Ontwikkeling Adaptermodel en Adapters

NOKAVISION
SOFTWARE

Nokavision Software
Bruistensingel 144
5232 AC
's-Hertogenbosch

tel: 073 64 08 490
@: www.nokavision.com
e: info@nokavision.com

Get ahead. Stay ahead.

Auteur:
Thomas van IJzendoorn

Breda 18-7-2012

Studentgegevens

Thomas van IJzendoorn
Studnr: 2140313
thomas@digimedia-design.nl
Schooljaar: 2011-2012

Stagebedrijf

Nokavision Software B.V.
's-Hertogenbosch
www.nokavision.com

Opdrachtgever

dhr. Rob Steenvoorden
r.steenvoorden@nokavision.com

Afstudeerbegeleider vanuit bedrijf

dhr. Pascal Naber
p.naber@nokavision.com

Stagebegeleider Fontys

dhr. Joost Kant
joost@kantsoftware.com

Voorwoord

Na een aantal jaren werken en studeren is het dan eindelijk zo ver, het moment van afstuderen. Een goede afstudeeropdracht vinden is hierbij van wezenlijk belang. De eigen werkomgeving, Nokavision Software, is de logische plaats om te zoeken naar mogelijkheden en hier is een mooie opdracht uitgekomen.

Op het moment dat ik begon aan de afstudeeropdracht had Nokavision net besloten om in house ontwikkelde software aan te gaan sluiten op verschillende ERP systemen door middel van een plugin structuur. Het ontwikkelen van deze structuur en adapters was een goede match met de vereisten van de afstudeeropdracht en het bleek geen enkel probleem om deze werkzaamheden in de vorm van een afstudeeropdracht uit te gaan voeren.

Tijdens het traject ben ik bijgestaan door verschillende personen en ik wil deze daar graag voor bedanken:

Rob Steenvoorden
Voor het aanbieden van de afstudeeropdracht en de faciliteiten om deze opdracht uit te voeren.

Pascal Naber
Voor het geduldig begeleiden van mijn afstudeerproces, het bieden van een enorme bron van kennis en ervaring en het verbreden van mijn technische kennis en inzicht.

Joost Kant
Voor het begeleiden van de stage vanuit Fontys en het reviewen van mijn documenten.

Inhoudsopgave

Afkortingen en begrippen

Adapter	Design pattern waarbij een koppeling wordt gelegd tussen twee componenten die geen informatie kunnen uitwisselen.
Backend	In de context van dit document: de webapplicatie in IIS en Ria services en de ontwikkelde adapters.
Design Pattern	Een ontwerppatroon of patroon in de informatica is een generiek opgezette softwarestructuur, die een bepaald veelvoorkomend type software-ontwerpprobleem oplost.
Entity Framework	Object relational mapping (ORM) framework waarmee een database op een object georienteerde manier kan worden benaderd.
ERP	Enterprise Resource Planning. Bedrijsproces ondersteunende software. In deze context de software waar de administratie in wordt bij gehouden.
Frontend	De user interface. In deze context van dit document: de Silverlight applicatie op de client pc.
IIS	Internet Information Services, software van Microsoft voor het hosten van websites.
IP	Afkorting voor Invoice Processing.
IRIS	Scansoftware.
LINQ	Language Integrated Query. LINQ biedt een werkwijze aan voor een meer uniforme omgang met gegevens uit heel verschillende systemen, bijvoorbeeld gegevens uit een relationele database, een webservice, een XML-bestand of een array. Met LINQ kunnen al deze verschillende soorten gegevens met één op SQL lijkende set taalelementen worden opgevraagd, gemanipuleerd en gecombineerd.
MSMQ	Microsoft Message Queuing. Verzorgt het versturen van berichten via een queue. Wordt door NServiceBus onder water gebruikt.
MVC	Model View Controller. Een vergelijkbare techniek als MVVM waarbij de verschillende onderdelen van een applicatie worden gescheiden. Model -> Database en entiteiten View -> De user interface Controller -> Alle business logica

MVVM	Model View Viewmodel, een techniek om in oa. Silverlight op een gelaagde manier te werken. De database, de views en de business logica worden hiermee gescheiden. Model -> database ViewModel -> Class met de business logica View -> Het scherm dat de gebruiker ziet
Plugin	In de context van dit document een ander woord voor adapter.
RavenDB	Object georienteerde database software.
RIA Services	Een techniek waarmee Silverlight een actieve verbinding met de IIS backend kan onderhouden. Hierdoor kan er vanuit Silverlight data van en naar de server worden gestuurd zonder dat hier eigen webservices oid. voor moeten worden opgezet.
SOLID	Solid beschrijft een aantal richtlijnen voor programmeurs om overzichtelijke en "clean" code te schrijven. Een van de eisen van de opdrachtgever is om de geschreven C# code zo "solid" mogelijk te maken.
TFS	Team Foundation Server. Software van Microsoft voor versiebeheer van in Visual Studio gemaakte projecten / software.
Unit Test	Een techniek voor .NET voor het maken van tests. Deze unittests zijn in staat verschillende classes en blokken code te testen aan de hand van vaste data en verwachtte output.
Unity Framework	Een inversion of control framework. Hiermee kunnen dynamisch .dll libraries worden ingeladen in een applicatie.
XML	Extensible Markup Language. Een standaard voor het weergeven van gestructureerde data

Samenvatting

De afstudeeropracht is uitgevoerd bij Nokavision Software, waar ik voorafgaand aan de stage in dienst ben getreden. Nokavision is een klein bedrijf met 13 personeelsleden met als core business het leveren van document-output software en consultancy. Daarnaast ontwikkelt ze in-house eigen software m.b.v. Silverlight.

Het belangrijkste softwarepakket dat wordt ontwikkeld is Invoice Processing, factuurverwerking software. Deze software wordt ontwikkeld in Silverlight op het Microsoft .NET framework 4.0. De gebruikte taal is C#.
In het traject van factuurverwerking wordt op een gegeven moment de factuur geboekt. Deze boeking bestaat uit meerdere regels waarbij een grootboekrekening, kostenplaats en bedrag kan worden gekozen. De gehele boeking kan worden geëxporteerd naar een tekstbestand in een vast formaat.

Tijdens het verkoop traject is gebleken dat de inhoud van een boeking per klant kan variëren en afhankelijk is van de inrichting van het ERP systeem. De standaard set van grootboekrekeningen en kostenplaatsen is niet meer voldoende.
De boekingen moeten tevens worden geëxporteerd naar het ERP systeem, welke leidend is voor de administratie. De export moet worden gedaan in een door het ERP systeem gedicteerd formaat.

Vanuit Nokavision is de opdracht ontstaan om te onderzoeken wat de mogelijkheden zijn voor het asynchroon communiceren met externe systemen, bijvoorbeeld d.m.v. een adapter. Allereerst moet er een geschikt model worden gevonden voor het ontwikkelen van een adapter, daarna moeten er meerdere adapters worden ontwikkeld.

In het doorlopen traject is gestart met onderzoek naar de beste methode om plugins te ontwikkelen voor Invoice Processing. Vanuit dit onderzoek heeft een pakketselectie plaatsgevonden om het meest geschikte framework te selecteren. Met dat framework is vervolgens een proof of concept ontwikkeld.

De meest geschikte methode is het gebruik van een servicebus gebleken. Een servicebus stelt je in staat om op een ontkoppelde wijze verschillende componenten met elkaar te laten communiceren. Deze communicatie verloopt via berichten die via MSMQ worden verzonden.

De uiteindelijke selectie voor een framework is NServiceBus geworden, met dit framework kan een servicebus worden geïmplementeerd.
Met NServiceBus is vervolgens een backend gemaakt met verschillende adapters die worden aangeroepen vanuit Silverlight.
Er zijn binnen het project meerdere adapters ontwikkeld en in productie geplaatst bij klanten van Nokavision.

Summary

The internship is performed at Nokavision Software, a small company with 13 employee's. Before I started the project I was already working for Nokavision as a software developer.
Nokavisions core business is supplying document-output software and consultancy, but they also develop their own software in Silverlight.

Invoice Processing is their main software which, as the name says, is used for processing invoices. This application is written in Silverlight.
While processing invoices, one of the steps will be booking the invoice. A booking can consist of multiple items like a ledger account or a cost unit. The software only supports a fixed set of these items and is able to export a booking in a fixed format.

Multiple customers want to be able to use their own items which they also use in their ERP. Also, they want to be able to export the booking data to their own ERP.
To achieve this, Nokavision came up with the idea to create multiple plugins for Invoice Processing, one plugin for each ERP.

This is where the project has started.
First off, research took place to identify the best way to create plugins for this application. Since this is written in Silverlight it's not as easy as it looks.
From there a search was started to find the best available framework to do the job.
With this framework a proof of concept has been made to demonstrate and double check the possibilities and ease of use.

NServiceBus is the selected framework which is a framework to implement a servicebus.
With NServiceBus a backend with multiple plugins is created. This backend is accessed by the Silverlight application.

At this moment, multiple plugins are created and are used by the customers in their production environment.

1) Inleiding

Ter onderbouwing van mijn werkzaamheden bij Nokavision ten behoeve van de afstudeeropdracht is dit document ontwikkeld.

Er wordt in dit document een beschrijving van het doorlopen traject gegeven.
Dit traject is gestart vanuit de opdracht: "Onderzoek verschillende manieren om vanuit de bestaande Invoice Processing een koppeling te maken met verschillende externe (ERP) systemen"
Om dit te bereiken is onderzoek uitgevoerd naar verscheidene methoden en heeft pakketselectie plaatsgevonden. Tevens is nagedacht over de inrichting van de ontwikkelomgeving en de installatie bij de klant.

Het traject wordt in dit document in detail weergegeven. In hoofdstuk 1 wordt een korte inleiding gegeven alsmede een kort overzicht van de hoofdstukken uit dit document.
Vervolgens wordt in hoofdstuk 2 een beschrijving gegeven van Nokavision Software, de organisatie waarbinnen de opdracht is uitgevoerd. Tevens laat dit hoofdstuk de organisatie structuur achter Nokavision Software zien.
Na deze beschrijving volgt een overzicht van de opdracht en wordt hier in hoofdstuk 3 inhoudelijk op ingegaan. Er wordt een beeld gegeven van de huidige situatie, de gewenste situatie en de randvoorwaarden waaraan moet worden voldaan.
Na de opdrachtomschrijving wordt in hoofdstuk 4 een beschrijving van het doorlopen traject gegeven.
Een belangrijk onderdeel binnen dit traject is het onderzoek, waarbij aan de hand van deelvragen en door Nokavision gestelde criteria het meest geschikte adapter model is gekozen. Hierop wordt dieper ingegaan in hoofdstuk 5.
Vanuit het onderzoek is gebleken dat het gebruik van een servicebus voor Invoice Processing de meest geschikte methode is om plugins te ontwikkelen. Hiervoor is een framework geselecteerd, NServiceBus. In hoofdstuk 6 zal hier inhoudelijk dieper op worden ingegaan.
Bij een geschikte methode om plugins te ontwikkelen is niet alleen gezocht naar een geschikt framework, tevens is ook nagedacht over een gedegen inrichting van de ontwikkelomgeving. Hoofdstuk 7 geeft inzicht in de inrichting.
Alle wijzigingen / uitbreidingen van de opdracht die zijn doorgevoerd tijdens het traject zijn beschreven in hoofdstuk 8.
Ter afsluiting volgt de conclusie, terug te vinden in hoofdstuk 9.

2) De organisatie

In dit hoofdstuk is een beknopte beschrijving terug te vinden van Nokavision Software, het bedrijf waarbinnen de opdracht is uitgevoerd. Daarnaast wordt een beeld geschetst van de eigen positie binnen dit bedrijf.

2.1) Bedrijfsbeschrijving

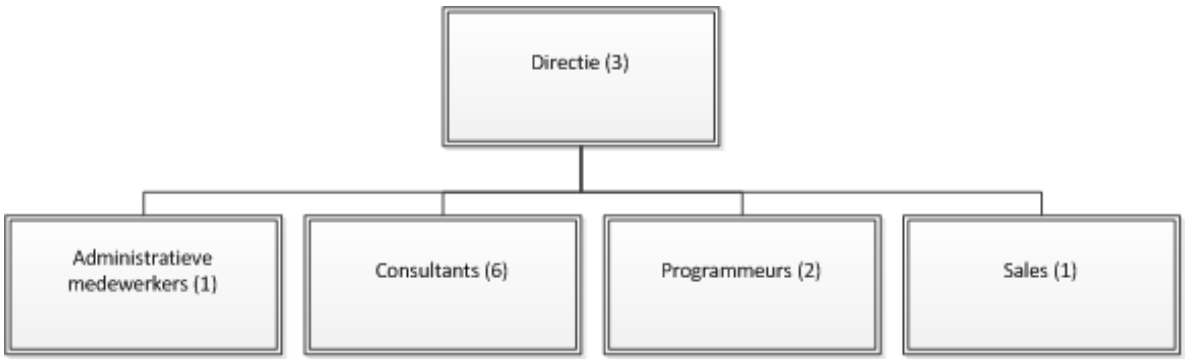
Nokavision Software is een kleine organisatie (13 personeelsleden), begonnen op het gebied van implementatie van Document-output systemen. Gehuisvest in Den Bosch levert Nokavision voornamelijk zijn diensten binnen Nederland.

Sinds een aantal jaren ontwikkelt Nokavision ook in-house software waarbij procesondersteuning het belangrijkste kenmerk is.
De belangrijkste applicaties zijn:

- Een applicatie voor totstandkoming van Reclamefolders
- Een applicatie voor digitale factuur verwerking: Invoice Processing

2.2) Organisatiestructuur

Binnen Nokavision zijn niet "hard" afdelingen gedefinieerd, wel bestaat er een onderverdeling qua functies: consultants, ontwikkelaars en sales. Zie figuur 1.
Met name de ontwikkelaars zullen betrokken zijn bij dit project en worden hierin geleid door dhr. Steenvoorden.
Dhr. Naber is lead developer bij Nokavision en is verantwoordelijk voor de technische invulling van de applicatie. Hij is degene die moet instemmen met de technieken die gebruikt gaan worden. Hiernaast dient nog rekening gehouden te worden met de systeembeheerder, die zeggenschap heeft over de infrastructuur binnen Nokavision. Deze systeembeheerder heeft primair de rol van consultant en is voornamelijk extern werkzaam.



figuur 1.

2.3) Mijn plaats binnen het bedrijf

Binnen Nokavision ben ik in vaste dienst als developer. Mijn primaire werkzaamheden zijn de ontwikkeling, het testen en bugfixen van Invoice Processing.
Hiernaast houd ik mij bezig met het overzetten van Invoice Processing van Silverlight naar MVC3 en html 5.

Een klein onderdeel van mijn werkzaamheden is het ontwikkelen van formulieren en workflows in het pakket Skelta. In Skelta worden klantspecifieke, proces ondersteunende workflows ontwikkeld. Naast verkoop en implementatie wordt hier door Nokavision ook consultancy op geleverd.

3) De opdracht

Er wordt in dit hoofdstuk een korte beschrijving gegeven van Invoice Processing, de door Nokavision ontwikkelde software waar de opdracht op van toepassing is. Daarnaast is een beschrijving van de opdracht alsmede een beschrijving van de huidige situatie terug te vinden.

3.1) Invoice Processing

Invoice Processing is een op het .NET framework ontwikkelde Silverlight applicatie geschreven in C#. Deze applicatie ondersteunt het proces van factuurverwerking binnen een organisatie.
In de meeste organisaties bestaat dit proces uit meerdere stappen die niet per se in onderstaande volgorde worden doorlopen.

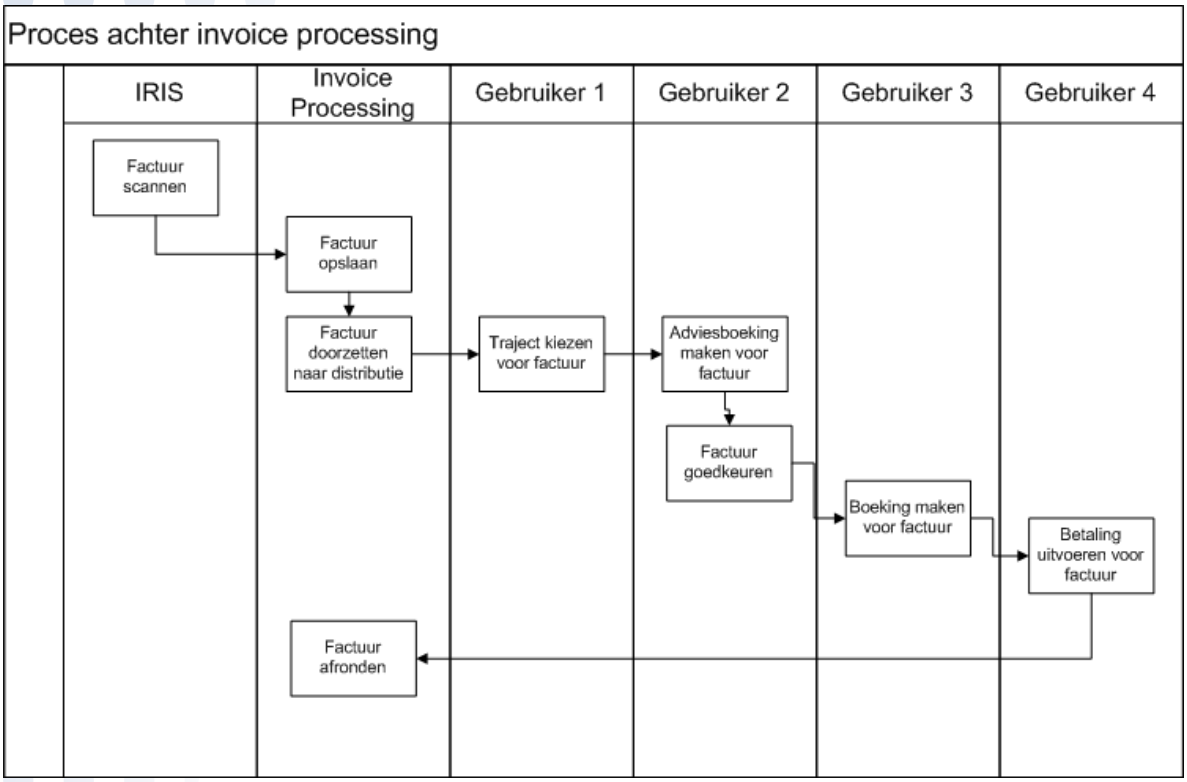
Ontvangst
Tijdens de trajectstap ontvangst wordt de gescande data onderworpen aan een controle door de eindgebruiker. Hierbij kan eventueel verkeerd gescande data worden gecorrigeerd. Over het algemeen hoeft de gebruiker deze stap alleen te verwerken.

Goedkeuring
In deze stap kan de factuur door een of meerdere personen worden goed- of afgekeurd.
De goedkeuring kan per traject anders worden geconfigureerd en kan bestaan uit een of meerdere goedkeuringsstappen. Op deze wijze kunnen facturen in verschillende ordes van grootte door andere gebruikers worden goedgekeurd.

Boeking
Om de factuur op te nemen in de administratie wordt deze geboekt. Dit gebeurd door het aanmaken van boekingsregels voor de gehele factuur of delen hiervan. Bij een boekingsregel wordt vaak een bedrag, btw percentage, grootboekrekening, kostenplaats en btw bedrag opgegeven.

Betaling
Een factuur dient over het algemeen betaald te worden, in de stap betaling kan dit worden gedaan. Tijdens deze stap kan worden gekozen om het volledige factuurbedrag of een deel hiervan te betalen.

Het gebruik in de praktijk
Invoice processing zal door klanten worden gebruikt om facturen door verschillende trajecten heen te leiden. Deze trajecten worden door de klant ingericht en bestaan uit een combinatie van bovenstaande stappen. Over het algemeen geldt dat, in verband met functiescheiding, elke stap door een andere gebruikersgroep wordt uitgevoerd.
De facturen worden door de gebruiker gescand. Tijdens het scanproces worden de facturen automatisch ingelezen in Invoice Processing. Automatisch wordt nu het juiste traject gestart, mits dit door een gebruiker is geconfigureerd. Is dit niet het geval, dan kan de gebruiker het juiste traject bij de factuur ingeven.
Zodra een factuur alle stappen van het gekozen traject doorlopen heeft, wordt deze aangemerkt als “verwerkt”. Op dit moment is de factuur nog wel zichtbaar in de applicatie maar kan niet meer gewijzigd worden.



3.2) Opdrachtomschrijving

Invoice Processing is inmiddels al meer dan een jaar in ontwikkeling.
Een van de openstaande punten is de mogelijheid om te communiceren met externe ERP systemen.

Onderzoek de mogelijkheid om Invoice Processing te verbinden met verschillende externe (ERP) systemen door middel van een adapter / plugin model.
Deze plugins moeten in kunnen haken op events binnen Invoice Processing en data moet geïmporteerd of geëxporteerd kunnen worden. Het formaat van de in- of export wordt gedicteerd door het ERP systeem.

Een van de stappen in dit proces is het maken van een boeking. Het maken van een boeking wordt gedaan met uit het ERP systeem geïmporteerde data (grootboekrekeningen, subrekeningen etc)
Deze boeking, bestaande uit meerdere boekingsregels, moet automatisch geëxporteerd worden naar een extern ERP.

Op dit moment is Invoice Processing aan meerdere klanten verkocht, daarom moet er een tweetal boeking adapters ontwikkeld worden voor:

- JD Edwards
- Omnivers.

3.3) Huidige situatie

Invoice Processing wordt ondersteund door een scan applicatie genaamd IRIS. Deze applicatie scant facturen en zet de factuur data in een specifieke map.
Invoice Processing importeert vervolgens deze data en voegt dit toe als factuur in een eigen database.
Deze facturen gaan hierna het traject in waarbij ze kunnen worden goedgekeurd, geboekt en / of betaald. Dit gebeurt niet noodzakelijkelijkerwijze in deze volgorde, dit is configureerbaar door de gebruiker.

In de huidige applicatie is het reeds mogelijk om boekingen te maken. Er dienen eerst grootboekrekeningen en kostenplaatsen aan de applicatie toegevoegd te worden.

In het boeking scherm wordt voor een factuur een boekingsregel toegevoegd waarbij deze kostenplaatsen en grootboekrekeningen worden gebruikt. Op een regel wordt vervolgens een bedrag en btw percentage geselecteerd. Zo kan per factuur een boeking worden aangemaakt.

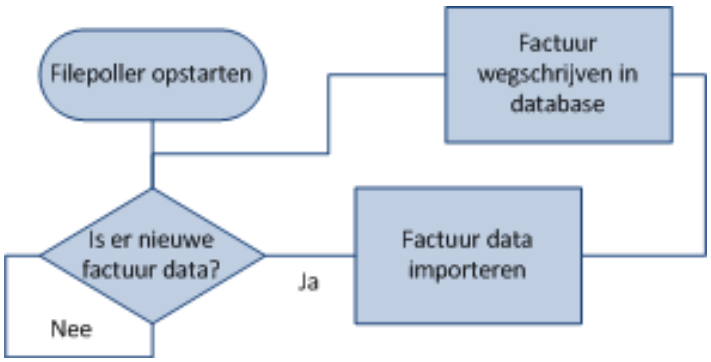
Als alle boekingen zijn klaargezet moet vervolgens naar een ander scherm worden genavigeerd. Op dit scherm worden alle facturen geselecteerd die worden geëxporteerd.
Voor de export wordt op de server een proces gestart die in een vaste directory een .txt bestand wegschrijft, de Silverlight client is tijdens dit proces onbruikbaar.

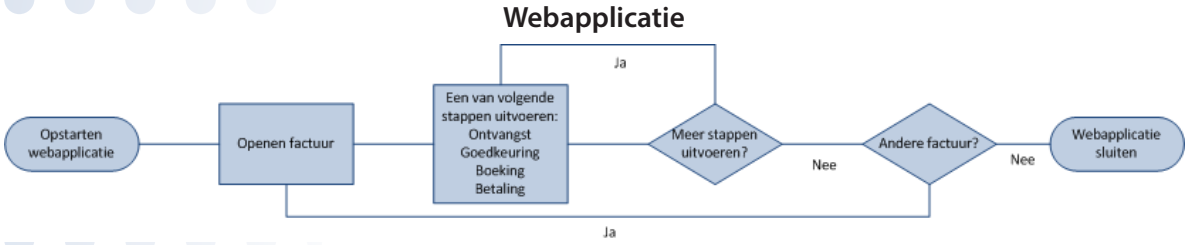
De huidige versie van Invoice Processing bestaat uit meerderen onderdelen, Iris, de Filepoller en de Silverlight applicatie. Deze laatste is onder te verdelen in de frontend (clientside) en backend (serverside).

De onderdelen doen als volgt hun werk:

IRIS

Filepoller





3.4) Probleemstelling

Aan de huidige situatie zitten een aantal grote nadelen:

- De boekingen bestaan altijd uit grootboekrekeningen en kostenplaatsen, dit kan niet per klant worden geconfigureerd of gewijzigd en heeft geen relatie met de inrichting van het ERP van de klant.
- De grootboekrekeningen en kostenplaatsen worden handmatig toegevoegd aan de applicatie. Als de klant een ERP systeem heeft moet deze informatie dubbel bijgehouden worden.
- De boekingen worden nu handmatig geëxporteerd, niet automatisch.
- Het exporteren van de boekingen valt buiten de trajectstap boeking. Een verwerkte boeking betekent niet noodzakelijk een geëxporteerde boeking.
- De export resulteert in een .txt file met een vast format, hierin wordt geen rekening gehouden met de verwerking naar rapportages of een ERP.
  De data van deze export staat vast en is dus per klant gelijk.
- Het laten klaarzetten van een adviesboeking door een andere gebruiker is niet mogelijk.
- De gebruikers moeten wachten op de export, dit gaat synchroon.
- Bij fouten tijdens de export krijgen de gebruikers een error te zien met een te technische en vaak irrelevante melding (maw. de gebruiker kan niets doen om deze fout te herstel len, bijvoorbeeld bij exporteren naar een niet bestaande directory)

3.5) Doelstellingen

De primaire doelstelling van het project is het ontwikkelen van een adapter model en meerdere adapters. Dit adaptermodel moet een generieke basis voor de adapters bevatten waarvanuit iedere nieuwe adapter kan worden ontwikkeld.

De meerwaarde van het adaptermodel kan worden uitgedrukt aan de hand van de volgende factoren:

- Het ontwikkelen van een plugin d.m.v. het model moet sneller zijn dan het van scratch ontwikkelen van een plugin.
- Werkoverdracht en samenwerking moet vereenvoudigd worden.
- Het toevoegen of update van plugins moet on-the-fly kunnen gebeuren.
  Met andere woorden, de plugin moet geupdate kunnen worden zonder dat de gebruiker dit merkt en zonder dat er data verloren gaat.

4) Het traject

In dit hoofdstuk wordt een globale beschrijving gegeven van de traject dat is doorlopen tijdens het uitvoeren van de opdracht. De hierna volgende hoofdstukken gaan dieper in op de onderwerpen, met name het onderzoek. Er zal kort worden beschreven wat er is gedaan tijdens de volgende fasen:

- Onderzoek
- Proof of concept
- Ontwikkeling
- Testen
- Revisioning
- Deployment

4.1) Onderzoek

De meest geschikte methode voor de ontwikkeling van adapters is onderzocht aan de hand verschillende deelvragen. In het hoofdstuk "onderzoek" zal hier dieper op ingegaan worden.

Vanuit de deelvragen zijn een aantal bestaande methoden onderzocht. Aan de hand van een door Nokavision opgestelde lijst met eisen aan de plugins is vervolgens een scoretabel samengesteld. Hierin wordt een score gegeven voor de mate waarin de methode voldoet aan deze eisen.

Aan de hand van deze scoretabel is vervolgens een keuze gemaakt uit de methoden. Voor deze methode is daarna onderzocht welke bestaande oplossingen, frameworks of software al beschibaar is en in welke mate er documentatie en ondersteuning beschikbaar is.

4.2) Proof of concept

Voor het meest passende framework is vervolgens een proof of concept ontwikkeld waarmee is getest of dit framework inzetbaar is in de soft- en hardware omgeving van Nokavision. Daarnaast is beoordeeld of de aanwezige documentatie en ondersteuning aan de verwachtingen voldoet.

4.3) Ontwikkeling

Nadat het framework is geselecteerd is er gestart met het ontwikkelen van de plugins. Dit is onderverdeeld in 2 delen, de ontwikkeling van de schermen voor de Silverlight client en de ontwikkeling van de adapters op de backend.
Er is eerst gestart met de boeking adapter voor Omnivers om hiermee de eerste klant te kunnen bedienen.

4.4) Testen

Na de eerste fase van ontwikkeling is er gestart met unittesten en gebruikerstesten. Aan de hand van de test resultaten zijn er voornamelijk wijzigingen aangebracht in de Silverlight schermen en de "base class" voor de schermen.

Vanuit deze update is een nieuwe testrelease uitgerold waarmee opnieuw is getest. Dit proces heeft zich een aantal malen herhaald tot zowel de schermen als de adapters naar behoren werkten.

4.5) Revisioning

Tijdens het ontwikkelen en testen van de JDEdwards adapter zijn nog wijzigingen aangebracht in het adaptermodel en de structuur van de projecten in de ontwikkelomgeving. Dit is gedaan omdat er een groter verschil in de adapters zit dan vooraf ingeschat.

Er is op dat moment in overleg besloten om per adapter meerdere projecten aan te maken. Hiernaast is een basis project gecreëerd met gedeelde classes. Dit heeft het geheel generieker gemaakt en voorkomt duplicate code. Daarnaast wordt hiermee het onderhouden en uitrollen van de applicatie eenvoudiger.

Na deze updates is vervolgens voor beide adapters opnieuw het test traject doorlopen.

4.6) Deployment

De adapters zijn na implementatie aan een acceptatie test onderworpen. Om deze test te kunnen uitvoeren is de volledige applicatie uitgerold naar zowel een productie als een test omgeving.
Uiteindelijk is IP met de verschillende adapters uitgerold bij meerdere klanten en in productie genomen.

5) Onderzoek

Dit hoofdstuk weidt uit over het onderzoek dat is uitgevoerd bij aanvang van de opdracht.

Het onderzoek is opgedeeld in 2 deelonderzoeken. In het eerste deelonderzoek is onderzoek gedaan naar de meest passende methode om adapters te ontwikkelen. Hierna heeft er in beperkte mate pakketselectie plaatsgevonden om tot het meest geschikte framework te komen.

Aan de hand van specifieke onderzoeksvragen, een scoretabel voor de methodes en een proof of concept is de beste techniek geselecteerd voor het ontwikkelen van adapters voor Invoice Processing.
In de scoretabel zijn de methodes tegen de eisen van de opdrachtgever afgezet. Hieruit ontstaat een beeld wat de meest passende methode is.

Voor de meest passende methode is vervolgens onderzocht wat het meest passende framework is. Van deze frameworks zijn de voor- en nadelen tegen elkaar en de eisen van de opdrachtgever afgezet en is een selectie gemaakt voor de meest passende match.
Met dit framework is een proof of concept ontwikkeld waarmee is beoordeeld of deze theoretisch beste match ook praktisch goed bruikbaar is.

5.1) Deelvragen

Bij het starten van het onderzoek zijn een aantal onderzoeksvragen vastgesteld, deze vormen de basis van het onderzoek. Deze deelvragen zijn ontstaan uit gesprekken met de opdrachtgever en uit eigen ervaring.

Hieronder volgt een overzicht van de vragen en de uitwerking per vraag:

- Wat zijn de beperkingen van Silverlight gezien Silverlight normaal gesproken client side werkt.
- Welke methoden / modellen bestaan er om plugins te ontwikkelen.
- Welke bestaande componenten zijn er die kunnen worden gebruikt voor de adapter/plugins.
- Welke functionaliteit hebben de adapters met elkaar gemeen en welke onderdelen zijn specifiek.
- Op welke manier kan de configuratie van de adapters het best worden opgeslagen in de Silverlight applicatie
- Wat is de beste methode om de informatie van een boeking op te slaan in het huidige domein model van Invoice Processing
- Welke layout kan het best worden gebruikt voor de schermen in Silverlight

5.2) Wat zijn de beperkingen van de client side werking van Silverlight.

Aan de hand van gesprekken met dhr. Naber en eigen ervaringen met Silverlight kunnen de volgende conclusies getrokken worden:

- De adapters moeten serverside werken, omdat Silverlight client side werkt. Er is niet noodzakelijk verbinding tussen het ERP systeem in het klantnetwerk en Silverlight op de client pc.
- Door middel van RIA Services kan er code serverside worden uitgevoerd, dit wordt dan

wel binnen het IIS worker process gedaan. RIA Services is een verzameling libraries die gratis te downloaden zijn en automatisch integreren met Visual Studio.
- IIS heeft een vaste tijd waarna alle processen gerecycled worden. Hierdoor kunnen processen binnen de IIS worker thread onverwachts worden afgesloten. Het proces van de adapter moet buiten IIS draaien, bijvoorbeeld in een Windows service.

5.3) Welke methoden / modellen bestaan er om plugins te ontwikkelen.

Aan de hand van eigen ervaringen en gesprekken met dhr. Naber zijn een aantal methoden benoemd waarmee plugins kunnen worden ontwikkeld:

- Dependency Injection, het inladen van een dll per adapter, deze wordt door middel van een interface aangesproken.
- Skelta, software waarin proces / bedrijfsondersteunende workflows ontwikkeld en "gehost" worden. Deze workflow kan met custom controls specifieke taken uitvoeren zoals het exporteren van een boeking.
- Servicebus, een messaging techniek waarbij berichten uitgewisseld worden tussen applicaties via een queue.

5.4) Welke bestaande componenten of frameworks zijn er beschikbaar voor de adapters.

Invoice Processing gebruikt Dependency Injection (DI) door middel van Unity. Aan de hand van ervaringen van dhr. Naber en aan de hand van resources op internet zijn een aantal DI frameworks benoemd.
De frameworks voor een servicebus zijn gevonden via internet.

Voor dependency injection zijn de bekende frameworks:
- Unity
- Spring
- Castle windsor
- Autofac
- Ninject

Voor een servicebus implementatie zijn de volgende frameworks beschikbaar:
- NServiceBus
- Masstransit
- RhinoBus

5.5) Welke functionaliteit wordt gedeeld door de adapters en welke onderdelen zijn specifiek.

Per adapter komen de volgende zaken overeen:

- Alle adapters moeten per factuur een boeking kunnen maken, opslaan en exporteren naar het ERP systeem.
- Elke boeking bestaat uit 1 of meerdere boekingsregels en een boekingsperiode.

- Alle gegevens die gebruikt worden op een boekingsregel (kostenplaats, grootboekrekening etc) moeten worden ingeladen vanuit het ERP systeem.
- Er moet per adapter een adviesboeking gemaakt worden door een andere gebruiker dan de persoon die de boeking doet.
- Een adviesboeking bestaat uit dezelfde gegevens als de boeking, behalve de periode.
- De gegevens die per adviesboekingsregel worden ingesteld zijn per gebruiker uniek en moeten per gebruiker per bedrijf instelbaar zijn.
- De adviesboekingsregels kunnen door de gebruiker ingeladen worden tijdens de trajectstap boeking.
- De boeking wordt aangemaakt en geëxporteerd tijdens de trajectstap boeking.
- De adviesboeking wordt aangemaakt tijdens de trajectstap goedkeuring en is optioneel.

De verschillen:

- De gegevens van een boekingregel verschillen per adapter, hierdoor zijn zowel de schermen voor de gebruiker als de geëxporteerde data verschillend.
- De manier van exporteren kan per adapter verschillen, dit is afhankelijk van de manier waarop het ERP systeem data kan inlezen. Bijvoorbeeld: een .txt / .csv bestand of webservices.

5.6) Wat is de beste methode om de informatie van een boeking op te slaan in het huidige domein model van Invoice Processing.

Zowel de boeking als de adviesboeking moeten per factuur worden opgeslagen. Dit kan per adapter verschillen en de data moet generiek worden opgeslagen.
Er worden 2 tabellen aan de database van Invoice Processing toegevoegd, een voor boeking en een voor adviesboeking. In deze tabellen zullen per factuur in xml formaat de boekingsregels worden opgeslagen.

5.7) Welke layout kan het best worden gebruikt voor de schermen in Silverlight.

Voor de schermen wordt onderscheid gemaakt tussen boekingen en adviesboekingen.
Een adviesboeking wordt gedaan voordat de definitieve boeking wordt uitgevoerd.
De gebruikers die de boekingen en adviesboeking maken kunnen verschillen.

Door Nokavision is besloten dat de schermen een layout krijgen die overeenkomt met Invoice Processing, de volgende elementen zullen op de schermen zichtbaar zijn:

- Een dropdown list voor een boekjaar met periode (alleen bij boeking, niet bij adviesboeking)
- Een grid (tabel) voor de boekingsregels
- Een veld voor de validatiemeldingen
- Buttons voor:
  - Opslaan
  - Verversen
  - Annuleren
  - Verwerken

- Nieuwe boekingsregel
- Boekingsregel verwijderen
- Adviesboekingen ophalen (alleen bij boeking)
- Valideren
- Navigatie door set met facturen
- Standaard maken van de boeking voor de leverancier van de factuur (alleen boeking scherm)
- Inladen standaard boeking (alleen boeking scherm)

5.8) Vergelijking methodes

Om de verschillende methodes met elkaar te vergelijken is er eerst een overzicht, zichtbaar op de volgende pagina, gemaakt van de voor en nadelen. De informatie hiervoor komt uit eigen ervaringen, ervaringen van dhr. Naber en van internet.
Vanuit deze lijst is een scorematrix opgesteld met een score van 0 - 10 per onderdeel.
Hiervoor geldt: hoger is beter.

Dependency injection

Voordelen	Nadelen
• Geen kosten • Eenvoudig te ontwikkelen • Geen specifieke kennis vereist • Testbaar door unittests • Versiebeheer door TFS	• Afhankelijk van IIS worker process • Slecht schaalbaar • Bij errors is het lastig opnieuw uit te voeren • Uitvoeren is alleen asynchroon bij gebruik threads

Servicebus

Voordelen	Nadelen
• Lage kosten • 100% asynchroon • Geen beperkingen IIS • Testbaar door unittests • Versiebeheer door TFS • Geen specifieke kennis vereist • Bij errors is opnieuw uitvoeren eenvoudig • Verwerking 100% serverside • Goed schaalbaar • Publisher en subscriber doen dit per type bericht met specifiek versienummer • Transactioneel • MSMQ standaard aanwezig in windows server	• Per listener apparte service • afhankelijkheid MSMQ • Afhankelijkheid updates van leverancier • MSMQ berichten maximaal 4KB groot

Skelta

Voordelen	Nadelen
• 100% asynchroon • Goed schaalbaar • Bij errors is opnieuw uitvoeren eenvoudig • Versiebeheer tussen workflows maakt updaten eenvoudig en houdt de impact klein • Verwerking 100% serverside	• Erg kostbaar in verhouding tot IP • Configuratie is lastig t.o.v. alternatieven • Slecht te testen • Vereist kennis van Skelta • Fouten worden niet teruggekoppeld naar IP • Geen versiebeheer TFS, geen automatische backup • Skelta bestaat uit meerdere services, dit creëert extra afhankelijkheid

5.9) Eisen en scoretabel

In samenspraak met de opdrachtgever is een lijst met eisen aan de adapters ontstaan. Deze is hieronder in een MoSCoW tabel gezet. Omdat het laatste niveau in deze methode niet is gebruikt is deze uit de tabel weggelaten.

Vanuit deze tabel is de score tabel gemaakt. In deze score tabel wordt per item uit de eisenlijst een score van 0 - 10 gegeven voor de mate waarin de methode hieraan voldoet. Ook hier geldt hoger is beter. De informatie waarmee de scoretabel is ingevuld komt deels uit eigen ervaringen en deel van een inschatting vanuit informatie van internet.

Vanuit de scoretabel is een definitieve tabel gemaakt waarbij de score op de onderdelen vermenigvuldigd is met de wegingsfactor uit de moscow tabel.

Voorbeeld:
Lage kosten is een should have, dus wegingsfactor 2. De score voor dependency injection is 10 de eindscore is dus 20.

Wegingsfactoren:
Must have 3
Should have 2
Could have 1

De MosCoW tabel en score overzichten worden op de volgende pagina's weergegeven.
In de overzichten gebruikte afkortingen:

DI = Dependency Injection
SB = ServiecBus

	Must have	Should have	Could have
Lage aankoopkosten		X	
Asynchrone verwerking van o.a. boeking	X		
Grote eenvoud configuratie		X	
Zo min mogelijk afhankelijkheid andere software		X	
Hoge mate van testbaarheid	X		
Geen gespecialiseerde kennis nodig			X
Niet beperkt door IIS		X	
Grote mate van schaalbaarheid			X
In TFS onder te brengen		X	
Bij errors mogelijk om actie opnieuw uit te voeren	X		
On the fly aanpasbaar			X
Serverside verwerking	X		
Ingebouwde foutafhandeling	X		

Scoretabel

	DI	SB	Skelta
Kosten (hoge score = lage kosten)	10	8	0
Asynchroon (hoge score = meer asynchroon)	5	10	10
Eenvoud configuratie (hoge score = meer eenvoud)	10	8	2
Afhankelijkheid software (hoge score = minder afhankelijk)	10	5	5
Testbaarheid (hoge score = beter testbaar)	10	10	5
Gespecialiseerde kennis (hoge score = minder specifieke kennis nodig)	10	8	5
Beperkt door IIS (hoge score = minder beperkt)	5	10	5
Schaalbaarheid (hoge score = beter schaalbaar)	5	10	10
In TFS onder te brengen (hoge score = meer compatible met TFS)	10	10	0
Bij errors mogelijk om actie opnieuw uit te voeren (hoge score = eenvoudiger opnieuw uitvoeren)	5	10	10
On the fly aanpasbaar (hoge score = beter makkelijker aanpasbaar)	0	10	10
Serverside verwerking (hoge score = meer serverside verwerking)	10	10	10
Foutafhandeling (hoge score = betere standaard foutafhandeling en betere integratie notificaties in Silverlight)	10	10	0

Definitieve scores met wegingsfactor

	DI	SB	Skelta
Kosten (hoge score = lage kosten)	20	16	0
Asynchroon (hoge score = meer asynchroon)	15	30	30
Eenvoud configuratie (hoge score = meer eenvoud)	20	16	4
Afhankelijkheid software (hoge score = minder afhankelijk)	20	10	10
Testbaarheid (hoge score = beter testbaar)	30	30	15
Gespecialiseerde kennis (hoge score = minder specifieke kennis nodig)	10	8	5
Beperkt door IIS (hoge score = minder beperkt)	10	20	10
Schaalbaarheid (hoge score = beter schaalbaar)	5	10	10
In TFS onder te brengen (hoge score = meer compatible met TFS)	20	20	0
Bij errors mogelijk om actie opnieuw uit te voeren (hoge score = eenvoudiger opnieuw uitvoeren)	15	30	30
On the fly aanpasbaar (hoge score = beter makkelijker aanpasbaar)	0	20	20
Serverside verwerking (hoge score = meer serverside verwerking)	30	30	30
Foutafhandeling (hoge score = betere standaard foutafhandeling en betere integratie notificaties in Silverlight)	30	30	0
TOTAAL:	**225**	**270**	**164**

DI = Dependency Injection
SB = ServiecBus

5.10) Selectie framework

Het gebruik van een servicebus heeft, afgezet tegen de eisen, de hoogste score en wordt de techniek waarmee de adapters worden ontwikkeld.
In een eerdere fase van het onderzoek is gekeken naar bestaande frameworks voor implementatie van een servicebus:

- NServiceBus
- Masstransit
- RhinoBus

Per framework is een lijst met voor en nadelen samengesteld. Op basis hiervan is een selectie gemaakt voor het meest passende framework.

Voor elk framework geldt als nadeel de afhankelijkheid van MSMQ. Er is voor de frameworks geen scoretabel opgesteld. In overleg met Nokavision zijn de voor- en nadelen uiteen gezet en is besloten met welk framework een proof of concept gemaakt wordt.

NServicebus

Voordelen:

- De gratis versie is uitstekend geschikt voor de doeleinden binnen Invoice Processing. De beperking van 1 bericht per seconde is geen bottleneck en de verplichting om wijzigingen in de code te publiceren is in deze licentie niet opgenomen. De volledig open source versie kent deze verplichting wel.
- Integratie met elke vorm van applicatie
- Uitgebreide lijst met samples te downloaden
- NServiceBus Yahoo forum wordt intensief gebruikt, dit betekent dat er een actieve community is en dat de reactietijd bij vragen waarschijnlijk klein is.
- Github repository met lijst met bugfixes. Hier is het eenvoudig terug te vinden welke issue of feature in welke versie wordt aangepakt. Zelf issues toevoegen is ook mogelijk.
- Open source software, een kijkje onder de motorkap is mogelijk, ook bij de samples.
- Documentatie voor standaard uitrol is compleet.
- Extra features voor unittesting
- De bijgeleverde host.exe zorgt voor automatisch installatie van de queue's en Windows services indien nodig.

Nadelen:

- Voor de meer geavanceerde setups is het lastig de juiste informatie te vinden
- Afhankelijk van RavenDB bij "out of the box" installatie

MassTransit

Voordelen:

- Gratis
- Github repository met lijst met bugfixes. Hier is het eenvoudig terug te vinden welke

issue of feature in welke versie wordt aangepakt. Zelf issues toevoegen is ook mogelijk.

Nadelen:

- Weinig samples
- Beperkte documentatie
- Geen grote community

RhinoBus

Voordelen:

- 100% Gratis
- Github repository met lijst met bugfixes. Hier is het eenvoudig terug te vinden welke issue of feature in welke versie wordt aangepakt. Zelf issues toevoegen is ook mogelijk.
- Grote community

Nadelen:

- Weinig samples
- Beperkte documentatie

5.11) Conclusie

Aan de hand van de gestelde criteria is het gebruik van een servicebus het meest aan te raden. Voornamelijk de onafhankelijkheid van IIS, de mogelijkheid om asynchroon te werken, het feit dat MSMQ transactioneel werkt (guaranteed delivery) en de methode voor foutafhandeling is doorslaggevend.

Het grootste nadeel van de opzet wordt het gebruik van meerdere losse applicaties of services. Nokavision heeft aangegeven dat de voordelen niet opwegen tegen de nadelen.
Als de backend / adapter buiten IIS moet draaien bestaat altijd afhankelijkheid van ten minste een losse service.

In de samples van NServiceBus zijn volledige voorbeelden aanwezig die vrijwel direct gebruikt kunnen worden voor de adapter, daarom is gekozen om hiermee een proof of concept te ontwikkelen. Vanuit dit proof of concept is later besloten om de volledige adapters met NServiceBus te ontwikkelen.

Het gebruik van een servicebus levert voornamelijk asynchoniteit en serverside verwerking op, daarom is er gekeken naar de mogelijkheid om meer functionaliteit af te laten handelen door de servicebus.
waardoor de Silverlight client minder van de business logica hoeft te bevatten.

In de toekomst wordt een overstap gemaakt naar MVC, Invoice Processing wordt dan een volledig webapplicatie. Door business logica uit de Silverlight client te verplaatsen naar een backend wordt een eventuele overstap naar MVC eenvoudiger.

6) NServiceBus in details

In overleg met Nokavision, vooral met dhr. Naber, is gekozen voor NServiceBus als framework voor de adapters. In dit hoofdstuk zal een korte uitleg worden gegeven over de werking hiervan.

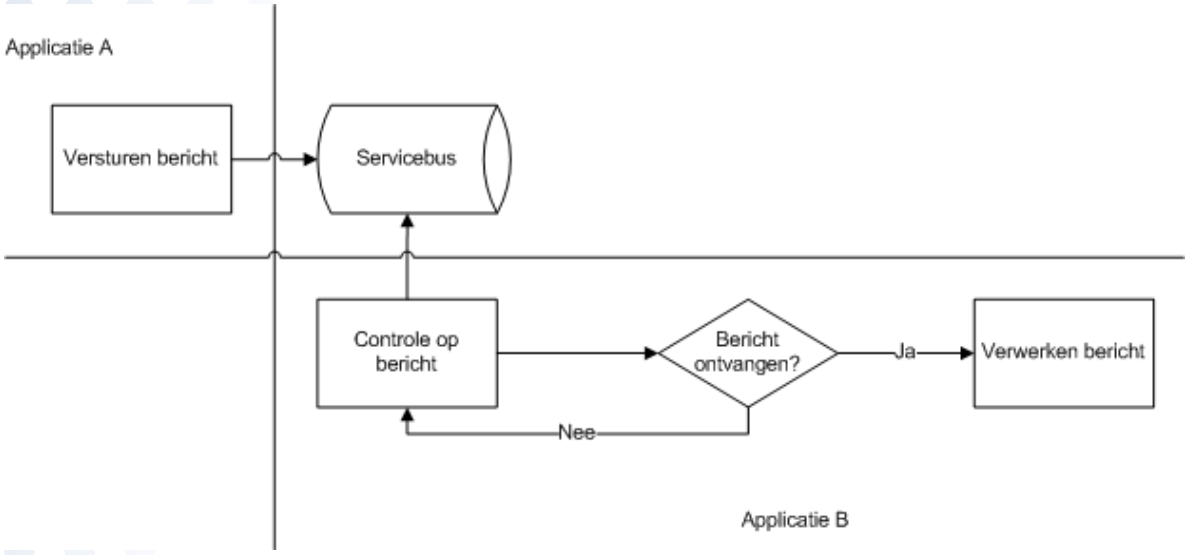
Principe van een servicebus

6.1) De werking van een servicebus:

Applicatie A stuurt een bericht, deze wordt opgeslagen in de bus (queue).

Applicatie B luistert op deze bus en zal elk gevonden bericht proberen te verwerken afhankelijk van de logica in de applicatie.

Eventueel publiceert applicatie B na het verwerken weer een bericht dat kan worden uitgelezen door applicatie A.



6.2) Microsoft Message Queuing

NServiceBus gebruikt onder water voor de opslag van de berichten Micosoft Message Queuing, afgekort MSMQ.

Bij het publiceren van een message plaatst NServiceBus het bericht in de juiste queue. In het configuratie bestand voor de NServiceBus.Host.exe kan per type bericht (het type van de C# class) worden aangegeven in welke queue dit type message moet worden geplaatst.
Bij het uitlezen van een message uit een queue wordt deze altijd verwijderd door MSMQ.

Bij de installatie van de NServiceBus.Host.exe zullen automatisch de juiste queue's worden opgehaald uit de configuratie en worden aangemaakt als deze nog niet aanwezig zijn.

MSMQ kent de beperking dat elk bericht maximaal 4KB groot kan zijn. NServiceBus zal automatisch messages opdelen in 4KB messages indien nodig. Bij het ontvangst zullen deze delen automatisch weer samengevoegd worden en aan de bijbehorende handler worden doorgegeven.

6.3) Messages en Handlers

NServiceBus maakt voor de uitwisseling van data gebruik van messages en handlers.
De messages worden uitgewisseld tussen applicaties via een of meerdere MSMQ queue's en kunnen worden gevuld met data.
Voor elke message kan een specifieke handler met custom business logica worden ontwikkeld. Deze handler is verantwoordelijk voor het verwerken van de data in de message.

Technische opzet:

Elke message heeft een specifiek type (het type van de C# class), bijvoorbeeld "BoekingVewerkt". De message heeft door de programmeur gekozen variabelen, zo heeft de message "BoekingVewerkt" bijvoorbeeld de variabele BoekingData.

Invoice Processing (de RIA Services Backend) maakt een message van dit type "BoekingVerwerkt" en vult deze met de boeking data uit de client.

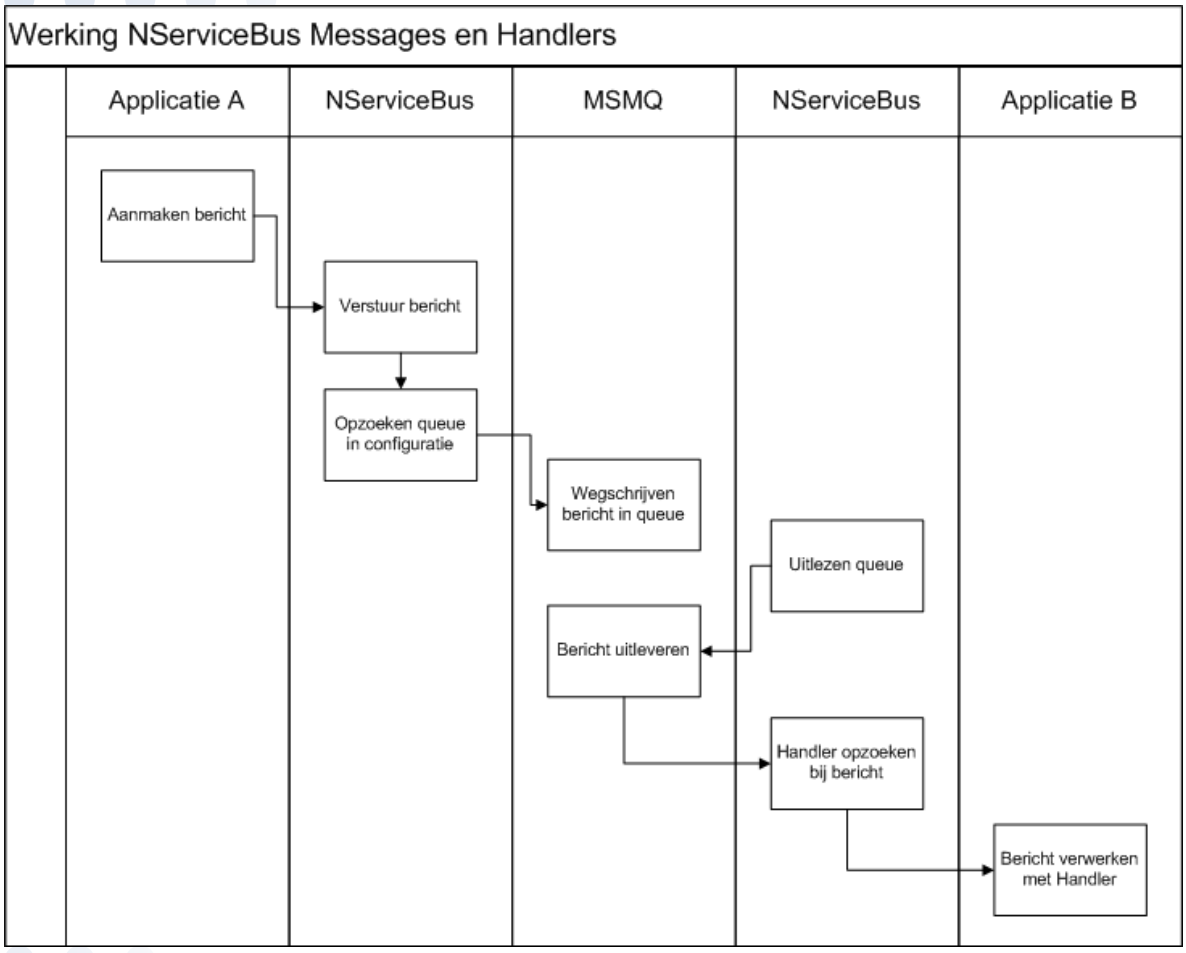
NServiceBus haalt de bijbehorende queue uit de configuratie en plaatst message in deze queue.

In de service op de backend (de dispatcher, zie hoofdstuk "Inrichting") is de Handler gedefinieerd.
Deze handler is een class die de juiste interfaces van NServiceBus implementeert en heeft als eigenschap dat deze messages van het type "BoekingVerwerkt" verwerkt.

NServiceBus controleert iedere x tijd de queue en vind het bericht van type "BoekingVerwerkt" Automatisch wordt juiste handler hierbij gezocht en gestart. Het bericht is nu door MSMQ uit de queue verwijderd.

De handler heeft een methode met de naam Handle met als parameter de Message van het type "BoekingVerwerkt". De inhoud van deze methode wordt bepaald door de programmeur, in dit voorbeeld zou hier code staan om de boeking te exporteren naar het ERP.

Op de volgende pagina wordt de werking van messages en handlers grafisch weergegeven.



6.4) Losse services

NServiceBus is geschikt om met 1 queue en 1 service messages te verwerken van verschillende types. Een service meerdere handlers hebben die deze messages verwerken.

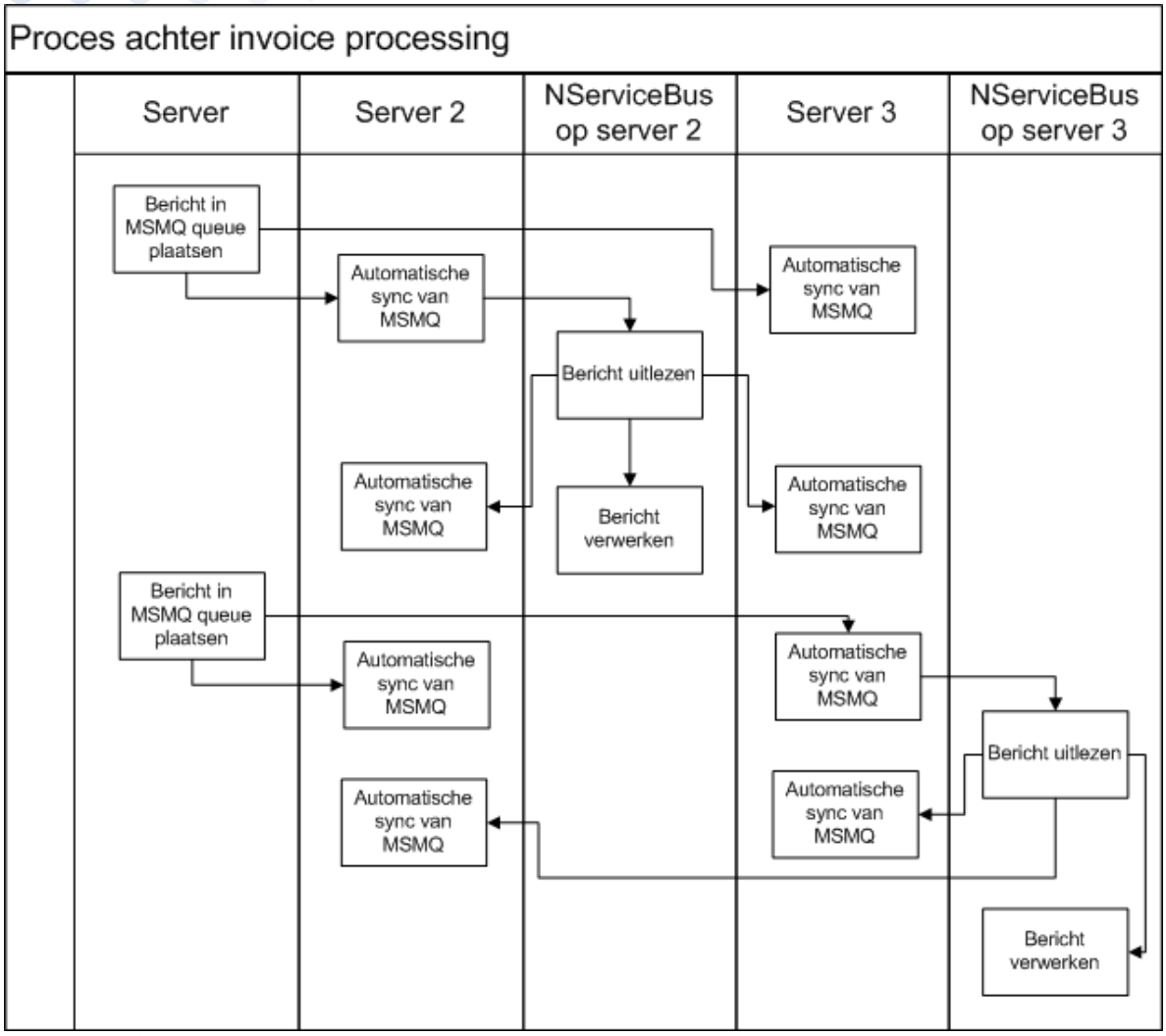
Voor het overzicht is er gekozen per adapter een losse service met meerdere specifieke handlers te ontwikkelen. Daarnaast hebben de adapters verschillende xslt en xsd bestanden.
Deze bestanden transformeren de boekingsdata uit InvoiceProcessing naar een formaat dat het ERP systeem kan importeren.
De xsd bestanden dicteren het formaat van de output, de xslt bestanden zorgen voor de werkelijke transformatie.

Tijdens het uitrollen bij klanten wordt per adapter een service geinstalleerd, dit maakt het mogelijk om meerdere adapters naast elkaar te laten functioneren.

6.5) Schaalbaarheid

MSMQ zorgt voor de nodige schaalbaarheid. Het is mogelijk om op meerdere systemen een queue met dezelfde naam aan te maken.
Deze queue's worden door MSMQ aan elkaar gelinkt. Dit resulteert in gelijktijdige ontvangst van messages op al deze systemen.

Wordt dezelfde adapter service op meerdere systemen gestart dan zullen de berichten verdeeld worden over deze services. Service 1 nagenoeg tegelijk met service 2 een message kunnen verwerken.

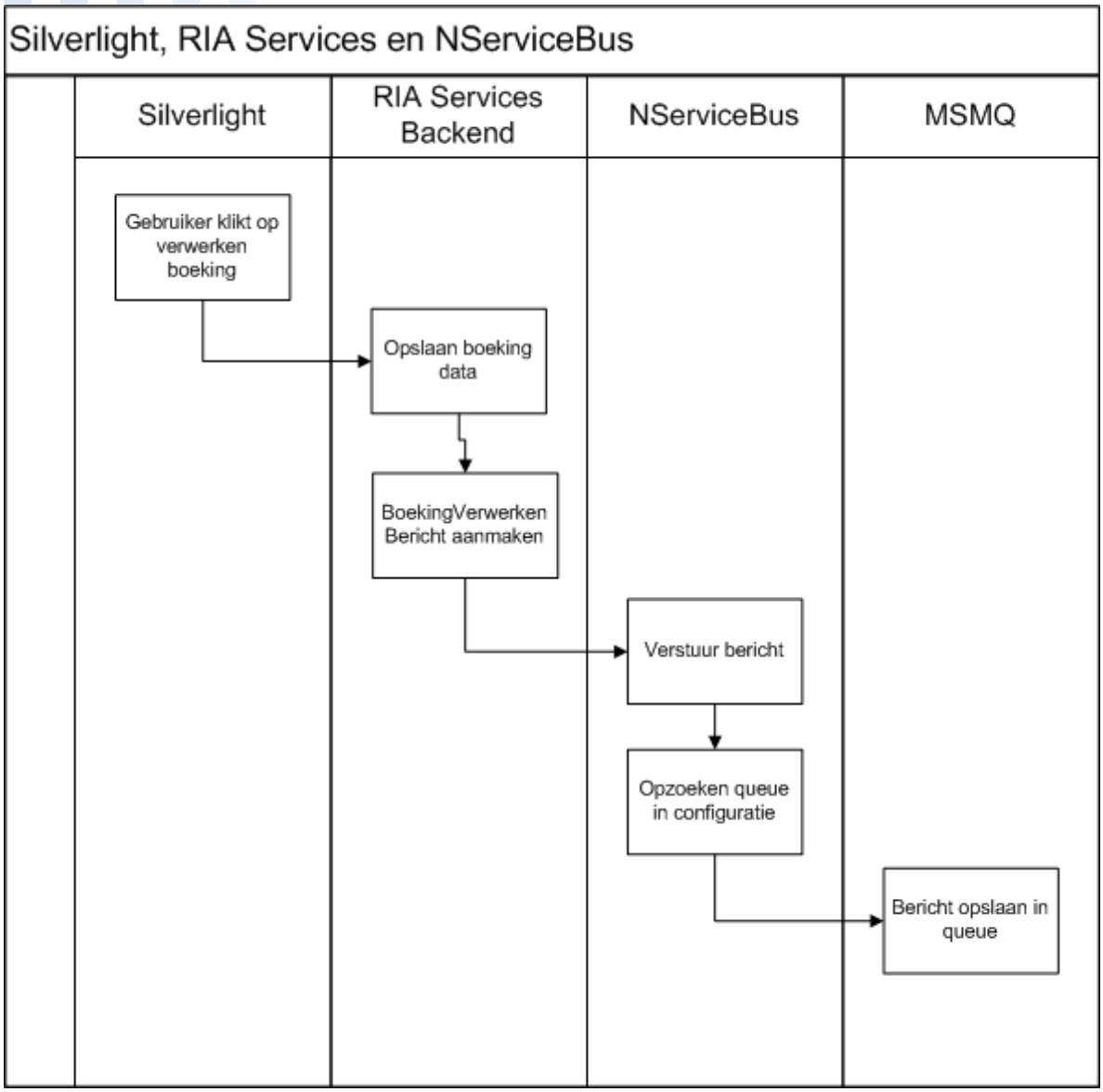


6.6) NServiceBus in combinatie met Silverlight

De techniek dicteert dat Silverlight door middel van RIA Services de messages verstuurd.

De Silverlight client heeft geen toegang tot de queue's op de server omdat deze op de pc van de gebruiker is gestart. De RIA Services backend en de adapters werken op de server om verbinding te kunnen hebben met het ERP systeem. Ook de queue's moeten op de server aanwezig zijn.

Op de volgende pagina wordt dit grafisch weergegeven.



6.7) Unittesting

De messages en handlers worden door programmeurs gebouwd, dit betekent dat deze uitgebreid moeten worden getest. Normaal gesproken kan dit gedaan worden door middel van Unit tests.
Vanwege het gebruik van queue's bij het gebruik van een servicebus is het testen van handlers standaard niet mogelijk. De unittest vereist dat direct na het versturen van een message deze ook als eerste message opgehaald wordt bij het uitlezen van de queue.
Dit hoeft niet noodzakelijk zo te zijn. Het kan voorkomen dat er andere (oudere) berichten in de queue staan. De geteste handler zal op dit moment het verkeerde bericht uit de queue lezen en niet goed worden uitgevoerd.

NServiceBus heeft extra functionaliteit ingebouwd waarmee dit probleem wordt getackled en de handlers getest kunnen worden. Om te unittesten moet in Visual Studio een test project worden toegevoegd middels het standaard template. Na het toevoegen van referenties naar verschillende NServiceBus libraries zijn de handlers testbaar.

6.8) Foutafhandeling

De foutafhandeling van NServiceBus en MSMQ is de basis van de foutafhandeling van Invoice Processing. Mocht het afhandelen van een message een exceptie opleveren, dan zal de message automatisch nogmaals worden verwerkt.
Na een aantal mislukte pogingen zal het bericht door NServiceBus automatisch in de error queue worden geplaatst, waardoor deze niet verloren gaat.

Bij het optreden van een exceptie in een handler wordt automatisch een custom error handling class aangeroepen. Deze class zorgt ervoor dat er in de database van Invoice Processing een exceptie melding wordt toegevoegd. Bij deze melding staat de ID van het bericht dat in de error queue is geplaatst.

NServiceBus heeft standaard een tool meegeleverd waarmee een bericht vanuit de error queue kan worden teruggeplaatst in de originele queue. Hierdoor wordt het bericht nogmaals verwerkt door de bijbehorende handler.

Invoice Processing is in staat om de gebruiker te notificeren van opgetreden fouten via email. Daarnaast is het mogelijk om het bericht nogmaals te laten verwerken.

Een praktijk voorbeeld:

In een handler mislukt het opslaan van een bestand, omdat de netwerk directory tijdelijk niet bereikbaar is. Het verwerken van de message wordt bij een standaard configuratie maximaal 5 keer uitgevoerd.

Na de vijfde mislukte poging wordt het bericht in de de error queue geplaatst en wordt er een exceptie melding aan de database toegevoegd.
De beheerder van Invoice Processing krijgt een email dat deze fout is opgetreden met daarin de message details en een link naar Invoice Processing.

Inmiddels is de directory weer bereikbaar.
Met behulp van een appart scherm in Invoice Processing kan de beheerder nu het bericht terugzetten van de error naar de originele queue. De handler zal nu opnieuw het bericht verwerken en wel het bestand kunnen opslaan.

7) Inrichting

De totale Invoice Processing applicatie bestaat uit meerdere onderdelen: de frontend, backend en database. In dit hoofdstuk zijn details over de verschillende onderdelen terug te vinden.

7.1) Aanvullende informatie over de backend

De backend van de applicatie draait op NServiceBus en is opgedeeld in meerdere services welke los geinstalleerd moeten worden. Deze onderdelen zijn de dispatcher, adapters en filepoller.

Elk van deze onderdelen bestaat uit meerdere projecten, een project met de core classes, een test project en een Host project.
Dit host project is een library die door de NServiceBus.Host.exe wordt ingeladen. Hierin worden ook de settings voor de queue's geconfigureerd. Deze library heeft referenties naar de core classes.

Zodra het Host project in Visual Studio wordt ge-build zal in de bin\debug folder van het project een NServiceBus.Host.exe, een aantal dll bestanden en enkele configuratie bestanden worden geplaatst.
Deze NServiceBus.Host.exe is de executable voor het de service.
(in het dispatcher project zal dit dus de dispatcher opstarten, in het adapter project de adapter etc)

De NServiceBus.Host.exe kan vanuit explorer of de command prompt worden opgestart. Als de juiste parameters worden meegegeven bij het starten zal deze .exe zichzelf automatisch als Windows Service installeren.

De Dispatcher

De dispatcher is de service die aangesproken wordt door de webapplicatie. De dispatcher verstuurt vervolgens de benodigde berichten naar de adapters. De functionaliteit van IP wordt hiermee opgesplitst in meerdere onderdelen:

- De core functionaliteit van Invoice Processing, alle acties die te maken hebben met het traject dat een factuur doorloopt.
- De adapters, alleen voor de communicatie met het ERP systeem.

Bij bepaalde trajectstappen, waaronder boeking, moeten er meerdere entiteiten worden geupdate. Aan de factuur moet een status, audit en historie regel worden toegevoegd.
Van de factuur wordt de boeking data opgehaald en deze zal door de adapter moeten worden geëxporteerd.

De Adapters

Voor elke adapter worden losse projecten toegevoegd, een core project, een test project en een host project. Ook hier build het host project de NServiceBus.Host.exe. Ook deze kan geïnstalleerd worden als een Windows service.

De adapters zijn verantwoordelijk voor alle communicatie tussen externe systemen en IP:

- Inladen inkoopfacturen uit IRIS
- Importeren van leveranciers uit het ERP systeem
- Importeren van gegevens voor de boeking (grootboekrekeningen etc) uit het ERP systeem
- Exporteren van data naar een ERP systeem
- Exporteren van leverancier gegevens naar IRIS voor factuurherkenning

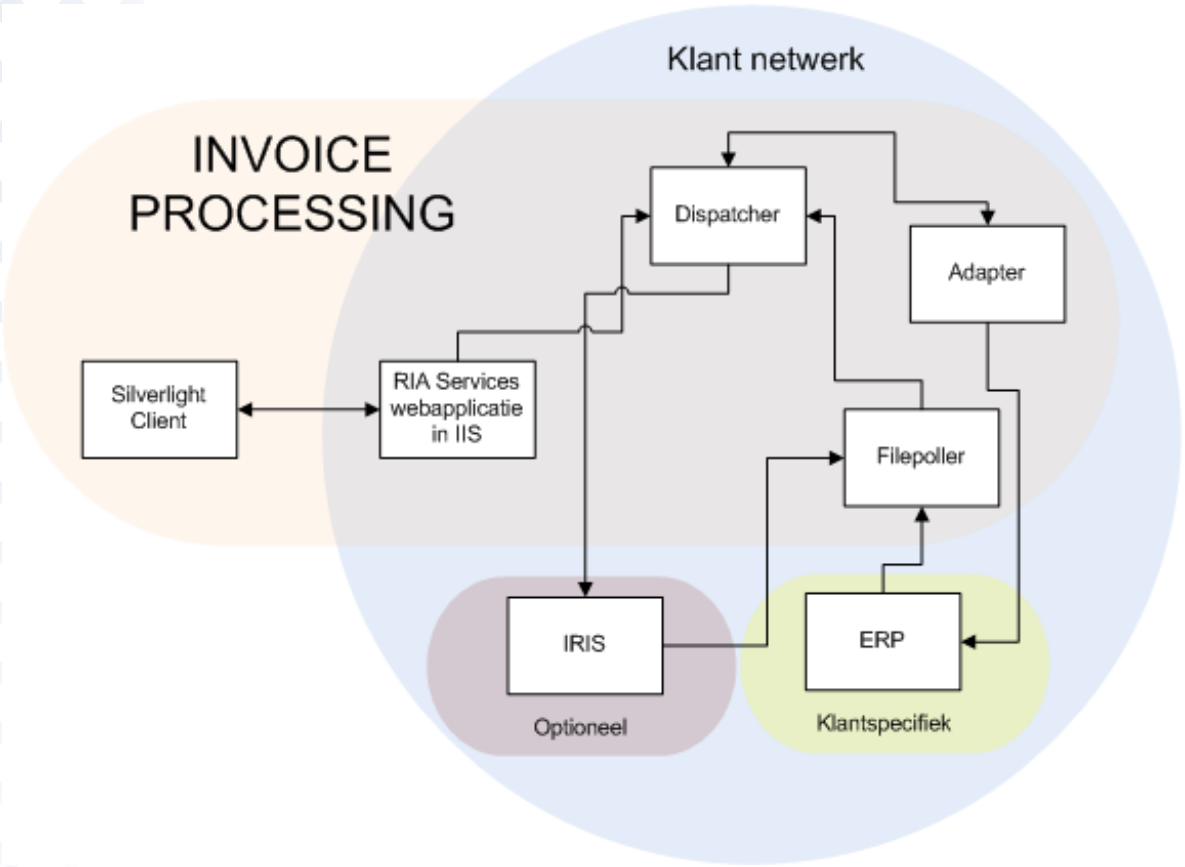
De FilePoller

De filepoller is een bestaand component in Invoice Processing. De filepoller is verantwoordelijk voor het detecteren van nieuwe import bestanden uit zowel IRIS als het ERP systeem.
In de filepoller is een wijziging in aangebracht waardoor deze nu ook via NServiceBus messages kan versturen.

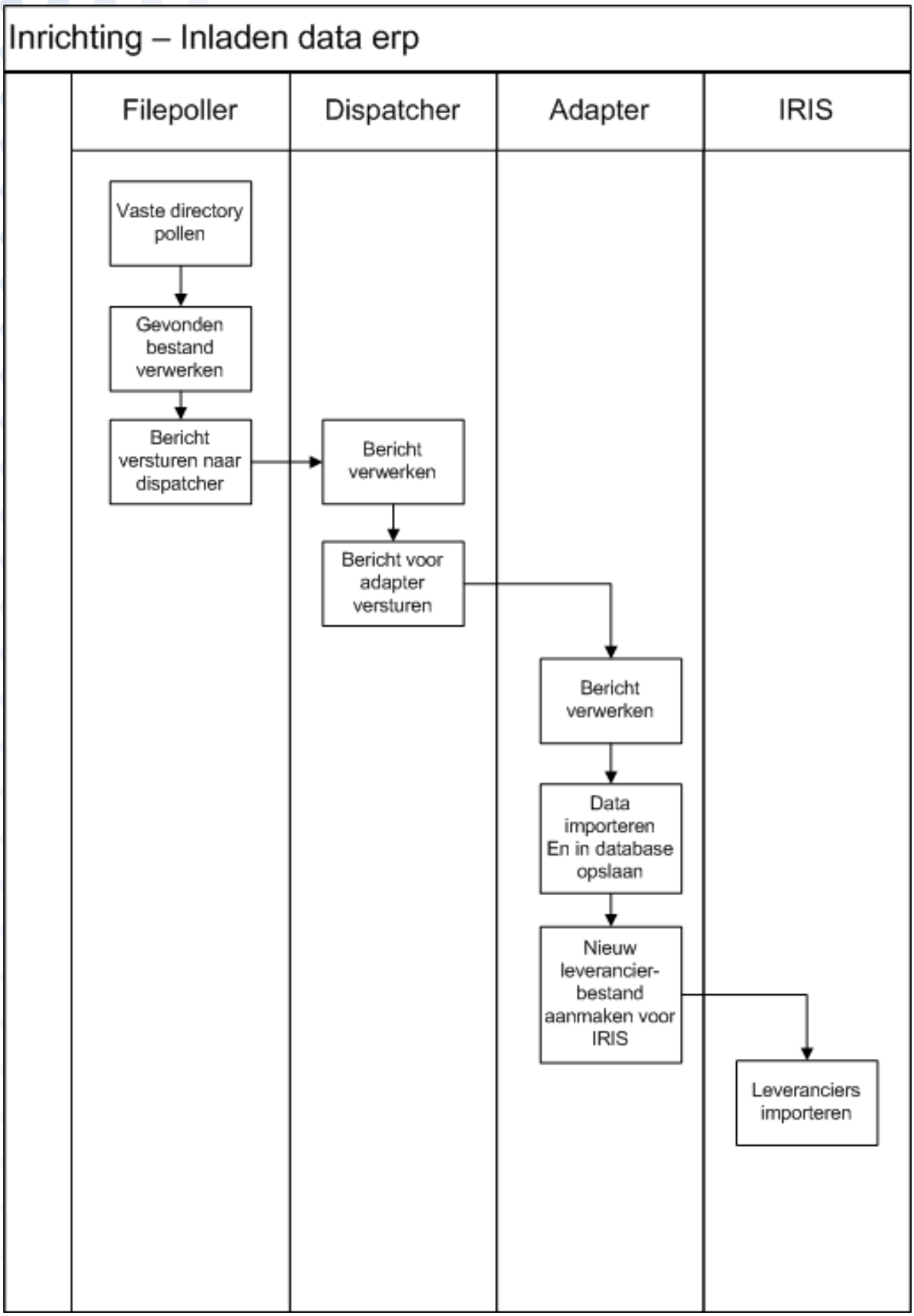
Met deze uitbreiding kan de filepoller nu ook leverancier bestanden uit een ERP herkennen. Zodra de filepoller een bestand herkent zal deze automatisch een bericht op de bus zetten naar de bijbehorende adapter. Deze adapter zal vervolgens de daadwerkelijke import doen.

7.2) De samenhang

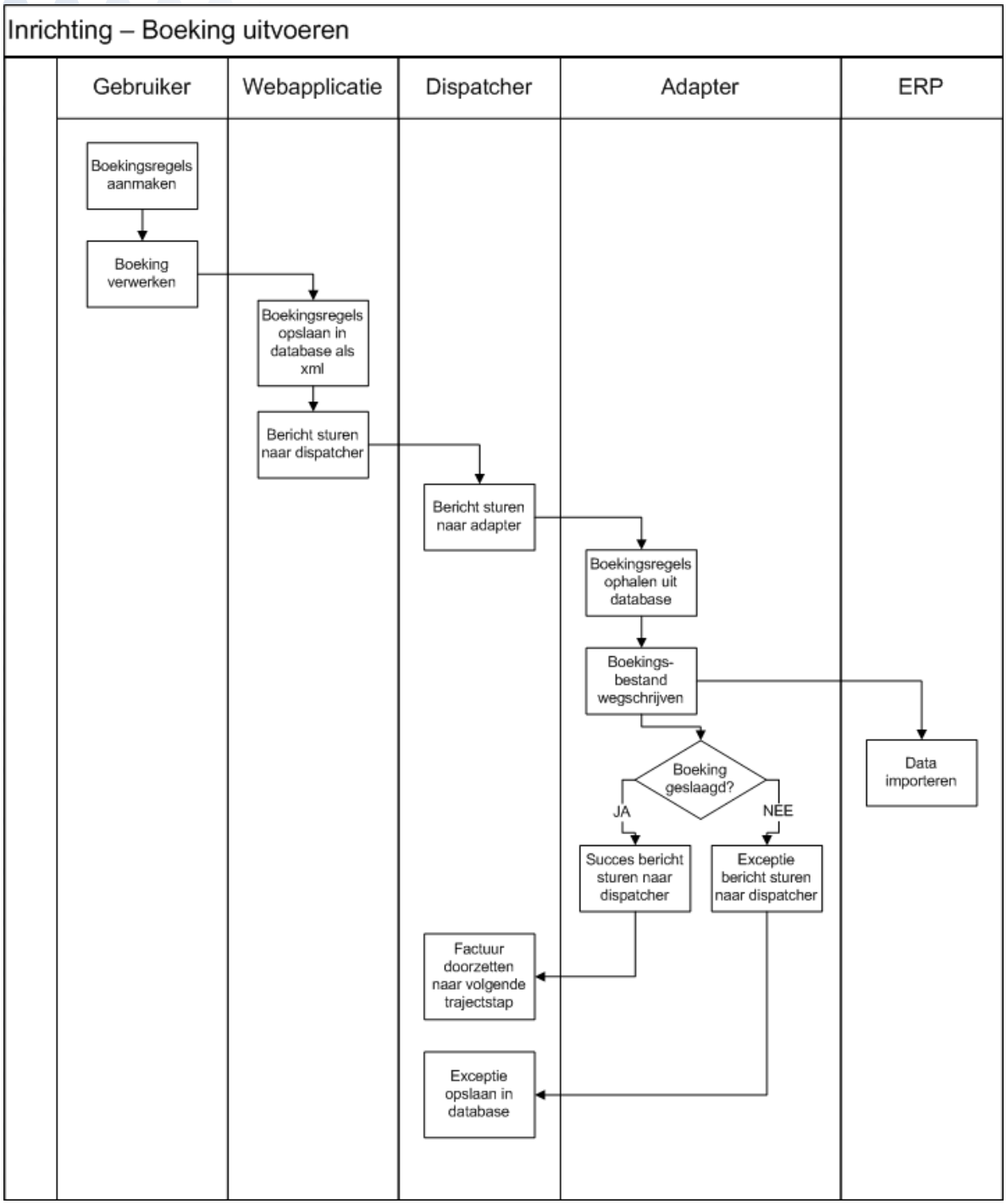
Het totaaloverzicht



Het inladen van data uit het ERP systeem



Het maken van een boeking



7.3) Visual Studio solution

In Visual Studio zijn meerdere solutions toegevoegd voor zowel de front als backend.
Deze solutions bevatten weer meerdere projecten die allemaal deel uitmaken van Invoice Processing als applicatie.

De frontend

Nokavision.AccountsPayable	De Silverlight applicatie
Nokavision.AccountsPayable.Beheer.UI	Een WCF applicatie voor installatie en beheer
Nokavision.AccountsPayable.Domain	Het domein model in Entity Framework
Nokavision.AccountsPayable.Web	De webapplicatie die Silverlight host. Hierin zit de RIA Services Backend

De backend

Nokavision.ExceptionHandling	Bevat alle classes voor errorhandling, wordt gedeeld over de adapters
Nokavision.FilePoller.Core	Bevat de core classes van de filepoller, hierin zit alle functionaliteit van de filepoller
Nokavision.FilePoller.Host	Bevat de NServiceBus host van de FilePoller, deze host maakt gebruik van de FilePoller.Core
Nokavision.P2P.Dispatcher	Bevat alle core classes van de dispatcher evenals de handlers van de dispatcher
Nokavision.P2P.Dispatcher.Adapter	Bevat alle handlers en core classes die gedeeld worden door alle adapters
Nokavision.P2P.Dispatcher.Adapter.Messages	Bevat alle messages die naar een adapter gestuurd kunnen worden
Nokavision.P2P.Dispatcher.Adapter.Tests	Bevat alle door de adapters gedeelde Unit tests
Nokavision.P2P.Dispatcher.Common	Bevat alle classes die door alle componenten gedeeld worden
Nokavision.P2P.Dispatcher.Host	Bevat de NServiceBus host voor de dispatcher, deze maakt gebruik van Nokavision.P2P.Dispatcher
Nokavision.P2P.Dispatcher.Messages	Bevat alle messages die naar de dispatcher gestuurd kunnen worden
Nokavision.P2P.Dispatcher.Tests	Bevat alle Unit tests voor de dispatcher
Nokavision.P2P.Domain	Bevat het domein model in Entity Framework, dit wordt gebruikt in alle componenten

Per adapter zijn de volgende projecten toegevoegd

Nokavision.P2P.Dispatcher.[erpnaam] Adapter	Bevat alle core classes en handlers voor de adapter
Nokavision.P2P.Dispatcher.[erpnaam] Adapter.Host	Bevat de NServiceBus host voor de adapter, deze gebruikt het [erpnaam]Adapter project
Nokavision.P2P.Dispatcher.[erpnaam] Adapter.Tests	Bevat alle unittests voor de adapter

Het kan voorkomen dat klanten hetzelfde ERP systeem gebruiken, maar deze toch anders hebben ingericht (kostendragers ipv. kostenplaatsen etc)
In zo'n geval kan dezelfde adapter worden geinstalleerd, de xslt en xsd bestanden worden per klant gewijzigd.

7.4) Invoice Processing database

Verwijderde tabellen

Uit de database van Invoice Processing zijn meerdere tabellen verwijderd, de tabellen voor grootboekrekeningen en kostenplaatsen zijn overbodig geworden omdat er niet meer met een vaste set boekingsgegevens gewerkt wordt.

Toegevoegde tabellen

Adapterboekingen
In deze tabel wordt de boeking in xml formaat per inkoopfactuur opgeslagen.
De code achter de schermen in Silverlight zet automatisch de aangemaakte regels om naar xml.

Adviesboekingen
In deze tabel wordt de adviesboeking in xml formaat per inkoopfactuur en per gebruiker opgeslagen.
De schermen in Silverlight zetten automatisch de aangemaakte regels om naar xml.

Adapterconfigurations
In deze tabel wordt per bedrijf opgeslagen naar welke bestanden moet worden gezocht om boekingdata te importeren. Per adapter staan hier regels in voor elke soort import.

Moeten er voor een boeking kostenplaatsen, grootboekrekeningen en kostendragers worden gebruikt dan zullen hier 3 regels in staan, 1 voor elk bestand dat geïmporteerd moet worden.

Een voorbeeld van de data:

Id	BedrijfId	AdapterItemType	BestandsNaam
73	2	jdedwardsActiva	activa.txt
74	2	jdedwardsGrootboekData	grootboekdata.txt

Adapterfilters
In deze tabel staat per gebruiker, per import bestand welke items gebruikt mogen worden bij de adviesboeking.
Er zal dus voor de jdedwardsActiva een regel staan voor gebruiker A. In deze regel wordt in xml opgeslagen welke geïmporteerde items gebruikt mogen worden.
Een voorbeeld van deze xml:

```
<Items>
 <Item>
  <Code>441</Code>
  <Omschrijving>Huisvesting</Omschrijving>
 </Item>
 <Item>
  <Code>44101</Code>
  <Omschrijving>Huishoudelijke dienst</Omschrijving>
 </Item>
</Items>
```

AdapterImports
In deze tabel staat de feitelijke data die wordt geïmporteerd door de adapter. Er staat hier per boeking item een apparte regel. Dus, bij het gebruik van kostenplaatsen, grootboekrekeningen en kostendragers staan er 3 regels. In het xml veld staat de werkelijk geïmporteerde data.
De xml is hier gelijk aan die van adviesboeking:

```
<Items>
 <Item>
  <Code>441</Code>
  <Omschrijving>Huisvesting</Omschrijving>
 </Item>
 <Item>
  <Code>44101</Code>
  <Omschrijving>Huishoudelijke dienst</Omschrijving>
 </Item>
</Items>
```

AdapterItems
Dit is de koppeltabel tussen de imports en de bedrijven.

PeriodeSets
Elke boeking wordt gedaan in een specifiek boekjaar en periode. Per bedrijf worden periodesets aangemaakt met daarin boekjaren met periodes. Deze tabel bevat de periodesets.

Boekjaren
De verschillende boekjaren staan in deze tabel.

Periodes
Elk boekjaar bestaat uit meerdere periodes, bijvoorbeeld elke maand. In deze tabel zijn deze terug te vinden

7.5) Schermen in de applicatie

In de Silverlight fontend zijn meerdere schermen toegevoegd voor het maken van boekingen en adviesboekingen. Silverlight maakt gebruik van het Model - View - ViewModel pattern (MVVM) een methode om de UI te scheiden van de business logica en het domein model.

Hierdoor zijn de volgende zaken aangemaakt:

BaseBoekingControlViewModel
Dit is de basis viewmodel met alle gedeelde functionaliteit van de boeking schermen, de adapterspecifieke viewmodels inheriten deze zodat er geen duplicate code ontstaat.

Per adapter zijn de volgende viewmodels toegevoegd:

[erpnaam]AdviesBoekingControlViewModel
Bevat alle adapterspecifieke functionaliteit voor het maken van een adviesboeking.

[erpnaam]BoekingControlViewModel
Bevat alle adapterspecifieke functionaliteit voor het maken van een boeking.

Hiernaast zijn per adapter de volgende controls aan de applicatie toegevoegd:

[erpnaam]AdviesBoekingControl
Bevat het visuele gedeelte van het adviesboeking scherm

[erpnaam]BoekingControl
Bevat het visuele gedeelte van het boeking scherm

Zodra de gebruiker in de applicatie bij het tabblad boeking of adviesboeking komt, zal door middel van een factory de juiste control worden aangemaakt en ingeladen. Alle schermen laden de geïmporteerde boekingdata, de eerder gemaakte boekingen en de gebruikersfilters in. Hiervan worden vervolgens collecties opgebouwd voor elk van de items (grootboekrekeningen, kostenplaatsen etc).

8) Wijzigingen en uitbreidingen in de oorspronkelijke opdracht

Tijdens het hele traject zijn een aantal wijzigingen gemaakt aan de oorspronkelijke opdracht. Een aantal vanuit klantwensen en een aantal vanuit technisch oogpunt. In dit hoofdstuk wordt hier een beknopt overzicht van gegeven.

RavenDb

Tijdens een van de updates van NServiceBus is door de ontwikkelaar besloten dat RavenDb, object-georienteerde database server, meegeleverd zou gaan worden. Tevens zou dit een vereiste zijn om zogeheten timeout berichten te sturen, berichten die na vaststaande tijd worden uitgevoerd.

Het gebruik van deze timeout berichten wordt in Invoice Processing gebruikt voor het bijhouden van escalaties van facturen. Als deze na een x aantal dagen niet zijn verwerkt dan wordt een escalatie gestart. Hierdoor wordt een gebruiker op de hoogte gesteld en krijgt hij de taak deze factuur te verwerken.

Om het aantal services van IP te beperken is ervoor gekozen om RavenDb eruit te halen en te vervangen door SQL server. SQL server wordt al gebruikt als database server voor IP.
Om dit te doen zijn uit de source van NServiceBus de classes gehaald die verantwoordelijk zijn voor dit timeout proces.
Deze classes zijn aangepast zodat deze met SQl server werken en aan IP toegevoegd.
In de configuratie van NServiceBus wordt vervolgens aangegeven dat deze classes gebruikt worden en niet de standaard classes.

ErrorManager

In NServiceBus is standaard functionaliteit aanwezig waarmee fouten in de message handlers worden afgehandeld. Als een message handler een exceptie krijgt, wordt er automatisch een bericht op de error queue geplaatst. Met een bijgeleverde tool kan dit bericht teruggezet worden op de originele queue. Op dat moment wordt dit bericht weer opgepakt door de handler en zal deze opnieuw worden uitgevoerd.

Omdat deze tool alleen los beschikbaar is hebben we ervoor gekozen om de ErrorManager classes uit de source te halen en aan te passen. Bij een error wordt nu nog steeds een error bericht gemaakt. De details van de error en dit error bericht worden nu opgeslagen in de database van IP in de vorm van een errorlog. In de Silverlight frondend is een scherm gemaakt dat deze errorlog toont aan de beheerder. Deze beheerder kan deze error messages nu vanuit Silverlight opnieuw laten uitvoeren.

Filters adviesboekingen

Vanuit de klant is de wens gekomen om bij het maken van een advies boeking de items (grootboekrekeningen etc.) te filteren per gebruiker, zodat verschillende gebruikers op verschillende items kunnen boeken.
Om dit te doen is een extra scherm toegevoegd aan de frontend. In dit scherm kan een beheerder per gebruiker instellen welke items deze tot zijn beschikking heeft. Hiervoor moet wel een import gebeurd zijn van de ERP boeking data.

Verplichtingen boeken

Voor een bepaalde klant is het noodzakelijk dat voordat een boeking wordt geïmporteerd deze al bekend is in het ERP systeem. Hiervoor is de zogeheten “verplichting boeking” functionaliteit ontwikkeld. Hierbij wordt een zeer beperkte export gedaan richting het ERP direct na het inladen van de facturen uit IRIS. In deze export staan beperkte gegevens van de factuur zoals het factuurnummer, bedrag en de leverancier.

9) Conclusie en aanbevelingen

Tijdens het traject is gebleken dat het gebruik van een servicebus meer mogelijheden biedt dan alleen het asynchroon exporteren van een boeking. Voor implementatie van een servicebus is NServiceBus een goede oplossing gebleken, toch hierbij nog een kritische noot:

Bij het upgraden van NServiceBus naar een nieuwere versie is meerdere malen gebleken dat backwards compatibility niet aanwezig is. Hierdoor is bij elke upgrade de applicatie onbruikbaar geworden en is er veel tijd gestoken in het aanpassen van de services.
Tevens is de documentatie wat betreft de wijzigingen te beknopt geweest, waardoor deze problemen niet voorzien konden worden en het aanpassen soms te veel "trial and error" was.
Om die reden beveel ik aan om het update van NServiceBus niet te onderschatten. Dit kan een tijdrovend en daarmee kostbare aangelegenheid zijn.

Naast deze waarschuwing wil ik aangeven dat het in de toekomst zeker rendement kan opleveren om NServiceBus voor meer onderdelen van de applicatie in te gaan zetten.
Hiermee kan een betere scheiding gemaakt worden tussen business logica en de user interface. Dit maakt de ontwikkeling van een nieuwe UI in MVC3 en HTML5 (of welke techniek dan ook) vele malen eenvoudiger en sneller. De backend kan nu los van de UI draaien.

Het gebruik van een servicebus heeft daarnaast ook voordelen in een multi-user omgeving omdat dit voorkomt dat meerdere processen data wegschrijven naar een database. Dit voorkomt deadlocks waardoor gebruikers data niet kunnen opslaan.

Evaluatie

Het hele traject van onderzoek, testen, bouwen en implementeren van de adapters is vrij soepel verlopen. Er is vrijwel nergens grote vertraging opgelopen en het resultaat sluit goed aan bij de initiële wensen en eisen vanuit Nokavision Software.

Inmiddels zijn meerdere adapters opgeleverd en in gebruik genomen door klanten en hier zijn nog geen grote issues uitgekomen.
Adapter 3 is momenteel in ontwikkeling en de verwachte ontwikkeltijd is enkele dagen. Dit is een grote vooruitgang ten opzicht van de ontwikkeling van de 1e adapter, dit duurde enkele weken.

Op persoonlijk vlak kan ik het traject ook een succes noemen. Tijdens het traject heb ik veel nieuwe inzichten opgedaan over het ontwikkelen van business applicaties. Alles bij elkaar is het een leerzame en interessante opdracht geweest met meer (technische) diepgang dan voorheen verwacht.
Mijn initiële beeld van een plugin model is langzaam getransformeerd in een beeld hoe de volledige applicatie ingericht kan worden voor meer efficiëntie, gebruiksvriendelijkheid, stabiliteit en flexibiliteit.

Literatuurlijst

Voor het uitvoeren van de opdracht zijn naast praktijkervaringen van medewerkers van Nokavision alleen bronnen van internet geraadpleegd. Hieronder volgt een overzicht van deze bronnen.

Algemeen:

- http://nl.wikipedia.org/wiki/Language_Integrated_Query
- http://nl.wikipedia.org/wiki/Ontwerppatroon
- http://nl.wikipedia.org/wiki/Enterprise_resource_planning
- http://nl.wikipedia.org/wiki/Extensible_Markup_Language
- http://nl.wikipedia.org/wiki/Dependency_injection
- http://en.wikipedia.org/wiki/Inversion_of_control
- http://nl.wikipedia.org/wiki/Enterprise_service_bus
- http://en.wikipedia.org/wiki/JD_Edwards

NServiceBus:
http://www.nservicebus.com/

Mass transit:
http://masstransit-project.com/

RhinoBus:
http://hibernatingrhinos.com/

Unity framework:
http://msdn.Microsoft.com/en-us/library/ff648512.aspx

Spring Framework:
http://www.springsource.org/

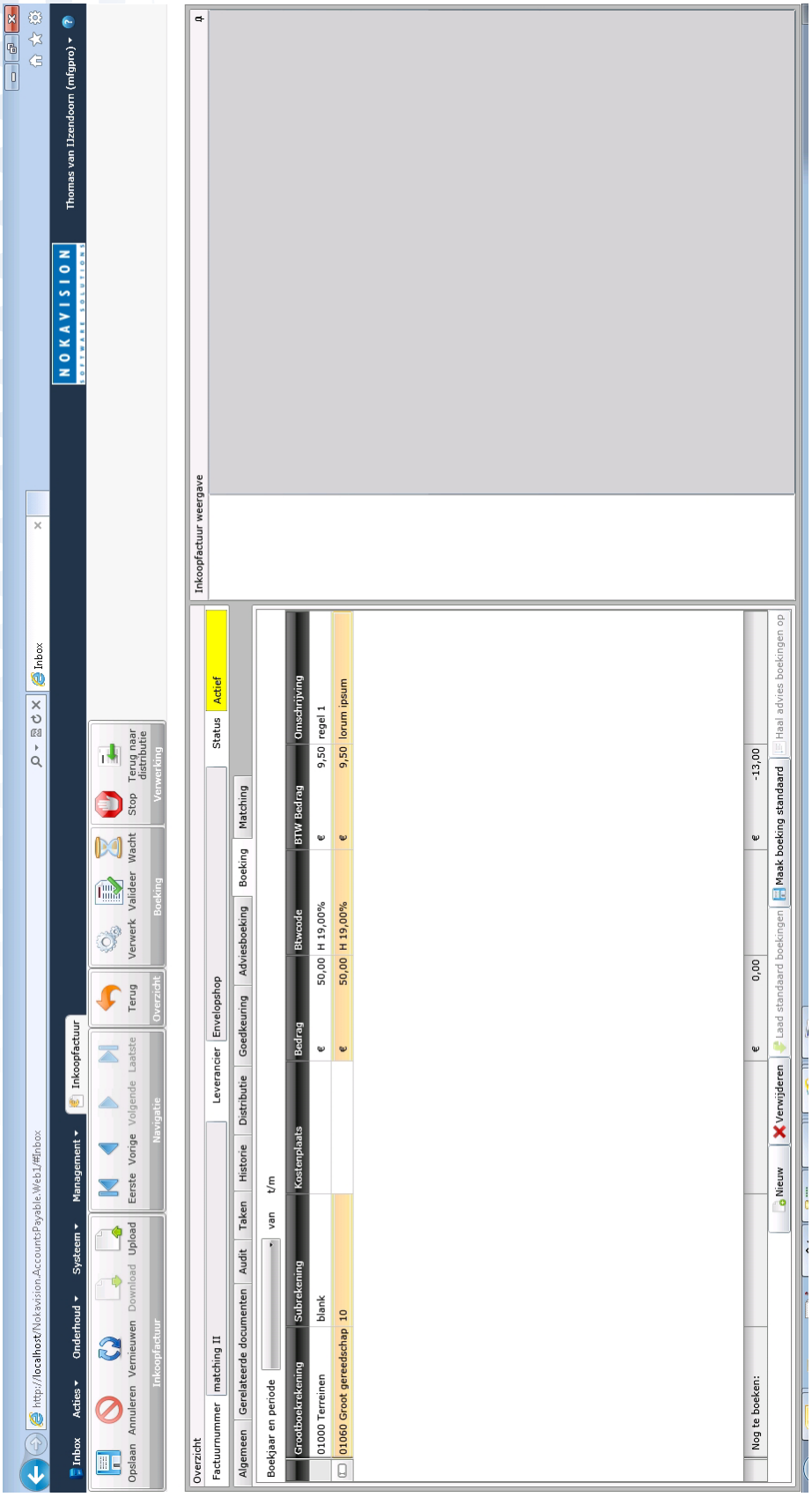
Castle windsor:
http://castleproject.org/

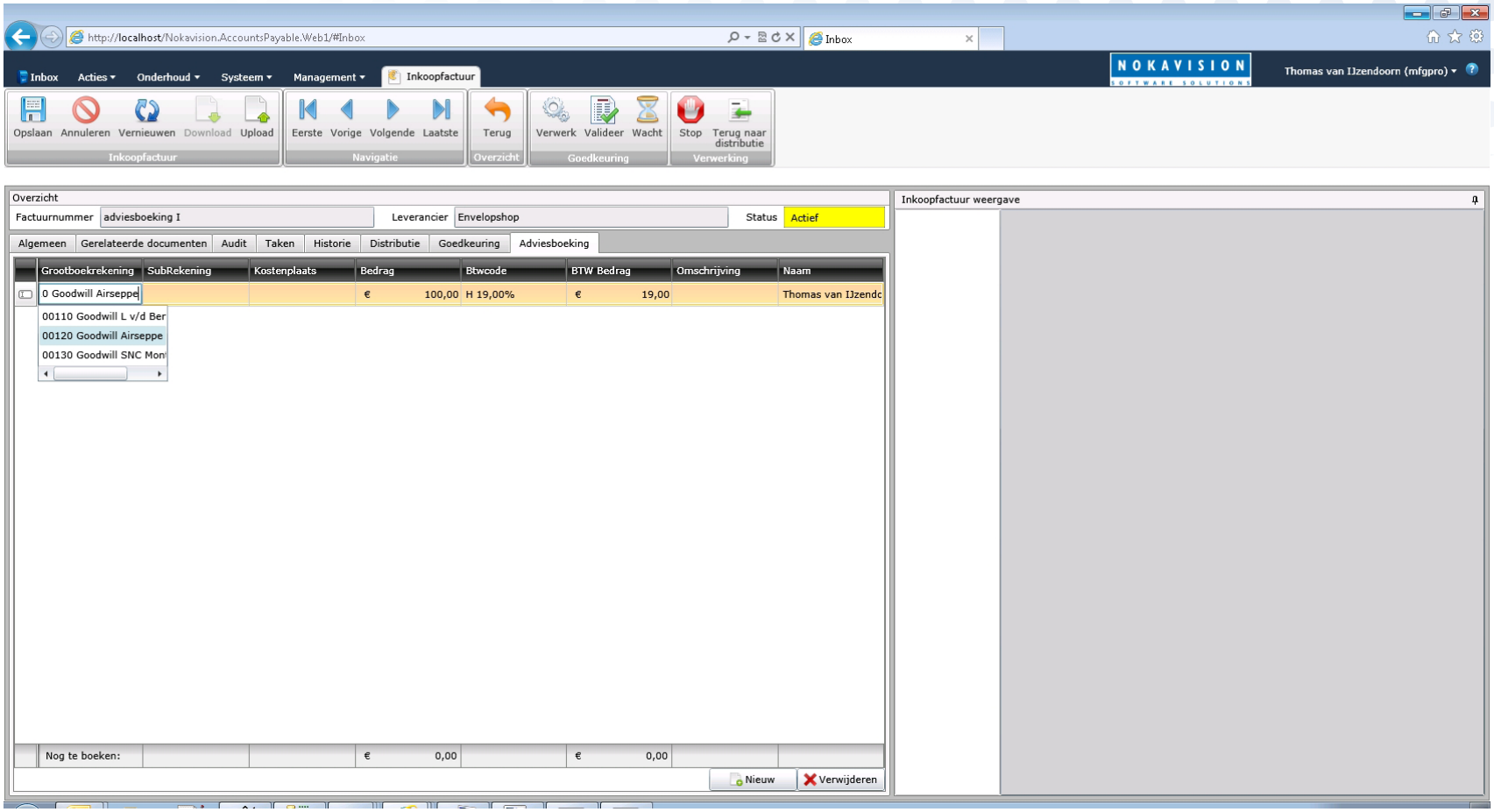
Autofac:
http://code.google.com/p/autofac/

Ninject:
http://www.ninject.org/

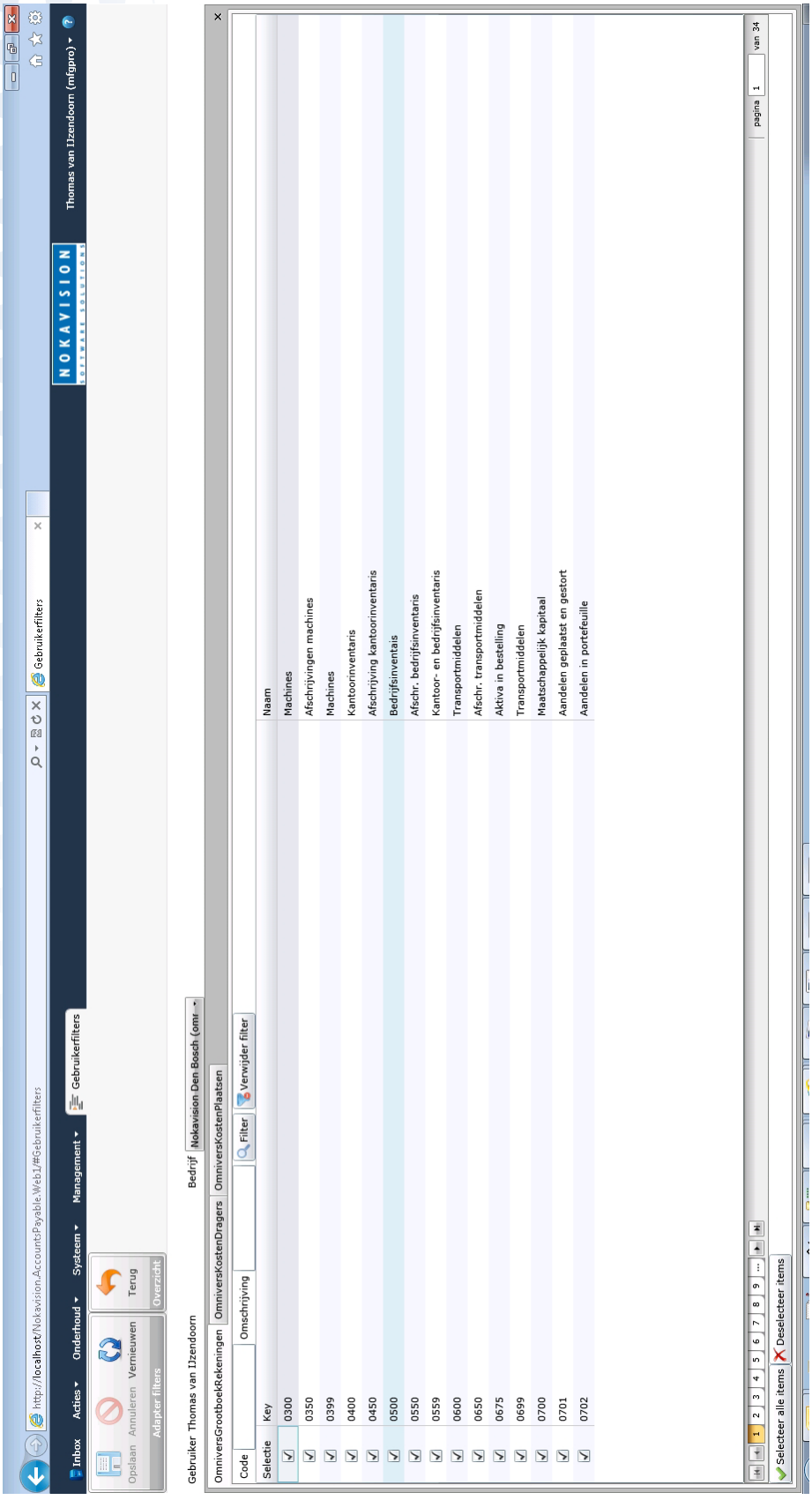
Bijlagen

Bijlage I) Afbeelding Boeking

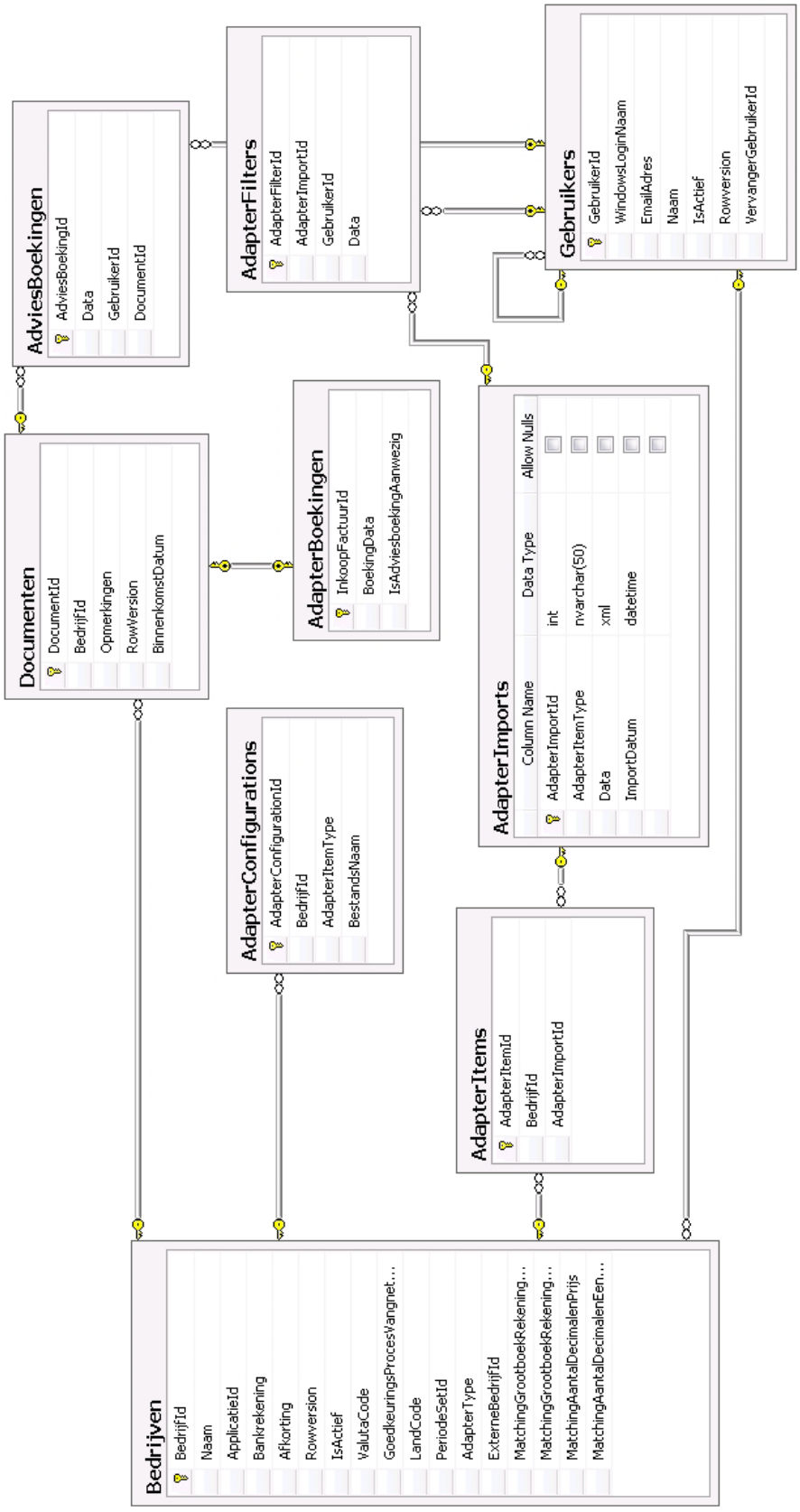



http://localhost/Nokavision.AccountsPayable.Web1/#Inbox
Inbox
Inbox
Acties
Onderhoud
Systeem
Management
Inkoopfactuur
NOKAVISION
Thomas van IJzendoorn (mfgpro)
Opslaan
Annuleren
Vernieuwen
Download
Upload
Inkoopfactuur
Eerste
Vorige
Volgende
Laatste
Navigatie
Terug
Overzicht
Verwerk
Valideer
Wacht
Goedkeuring
Stop
Terug naar distributie
Verwerking
Overzicht
Factuurnummer
adviesboeking I
Leverancier
Envelopshop
Status
Actief
Algemeen
Gerelateerde documenten
Audit
Taken
Historie
Distributie
Goedkeuring
Adviesboeking
Grootboekrekening
SubRekening
Kostenplaats
Bedrag
Btwcode
BTW Bedrag
Omschrijving
Naam
0 Goodwill Airseppe
€ 100,00
H 19,00%
€ 19,00
Thomas van IJzendo
00110 Goodwill L v/d Ber
00120 Goodwill Airseppe
00130 Goodwill SNC Mon
Nog te boeken:
€ 0,00
€ 0,00
Nieuw
Verwijderen
Inkoopfactuur weergave

Bijlage III) Afbeelding Instellen gebruikersfilters voor adviesboeking



Bijlage IV) Afbeelding Datamodel boekingen



Bijlage V) Afbeelding Datamodel periodesets voor gebruik bij een boeking