker wil ik dat er eerst moet worden ingelogd zodat de gegevens beveiligd zijn tegen ongeautoriseerde toegang.	Hoog
9.	Als een gebruiker wil ik dat er kan worden uitgelogd en de autorisatie vervalt.	Midden
10	Als een gebruiker wil ik een dashboard met totaalcijfers voor omzet, OHW en storno.	Midden
11	Als een gebruiker wil ik een omzet specificatie kunnen inzien voor de opbouw van de totaalcijfers in het dashboard.	Midden
12	Als een gebruiker wil ik kunnen synchroniseren met Twinfield waarmee boekingen worden opgehaald die nog niet in het systeem stonden.	Hoog
13	Als een gebruiker wil ik de details van een memoriaal boeking kunnen bekijken zodat ik een beter inzicht krijg in de opbouw van deze boeking.	Laag
14	Als een gebruiker wil ik de details van een memoriaal boeking kunnen printen zodat ik deze aan de administratie kan toevoegen.	Laag

Tabel 1: User stories die zijn uitgewerkt in het software systeem

4.2. Opstellen plan van aanpak

Zoals de meeste projecten begint het met een plan van aanpak waarin het doel, de scope, eventuele risico's en een globale planning worden beschreven. In het project plan dat voorafgaand aan de stage is geschreven stonden het doel en de scope al. Deze hoefden dus alleen gecontroleerd te worden en waren in hoofdstuk 3 "de opdracht gespecificeerd" al te lezen.

Het risico dat is geïdentificeerd is de Twinfield API. De opdrachtgever en project begeleider hebben laten weten dat de Twinfield API verwarrend is. Er zijn andere programmeurs geweest waarbij het niet gelukt is om eruit te krijgen wat nodig is. Een probleem dat zou kunnen ontstaan is dat het voor mij ook niet lukt om alle benodigde handelingen uit te voeren.

Risico	1. Problemen met de Twinfield API
Omschrijving	De opdrachtgever en project begeleider hebben laten weten dat de Twinfield API verwarrend is. Er zijn andere programmeurs geweest waarbij het niet gelukt is om eruit te krijgen wat nodig is. Een probleem dat zou kunnen ontstaan is dat het voor mij ook niet lukt om alle benodigde handelingen uit te voeren.
Gevolgen	Niet alle resultaten zoals beschreven in het afstudeerplan kunnen behaald worden.
Kans	Middel, gezien het feit dat er al andere programmeurs zijn waarbij het niet goed is gelukt.
Impact	Middel, er zal mogelijk een klein deel van de beschreven functionaliteit niet worden opgeleverd maar het systeem kan nog steeds bruikbaar zijn.
Risiconiveau	Middel
Oplossing	Vroeg beginnen met communicatie van en naar de Twinfield API om eventuele problemen zo snel mogelijk te detecteren en trachten deze op te lossen.

Tabel 2: Belangrijkste risico uit het plan van aanpak

De belangrijkste user stories en risico's zijn nu bekend en op basis hiervan kan een globale planning worden gemaakt. De planning zoals deze is opgesteld tijdens het schrijven van het plan van aanpak is te zien in de tabel op de volgende pagina. Later zijn hier een aantal wijzigingen aan gemaakt maar de planning is in grote lijnen aangehouden.

De focus zal in de eerste sprints liggen op het leggen van verbindingen tussen Condor en de externe systemen. Het vastknopen van de gegevens en boekingen om te komen tot een eerste opzet van de omzet sheet. De sprints daarna zullen dienen voor het maken van correcties binnen deze pagina en vervolgens het definitief maken van deze correcties.

Daarna wordt een dashboard toegevoegd waarin totaalcijfers per maand wordt getoond. De gemaakte functionaliteit zal worden geoptimaliseerd en de test fase zal worden gestart. Als de testfase is doorlopen zal alles naar de acceptatie omgeving worden overgezet.

Sprint nr.	Start datum	Eind datum	Producten	Werkzaamheden
0	1-9-14	5-9-14	Plan v aanpak, Requirements	Schrijven plan van aanpak. Ontwikkelomgeving opzetten. Begin aan requirements document.
1	5-9-14	17-9-14	Plan v aanpak, Applicatie, requirements, testen	Ophalen van de eerste gegevens uit Postduif. Volledige technische stack bouwen van backend tot frontend.
2	17-9-14	30-9-14	Applicatie, requirements, testen	Twinfield connectie leggen en facturen ophalen. Toevoegen van unit testen voor de geschreven code.
3	30-9-14	14-10-14	Applicatie, database, requirements, testen	Twinfield connectie uitbreiden en meer gegevens ophalen. Eigen database toevoegen om gegevens te persisteren. Concept correcties toevoegen, wijzigen en verwijderen.
4	14-10-14	28-10-14	Applicatie, database, requirements, testen	Wegboeken naar Twinfield toevoegen. Correcties toevoegen aan database en applicatie. Toevoegen van security aan frontend en backend.
5	28-10-14	11-11-14	Applicatie, database, testen	Optimaliseren van al opgeleverde functies zoals splitsen en verwerken concepten. Omzet sheet verbeteren. Storno en OHW overzicht toevoegen.
6	11-11-14	25-11-14	Applicatie, database, testen	Omzetgeschiedenis en forecast per project. Storno en OHW overzicht optimaliseren. Laatste grote requirements afmaken. Deployment naar de test server
7	25-11-14	9-12-14	Applicatie, database, testen	Feedback verwerken vanuit test omgeving. Dashboard optimaliseren. Laatste kleinere requirements afmaken. Toevoegen van een audit trail.
8	9-12-14	23-12-14	Documentatie & Deployment	Laatste optimalisaties aan de frontend. Oplevering naar productieomgeving. Volledige documentatie op Confluence.

Tabel 3: Globale planning uit het plan van aanpak

4.3. Opzetten van het project

Nu er een globale planning is gemaakt en de eerste requirements zijn vastgelegd kan er begonnen worden met het opzetten van het project. Veel technieken staan vast binnen het bedrijf. Samenwerking binnen het bedrijf vindt 42 belangrijk maar dan moet iedereen wel dezelfde technieken hanteren.

Zo moet de backend een REST API in Java en Spring MVC zijn. Daarnaast moet er voor de Database PostgreSQL gebruikt worden en JPA/Hibernate om deze met Java classes te mappen. Deze backend wordt dan gebruikt door de frontend die API calls zal maken naar de eigen backend. Deze frontend zal geheel los staat van de backend en alleen door API calls verboden zijn.

Spring MVC: Dit is het op dit moment meest gebruikte Java framework en bevat enorm veel standaard functionaliteit. Het fungeert ook als een inversion of control container.

De frontend moet worden gebouwd in Angular JS, HTML en CSS. Er wordt gewerkt met Maven voor het ophalen van alle dependencies en met git voor versiebeheer. Deze verplichtingen bestaan vooral om te zorgen dat 42 als bedrijf efficiënter wordt met een kleine set aan tools waar we veel mee werken.

Angular JS: Dit is een open source frontend framework dat gebruik maakt van controllers en services in javascript. Daarnaast maakt het gebruik van attributen in de HTML en two way data binding. Daarmee worden gegevens in javascript gewijzigd en direct in de pagina veranderd.

De IDE waarmee gewerkt wordt mag vrij worden gekozen. Ik ben begonnen in STS (Spring tool suite), deze IDE is open source en gebaseerd op Eclipse met wat toegevoegde functionaliteit speciaal voor het spring framework. Maar omdat debuggen hierin niet naar wens werkte (het was omslachtig) is de overstap naar IntelliJ gemaakt. Hierin werkt het uitvoeren en debuggen van unit testen, lokale Tomcat servers en Maven plug-ins een stuk makkelijker.

Condor krijgt een koppeling met Postduif (de API die contact heeft met de gemeenschappelijke voliere database) en met de API van het externe boekhoudpakket. Om een beter beeld te krijgen van het probleemdomein, lokaal te kunnen testen en eventueel aan te passen van andere systemen worden een aantal van de andere systemen geïnstalleerd. Daarnaast wordt een dump van de voliere database gebruikt om lokaal dichter tegen de echte gebruikerservaring aan te zitten.

Vervolgens wordt er een nieuw project opgezet en de eerste dependencies van Spring worden toegevoegd. Dit project wordt toegevoegd aan Stash via een commit in git naar de lege repository. Bij een commit naar de stash repository kan een JIRA issue nummer worden meegegeven waardoor de commit aan een issue wordt gekoppeld. Je kunt hierdoor voor een issue in Jira zien welke comits hiervoor zijn gedaan.

Stash: Onderdeel van de Atlassian suite waarmee versie beheer wordt geregeld, functioneert als een git repository systeem.

5. Sprint 1

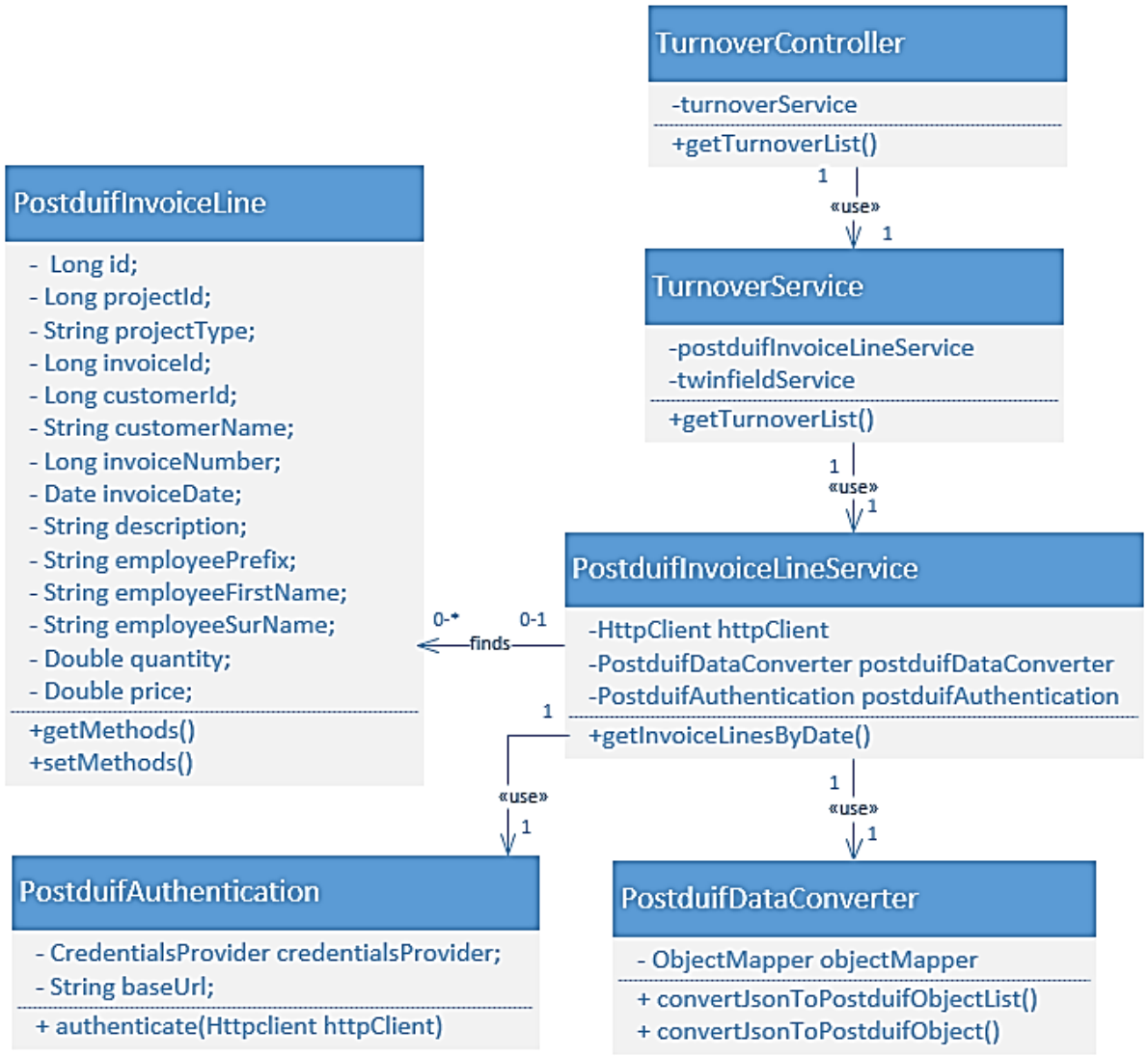
5.1. Vaststellen van de requirements

De opdrachtgever heeft in de kick-off meeting aangegeven snel de contouren van de Excel sheet terug te willen zien in de frontend. De frontend moet ook snel interactief zijn en niet naar het laatste moment opgeschoven worden.

Om dit te realiseren zal er dus een connectie met postduif gemaakt moeten worden om de gegevens van de Excel sheet op te halen. Tevens moet de volledige stack van het programma opgezet worden inclusief frontend. Deze twee taken zullen aan de sprint backlog in Jira worden toegevoegd voor sprint 1.

5.2. Ontwerpen van de systeemonderdelen

Onderstaand klassendiagram toont de eerste klassen die de volledige stack van de backend bevatten. De frontend zal bij de turnover controller de gegevens opvragen.



Figuur 3: Klassendiagram eerste klassen

5.3. Bouwen van de eerste verbinding Postduif

Als start wordt er een API call gemaakt naar de Postduif API waarmee de laatste tien facturen worden opgehaald. Hiermee kan de httpClient worden getest die de API calls naar het andere systeem uitvoert. En de JSON parser die de ontvangen JSON response omzet naar Java objecten.

Om dit in de eigen frontend te tonen wordt een eigen API controller gemaakt die de gegevens weer aanbied op een eigen URL. Om te beginnen worden de gegevens één op één doorgegeven naar de eigen API. De eerste opzet is nu compleet en de frontend kan bij de eigen API nu via API calls de facturen gaan opvragen en vervolgens tonen.

Deze facturen konden echter nog niet per periode uit de postduif API gehaald worden vanwege een aantal redenen:

- Omdat deze optie nog niet was toegevoegd.
- Facturen bevatten geen omschrijving.
- In de boekhouding worden de bedragen per factuurregel los geboekt (De ene factuurregel kan op een ander grootboek rekening worden geboekt dan een andere regel op dezelfde factuur).

Daarom is er gekozen om de factuurregels op te halen deze hebben wel een duidelijke omschrijving zo staat er indien er sprake is van een factuur op uur basis vermeld welke medewerker en welke periode. In de Excel sheet werd deze splitsing ook gemaakt en werden de uren er handmatig bijgezet. Een bijkomend voordeel is dat in de Postduif API voor het factuurregel deel wel al het opvragen per periode was ingebouwd.

5.4. Bouwen van de frontend

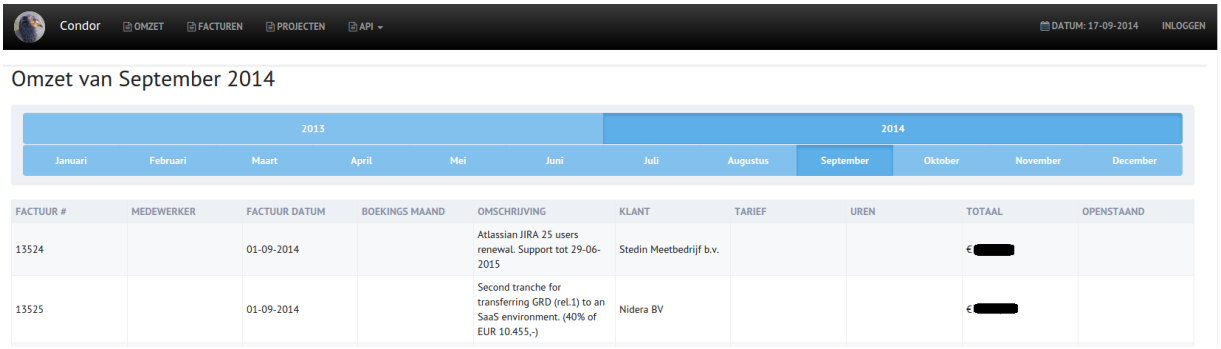
Nu de eerste connectie is gelegd met Postduif en de gegevens worden aangeboden in de eigen API is het tijd om de frontend helemaal op te zetten. Zoals eerder vermeld dit was een wens van de opdrachtgever die liever niet naar moeilijk te lezen witte pagina's met lijstjes data keek. Er moet dus een overzichtelijke tabel gemaakt worden om de leesbaarheid te verbeteren.

De wens van de opdrachtgever is dat de stijl in overeenkomst is met twee van de andere interne systemen. De opdrachtgever wil dit om consistentie te behouden binnen de systemen. Zie onderstaande afbeeldingen voor voorbeelden van de frontend van deze twee systemen.

Figuur 4: Links frontend Kauw PMS en rechts frontend SNIP Facturatiesysteem

De omzet in de Excel sheet is per maand verdeeld en gebaseerd op de verkoopfacturen. Om dit over te nemen zullen de facturen uit Postduif moeten worden gehaald met een factuurdatum die binnen de opgevraagde maand valt. Daarnaast is de sortering op basis van factuurnummer dus dit zal ook over genomen worden om de Excel sheet in eerste instantie zoveel mogelijk na te bootsen. Later worden er mogelijk nog meer sorteringen en filter mogelijkheden toegevoegd.

Nu de backend API met de eerste gegevens klaar staat moet er nog een werkende frontend komen die deze API aanroept. De kolommen van de Excel sheet worden omgezet naar kolommen van de frontend tabel en de eerste ophaalbare informatie wordt getoond. Als een andere maand wordt gekozen zal via AngularJS steeds de wisselende inhoud opgehaald worden door de API call te wijzigen en de inhoud te veranderen. Er hoeft dus nooit een page refresh te worden gedaan waardoor de pagina's snel verwisselen. Zie onderstaande afbeelding voor de eerste versie van de omzet pagina in de frontend.



Figuur 5: Eerste versie frontend scherm voor Omzet per maand.

5.5. Testen van de systeem onderdelen

Het is de bedoeling dat de test coverage in de buurt van de 100% ligt maar tijdens de eerste sprint was er nog niets gedaan aan testen. Het is gebruikelijk om de onderdelen ook direct te testen en ze zijn eigenlijk pas af als dit gebeurd is. Helaas was er geen tijd meer voor om in deze sprint unit tests toe te voegen en eigenlijk was er dus nog niks af in sprint 1. Tijdens sprint 2 zal alle functionaliteit van sprint 1 alsnog getest moeten worden.

5.6. Reflecteren van het proces

Er zijn al aardige stappen gemaakt en is het gelukt om alle goals te behalen. Vooral de frontend ziet er beter uit dan ik verwacht had en ging het aanpassen van Postduif soepel. Ik ben tevreden over de gemaakte vooruitgang in de eerste sprint.

Enige jammere was dat de functionaliteit nog niet getest was, dit kwam doordat er veel nieuw was in de eerste weken. Hierdoor was er geen tijd meer om de test suite goed in te richten en de eerste tests met Junit te schrijven. Ik had beter meer tijd kunnen besteden aan testen dan kleine aanpassingen in de frontend.

6. Sprint 2

6.1. Vaststellen van de requirements

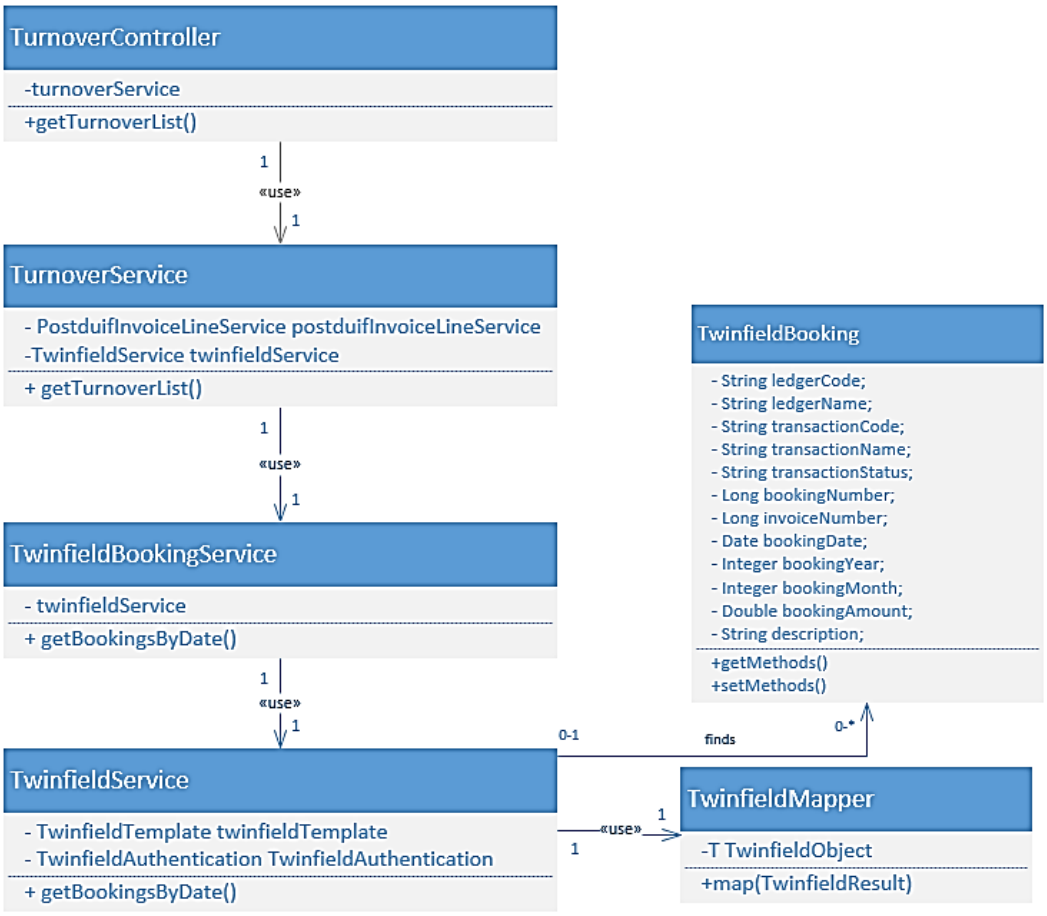
Voor de tweede sprint is besloten om de code die in de eerste sprint geschreven is onder het vergrootglas te leggen. Alle geschreven code wordt nog is bekeken om te zoeken naar verbeterpunten. De rede hiervoor is om te voorkomen dat problemen zich gaan opstapelen en fouten vroegtijdig op te sporen.

Daarnaast wordt de testdekking die aan het einde van de eerste sprint nog 0% bedroeg opgehoogd, het streven is om hier 80% van te maken. Daarnaast zal in de ontwikkeling van de applicatie vooral de connectie met het boekhoudpakket worden uitgebreid om te proberen de koppeling met de omzetsheet te maken. De connectie met het boekhoudpakket is nodig omdat dingen als boekstuknummer en grootboekrekeningnummer alleen daar zijn opgeslagen en niet via Postduif op te vragen.

6.2. Ontwerpen van de systeem onderdelen

Er zal een extra laag worden toegevoegd bij de services waarbij de " PostduifService" en "TwinfieldService" de enige twee klassen zijn die contact hebben met de buitenwereld.

Ook wordt tijdens deze sprint het Twinfield gedeelde uitgewerkt zoals te zien is in onderstaand klassendiagram.



Figuur 6: Klassendiagram sprint 2

6.3. Bouwen van connectie met Twinfield

Nu via de connectie met postduif de factuurgegevens worden opgehaald voor een periode moet voor iedere factuur de boekingsgegevens los opgehaald worden. De Twinfield API is helaas niet zo snel en omdat dit een extern boekhoudpakket/API is valt hier niets aan te veranderen. Daarom is er gekozen om maar één call te maken naar de Twinfield API om alles van een periode op te halen.

Er wordt een meeting gehouden met de eindgebruiker / product owner (Marvin Koot, functie: Administrateur) om te kijken wat er allemaal met het boekhoudprogramma gedaan wordt. Directe aanleiding voor deze meeting was het feit dat het mij nog niet duidelijk was wanneer ik de complete data set had. De hoofdvraag was dus ook wanneer heb ik alle gegevens die nodig zijn voor de omzet berekening. Het resultaat van deze meeting was dat vooral de memoriaalboekingen nog ontbraken.

Memoriaalboeking: Via dit type boeking worden niet reguliere boekingen zoals correcties gedaan.

Alle facturen die gemaakt worden in het facturatiesysteem van 42 worden automatisch geboekt op een omzetrekening en krijgen een factuurnummer uit Twinfield. Dat houdt in dat het ophalen uit Twinfield volledig dekkend is voor definitieve facturen en memoriaal boekingen. Daarom kan er een wijziging plaatsvinden waarbij de Twinfield gegevens leidend worden.

6.4. Bouwen van ophalen van boekingsinformatie

Tijdens het schrijven van het plan van aanpak was de Twinfield API als belangrijkste risico aangemerkt omdat het lastig is om deze te gebruiken. Andere programmeurs waren er mee aan de slag gegaan maar kregen er niet uit wat gevraagd werd. Er zal dus eerst moeten worden gekeken of het wel lukt om alle gegevens op een correcte manier uit de Twinfield API op te halen.

De Twinfield API is niet erg gebruikersvriendelijk en het is lastig om eruit te krijgen wat je wilt hebben. Je moet in de header eerst meegeven welke velden je precies wilt en kan dan ook parameters van bijvoorbeeld de periode hieraan meegeven. De aangevraagde velden krijg je dan weer in de XML response terug en moet je dan via dezelfde veldnaam als de header terugvragen. Ook ben je gelimiteerd aan de velden en header selectie opties die Twinfield aanbied.

De test account van het boekhoudprogramma was erg leeg en zaten geen echte boekingen in waardoor dit niet te matchen viel met de Postduif gegevens. Daarom is wat echte data uit productie overgenomen op de test account zodat de koppeling beter te maken valt.

Even kort samengevat er gaat nu één call gemaakt worden naar Twinfield om alle gegevens van een periode op te halen. Vanuit dit resultaat zal op basis van factuurnummer de factuur regels van die factuur erbij worden gehaald. De juiste regel wordt gematched op basis van bedrag.

Om dit te bewerkstelligen heb ik een tijdje moeten zoeken in de Twinfield API documentatie maar het bleek wel mogelijk om al het gewenste via één call op te vragen. Er moesten een aantal instellingen veranderd worden maar het kon wel om alle omzet gegevens voor één periode te krijgen. Via deze connectie kunnen er ook memoriaalboekingen worden opgehaald.

Nadat de gegevens uit Twinfield gehaald zijn worden er API calls gemaakt naar de postduif API en op basis van Factuurnummer en het bedrag van de factuur regel worden deze gematched met de

Twinfield gegevens. De memoriaalboekingen worden niet met een factuur gematched maar staan nu wel in het omzet overzicht.

Op deze manier ontbreken alleen nog de conceptfacturen vanuit Postduif, deze zullen in een volgende sprint moeten worden toegevoegd. De gecombineerde gegevens worden daarna via de eigen API aan de frontend aangeboden.

6.5. Bouwen van eigen annotatie en mapper

Om de koppeling tussen de resultaten uit Twinfield en de Twinfield modellen in condor te verbeteren worden Java annotaties en een mapper toegevoegd. Omdat dit een vrij technisch onderdeel is zal de werkwijze aan de hand van code voorbeelden worden beschreven.

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Twinfield {
    String fieldName();
}
```

Sinds de toevoeging van Java annotaties in JDK 1.6 is het door veel libraries opgepakt. Zo zijn er built-in annotaties van Java, Spring annotaties en andere externe libraries gebruiken ze ook.

Figuur 7: Twinfield annotatie klasse

Door aan te gegeven “@Target(ElementType.Field)” weet de compiler dat de annotatie op velden gebruikt wordt. En zal een waarschuwing geven als het ergens anders gebruikt wordt. Deze annotatie kan daarna op het veld worden toegevoegd om de Twinfield veldnaam aan het model veld te koppelen.

```
@Twinfield(fieldName = "fin.trs.line.invnumber") private String invoiceNumber;
```

Figuur 8: Twinfield annotatie toegepast op invoiceNumber veld

De waarde van deze annotatie kan nu uit de veldinformatie van de klasse gehaald worden. Daarvoor is de klasse “TwinfieldAnnotationHandler” geschreven met de methode getFieldMappings(Object). Deze methode geeft een map terug met als key de veldnaam van het Twinfield model dus bijvoorbeeld “invoiceNumber”. De waarde die aan de key wordt verbonden wordt uitgelezen uit de Twinfield annotatie.

```
public Map<String, String> getFieldMappings(Object ob) {
    Map<String, String> fieldMappings = new HashMap<~>();
    Field[] fields = ob.getClass().getDeclaredFields();
    for (Field field : fields) {
        if (field.isAnnotationPresent(Twinfield.class)) {
            fieldMappings.put(field.getName(), field.getAnnotation(Twinfield.class).fieldName());
        }
    }
    return fieldMappings;
}
```

Figuur 9: getFieldMappings methode uit de TwinfieldAnnotationHandler class

Deze annotatie handler wordt door de Twinfield Mapper gebruikt om de field mappings op te halen voor het te mappen object. Daarna wordt er een Bean Wrapper aangemaakt, dit is een class uit de Spring MVC library die gebruikt kan worden om de setters aan te roepen op basis van de veldnaam.

Er wordt door de map met veldnamen gelopen en steeds de setter van het veld aangeroepen op basis van de veldnaam. De waarde die aan de setter wordt meegegeven wordt uit het Twinfield resultaat object gehaald op basis van de Twinfield veld naam.

In de TwinfieldService kan nu na de API call in één regel vanuit het return object de Java objecten aangemaakt worden. Zonder dat steeds voor elk veld los de setter moet worden toegevoegd. De onderhoudbaarheid is hierdoor verbeterd, mochten er meer waarden opgehaald worden of hetzelfde object in andere API calls dan hoeft alleen het model aangepast te worden.

6.6. Testen van de systeemonderdelen

Voor sprint 2 moesten de onderdelen van sprint 1 nog worden getest en dat hield vooral in het schrijven van een groot aantal unit tests. Aan de hand van code voorbeelden zal één van de testen worden getoond.

Deze test zal als voorbeeld dienen voor de andere tests die niet worden getoond. Er is gekozen om de “TwinfieldServiceTest” uit te lichten omdat dit één van de langere tests was en van een cruciaal onderdeel. Tevens bevat deze test alle aspecten die van de test suite gebruikt worden.

```
public class TwinfieldServiceTest {
    protected TwinfieldService twinfieldService;
    TwinfieldMapper<TwinfieldBooking> mapper;
    @Injectable protected TwinfieldTemplate twinfieldTemplate;
    protected TwinfieldResult<Browse> twinfieldResult;

    @Before
    public void initTwinfieldService(){
        setupTwinfieldResult();
        new NonStrictExpectations(){{
            twinfieldTemplate.execute((BrowseQuery) any); result = twinfieldResult;
        }};
        twinfieldService = new TwinfieldService();
        Deencapsulation.setField(twinfieldService, "twinfieldTemplate", twinfieldTemplate);
        Deencapsulation.setField(twinfieldService, "mapper", mapper);
        Deencapsulation.setField(twinfieldService, "result", twinfieldResult);
    }
```

Figuur 10: TwinfieldServiceTest test initialisatie

In bovenstaande figuur bij de lijst met variabelen staat @Injectable als annotatie voor het twinfieldTemplate object. Deze annotatie maakt van dit object een mock object dat in de te testen klassen kan worden geïnjecteerd. Voor de TwinfieldMapper en TwinfieldResult worden echte objecten gebruikt omdat deze geen contact hebben naar buiten de applicatie en op deze manier direct worden mee getest.

De testen dienen eerst geïnitialiseerd te worden voordat deze gestart kan worden. Deze methode wordt gemarkeerd met de @Before annotatie zodat deze voor iedere test wordt uitgevoerd. Binnen de initialisatie worden de return objecten voorbereid dat gebeurd in deze test door de methode setupTwinfieldResult() aan te roepen. Dit return object wordt als resultaat voor een mock call meegegeven.

Dit gebeurt in deze test binnen new NonStrictExpectations(){{}}, op het moment dat er een execute wordt aangeroepen op het twinfieldTemplate object met een Browsequery object als parameter dan zal het Twinfield resultaat door de mock worden teruggegeven.

Nu de mock objecten ook functionaliteit hebben en bij aanroep het gewenste resultaat returnen moet het te testen object nog worden aangemaakt met new TwinfieldService(). Doordat de waardes private zijn binnen de TwinfieldService klasse moeten deze geforceerd worden geïnjecteerd in het te testen object. Dat gebeurd via Deencapsulation.setField() waarmee de privé variabelen dus gezet worden. Dit voorkomt dat productie source moet worden aangepast om Mocking mogelijk te maken,

```
protected void setupTwinfieldResult(){
    Browse rows = new Browse();
    List<BrowseRow> browseRowList = new ArrayList<~>();
    BrowseRow browseRow = new BrowseRow();
    List<BrowseFieldData> browseFieldDataList = new ArrayList<~>();
    browseFieldDataList.add(setTwinfieldBrowseFieldData("fin.trs.head.yearperiod", "2014/07"));
    browseFieldDataList.add(setTwinfieldBrowseFieldData("fin.trs.line.valuesigned", "8400.00"));
    browseFieldDataList.add(setTwinfieldBrowseFieldData("fin.trs.head.date", "20140707"));
    browseFieldDataList.add(setTwinfieldBrowseFieldData("fin.trs.head.number", ""));
    browseFieldDataList.add(setTwinfieldBrowseFieldData("fin.trs.line.invnumber", ""));
    browseRow.setFields(browseFieldDataList);
    browseRowList.add(browseRow);
    rows.setRows(browseRowList);
    twinfieldResult = new TwinfieldResult<Browse>(null ,rows);
}

private BrowseFieldData setTwinfieldBrowseFieldData(String twinfieldCode, String value) {
    BrowseFieldData browseFieldData = new BrowseFieldData();
    browseFieldData.setFieldCode(twinfieldCode);
    browseFieldData.setValue(value);
    return browseFieldData;
}
```

Figuur 11: TwinfieldServiceTest test resultaten voorbereiden

Het resultaat dat wordt voorbereid via setupTwinfieldResult() moet helemaal worden nagebouwd zoals het resultaat object dat normaal uit de twinfieldTemplate.execute() methode komt. In bovenstaande figuur is te zien hoe voor deze test het resultaat van het mock object wordt gebouwd.

Nadat bij de initialisatie alles is opgezet wordt de methode met @Test uitgevoerd die in het figuur op de volgende pagina te zien is. Eerst wordt de te testen methoden van de te testen klassen echt uitgevoerd en het return resultaat wordt opgeslagen. In dit geval geeft de methode een lijst met TwinfieldBooking objecten terug.

Daarna worden verificaties uitgevoerd om te kijken of bepaalde dingen zijn uitgevoerd zoals verwacht. Met de methode “assertEquals(Expectedvalue, value)” methode kan worden vergeleken of wat eruit gelijk is aan wat verwacht wordt (assertEquals is een Junit methode). Omdat voor iedere Colomn 4 assertEquals() worden uitgevoerd is dit in eigen methodes ondergebracht. De test zal alleen slagen als alle assertEquals correct zijn.

```
@Test
public void testGetBookingsByDate() throws IllegalAccessException, InstantiationException {
    final List<TwinfieldBooking> twinfieldBookingList = twinfieldService.getBookingsByDate(2014,7,"8000","8050");
    new Verifications(){{
        TwinfieldQuery<BrowseQuery> query;
        twinfieldTemplate.execute(query = withCapture());
        Columns columns = Deencapsulation.getField(query, "columns");
        assertColumn(columns.getColumns().get(0), "fin.trs.line.dim1", "between", "8000", "8050");
        assertColumn(columns.getColumns().get(1), "fin.trs.line.dim1name", "none");
        assertColumn(columns.getColumns().get(2), "fin.trs.head.code", "none");
        assertColumn(columns.getColumns().get(3), "fin.trs.head.status", "none");
        assertColumn(columns.getColumns().get(4), "fin.trs.head.shortname", "none");
        assertColumn(columns.getColumns().get(5), "fin.trs.head.number", "none");
        assertColumn(columns.getColumns().get(6), "fin.trs.line.invnumber", "none");
        assertColumn(columns.getColumns().get(7), "fin.trs.head.date", "none");
        assertColumn(columns.getColumns().get(8), "fin.trs.head.yearperiod", "between", "2014/7", "2014/7");
        assertColumn(columns.getColumns().get(9), "fin.trs.line.valuesigned", "none");
        assertColumn(columns.getColumns().get(10), "fin.trs.line.description", "none");
        Sort sort = columns.getSorts().get(0);
        assertEquals("ascending", sort.getOrder());
        assertEquals("fin.trs.line.invnumber", sort.getField());
        assertEquals("2014/07", twinfieldBookingList.get(0).getBookingPeriod());
    }};
}
```

Figuur 12: TwinfieldServiceTest test uitvoeren

6.7. Reflecteren van het proces

Testen duurde wat langer omdat het voor heel sprint 1 met terugwerkende kracht gedaan moest worden en Junit/Jmockit waren nog onbekend terrein. In het begin was het nog lastig te begrijpen hoe de tests met Jmockit en Junit werkten. Deze twee modules moeten door elkaar heen gebruikt worden en de syntax is niet altijd even duidelijk. De eerste paar dagen ging het percentage maar langzaam omhoog, maar na lang proberen en jmockit en junit documentatie doorlezen werd wel duidelijk hoe het werkte.

Het ophalen van de data uit Twinfield ging een stuk soepeler dan verwacht. Dit kwam mede door het uitvinden van een goede manier om te debuggen. Waarbij alleen de classes die veranderd zijn worden verwisseld met de oude zonder dat een redeploy of server restart nodig is. Zo konden de calls steeds een klein beetje worden aangepast en direct opnieuw getest.

Ik ben vooral tevreden over de eigen annotaties en een mapper voor het schoner maken van de backend. Dit was op zich niet strikt noodzakelijk maar maakte de applicatie wel beter en was leerzaam om uit te voeren.

7. Sprint 3

7.1. Vaststellen van de requirements

In de eerste twee sprints werd alle informatie bij iedere aanvraag opnieuw opgehaald bij de Twinfield en Postduif API's. Voor sprint 3 zal er een database worden aangemaakt waarin de informatie opgeslagen blijft. Op deze manier zal de applicatie nog sneller werken en minder afhankelijk zijn van de externe systemen. Zo kunnen bijvoorbeeld de conceptboekingen (die ook deze sprint worden toegevoegd) gewoon worden voorbereid als Twinfield of Postduif offline zijn en verdwijnt de vertraging die door de API calls naar de externe systemen ontstaat.

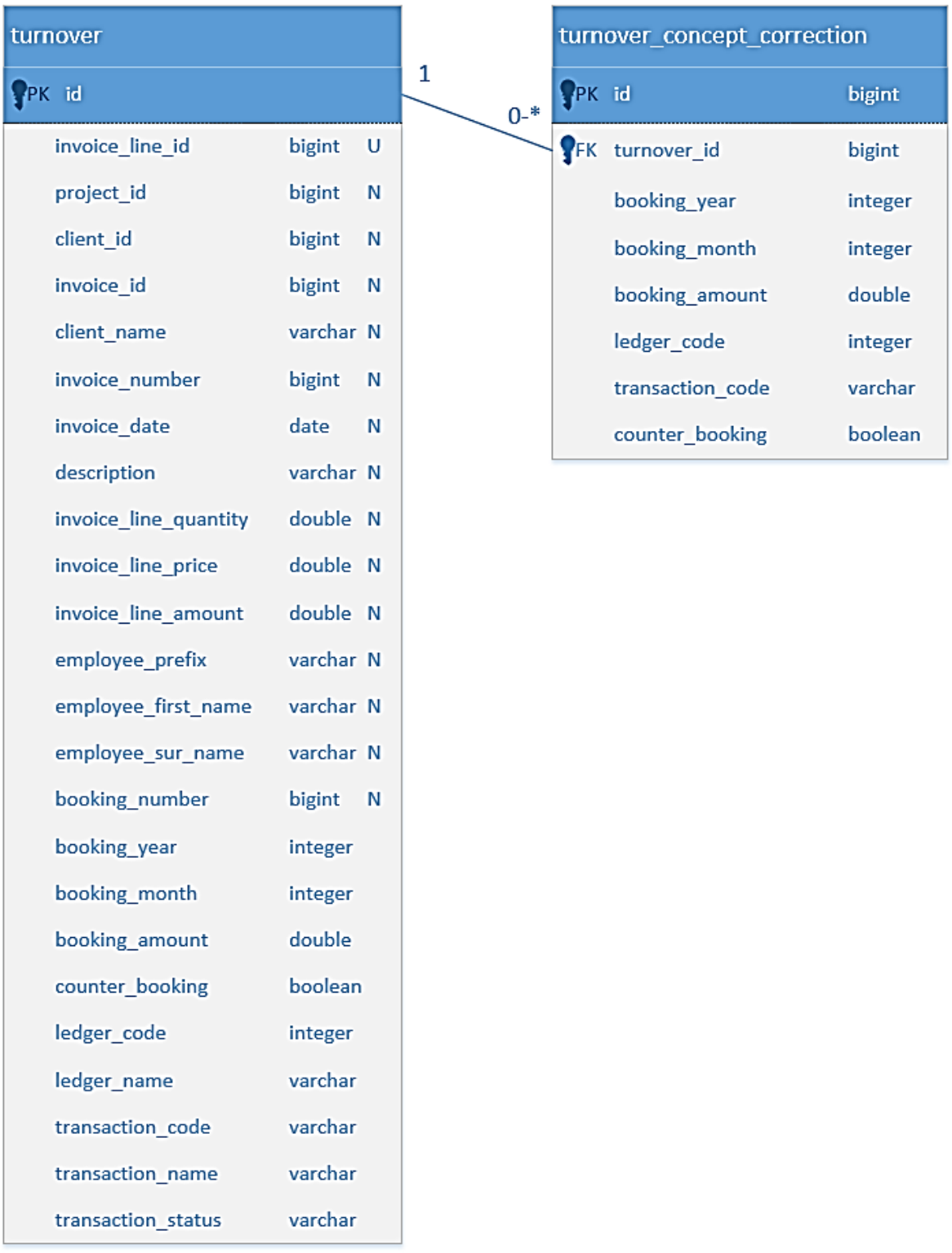
Omdat de boekingen toch niet meer veranderden en er kosten zijn verbonden aan de data transfer vanaf het externe boekhoudsysteem zal alles voor oudere periodes éénmalig worden opgehaald en opgeslagen. Facturen en boekingen horen namelijk bij elkaar en kunnen niet meer worden gewijzigd als ze definitief zijn. Voor correcties moeten dus nieuwe correctieboekingen of nieuwe facturen worden aangemaakt.

7.2. Ontwerpen van de database

Het ontwerp voor de eerste versie van de database omvat twee tabellen en is te zien in het figuur op de volgende pagina. De eerste tabel heet "turnover" en representeert een regel in de omzet sheet. Elke regel binnen de tabel bevat een koppeling van gegevens uit Twinfield en Postduif. Voor memoriaal boekingen zijn er echter geen gegevens uit Postduif dus alle kolommen met Postduif gegevens kunnen leeg zijn (aangegeven met de N achter de kolomnaam in het database model).

De enige manier om de boekingsregels te matchen met bijbehorende invoice regel is op basis van factuurnummer de factuurregels van die factuur op te halen. En daarna op bedrag van de boekingsregel de juiste invoice regel te koppelen. Maar als een factuur dus meerdere regels bevat met hetzelfde bedrag dan zouden regels dubbel gematched kunnen worden en andere helemaal niet meegenomen. Daarom is invoice line id uniek gemaakt (**U** markering in model) om te forceren dat een invoice line niet twee keer met verschillende boekingsregels wordt gematched.

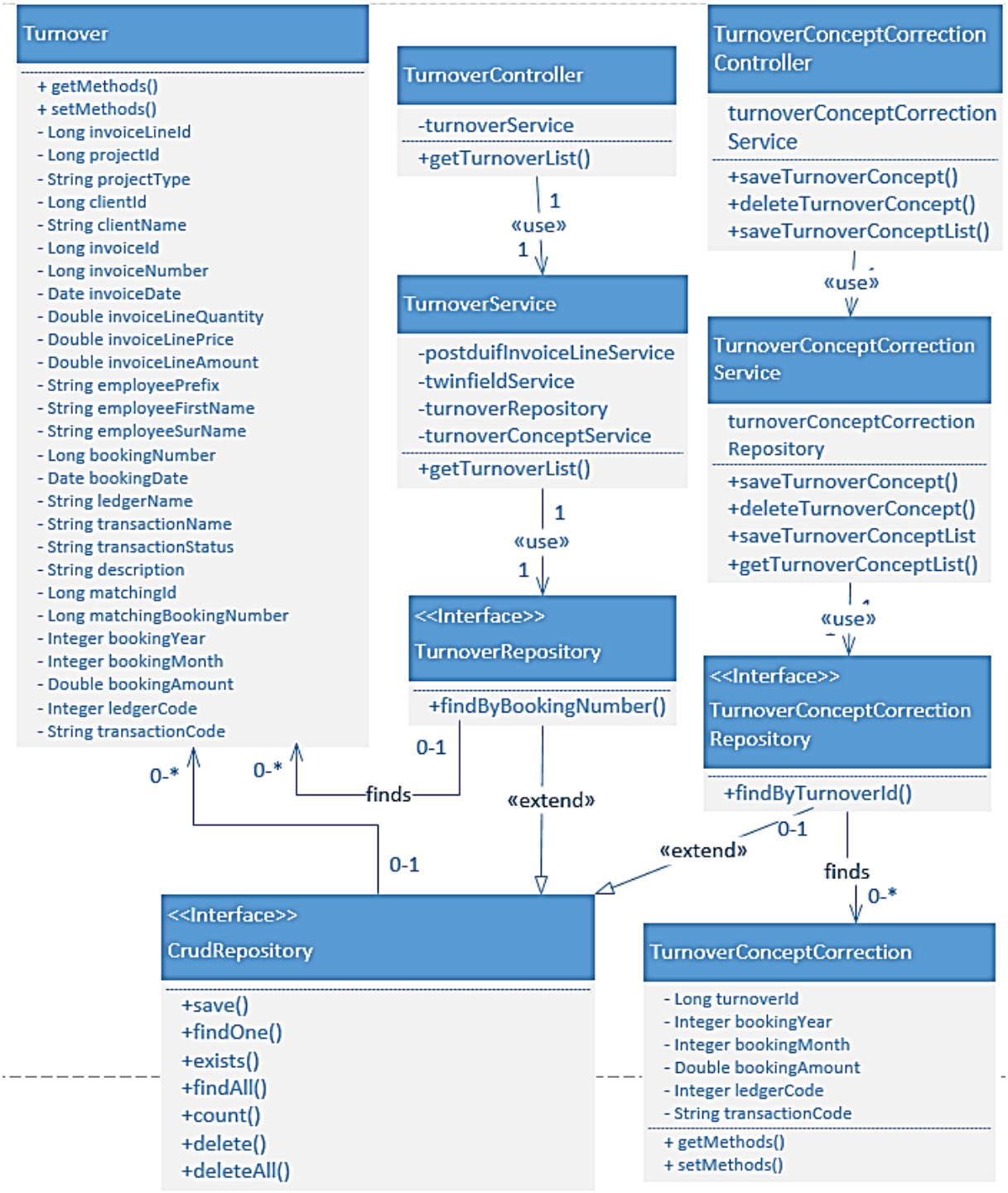
De tweede tabel bevat de conceptboekingen die zullen worden aangemaakt via Condor. Deze conceptboekingen zijn nog niet in Twinfield doorgevoerd maar worden wel meegerekend in de berekeningen van de totale omzet. De conceptboekingen zijn gelinkt aan een turnover regel via het id veld. Deze conceptboekingen kunnen vrijblijvend worden gewijzigd en op een later moment definitief gemaakt door ze naar Twinfield weg te boekingen. Het wegboeken naar Twinfield en het opslaan van de definitieve correctieboekingen zal in een latere sprint worden toegevoegd. Voor deze sprint worden alleen nog concepten opgeslagen.



Figuur 13: Database model van database versie 1

7.3. Ontwerpen van de systeemonderdelen

Onderstaand klassendiagram toont de systeemonderdelen die in deze sprint worden ontwikkeld. Er zal worden gewerkt aan de concept correcties en omzet gegevens. Deze zullen in de database worden opgeslagen via repository interfaces. Deze maken gebruik van de CrudRepository interface dat onderdeel is van Spring data. Waarom en hoe dit wordt gebruikt zal in de volgende paragrafen duidelijk worden.



Figuur 14: Deel klassendiagram sprint 3

7.4. Bouwen eigen database

Op basis van dit ontwerp is een migratiescript geschreven (een deel hiervan is te zien in onderstaand figuur) dat automatisch bij het uitvoeren van de applicatie uitgevoerd zal worden. Hiervoor wordt gebruik gemaakt van Liquibase omdat dit universeel is en mocht er van database veranderd worden kan door alleen het wijzigen van de vertaaldriver het script ook tabellen aanmaken in alle andere ondersteunde databases. De belangrijkste reden om Liquibase te gebruiken is om refactoring te introduceren in het domein van de database.

Daarnaast wordt er automatisch een changelog bijgehouden en als er een nieuw script met wijzigingen of toevoegingen wordt toegevoegd aan de changelog map wordt deze automatisch uitgevoerd en de database en changelog worden bijgewerkt.

```
databaseChangeLog() {
    changeSet(author: "lennart.kramer@42.nl", id: "1") {
        comment("Initialize the database")
        createTable(tableName: "turnover") {...}
        addUniqueConstraint(tableName: "turnover", columnNames: "invoice_line_id", constraintName: "unq_turnover_invoice_line_id")

        createTable(tableName: "turnover_concept") {
            column(autoIncrement: true, name: "id", type: "BIGINT") {
                constraints(nullable: false, primaryKey: true, primaryKeyName: "pk_turnover_concept")
            }
            column(name: "turnover_id", type: "BIGINT") {
                constraints(nullable: false, foreignKeyName: "fk_turnover_concept_turnover_id_turnover_id", references: "turnover(id)")
            }
            column(name: "booking_year", type: "INT") { constraints(nullable: false) }
            column(name: "booking_month", type: "INT") { constraints(nullable: false) }
            column(name: "booking_amount", type: "DOUBLE") { constraints(nullable: false) }
            column(name: "ledger_code", type: "INT") { constraints(nullable: false) }
            column(name: "transaction_code", type: "VARCHAR(255)") { constraints(nullable: false) }
        }
    }
}
```

Figuur 15: Migratie script 001 voor aanmaken eerste tabellen

De door het migratiescript aangemaakte tabellen zullen via een repository klasse door het programma benaderd worden. Zoals in sprint 0 vernoemd is het werken met Hibernate / JPA voor het mappen van de database tabellen naar de objecten van het programma een vereiste. Het meeste werk zit in het éénmalig instellen. Als daarna de conventies worden gevolgd is het vrij eenvoudig uit te breiden.

De repository klassen zullen gebruik maken van de CrudRepository interface van het Spring framework zoals te zien is in het klassendiagram aan het begin van dit hoofdstuk. Deze interface biedt een aantal generieke methodes aan die in de meeste repositories gebruikt worden. Zoals het opslaan, wijzigen, ophalen en verwijderen van één object of een lijst van objecten.

Naast deze standaard methoden kunnen ook nog extra toevoegingen aan de repository worden gemaakt. Dit kan op basis van methode naam en door een implementatie klasse van de interface te maken om een eigen query te schrijven.

Bij de op basis van methode naam gegenereerde query's zal een parser de methode naam uitlezen en op basis hiervan een query genereren. Er is gekozen voor Spring data omdat hiermee heel gemakkelijk de standaard query's kunnen worden gegeneerd.

Voor nu zijn de standaard methodes van CrudRepository en de op basis van methode naam gegenereerde query's afdoende. CrudRepository hoeft alleen maar te weten welke klasse gebruikt

wordt en het data type van de primaire key van deze klasse. Er hoeft dus voor alle standaard handelingen geen enkele regel implementatie geschreven te worden.

De database zal handmatig geupdate moeten worden na het versturen van nieuwe facturen, correcties zullen automatisch worden toegevoegd.

7.5. Bouwen van concept correcties onderdeel

Al het voorbereidende werk wordt in de frontend gedaan en dan naar de backend API toegestuurd. De backend controller zal vervolgens deze gegevens doorgeven aan de juiste service methode. Deze service methode gebruikt de repository om de toevoeging, wijziging of verwijdering op te slaan.

De werking van het splits systeem is in de frontend een aantal keer veranderd doordat niet helemaal duidelijk was wat nu het handigst was. Bij het splitsen werd eerst het bedrag gelijkmatig verdeeld en er steeds een maand bij de boekingsmaand opgeteld. Dit was gedaan omdat ik in de veronderstelling was dat er vooral veel gesplitst werd over toekomstige periode.

Onderstaand figuur toont een voorbeeld van het splits systeem waarbij de boeking over drie maanden gelijkmatig verdeeld wordt.

Figuur 16: Frontend ui met confirm message

7.6. Testen van de systeemonderdelen

Doordat in sprint 3 de gegevens niet meer alleen direct door worden gegeven naar de frontend maar ook bewaard blijven in de eigen database moesten de modellen toch nog wel behoorlijk aangepast. Niet alleen de direct in de sheet benodigde informatie wordt bewaard maar ook nog wat koppelingen naar andere objecten die later van waarde zullen zijn. Zoals de link naar een project id, klant id en invoice id.

7.7. Reflecteren van het proces

Weer een goede sprint geweest, alle goals gehaald. Ontwerp database blijkt te werken. CrudRepository werkt heel handig, configuratie goed krijgen was even wat werk maar daarna verliep alles vlotjes. Database vullen met gegevens was eventjes werk om het script helemaal perfect te krijgen maar weinig problemen.

Splits functie redelijk goed gelukt en ziet er goed uit maar kan nog wel iets beter, er wordt ook vaak één periode terug geboekt en dat is in de huidige setup veel werk. Kwam naar boven bij demonstratie van sprint 3. De split functie moet dus later nog iets aangepast worden zodat het minder werk is om bepaalde standaardhandelingen vaker uit te voeren.

8. Sprint 4

8.1. Vaststellen van de requirements

Hoofd onderdeel van sprint 4 zal het verwerken van de concepten zijn en hier definitieve correcties van te maken. De product owner heeft de wens uitgesproken dat voor deze definitieve boeking doorgevoerd wordt er eerst een confirmatie scherm wordt getoond. In dit confirmatie scherm zijn dan de journaalposten te zien die aangemaakt zullen gaan worden. Als daarna voor definitief boeken gekozen wordt worden deze voorbereide journaalposten via de API van Twinfield in de boekhouding toegevoegd.

Daarnaast is het de bedoeling dat andere developers makkelijker lokaal kunnen ontwikkelen. Om dit te bewerkstelligen zal een in memory database worden gemaakt met test data. Daardoor hoeft voor het opzetten van een test / develop omgeving niet eerst een PostgreSQL database opgezet te worden.

Ook zal het project worden toegevoegd aan Bamboo en Sonar, dit zijn twee programma's in de ontwikkelstraat van 42. In Bamboo wordt automatisch bij elke push naar de Stash (git) repository een nieuwe build gedraaid om te kijken of er build errors ontstaan. Daarbij wordt ook een rapport voor Sonar gegenereerd waarmee gezien kan worden of aan de programmeerstandaard voldaan wordt.

Als laatste zal er security aan de frontend en backend worden toegevoegd zodat mensen die niet zijn ingelogd niet meer bij de gegevens kunnen. Er is met de product owner afgesproken dat er twee rollen komen, een user en een admin. De user mag alleen gegevens lezen en de admin mag alle acties uitvoeren zoals wijzigen en verwijderen van bepaalde dingen. Tot slot wordt er een opzet voor het dashboard toegevoegd, hier hoeven nog geen echte gegevens gebruikt te worden.

8.2. Ontwerpen van de database

Op basis van de concept correcties worden journaalposten toegevoegd aan Twinfield en veranderen alle concepten van die periode in correcties. Deze correcties krijgen een eigen tabel in de database en ook een eigen repository, model, service en controller in Condor. Hiervoor wordt een tweede migratie script geschreven dat de extra tabel toevoegt aan de database.

Met dit migratie script kan iedereen die versie 1 had van de database automatisch updaten naar versie 2 bij het starten van het programma. Als iemand nog geen database had zal vanaf nu automatisch versie 2 worden aangemaakt. Dit toont ook direct de kracht van het gebruik van Liquibase migratie scripts.

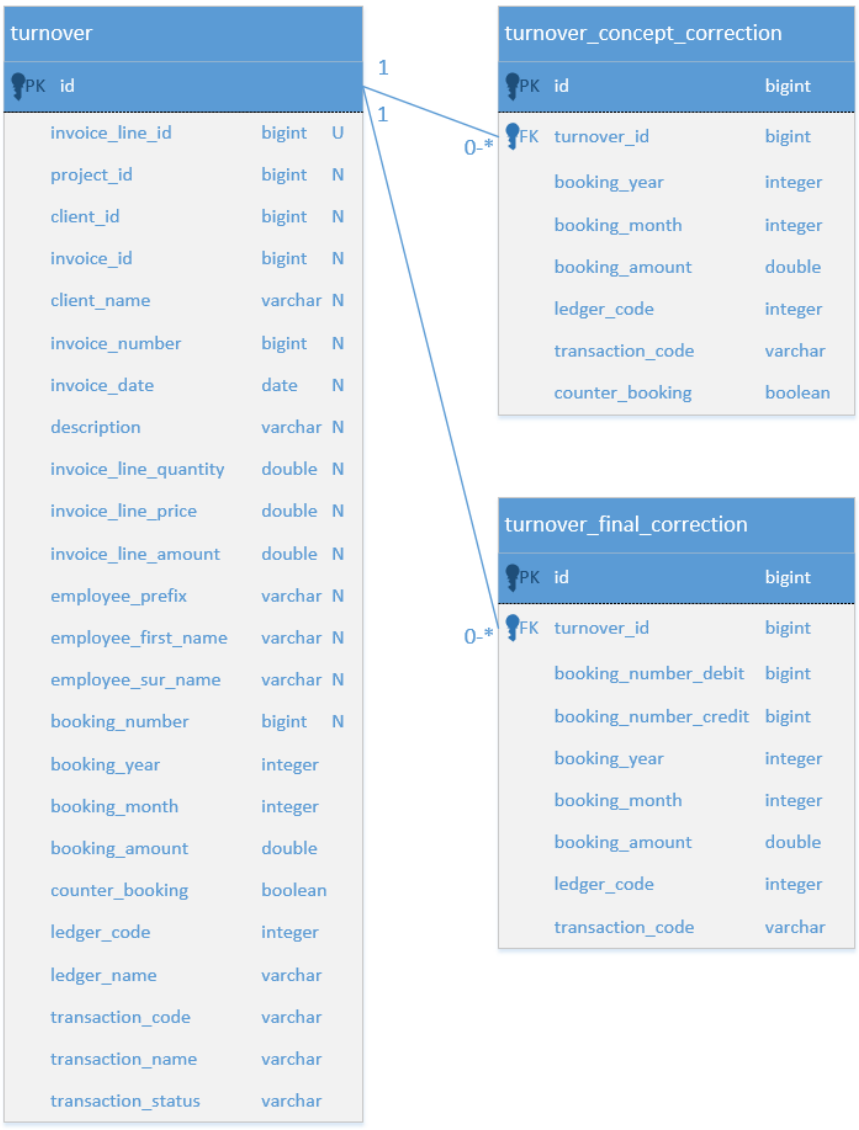
De nieuw toegevoegde turnover correction tabel is op drie manieren gekoppeld aan de turnover tabel. Zo zijn ze verbonden via de turnover waarop de correctie plaatsvind door middel van turnover_id, het memoriaalstuk waar wordt afgeboekt en het memoriaal stuk waar het wordt bijgeboekt via booking number debit en credit.

Zoals misschien opvalt in onderstaand model is dat de drie tabellen allemaal een aantal overeenkomende kolommen hebben. Deze zijn niet in een turnover details tabel gehuisvest vanwege een aantal redenen:

- Deze er steeds bij gejoined had moeten worden.
- De database niet onnodig complex maken voor volgende programmeurs.
- Het ook in het programma netjes opgelost kan worden.

Alle query's naar één van de drie tabellen krijgen dan extra joins en als de turnover moet worden opgehaald met zijn details erbij en alle concepten en correcties en daarvan ook de details erbij gejoined dan krijg je enorm veel extra joins.

Als alle dubbele velden in gemeenschappelijke tabellen worden weg gestopt dan wordt het een aaneenschakeling van id's. Dan kan er in het programma niet meer gebruik gemaakt worden van de query's op basis van naam en moet worden overgestopt naar handmatige query's. Het zal dus uiteindelijk tijd besparen en kan ook worden opgelost in de code van het programma, zie volgende pagina.

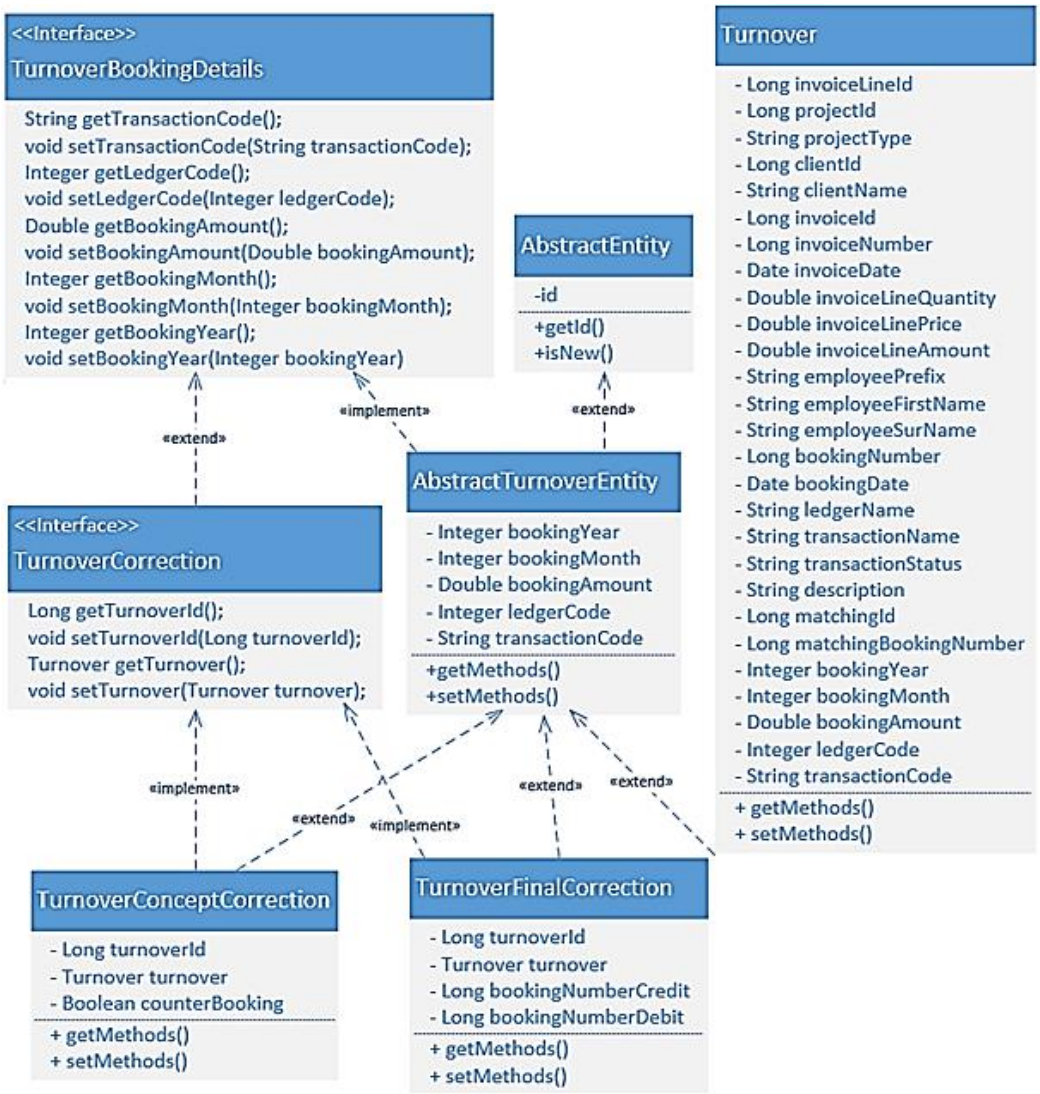


Figuur 17: Database model versie 2 met correcties

8.3. Ontwerpen van de systeem onderdelen

Het ophalen, wegschrijven en omzetten van en naar Json is een stuk makkelijker en schoner door de database niet te complex te maken. Maar in het programma is het niet de bedoeling dat er ook dubbele code komt vanwege het versimpelen van de database. Daarom wordt er een abstracte turnover klasse en een booking details interface in het leven geroepen om alle overeenkomende variabelen te huisvesten.

Het klassendiagram hiervan is te zien op onderstaande afbeelding. Deze abstract turnover entity krijgt in de code de @MappedSuperClass annotatie. Daarmee weet Hibernate dat deze velden ook in de tabel gebruikt worden en worden dus opgenomen in het meta model. Als er iets moet worden gedaan met booking details dan kan vanaf nu de interface gebruikt worden.

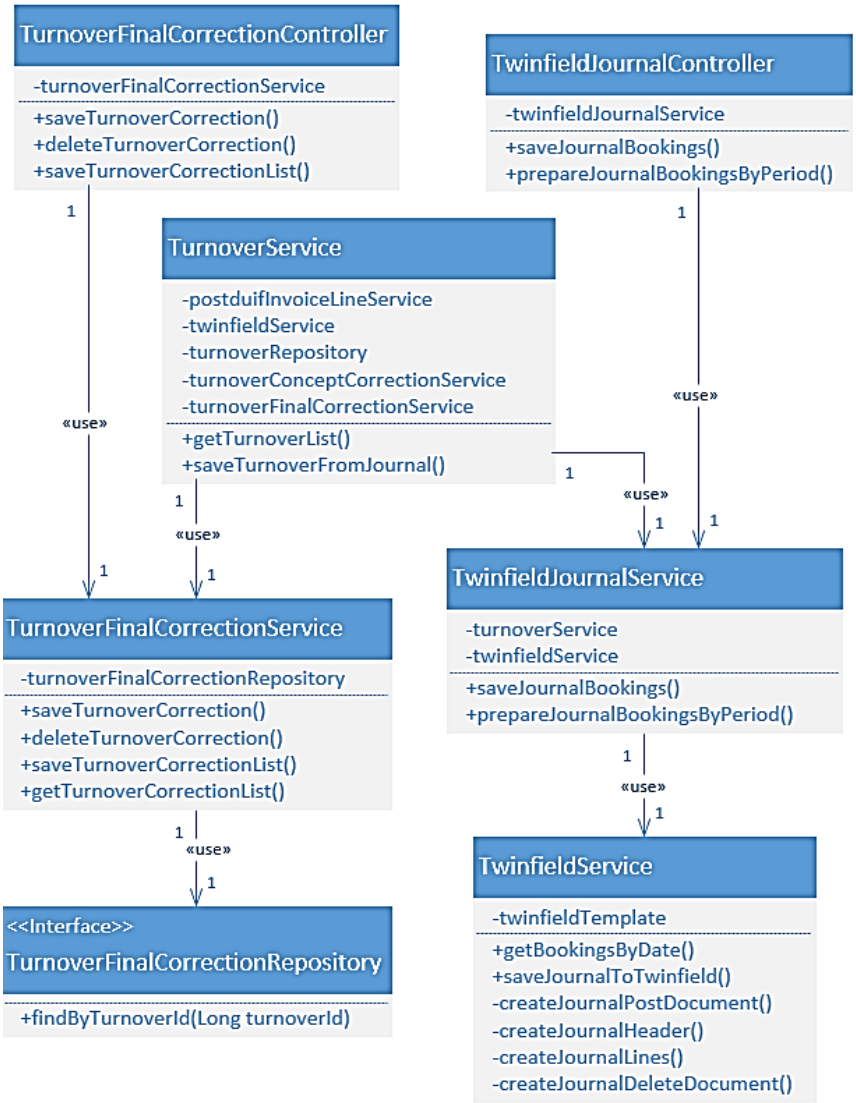


Figuur 18: Klassenstructuur turnover modellen

Het klassendiagram op deze pagina laat de klassen voor sprint 4 zien. Aan de rechterkant van het model staan alle klassen die te maken hebben met Twinfield interactie (deze klassen beginnen allen met het woord Twinfield). De Twinfield service klasse is de klasse die met de buitenwereld verbinding heeft via de TwinfieldTemplate (een klasse vanuit de Twinfield Api).

Aan de linkerkant van het diagram is een deel te zien van het Turnover gedeelte van deze sprint. Nadat de Twinfield service de correcties heeft opgeslagen worden hier Twinfield Journal boekingen van gemaakt. Vanuit deze Journals zal de Turnover service een boekingsregel toevoegen (dit is de memoriaal boeking die bij de correctie hoort, deze boeking moet ook terug te zien zijn in het overzicht).

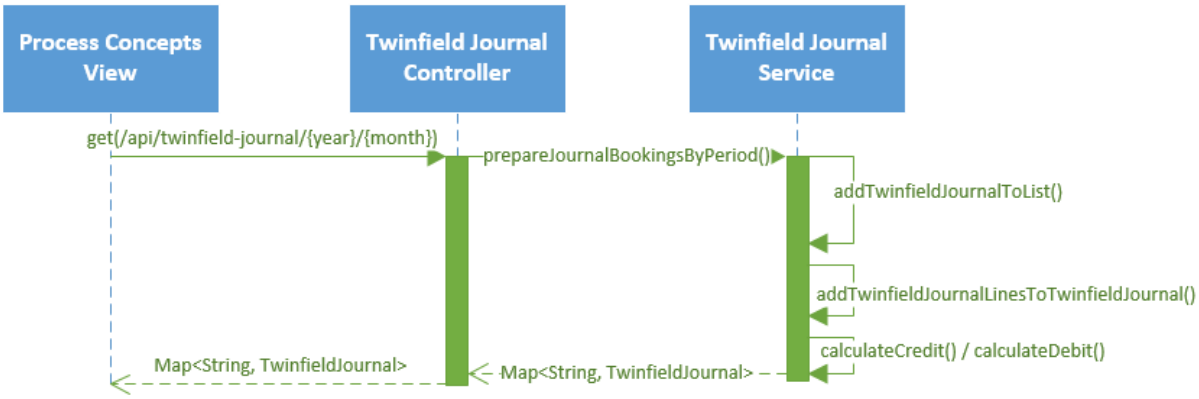
Daarnaast zal de Turnover Service de turnover final correction service gebruiken om de Turnover definitieve correcties aan te maken en op te slaan in de database. De concept correcties zullen na het opslaan van de definitieve correcties ook nog verwijderd worden (niet te zien in het klassendiagram).



Figuur 19: klassendiagram sprint 4

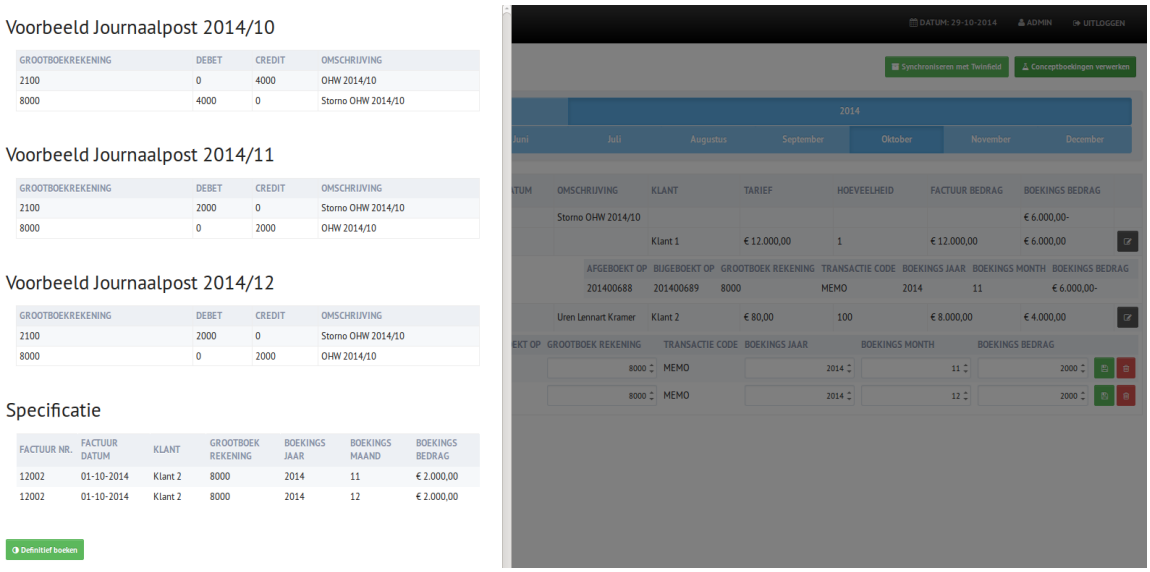
8.4. Bouwen onderdeel concepten wegboeken naar Twinfield

Een wens vanuit de opdrachtgever was om eerst een confirmatie scherm te zien te krijgen. In dit scherm is te zien welke journaalposten er aangemaakt zullen worden. Daarnaast is het overzicht nog zichtbaar om te kunnen controleren of alles klopt. De journaalposten worden in de backend voorbereid omdat hier veel berekeningen voor nodig zijn en de frontend op deze manier geen onnodige logica bevat.



Figuur 20: Sequentiediagram voorbereiden journaalposten voor wegboeken concepten

Bovenstaand sequentiediagram laat de stappen zien van het voorbereiden van de journaalposten, onderstaand figuur toont het confirmatiescherm met de voorbereide voorbeeld journaalposten. Op dit moment is er nog niks echt naar het boekhoudprogramma verstuurd.


Voorbeeld Journaalpost 2014/10

GROOTBOEKREKENING	DEBET	CREDIT	OMSCHRIJVING
2100	0	4000	OHW 2014/10
8000	4000	0	Storno OHW 2014/10

Voorbeeld Journaalpost 2014/11

GROOTBOEKREKENING	DEBET	CREDIT	OMSCHRIJVING
2100	2000	0	Storno OHW 2014/10
8000	0	2000	OHW 2014/10

Voorbeeld Journaalpost 2014/12

GROOTBOEKREKENING	DEBET	CREDIT	OMSCHRIJVING
2100	2000	0	Storno OHW 2014/10
8000	0	2000	OHW 2014/10

Specificatie

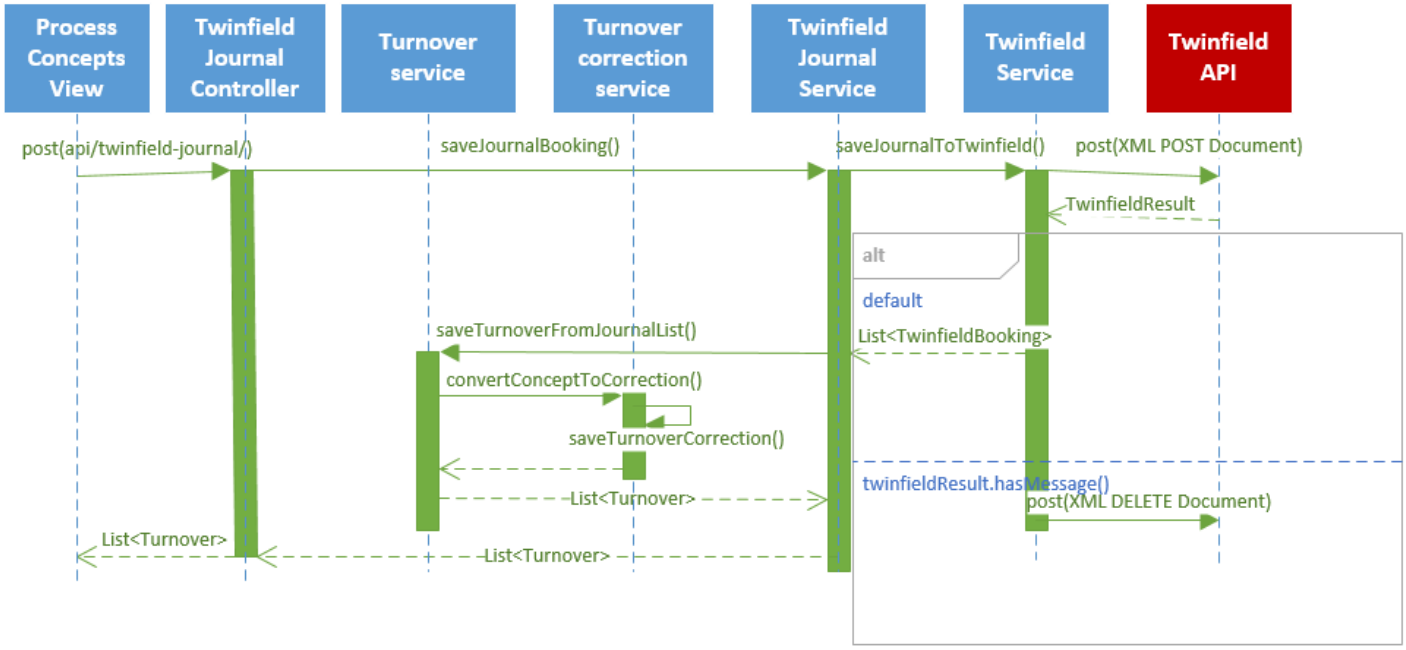
FACTUUR NR.	FACTUUR DATUM	KLANT	GROOTBOEK REKENING	BOEKINGS JAAR	BOEKINGS MAAND	BOEKINGS BEDRAG
12002	01-10-2014	Klant 2	8000	2014	11	€ 2.000,00
12002	01-10-2014	Klant 2	8000	2014	12	€ 2.000,00

Definitief boeken

Figuur 21: Frontend voorbeeld confirmatiescherm

Onderstaand sequentiediagram toont het daadwerkelijk boeken naar Twinfield waarbij een aantal private methods zijn weg gelaten omdat het anders te ingewikkeld werd.

Als alles klopt en er wordt in het confirmatiescherm gekozen voor definitief boeken dan worden de voorbereide journaalposten naar de backend API gestuurd. Dit is de eerste post naar de API in het sequentiediagram. Vanuit daar worden de journaalposten omgezet naar een XML document omdat de API van het Twinfield boekhoudpakket een XML document verwacht.



Figuur 22: Sequentiediagram wegboeken concepten naar Twinfield

Mocht het Twinfield resultaat error berichten bevatten dan worden de journaalposten weer verwijderd en wordt er een error naar de frontend gestuurd. Mocht deze delete call onverhoopt falen doordat Twinfield precies uitvalt tussen het aanmaken van het concept en verwijderen ervan bij een fout dan zal er een concept achterblijven in Twinfield.

Dit concept zal dan handmatig verwijderd moeten worden in Twinfield. Er was helaas geen mogelijkheid om van meerdere transacties in verschillende perioden een transactie te maken (zodat bij een fout automatisch alles ongedaan wordt gemaakt).

Indien de journaalposten succesvol zijn toegevoegd zullen er turnover objecten voor de aangemaakte journaalposten worden gemaakt. Daarnaast worden alle concepten van die periode omgezet naar correcties waarbij deze worden gekoppeld aan het boekingsnummer waar wordt afgeboekt en het boekingsnummer waar wordt bijgeboekt. Deze correcties en bijbehorende boekingen kunnen niet meer worden aangepast via Condor of via Twinfield.

De laatste eis voor dit onderdeel was het toevoegen van een overzicht per gemaakt memoriaalboekingen. Waarbij alle correcties staan die op deze boeking zijn verwerkt. Dit overzicht moest tevens geprint kunnen worden zodat het in een map bewaard kan worden. Als de account langs komt dan kan deze de overzichten gebruiken.

8.5. Bouwen van in memory database met test data

Voor het opzetten van de in memory database moet vooral de configuratie goed gezet worden. Het maken van de tabellen kan door middel van de migratie scripts die ook zijn gebruikt voor het maken van de vaste database. Daarnaast wordt er een test data builder gemaakt die bij het opstarten van de applicatie met het “test-data” profiel de in memory database vult.

Deze test data kan op meerdere manieren in de in-memory database worden gezet, er kan bijvoorbeeld gekozen worden voor het uitvoeren van SQL bestanden iedere keer dat het programma wordt opgestart. Een andere mogelijkheid is het aanroepen van de repositories. De save methode zal dan de gegevens opslaan in de in memory database in plaats van de echte database.

Er is gekozen om een test data builder te schrijven in Java die de repositories aanroept bij het opstarten van het programma. Deze variant had de voorkeur boven SQL bestanden omdat het beter te controleren is, als een set methode verdwijnt dan wordt er door de IDE direct een melding gegeven. Daarnaast is het aanpassen van de builder in Java gemakkelijker dan de SQL bestanden langs gaan. Op deze manier is er meer zekerheid dat de test data compiletime correct is.

Als de server wordt gestopt dan verdwijnen alle gegevens weer uit de in-memory database. Op deze manier kan er elke keer weer met dezelfde beginsituatie worden gewerkt.

8.6. Testen van de systeemonderdelen

In sprint 4 is er veel in de backend veranderd waardoor de test dekking ineens was gezakt van de 100% naar de 60%. Om het testen vanaf nu beter bij te houden (zodat het rond de 100% blijft) wordt het project toegevoegd aan Bamboo en SonarQube.

Bamboo: Onderdeel van de Atlassian suite waarmee na iedere commit een build gedraaid wordt om te kijken of er fouten in zitten en alle testen nog slagen. Bij het breken van de build kan iemand verantwoordelijkheid nemen en een issue hiervoor aanmaken in JIRA.

Tijdens deze build wordt er een test rapport gegenereerd voor SonarQube. Dit rapport bevat de bevindingen op basis van Rule Compliance (mate van overeenstemming met de opgestelde programmeerregels), code test coverage (mate waarin de code is afgedekt door unit testen), eventuele code duplicatie en de complexiteit. Op basis van deze rapporten wordt het gelijk duidelijk wat de kwaliteit is van de toegevoegde code en reparaties gedaan waar nodig.

SonarQube: Software systeem waarmee inzicht wordt verkregen in de kwaliteitsmetingen van het systeem. Daarnaast controleert het of aan de opgestelde programmeerregels wordt voldaan en zoekt het naar code duplicaties en complexiteit.

Condor ontbrak de eerste sprints nog op Bamboo en Sonar dit kwam doordat het programma verwachtte dat er een database aanwezig was. Zonder deze database kon er geen build gedraaid worden op de Bamboo server.

Nu eerder deze sprint de in memory database is toegevoegd kan het programma ook draaien zonder dat er een PostgreSQL database hoeft te zijn. In de configuratie wordt het zo ingesteld dat er

standaard van de in memory database gebruik wordt gemaakt op de testomgeving. In productie omgeving of als dat specifiek wordt aangegeven kan de PostgreSQL database gebruikt worden.

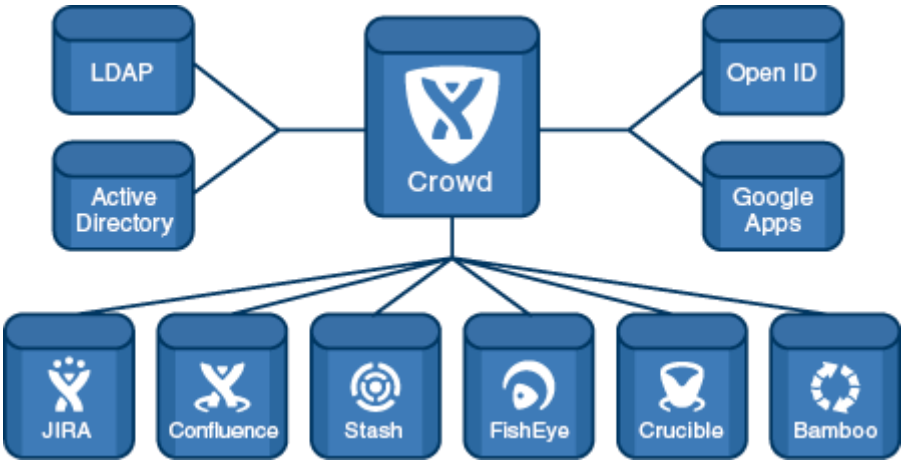
Op moment van toevoegen was de test dekking 60% en de rule compliance was 75%. De test dekking was zo laag gezakt door alle toevoegingen met betrekking tot het wegboeken van de concepten. De rule compliance was nu pas voor het eerst zichtbaar. Op basis hiervan is er de laatste dagen van de sprint gewerkt om de kwaliteit van de applicatie te verbeteren.

Voor het verhogen van de test dekking worden extra unit testen toegevoegd voor de nieuwe methodes die tijdens de sprint ontwikkeld zijn. Voor het verhogen van rule compliance moet je denken aan het toevoegen van JavaDoc (documentatie bij iedere methoden). De correcte formatering van de code (dit is gedaan door het toevoegen van de 42 programmeerformat aan de IDE IntelliJ). Ook veel kleine dingen als witregels aan het einde van bestanden, toevoegen van pakket informatie files (package-info.java bestanden in iedere package) worden meegerekend.

8.7. Bouwen van beveiliging binnen de applicatie

Er zal security aan de frontend en backend worden toegevoegd zodat mensen die niet zijn ingelogd niet meer bij de gegevens kunnen. Daarnaast is er met de opdrachtgever afgesproken dat er twee rollen komen, een user en een admin. De user mag alleen gegevens lezen en de admin mag alle acties uitvoeren zoals wijzigen en verwijderen van bepaalde dingen. Als extra taak in de sprint is het maken van een opzet voor het dashboard, hier hoeven nog geen echte gegevens gebruikt te worden.

Voor security moet gebruik gemaakt worden van Atlassian Crowd, dit is de single sign on software van het Atlassian pakket. Voor alle producten in de Atlassian suite wordt dit gebruikt en kan dus overal met dezelfde gegevens worden ingelogd. De rechten hoeven ook maar op één plek ingesteld te worden. De interne software systemen van 42 maken ook gebruik van diezelfde login gegevens zodat je voor alle interne applicaties dezelfde gegevens kan gebruiken.



Figuur 23: Atlassian Crowd model

In Crowd worden er twee extra rollen toegevoegd Condor User en Condor Admin, door in te loggen krijg je een Crowd token terug. Door deze Crowd-token mee te geven aan de API calls kunnen de rechten worden opgehaald bij Crowd.

Voor het onderscheppen van de API call en het controleren van de rechten wordt Spring Security gebruikt. Spring Security biedt enorm veel verschillende mogelijkheden voor authenticatie en past binnen het gebruikte Spring framework.

De security strategie van 42 berust op de principes:

- Dwingend op de backend.
- Behulpzaam op de frontend.

Met Spring Security wordt aangegeven welke API calls gecontroleerd worden en welke rol er nodig is om de call uit te voeren. Als de gebruiker niet de juiste rol heeft of niet is ingelogd dan krijgt deze een 403 forbidden access error terug.

In de frontend wordt steeds geverifieerd of de gebruiker ingelogd is en welke rollen deze heeft. Bepaalde knoppen waarvan de acties niet mogen worden uitgevoerd worden verborgen. Aangezien dit aan de browser kant gebeurt kan de gebruiker deze weer zichtbaar maken, maar de actie die wordt uitgevoerd door de knop zal niet werken.

8.8. Reflecteren van het proces

Er is in sprint 4 veel toegevoegd maar doordat er in de eerste sprints steeds veel aandacht is besteed aan de uitbreidbaarheid heeft dit niet veel tijd in beslag genomen. Ik ben tevreden over hoe alles is opgezet en geconfigureerd. Door de opzet van Liquibase was het gemakkelijk een aanpassing te maken in de database. Er hoefde hiervoor alleen een extra script te worden toegevoegd en de rest ging vanzelf. Daarnaast is het toevoegen van nieuwe klassen ook geen probleem, als de naming conventions worden nageleefd dan werkt het direct.

Ik ben minder tevreden over het voorbereiden van de journaalposten voordat de boeking naar Twinfield plaatsvind. Dit deed ik eerst allemaal in de frontend en toonde dit in het confirmatiescherm. Daarna stuurde ik deze in de frontend voorbereide journaalpost naar de backend.

Ik had dit gedaan omdat ik dacht dat er niet zoveel logica in zou zitten maar dat bleek tegen te vallen. Daarom heb ik alles in de backend moeten herschrijven en ben ik onnodig tijd kwijt geraakt om deze handelingen eerst in javascript te programmeren. Een volgende keer zou ik dit dus eerder moeten signaleren en sneller naar de backend overstappen. Dit past ook beter in de security strategie van 42.

Ik had iets eerder actie kunnen nemen om het project sneller op Bamboo en Sonar te krijgen door eerder de in memory database toe te voegen. Hiermee had ik sneller de status van mijn project inzichtelijk gehad en het testen beter bij kunnen houden.

9. Sprint 5

9.1. Vaststellen van de requirements

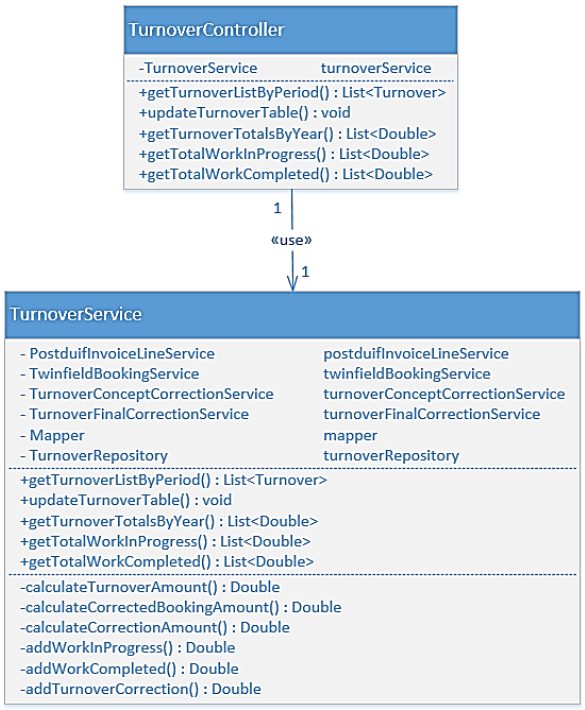
Tijdens de sprint 5 planning meeting was er een nieuwe sheet aangevraagd die als een andere kijk op dezelfde data vernoemd werd. In de praktijk bleek er toch een stuk meer gedaan te moeten worden om deze sheet van data te voorzien. Zo moesten er ook projecten, uren, toewijzingen, roosters worden opgehaald.

Hiervoor waren de eerste backend delen geschreven en een eerste setup gemaakt van de frontend tabel. Toen bleek echter dat dit erg veel werk zou worden om toe te voegen en is er een gesprek aangevraagd met de opdrachtgever en eindgebruiker om het project af te bakenen. In dit gesprek wordt gekeken wat er in de laatste sprints nog zal passen. Er is besloten om meer de diepte in te gaan voor de al bestaande onderdelen. Een volgende afstudeerder zal deze nieuwe sheet ontwikkelen.

Het dashboard zal worden gerealiseerd met echte data op basis van de opzet vanuit de vorige sprint. De splits functie zal verbeterd worden nadat de eisen hiervoor duidelijker zijn geworden. Tot slot wordt het mogelijk gemaakt om concept facturen alvast te boeken op de juiste periode. Deze alvast geboekte concepten moeten later nog wel gematched kunnen worden met definitieve facturen. Dit matchen met definitieve facturen zal in sprint 6 worden toegevoegd.

9.2. Ontwerpen van de systeemonderdelen

Het onderstaande klassendiagram bevat twee klassen, deze twee klassen worden gebruikt om de omzetgegevens te berekenen. Er zijn vooral veel methoden toegevoegd aan de turnover service om alle berekening uit te voeren.



Figuur 24: Klassendiagram deel sprint 5

9.3. Bouwen van het dashboard

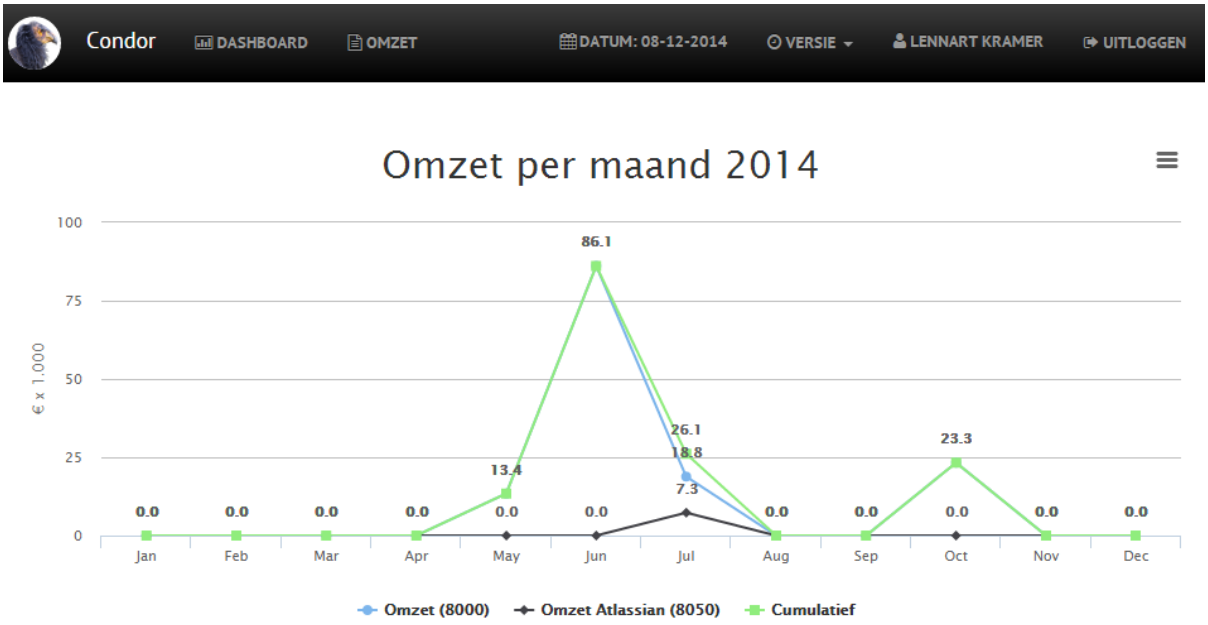
De opdrachtgever wilde graag een scherm waarin hij in één oogopslag kon zien wat de huidige stand was en de trend. De pagina moest veel informatie bevatten maar als er iets geks te zien was in het dashboard vond hij het niet erg om ergens anders heen te moeten om de oorzaak te achterhalen.

Voor het dashboard wordt er gebruik gemaakt van grafieken zoals lijn en kolom grafieken. Omdat het vanaf de grond af bouwen van zulke code veel te lang zou duren wordt er gebruik gemaakt van een externe library. Na een eerste zoektocht op internet bleek er veel keuze op het gebied van frontend Javascript / JQuery grafiek libraries. 9.4.

Daarom is er onderzocht wat andere projecten binnen 42 gebruiken om consistentie tussen de systemen te bewaren. Het bleek dat als er gebruik werd gemaakt van grafieken deze gebruik maakten van de HighCharts library. Deze library wordt gebruikt omdat hij uitgebreid is (bevat veel verschillende grafieken en veel opties per grafiek). De grafieken zijn gemakkelijk aan te passen doordat alles kan worden ingesteld via Json objecten. Daarnaast is de stijl direct passend bij de Bootstrap stijl die door de rest van het programma wordt gebruikt.

Nadat de keuze definitief was gemaakt zijn de eerste twee grafieken met voorbeeld data toegevoegd. En daarna verbeterd door het toevoegen van een Chart service die helpt bij het instellen van de grafiek en het toevoegen van data hieraan. Teksten van de titel, subtitel, legenda, x as en y as teksten, etc. worden ingesteld via deze service en tot slot wordt de grafiek hiermee gebouwd.

Het voorbeeld van het dashboard dat in sprint 4 gemaakt is zal deze sprint worden gevuld met echte gegevens. Hiervoor moeten in de backend wel nieuwe methoden worden aangemaakt om dit goed uit te rekenen. Het totaalbedrag zal worden getoond in de grafiek en er zal in een latere sprint een pagina worden toegevoegd waarin de specificatie te zien is.



Figuur 25: Dashboard omzet grafiek

9.4. Bouwen van de verbeterde Splits functie.

Sinds de meeting einde sprint 3 stond het verbeteren van de splits functionaliteit nog in de back log omdat er in sprint 4 geen plek voor was. Waar bij het splitsen eerst altijd het bedrag gelijkmatig verdeeld werd en er steeds een maand bij de boekingsmaand werd opgeteld worden er nu standaard opties toegevoegd. Het blijft nog wel mogelijk om losse regels toe te voegen maar er kan ook voor uur basis en onderhoud gekozen worden. Bij uur basis wordt het gehele bedrag één maand terug verplaatst. Bij onderhoud wordt het gelijkmatig verdeeld over drie maanden.

Daarnaast wordt er bij het ophalen vanuit de Postduif API nu ook het project type meegegeven. Tijdens het binnenhalen wordt hiernaar gekeken en als het project type gelijk is aan uur basis dan worden de omzetregels automatisch gesplitst met de uur basis standaard. De presets en automatische herkenning zijn beide gedaan om het aantal handelingen voor het splitsen zo veel mogelijk terug te brengen.

In de meeting van sprint 5 kwam naar boven dat er nog een functie ontbrak en dat is het wegboeken van een correctie zonder tegen periode. Dit wordt toegevoegd aan het nieuwe splits menu onder de naam parkeren. Dit is voor het geval er een factuur is gestuurd waarvan de omzet nog niet in een bepaalde periode wordt geboekt. Deze omzet kan dan tijdelijk op de tussenrekening worden geparkeerd.

9.5. Bouwen van alvast boeken Concept factuur

Zoals eerder vernoemd is het de bedoeling om de onderdelen die vooral zorgen tot het verdwijnen van de omzet Excel sheet helemaal af zijn. Een ding wat wel in de Excel sheet is opgenomen maar nog niet in Condor zat is het alvast opnemen van de nog te factureren omzet. Dit is omzet die alvast wordt genomen maar waarvan de factuur nog gestuurd moet worden.

Op het moment dat de factuur daadwerkelijk werd verstuurd dan werd in de Excel sheet het factuurnummer toegevoegd. In Condor moet dit ook mogelijk worden en zal in deze sprint het boeken van de concept factuur mogelijk worden. De volgende sprint zal er worden gewerkt aan het achteraf matchen van de concept factuur met een daadwerkelijk verstuurde factuur.

Omdat deze concept facturen nog niet echt bestaan in Condor (ze worden live opgehaald uit Postduif) kunnen er nog geen correcties op worden aangemaakt. Daarom is het boeken van concept facturen via concept correcties niet mogelijk. Er zullen dus losse methoden en een eigen frontend form gemaakt moeten worden om de concept facturen te kunnen boeken.

9.6. Testen van systeemonderdelen

In sprint 5 is er aan het begin van de sprint gewerkt om eerst de code van sprint 4 te testen waardoor er weer rond de 95% was uitgekomen. Dit zakte in de loop van de sprint weer door de nieuwe toevoegingen. Dit had voorkomen kunnen worden door meer TDD (Test Driven Development) te werken.

Vooral voor alle berekening met betrekking tot de totaalcijfers moest goed getest worden en waren per onderdeel veel verschillende test cases. Zo moest steeds worden gekeken of alles wel goed werd opgeteld en totalen overeenkwamen met de verwachtte uitkomst.

9.7. Reflecteren van het proces

Het is jammer dat er twee dagen verloren zijn gegaan doordat de projectsheet uiteindelijk toch te veel werk bleek te zijn om toe te voegen. In de sprint meeting was het naar voren gekomen als een andere kijk op dezelfde data maar het bleek toch meer te zijn dan dat.

Ik ben wel blij dat het al vrij snel naar boven is gekomen en ik naar aanleiding hiervan een extra meeting heb georganiseerd om het project goed af te bakenen. Anders was misschien pas aan het einde van het project gebleken dat er geen ruimte was om alles helemaal goed op te leveren. Sprint 5 is een beetje een tussen sprint geworden waarin vooral optimalisaties aan bestaande functionaliteit zijn gemaakt en het project is afgebakend.

10. Sprint 6

10.4. Vaststellen van de requirements

Het meeste voor sprint 6 was al besproken in de tussentijdse meeting van het afbakenen. Tijdens de meeting van sprint 6 zijn alleen de onderdelen nog even kort besproken en aan de sprint backlog toegevoegd. Zo moeten de geboekte concept facturen van sprint 5 nog gematched kunnen worden en wordt het nieuw toegevoegde dashboard nog iets geoptimaliseerd.

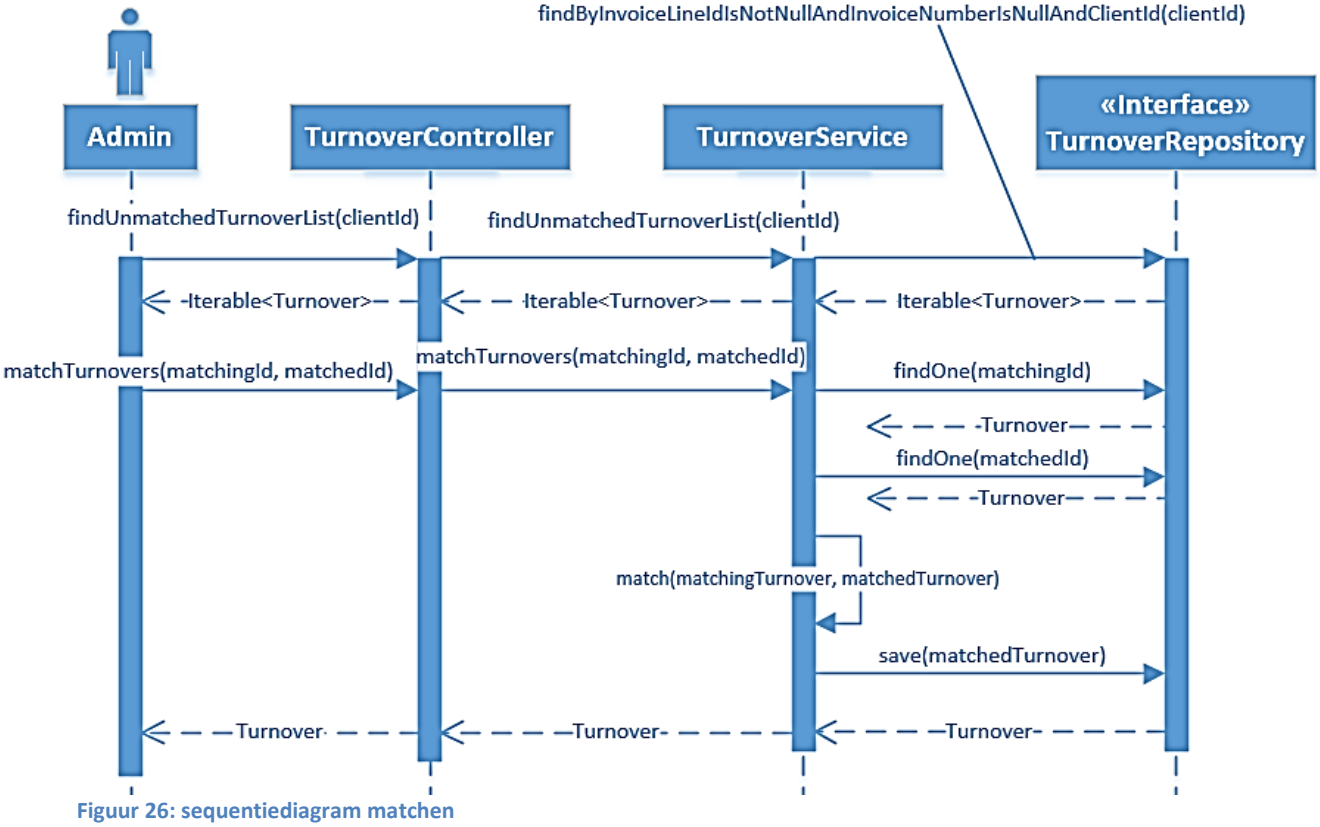
Daarnaast wordt nog een specificatie tabblad toegevoegd. Dit zal als referentie dienen voor de opbouw van de totaalcijfers in de grafieken van het dashboard. De definitieve requirements en use cases zullen worden opgesteld. En op basis daarvan zal er een test plan gemaakt worden voor de gebruikersacceptatie testen. Om te beginnen wordt er deze sprint een checklist gemaakt om alle testcases te identificeren.

10.5. Ontwerpen van de systeemonderdelen

Het laatste systeemonderdeel dat nog ontworpen moest worden is het matchen van de concept facturen. Onderstaand sequentiediagram laat de flow zien van dit systeem onderdeel. Deze bestaat uit twee delen, waarbij eerst een lijst met nog niet gematchde omzet regel wordt opgevraagd.

Als er dan besloten wordt om de matching uit te voeren dan wordt dit naar de backend gestuurd en deze zal de twee regels met elkaar koppelen. Er worden dan wat gegevens van de ene regel opgenomen bij de andere zoals het factuurnummer en de regel waarmee gekoppeld wordt.

Omdat er niet veel spannends is gebeurd bij de bouw zal daar geen paragraaf aan gewijd worden.



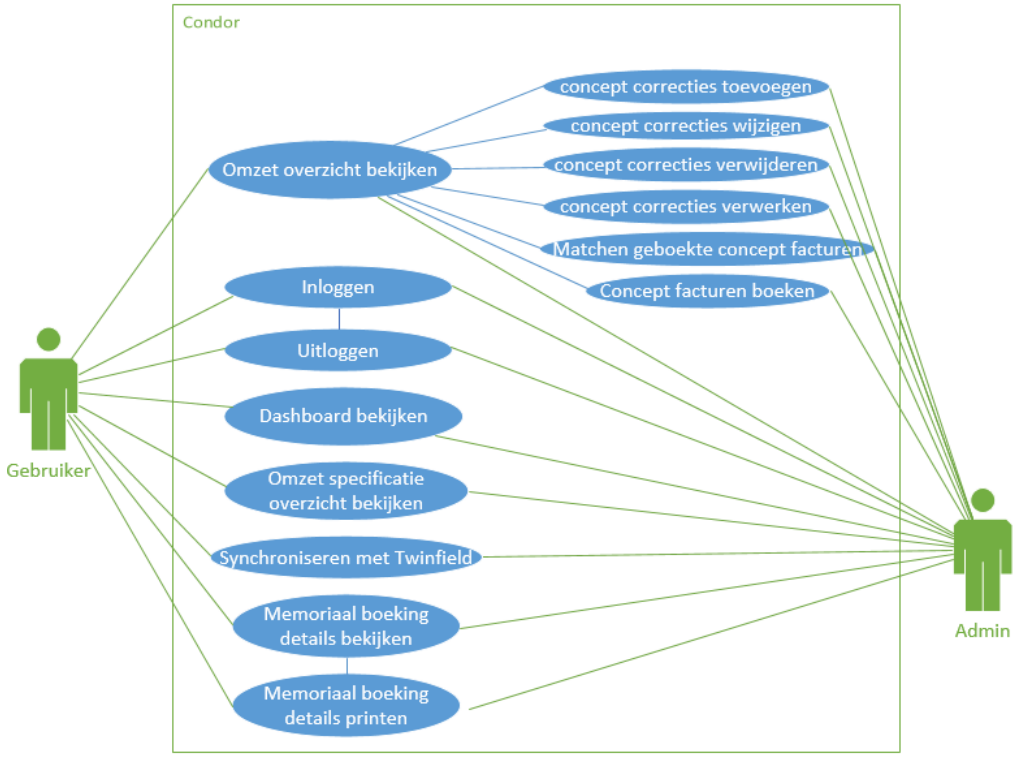
Figuur 26: sequentiediagram matchen

10.6. Opstellen van het Testplan

Voor alle geschreven functionaliteit zullen eerst unit testen worden geschreven zodat de test dekking rond de 100% is. Dit zal dienen als een eerste indicatie dat de geschreven functionaliteit correct functioneert.

Om te bevestigen of alles naar verwachting functioneert zullen er naast de unit testen ook gebruikers acceptatie testen worden uitgevoerd. Om deze acceptatietesten goed te laten verlopen zal eerst een plan worden geschreven waarbij alle use cases getest worden op alle mogelijk uitkomsten.

Op basis van de user stories worden use cases opgesteld. Er zijn twee rollen binnen Condor waarbij de user rol vooral gegevens mag inzien en de admin rol mag alles. Onderstaand figuur toont het use case diagram van Condor.



Figuur 27: Use case diagram

Onderstaande tabel bevat een lijst met de use cases die als test basis dienen voor de scenario's.

UC #	Naam	UC #	Naam
User & Admin use cases		**Admin use cases**	
1.1	Inloggen	8	Concept correcties toevoegen
1.2	Uitloggen	9	Concept correcties wijzigen
2	Dashboard bekijken	10	Concept correcties verwijderen
3	Omzet overzicht bekijken	11	Concept correcties verwerken
4	Omzet specificatie overzicht bekijken	12	Concept facturen boeken
5	Synchroniseren met Twinfield	13	Geboekte concept facturen matchen
6	Memoriaal boeking details bekijken		
7	Memoriaal boeking details printen		

Tabel 4: Use cases van Condor

Voor deze use cases worden dan beschrijvingen gemaakt. Deze beschrijvingen laten de verwachtte functionaliteit zien van het systeemonderdeel. Een voorbeeld van zo'n beschrijving is te zien in onderstaande tabel. Voor een volledig overzicht van alle use case beschrijvingen kunt u kijken in Bijlage C: Requirement document.

Naam	**Inloggen**
Use Case ID	UC 1.1
Omschrijving	De actor gebruikt het systeem om in the loggen.
Actoren	Admin, Gebruiker
Pre condities	1. De actor ziet het login scherm.
Beschrijving	1. De actor vult zijn login gegevens in. 2. Het systeem valideert de login gegevens. [1]
Post condities	1. De actor is ingelogd en ziet het dashboard.
Alternatieve flow	[1] De login gegevens zijn ongeldig, use case stopt.

Tabel 5: Voorbeeld use case beschrijving

Per use case kunnen er meerdere test scenario's zijn en voordat al deze scenario's geschreven gaan worden heb ik eerst een checklist gemaakt. Deze check list wordt gebruikt om samen met de eindgebruiker te kijken of hiermee alle mogelijke scenario's zijn afgedekt. In onderstaande tabel is deze checklist te zien voor alle use cases.

TC #	**Test case**	**Test omschrijving**
UC 1.1 Inloggen		
1.1.1	Invoeren foutieve inlog gegevens	De actor probeert in te loggen met onjuiste inlog gegevens.
1.1.2	Correcte inlog gegevens	De actor logt in met correcte inlog gegevens.
UC 1.2 Uitloggen		
1.2.1	Uitlog confirmatie	De actor logt uit.
UC 2 Dashboard bekijken		
2.1	Gegevens en interface controle	De gegevens in de grafiek moeten worden gecontroleerd op juistheid.
UC 3 Omzet overzicht bekijken		
3.1	Periode selectie	Bij het verwisselen van de periode moet de correcte informatie worden opgehaald.
UC 4 Omzet specificatie overzicht bekijken		
4.1	Periode selectie	Bij het verwisselen van de periode moet de correcte informatie worden opgehaald.
UC 5 Synchroniseren met Twinfield		
5.1	Synchroniseren van een periode	Er zal een periode worden gesynchroniseerd en de gegevens gecontroleerd.
UC 6 Memoriaal boeking details bekijken		
6.1	Bekijken van een memoriaal boeking	Er zal voor een boeking de details worden bekeken en gecontroleerd.
UC 7 Memoriaal boeking details printen		
7.1	Printen van een memoriaal boeking	De boeking details van een memoriaal boeking worden geprint.
UC 8 Concept correcties toevoegen		
8.1	Splitsen op uur basis	Een boeking wordt gesplitst met de uur basis preset en de

		toegevoegde concept correcties gecontroleerd op correctheid.
8.2	Splitsen voor onderhoud contract	Een boeking wordt gesplitst met de onderhoud preset en de toegevoegde concept correcties gecontroleerd op correctheid.
8.3	Parkeren van een boeking	Een boeking wordt geparkeerd en de concept correcties die hiervoor wordt gemaakt wordt gecontroleerd.
8.4	Handmatige splitsing	Een boeking wordt gesplitst naar eigen indeling zonder preset.
UC 9 Concept correcties wijzigen		
9.1	Aanpassen van een correctie	Een toegevoegde concept correctie wordt gewijzigd.
UC 10 Concept correcties verwijderen		
10.1	verwijderen van een correctie	Een toegevoegde concept correctie wordt verwijderd.
UC 11 Concept correcties verwerken		
11.1	Controleren journaalposten in het confirmatiescherm.	Er wordt gekozen om de concept correcties te gaan verwerken. In het confirmatiescherm worden voorbereide journaalposten getoond en gecontroleerd op correctheid.
11.2	Definitief boeken journaalposten.	In het confirmatiescherm wordt gekozen voor definitief boeken. De boekingen worden doorgevoerd en gecontroleerd.
UC 12 Concept facturen boeken		
12.1	Boeken van een concept factuur	Een concept factuur wordt geboekt op een periode en de boeking plus het resultaat worden gecontroleerd.
UC 13 Geboekte concept facturen matchen		
13.1	Het matchingscherm wordt getest.	Testen van het scherm waarmee gematched zal worden.
13.2	De facturen worden gematched	Een gemaakte matching in het matchingscherm wordt definitief gematched.

Tabel 6: Checklist voor alle scenario's van user en admin use cases.

10.7. Reflecteren van het proces

Nadat in sprint 5 alles was afgebakend wist ik precies wat er nog gedaan moest worden en dat er genoeg tijd was om de laatste onderdelen af te maken. Deze sprint is dus vrij rustig verlopen en vooral besteed aan de laatste twee user stories.

Het maken van een tabblad met een omzet specificatie per periode. Deze telling moest overeen komen met die in de grafieken van het dashboard maar dan met alle details. Daarom heb ik wat extra tijd besteed om zoveel mogelijk code te hergebruiken. Waar dit niet mogelijk was is de code generiek gemaakt zodat het wel mogelijk was. Ik ben tevreden over het eindresultaat waarbij goed gebruik wordt gemaakt van interfaces.

Nadat de laatste functionaliteit er in zat is het tijd om te beginnen aan het test plan maar ik heb besloten om eerst alle use cases op te schrijven en uit te werken. Op die manier heb ik een goed beeld van alles wat het systeem moet kunnen en de verwachte werking. Voor deze use cases kunnen nu test scenario's geschreven gaan worden. Ik ben tevreden over deze keuze omdat het maken van scenario's nu meer gestructureerd kan verlopen.

Daarnaast is de applicatie naar de test omgeving gezet, hier ben ik redelijk tevreden over. Het heeft wel twee dagen geduurd voordat alles geupdate was (inclusief andere systemen en databases) op de testomgeving.

11. Sprint 7

11.4. Vaststellen van de requirements

In deze laatste sprint zullen de 20 test scenario's worden afgemaakt op basis van de checklist vanuit sprint 6 en de laatste verbeteringen die uit de test fase komen nog worden uitgevoerd. Er is dus in deze sprint expres ruimte gelaten in de sprint backlog. Bij het signaleren van problemen kunnen deze direct worden opgepakt.

11.5. Uitwerken van test scenario's.

Om te voorkomen dat er bij het schrijven van dit plan en de scenario's er in al fouten ontstaan zal de gebruiker eerst de opgestelde test scenario's controleren. Deze zal kijken of de verwachte uitkomst klopt en overeenstemt dat dit ook zijn verwachtte uitkomst is. Ook wordt er gekeken of er nog bepaalde test cases vergeten zijn die moeten worden toegevoegd.

Met de checklist kunnen nu alle gevonden scenario's uitgewerkt worden in uitvoerbare tests. Twee voorbeelden hiervan zijn te zien in onderstaande tabellen. Bij deze scenario's is extra ruimte gelaten zodat de gebruiker zijn feedback kan geven. Deze feedback zal na het uitvoeren van de scenario's verwerkt worden om de applicatie te verbeteren.

Voor een volledig overzicht van alle test scenario's kunt u Bijlage F: Testplan inkijken.

Scenario 1.1.1	
Naam	Invoeren foutieve inlog gegevens
Use case	1.1 Inloggen
Omschrijving	De actor probeert in te loggen met onjuiste inlog gegevens.
Stap 1. Vraag de login pagina op	
Stap 2. Voer de login gegevens in	
Invoer	**Gebruikersnaam:** piet, **Wachtwoord:** hamburger
Stap 3. Klik op de knop inloggen	
Verwacht resultaat	De gebruiker wordt de toegang geweigerd en krijgt een melding getoond.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Tabel 7: Voorbeeld test scenario van invoeren foutieve login gegevens.

Scenario 1.1.2	
Naam	Invoeren correcte inlog gegevens
Use case	1.1 Inloggen
Omschrijving	De actor logt in met correcte inlog gegevens.
Vereiste	De actor moet een account met een condor rol hebben.
Stap 1. Vraag de login pagina op	
Stap 2. Voer de login gegevens in	
Invoer	**Gebruikersnaam:** “uw gebruikersnaam”, **Wachtwoord:** “uw wachtwoord“
Stap 3. Klik op de knop inloggen	
Verwacht resultaat	Er is ingelogd en het dashboard wordt getoond.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 3. Controleer of het inloggen helemaal goed is verlopen	
Verwacht resultaat	Uw naam en de knop uitloggen worden getoond in de bovenbalk
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Tabel 8: Voorbeeld test scenario invoeren van correcte login gegevens.

11.6. Uitvoeren van de test scenario’s en verwerken resultaten

Samen met de eindgebruiker / product owner zullen alle test scenario’s worden doorgelopen om te kijken. Hiermee kwamen vooral veel kleine dingen naar voren die in de frontend verbeterd konden worden.

Voor het optimaliseren van de frontend functionaliteit worden er een aantal extra functies toegevoegd aan de omzet sheet. Een filter waarmee op een zoekterm gezocht kan worden en de andere regels verborgen worden als ze niet aan de zoekterm voldoen. Er kan hierbij worden gezocht op een aantal velden zoals factuurnummer, boekstuknummer, grootboek rekening nummer, klantnaam en omschrijving.

Daarnaast werd het overzicht soms wat druk als er veel concepten en correcties waren. Daarom zijn er twee switch knoppen toegevoegd. Met deze knoppen kan het tonen en verbergen van concepten of correcties worden gedaan.

Voor het optimaliseren van de frontend functionaliteit worden er een aantal extra functies toegevoegd aan de omzet sheet. Een filter waarmee op een zoekterm gezocht kan worden en de andere regels verborgen worden als ze niet aan de zoekterm voldoen. Er kan hierbij worden gezocht op een aantal velden zoals factuurnummer, boekstuknummer, grootboek rekening nummer, klantnaam en omschrijving.

Daarnaast werd het overzicht soms wat druk als er veel concepten en correcties waren. Daarom zijn er twee switch knoppen toegevoegd. Met deze knoppen kan het tonen en verbergen van concepten of correcties worden gedaan.

Daarnaast zijn er nog een aantal andere verbeteringen, zo wordt nu de filterterm die is ingevuld onthouden bij het wisselen van perioden en tabblad. Het geselecteerde tabblad en periode wordt nu via de get parameters in de URL bewaard en kan nu gedeeplinkt worden.

Er is een selectie van jaar toegevoegd aan het dashboard zodat ook gegevens uit het vorige en komende jaar kunnen worden opgevraagd. Er zijn totaalcijfers en een filter mogelijkheid toegevoegd aan het specificatie tabblad. En tot slot zijn de boeking details die geprint kunnen worden uitgebreid met extra informatie en de lay-out is verbeterd.

11.7. Reflecteren van het proces

Het schrijven van de scenario's en verwachtte uitkomst is goed verlopen maar wel meer werk dan verwacht. Ik ben tevreden dat er veel kleine dingen die in de frontend beter konden naar voren kwamen zodat dit alvast is opgelost voor de eerste release. Het later aanpassen hiervan door iemand anders zou veel meer tijd hebben gekost.

De laatste sprint was op programmeergebied niet heel spannend maar er is wel veel getest en verbeterd. Ik ben tevreden over hoe deze sprint is verlopen maar had het schrijven van alle testen enigszins onderschat.

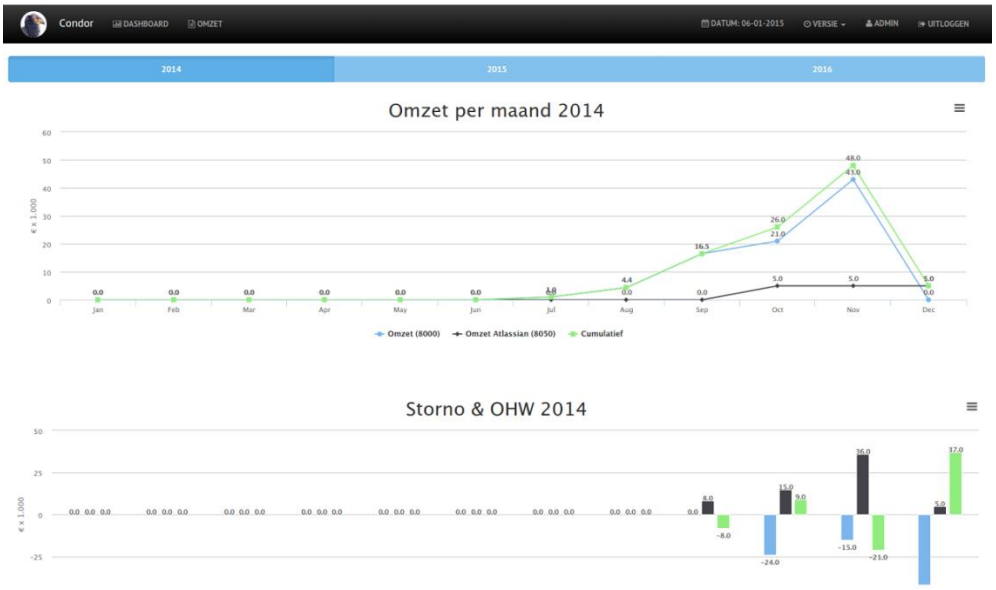
12. Product evaluatie

In dit hoofdstuk zal er een laatste kijk worden gegeven op het uiteindelijke eindproduct dat is opgeleverd. Er wordt gekeken wat de status is van de applicatie en wat er verder mee zal gaan gebeuren.

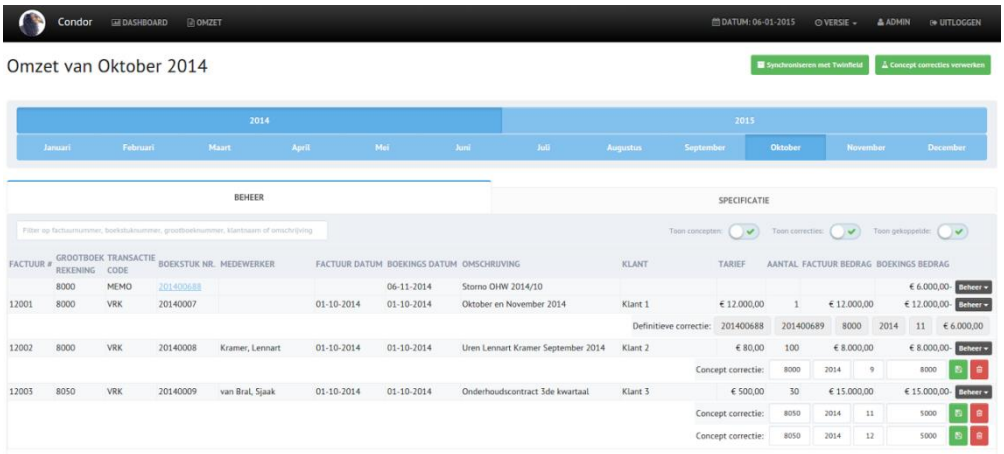
De eerste versie van de applicatie is nu af en de onderdelen die het bevat zijn goed getest. Er is gekozen om vooral één deel geheel in de diepte uit te werken waardoor de kwaliteit van het eindproduct hoog is geworden.

De opdrachtgever is zeer tevreden en heeft door het succes alweer nieuwe plannen voor de eerste uitbreiding. Ik zal zelf direct na afsluiting van deze afstudeerstage in vaste dienst treden bij 42 om deze uitbreiding te realiseren. Nadat deze uitbreiding af is zal de applicatie ook geheel in productie worden genomen. Vanwege de omslag van het boekjaar zal de applicatie pas volledig gebruikt worden na toevoeging van deze uitbreiding.

Zelf ben ik ook tevreden over het opgeleverde product vooral na het testen en verbeteren van de laatste dingen is het een mooi geheel geworden. Onderstaande figuren tonen het eindresultaat.



Figuur 28: Eindresultaat dashboard



Figuur 29: Eindresultaat omzet pagina

13. Beroepstaken

In dit hoofdstuk zal per gekozen beroepstaak worden aangegeven op welke manier deze behaald is.

Uitvoeren analyse door definitie van requirements (beroepstaak 1.4)

Tijdens het gehele traject door zijn de requirements onderhouden. Zo is iedere sprint opnieuw samen met de product owner besloten wat er vanuit de product backlog in de sprint backlog moest komen. Halverwege sprint 5 is de definitieve lijst van requirements opgesteld en wordt de product backlog leeg gemaakt.

Na het duidelijk worden van de definitieve requirements zijn met het oog op de gebruikers acceptatie testen alle use cases uitgewerkt. Hierin is alle functionaliteit omschreven die het systeem uiteindelijk zal moeten vervullen en wat de verwachte werking is. Door middel van het opstellen en bijhouden van de requirements en het opstellen en uitwerken van alle use cases denk ik voldaan te hebben aan deze beroepstaak.

Ontwerpen systeemdeel (beroepstaak 3.2)

Voor iedere sprint is er een ontwerp gemaakt door middel van een klassendiagram. Daarnaast is er bij iedere toevoeging aan de database een nieuw database model gemaakt. De lagen waaruit het systeem is opgebouwd zijn duidelijk te volgen. Er is een service laag die de communicatie naar buitenaf afwikkelt en daarboven een service laag die de modellen beheerd. Deze worden aangeroepen door de bovenste service laag die de meerdere services gebruikt.

Er is een duidelijke scheiding gemaakt tussen backend en frontend waardoor er een hele duidelijke lijn is waar wat in moet gebeuren. De frontend moet dom zijn en makkelijk gewisseld kunnen worden en dat is op deze manier het geval. Er wordt door de hele applicatie heen goed gebruik gemaakt van externe libraries en annotaties om in zo min mogelijk regels alles voor elkaar te krijgen. Ook de leesbaarheid en de onderhoudbaarheid van de code zijn tijdens de hele stage belangrijk geweest.

Bouwen applicatie (beroepstaak 3.3)

Iedere sprint is de applicatie verder ontwikkeld en zijn er delen van het systeem verbeterd of bijgebouwd. Daarnaast zijn alle methoden voorzien van Java documentatie en zijn deze zo klein mogelijk en voeren één duidelijke taak uit. Er is een geheel en goed werkend systeem vanuit niets opgebouwd.

Er is een 50 tal Java klassen geschreven en het programma bevat ook nog een aantal complexe stukken code. Vooral voor het wegboeken van concepten, om goede journaalposten op te stellen vanuit een enkele conceptregel was lastig. Daarnaast was er communicatie met een lastige externe XML soap API waardoor er een XML document moest worden gebouwd in de Java code. Ook het goed verwerken van de resultaten die deze API terug gaf kwam daarbij.

Uitvoeren van en rapporteren over het testproces (beroepstaak 3.5)

Er zijn in totaal enorm veel unit testen geschreven door de gehele looptijd heen en de test dekking van het uiteindelijke systeem is nagenoeg 100%. Naast het bouwen en uitvoeren van de unit tests is er ook een test plan opgesteld voor de gebruikers acceptatie testen. In dit test plan wordt per usecase een aantal scenario's opgesteld en wat de verwachte uitkomst is van dit scenario. Op basis van de resultaten hiervan zijn dan weer de nodige aanpassingen gemaakt.

De applicatie is toegevoegd aan sonar waardoor alle code conventies en programmeerregels gecontroleerd kunnen worden. Door te zorgen dat alle indicatoren daar goed zijn, de unit tests en de gebruikers acceptatietesten denk ik voldaan te hebben aan deze beroepstaak.

Figurenlijst

Bijlagen

Bijlage A: Afstudeerplan

Versie 1.0

Inhoudsopgave

1. Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2014-2.1 (start uiterlijk 1 september 2014)
Startdatum uitvoering afstudeeropdracht:
Inleverdatum afstudeerdossier volgens jaarrooster: 9 januari 2015

Studentnummer: 07042264
Achternaam: dhr Kramer
Voorletters: L.J.
Roepnaam: Lennart
Adres: Grondelsloot 37
Postcode: 2724 BT
Woonplaats: Zoetermeer
Telefoonnummer: 06-45 333 877
Mobiel nummer: 06-45 333 877
Privé emailadres: lennartkramer1988@gmail.com

Opleiding: Informatica
Locatie: Zoetermeer
Variant: voltijd

Naam studieloopbaanbegeleider: Vincent Broeren
Naam begeleidend examinator: Vincent Broeren
Naam tweede examinator: Emeri Koenen

Naam bedrijf: 42 BV
Afdeling bedrijf:
Bezoekadres bedrijf: Koraalrood 33
Postcode bezoekadres: 2718 SB
Postbusnummer:
Postcode postbusnummer:
Plaats: Zoetermeer
Telefoon bedrijf: 088-424 20 42
Telefax bedrijf:
Internetsite bedrijf: www.42.nl

Achternaam opdrachtgever: dhr Meijer
Voorletters opdrachtgever: E (Eric)
Titulatuur opdrachtgever:
Functie opdrachtgever: Bedrijfsleider
Doorkiesnummer opdrachtgever:
Email opdrachtgever: eric.meijer@42.nl

Achternaam bedrijfsmentor: dhr Bor
Voorletters bedrijfsmentor: R (Rob)
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor:
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: robert.bor@42.nl

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:
Realiseren van een geautomatiseerde projectadministratie met betrekking tot de financiële verantwoording.

2. Opdrachtomschrijving

2.1. Bedrijf

Opgericht in 2003 als een in JAVA gespecialiseerd software bedrijf en in 2009 samengegaan met Kensas, een expert in maatwerk oplossingen voor complexe vraagstukken. Hierdoor ontstond er een ICT organisatie met zo'n 35 eigen medewerkers met een hoofdkantoor in Zoetermeer. Om waar te maken dat 42 daadwerkelijk het "ultieme antwoord" is, richten zij zich dus op het ontwikkelen van hoogwaardige oplossingen en diensten.

Om te beginnen, diensten die het totale ontwikkeltraject van begin tot eind ondersteunen:

- Analyse gebruikerswensen
- Opstellen functioneel ontwerp
- Technisch ontwerp en realisatie
- Support en beheer.

Daarnaast levert 42 diensten op het gebied van audits en consultancy. En zijn ze Atlassian Expert

2.2. Probleemstelling

42 realiseert langlopende projecten waarvoor een uren- en projectadministratie wordt gevoerd. Hiervoor worden interne en externe systemen gebruikt:

- De financiële zaken worden vastgelegd in een boekhoudpakket (Twinfield).
- Medewerkers registreren gewerkte uren in het urenregistratiesysteem.
- In de projectadministratie worden afspraken vastgelegd met betrekking tot de financiën.
- In het boekhoudpakket is de huidige en historische status van de projecten terug te vinden.
- De reeds verzonden of conceptfacturen worden bijgehouden in het facturatiesysteem.
- Excel sheet met lopende periode en forecast.
- Handmatig toevoegen van de maandelijkse journaalposten.

De integratie tussen de verschillende systemen is beperkt. Het is nu mogelijk om vanuit SNIP (facturering) naar Twinfield (boekhouding) factuurinformatie over te zetten. Wat nog niet kan is het uitlezen van informatie en het koppelen van gegevens om inzicht te krijgen in de lopende periode en de forecast.

2.3. Doelstelling

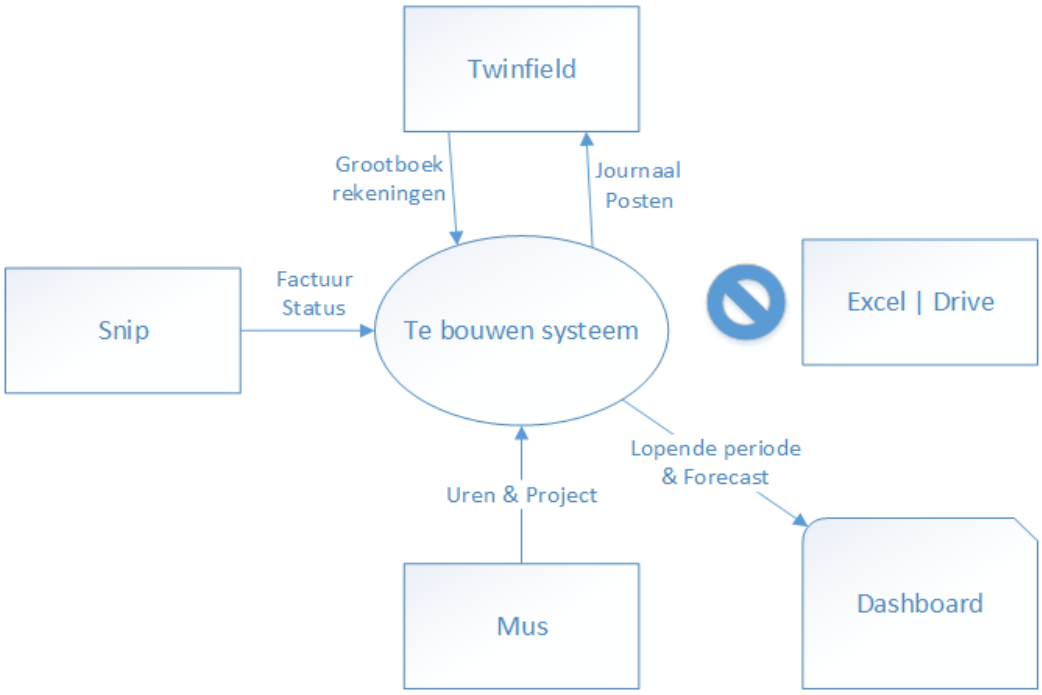
Een betere integratie tussen het boekhoudpakket, urenregistratie en facturatie systeem waarbij een duidelijk overzicht van de lopende periode en een forecast worden ontwikkeld. En daarbij ook het makkelijker maken van het boekhoudkundig proces door het automatiseren van het inboeken van journaalposten.

2.4. Resultaat

Na realisatie van het project zal er meer inzicht zijn in de huidige status van project met betrekking tot uren en kosten. Een inzichtelijk dashboard zal de huidige Excel sheet vervangen en meer informatie bieden m.b.t. huidige projecten en een forecast bieden. Het handmatig werk van het boekhoudpakket zal zijn afgenomen en meer geautomatiseerd.

Daarnaast ontstaat er een betere koppelingen tussen de verschillende systemen van het bedrijf:

- ***Twinfield**, extern boekhoudpakket (vergelijkbaar met bijvoorbeeld Exact).*
- ***MUS**, urenregistratie-, project- en klantinformatiesysteem.*
- ***SNIP**, facturering.*
- ***Raaf**, Twinfield informatie extractie tool*



De primair belanghebbenden zijn Eric Meijer (directeur) en Marvin Koot (boekhouder). Secundair, het management team, bestaande uit Eric Meijer, Ron Oudshoorn, Mat Huizing en Robert Bor.

2.5. Werkzaamheden, Globale planning en fasering

Uit te voeren werkzaamheden:

- Als start van de opdracht zullen de requirements moeten worden vastgelegd.
- Een plan van aanpak moeten worden geschreven over de realisatie van de verschillende onderdelen.
- Uitzoeken van API Twinfield (extern boekhoudpakket) en deze bruikbaar maken.
- Ophalen relevante data uit uren- en projectadministratie via API calls.
- Ophalen relevante data uit boekhoudpakket (grootboekrekeningen) via API calls.
- Combineren van deze data met factuurinformatie.
- Database voor het opslaan van deze gekoppelde gegevens.
- Creëren overzicht lopende periode en forecast.
- Automatiseren journaalposten naar boekhoudpakket.
- Testen van de geschreven code (hoge testcoverage gevraagd).
- Opleveren koppelsysteem

Voorafgaand aan de spints binnen het project zullen eerst alle requirements duidelijk moeten worden door gesprekken met de stakeholders. Dit zal gebeuren in de pre scrum fase in de eerste twee weken. De opdrachtgever vindt het wenselijk om snel resultaat te zien en wil steeds op basis van de vorige sprint bijsturen.

De indeling van deze sprints zal gemaakt worden op basis van de priorisering van de requirements die in de eerste twee projectweken worden opgesteld. Naar gelang er tijd is zullen er zoveel mogelijk van deze requirements opgeleverd worden.

De bovenstaande uit te voeren werkzaamheden betreffen de hoofdlijnen van het project en zullen verder worden gespecificeerd. In indeling vooraf is dus lastig te maken omdat de requirements nog exact duidelijk moeten worden en de prioriteit van de opdrachtgever op dat moment een grote rol spelen hierin.

2.6. Op te leveren producten

- *Requirementsdocument*
- *Plan van Aanpak*
- *Testrapportage*
- *Projectinformatiesysteem*
- *Database*
- *Documentatie op Confluence (architectuur, beslissingen, keuzes), dit waar gepast ondersteunt*
- *met diagrammen, plaatjes, tekeningen*
- *Deployment handleiding*
- *Testsuite*
- *Broncode in GIT*

2.7. Te demonstreren competenties

1.4 Uitvoeren analyse door definitie van requirements

De eerste twee weken van het project zullen geheel op de requirements gericht zijn en het achterhalen er van bij alle betrokken stakeholders. Dit zal gebeuren door middel van gesprekken met de stakeholders en het bijhouden van een requirements document. Daarna zal door de loop van het project deze requirements bijgewerkt moeten worden omdat de opdrachtgever betrokken blijft en tijdens de loop van het project wil blijven bijsturen.

3.2 Ontwerpen systeemdeel

Naast de requirements zal er ook een ontwerp gemaakt moeten worden dat ook met de veranderingen die tijdens het project ontstaan mee wijzigt. Dit ontwerp zal bestaan uit meerdere delen waarbij voor elk
systeemdeel er een deelontwerp bij zal komen. In dit ontwerp zullen dan eventuele design patterns worden opgenomen of moeilijke functionaliteit gemodelleerd. Daarnaast moet dit ontwerp worden aangesloten op de functionele eisen die in de pre scrum fase zijn opgesteld.

3.3 Bouwen applicatie

De applicatie moet in een MVC framework worden gebouwd en er moeten testen worden opgesteld voor de geschreven code. Daarnaast moet de code een lage complexiteit hebben, herbruikbaar zijn en
er moet gebruik gemaakt worden van programmeerstandaarden.

Er zal gewerkt worden met een gehele technische stack:

- Front-end (CSS, HTML, AngularJS, JavaScript)
- Midtier (Java, Spring, Maven, JPA/Hibernate, EJB en Tapestry)
- Database (PostgreSQL, MySQL)

3.5 Uitvoeren van en rapporteren over het testproces

Er moeten testen (testscripts) voor niet-functionele (en functionele) Kwaliteitsattributen worden opgesteld. Er wordt gebruik gemaakt van een test framework. De testrapportage bestaat uit conclusies over het in productie nemen van het systeem.

Bijlage B: Plan van aanpak

Versie 1.0

Inhoudsopgave

1. Opdrachtomschrijving

1.1. Probleemstelling

42 realiseert langlopende projecten waarvoor een uren- en projectadministratie wordt gevoerd. Hiervoor worden interne en externe systemen gebruikt:

- De financiële zaken worden vastgelegd in een boekhoudpakket (Twinfield).
- Medewerkers registreren gewerkte uren in het urenregistratiesysteem.
- In de projectadministratie worden afspraken vastgelegd met betrekking tot de financiën.
- In het boekhoudpakket is de huidige en historische status van de projecten terug te vinden.
- De reeds verzonden of conceptfacturen worden bijgehouden in het facturatiesysteem.
- Excel sheet met lopende periode en forecast.
- Handmatig toevoegen van de maandelijkse journaalposten.

De integratie tussen de verschillende systemen is beperkt. Het is nu mogelijk om vanuit SNIP (facturering) naar Twinfield (boekhouding) factuurinformatie over te zetten. Wat nog niet kan is het uitlezen van informatie en het koppelen van gegevens om inzicht te krijgen in de lopende periode en de forecast.

1.2. Doel

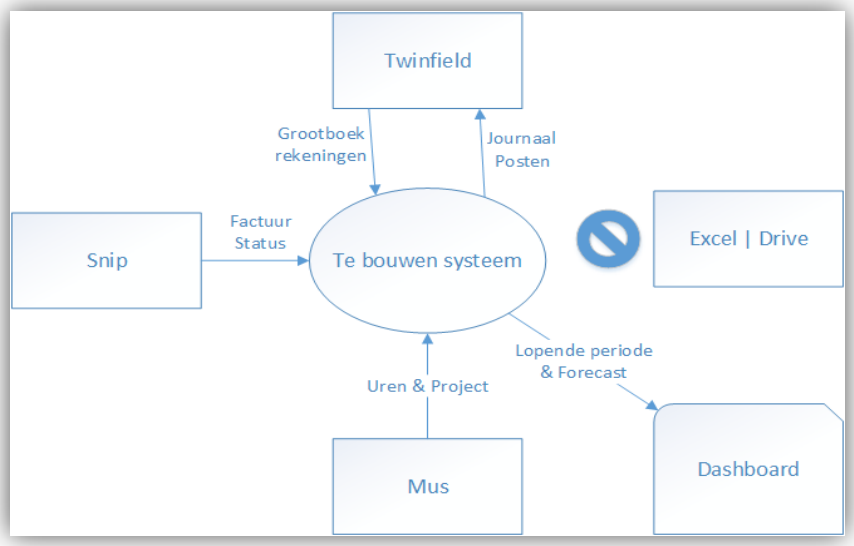
Een betere integratie tussen het boekhoudpakket, urenregistratie en facturatie systeem waarbij een duidelijk overzicht van de lopende periode en een forecast worden ontwikkeld. En daarbij ook het makkelijker maken van het boekhoudkundig proces door het automatiseren van het inboeken van journaalposten.

1.3. Scope

Na realisatie van het project zal er meer inzicht zijn in de huidige status van project met betrekking tot uren en kosten. Een inzichtelijk dashboard zal de huidige Excel sheet vervangen en meer informatie bieden m.b.t. huidige projecten en een forecast bieden. Het handmatig werk van het boekhoudpakket zal zijn afgenomen en meer geautomatiseerd.

Daarnaast ontstaat er een betere koppelingen tussen de verschillende systemen van het bedrijf:

- ***Twinfield**, extern boekhoudpakket (vergelijkbaar met bijvoorbeeld Exact).*
- ***MUS**, urenregistratie-, project- en klantinformatiesysteem.*
- ***SNIP**, facturering.*



2. Risico's

In dit hoofdstuk van het plan van aanpak worden de specifieke risico's voor dit project vernoemd. Elk risico zal worden omschreven en welke gevolgen het kan hebben op het verloop van het project. Daarnaast wordt de kans en impact bekeken en op basis hiervan een risiconiveau aangegeven. Ten slotte wordt per risico alvast een mogelijke oplossing geboden.

Risico	1. Problemen met de Twinfield API
Omschrijving	De opdrachtgever en project begeleider hebben aangegeven dat de Twinfield API lastig te begrijpen is. Er zijn andere programmeurs geweest waarbij het niet gelukt is om eruit te krijgen wat nodig is. Een probleem dat zou kunnen ontstaan is dat het voor mij ook niet lukt om alle benodigde handelingen uit te voeren.
Gevolgen	Niet alle resultaten zoals beschreven in het afstudeerplan kunnen behaald worden.
Kans	Middel, gezien het feit dat er al andere programmeurs zijn waarbij het niet goed is gelukt.
Impact	Middel, er zal mogelijk een klein deel van de beschreven functionaliteit niet worden opgeleverd maar het systeem kan nog steeds bruikbaar zijn.
Risiconiveau	Middel
Oplossing	Vroeg beginnen met communicatie van en naar de Twinfield API om eventuele problemen zo snel mogelijk te detecteren en trachten deze op te lossen.

3. Globale Planning

Onderstaande tabel bevat de globale planning voor de sprints, tussentijdens kunnen hier nog werkzaamheden aan toegevoegd worden. Testen zal over alle sprints heen gedaan worden en de eerste helft van het project zullen er nog requirements duidelijk worden uit de meetings.

Sprint nr.	Start datum	Eind datum	Producten	Werkzaamheden
	1-9-14	5-9-14	Plan v aanpak, Requirements	Schrijven plan van aanpak. Ontwikkelomgeving opzetten. Begin aan requirements document.
1	5-9-14	17-9-14	Plan v aanpak, Applicatie, requirements, testen	Ophalen van de eerste gegevens uit Postduif. Volledige technische stack bouwen van backend tot frontend.
2	17-9-14	30-9-14	Applicatie, requirements, testen	Twinfield connectie leggen en facturen ophalen. Extra focus op testen, van 0% naar minimaal 80% brengen.
3	30-9-14	14-10-14	Applicatie, database, requirements, testen	Twinfield connectie uitbreiden en alle omzet goed ophalen. Eigen database toevoegen om gegevens te persisteren. Concept correcties toevoegen, wijzigen en verwijderen.
4	14-10-14	28-10-14	Applicatie, database, requirements, testen	Wegboeken naar Twinfield toevoegen. Correcties toevoegen aan database en applicatie. Toevoegen van security aan frontend en backend.
5	28-10-14	11-11-14	Applicatie, database, testen	Optimaliseren van al opgeleverde functies zoals splitsen en verwerken concepten. Omzet sheet verbeteren. Storno en OHW overzicht toevoegen.
6	11-11-14	25-11-14	Applicatie, database, testen	Omzetgeschiedenis en forecast per project. Storno en OHW overzicht optimaliseren. Laatste grote requirements afmaken. Deployment naar de test server
7	25-11-14	9-12-14	Applicatie, database, testen	Feedback verwerken vanuit test omgeving. Dashboard optimaliseren. Laatste kleinere requirements afmaken. Toevoegen van een audit trail.
8	9-12-14	23-12-14	Documentatie & Deployment	Laatste optimalisaties aan de frontend. Oplevering naar productieomgeving. Volledige documentatie op confluence.

Bijlage C: Requirement document

Versie 1.0

Inhoudsopgave

1. User stories

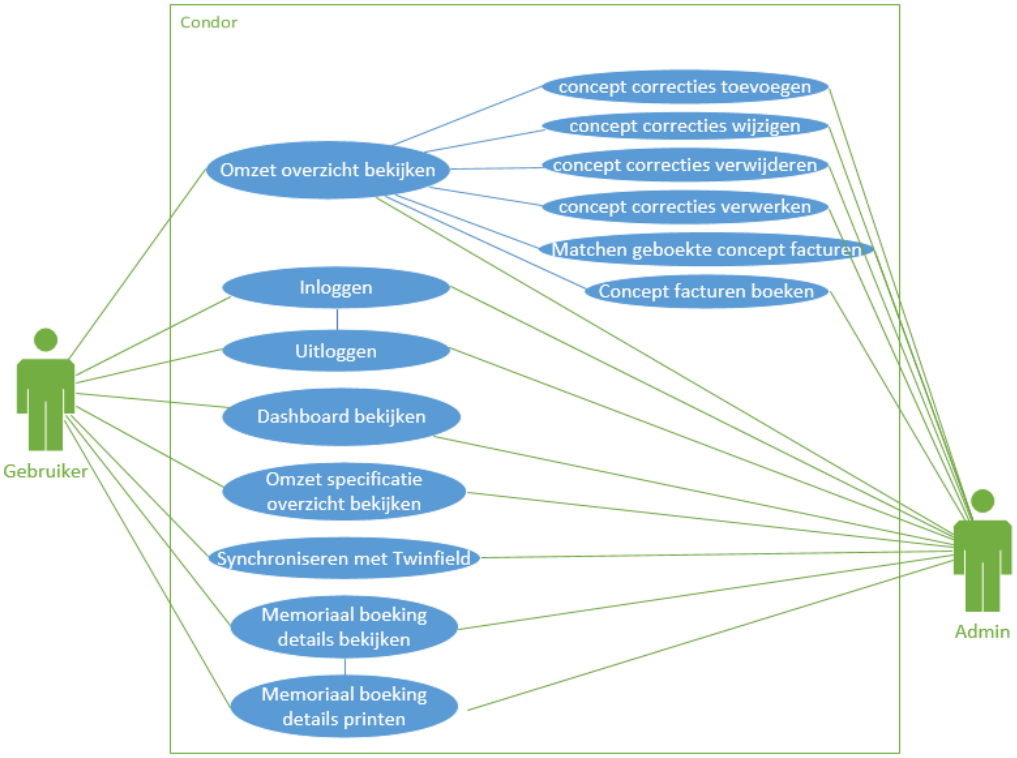
#	Omschrijving
1.	Als een gebruiker wil ik de omzet kunnen bekijken gesorteerd op factuurnummer per maand.
2.	Als een administrateur wil ik concept correcties kunnen toevoegen.
3.	Als een administrateur wil ik concept correcties kunnen wijzigen.
4.	Als een administrateur wil ik concept correcties kunnen verwijderen.
5.	Als een administrateur wil ik concept correcties definitief kunnen maken door ze in Twinfield te boeken.
6.	Als een administrateur wil ik concept facturen alvast kunnen boeken zodat deze in de omzet worden opgenomen voordat de factuur is verstuurd.
7.	Als een administrateur wil ik geboekte concept facturen kunnen matchen met een definitieve factuur of het bedrag laten vervallen.
8.	Als een gebruiker wil ik dat er eerst moet worden ingelogd zodat de gegevens beveiligd zijn tegen ongeautoriseerde toegang.
9.	Als een gebruiker wil ik dat er kan worden uitgelogd en de autorisatie vervalt.
10.	Als een gebruiker wil ik een dashboard met totaalcijfers voor omzet, OHW en storno OHW.
11.	Als een gebruiker wil ik een omzet specificatie kunnen inzien voor de opbouw van de totaalcijfers in het dashboard.
12.	Als een gebruiker wil ik kunnen synchroniseren met Twinfield waarmee boekingen worden opgehaald die nog niet in het systeem stonden.
13.	Als een gebruiker wil ik de details van een memoriaal boeking kunnen bekijken zodat ik een beter inzicht krijg in de opbouw van deze boeking.
14.	Als een gebruiker wil ik de details van een memoriaal boeking kunnen printen zodat ik deze aan de administratie kan toevoegen.

2. Use cases

Dit hoofdstuk van het requirement document bevat een use case diagram en bijbehorende use case beschrijvingen, opgedeeld in gebruiker en admin use cases in paragraaf 3.2 en alleen voor admin toegankelijke use cases in 3.3.

2.1. Use case diagram

Onderstaand use case diagram geeft schematisch alle functionaliteit weer die het systeem bevat. Een persoon met gebruikers rechten heeft toegang om dingen te bekijken en printen. Bijvoorbeeld de totaaloverzichten van het dashboard, specificaties, etc. Een admin gebruikt het omzet overzicht om alles te beheren en correcties te maken een gebruiker kan dit overzicht alleen bekijken.



2.2. Gebruiker en admin use case beschrijvingen

Naam	Inloggen
Use Case ID	UC 1.1
Omschrijving	De actor gebruikt het systeem om in the loggen.
Actoren	Admin, Gebruiker
Pre condities	2. De actor ziet het login scherm.
Beschrijving	3. De actor vult zijn login gegevens in. 4. Het systeem valideert de login gegevens. [1]
Post condities	2. De actor is ingelogd en ziet het dashboard.
Alternatieve flow	[1] De login gegevens zijn ongeldig, use case stopt.

Naam	Uitloggen
Use Case ID	UC 1.2
Omschrijving	De actor gebruikt het systeem om uit te loggen.
Actoren	Admin, Gebruiker
Pre condities	1. De actor is ingelogd.
Beschrijving	1. De actor verstuurd een uitlog verzoek. 2. Het systeem logt de actor uit.
Post condities	1. De actor is uitgelogd en ziet het login scherm.
Alternatieve flow	

Naam	Dashboard bekijken
Use Case ID	UC 2
Omschrijving	De actor gebruikt het systeem om het dashboard te bekijken.
Actoren	Admin, Gebruiker
Pre condities	1. De actor is ingelogd.
Beschrijving	1. De actor vraagt de dashboard pagina op. 2. Het systeem toont de dashboard pagina.
Post condities	1. De actor ziet de dashboard pagina.
Alternatieve flow	

Naam	Omzet overzicht bekijken
Use Case ID	UC 3
Omschrijving	De actor gebruikt het systeem om het omzet overzicht te bekijken.
Actoren	Admin, Gebruiker
Pre condities	1. De actor is ingelogd.
Beschrijving	1. De actor vraagt het omzet overzicht op. 2. Het systeem toont het omzet overzicht.
Post condities	1. De actor ziet het omzet overzicht.
Alternatieve flow	

Naam	Omzet specificatie overzicht bekijken
Use Case ID	UC 4
Omschrijving	De actor gebruikt het systeem om het omzet specificatie overzicht te bekijken.
Actoren	Admin, Gebruiker
Pre condities	1. De actor is ingelogd.
Beschrijving	1. De actor vraag het omzet specificatie overzicht op. 2. Het systeem toont het omzet specificatie overzicht.
Post condities	1. De actor ziet het omzet specificatie overzicht.
Alternatieve flow	

Naam	Synchroniseren met Twinfield
Use Case ID	UC 5
Omschrijving	De actor gebruikt het systeem om het omzet overzicht te synchroniseren met Twinfield.
Actoren	Admin, Gebruiker
Pre condities	1. De actor is ingelogd.
Beschrijving	1. De actor vraagt een synchronisatie aan met Twinfield 2. Het systeem synchroniseert met Twinfield.[1]
Post condities	1. De actor ziet het geupdate omzet overzicht.
Alternatieve flow	[1] Twinfield of Postduif zijn offline, use case stopt.

Naam	Memoriaal boeking details bekijken
Use Case ID	UC 6.1
Omschrijving	De actoren gebruiken het system om de memoriaal boeking details te bekijken.
Actoren	Admin, Gebruiker
Pre condities	1. De actor is ingelogd.
Beschrijving	1. De actor opent het memoriaal detail scherm. 2. Het systeem toont de boeking details van dat memoriaal.
Post condities	1. De actor ziet het memoriaal detail scherm.
Alternatieve flow	

Naam	Memoriaal boeking details printen
Use Case ID	UC 6.2
Omschrijving	De actoren gebruiken het system om de memoriaal boeking details te bekijken.
Actoren	Admin, Gebruiker
Pre condities	1. De actor is ingelogd. 2. De actor ziet het memoriaal details scherm.
Beschrijving	1. De actor opent het memoriaal detail scherm.
Post condities	2. Het systeem toont de boeking details van dat memoriaal.
Alternatieve flow	

2.3. Alleen voor admin toegankelijke use cases.

De volgende use cases zijn alleen toegankelijk voor admins.

Naam	Concept correcties toevoegen
Use Case ID	UC 8
Omschrijving	De actor gebruikt het systeem om concept correcties toe te voegen.
Actoren	Admin
Pre condities	1. De actor is ingelogd. 2. De actor heeft admin rechten. 3. De actor ziet het omzet overzicht
Beschrijving	1. De actor voegt concept correcties toe. 2. Het systeem slaat de concept correcties op.
Post condities	1. De actor ziet de toegevoegd concept correcties in het omzet overzicht.
Alternatieve flow	

Naam	Concept correcties wijzigen
Use Case ID	UC 9
Omschrijving	De actor gebruikt het systeem om concept correcties te wijzigen.
Actoren	Admin
Pre condities	1. De actor is ingelogd. 2. De actor heeft admin rechten. 3. De actor ziet het omzet overzicht
Beschrijving	1. De actor wijzigt de concept correcties 2. Het systeem slaat de wijzigingen op.
Post condities	1. De actor ziet de gewijzigde concept correcties in het omzet overzicht.
Alternatieve flow	

Naam	Concept correcties verwijderen
Use Case ID	UC 10
Omschrijving	De actor gebruikt het systeem om concept correcties te verwijderen.
Actoren	Admin
Pre condities	1. De actor is ingelogd. 2. De actor heeft admin rechten. 3. De actor ziet het omzet overzicht
Beschrijving	1. De actor verwijdert concept correcties. 2. Het systeem verwijdert de concept correcties uit de opslag.
Post condities	1. De actor ziet dat de concept correcties uit het omzet overzicht verwijderd zijn.,
Alternatieve flow	

Naam	Concept correcties verwerken
Use Case ID	UC 11
Omschrijving	De actor gebruikt het systeem om de concept correcties te verwerken.
Actoren	Admin
Pre condities	1. De actor is ingelogd. 2. De actor heeft admin rechten. 3. De actor ziet het omzet overzicht . 4. Er zijn concept correcties om te verwerken.
Beschrijving	1. De actor kiest verwerken concept correcties. 2. Het systeem toont een voorbeeld scherm met de journaalposten die aangemaakt zullen worden. 3. De actor kiest ervoor om de correcties definitief te boeken. 4. Het systeem voegt de boeking toe aan Twinfield en maakt de correcties definitief.
Post condities	1. De boekingen zijn toegevoegd aan Twinfield. 2. De correcties zijn definitief gemaakt. 3. De actor ziet het omzet overzicht met de toegevoegde boekingen en definitieve correcties.
Alternatieve flows	

Naam	Concept facturen boeken
Use Case ID	UC 12
Omschrijving	De actor gebruikt het systeem om de concept facturen te boeken.
Actoren	Admin
Pre condities	1. De actor is ingelogd. 2. De actor heeft admin rechten. 3. De actor ziet het omzet overzicht. 4. Er zijn nog niet geboekte concept facturen.
Beschrijving	1. De actor boekt de concept factuur op de gewenste periode. 2. Het systeem boekt de concept factuur in Twinfield.
Post condities	1. De concept factuur is geboekt. 2. De actor ziet de geboekte concept factuur in de juiste periode van het omzet overzicht.
Alternatieve flow	

Naam	Geboekte concept facturen matchen
Use Case ID	UC 13
Omschrijving	De actor gebruikt het systeem om de geboekte concept facturen te matchen.
Actoren	Admin
Pre condities	1. De actor is ingelogd. 2. De actor heeft admin rechten. 3. De actor ziet het omzet overzicht. 4. Er zijn geboekte concept facturen die nog niet zijn gematched.
Beschrijving	1. De actor kiest voor matchen concept facturen. 2. Het systeem voert de gemaakte matching uit.
Post condities	1. De matching is opgeslagen. 2. De actor ziet dat de matching verwerkt is in het omzet overzicht.
Alternatieve flow	

Bijlage D: Logboek

Versie 1.0

Inhoudsopgave

1. Logboek september

Ma 1 september	Di 2 september	Wo 3 september	Do 4 september	Vr 5 september
Installatie computers en programma. Plan van aanpak. Vragen voor opdrachtgever opstellen.	Verkenning van bestaand bedrijf software en klonen van die repositories.	Kick-off van project met opdrachtgever, eindgebruiker en begeleider. **Start sprint 1**	Start maken met requirements document.	Opzetten eigen project backend.
Ma 8 september	Di 9 september	Wo 10 september	Do 11 september	Vr 12 september
Opzetten eigen project frontend. Repository aangemaakt en project toegevoegd.	Frontend verbeteren door toevoeging van tabellen. Connectie met postduif API gelegd.	Backend API uitgebreid en view in voliere database aangepast. Daarnaast postduif API uitgebreid.	Backend gerefactored en services toegevoegd. Querybuilder voor optionele parameters.	Begonnen met connectie leggen naar Twinfield (extern boekhoudpakket) API.
Ma 15 september	Di 16 september	Wo 17 september	Do 18 september	Vr 19 september
Eerste draft voor een scherm dat de vervanging van Excelsheet gaat worden. Frontend en Backend.	Configuratie XML bestanden vervangen voor Java config. Java class initializer in plaats van web.xml	**Einde Sprint 1** Code Review **Start Sprint 2** Documentatie Sprint 1 bijwerken. Jetty server vervangen met Tomcat 7	Hele dag valt weg vanwege bedrijf dag (bijeenkomst voor alle medewerkers ten behoeve van informatie-uitwisseling).	Verwerken van de feedback van de code review van woensdag en algemene verbeteringen aan code.
Ma 22 september	Di 23 september	Wo 24 september	Do 25 september	Vr 26 september
Opzetjes gemaakt voor de te maken testen. Testadministratie Twinfield iets aangevuld Uren en Tarief toegevoegd aan omzet sheet.	Employee naam toegevoegd aan omzet sheet. Alle test classes aangemaakt Eerste paar tests werkend, test coverage nu 40%.	Test coverage van 40% naar 70 % gebracht Begonnen met herschrijven van HTTPClient Testen gaat al beter, jmockit en junit al beter begrepen.	HttpClient, account, account credentials weg gerefactored. Testen opgehoogd naar 80%. Eerste dingen uit Twinfield weten te halen.	Eigen Java annotatie gemaakt. Twinfield mapper die de annotatie gebruikt. Testen voor nieuwe Twinfield gedeelte toegevoegd, coverage 98%.
Ma 29 september	Di 30 september			
Twinfield query's aangepast zodat meer wordt opgehaald. Twinfield leidend gemaakt in omzet sheet.	**Einde Sprint 2** Code review **Start Sprint 3** Documentatie middag			

2. Logboek oktober

		Wo 1 oktober	Do 2 oktober	Vr 3 oktober
		Ontwerp database. Create statements voor eerste 3 tabellen. Opzetten eerste repository/ PostgreSQL connectie.	Profielen ingesteld voor spring, Groovy script voor database migratie, Repositories aangemaakt.	Script aangepast, repository werkend, Method gemaakt voor vullen tabel. Begonnen met testen schrijven.
Ma 6 oktober	Di 7 oktober	Wo 8 oktober	Do 9 oktober	Vr 10 oktober
Test coverage verder herstelt, 80% nu. Configuratie aangepast, jndi leest context.xml	updateScript verbetert. Geïmporteerd vanuit Twinfield productie, Config Tests	Begonnen met split functie, toevoegen en verwijderen. Netjes krijgen van de frontend.	Message handling toegevoegd aan de split functie in de frontend.	Gebruiker input validation toegevoegd in frontend en in backend. Documentatie middag.
Ma 13 oktober	Di 14 oktober	Wo 15 oktober	Do 16 oktober	Vr 17 oktober
Conceptfacturen toegevoegd aan postduif API en omzet sheet. Test coverage verhoogt.	**Einde Sprint 3** Code review **Begin Sprint 4** Correcties tabel script, model en model test gemaakt	Frontend preview scherm met correctie journaalpost, XML opbouwen en versturen naar Twinfield werkend.	Alle correctie journaalposten ook voor andere periodes worden nu aangemaakt.	Mapper gemaakt die het TwinfieldResultaat van het aanmaken van de journaalpost naar een object verwerkt.
Ma 20 oktober	Di 21 oktober	Wo 22 oktober	Do 23 oktober	Vr 24 oktober
Frontend opgeschoond en meer verplaatst naar backend. Detail overzicht toegevoegd en printen mogelijk gemaakt	berekeningen van frontend naar backend verplaatst.	Backend gerefactored en opgeschoond. In memory database draait, nog geen test data aanwezig.	Test data builder gemaakt die test data aanmaakt op basis van JSON. Rule compliance verbeterd.	Test coverage was gezakt naar 60% nu weer naar 80% gebracht. Security toegevoegd aan frontend en backend.
Ma 27 oktober	Di 28 oktober	Wo 29 oktober	Do 30 oktober	Vr 31 oktober
Grafieken in de frontend toegevoegd met Highcharts en verbeterd door middel van service. Test coverage	**Einde Sprint 4** **Begin Sprint 5** Inschieten issues sprint 5, testen en workshop GRID.	Test coverage hersteld van 80% naar 96%. Rule compliance naar 100%. **Documentatie middag.**	Splitsen verbeterd met presets en automatische detectie voor hour based. Verberg knoppen voor concepten + correcties.	Print overzicht van memoriaalboekingen uitbereid. Exception handling verbeterd door de applicatie heen. Code review

3. Logboek november

Ma 3 november	Di 4 november	Wo 5 november	Do 6 november	Vr 7 november
Boeken zonder tegen periode toegevoegd. Exception handler gemaakt die json als response returned bij een exception.	Begin gemaakt aan nieuwe overzicht voor financiële projectstatus .	Met opdrachtgever en eindgebruiker project afgebakend (definitieve requirements opgesteld). Begonnen met nog te factureren.	Deel 1 van nog te factureren af, er kan nu een concept naar Twinfield geboekt worden.	Gegevens voor de eerste grafiek in het dashboard worden nu echt opgehaald.
Ma 10 november	**Di 11 november**	**Wo 12 november**	**Do 13 november**	**Vr 14 november**
Tweede dashboard grafiek met echte gegevens. Meer testdata toegevoegd. Test coverage verhoogd.	**Einde Sprint 5** Code review **Begin Sprint 6**	Use case diagram opgesteld en eerste use case beschrijvingen toegevoegd.	Concept afstudeerverslag besproken. Wijzigingen doorgevoerd in proces verslag.	Use case beschrijvingen afgemaakt.
Ma 17 november	**Di 18 november**	**Wo 19 november**	**Do 20 november**	**Vr 21 november**
Omzet specificatie toegevoegd in een tab blad bij omzet overzicht.	Gerefactored zodat dezelfde methoden die worden gebruikt voor dashboard berekening kunnen worden hergebruikt.	Begonnen met user story voor matchen van geboekte concept facturen.	Matchen verder uitgewerkt, factuurnummer wordt overgenomen en correctieboeking gemaakt.	Documentatie & stageverslag verbetert.
Ma 24 november	**Di 25 november**	**Wo 26 november**	**Do 27 november**	**Vr 28 november**
Test coverage verhoogt van 70% naar 90%.	Laatste delen van het matching systeem afgemaakt. Alles wordt nu gekoppeld zoals moet.	Test plan verder uitgewerkt en geschreven scenario's direct uitgetest op lokale omgeving.	Verder gegaan met het schrijven van de test scenario's en het uitvoeren er van. Applicatie klaar gemaakt voor deployment naar test server.	Test server is klaargezet en de applicatie is toegevoegd en werkt grotendeels, test versie van postduif update wordt samengevoegd.

4. Logboek december

Ma 1 december **Einde Sprint 6** Code review **Begin Sprint 7** Laatste versie condor naar test server gezet. Test scenario's verder uitgewerkt.	Di 2 december Frontend meer responsive gemaakt en versie informatie toegevoegd.	Wo 3 december Bijwerken van het verslag voor tussentijds assessment.	Do 4 december Bijwerken van het verslag voor tussentijds assessment.	Vr 5 december Bijwerken van het verslag voor tussentijds assessment.
Ma 8 december Bijwerken van het verslag voor tussentijds assessment.	Di 9 december Geselecteerde periode en tab naar get parameters verplaats voor deeplinken.	Wo 10 december Ingestelde filter opties worden nu onthouden binnen de omzet pagina.	Do 11 december Connectie met postduif verbeterd zodat deze betrouwbaarder is.	Vr 12 december Tussentijds assessment. Verslag verbeteren na feedback van assessment.
Ma 15 december Laatste verbeteringen aan het booking details scherm.	Di 16 december Laatste test cases afgemaakt en uitgevoerd.	Wo 17 december Alle gebruikers testen nogmaals uitgevoerd vanaf het begin. Test coverage nog iets verhoogt.	Do 18 december **Einde Sprint 7** Code review **Afronding** Uitvoeren alle testen samen met eindgebruiker.	Vr 19 december Verslag en KERST KTM.
Ma 22 december Testen of totalen kloppen na toevoegen van alle data.	Di 23 december Applicatie naar acceptatie omgeving overzetten.	Wo 24 december	Do 25 december KERST	Vr 26 december KERST
Ma 29 december Bijwerken van het verslag voor definitief inleveren.	Di 30 december Bijwerken van het verslag voor definitief inleveren.	Wo 31 december Bijwerken van het verslag voor definitief inleveren.		

Bijlage E: Voortgangsverslag

Versie 1.0

Datum 31-10-2014

Inhoudsopgave

1. Voortgang

Dit voortgangsverslag wordt geschreven op 45% van de doorlooptijd van het project om de gemaakte voortgang tussentijds te evalueren. De bereikte mijlpalen worden besproken aan de hand van de behaalde requirements of grotendeels afgeronde requirements. Het hoofdstuk wordt afgesloten met de obstakels en belangrijke besluiten.

1.1. Bereikte mijlpalen

Onderstaande tabel bevat alle behaalde functionele requirements van de eerste 4 sprints. Op 45% van de stage zijn al meer dan de helft van de requirements behaald.

FR	Omschrijving	Behaald
1.	Het systeem moet journaalposten kunnen voorbereiden	Toegevoegd in sprint 4
2.	Voorstellen die de gebruiker nog kan wijzigen en later doorvoeren.	Dit is in sprint 3 toegevoegd en in sprint 4/5 geoptimaliseerd.
5.	Het systeem moet zowel al gefactureerde als nog te factureren projecten verwerken.	Deels toegevoegd, Restant toegevoegd sprint 5/6.
6.	Het systeem moet boekingen over meerdere periodes kunnen verdelen.	Toegevoegd in sprint 4, wordt geoptimaliseerd in sprint 5.
9.	Het systeem moet de omzetgegevens opslaan in een eigen database.	Toegevoegd in sprint 2/3
10.	Het systeem moet correctie / memoriaal boekingen kunnen toevoegen.	Toegevoegd in sprint 4
12.	Het systeem moet een admin en Gebruiker rol bevatten waarmee data toegang wordt beveiligt.	Toegevoegd in sprint 4

1.2. Obstakels

Alle requirements kunnen gehaald worden, op dit moment zijn er geen obstakels te noemen die dit kunnen verhinderen.

1.3. Belangrijke besluiten

Er zijn een aantal belangrijke besluiten geweest op programmeer technisch en database ontwerp niveau. Zo is er bij het ontwerpen van de database veel gekeken naar snelheid van ophalen en de indeling zo maken dat er zo min mogelijk joins worden gedaan. In de backend komt dit terug, de concepten en correcties worden niet opgehaald door ze mee te joinen maar met losse queries omdat dit sneller is.

Een belangrijke ontwerpkeuze is de manier waarop wordt gesplitst wordt en de wijze waarop dit wordt getoond. Daarbij ook rekening houdend met de verdere stappen, dat de concepten moeten worden veranderd in journaalposten en daarna veranderen in correcties die niet meer gewijzigd kunnen worden. Een belangrijke beslissing aan het begin van het project was hoe het ophalen en koppelen ging verlopen. Dit moest met zo min mogelijk data transfer naar Twinfield.

2. Resterende planning

Dit hoofdstuk bespreekt de resterende planning en is een controle om te kijken of de gemaakte planning in het plan van aanpak wel gerealiseerd kan worden.

Onderstaand de resterende planning uit het plan van aanpak, op dit moment ligt alles op schema en zal deze planning gehanteerd worden.

Sprint	Start datum	Eind datum	Producten	Werkzaamheden
5	28-10-14	11-11-14	Applicatie, database, testen	Optimaliseren van al opgeleverde functies .zoals splitsen en verwerken concepten. Omzet sheet verbeteren. Financiële projectstatus overzicht toevoegen.
6	11-11-14	25-11-14	Applicatie, database, testen	Financiële projectstatus overzicht optimaliseren. Laatste grote requirements afmaken. Dashboard met live gegevens vullen. Forecast toevoegen. Deployment naar de test server.
7	25-11-14	9-12-14	Applicatie, database, testen	Feedback verwerken vanuit test omgeving. Dashboard optimaliseren. Forecast optimaliseren. Laatste kleinere requirements afmaken. Toevoegen van een audit trail.
8	9-12-14	23-12-14	Applicatie, Documentatie & Deployment	Laatste optimalisaties aan de frontend. Oplevering naar productieomgeving. Volledige documentatie op confluence.

Onderstaande tabel bevat de nog te behalen requirements vanuit het requirements document met daarbij voor welke sprint deze gepland staan.

FR	Omschrijving	Planning
3.	Het systeem moet een forecast bieden.	Gepland voor sprint 6/7
4.	Het systeem moet een audit trail bevatten.	Gepland voor sprint 7
5.	Het systeem moet zowel al gefactureerde als nog te factureren projecten verwerken.	Deels toegevoegd, wordt afgemaakt in sprint 5/6.
7.	Het systeem moet de factuurgeschiedenis van een project kunnen tonen.	Gepland voor sprint 5/6
8.	Het systeem moet indicaties kunnen geven of de factuurbetaling goed verloopt.	Gepland voor sprint 6/7
11.	Het systeem moet een overzicht bieden waarin financiële status van een project gespecificeerd is.	Gepland voor sprint 5-7

Op dit moment is het de bedoelingen dat er alleen nog maar kleine requirements bij kunnen komen. Die te maken hebben met het verbeteren van de functionaliteit maar die niet geen grote nieuwe functionaliteit toevoegen.

Bijlage F: Test plan

Versie 1.0

Inhoudsopgave

1. Test aanpak

Voor alle geschreven functionaliteit zullen eerst unit testen worden geschreven zodat de test dekking rond de 100% is. Dit zal dienen als een eerste indicatie dat de geschreven functionaliteit correct functioneert.

Om te bevestigen of alles naar verwachting functioneert zullen er naast de unit testen ook gebruikers acceptatie testen worden uitgevoerd. Om deze acceptatietesten goed te laten verlopen zal eerst een plan worden geschreven waarbij alle use cases getest worden op alle mogelijk uitkomsten.

Om te voorkomen dat er bij het schrijven van dit plan en de scenario's er in al fouten ontstaan zal de gebruiker eerst de opgestelde test scenario's controleren. Deze zal kijken of de verwachte uitkomst klopt en overeenstemt dat dit ook zijn verwachtte uitkomst is. Ook wordt er gekeken of er nog bepaalde test cases vergeten zijn die moeten worden toegevoegd.

Om dit gemakkelijk te laten verlopen is er een afvinklijst gemaakt die voor alle use case de mogelijke paden per use case aangeeft. Deze checklist bevat ook direct lege velden waarin de gebruiker nog andere mogelijke paden kan aangeven.

1.1. Test basis

Use cases

UC #	Naam	UC #	Naam
User & Admin use cases		**Admin use cases**	
1.1	Inloggen	8	Concept correcties toevoegen
1.2	Uitloggen	9	Concept correcties wijzigen
2	Dashboard bekijken	10	Concept correcties verwijderen
3	Omzet overzicht bekijken	11	Concept correcties verwerken
4	Omzet specificatie overzicht bekijken	12	Concept facturen boeken
5	Synchroniseren met Twinfield	13	Geboekte concept facturen matchen
6	Memoriaal boeking details bekijken		
7	Memoriaal boeking details printen		

2. Checklist test scenario’s

In dit hoofdstuk vindt u een checklist die u kunt gebruiken om bij te houden welke scenario’s zijn afgerond.

2.1. User en admin test scenario’s

Onderstaande tabel bevat alle scenario’s die met zowel user als admin rechten kunnen worden uitgevoerd.

TC #	Test case	Test omschrijving	Afgerond
UC 1.1 Inloggen			
1.1.1	Invoeren foutieve inlog gegevens	De actor probeert in te loggen met onjuiste inlog gegevens.	
1.1.2	Correcte inlog gegevens	De actor logt in met correcte inlog gegevens.	
UC 1.2 Uitloggen			
1.2.1	Uitlog confirmatie	De actor logt uit.	
UC 2 Dashboard bekijken			
2.1	Gegevens en interface controle	De gegevens in de grafiek moeten worden gecontroleerd op juistheid.	
UC 3 Omzet overzicht bekijken			
3.1	Periode selectie	Bij het verwisselen van de periode moet de correcte informatie worden opgehaald.	
UC 4 Omzet specificatie overzicht bekijken			
4.1	Periode selectie	Bij het verwisselen van de periode moet de correcte informatie worden opgehaald.	
UC 5 Synchroniseren met Twinfield			
5.1	Synchroniseren van een periode	Er zal een periode worden gesynchroniseerd en de gegevens gecontroleerd.	
UC 6 Memoriaal boeking details bekijken			
6.1	Bekijken van een memoriaal boeking	Er zal voor een boeking de details worden bekeken en gecontroleerd.	
UC 7 Memoriaal boeking details printen			
7.1	Printen van een memoriaal boeking	De boeking details van een memoriaal boeking worden geprint.	

2.2. Admin test scenario's

Onderstaande tabel bevat scenario's die alleen met admin rechten kunnen worden uitgevoerd.

TC #	Test case	Test omschrijving	Afgerond
UC 8 Concept correcties toevoegen			
8.1	Splitsen op uur basis	Een boeking wordt gesplitst met de uur basis preset en de toegevoegde concept correcties gecontroleerd op correctheid.	
8.2	Splitsen voor onderhoud contract	Een boeking wordt gesplitst met de onderhoud preset en de toegevoegde concept correcties gecontroleerd op correctheid.	
8.3	Parkeren van een boeking	Een boeking wordt geparkeerd en de concept correcties die hiervoor wordt gemaakt wordt gecontroleerd.	
8.4	Handmatige splitsing	Een boeking wordt gesplitst naar eigen indeling zonder preset.	
UC 9 Concept correcties wijzigen			
9.1	Aanpassen van een correctie	Een toegevoegde concept correctie wordt gewijzigd.	
UC 10 Concept correcties verwijderen			
10.1	verwijderen van een correctie	Een toegevoegde concept correctie wordt verwijderd.	
UC 11 Concept correcties verwerken			
11.1	Controleren van de voorbereide journaalposten in het confirmatiescherm.	Er wordt gekozen om de concept correcties te gaan verwerken. In het confirmatiescherm worden voorbereide journaalposten getoond en gecontroleerd op correctheid.	
11.2	Definitief boeken van de voorbereide journaalposten.	In het confirmatiescherm wordt gekozen voor definitief boeken. De boekingen worden doorgevoerd en gecontroleerd.	
UC 12 Concept facturen boeken			
12.1	Boeken van een concept factuur	Een concept factuur wordt geboekt op een periode en de boeking plus het resultaat worden gecontroleerd.	
UC 13 Geboekte concept facturen matchen			
13.1	Het matchingscherm wordt getest.	Testen van het scherm waarmee gematched zal worden.	
13.2	De facturen worden gematched	Een gemaakte matching in het matchingscherm wordt definitief gematched.	

3. Test scenario's

U kunt het beste de scenario's op volgorde uitvoeren, soms is dat ook een vereiste omdat het ene test scenario verder gaat vanuit het vorige.

Scenario 1.1.1	
Naam	Invoeren foutieve inlog gegevens
Use case	1.1 Inloggen
Omschrijving	De actor probeert in te loggen met onjuiste inlog gegevens.
Stap 1. Vraag de login pagina op	
Stap 2. Voer de login gegevens in	
Invoer	**Gebruikersnaam:** piet, **Wachtwoord:** hamburger
Stap 3. Klik op de knop inloggen	
Verwacht resultaat	De gebruiker wordt de toegang geweigerd en krijgt een melding getoond.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 1.1.2	
Naam	Invoeren correcte inlog gegevens
Use case	1.1 Inloggen
Omschrijving	De actor logt in met correcte inlog gegevens.
Vereiste	De actor moet een account met een condor rol hebben.
Stap 1. Vraag de login pagina op	
Stap 2. Voer de login gegevens in	
Invoer	**Gebruikersnaam:** "uw gebruikersnaam", **Wachtwoord:** "uw wachtwoord"
Stap 3. Klik op de knop inloggen	
Verwacht resultaat	Er is ingelogd en het dashboard wordt getoond.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 3. Controleer of het inloggen helemaal goed is verlopen	
Verwacht resultaat	Uw naam en de knop uitloggen worden getoond in de bovenbalk
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 1.2.1 Uitlog confirmatie	
Naam	Uitlog confirmatie
Use case	1.2 uitloggen
Omschrijving	De actor moet succesvol kunnen uitloggen.
Vereiste	De actor moet ingelogd zijn
Stap 1. Log in door scenario 1.1.2 uit te voeren.	
Stap 2. Druk op de uitlog knop rechtsboven in de menubalk.	
Verwacht resultaat	De inlog pagina wordt getoond en in de menubalk zijn de links verborgen en is een knop inloggen te zien.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 2.1	
Naam	Gegevens en interface controle
Use case	2. Dashboard bekijken
Omschrijving	De gegevens in de grafiek moeten worden gecontroleerd op juistheid.
Vereiste	De actor moet ingelogd zijn. Er zijn nog geen wijzigingen gemaakt of de database is zojuist gereset.
Stap 1. Vraag de Dashboard pagina op.	
Stap 2. Controleer of alle maanden 0 zijn.	
Verwacht resultaat	Omzet Atlassian: 0, Omzet: 0, Cumulatief:0
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 3. Vraag de omzet pagina op.	
Stap 4. Vraag de periode 2014 / juni op.	
Stap 5. Klik op synchroniseer met Twinfield.	
Verwacht resultaat	Er worden twee regels toegevoegd 15.114,30 en 13.416,80 waarbij de laatste een concept correctie krijgt die naar 2014/mei wordt verplaatst
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 6. Ga terug naar de dashboard pagina.	
Verwacht resultaat	**Omzet per maand tabel:** MEI: Omzet 13.4, Atlassian: 0.0, Cumulatief:13.4 JUNI: Omzet 15.1, Atlassian:0.0, Cumulatief:15.1 **Storno & OHW tabel:** MEI: OHW 13.4, saldo -13.4 JUNI: Storno 13.4, saldo 13.4
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 3.1	
Naam	Periode selectie
Use case	3. omzet overzicht bekijken
Omschrijving	Bij het verwisselen van periode moet de correcte informatie worden opgehaald.
Vereiste	De actor moet ingelogd zijn.
Stap 1. Vraag de omzet pagina op.	
Stap 2. Vraag de periode 2014 / juni op.	
Verwacht resultaat	Er worden facturen getoond met een factuurdatum in de maand juni (06)
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 3. Vraag periode 2014 / juli op.	
Verwacht resultaat	De tabel is leeg
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 4. Klik op synchroniseren met Twinfield	
Verwacht resultaat	Er worden facturen binnengehaald met een factuur en boekings datum in de maand juli (07).
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 4.1	
Naam	Periode selectie
Use case	4. omzet specificatie overzicht bekijken
Omschrijving	Bij het verwisselen van de periode moet de correcte informatie worden opgehaald.
Vereiste	De actor moet ingelogd zijn. De actor heeft zojuist scenario 3.1 uitgevoerd en ziet periode 2014 / july
Stap 1. Klik op de tab specificatie	
Verwacht resultaat	7 facturen uit Juli, totaal voor 23.964,36
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 2. Vraag de periode 2014 / juni op.	
Verwacht resultaat	1 factuur uit juni en 8 uit juli, totaal voor 86.123,05
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 5.1	
Naam	Periode selectie
Use case	5. Synchroniseren met Twinfield.
Omschrijving	Er zal een periode worden gesynchroniseerd en de gegevens gecontroleerd.
Vereiste	De actor moet ingelogd zijn. De actor heeft zojuist scenario 4.1 uitgevoerd en ziet periode 2014 / juni
Stap 1. Vraag de periode 2014 / augustus op.	
Stap 2. Klik op de tab beheer.	
Stap 3. Klik op synchroniseren met Twinfield.	
Verwacht resultaat	Er wordt 1 factuur opgehaald.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 6.1	
Naam	Bekijken van een memoriaal boeking
Use case	6. Memoriaal boeking details bekijken
Omschrijving	Er zal voor een boeking de details worden bekeken en gecontroleerd.
Vereiste	De actor moet ingelogd zijn. De actor heeft zojuist scenario 5.1 uitgevoerd en ziet periode 2014 / augustus.
Stap 1. Klik op conceptboekingen verwerken.	
Stap 2. Klik op definitief boeken.	
Verwacht resultaat	De correctie voor factuur 13484 op uur basis is definitief gemaakt .
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 3. Klik op het boekstuk nummer van de zojuist aangemaakte memo boeking van 2170,00 euro.	
Verwacht resultaat	Alle details van de boeking worden getoond en zijn correct.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 7.1	
Naam	Printen van een memoriaal boeking
Use case	7. Memoriaal boeking details printen
Omschrijving	De boeking details van een memoriaal boeking worden geprint.
Vereiste	1. De actor moet ingelogd zijn. 2. De actor heeft zojuist scenario 6.1 uitgevoerd en ziet de boeking details.
Stap 1. Druk op de knop printen	
Stap 2. Kies een printer en print het overzicht uit.	
Verwacht resultaat	Het detail overzicht is uitgeprint.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 8.1	
Naam	Splitsen op uur basis
Use case	8. Concept correcties toevoegen
Omschrijving	Een boeking wordt gesplitst met de uur basis preset en de toegevoegde concept correcties gecontroleerd op correctheid.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 7.1 uitgevoerd en ziet de boeking details.
Stap 1. Sluit de boeking details	
Stap 2. Vraag de periode 2014/Juli op	
Stap 3. Klik op de knop beheer van factuur 13428	
Stap 4. Selecteer uur basis (-1A)	
Verwacht resultaat	Er wordt een concept correctie toegevoegd voor 1 periode eerder en het gehele bedrag.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 8.2	
Naam	Splitsen voor onderhoud contract
Use case	8. Concept correcties toevoegen
Omschrijving	Een boeking wordt gesplitst met de onderhoud preset en de toegevoegde concept correcties gecontroleerd op correctheid.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 8.1 uitgevoerd en ziet de periode 2014/Juli.
Stap 1. Typ in het filter veld " 3e kwartaal"	
Verwacht resultaat	Er worden twee ASR facturen getoond
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 2. Kies voor beide facturen beheer en dan de optie onderhoud	
Verwacht resultaat	Er zijn voor beide facturen 2 concept correcties toegevoegd waarmee beide facturen gelijkmatig over drie maanden worden gesplitst.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 8.3	
Naam	Parkeren van een boeking
Use case	8. Concept correcties toevoegen
Omschrijving	Een boeking wordt geparkeerd en de concept correcties die hiervoor wordt gemaakt wordt gecontroleerd.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 8.2 uitgevoerd en ziet de periode 2014/Juli.
Stap 1. Klik op de Reset filter knop	
Stap 2. Type 13427 in het filter veld	
Stap 3. Kies voor beide facturen beheer en dan de optie parkeren.	
Verwacht resultaat	Voor beide facturen is een concept correctie aangemaakt waarbij grootboekrekening en periode niet aangepast kunnen worden.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 8.4	
Naam	Handmatige splitsing
Use case	8. Concept correcties toevoegen
Omschrijving	Een boeking wordt gesplitst naar eigen indeling zonder preset.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 8.3 uitgevoerd en ziet de periode 2014/Juli.
Stap 1. Klik op de Reset filter knop	
Stap 2. Typ in het filter 13441	
Stap 3. Kies voor beide regels 3 maal beheer en dan de optie anders.	
Verwacht resultaat	Er zijn voor beide regels 3 concept correcties aangemaakt waarbij de perioden oplopen en de bedragen gelijkmatig verdeeld zijn.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 9.1	
Naam	Aanpassen van een correctie
Use case	9. Concept correcties wijzigen
Omschrijving	Een toegevoegde concept correctie wordt gewijzigd.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 8.4 uitgevoerd en ziet de periode 2014/Juli.
Stap 1. Wijzig de concept correctie van 37,50 in periode 2014/8 naar 75,00	
Stap 2. Klik op de groene opslaan knop achter de concept correctie	
Verwacht resultaat	Er blijft op de huidige periode een boekingsbedrag achter van 0,00
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 3. Wijzig de concept correctie van 42,50 in periode 2014/8 naar 85,00	
Stap 4. Stap 2. Klik op de groene opslaan knop achter de concept correctie	
Verwacht resultaat	Er blijft op de huidige periode een boekingsbedrag achter van 0,00
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 10.1	
Naam	verwijderen van een correctie
Use case	10. Concept correcties verwijderen
Omschrijving	Een toegevoegde concept correctie wordt verwijderd.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 9.1 uitgevoerd en ziet de periode 2014/Juli.
Stap 1. Verwijder voor beide regels de concept correcties van 2014/9 en 2014/10	
Verwacht resultaat	Er blijft voor beide regels maar 1 concept correctie over
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 11.1	
Naam	Controleren van de voorbereide journaalposten in het confirmatiescherm.
Use case	11. Concept correcties verwerken
Omschrijving	Er wordt gekozen om de concept correcties te gaan verwerken. In het confirmatiescherm worden voorbereide journaalposten getoond en gecontroleerd op correctheid.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 10.1 uitgevoerd en ziet de periode 2014/Juli.
Stap 1. Klik op de knop reset filter	
Stap 2. Klik op de knop conceptboekingen verwerken	
Stap 3. Controleer of alle journaalposten kloppen door het na te rekenen	
Verwacht resultaat	De journaalposten die worden getoond zijn naar verwachting
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 11.2	
Naam	Definitief boeken van de voorbereide journaalposten.
Use case	11. Concept correcties verwerken
Omschrijving	In het confirmatiescherm wordt gekozen voor definitief boeken. De boekingen worden doorgevoerd en gecontroleerd.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 11.1 uitgevoerd en ziet het confirmatiescherm.
Stap 1.Klik op de knop definitief boeken.	
Verwacht resultaat	Er zijn twee MEMO regels bijgekomen, 1 voor rekening 8000 en 1 voor 8050 en alle concept correcties zijn definitieve correcties geworden.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 12.1	
Naam	Boeken van een concept factuur
Use case	12. Concept facturen boeken
Omschrijving	Een concept factuur wordt geboekt op een periode en de boeking plus het resultaat worden gecontroleerd.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 11.2 uitgevoerd en ziet periode 2014/july
Stap 1. Vraag periode 2014/November op	
Stap 2. Filter op "belastingdienst"	
Stap 3. Kies voor de 3 facturen boeken en dan concept factuur boeken.	
Stap 4. Kies voor de 3 facturen de oranje opslaan knop.	
Verwacht resultaat	De drie concept facturen zijn geboekt
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 13.1	
Naam	Het matchingscherm wordt getest.
Use case	13. Geboekte concept facturen matchen
Omschrijving	Testen van het scherm waarmee gematched zal worden.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 12.1 uitgevoerd en ziet periode 2014/november.
Stap 1. Vraag periode 2014/December op	
Stap 2. Klik op synchroniseren met Twinfield	
Stap 3. Klik op de knop beheer en kies voor matchen.	
Stap 4. Verwijder de automatisch aangemaakte concept correctie.	
Verwacht resultaat	Er wordt een overzicht getoond met geboekte concept facturen die nog niet gematched zijn.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Scenario 13.2	
Naam	De facturen worden.
Use case	13. Geboekte concept facturen matchen
Omschrijving	Een gemaakte matching in het matchingscherm wordt definitief.
Vereiste	De actor moet ingelogd zijn als admin. De actor heeft zojuist scenario 13.1 uitgevoerd en ziet periode 2014/December.
Stap 1. Vink de concept facturen aan om ze te matchen	
Verwacht resultaat	Er wordt een voorbeeld van de journaalpost getoond
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Stap 2. Klik op facturen matchen.	
Verwacht resultaat	Er is een memo toegevoegd en de drie omzet regels zijn gekoppeld.
Echt resultaat	Correct / Incorrect (doorhalen wat niet van toepassing is)
Opmerkingen (vermeld bij welke stap het fout ging en wat er incorrect was)	

Bijlage G: Beoordeling bedrijfsmentor

Versie 1.0

DE HAAGSE HOGESCHOOL

Academie voor ICT & Media

Den Haag

Evaluatieformulier afstuderen

In te vullen door opdrachtgever c.q. bedrijfsmentor(en)

Student: Lennart Kramer

Periode: september 2014 – januari 2015

Bedrijf c.q. instelling: 42 BV

Bedrijfsmentor: Robert Bor

Plaats: Zoetermeer

Datum: 6 januari 2015

1. **Heeft de student zich zelf snel en goed ingewerkt in het bedrijf en de uit te voeren afstudeeropdracht?**

Lennart heeft zich bijzonder snel ingewerkt in de materie en de uit te werken opdracht. Vrijwel vanaf dag 1 heeft hij laten zien dat het schakelen met de opdrachtgever tot zijn sterke punten behoort.

2. **Hoe beoordeelt u de communicatieve vaardigheden van de student (in de samenwerking met collega's, in contacten met de opdrachtgever, bij mondelinge presentaties, schriftelijke rapportages)?**

Zeer goed. Lennart luistert goed, gaat met feedback aan de slag en koppelt dit ook terug. Hij is bedreven in het businessdomein, begrijpt de klant niet alleen goed maar kan ook prima suggesties aangeven hoe zaker beter kunnen.

3. **Hoe heeft de student tijdens het uitvoeren van de opdracht gefunctioneerd?**

- **Qua verantwoordelijkheid** goed
- **Qua zelfstandigheid** goed
- **Qua planmatig werken** goed
- **Qua creativiteit** goed
- **Qua productiviteit** goed
- **Qua samenwerken met collega's** goed
- **Qua draagvlakontwikkeling** goed
- **Qua inspelen op bedrijfscultuur** goed
- **Qua rekening houden met de specifieke context van het bedrijf** goed
- **Qua het op gang brengen van de nodige veranderingen** goed

4. **Hoe beoordeelt u de kennis en kunde van de student in verhouding tot wat u verwacht van een bijna afgestudeerde?**

Lennart opereert boven het niveau van wat ik verwacht van een afstudeerder.

5. **Hoe beoordeelt u de kwaliteit van de opgeleverde (tussen)producten?**

Bijzonder goed. Lennart heeft meer opgeleverd en van een hogere kwaliteit dan werd verwacht. Het platform wat hij heeft gebouwd is enthousiast ontvangen door de twee opdrachtgevers en gaat een sleutelrol spelen in de organisatie.

6. **Bent u tevreden over het opgeleverde (eind)product?**

- **In hoeverre heeft u gekregen wat is afgesproken?**

Meer dan wat is afgesproken.

- **In hoeverre voldoet het (eind)product aan uw verwachtingen?**

Bovenverwachting goed.

- **Wat is de bruikbaarheid en onderhoudbaarheid hiervan?**

Direct bruikbaar.

- **Wat gebeurt er met het opgeleverde (eind)product?**

Wordt gelijk ingezet door de boekhouder.

- **Kunt u direct met het opgeleverde product aan de slag?**

Ja

7. **Zijn er nog aspecten voor u van belang die nog niet aan de orde zijn geweest?**

Lennart is binnengekomen met vrijwel geen kennis van onze technologiestack en is tijdens zijn afstudeerperiode een aandachtig student geweest. Hij heeft heel veel geleerd en heeft nog veel meer in zijn mars.

Zijn communicatie met de opdrachtgever was van een dusdanig niveau dat ik, als begeleider, ertussen uit stapte om de synergie tussen deze vakmensen niet te verstoren. Dit heb ik nog nooit gezien en verdient erkenning.

8. **Bent u bereid een volgende keer weer uw medewerking te verlenen aan het beschikbaar stellen van een afstudeerplaats** (graag met toelichting)**?**

Ja, zoals gebruikelijk.