Bijlagen

Data Repository

Inhoudsopgave

1 Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2016-2.1 (start uiterlijk 29 augustus 2016)
Startdatum uitvoering afstudeeropdracht:
Inleverdatum afstudeerdossier volgens jaarrooster: 22 december 2016

Studentnummer: 09094202
Achternaam: dhr van Leeuwen
Voorletters: G.A.J.
Roepnaam: Glenn
Adres: Oudelandselaan 255
Postcode: 2652 ER
Woonplaats: Berkel en Rodenrijs
Telefoonnummer: 0613219954
Mobiel nummer: 0613219954
Privé emailadres: gvanleeuwen13@gmail.com

Opleiding: Informatica
Locatie: Zoetermeer
Variant: voltijd

Naam studieloopbaanbegeleider: Rob E. Loke
Naam begeleidend examinator: Ton Biegstraaten
Naam tweede examinator: Rob E. Loke

Naam bedrijf: Ask Roger!
Afdeling bedrijf: Development
Bezoekadres bedrijf: Mijnbouwstraat 106

Postcode bezoekadres: 2628 RX
Postbusnummer:
Postcode postbusnummer:
Plaats: Delft
Telefoon bedrijf: 088 – 27 57 600
Telefax bedrijf:
Internetsite bedrijf: http://www.askroger.nl/

Achternaam opdrachtgever: Mevrouw Horsten
Voorletters opdrachtgever: C. (Chianne)
Titulatuur opdrachtgever: -
Functie opdrachtgever: HR Manager
Doorkiesnummer opdrachtgever: 06-40640448/088 2757632
Email opdrachtgever: chorsten@askroger.nl

Achternaam bedrijfsmentor: dhr Heemskerk
Voorletters bedrijfsmentor: F. (Frank)
Titulatuur bedrijfsmentor: -
Functie bedrijfsmentor: Teamlead development
Doorkiesnummer bedrijfsmentor: 088 2757621
Email bedrijfsmentor: fheemskerk@askroger.nl

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Ontwikkelen van een database service voor het koppelen van meerdere databronnen bij Ask Roger!.

Opdrachtomschrijving

1. Bedrijf

- Ask Roger! is een bedrijf in de branche IT, Telecom en communicatie. Zij bouwen communicatiesystemen voor MKB'ers en het grotere segment. (kracht op > 250- 1000 gebruikers)
- De focus ligt op robuuste systemen voor de lange termijn.
-
- Ask Roger! is een groeiend bedrijf en zij verwachten deze groei voort te zetten de komende maanden. In september zijn er naar verwachting 40 medewerkers in dienst.
- Het bedrijfsonderdeel waar de opdracht wordt uitgevoerd is de afdeling Development. Deze afdeling bestaat uit 3.6 fte.

2. Probleemstelling

ToastAR is een applicatie die bij binnenkomende oproepen in Skype for Business relevante klantgegevens toont in een pop-up venster. Deze klantgegevens komen uit CRM en ERP systemen die aan het systeem gekoppeld zijn.
HoastAR is de server applicatie, ToastAR is de client applicatie. ToastAR heeft HoastAR nodig om data op te halen.

Via ToastAR kan ook gezocht worden in de contactinformatie afkomstig van een gekoppeld CRM of ERP systeem. Hiermee kan er naar personen worden gezocht en kan er uit de zoekresultaten een persoon worden geselecteerd, die dan gebeld kan worden via Skype for Business.
Verder kan via de ToastAR een nummer of postcode aangeklikt worden in de applicatie, waarna met een hotkey (F8) die persoon gebeld wordt.

HoastAR is een oplossing die in eerste instantie gebruikt is in combinatie met Skype for Business. Deze is ondertussen uitgebreid voor TAPI (Telephony Application Programming Interface) toepassingen zoals Cisco Call Manager en Avaya Communication Manager.

Wanneer er een inkomende conversation (telefoongesprek, chatbericht etc.) is, dan zoekt HoastAR aan de hand van de Caller ID, postcode of andere kenmerken verdere informatie op uit het gekoppelde CRM of ERP systeem. Deze informatie wordt daarna geleverd aan ToastAR en getoond op het scherm.

Er zijn veel verschillende CRM en ERP systemen met klantinformatie, die allemaal op verschillende manieren benaderbaar zijn. Om de HoastAR niet te belasten met alle verschillende koppelingen zit hier een stuk software tussen, de data repository.

De data repository fungeert als interface die klantgegevens levert aan de HoastAR.
Bij de data repository kunnen replicators ingesteld worden. Een replicator haalt data op via een koppeling en slaat deze gegevens lokaal op. De replicators kunnen ingesteld worden op bijvoorbeeld een ODBC koppeling, CSV, Excel of een custom replicator.
Hiermee heeft de data repository lokaal de data uit de verschillende databronnen zoals CRM en ERP systemen. Deze data wordt eerst omgevormd naar het gewenste format van

klantgegevens voordat het wordt opgeslagen, zodat de HoastAR direct de gegevens kan gebruiken zonder hier last te hebben.
Er kan ingesteld worden hoe vaak een replicator gegevens repliceert, het is gebruikelijk dat de replicator dit dagelijks doet. Omdat dit dus geen realtime data betreft wordt in de data repository vooral statische data opgeslagen zoals adresgegevens. Deze data is dus vrijwel direct te zien in de ToastAR. Realtime data wordt niet via de data repository opgehaald en dit gaat bijvoorbeeld via webservices. Denk bij realtime data aan data zoals openstaande facturen van klanten, waarbij het van belang is dat deze informatie actueel is.

De data repository is op dit moment software van een extern bedrijf en de wens is om deze software zelf te maken en de externe software te vervangen. Dit wordt gedaan omdat Ask Roger! zo zelf uitbreidingen kan maken voor deze software en daarnaast niet meer afhankelijk is van een extern bedrijf.

Het plan is om als uitbreiding modules voor ieder CRM systeem te maken die klanten van Ask Roger! gebruiken, zodat via een wizard een CRM systeem van een klant gekoppeld kan worden aan de data repository zonder dit handmatig te doen. Bij het ontwerpen van de te ontwikkelen data repository moet hier dus al rekening mee worden gehouden.

Voor meer uitleg over ToastAR, zie: http://www.askroger.nl/nl-NL/wat/toastar

3. Doelstelling van de afstudeeropdracht

Het doel van de opdracht is het ontwikkelen en het opleveren van een werkende data repository service en eventuele module(s) hiervoor ontwikkelen als daar nog tijd voor is.

4. Resultaat

Het resultaat is een interne service die het systeem geheel in eigen handen brengt van Ask Roger! en het bedrijf niet meer afhankelijk maakt van een externe service.
Hiermee kan er een pakket aangeboden worden aan klanten met geheel eigen software. Als de modules voor de verschillende CRM/ERP systemen ingebouwd zijn, kan er ook gedacht worden aan trial versies voor het softwarepakket.
Daarnaast kan er ook gedacht worden aan de markt in het buitenland, omdat dan niet meer handmatig de koppeling gemaakt hoeft te worden bij de klant, dit wordt dan geregeld door de modules.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Het plan is om een aangepaste versie van de agile ontwikkelmethode scrum te gebruiken. Hierbij zullen er sprints gedaan worden van twee weken, met de eerste sprint van drie weken. Iedere week zal er 1 dag besteed worden aan het afstudeerverslag.

Sprint 0 (3 weken): Plan van Aanpak
- Globale Requirements vaststellen
- Systeem ontwerpen

Sprint 1 (2 weken): Basis van de data repository ontwerpen
- Basis van de data repository bouwen
- Basis van de data repository testen

Sprint 2 (2 weken): Lokale data opslag van de repository ontwerpen
- Lokale data opslag van de repository bouwen
- Lokale data opslag van de repository testen

Sprint 3 (2 weken): Replicator voor CSV databronnen ontwerpen
- Replicator voor CSV databronnen bouwen

Replicator voor CSV databronnen testen
Sprint 4 (2 weken): Replicator voor Excel databronnen ontwerpen
Replicator voor Excel databronnen bouwen
Replicator voor Excel databronnen testen
Sprint 5 (2 weken): Replicator voor ODBC databronnen ontwerpen
Replicator voor ODBC databronnen bouwen
Replicator voor ODBC databronnen testen
Sprint 6 (4 weken): Custom replicator voor handmatige configuratie ontwerpen
Custom replicator voor handmatige configuratie bouwen
Custom replicator voor handmatige configuratie testen
Sprint 7(overige tijd): ontwerpen van een module voor de data repository
Bouwen van de ontworpen module
Testen van de module

In iedere sprint zal een systeemdeel ontwikkeld worden zodat de voortgang zichtbaar blijft. Het ontwikkelen van een systeemdeel bestaat uit ontwerpen, bouwen en het testen hiervan.

Als er tijd is voor de 7e sprint dan zal daarin een module gemaakt worden om één van de CRM of ERP systemen met deze module automatisch te kunnen koppelen aan de data repository.

6. Op te leveren (tussen)producten

(Tussen)producten die opgeleverd worden zijn het Plan van Aanpak, Requirements, UML ontwerp in de vorm van een Klassediagram, RRM lokale data opslag, Testplan, Testdocumentatie en de code.

7. Te demonstreren competenties en wijze waarop

De volgende competenties zullen worden gedemonstreerd:

Analyseren en Adviseren
1.4 Uitvoeren analyse door definitie van requirements
In sprint 0 zullen er requirements opgesteld worden voor het systeem, zodat hier rekening mee gehouden wordt bij het ontwerpen van het systeem.

Softwareontwerp en -ontwikkeling

3.2 Ontwerpen systeemdeel
Vanaf sprint 1 zal er in iedere sprint een systeemdeel ontworpen worden. In sprint 0 wordt er een globaal systeemontwerp gemaakt.

3.3 Bouwen applicatie
Vanaf sprint 1 zal er in iedere sprint een systeemdeel gebouwd worden.

3.5 Uitvoeren van en rapporteren over het testproces
Vanaf sprint 1 zal er in iedere sprint een systeemdeel getest worden.

2 Plan van Aanpak

Inhoudsopgave

2.1 Opdracht omschrijving

2.1.1 Organisatie

Ask Roger! is een bedrijf in de branche IT, Telecom en communicatie. Zij bouwen communicatiesystemen voor bedrijven. Dit zijn middelgrote en grotere bedrijven. Het aantal gebruikers per klant ligt tussen de 250 en 1000 gebruikers.

Voor het online samenwerken met collega's en klanten biedt Ask Roger! oplossingen als video conferencing en unified communications. Met video conferencing kan er vergaderd worden op afstand met behulp van videogesprekken. Bij unified communications draait het om tijd, plaats en apparaat onafhankelijk communiceren. Dit is niet één vast product maar een combinatie van meerdere oplossingen.

Ter verbetering van het klantcontact bij bedrijven biedt Ask Roger! oplossingen als CRM/ERP integratie en rapportage systemen. Bij CRM/ERP integratie worden klantgegevens uit deze systemen samengevoegd binnen één applicatie die is gekoppeld aan een IP telefooncentrale of aan Microsoft Skype for Business. Hiermee hebben bedrijven direct klantinformatie van de klant die hen belt of een chat start. Voor rapportage ontwikkelt Ask Roger! verschillende eigen oplossingen die bovenop basiselementen kunnen draaien, zoals een callcenter wallboard bovenop Cisco Unified Contact Center.

Voor een optimale bereikbaarheid van bedrijven biedt Ask Roger! oplossingen als omnichannel contact centers, vast-mobiel integratie en enterprise networks. Omnichannel contact centers zijn contact centers die niet alleen via een telefoon bereikbaar zijn, maar bijvoorbeeld ook via een browser, chat of video contact met een medewerker. Met vast-mobiel integratie maakt het niet uit of er een vaste of mobiele telefoon wordt gebruikt, gesprekken van klanten kunnen worden aangenomen onafhankelijk hiervan. Ask Roger! biedt het beheren en monitoren van netwerk infrastructuren aan onder het mom van enterprise networks.
Naast het aanleggen van bedrijfsnetwerken biedt Ask Roger! ook internet en telefonie netwerken aan bedrijven.

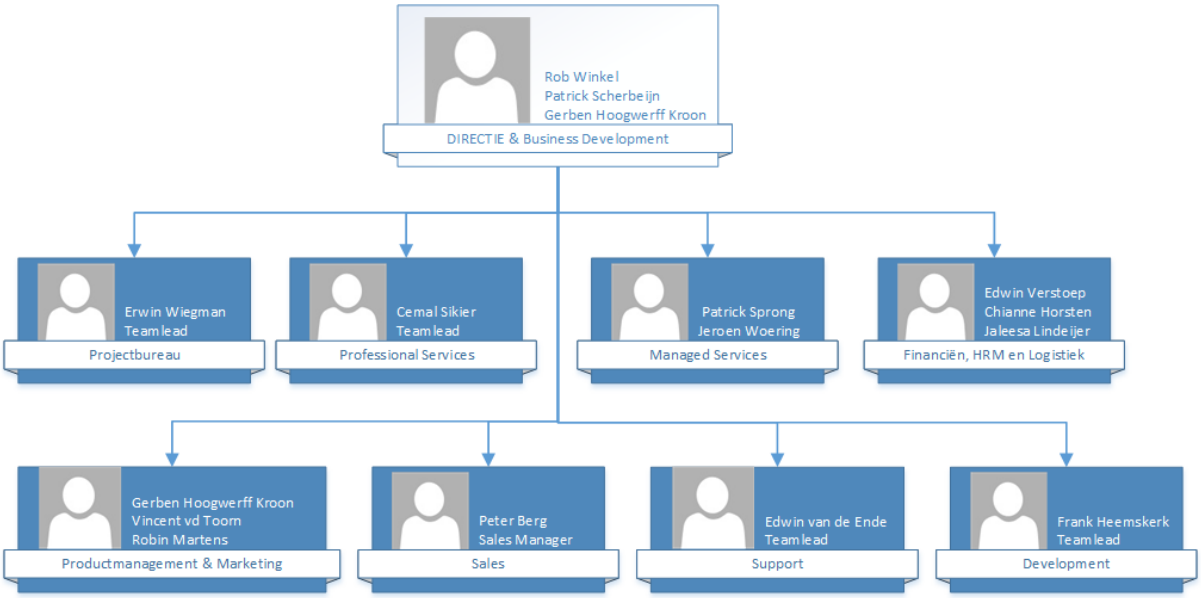
Verder heeft Ask Roger! consultants die advies op maat geven aan bedrijven. Ook leveren zij demo's en training sessies. Ten slotte heeft Ask Roger! ook een support afdeling waar support wordt geboden voor alle producten die bij het bedrijf worden afgenomen.

De focus ligt op robuuste systemen voor de lange termijn, het bedrijf wil een lange termijnrelatie aangaan met haar klanten. Het bedrijf is oplossingsgericht en zal niet snel aangeven dat iets onmogelijk is.

Ask Roger! heeft als klanten bijvoorbeeld de bedrijven en instellingen Corendon, Staatsbosbeheer, BPD, Nova College en Davo Wittebrug. Hiermee heeft Ask Roger! klanten in verscheidene branches.[1]

Ask Roger! is een groeiend bedrijf en zij verwachten deze groei voort te zetten de komende maanden. In september waren er ongeveer 40 medewerkers in dienst. Dit is

ondertussen uitgegroeid tot ongeveer 50 medewerkers. Het bedrijf heeft op dit moment twee locaties: Houten en Delft, waarbij Delft de hoofdlocatie is. Er bevinden zich meerdere afdelingen binnen de organisatie.



Figuur 1. Organogram.

Een uitgebreide versie van het organogram is te vinden in bijlage 5.
Het bedrijf heeft als toplaag de afdeling Directie & Business Development. Deze afdeling bestaat uit drie personen.
In figuur 1 zijn de acht andere afdelingen te zien met bijbehorende team leiders of managers.

De opdracht wordt uitgevoerd op de afdeling Development, deze afdeling bevindt zich alleen op de hoofdlocatie Delft. Er zijn ook andere afdelingen aanwezig op deze locatie. De afdeling Development bestaat uit 3 personen, waarbij ik de 4^{e} persoon op de afdeling ben. De werkdruk van de afdeling zonder mij bestaat uit 3.6 FTE. FTE is een eenheid die staat voor Full-time Equivalent, in het Nederlands wordt deze eenheid ook wel werktijdfactor genoemd. Een FTE van 1 staat gelijk aan de werkdruk van een fulltime werknemer.

2.1.2 Probleembeschrijving

ToastAR is een applicatie die bij binnenkomende oproepen in Skype For Business of een IP telefooncentrale relevante klantgegevens toont in een pop-up venster. Deze klantgegevens komen uit CRM en/of ERP systemen van de bedrijven zelf die de ToastAR gebruiken, deze systemen zijn lokaal aan het ToastAR systeem gekoppeld. HoastAR is de server applicatie, ToastAR is de client applicatie. ToastAR heeft HoastAR nodig om data op te halen. Zie figuur 2 voor een duidelijk beeld van de samenhang van deze applicaties.

Via ToastAR kan ook gezocht worden in de contactinformatie afkomstig van een gekoppeld CRM of ERP systeem. Hiermee kan er naar personen worden gezocht en kan er uit de zoekresultaten een persoon worden geselecteerd, die dan gebeld kan

worden.
Verder kan via de ToastAR een nummer of postcode aangeklikt worden in de applicatie, waarna met een hotkey (F8) die persoon gebeld wordt.

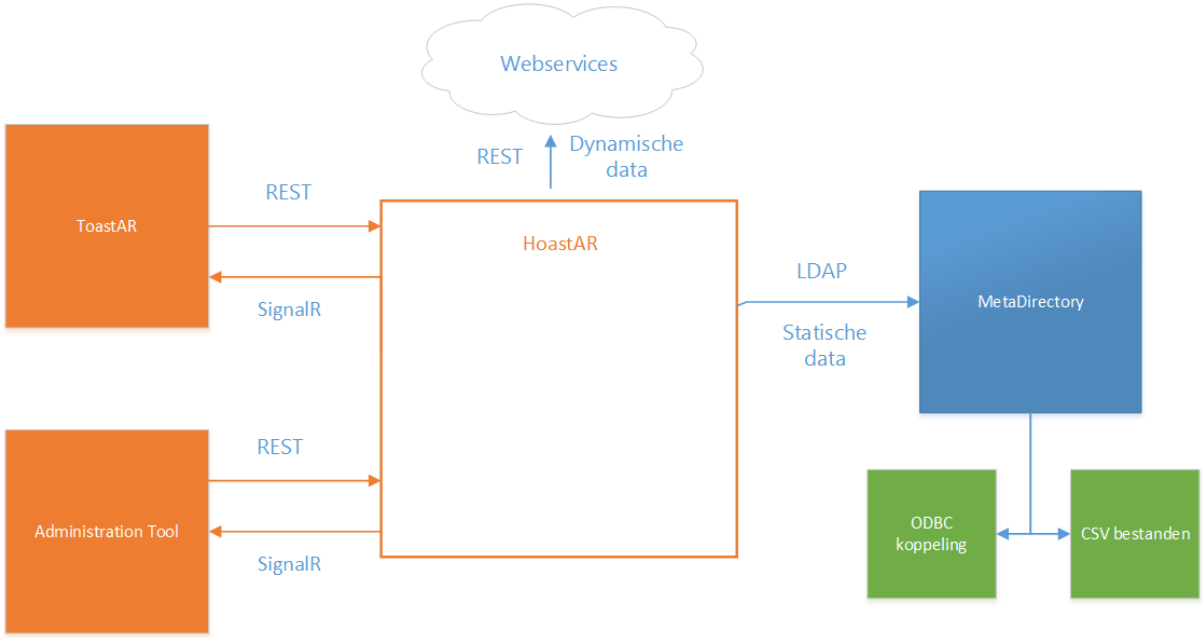
ToastAR is een oplossing die in eerste instantie ontwikkeld is voor gebruik in combinatie met Skype For Business. Dit is ondertussen uitgebreid voor IP telefonie via een TAPI koppeling. Telephony Application Programming Interface (TAPI) is een Microsoft Windows API dat telefonie mogelijk maakt via het Windows besturingssysteem. Toepassingen zoals Cisco Call Manager en Avaya Communication Manager kunnen via deze interface aangestuurd worden.[2]

Wanneer er een inkomend telefoongesprek of inkomend chatbericht is, dan vraagt ToastAR informatie over deze beller of chatter op aan HoastAR. HoastAR haalt aan de hand van de Caller ID, postcode of andere kenmerken verdere informatie op uit het gekoppelde CRM of ERP systeem. Deze informatie wordt daarna geleverd aan ToastAR en getoond op het scherm in de vorm van een pop-up. In de pop-up is ook de mogelijkheid de telefoon op te nemen of de chat te openen.
Als de telefoon wordt opgenomen of de chat is geopend, dan wordt er een groter scherm weergegeven met relevante informatie over de beller of chatter.

Informatie aanvragen van ToastAR aan de HoastAR gaan via een Representational state transfer (REST) architectuur. Ook wel RESTful web services genoemd. Deze architectuur is een manier om interoperabiliteit te leveren tussen computersystemen over het internet.[3] Op deze manier kan ToastAR aanvragen doen aan HoastAR zonder fysiek in dezelfde omgeving te hoeven draaien.
Om data terug te sturen van HoastAR naar ToastAR wordt er gebruik gemaakt van ASP.NET SignalR. Dit is een ASP.NET library die het mogelijk maakt om data van de server kant te sturen naar een client als deze data beschikbaar komt.[4] Dit is ideaal voor dit systeem omdat de door ToastAR opgevraagde data niet altijd direct beschikbaar is.
HoastAR is ook bereikbaar voor een administratie tool van Ask Roger, deze maakt op dezelfde manier als ToastAR een connectie met HoastAR.

Statische data zoals adresgegevens en telefoonnummers worden door de HoastAR anders opgehaald dan dynamische data. Denk bij dynamische data aan data zoals openstaande facturen van klanten, waarbij het van belang is dat deze informatie actueel is.
Statische data wordt gerepliceerd uit de gekoppelde CRM en/of ERP systemen en lokaal opgeslagen. Deze data kan vrijwel direct geleverd worden aan ToastAR en is bij binnenkomende telefoongesprekken en chatberichten direct te zien in de pop-up.
Het repliceren van de data gebeurt dagelijks. Deze snelle manier van informatie leveren aan ToastAR is dus niet geschikt voor dynamische data, omdat dit type data niet verouderd mag zijn.
Dynamische data wordt via RESTful web services opgevraagd en dit vergt meer tijd, deze informatie is dus later in ToastAR op het scherm te zien dan de statische gegevens.

De koppeling tussen de CRM en ERP systemen met klantinformatie en HoastAR verloopt in de huidige situatie via een directory service genaamd MetaDirectory van het bedrijf Estos.
Een directory service is een dienst die het mogelijk maakt om toegang te krijgen tot hiërarchisch georganiseerde gegevens die eventueel verspreid zijn opgeslagen in een computernetwerk.[5,6]
MetaDirectory kan gegevens uit bepaalde databronnen repliceren en lokaal opslaan. Deze bronnen zijn o.a. Comma Separated Value (CSV) bestanden en databases via een Open Database Connectivity (ODBC) koppeling. ODBC is een interface voor het benaderen van database management systemen.[7] Gegevens uit CRM en ERP systemen worden in de huidige situatie meestal geëxporteerd naar CSV bestanden alvorens deze bestanden worden gerepliceerd en opgeslagen door de MetaDirectory. De lokaal opgeslagen data wordt opgevraagd door HoastAR. Er kan ingesteld worden met welk interval de data wordt gerepliceerd door de MetaDirectory, in de huidige situatie is dit meestal een interval van een dag. HoastAR en MetaDirectory draaien beide als Windows Service, de communicatie tussen de HoastAR en de directory service verloopt via het Lightweight Directory Access Protocol (LDAP). LDAP is een protocol voor het benaderen van gedistribueerde directory services over een IP netwerk.[8]



Figuur 2. Systeem Context Diagram huidige situatie.

In figuur 2 is een overzicht te zien waarin de samenwerking van de systemen wordt weergegeven in een Systeem Context Diagram. De CRM en ERP systemen zijn gekoppeld aan de MetaDirectory via een ODBC koppeling of via geëxporteerde CSV bestanden zoals eerder beschreven. Het onderscheid in het type data is ook in het diagram weergegeven, de statische data wordt door de HoastAR opgehaald van de MetaDirectory via LDAP en de dynamische data via RESTful web services.

ToastAR, HoastAR en de administratie tool zijn geschreven in C# en maken gebruik van het .NET Framework.

In de HoastAR wordt gebruik gemaakt van de design pattern Dependency injection. Deze pattern implementeert het Inversion of Control principle, waarbij delen van een programma aangestuurd worden door een generiek framework. De modules van HoastAR hebben in hun constructor staan welke service zij nodig hebben en deze wordt geleverd bij het opstarten van deze modules. Een goed voorbeeld is de logger module die bijna in alle constructors aanwezig is om foutmeldingen te kunnen loggen. Om deze pattern toe te passen wordt er gebruik gemaakt van een open source dependency injector genaamd Ninject. Dit is een library voor .NET applicaties. Ninject is het framework dat gebruikt wordt om de modules binnen HoastAR aan te sturen.[9,10]

De wens van Ask Roger! is het vervangen van de MetaDirectory met een eigen softwareoplossing. Dit wordt gedaan omdat Ask Roger! zo zelf uitbreidingen kan maken voor deze software en daarnaast niet meer afhankelijk is van een extern bedrijf.
De vervanging van de MetaDirectory wordt een module van HoastAR, zo wordt er maar één Windows service gebruikt. De MetaDirectory draait in de huidige situatie in een eigen Windows service.

Er is een plan om in de toekomst modules te ontwikkelen voor deze vervangende software die zorgen voor een directe koppeling met CRM en ERP systemen die worden gebruikt door klanten van Ask Roger!. Bij het ontwerpen van de te ontwikkelen data repository service moet hier dus al rekening mee worden gehouden.

2.1.3 Doelstelling

Het doel van de opdracht is het ontwikkelen en het opleveren van een werkende data repository service die data uit externe bronnen lokaal opslaat en eventuele module(s) hiervoor ontwikkelen die data ophalen uit specifieke CRM of ERP systemen als hier nog tijd voor is.

2.1.4 Resultaat

Het resultaat is een interne service die het systeem geheel in eigen handen brengt van Ask Roger! en het bedrijf niet meer afhankelijk maakt van een externe service.

Hiermee kan er inde toekomst een pakket aangeboden worden aan klanten met geheel eigen software. Als de modules voor de verschillende CRM/ERP systemen ingebouwd zijn, kan er ook gedacht worden aan trial versies voor het softwarepakket. Daarnaast kan er ook gedacht worden aan de markt in het buitenland, omdat dan niet meer handmatig de koppeling gemaakt hoeft te worden bij de klant, dit wordt dan automatisch geregeld met behulp van de modules.

2.2 Afbakening

De opdracht beperkt zich tot de Data Repository, een module die data uit verschillende soorten databronnen lokaal opslaat en de data via de HoastAR aanbiedt

aan de ToastAR applicatie. Er moet data vanuit de HoastAR opgevraagd kunnen worden waarmee het repliceren van de data uit een bepaalde bron ingepland kan worden.

2.3 Projectmethodiek

Tijdens het maken van het afstudeerplan heb ik gekeken naar verschillende ontwikkelmethodieken om te bepalen welke het best past bij de stageopdracht. Ten eerste is bepaald dat een iteratieve ontwikkelmethode de voorkeur heeft, sinds niet alle requirements vast staan bij aanvang van het project en er later nog requirements bij kunnen komen. Door een iteratieve methodiek te gebruiken kunnen requirements die later toegevoegd worden nog mee worden genomen in het huidige project en geïmplementeerd worden.
Verder is het bij deze opdracht gewenst incrementeel te ontwikkelen, om zo het softwareproduct in delen op te kunnen leveren. Als er een verkeerde inschatting is gemaakt tijdens de planning of als er zich onvoorziene vertragingen voordoen in het project kan op deze manier zeker zijn van werkende systeemdelen in plaats van een softwareproduct dat onvolledig is en misschien niet functioneert.

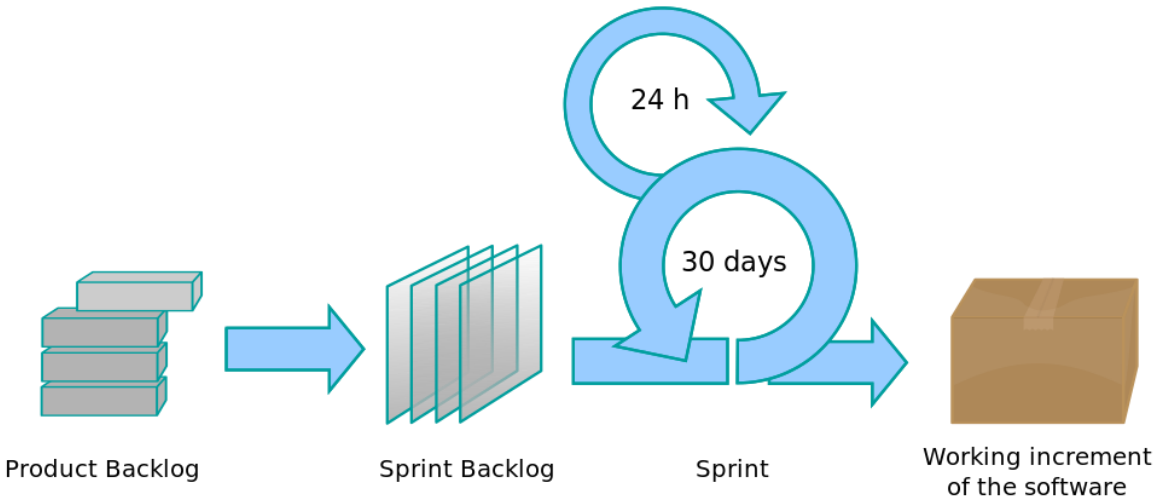
Bij het zoeken naar een incrementele en iteratieve software ontwikkelmethodiek vielen bepaalde methodieken zoals (Rational) Unified Process ((R)UP) af, sinds deze wel iteratief is maar niet incrementeel.
Er is uiteindelijk besloten om een Agile ontwikkelmethodiek te volgen, deze methodieken volgen het Manifest voor Agile Software Ontwikkeling[11], deze luidt als volgt:

- Mensen en hun onderlinge interactie boven processen en hulpmiddelen.
- Werkende software boven allesomvattende documentatie.
- Samenwerking met de klant boven contractonderhandelingen.
- Inspelen op verandering boven het volgen van een plan.

Hierbij wordt aangegeven dat dingen aan de rechterkant, zoals het volgen van een plan, niet onbelangrijk zijn.
De meeste Agile ontwikkelmethodieken focussen op incrementele oplevering van producten in korte iteraties.[12] Dit sluit daarom dus goed aan bij deze opdracht.

Het oog is hierbij gevallen op Scrum, een iteratieve en incrementele software ontwikkelmethodiek. Andere Agile methodieken vielen af omdat deze niet aansloten bij de opdracht. Bijvoorbeeld de ontwikkelmethodiek eXtreme Programming (XP) zou gebruik maken van Pair Programming, waarbij twee programmeurs samen achter één computer werken aan hetzelfde stuk code. Dit is onmogelijk sinds deze afstudeeropdracht individueel is.
Agile methodes die wel voldeden aan de eisen maar toch niet gekozen zijn, zoals DSDM, zijn afgevallen door persoonlijke voorkeur. Er is door mij al eerder ervaring opgedaan met Scrum en de werkwijze bevalt me.

Bij Scrum wordt het volgende proces gevolgd.[13]



Figuur 1. Het Scrum proces.

Een Scrum project bestaat uit meerdere iteraties waarin een werkend deel van de software wordt geleverd, wat Scrum dus iteratief en incrementeel maakt.
Iedere iteratie wordt een Sprint genoemd. Een sprint heeft een lengte van één tot vier weken.

De lijst met requirements van het project wordt de Product Backlog genoemd. Iedere sprint heeft een lijst met taken die tot een werkend systeemdeel moeten leiden, dit wordt de Sprint Backlog genoemd. Het is gebruikelijk aan het begin van de dag een meeting te houden met het team dat aan de sprint werkt, om zo de voortgang en de planning voor die dag door te nemen. Dit wordt ook wel de dagelijkse (Daily) Scrum of stand-up (meeting) genoemd.
Aan het eind van een sprint worden twee dingen gedaan, een Sprint Review en een Sprint Retrospective. In de Sprint Review wordt er bekeken welke geplande onderdelen af zijn gekregen en welke niet. Het is ook gebruikelijk een demo te geven aan de opdrachtgever van het werk dat af is deze sprint.
In de Sprint Retrospective wordt gereflecteerd op de uitgevoerde sprint, er worden twee vragen beantwoord:
Wat ging er goed deze Sprint?
Wat kan er verbeterd worden in de volgende Sprint?
Dit wordt gebruikt om een continue verbeterproces in stand te houden.

Binnen dit project zal er een aangepaste versie van Scrum worden gebruikt, de opdracht wordt immers door één developer uitgevoerd, namelijk ik als afstudeerder. In de planning is de sprint verdeling te zien voor dit project, verder wordt als Product Backlog het Requirements document gebruikt. Per Sprint zal er een Sprint Backlog gemaakt worden, met aan het eind van de Sprint een Review en Retrospective. Er is afgesproken met de bedrijfsmentor om wekelijks de voortgang te bespreken, dit moment zal ook gebruikt worden als sprint review als er een sprint is afgerond.
Er wordt binnen de Development afdeling van Ask Roger! al gebruik gemaakt van

stand-up meetings om het hele team van elkaar op de hoogte te houden waar aan gewerkt wordt en wat hierin de voortgang is. Ik doe hier ook aan mee om anderen op de hoogte te houden en mee te draaien in het Development team. Dit is ook handig om meer inzicht te krijgen in het HoastAR en ToastAR systeem van Ask Roger!.

Naast deze ontwikkelmethodiek wordt er gebruik gemaakt van Object-Oriented Design (OOD). Dit gebruik ik omdat dit helpt een complex software probleem op te delen in kleinere onderdelen die ik per onderdeel gestructureerd kan oplossen, hiermee behoud ik makkelijker het overzicht. OOD focust zich op het concept van objecten en het toepassen hiervan in het ontwerpen van een applicatie. Het voordeel van het gebruik van objecten is dingen definiëren in het ontwerp die verantwoordelijk zijn voor zichzelf. Een object is iets met eigen verantwoordelijkheden. Een object weet wat voor type het is, de data in een object zorgt dat het object weet in welke toestand het zich bevindt en de code in het object zorgt dat het juist functioneert en de juiste functionaliteiten heeft.
Er kan ook gekeken worden naar objecten vanuit de verschillende perspectieven van Fowler[14]:

- Op een conceptueel niveau is een object een verzameling van verantwoordelijkheden.
- Op een specificatie niveau is een object een verzameling van methodes ofwel gedragingen die aangeroepen kunnen worden door andere objecten of het object zelf.
- Op een implementatie niveau is een object code en data en de computationele interacties tussen de twee.

Bij object georiënteerd ontwerpen komen klassen voort uit conceptuele objecten. Een klasse is een definitie van het gedrag van een object. Het bevat een beschrijving voor de data elementen die een object bevat, de methodes die een object bevat en de manier waarop de data elementen en methodes benaderd kunnen worden.

Omdat de data elementen van een object kunnen variëren kan ieder object van hetzelfde type andere data bevatten, maar zullen dezelfde functionaliteiten bevatten die gedefinieerd zijn door de methodes. Objecten zijn instanties van klassen.

Abstractie, encapsulatie en polymorfisme zijn drie belangrijke principes binnen OOD. Abstractie is een techniek om de complexiteit van computer systemen te arrangeren. Alleen relevante informatie wordt gebruikt in een bepaalde context, onderliggende details zijn te vinden op een lager abstractie niveau.
Encapsulatie is het verbergen van informatie binnen een object of module voor andere objecten en modules zodat deze informatie niet benaderd kan worden. Alleen informatie die door middel van een interface beschikbaar is gemaakt kan benaderd worden. Een interface beschermt de rest van een programma voor implementatie details die snel kunnen veranderen. Daarnaast is het een beveiliging van informatie, er wordt in een interface gekozen wat voor informatie andere objecten of modules mogen zien of gebruiken.
Polymorfisme is de mogelijkheid om iets conceptueels aan te roepen door middel van

een abstracte referentie, maar verschillend gedrag terug te krijgen afhankelijk van de context.

Bij het ontwerpen van de applicatie en het herstructureren van de gemaakte code wordt er gebruik gemaakt van Design Patterns (DP) ofwel ontwerp patronen en Principles, principes binnen software ontwikkeling.
DP zijn herbruikbare ontwerp patronen die vaak voorkomende softwareproblemen binnen een bepaalde context oplossen. DP richten zich op een generale aanpak. Het zijn blauwdrukken om een ontwerp op te baseren, het zijn geen strikte regels waaraan gehouden moet worden.[15]

Principles zijn vuistregels die gevolgd kunnen worden om bepaalde problemen binnen het programmeren te voorkomen. Sommige DP zijn een implementatie van een bepaalde Principle. Een goed voorbeeld hiervan is al beschreven in de beschrijving van de huidige situatie, de Dependency Injection DP implementeert het Inversion of Control Principle.

Ten eerste zal ik in dit project de SOLID Principles volgen. Dit is een afkorting voor de vijf belangrijkste Principles binnen object-georiënteerd programmeren. Dit zijn:

- Single responsibility Principle. Dit houdt in dat een klasse één verantwoordelijkheid moet hebben. Dit wordt ook wel het High Cohesion Principle genoemd, waarin de elementen binnen een klasse een hoge functionele verwantheid hebben aan elkaar ofwel een hoge cohesie. Dit moet de robuustheid van klassen bevorderen. De functionaliteiten van een klasse met meerdere verantwoordelijkheden kunnen makkelijker kapot gaan bij aanpassingen dan bij een klasse met één verantwoordelijkheid.
- Open Closed Principle. Dit houdt in dat klassen, modules en functies open moeten staan voor uitbreiding, maar gesloten voor verandering. Als een verandering aan een programma resulteert in ongewenste veranderingen in gerelateerde modules is dit slecht ontworpen, het programma is dan fragiel, onbuigzaam en onvoorspelbaar. Dit principe geeft aan dat modules ontworpen moeten worden met het doel dat deze nooit veranderen. Als er nieuwe functionaliteiten van een module bij komen, moet er nieuwe code worden geschreven en mag de oude code niet worden aangepast. Om dit toe te passen spelen abstractie en polymorfisme een belangrijke rol.
- Liskov Substitution Principle. Dit houdt in dat klassen die afgeleid zijn van een basisklasse uitwisselbaar en vervangbaar moeten zijn voor hun basisklasse zonder veranderingen te veroorzaken aan het gedrag van het systeem. Het principe garandeert semantische interoperabiliteit van types in een hiërarchie.
- Interface Segregation Principle. Dit houdt in dat interfaces client specifiek moeten zijn. Clients moeten niet geforceerd worden om interfaces te implementeren die zij niet gebruiken. Dit principe voorkomt grote interfaces voor meerdere clients, waardoor die clients onnodige functies moeten implementeren die niet relevant zijn voor hun functioneren.
- Dependency Inversion Principle. Dit houdt in dat men afhankelijk moet zijn van abstracties en niet van implementaties. Details moeten afhankelijk zijn van

abstracties, abstracties mogen niet afhankelijk zijn van details. Zo blijven modules op een hoger niveau onafhankelijk van implementatie details, dit bevordert Low Coupling.[16]

Als deze vijf principes gevolgd worden moeten deze samen zorgen voor een systeem dat makkelijk is te onderhouden en makkelijk is uit te breiden met nieuwe functionaliteiten in de toekomst.
Verder worden ook de volgende Principles toegepast:

- Don't Repeat Yourself (DRY). Dit houdt in dat ieder stukje informatie binnen een systeem maar één keer mag voorkomen. Het alternatief is gedupliceerde code dat op meerdere plekken voorkomt in de applicatie. Als deze code veranderd moet worden, moet de gedupliceerde code opgezocht worden en op meerdere plekken veranderd worden. Als het DRY principe gevolgd wordt hoeft code maar op één locatie aangepast te worden en blijft de code onderhoudbaar.
- Low Coupling ofwel Loose Coupling. Dit houdt in dat softwaremodules niet te veel om moeten gaan met andere modules, de modules moeten zo los ofwel onafhankelijk mogelijk van elkaar zijn. Coupling is de mate van afhankelijkheid tussen modules. Als de afhankelijkheid laag is dan is de kans klein dat andere modules aangepast moeten worden die afhankelijk zijn van een module als die module wordt aangepast, dit bevordert de onderhoudbaarheid van de code. Dependency Inversion is een specifieke vorm van decoupling, het streven naar een lage afhankelijkheid tussen modules.
- Abstractions Live Longer Than Details, ook wel Generalization Principle genoemd. Dit houdt in dat een algemene oplossing dat niet één maar meerdere problemen oplost beter is dan een specifieke oplossing. Specifieke oplossingen zijn meestal fragiel, bij veranderende requirements voldoen deze meestal niet meer. Een algemenere oplossing is stabieler en kan vaak hergebruikt worden voor meerdere situaties, dit voorkomt duplicatie en sluit goed aan bij het DRY principe.
- Crash Early, ook wel Fail Fast genoemd. Dit houdt in dat fouten in het systeem zo vroeg mogelijk opgespoord en gerapporteerd moeten worden om verborgen problemen met een applicatie te voorkomen. Verborgen problemen kunnen tot ongewenst gedrag leiden van een systeem. Dit kan voorkomen worden door bijvoorbeeld validiteit van parameters te checken en Exceptions op te vangen en te loggen.[17]

Als deze principes toegepast worden dan bevorderen zij de onderhoudbaarheid van de code nog verder.

Voor het ontwikkelen van de applicatie volg ik de Framework Design Guidelines voor het .NET Framework. Dit doe ik om een intern consistent programma te leveren die de best practices hanteert voor het .NET Framework. De Data Repository wordt een module van HoastAR, dit is geschreven in C# en maakt gebruik van het .NET Framework.[18,19]

2.4 Projectorganisatie

Functie	Naam	Telefoon	E-mail
Afstudeerder	Glenn van Leeuwen	06-13219954 088-2757617	09094202@student.hhs.nl gvanleeuwen13@gmail.com gvanleeuwen@askroger.nl
1e Examinator	Ton Biegstraaten	06-46244461	ton.biegstraaten@gmail.com a.w.w.m.biegstraaten@hhs.nl
2e Examinator	Rob Loke	06-86805937	r.e.loke@hhs.nl
Opdrachtgever / HR Manager	Chianne Horsten	06-40640448 088-2757632	chorsten@askroger.nl
Bedrijfsmentor / Teamlead Development	Frank Heemskerk	088-2757610	fheemskerk@askroger.nl

2.5 Planning

Het plan is om een aangepaste versie van de Agile ontwikkelmethode Scrum te gebruiken. Hierbij zullen er sprints gedaan worden van twee weken, met de eerste sprint van drie weken.

Iedere week zal er 1 dag besteedt worden aan het afstudeerverslag.

Sprint 0 (3 weken):

- Plan van Aanpak
- Globale Requirements vaststellen
- Systeem ontwerpen

Sprint 1 (2 weken):

- Basis van de data repository ontwerpen
- Basis van de data repository bouwen
- Basis van de data repository testen

Sprint 2 (2 weken):

- Lokale data opslag van de repository ontwerpen
- Lokale data opslag van de repository bouwen
- Lokale data opslag van de repository testen

Sprint 3 (2 weken):

- Replicator voor CSV databronnen ontwerpen
- Replicator voor CSV databronnen bouwen
- Replicator voor CSV databronnen testen

Sprint 4 (2 weken):

- Replicator voor Excel databronnen ontwerpen
- Replicator voor Excel databronnen bouwen
- Replicator voor Excel databronnen testen

Sprint 5 (2 weken):

- Replicator voor ODBC databronnen ontwerpen
- Replicator voor ODBC databronnen bouwen
- Replicator voor ODBC databronnen testen

Sprint 6 (4 weken):

- Custom replicator voor handmatige configuratie ontwerpen
- Custom replicator voor handmatige configuratie bouwen
- Custom replicator voor handmatige configuratie testen

Sprint 7(overige tijd):

- Replicator voor een specifiek CRM of ERP systeem ontwerpen
- Replicator voor een specifiek CRM of ERP systeem bouwen

Replicator voor een specifiek CRM of ERP systeem testen

In iedere sprint zal een systeemdeel ontwikkeld worden zodat de voortgang zichtbaar blijft. Het ontwikkelen van een systeemdeel bestaat uit ontwerpen, bouwen en het testen hiervan.

Als er tijd is voor de 7e sprint dan zal daarin een module gemaakt worden om één van de CRM of ERP systemen met deze module automatisch te kunnen koppelen aan de data repository.

2.6 Mijlpaalproducten

Plan van Aanpak – Dit document.

Requirements – Document met de requirements voor de te ontwikkelen software.

Klassediagram – Ontwerp van de software in de vorm van een UML klassediagram.

Relational Representation Model (RRM) lokale data opslag – Een model om de datastructuur van de lokale data opslag weer te geven.

Testplan – Document waarin de geplande tests beschreven staan.

Testdocumentatie – Document met de beschrijving van de uitgevoerde tests.

Code – Software prototypes van de te ontwikkelen software en de code van het eindproduct.

Afstudeerverslag – Het op te leveren verslag aan de hogeschool om te bewijzen aan de relevante competenties te voldoen.

2.7 Literatuurlijst

1. Ask Roger! – Specialist in Microsoft Skype for Business en Cisco Communicatie oplossingen. *Ask Roger!* http://www.askroger.nl/. Bekeken op 10-12-2016.
2. Telephony Application Programming Interface. *Wikipedia*. https://en.wikipedia.org/wiki/Telephony_Application_Programming_Interface. Voor het laatst aangepast op 1-12-2016. Bekeken op 12-12-2016.

3. Representational state transfer. *Wikipedia*. https://en.wikipedia.org/wiki/Representational_state_transfer. Voor het laatst aangepast op 12-12-2016. Bekeken op 12-12-2016.
4. SignalR. *ASP.NET*. https://www.asp.net/signalr. Bekeken op 12-12-2016.
5. Directoryservice. *Wikipedia*. https://nl.wikipedia.org/wiki/Directoryservice. Voor het laatst aangepast op 11-12-2013. Bekeken op 12-12-2016.
6. Directory service. *Wikipedia*. https://en.wikipedia.org/wiki/Directory_service. Voor het laatst aangepast op 26-11-2016. Bekeken op 12-12-2016.
7. Open Database Connectivity. *Wikipedia*. https://en.wikipedia.org/wiki/Open_Database_Connectivity. Voor het laatst aangepast op 9-12-2016. Bekeken op 12-12-2016.
8. Lightweight Directory Access Protocol. *Wikipedia*. https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol. Voor het laatst aangepast op 12-12-2016. Bekeken op 12-12-2016.
9. Dependency injection. *Wikipedia*. https://en.wikipedia.org/wiki/Dependency_injection. Voor het laatst aangepast op 11-12-2016. Bekeken op 12-12-2016.
10. Ninject – Open source dependency injector for .NET. *Ninject.org*. http://www.ninject.org/. Bekeken op 12-12-2016.
11. Manifest voor Agile Software Ontwikkeling. *Agile Manifesto*. http://agilemanifesto.org/iso/nl/manifesto.html. Gepubliceerd in 2001. Bekeken op 7-12-2016.
12. Agile software development. *Wikipedia*. https://en.wikipedia.org/wiki/Agile_software_development. Voor het laatst aangepast op 6-12-2016. Bekeken op 9-12-2016.
13. Scrum (software development). *Wikipedia*. https://en.wikipedia.org/wiki/Scrum_(software_development). Voor het laatst aangepast op 6-12-2016. Bekeken op 8-12-2016.
14. Fowler M. *UML Distilled: A Brief Guide tot he Standard Object Modeling Language*. 3e Editie. Boston, MA: Addison-Wesley Professional; 2004.
15. Shalloway A, Trott JR. *Design Patterns Explained: A New Perspective on Object-Oriented Design*. 2e Editie. Pearson Education; 2004.
16. Martin RC. *Agile Software Development: Principles, Patterns, and Practices*. Pearson Education; 2003.
17. Hunt A, Thomas D. *The Pragmatic Programmer: From Journeyman to Master*. 18e Editie. Addison-Wesley; 2006.
18. Framework Design Guidelines. *msdn.microsoft.com* https://msdn.microsoft.com/en-us/library/ms229042(v=vs.110).aspx. Bekeken op 14-10-2016.
19. Cwalina K, Abrams B. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries*. 2e Editie. Boston, MA: Addison-Wesley Professional; 2008.

3 Relational Representation Model

Hieronder is de tabelstructuur te zien van de lokale dataopslag van de Data Repository module van HoastAR.

Onderstreepte attributen zijn de Primary Keys van de tabellen.
Dikgedrukte attributen zijn attributen die geen NULL waarde mogen bevatten.

Tabel	Attributen
Contact*	(**Id**, Zip, City, Company, Custom0, Custom1, Custom2, Custom3, Custom4, Custom5, Custom6, Custom7, Custom8, Custom9, DisplayName, FirstName, LastName, **SipAddress**, Street, Uri, LocalContact, ExtensionUrl, ContactSource)
TableInformation	(**DC**, **CurrentTable**)

**Iedere bron waarvan data wordt gerepliceerd heeft een eigen tabel met contacten. De tabelnaam van deze tabel wordt gegenereerd door de Data Repository. Deze tabellen hebben dus een random naam om naam conflicten te vermijden met tabelnamen in de database.*

Zoals hierboven is te lezen, heeft iedere replicatie bron een eigen Contact tabel met attributen in de structuur van de klasse ContactData, om te voldoen aan requirement FR4. In de TableInformation tabel wordt bijgehouden wat de huidige tabelnaam is voor een specifieke databron.

4 Requirements

De client applicatie ToastAR vraagt contact data op bij de Windows Service HoastAR. HoastAR moet gevoed worden met contact data uit andere data bronnen. De te ontwikkelen module zal data uit deze bronnen repliceren, lokaal opslaan en leveren aan HoastAR.

De data bronnen zijn CSV bestanden, Excel bestanden en systemen bereikbaar via ODBC. De huidige software die vervangen wordt bezit deze functies namelijk al.

De contact data bestaat uit statische contactgegevens zoals NAW-gegevens en telefoonnummers.

De interface waarmee HoastAR de data repository service zal ondervragen ligt al grotendeels vast. Er zal gezocht worden naar contacten via naam, telefoonnummer of SIP adres.

De te maken softwaremodule zal Data Repository genoemd worden, in de requirements zal deze naam ook gebruikt worden.
De Data Repository zal aan de volgende requirements moeten voldoen.

4.1.1 Functionele requirements

- FR1: De Data Repository moet lokaal data gestructureerd kunnen opslaan.
- FR2: De Data Repository moet data uit externe bron kunnen repliceren en deze lokaal opslaan.
- FR3: De Data Repository moet voorbereid zijn op uitbreidingen in de vorm van modules die data van specifieke externe data bronnen kunnen repliceren.
- FR4: De Data Repository moet lokale data kunnen leveren in een vooraf vastgesteld format. Dit is de ContactData klasse. (Zie bijlage)
- FR5: De Data Repository moet iedere replicatie kunnen inplannen aan de hand van instellingen geleverd vanuit HoastAR.
- FR6: De Data Repository moet een lijst met alle ingeplande replicaties kunnen leveren aan de HoastAR.
- FR7: De Data Repository moet iedere replicatie op een bepaalde start tijd kunnen uitvoeren met als optie een bepaald interval tussen de hierop volgende replicaties.
- FR8: De HoastAR moet kunnen achterhalen wat de specifieke gegevens zijn die ieder type replicatie module nodig heeft om te kunnen functioneren.

4.1.2 Niet-functionele requirements

- NFR1: Zoekopdrachten via de attributen SipAddress en Uri moeten binnen 0,1 seconde resultaat leveren op een database met 10.000 records.
- NFR2: Zoekopdrachten via de attributen Company en DisplayName moeten binnen 1 seconde resultaat leveren op een database met 10.000 records.

- NFR3: Tijdens het repliceren van data uit externe bronnen bij een normale serverload moet de lokale data 95% van de tijd beschikbaar zijn.

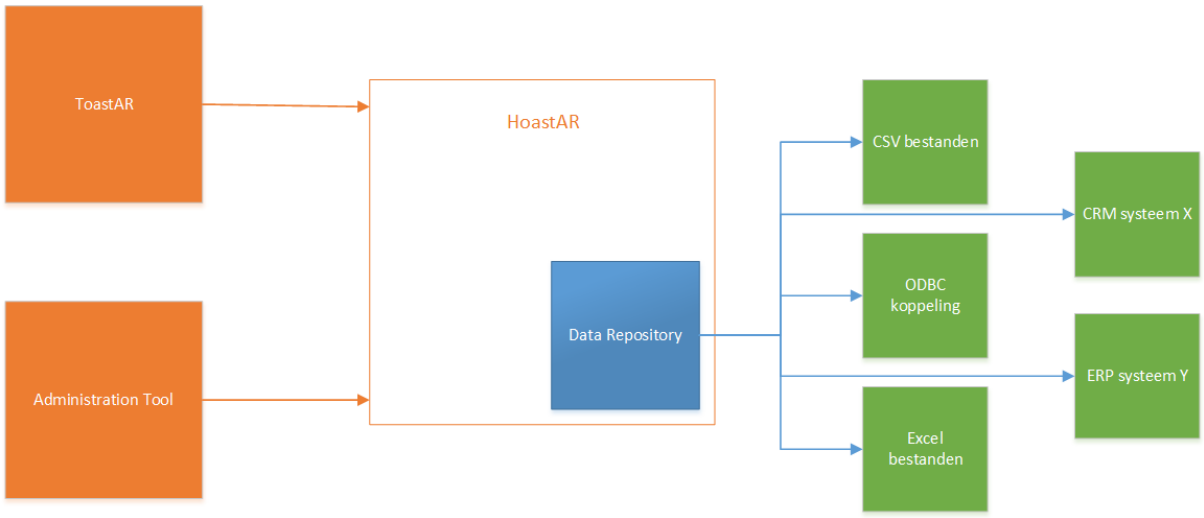
4.1.3 Scope

- De Data Repository wordt onderdeel van een Windows service die de Data Repository intern zal aanroepen. De Data Repository is niet vereist instellingen bij te houden voor de werking van de Windows service, maar moet wel zelf onderhoudend zijn.
- De beveiliging van de lokale data heeft een lagere prioriteit en valt buiten de scope.

4.2 Interactie met andere systemen

De directory service wordt onderdeel van een Windows service genaamd HoastAR die de Data Repository intern zal aanroepen.

HoastAR zelf is bereikbaar voor client applicaties zoals ToastAR en een administrator tool.



Figuur 1. Systeem Context Diagram nieuwe situatie.

Hierboven is te zien wat interactie is van de Data Repository met andere systemen en in welke context het zicht bevindt. Zoals in deze figuur te zien is wordt de Data Repository een module van de HoastAR. De HoastAR wordt gebruikt als server van de client applicatie ToastAR en een Administration Tool. Alle gegevens die ToastAR en de administratie tool nodig hebben worden geleverd door de HoastAR.
Verder is te zien dat de Data Repository in de nieuwe situatie uit verscheidene databronnen data op kan halen.

4.3 Legacy systemen

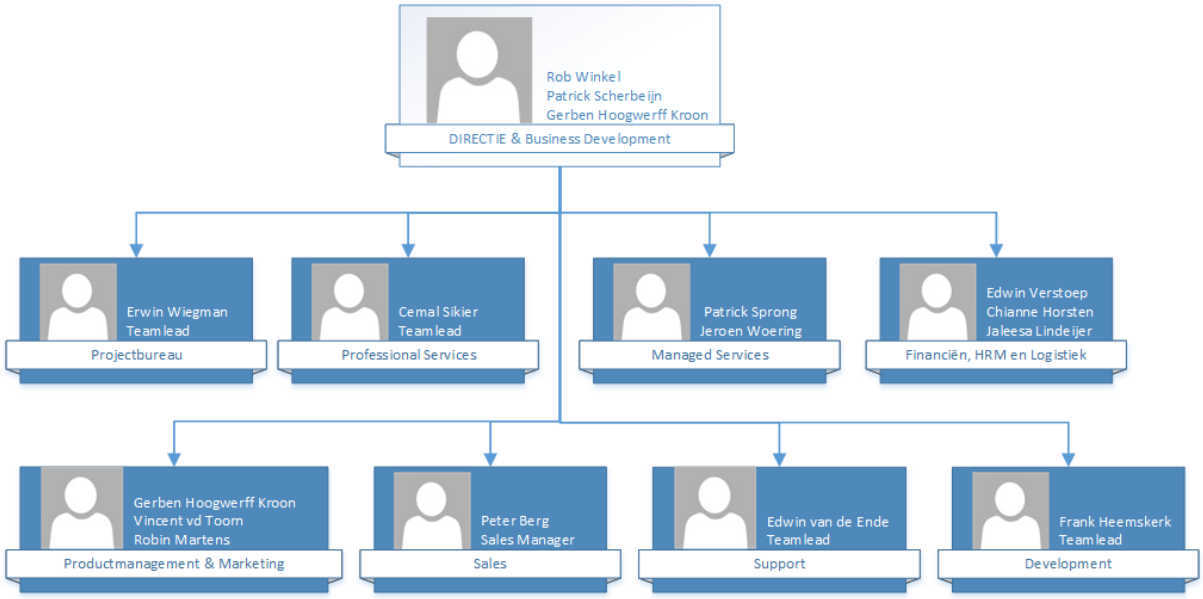
De Data Repository vervangt de huidige directory service genaamd MetaDirectory gemaakt door Estos die nu gebruikt wordt om data uit verschillende bronnen te repliceren en lokaal op te slaan.

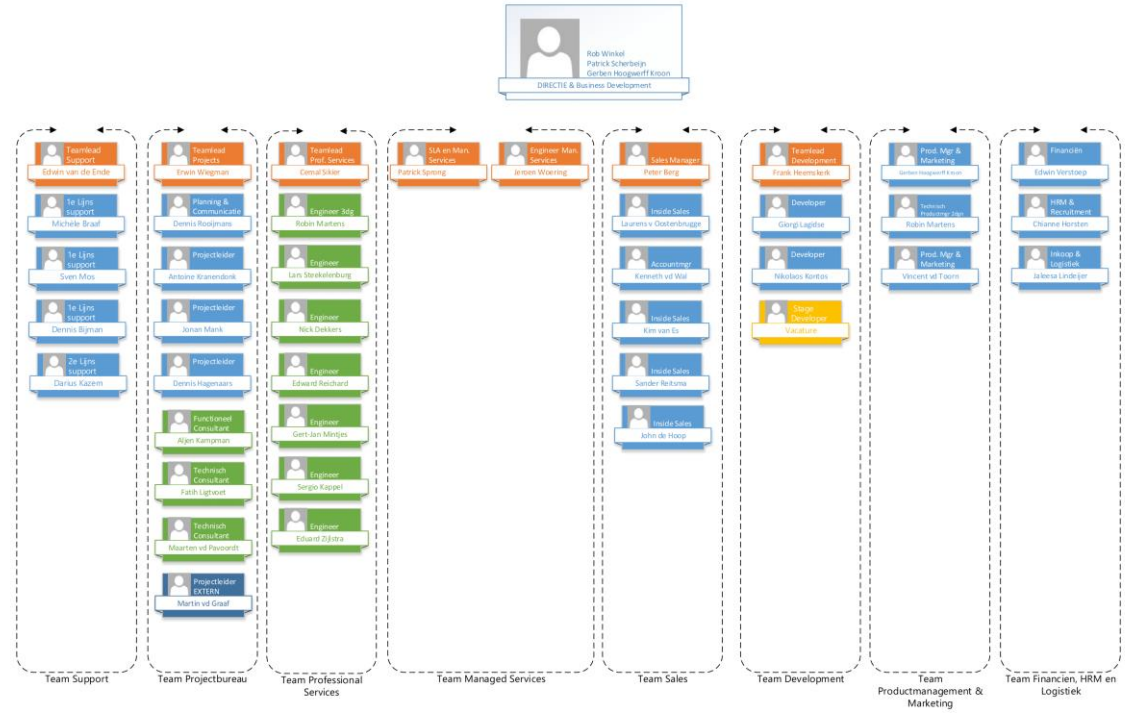
4.4 Bijlage: ContactData klasse

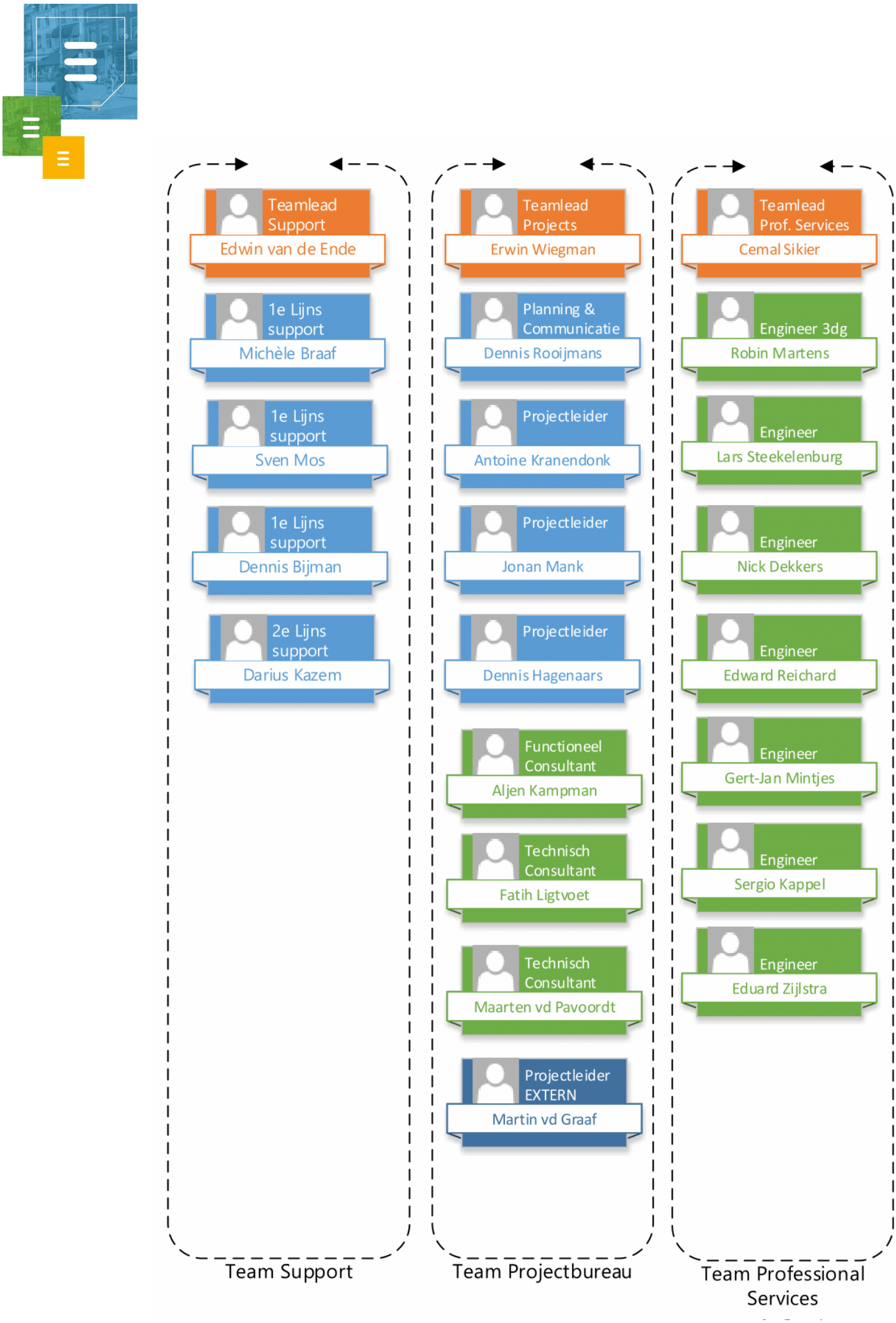
```
public class ContactData
  {
    public string Zip { get; set; }
    public string City { get; set; }
    public string Company { get; set; }
    public string Custom0 { get; set; }
    public string Custom1 { get; set; }
    public string Custom2 { get; set; }
    public string Custom3 { get; set; }
    public string Custom4 { get; set; }
    public string Custom5 { get; set; }
    public string Custom6 { get; set; }
    public string Custom7 { get; set; }
    public string Custom8 { get; set; }
    public string Custom9 { get; set; }
    public string Custom10 { get; set; }
    public string Custom11 { get; set; }
    public string Custom12 { get; set; }
    public string Custom13 { get; set; }
    public string Custom14 { get; set; }
    public string Custom15 { get; set; }
    public string Custom16 { get; set; }
    public string Custom17 { get; set; }
    public string Custom18 { get; set; }
    public string Custom19 { get; set; }
    public string Email { get; set; }
    public string DisplayName { get; set; }
    public string FirstName { get; set; }
    public string Id { get; set; }
    public string LastName { get; set; }
    public string SipAddress { get; set; }
    public string Street { get; set; }
    public string Uri { get; set; }

    public Dictionary<string, string> ContactUris { get; private set; }

    public ContactData() { ContactUris = new Dictionary<string, string>(); }


    /// <summary>
    /// E.g. Outlook contacts will be local contacts, non-local will be prefered
    /// </summary>
    public bool LocalContact { get; set; }
    /// <summary>
    /// The url to show extra data for this contact
    /// </summary>
    public string ExtensionUrl { get; set; }
    /// <summary>
```

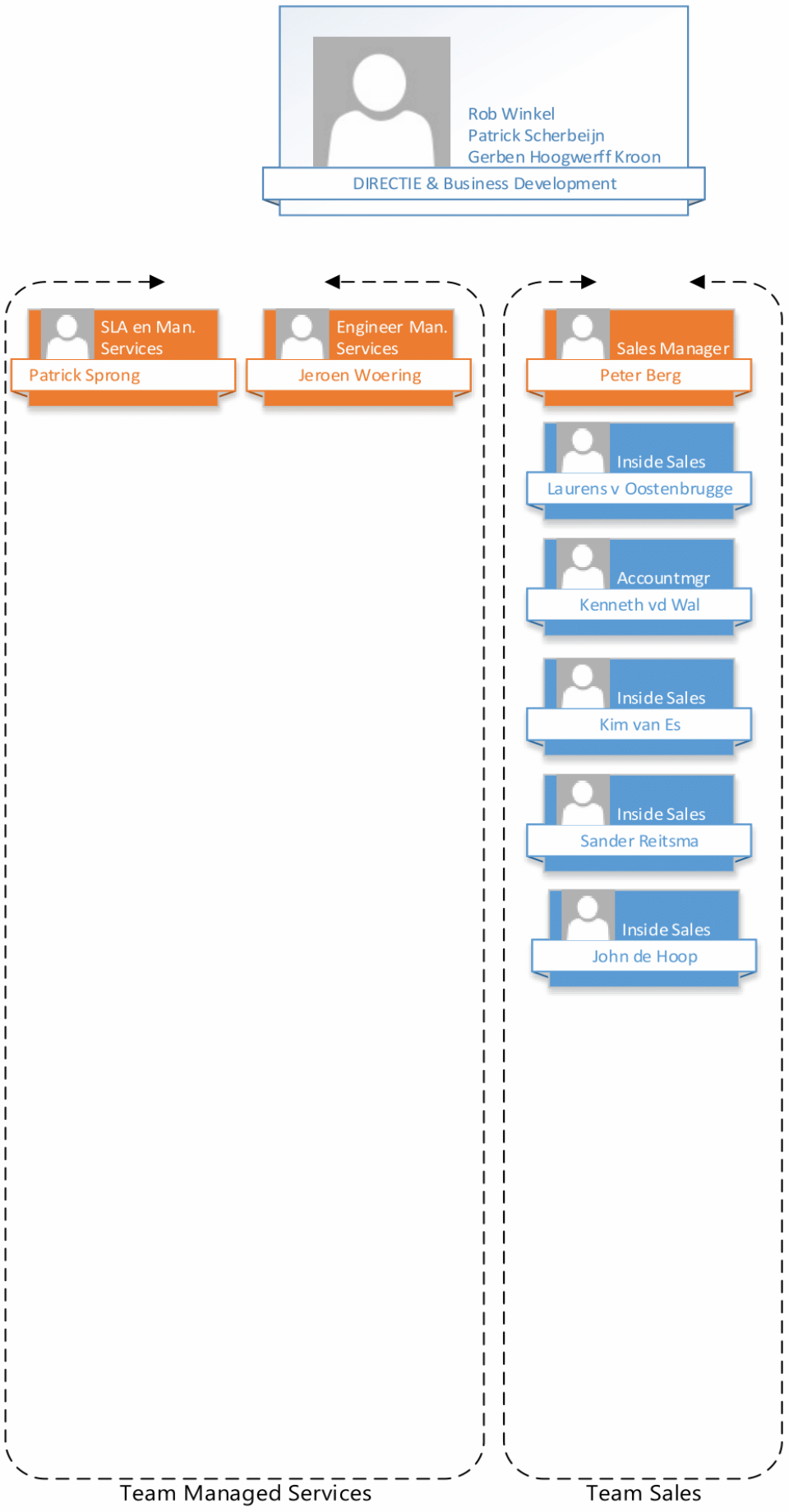
```
        /// The friendly name of the source where the contact was found
        /// </summary>
        public string ContactSource { get; set; }
        public string ContactUri0 { get; set; }
        public string ContactUri1 { get; set; }
        public string ContactUri2 { get; set; }
        public string ContactUri3 { get; set; }
        public string ContactUri4 { get; set; }
        public string ContactUri5 { get; set; }
    }
```

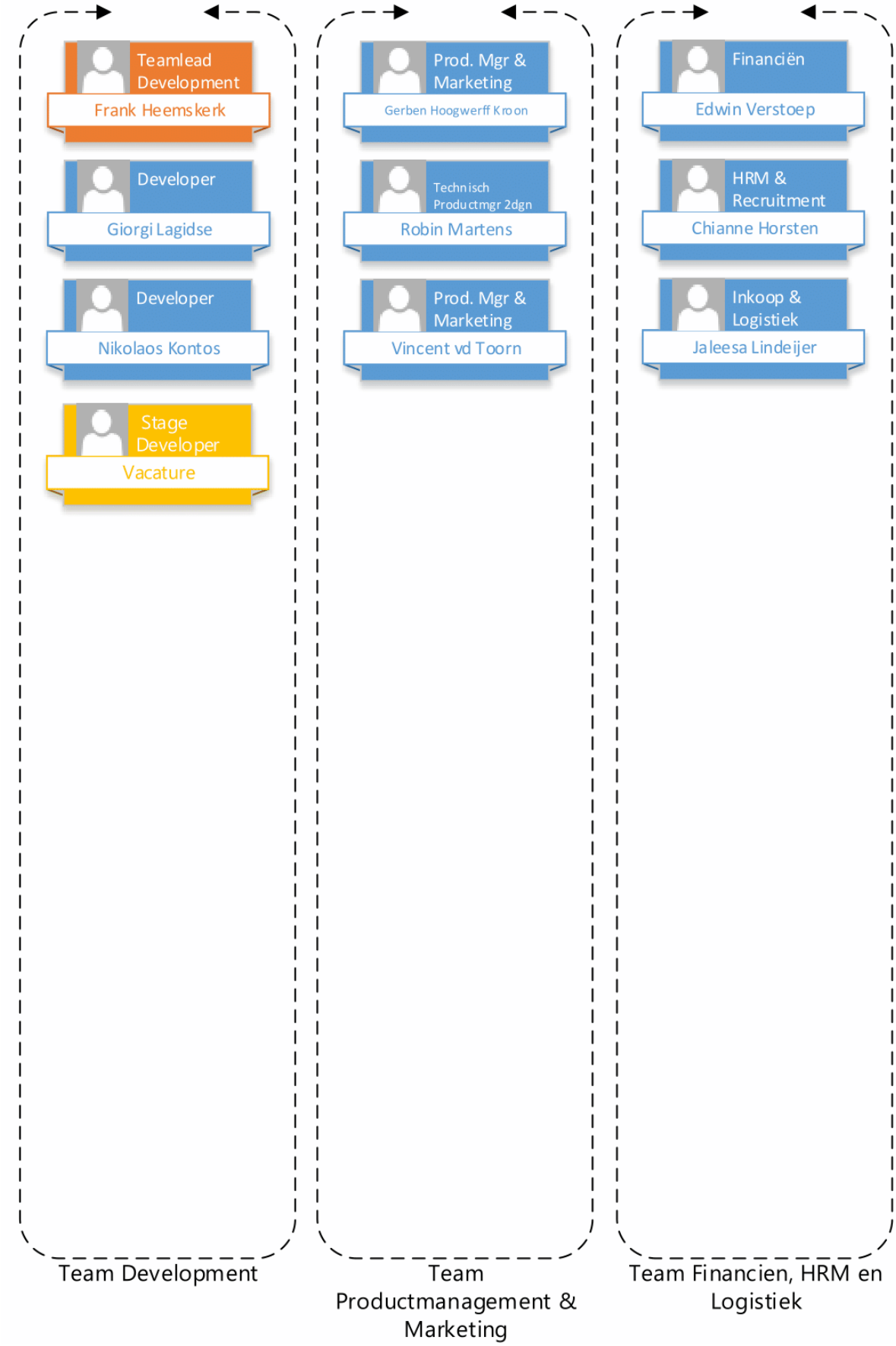
5 Organogram



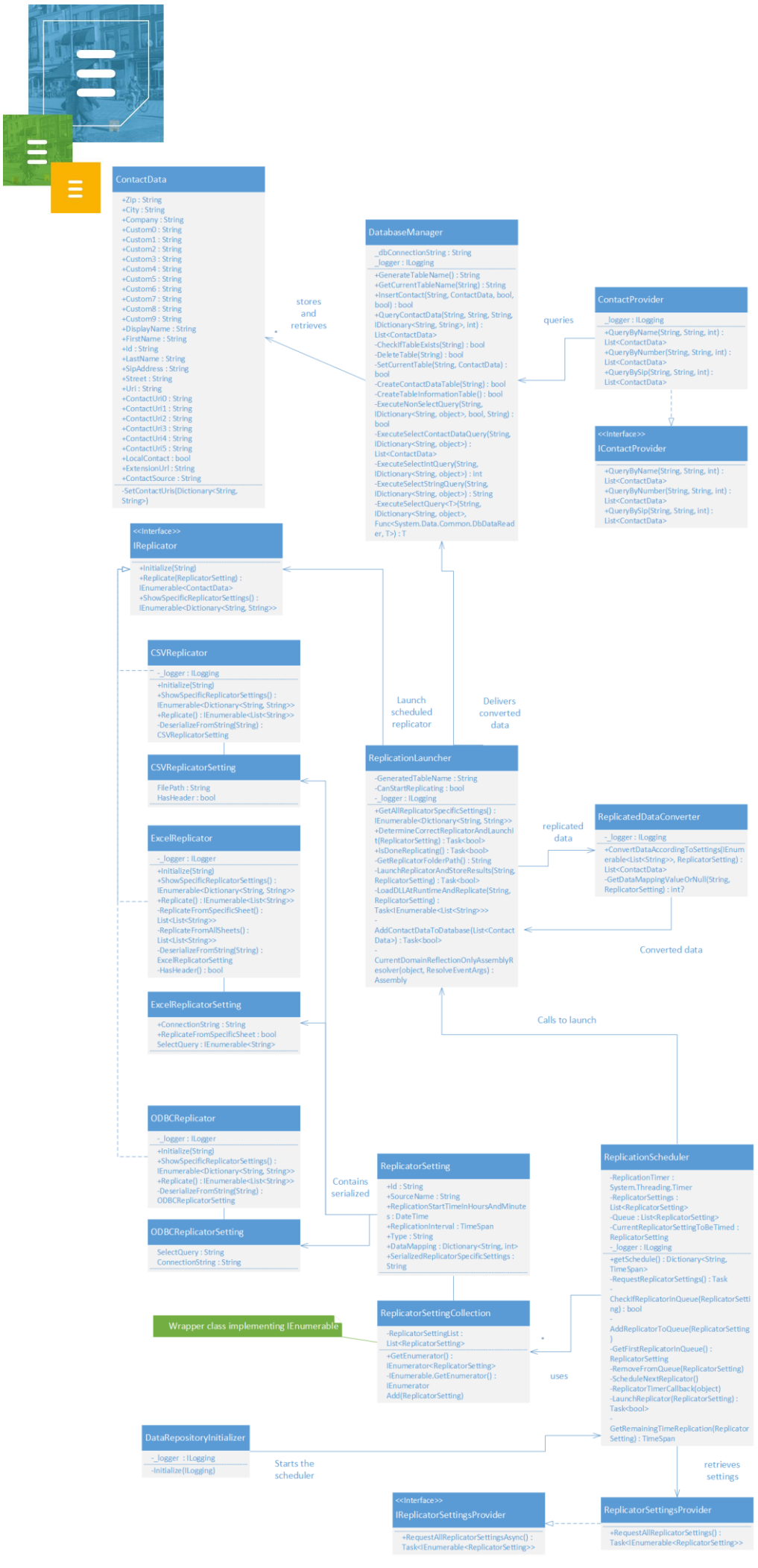


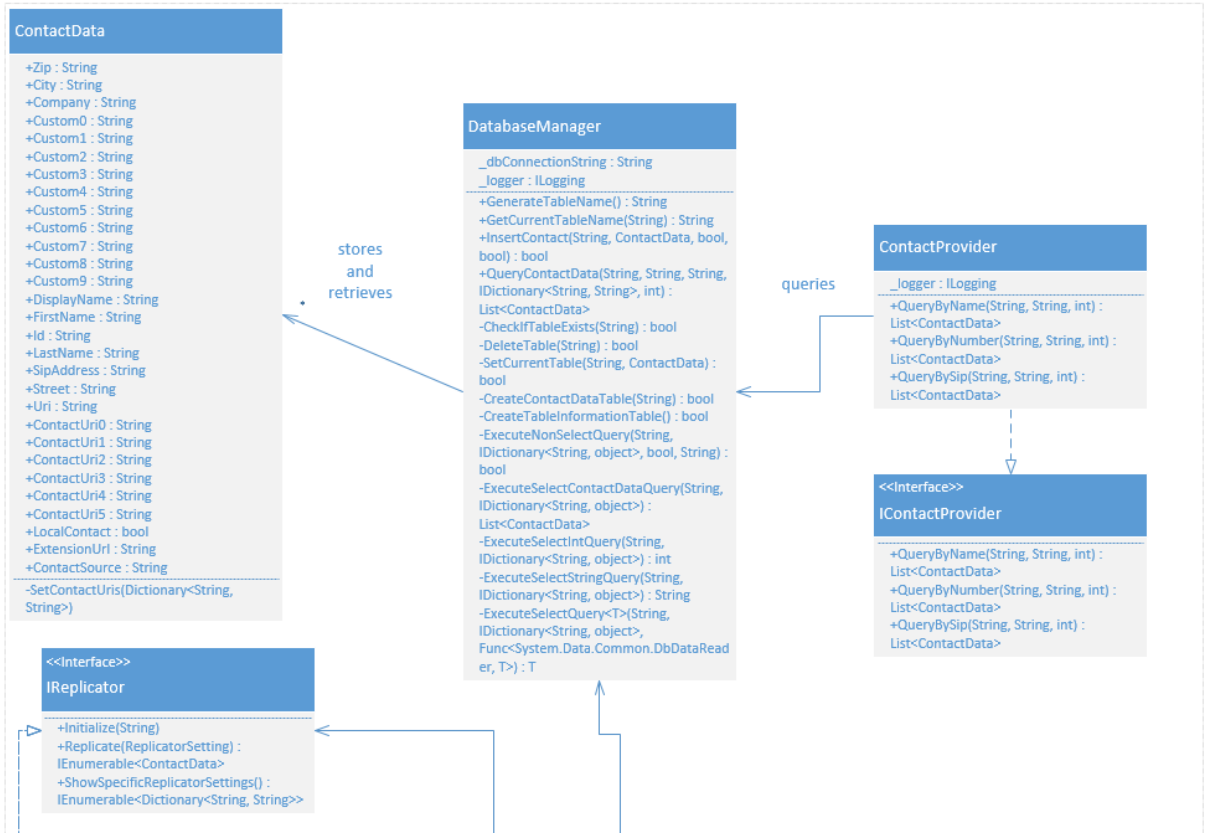

Teamlead Support
Edwin van de Ende
1e Lijns support
Michèle Braaf
1e Lijns support
Sven Mos
1e Lijns support
Dennis Bijman
2e Lijns support
Darius Kazem
Team Support
Teamlead Projects
Erwin Wiegman
Planning & Communicatie
Dennis Rooijmans
Projectleider
Antoine Kranendonk
Projectleider
Jonan Mank
Projectleider
Dennis Hagenaars
Functioneel Consultant
Aljen Kampman
Technisch Consultant
Fatih Ligtvoet
Technisch Consultant
Maarten vd Pavoordt
Projectleider EXTERN
Martin vd Graaf
Team Projectbureau
Teamlead Prof. Services
Cemal Sikier
Engineer 3dg
Robin Martens
Engineer
Lars Steekelenburg
Engineer
Nick Dekkers
Engineer
Edward Reichard
Engineer
Gert-Jan Mintjes
Engineer
Sergio Kappel
Engineer
Eduard Zijlstra
Team Professional Services

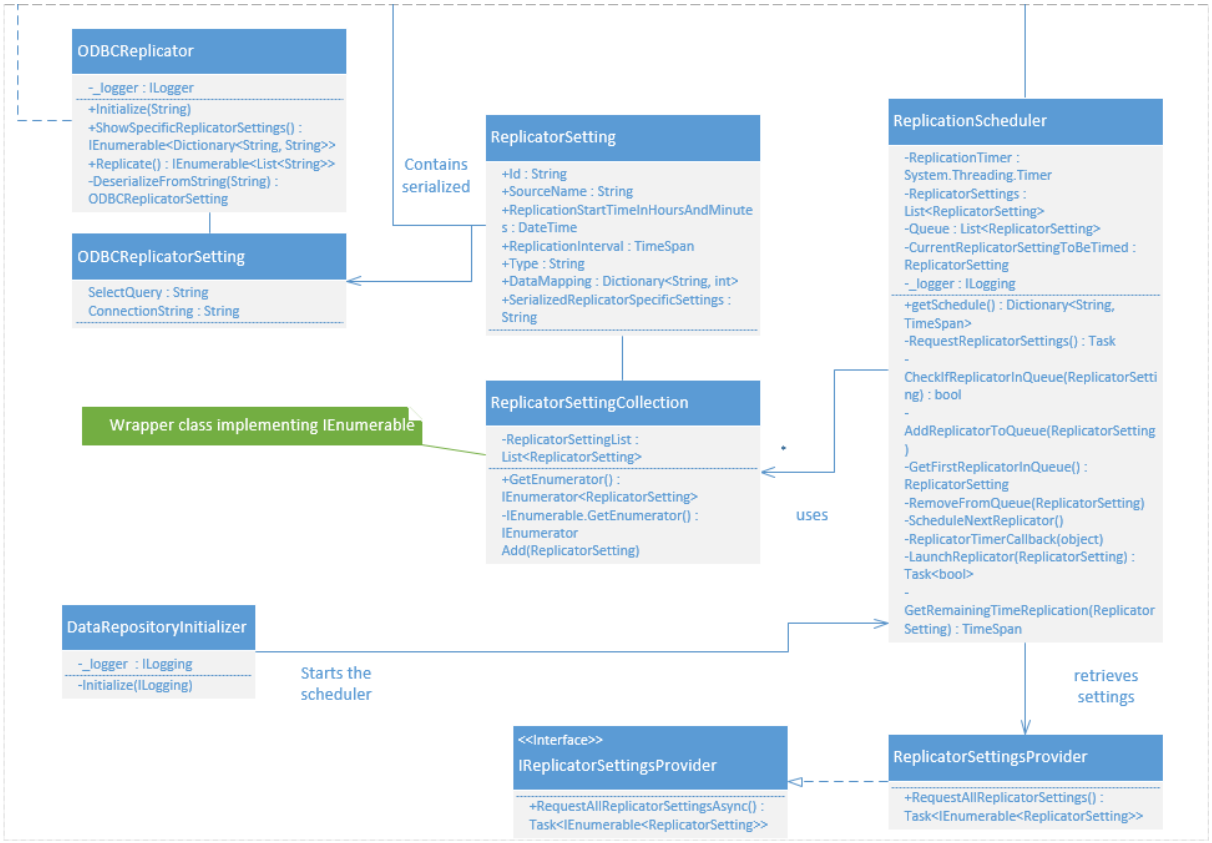

Rob Winkel
Patrick Scherbeijn
Gerben Hoogwerff Kroon
DIRECTIE & Business Development
SLA en Man. Services
Patrick Sprong
Engineer Man. Services
Jeroen Woering
Sales Manager
Peter Berg
Inside Sales
Laurens v Oostenbrugge
Accountmgr
Kenneth vd Wal
Inside Sales
Kim van Es
Inside Sales
Sander Reitsma
Inside Sales
John de Hoop
Team Managed Services
Team Sales


Teamlead Development
Frank Heemskerk
Developer
Giorgi Lagidse
Developer
Nikolaos Kontos
Stage Developer
Vacature
Prod. Mgr & Marketing
Gerben Hoogwerff Kroon
Technisch Productmgr 2dgn
Robin Martens
Prod. Mgr & Marketing
Vincent vd Toorn
Financiën
Edwin Verstoep
HRM & Recruitment
Chianne Horsten
Inkoop & Logistiek
Jaleesa Lindeijer
Team Development
Team Productmanagement & Marketing
Team Financien, HRM en Logistiek

6 Klassendiagram


ContactData
+Zip : String
+City : String
+Company : String
+Custom0 : String
+Custom1 : String
+Custom2 : String
+Custom3 : String
+Custom4 : String
+Custom5 : String
+Custom6 : String
+Custom7 : String
+Custom8 : String
+Custom9 : String
+DisplayName : String
+FirstName : String
+Id : String
+LastName : String
+SipAddress : String
+Street : String
+Uri : String
+ContactUri0 : String
+ContactUri1 : String
+ContactUri2 : String
+ContactUri3 : String
+ContactUri4 : String
+ContactUri5 : String
+LocalContact : bool
+ExtensionUri : String
+ContactSource : String
-SetContactUris(Dictionary<String, String>)
stores and retrieves
DatabaseManager
_dbConnectionString : String
_logger : ILogging
+GenerateTableName() : String
+GetCurrentTableName(String) : String
+InsertContact(String, ContactData, bool, bool) : bool
+QueryContactData(String, String, String, IDictionary<String, String>, int) : List<ContactData>
-CheckIfTableExists(String) : bool
-DeleteTable(String) : bool
-SetCurrentTable(String, ContactData) : bool
-CreateContactDataTable(String) : bool
-CreateTableInformationTable() : bool
-ExecuteNonSelectQuery(String, IDictionary<String, object>, bool, String) : bool
-ExecuteSelectContactDataQuery(String, IDictionary<String, object>) : List<ContactData>
-ExecuteSelectIntQuery(String, IDictionary<String, object>) : int
-ExecuteSelectStringQuery(String, IDictionary<String, object>) : String
-ExecuteSelectQuery<T>(String, IDictionary<String, object>, Func<System.Data.Common.DbDataReader, T>) : T
queries
ContactProvider
_logger : ILogging
+QueryByName(String, String, int) : List<ContactData>
+QueryByNumber(String, String, int) : List<ContactData>
+QueryBySip(String, String, int) : List<ContactData>
<<Interface>>
IContactProvider
+QueryByName(String, String, int) : List<ContactData>
+QueryByNumber(String, String, int) : List<ContactData>
+QueryBySip(String, String, int) : List<ContactData>
<<Interface>>
IReplicator
+Initialize(String)
+Replicate(ReplicatorSetting) : IEnumerable<ContactData>
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
CSVReplicator
-_logger : ILogging
+Initialize(String)
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
+Replicate() : IEnumerable<List<String>>
-DeserializeFromString(String) : CSVReplicatorSetting
CSVReplicatorSetting
FilePath : String
HasHeader : bool
Launch scheduled replicator
Delivers converted data
ReplicationLauncher
-GeneratedTableName : String
-CanStartReplicating : bool
-_logger : ILogging
+GetAllReplicatorSpecificSettings() : IEnumerable<Dictionary<String, String>>
+DetermineCorrectReplicatorAndLaunch(ReplicatorSetting) : Task<bool>
+IsDoneReplicating() : Task<bool>
-GetReplicatorFolderPath() : String
-LaunchReplicatorAndStoreResults(String, ReplicatorSetting) : Task<bool>
-LoadDLLAtRuntimeAndReplicate(String, ReplicatorSetting) : Task<IEnumerable<List<String>>>
-AddContactDataToDatabase(List<ContactData>) : Task<bool>
-CurrentDomainReflectionOnlyAssemblyResolver(object, ResolveEventArgs) : Assembly
replicated data
ReplicatedDataConverter
-_logger : ILogging
+ConvertDataAccordingToSettings(IEnumerable<List<String>>, ReplicatorSetting) : List<ContactData>
-GetDataMappingValueOrNull(String, ReplicatorSetting) : int?
Converted data
ExcelReplicator
-_logger : ILogger
+Initialize(String)
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
+Replicate() : IEnumerable<List<String>>
-ReplicateFromSpecificSheet() : List<List<String>>
-ReplicateFromAllSheets() : List<List<String>>
-DeserializeFromString(String) : ExcelReplicatorSetting
-HasHeader() : bool
ExcelReplicatorSetting
+ConnectionString : String
+ReplicateFromSpecificSheet : bool
SelectQuery : IEnumerable<String>
Calls to launch
ODBCReplicator
-_logger : ILogger
+Initialize(String)
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
+Replicate() : IEnumerable<List<String>>
-DeserializeFromString(String) : ODBCReplicatorSetting
ODBCReplicatorSetting
SelectQuery : String
ConnectionString : String
Contains serialized
ReplicatorSetting
+Id : String
+SourceName : String
+ReplicationStartTimeInHoursAndMinutes : DateTime
+ReplicationInterval : TimeSpan
+Type : String
+DataMapping : Dictionary<String, int>
+SerializedReplicatorSpecificSettings : String
ReplicationScheduler
-ReplicationTimer : System.Threading.Timer
-ReplicatorSettings : List<ReplicatorSetting>
-Queue : List<ReplicatorSetting>
-CurrentReplicatorSettingToBeTimed : ReplicatorSetting
-_logger : ILogging
+getSchedule() : Dictionary<String, TimeSpan>
-RequestReplicatorSettings() : Task
-CheckIfReplicatorInQueue(ReplicatorSetting) : bool
-AddReplicatorToQueue(ReplicatorSetting)
-GetFirstReplicatorInQueue() : ReplicatorSetting
-RemoveFromQueue(ReplicatorSetting)
-ScheduleNextReplicator()
-ReplicatorTimerCallback(object)
-LaunchReplicator(ReplicatorSetting) : Task<bool>
-GetRemainingTimeReplication(ReplicatorSetting) : TimeSpan
Wrapper class implementing IEnumerable
ReplicatorSettingCollection
-ReplicatorSettingList : List<ReplicatorSetting>
+GetEnumerator() : IEnumerator<ReplicatorSetting>
-IEnumerable.GetEnumerator() : IEnumerator
Add(ReplicatorSetting)
uses
DataRepositoryInitializer
-_logger : ILogging
-Initialize(ILogging)
Starts the scheduler
retrieves settings
<<Interface>>
IReplicatorSettingsProvider
+RequestAllReplicatorSettingsAsync() : Task<IEnumerable<ReplicatorSetting>>
ReplicatorSettingsProvider
+RequestAllReplicatorSettings() : Task<IEnumerable<ReplicatorSetting>>


ContactData
+Zip : String
+City : String
+Company : String
+Custom0 : String
+Custom1 : String
+Custom2 : String
+Custom3 : String
+Custom4 : String
+Custom5 : String
+Custom6 : String
+Custom7 : String
+Custom8 : String
+Custom9 : String
+DisplayName : String
+FirstName : String
+Id : String
+LastName : String
+SipAddress : String
+Street : String
+Uri : String
+ContactUri0 : String
+ContactUri1 : String
+ContactUri2 : String
+ContactUri3 : String
+ContactUri4 : String
+ContactUri5 : String
+LocalContact : bool
+ExtensionUrl : String
+ContactSource : String
-SetContactUris(Dictionary<String, String>)
stores and retrieves
DatabaseManager
_dbConnectionString : String
_logger : ILogging
+GenerateTableName() : String
+GetCurrentTableName(String) : String
+InsertContact(String, ContactData, bool, bool) : bool
+QueryContactData(String, String, String, IDictionary<String, String>, int) : List<ContactData>
-CheckIfTableExists(String) : bool
-DeleteTable(String) : bool
-SetCurrentTable(String, ContactData) : bool
-CreateContactDataTable(String) : bool
-CreateTableInformationTable() : bool
-ExecuteNonSelectQuery(String, IDictionary<String, object>, bool, String) : bool
-ExecuteSelectContactDataQuery(String, IDictionary<String, object>) : List<ContactData>
-ExecuteSelectIntQuery(String, IDictionary<String, object>) : int
-ExecuteSelectStringQuery(String, IDictionary<String, object>) : String
-ExecuteSelectQuery<T>(String, IDictionary<String, object>, Func<System.Data.Common.DbDataReader, T>) : T
queries
ContactProvider
_logger : ILogging
+QueryByName(String, String, int) : List<ContactData>
+QueryByNumber(String, String, int) : List<ContactData>
+QueryBySip(String, String, int) : List<ContactData>
<<Interface>>
IContactProvider
+QueryByName(String, String, int) : List<ContactData>
+QueryByNumber(String, String, int) : List<ContactData>
+QueryBySip(String, String, int) : List<ContactData>
<<Interface>>
IReplicator
+Initialize(String)
+Replicate(ReplicatorSetting) : IEnumerable<ContactData>
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
CSVReplicator
-_logger : ILogging
+Initialize(String)
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
+Replicate() : IEnumerable<List<String>>
-DeserializeFromString(String) : CSVReplicatorSetting
CSVReplicatorSetting
FilePath : String
HasHeader : bool
ExcelReplicator
-_logger : ILogger
+Initialize(String)
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
+Replicate() : IEnumerable<List<String>>
-ReplicateFromSpecificSheet() : List<List<String>>
-ReplicateFromAllSheets() : List<List<String>>
-DeserializeFromString(String) : ExcelReplicatorSetting
-HasHeader() : bool
ExcelReplicatorSetting
+ConnectionString : String
+ReplicateFromSpecificSheet : bool
SelectQuery : IEnumerable<String>
Launch scheduled replicator
Delivers converted data
ReplicatorLauncher
-GeneratedTableName : String
-CanStartReplicating : bool
-_logger : ILogging
+GetAllReplicatorSpecificSettings() : IEnumerable<Dictionary<String, String>>
+DetermineCorrectReplicatorAndLaunchIt(ReplicatorSetting) : Task<bool>
+IsDoneReplicating() : Task<bool>
-GetReplicatorFolderPath() : String
-LaunchReplicatorAndStoreResults(String, ReplicatorSetting) : Task<bool>
-LoadDLLAtRuntimeAndReplicate(String, ReplicatorSetting) : Task<IEnumerable<List<String>>>
-AddContactDataToDatabase(List<ContactData>) : Task<bool>
-CurrentDomainReflectionOnlyAssemblyResolver(object, ResolveEventArgs) : Assembly
replicated data
ReplicatedDataConverter
-_logger : ILogging
+ConvertDataAccordingToSettings(IEnumerable<List<String>>, ReplicatorSetting) : List<ContactData>
-GetDataMappingValueOrNull(String, ReplicatorSetting) : int?
Converted data
Calls to launch


ODBCReplicator
- _logger : ILogger
+Initialize(String)
+ShowSpecificReplicatorSettings() : IEnumerable<Dictionary<String, String>>
+Replicate() : IEnumerable<List<String>>
-DeserializeFromString(String) : ODBCReplicatorSetting
ODBCReplicatorSetting
SelectQuery : String
ConnectionString : String
Contains serialized
ReplicatorSetting
+Id : String
+SourceName : String
+ReplicationStartTimeInHoursAndMinutes : DateTime
+ReplicationInterval : TimeSpan
+Type : String
+DataMapping : Dictionary<String, int>
+SerializedReplicatorSpecificSettings : String
Wrapper class implementing IEnumerable
ReplicatorSettingCollection
-ReplicatorSettingList : List<ReplicatorSetting>
+GetEnumerator() : IEnumerator<ReplicatorSetting>
-IEnumerable.GetEnumerator() : IEnumerator
Add(ReplicatorSetting)
uses
ReplicationScheduler
-ReplicationTimer : System.Threading.Timer
-ReplicatorSettings : List<ReplicatorSetting>
-Queue : List<ReplicatorSetting>
-CurrentReplicatorSettingToBeTimed : ReplicatorSetting
-_logger : ILogging
+getSchedule() : Dictionary<String, TimeSpan>
-RequestReplicatorSettings() : Task
-CheckIfReplicatorInQueue(ReplicatorSetting) : bool
-AddReplicatorToQueue(ReplicatorSetting)
-GetFirstReplicatorInQueue() : ReplicatorSetting
-RemoveFromQueue(ReplicatorSetting)
-ScheduleNextReplicator()
-ReplicatorTimerCallback(object)
-LaunchReplicator(ReplicatorSetting) : Task<bool>
-GetRemainingTimeReplication(ReplicatorSetting) : TimeSpan
DataRepositoryInitializer
-_logger : ILogging
-Initialize(ILogging)
Starts the scheduler
retrieves settings
<<Interface>>
IReplicatorSettingsProvider
+RequestAllReplicatorSettingsAsync() : Task<IEnumerable<ReplicatorSetting>>
ReplicatorSettingsProvider
+RequestAllReplicatorSettings() : Task<IEnumerable<ReplicatorSetting>>

7 Mastertestplan

7.1 Samenvatting

In dit Mastertestplan staat beschreven welke testsoorten er gebruikt zullen worden in dit project.[1] Bij het maken van dit Mastertestplan maak ik gebruik van een template van TMAP.[2,3]

Ik geef per testsoort een beschrijving met de onderbouwing voor de keuze van de testsoort.

7.2 Inhoudsopgave

7.3 Testsoorten

Voor dit project heb ik drie testsoorten gekozen om het systeem mee te testen. Dit zijn: Moduletesten, Systeemtesten en een Systeem integratietest.

7.3.1 Moduletest

Het doel van een moduletest, ook wel unit test genoemd, is het testen van een specifiek onderdeel van een systeem. Een moduletest is een white-box test. Moduletesten richten zich op de elementaire bouwblokken in de code. Ze tonen aan dat de modules voldoen aan het technisch ontwerp.[1] Unit tests worden gemaakt in de vorm van code fragmenten die code van de te testen module aanroepen en controleren op een verwacht resultaat.

Als een systeemdeel getest is met unit tests kan dit systeemdeel eventueel hergebruikt worden met de zekerheid dat dit deel functioneert. Een ander voordeel is de her testbaarheid door middel van unit tests. Als er gewerkt is aan het systeem kan er snel getest worden of dit geen ongewenste effecten heeft gehad op eerder gemaakte modules van het systeem.

Voor de moduletesten is een Detailtestplan gemaakt, zie bijlage 8.

7.3.2 Systeemtest

Het doel van een systeemtest is zoals de naam al aangeeft het hele systeem testen. De systeemtest is een black-box test. Testen gebeurt vaak aan de hand van de reeds ontwikkelde systeeminterfaces. De systeemtest toont aan dat het systeem werkt conform het functionele ontwerp.

Ik heb gekozen om systeem testen toe te passen om zeker te zijn dat het systeem werkt en geen fouten heeft die mogelijk langs de unit testen zijn geglipt.

7.3.3 Systeem integratie test

Het doel van een systeem integratie test, ook wel ketentest genoemd, is het testen van het functioneren van een systeem in combinatie met andere systemen. Bij een ketentest wordt de samenwerking van het systeem met de aanliggende systemen getest. In de ketentest worden deze systemen vaak voor het eerst aan elkaar gekoppeld. De ketentest richt zich op het vinden van de fouten die ontstaan als systemen niet goed samenwerken.

Om zeker te zijn dat de DataRepository goed geïntegreerd wordt in de HoastAR en data via HoastAR aan de ToastAR geleverd kan worden wordt er een systeem integratie test uitgevoerd.

7.4 Literatuurlijst

1. de Grood D-J. Aanpak: Testsoorten. In: de Grood D-J. *TestGoal. Leerboek resultaatgedreven software testen.* 1e druk. Den Haag: Sdu Uitgevers bv; 2008: 49-51.
2. de Grood D-J. Testplan. In: de Grood D-J. *TestGoal. Leerboek resultaatgedreven software testen.* 1e druk. Den Haag: Sdu Uitgevers bv; 2008: 123-143.
3. Nederlandstalige TMAP Downloads. *TMAP*. http://www.tmap.net/nederlandstalige-tmap-downloads. Bekeken op 15-12-2016.

8 Detailtestplan Moduletest

8.1 Samenvatting

In dit testplan staat beschreven wat een moduletest inhoudt voor dit project in hoofdstuk 1. In hoofdstuk 4 en 5 staan de logische en fysieke testgevallen beschreven. Om dit detailtestplan samen te stellen zijn relevante hoofdstukken gebruikt uit het boek TestGoal.[1,2,3,4]

8.2 Inhoudsopgave

8.3 Teststrategie

Moduletesten worden gebruikt om delen van de software te testen op juiste functionaliteiten. Dit zijn meestal geautomatiseerde testen die ook wel Unit Testen worden genoemd.
Door deze unit testen uit te voeren kan worden vastgesteld of de geteste software modules nog functioneren. Dit is handig als er code is aangepast die invloed kan hebben op eerder gemaakte modules.
Omdat er gebruikt wordt gemaakt van een Agile ontwikkelmethodiek, is het goed om van alle gemaakte softwaremodules zeker te zijn dat deze goed functioneren. Er kan namelijk worden besloten bepaalde modules niet te maken, eerder of later te maken.
Moduletesten zorgen voor kwaliteitswaarborging in flexibele ontwikkelmethodieken zoals Agile door de gemaakte software incrementeel te testen. Ieder nieuw systeemdeel ofwel module wordt getest, samen met eerder gemaakte systeemdelen om zo de alle functionaliteiten te waarborgen.
Moduletesten vallen onder White Box Testen. Dit houdt in dat de onderliggende structuur van de te testen module bekend is bij de tester. Er wordt immers direct code aangeroepen van de te testen module in de fysieke testgevallen in de vorm van Unit Tests.

8.4 Producten

De door het testtraject op te leveren producten zijn de logische testgevallen en de fysieke testgevallen in de vorm van unit tests. Zie hoofdstuk 4 en 5 voor deze testgevallen.

8.5 Benodigdheden

Voor de moduletesten is de hoofd benodigdheid de gemaakte code van de Data Repository. De unit testen zullen deze code gebruiken voor de fysieke testgevallen.

8.6 Logische testgevallen

Hier zijn de logische testgevallen beschreven. "Een logisch testgeval omschrijft *wat* er getest dient te worden, een fysiek testgeval omschrijft *hoe* dit moet gebeuren." [3]
Er zijn geen relevante testontwerp technieken gevonden voor het maken van logische testgevallen voor moduletesten[3], er zal hier beschreven worden welke functionaliteiten en eventuele grensgevallen worden getest.

Bij iedere gemaakte klasse worden Unit Testen gemaakt om alle functionaliteiten van die klasse te waarborgen. Hieronder zijn de logische testgevallen per klasse weergegeven.

DatabaseManager

Code testgeval	Beschrijving testgeval
LT01	De gegenereerde tabelnaam mag niet een lege String zijn of een NULL waarde zijn.
LT02	Een ContactData moet correct opgeslagen kunnen worden in de database.
LT03	Er moet een nieuwe tabel aangemaakt worden voordat de eerste ContactData in een collectie van gerepliceerde data daarin opgeslagen wordt.
LT04	Na het invoeren van alle ContactData in een nieuwe tabel moet de nieuwe tabelnaam in de InformationTable tabel staan.
LT05	Als een oude tabel vervangen is voor een nieuwe tabel met de laatste versie van contactgegevens uit een databron, dan moet de oude tabel verwijderd worden uit de database.
LT06	Een verkeerde query moet een SQLiteException veroorzaken.
LT07	Een ContactData invoeren zonder een tabelnaam te leveren moet een Exception veroorzaken.
LT08	Bij het opvragen van de huidige tabelnaam van een bepaalde databron moet de juiste naam geleverd worden.
LT09	Bij het opvragen van de huidige tabelnaam uit een onbekende bron moet een lege String geleverd worden.
LT10	Bij het zoeken naar een ContactData in de database moet de correcte ContactData geleverd worden.
LT11	Het zoeken naar een ContactData in een niet bestaande tabel moet een SQLiteException veroorzaken.

LT12	Het zoeken naar een ContactData zonder alle benodigde parameters mee te leveren moet een Exception veroorzaken.
LT13	Bij het zoeken naar ContactData zonder gevonden resultaten moet een lege collectie geleverd worden.

ContactProvider

Code testgeval	Beschrijving testgeval
LT14	Bij het zoeken op naam naar ContactData moet de correcte ContactData geleverd worden.
LT15	Bij het zoeken op nummer naar ContactData moet de correcte ContactData geleverd worden.
LT16	Bij het zoeken op SIP adres naar ContactData moet de correcte ContactData geleverd worden.

ReplicationScheduler

Code testgeval	Beschrijving testgeval
LT17	Bij het opstarten van de ReplicationScheduler moet deze automatisch alle ReplicatorSettings ophalen en replicaties inplannen.
LT18	Bij het opvragen van het planningsschema moeten correcte tijden geleverd worden.
LT19	Als het tijd is om te repliceren moet een replicatie aangeroepen kunnen worden.

ReplicationLauncher

Code testgeval	Beschrijving testgeval
LT20	De CSV replicator DLL moet aangeroepen kunnen worden.
LT21	De CSV replicator moet correcte data leveren aan de ReplicationLauncher na het repliceren van de data uit een CSV bestand.
LT22	De gerepliceerde data moet correct zijn opgeslagen in de database.

ReplicatorSettingsProvider

Code testgeval	Beschrijving testgeval
LT23	Er moeten ReplicatorSettings opgevraagd kunnen worden aan de HoastAR.

ReplicationDataConverter

Code testgeval	Beschrijving testgeval
LT24	De geleverde data moet correct worden omgezet naar een collectie vol met ContactData.

ODBCReplicator

Code testgeval	Beschrijving testgeval
LT25	De ODBC replicator moet correcte data leveren aan de ReplicationLauncher na het repliceren van de data uit een databron via een ODBC koppeling.

8.7 Fysieke testgevallen

Om de fysieke testgevallen te beschrijven zal ik niet alle gemaakte unit tests naar dit document kopiëren om het aantal pagina's te beperken. Hieronder is de basis structuur van mijn unit tests te zien.

```
[TestMethod]
public void TestNaam()
{
    try
    {
        // Arrange

        // Act

        // Assert

    }
    finally
    {
        // Cleanup after test

    }
}
```

Figuur 1. Basis structuur unit test.

Als er operaties tijdens de unit test worden uitgevoerd die handmatige opschoon werkzaamheden vereisen wordt de bovenstaande structuur gebruikt. Denk hierbij aan unit tests die tabellen aanmaken in de database die na de test weer verwijderd moeten worden. In het try blok wordt de daadwerkelijke test uitgevoerd en het finally blok wordt na de test uitgevoerd.

In de meeste gevallen is er geen finally blok nodig om opschoon werkzaamheden te verrichten, dan wordt alleen de structuur gebruikt binnen het try blok.

Onder Arrange versta ik het initialiseren van klassen en het aanmaken van objecten die gebruikt zullen worden in de unit test om bepaalde functionaliteiten na te bootsen.
Onder Act versta ik het uitvoeren van de test door middel van het aanroepen van specifieke methodes.
Onder het Assert gedeelte wordt het resultaat onder het Act gedeelte van de test vergeleken met het verwachtte resultaat en dit bepaalt of de test is geslaagd of niet.

Een simpel voorbeeld van een gemaakte unit test is de GenerateTableNameNotNullOrEmpty test methode die het logische testgeval LT01 implementeert.

Code testgeval	Beschrijving testgeval
LT01	De gegenereerde tabelnaam mag niet een lege String zijn of een NULL waarde zijn.

Hieronder is de unit test te zien.

```
[TestMethod]
public void GenerateTableNameNotNullOrEmpty()
{
    // Arrange
    string actualResult;
    bool resultBool;
    DatabaseManager dbm = new DatabaseManager(_logger);

    // Act
    actualResult = dbm.GenerateTableName();
    resultBool = isNullOrEmpty(actualResult);

    // Assert
    Assert.IsFalse(resultBool);
}
```

Figuur 2. GenerateTableNameNotNullOrEmpty unit test.

Hier is goed te zien dat ik eerst een DatabaseManager klasse initialiseer onder Arrange en de klasse onder Act gebruik om de te testen methode aan te roepen.
Onder Assert vergelijk ik het resultaat uit de methode.

8.8 Literatuurlijst

1. de Grood D-J. Aanpak: Testsoorten. In: de Grood D-J. *TestGoal. Leerboek resultaatgedreven software testen.* 1e druk. Den Haag: Sdu Uitgevers bv; 2008: 49-51.

2. de Grood D-J. Testplan. In: de Grood D-J. *TestGoal. Leerboek resultaatgedreven software testen.* 1e druk. Den Haag: Sdu Uitgevers bv; 2008: 123-143.

3. de Grood D-J. Logisch testontwerp en testontwerptechnieken. In: de Grood D-J. *TestGoal. Leerboek resultaatgedreven software testen.* 1e druk. Den Haag: Sdu Uitgevers bv; 2008: 153-199.

4. de Grood D-J. Het fysieke testontwerp. In: de Grood D-J. *TestGoal. Leerboek resultaatgedreven software testen.* 1e druk. Den Haag: Sdu Uitgevers bv; 2008: 201-207.

9 Detailtestplan Systeemintegratietest

9.1 Samenvatting

In dit testplan staat beschreven wat een Systeemintegratietest inhoudt voor dit project in hoofdstuk 1. In hoofdstuk 4 en 5 staan de logische en fysieke testgevallen beschreven. Om dit detailtestplan samen te stellen zijn relevante hoofdstukken gebruikt uit het boek TestGoal.[1,2,3,4]

9.2 Inhoudsopgave

9.3 Teststrategie

Het doel van een systeem integratie test, ook wel ketentest genoemd, is het testen van het functioneren van een systeem in combinatie met andere systemen. Bij een ketentest wordt de samenwerking van het systeem met de aanliggende systemen getest. In de ketentest worden deze systemen vaak voor het eerst aan elkaar gekoppeld. De ketentest richt zich op het vinden van de fouten die ontstaan als systemen niet goed samenwerken.

Om zeker te zijn dat de DataRepository goed geïntegreerd wordt in de HoastAR en data via HoastAR aan de ToastAR geleverd kan worden wordt er een systeem integratie test uitgevoerd.

9.4 Producten

De door het testtraject op te leveren producten zijn de logische testgevallen en de fysieke testgevallen. Zie hoofdstuk 4 en 5 voor deze testgevallen.

9.5 Benodigdheden

Om de systeemintegratietest uit te voeren is de ToastAR client en de HoastAR server nodig met daarin de geïntegreerde Data Repository module.

9.6 Logische testgevallen

Voor het uitvoeren van de systeemintegratietest zijn een aantal logische testgevallen opgesteld met daarin wat er getest moet worden.

Code testgeval	Beschrijving testgeval
LT01	De ToastAR client moet kunnen zoeken naar contacten, de contact data moet vanuit de lokale database in de Data Repository geleverd worden aan de hand van de zoektermen.
LT02	Er moet een replicatie ingepland kunnen worden via HoastAR en uitgevoerd worden.
LT03	Er moet via HoastAR de ingeplande replicaties opgevraagd kunnen worden.
LT04	Replicaties moeten gedaan kunnen worden voor CSV bestanden.
LT05	Replicaties moeten gedaan kunnen worden voor data bronnen bereikbaar via een ODBC koppeling.

9.7 Fysieke testgevallen

Voor het uitvoeren van de systeemintegratietest zijn een aantal fysieke testgevallen opgesteld met daarin hoe de logische testgevallen getest moeten worden.

Code testgeval	Beschrijving testgeval
FT01	De lokale database van de Data Repository zal eerst gevuld worden met ContactData. Hierna moeten de ToastAR en HoastAR opgestart worden. Ten slotte moet er in de ToastAR client gezocht worden naar ContactData die is toegevoegd aan de lokale database.
FT02	Er bestaat nog geen user interface voor dus zal dit in HoastAR via code aangeroepen moeten worden via een interface van de Data Repository. Nadat de replicatie voltooid zou moeten zijn moet er in de lokale database gekeken worden of de data daadwerkelijk gerepliceerd en opgeslagen is.
FT03	Dit zal ook via code in de HoastAR aangeroepen moeten worden sinds hier ook geen user interface voor bestaat op dit moment.
FT04	Er moet via HoastAR eerst een replicatie ingepland worden van het type CSV. Nadat de replicatie voltooid zou moeten zijn moet er in de lokale database gekeken worden of de data daadwerkelijk gerepliceerd en opgeslagen is.
FT05	Er moet via HoastAR eerst een replicatie ingepland worden van het type ODBC. Nadat de replicatie voltooid zou moeten zijn moet er in de lokale database gekeken worden of de data daadwerkelijk gerepliceerd en opgeslagen is.

10 Systeemtest

Bij het maken van de applicatie heb ik een systeemtest uitgevoerd als ik een grote of belangrijke functionaliteit gemaakt had. Dit heb ik gedaan als vangnet om fouten op te sporen die niet opgevangen zijn door de gemaakte unit tests.

Bij systeemtesten wordt in principe een heel systeem getest. De systeemtest is een black-box test. Testen gebeurt vaak aan de hand van de reeds ontwikkelde systeeminterfaces. De systeemtest toont aan dat het systeem werkt conform het functionele ontwerp. De systeemtest is vaak het domein van de bouwende partij en wordt uitgevoerd voordat hij het systeem oplevert aan de acceptant.[1]

In sprint 1 heb ik een systeemtest uitgevoerd om de functionaliteiten van de lokale opslag te testen. Ik heb hierbij methodes uit de DatabaseManager aangeroepen vanuit de main methode van de console applicatie. Ik heb hierna het database bestand benaderd met een SQLite browser programma om te zien of de data daadwerkelijk werd opgeslagen in de database.

In sprint 2 heb ik een systeemtest uitgevoerd om de functionaliteiten te testen van het replicatie planningssysteem. Ik heb dit gedaan door in de timer callback functie die wordt aangeroepen als een timer verloopt, een regel te schrijven naar het output scherm in de debugger. Daarnaast heb ik via de main methode van de console applicatie het replicatieschema opgevraagd en deze data ook naar het output scherm weg te schrijven in de debugger.

In sprint 3 heb ik een systeemtest uitgevoerd om de replicatie functionaliteiten te testen. Ik heb dit getest door via de main methode in de console applicatie het replicatieproces aan te roepen en via een SQLite browser programma het database bestand benaderd om te zien of de gerepliceerde data daadwerkelijk werd opgeslagen en op een correcte manier.

In sprint 4 heb ik een systeemtest uitgevoerd om de replicatie functionaliteiten te testen van de ODBC replicator. Ik heb dit getest door via de main methode in de console applicatie het replicatieproces van het type ODBC aan te roepen en via een SQLite browser programma het database bestand benaderd om te zien of de gerepliceerde data daadwerkelijk werd opgeslagen en op een correcte manier.

10.1 Literatuurlijst

1. de Grood D-J. Aanpak: Testsoorten. In: de Grood D-J. *TestGoal. Leerboek resultaatgedreven software testen.* 1e druk. Den Haag: Sdu Uitgevers bv; 2008: 49-51.

11 Relational Representation Model

Hieronder is de tabelstructuur te zien van de lokale dataopslag van de Data Repository module van HoastAR.

Onderstreepte attributen zijn de Primary Keys van de tabellen.
Dikgedrukte attributen zijn attributen die geen NULL waarde mogen bevatten.

Tabel	Attributen
Contact*	(**Id**, Zip, City, Company, Custom0, Custom1, Custom2, Custom3, Custom4, Custom5, Custom6, Custom7, Custom8, Custom9, DisplayName, FirstName, LastName, **SipAddress**, Street, Uri, LocalContact, ExtensionUrl, ContactSource)
TableInformation	(**DC**, **CurrentTable**)

**Iedere bron waarvan data wordt gerepliceerd heeft een eigen tabel met contacten. De tabelnaam van deze tabel wordt gegenereerd door de Data Repository. Deze tabellen hebben dus een random naam om naam conflicten te vermijden met tabelnamen in de database.*

Zoals hierboven is te lezen, heeft iedere replicatie bron een eigen Contact tabel met attributen in de structuur van de klasse ContactData, om te voldoen aan requirement FR4. In de TableInformation tabel wordt bijgehouden wat de huidige tabelnaam is voor een specifieke databron.

12 Evaluatieformulier

DE HAAGSE HOGESCHOOL

Faculteit IT & Design

Delft

Den Haag

Evaluatieformulier afstuderen

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

Student: Glenn van Leeuwen

Periode: 2016-2.1

Bedrijf c.q. instelling: Ask Roger!

Bedrijfsmentor: Frank Heemskerk

Plaats: Delft

Datum: 21-12-2016

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

   Ja. Had snel een goed beeld van de opdracht.

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)?**

Glenn is een rustige jongen en zeker in het begin was hij wat terughoudend. Contact met collega's verliep goed. Terugkoppeling werd goed toegepast, schriftelijke rapportages worden binnen het bedrijf niet echt gebruikt.

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

- **Qua verantwoordelijkheid** goed / voldoende / matig / onvoldoende
- **Qua zelfstandigheid** goed / voldoende / matig / onvoldoende
- **Qua planmatig werken** goed / voldoende / matig / onvoldoende
- **Qua creativiteit** goed / voldoende / matig / onvoldoende
- **Qua productiviteit** goed / voldoende / matig / onvoldoende
- **Qua samenwerken met collega's** goed / voldoende / matig / onvoldoende
- **Qua draagvlakontwikkeling** goed / voldoende / matig / onvoldoende / nvt
- **Qua inspelen op bedrijfscultuur** goed / voldoende / matig / onvoldoende
- **Qua rekening houden met de specifieke context van het bedrijf** goed / voldoende / matig / onvoldoende

- **Qua het op gang brengen van de nodige veranderingen** goed / voldoende / matig / onvoldoende / nvt

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

   Goed. De meeste zaken die naar voren kwamen hebben te maken met ervaring.

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

   Op zich goed, de overdracht van het eindproduct is nog niet helemaal aan bod gekomen door drukte aan de kant van Ask Roger! en de nadruk op het rapport van Glenn. Ook is er geen tijd geweest voor de configuratie kant van het onderdeel. Hierdoor is het product nog niet geïntegreerd in het grotere geheel. We zijn dit wel van plan te doen.

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

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

   80%. Er moet nog wat werk gedaan worden om het goed in het product te verwerken.

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

Voldoende

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

Er zal tijdens het afmaken nog wat refactoring op gedaan worden, maar we verwachten dat de onderdelen voldoen.

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

Moet verder afgemaakt worden om vervolgens onderdeel te worden van het grotere project.

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

Nee

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

   Nee

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

   Ja, de samenwerking is goed bevallen en de kennis was goed genoeg om met elkaar te kunnen praten. Wel zou, wat ons betreft, de opleiding duidelijker mogen aangeven wat de verwachting is. Naar de student, danwel naar ons. De terugkoppeling met de

opleidingsbegeleider was nu dusdanig dat Glenn de laatste 2 weken volledig aan zijn verslag heeft moeten werken. Dit zouden wij liever wat verspreid zien. Juist in de laatste weken is samenwerking en overdracht belangrijker omdat in de eerste periode ook een hoop vooronderzoek zit.