Bijlagen

Het ontwikkelen van een geautomatiseerde testapplicatie

Bijlage A	Afstudeerplan
Bijlage B	Plan van aanpak
Bijlage C	Requirementsrapport
Bijlage D	Test cases document
Bijlage E	Architectuurrapport
Bijlage F	Testrapport
Bijlage G	Handleiding

Project:	Het ontwikkelen van een geautomatiseerde testapplicatie
Auteur:	Floris Otto 11073551
Bedrijf:	Crave Interactive B.V.
Opdrachtgever:	Dhr. G. van der Kruijk
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg
Tweede examinator:	Dhr. P.R. Smit
Datum:	3 June 2014

Bijlage A: Afstudeerplan

Het ontwikkelen van een geautomatiseerde testapplicatie

Project:	Het ontwikkelen van een geautomatiseerde testapplicatie	
Bijlage:	Afstudeerplan	
Auteur:	Floris Otto	11073551
Bedrijf:	Crave Interactive B.V.	
Opdrachtgever:	Dhr. G. van der Kruijk	
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg	
Tweede examinator:	Dhr. P.R. Smit	
Datum:	10 februari 2014	
Versie:	003	

Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2014-1.1 (start uiterlijk 10 februari 2014)
Startdatum uitvoering afstudeeropdracht: 10 februari 2014
Inleverdatum afstudeerdossier volgens jaarrooster: 6 juni 2014

Studentnummer:	11073551
Achternaam:	dhr Otto
Voorletters:	F.F.C.
Roepnaam:	Floris
Adres:	Van den Binckhorststraat 110
Postcode:	2691 SE
Woonplaats:	‘s-Gravenzande
Telefoonnummer:	06-29017235
Mobiel nummer:	06-29017235
Privé emailadres:	florisotto@hotmail.com
Opleiding:	Informatica
Locatie:	Den Haag
Variant:	Deeltijd
Naam studieloopbaanbegeleider:	De heer F.J.P. Teule
Naam begeleidend examinator:	Mevr. A.M.J.J. Lousberg
Naam tweede examinator:	De heer P.R. Smit
Naam bedrijf:	Crave Interactive B.V.
Afdeling bedrijf:	Software ontwikkeling
Bezoekadres bedrijf:	Warmoezenierstraat 5
Postcode bezoekadres:	2671 ZP
Postbusnummer:	-
Postcode postbusnummer:	-
Plaats:	Naaldwijk
Telefoon bedrijf:	0174-614014
Telefax bedrijf:	-
Internetsite bedrijf:	http://www.crave-hospitality.com/
Achternaam opdrachtgever:	dhr van der Kruijk
Voorletters opdrachtgever:	G.
Titulatuur opdrachtgever:	MScBA
Functie opdrachtgever:	Directeur software ontwikkeling
Doorkiesnummer opdrachtgever:	-
Email opdrachtgever:	gabriel.vanderkruijk@crave-emenu.com
Achternaam bedrijfsmentor:	dhr van der Kruijk
Voorletters bedrijfsmentor:	G.
Titulatuur bedrijfsmentor:	MScBA
Functie bedrijfsmentor:	Directeur software ontwikkeling
Doorkiesnummer bedrijfsmentor:	-
Email bedrijfsmentor:	gabriel.vanderkruijk@crave-emenu.com
Doorkiesnummer afstudeerder:	-
Functie afstudeerder (deeltijd/duaal):	Software ontwikkelaar

Titel afstudeeropdracht:

Het ontwikkelen van een geautomatiseerde testapplicatie bij Crave Interactive.

Opdrachtomschrijving

1. Bedrijf

Crave Interactive B.V.
Crave Interactive is een internationaal bedrijf dat innovatieve oplossingen biedt voor de bediening in de horeca. Door gebruik te maken van E-menu's kunnen klanten bijvoorbeeld vanaf hun tafel bestellingen plaatsen. Naast restaurants is Crave Interactive vooral gericht op hotels waar vanuit kamers roomservice besteld kan worden.

Het bedrijf is opgesplitst in twee onderdelen. Het management en het testteam bevindt zich in Milton Keynes, Engeland en het ontwikkelteam in Naaldwijk, Nederland. Door goed en vaak te communiceren tussen test- en ontwikkelteam tracht Crave Interactive een kwalitatief hoog en gebruiksvriendelijk product te ontwikkelen. Het complete team in Engeland bestaat uit ongeveer 10 personen, waarvan 4 mensen constant bezig zijn met het testen van de software. Het ontwikkelteam in Nederland bestaat uit uit 5 personen.

Binnen het bedrijf ben ik werkzaam als software ontwikkelaar. Dagelijks ben ik bezig met het implementeren van nieuwe functionaliteit en het oplossen van fouten in de programmatuur. Er wordt voornamelijk geprogrammeerd in C# .NET en Java.

2. Probleemstelling

Softwareontwikkelmethodiek
Om de ontwikkeling gestructureerd te laten verlopen, wordt er gebruik gemaakt van de softwareontwikkelmethodiek Scrum. Dit betekent dat er met behulp van timeboxing periodes van ongeveer 4 weken worden ingedeeld; zogeheten sprints. Voorafgaand aan een sprint wordt een lijst met nieuwe functionaliteiten samengesteld wat gedurende de sprint geïmplementeerd dient te worden. Na elke sprint wordt een werkend stuk software opgeleverd aan het testteam in Engeland waarna de kwaliteit van de software getest wordt.

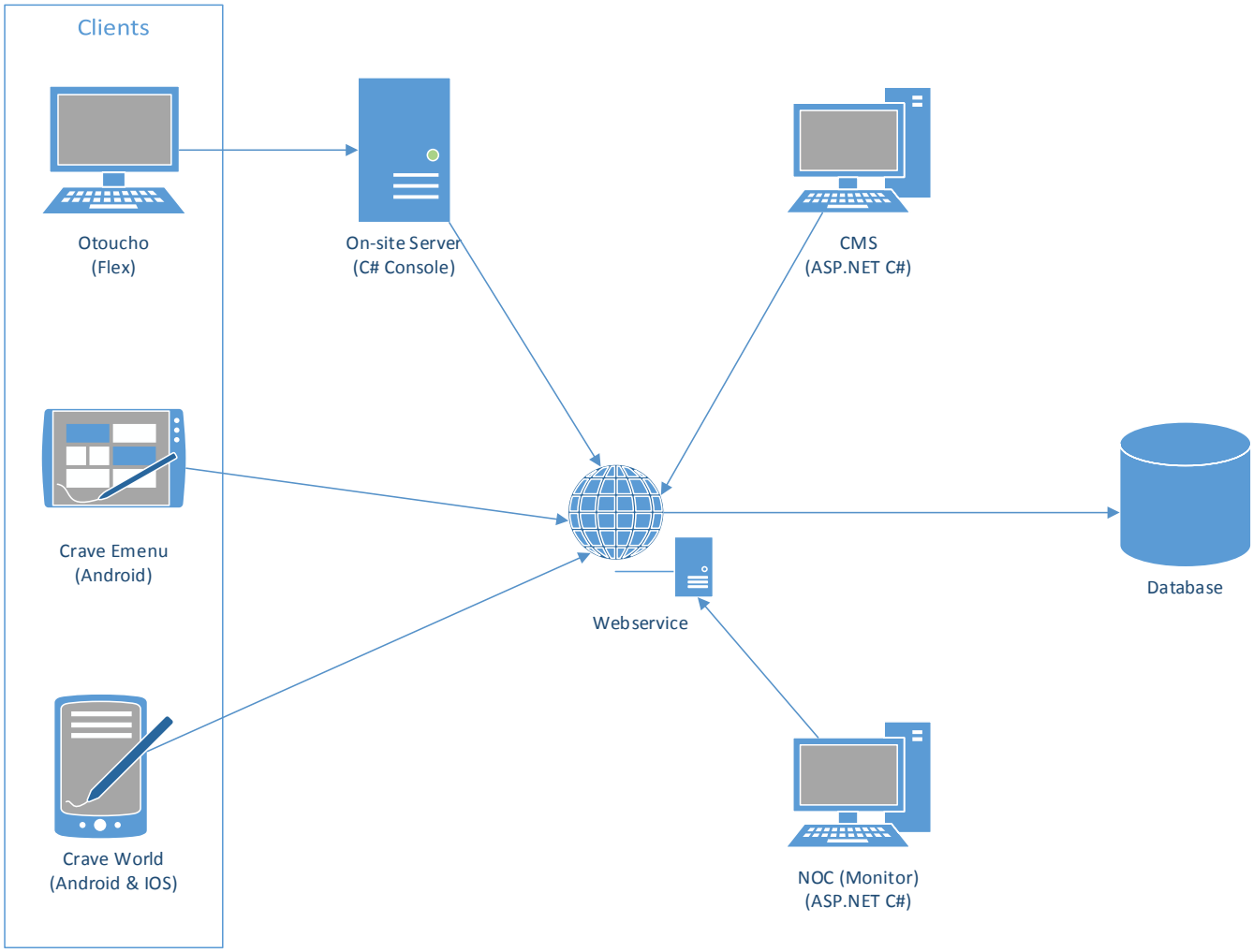
OTAP
Na elke sprint wordt de structuur van de software op een bepaald moment bevroren. Het gaat hier zowel om de structuur van de database, als de structuur van de software. Deze versie wordt vervolgens gekopieerd van de ontwikkelomgeving naar de testomgeving. Nu is het de beurt aan het testteam uit Engeland om te testen of de software van genoeg kwaliteit is. Gevonden fouten worden vervolgens door de programmeurs opgelost in de branch.

Als ten slotte de gevonden fouten opgelost zijn, en er genoeg vertrouwen is in de geteste versie, wordt de versie van de testomgeving overgebracht naar de productieomgeving.

De architectuur
Server
Aan de server kant wordt gebruik gemaakt van een webservice die communiceert met de database. De webservice is geschreven in C# en bevat ook de business logica van het bedrijf. De webservice is publiekelijk te benaderen via het internet. In de webservice bevindt zich een aantal methoden die gebruikt worden om zowel data op te vragen uit de database als data te versturen naar de database.

Clients

De clients bestaan op het moment uit een aantal verschillende devices. Hieronder vallen Samsung tablets, Otoucho schermen (kleine pc's) en sinds kort ook mobiele devices. De devices communiceren of direct, of via een On-site server met de database. Via de methodes wordt bijvoorbeeld een menu met producten opgehaald, of worden bestellingen geplaatst.



Figuur 1: De huidige architectuur van Crave Interactive B.V.

Het probleem

Na elk sprint wordt een nieuwe versie gemaakt van de structuur waarna het testteam begint met testen. Het testen van het systeem wordt op dit moment handmatig gedaan met behulp van test management software. Hierin kunnen test cases aangemaakt worden. Een test case beschrijft welke stappen doorlopen moeten worden in het programma zodat objectief gevalideerd kan worden of er nog steeds aan het verwachte resultaat voldaan wordt. De stappen in de test cases moeten hierbij dus handmatig doorlopen worden. Voor elke nieuwe versie moeten alle test cases succesvol getest zijn alvorens de versie live uit wordt gebracht. Dit neemt veel tijd in beslag.

Bij elke release wordt een nieuwe versie van de webservice opgeleverd, waar nieuwe functionaliteit in kan zitten. Er moet echter wel rekening worden gehouden met backwards compatibiliteit. De clients die op dat moment live draaien, moeten immers nog steeds zonder problemen kunnen functioneren als de webservice ge-update wordt.

Een groot deel van de test cases zijn gericht op het functionele deel van het systeem en anderen vooral op het grafische deel. Een goed voorbeeld van een functionele test is bijvoorbeeld het controleren of voor elke tafel / kamer een bestelling geplaatst kan worden in een hotel of restaurant. Handmatig neemt dit veel tijd in beslag. Een grafische test is bijvoorbeeld het controleren of de browser op de tablet een website naar behoren schaalt. De resultaten van een grafische test zijn niet gemakkelijk automatisch te valideren.

Nu de ontwikkeling van de mobiele applicatie bijna afgerond is, moet er rekening gehouden worden met een toename in dataverkeer. Als het gebruik van de applicatie aanslaat bij de consument kan dit leiden tot een grotere belasting van de webservice. Er worden op dit moment nog geen stress testen uitgevoerd. Het is daarom dus ook niet duidelijk wat voor belasting de webservice precies aan kan.

Als er op dit moment een fout gevonden wordt, moet er vaak een device geconfigureerd worden voor een bepaald bedrijf om de situatie na te bootsen. De webservice houdt er rekening mee wat voor device een aanvraag doet en past de business logica hier op aan. Een device moet zich authentiseren in elke webservice methode alvorens de methode wordt uitgevoerd. Als een device opstart, wordt een device aangemaakt in de database op basis van zijn MAC-adres en het type device. Bij het uitvoeren van methoden wordt het MAC-adres van het device meegegeven zodat bepaald kan worden om wat voor type device het gaat. Het handmatig configureren van een device om de fout te kunnen reproduceren, neemt veel tijd in beslag. In veel gevallen blijkt achteraf dat de fout zich niet in de client applicatie bevond, maar in de webservice.

3. Doelstelling van de afstudeeropdracht

1. Het besparen van tijd door het testen van de webservice methoden te automatiseren.

De afstudeeropdracht zorgt ervoor dat het testen van de webservice methoden sneller verloopt, zonder dat hierbij het kwaliteit van het testen achteruit gaat. Door gebruik te maken van een geautomatiseerde testapplicatie wordt er tijd bespaard. Er zijn twee belangrijke uitgangspunten waar rekening mee gehouden moet worden.

- Ontwikkelaars moeten snel en eenvoudig test cases op kunnen stellen.

De ontwikkelaars zullen de test cases gaan schrijven. Bij het maken van de testapplicatie zal een deel van deze test cases al geïmplementeerd worden. De test cases zullen geschreven worden in C#. Hier wordt namelijk dagelijks al mee gewerkt. Zo hoeven de ontwikkelaars geen nieuwe programmeertaal te leren of uit te zoeken hoe een bestaande tool gebruikt moet worden. Hier wordt veel tijd mee bespaard. Eventuele nieuwe test cases zullen alleen wel handmatig toegevoegd moeten worden, waarna een nieuwe versie van de testapplicatie uitgebracht zal worden.

- Gebruikers moeten de testapplicatie eenvoudig kunnen gebruiken, zonder achterliggende technische kennis.

De testapplicatie zal vervolgens door zowel het testteam als de ontwikkelaars gebruikt worden. Doordat de test cases vooraf geprogrammeerd zijn, hoeft de gebruiker geen verstand te hebben van de programmatuur, of van het inrichten van de test cases. Dit betekent een forse besparing op de inwerktijd voor de gebruikers.

2. Het besparen van tijd door het testen van device-gerelateerde logica in de webservice.

Het is tijdsintensief werk om een client te configureren. Vaak wordt na het configureren pas duidelijk dat een fout niet in de programmatuur van de client zit, maar in de business logica van de webservice.

3. Het bepalen van de stabiliteit van de webservice.

Zodra het aantal clients toeneemt, zal het dataverkeer van en naar de webservice ook toenemen. Het is belangrijk om periodiek te controleren of de webservice deze toename wel aan kan. Hierdoor kunnen problemen met de webservice preventief verholpen worden.

4. Resultaat

1. Het geautomatiseerd testen van de webservice methoden

In de testapplicatie kan een aantal test cases geselecteerd worden die uitgevoerd moeten worden. In deze test cases worden verschillende webservice methoden aangeroepen om te valideren of alles nog naar behoren werkt. Voor test cases die data versturen naar de webservice moet ook gevalideerd worden of de data ook daadwerkelijk in de database terecht is gekomen. Indien er fouten optreden, wordt dit gerapporteerd aan de gebruiker. Een ontwikkelaar kan vervolgens de gevonden fouten alsnog oplossen. Door de testapplicatie direct na een sprint te laten testen, kunnen fouten nog voor een update opgelost worden. Zo wordt een groot deel van de fouten preventief opgelost, waardoor er minder terugkoppeling nodig is vanuit het testteam in Engeland.

Een deel van de test cases zal bij het maken van de testapplicatie geïmplementeerd worden. Deze test cases zullen betrekking hebben tot het bestellen van producten. Dat is namelijk het belangrijkste gedeelte van de webservice. Deze test cases zullen aantonen dat het concept werkt. Indien er aan het eind tijd beschikbaar is, kan de testapplicatie eventueel aangevuld worden met extra test cases.

Als er in de toekomst nieuwe test cases worden toegevoegd, schrijft een ontwikkelaar deze test case in de programmatuur van de test applicatie. Zodra de test case werkt en getest is, zal er een nieuwe versie van de testapplicatie worden gemaakt. De testapplicatie wordt vervolgens beschikbaar gesteld aan zowel het testteam, als het ontwikkelteam. Ontwikkelaars kunnen zo de test cases schrijven in de programmeertaal waar zij al jaren mee bekend zijn.

2. Het testen van device-gerelateerde logica in de webservice

De testapplicatie zal ieder device na kunnen bootsen zodat elke aanvraag snel en gemakkelijk gemaakt kan worden. Het type device zal namelijk instelbaar zijn per aan te roepen test case, of voor alle test cases. Bij het authentiseren van de client, zal het type device meegegeven worden. Zo wordt de testapplicatie registreert als een ander type device. Dit zal de ontwikkelaars in de toekomst een hoop tijd besparen.

3. Het bepalen van de stabiliteit van de webservice

De testapplicatie zal stress testen uit kunnen voeren op de webservice. Met behulp van de testapplicatie kan ingesteld worden welke test cases uitgevoerd moeten worden en hoe vaak dit achter elkaar moet gebeuren. Er zal ook een mogelijkheid zijn waarbij het aantal automatisch opgehoogd zal worden. Zo kan bepaald worden op welk punt de webservice bezwijkt onder het aantal aanvragen. Potentiële problemen kunnen zo preventief opgelost worden. Zo kan er vroegtijdig worden besloten om een extra server toe te voegen.

Door testapplicaties met elkaar te verbinden, zullen test cases tegelijkertijd vanaf verschillende PC’s uitgevoerd kunnen worden. Op deze manier wordt de productomgeving nagebootst. In de productieomgeving zijn er ook meerdere devices die communiceren met de webservice.

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

Scrum
Voor de ontwikkeling van de testapplicatie zal gebruik worden gemaakt van de softwareontwikkelmethodiek Scrum. Voorafgaand aan de ontwikkeling zal een product backlog opgesteld worden waarin de functionaliteiten van de te bouwen testapplicatie worden opgedeeld. Met behulp van het MoSCoW principe zal per functionaliteit de prioriteit bepaald worden in samenspraak met de werkgever. Op basis van de product backlog zal voor elke sprint een sprint backlog opgesteld worden. De functionaliteiten met de hoogste prioriteit zullen als eerste geïmplementeerd worden, waarna vervolgens minder belangrijke punten opgepakt kunnen worden. Zo wordt voorkomen dat er aan het eind van het project een niet werkende applicatie wordt opgeleverd. Daarnaast kan door het gebruik van Scrum worden ingespeeld op veranderingen in de requirements.

- Plan van aanpak schrijven (4 – 6 dagen)
- Product backlog opstellen (2 – 4 dagen)
- Sprint backlog opstellen (Periodiek 1 dag)
- Vaststellen van lijst met test cases voor prototype (2 - 4)
- Ontwerpen van de architectuur (5 – 10 dagen)
- Ontwerpen van de testapplicatie (5 – 10 dagen)
- Ontwikkelen van de testapplicatie (25 – 35 dagen)
- Testen van de testapplicatie (5 – 10 dagen)
- Opstellen handleiding (2 – 4 dagen)
- Overdracht aan ontwikkelteam (1 – 2 dagen)

6. Op te leveren (tussen)producten

- Plan van aanpak
- Rapport met test cases
- Testapplicatie
- Testrapport
- Architectuurrapport
- Handleiding
- Afstudeerdossier

7. Te demonstreren competenties en wijze waarop

3.1 Ontwerpen softwarearchitectuur (niveau 3)

De testapplicatie zal test cases kunnen distribueren naar testapplicaties op andere PC’s, waarna zij deze gezamenlijk uit kunnen voeren. Op deze manier wordt een simulatie van de productieomgeving gemaakt, en kan er bepaald worden hoeveel dataverkeer de webservice aan kan. Het is belangrijk dat de communicatie tussen deze testapplicaties goed wordt ontworpen.

3.2 Ontwerpen systeemdeel (niveau 4)

De testapplicatie zal vanaf de grond af opgebouwd moeten worden. Hierbij moet een structuur ontworpen worden die in de toekomst makkelijk onderhouden kan worden. De kans is namelijk groot dat de testapplicatie in de toekomst zal worden gebreid met nieuwe test cases. Er zal dan ook zeker gebruik worden gemaakt van design patterns. Ik zal zelf de applicatie gaan ontwerpen aan de hand van opgedane werkervaring en kennis die vanuit school meegegeven is.

3.3 Bouwen applicatie (niveau 3)

Na het ontwerpen van de applicatie moet de applicatie gebouwd worden. Vooral de achterliggende code wordt belangrijk. Het functionele deel van de testapplicatie moet vooral in orde zijn zodat er geen foute testresultaten uit komen. De applicatie zal geschreven worden in de programmeertaal C#. Ook het bouwen van de applicatie zal volledig door mijzelf gedaan worden.

3.5 Uitvoeren van en rapporteren over het testproces (niveau 3)

Zodra de testapplicatie gebouwd is, moet getest worden of de uitgekomen resultaten ook kloppen. Mogelijk zal dit met behulp van unit testing gaan gebeuren. Door gebruik te maken van unit testing kan snel en geautomatiseerd vastgesteld worden of alle functionaliteit nog werkt. Eventuele fouten in het systeem moeten als resultaat op het testproces verholpen worden. Net als de andere beroepstaken zal dit zelfstandig uitgevoerd worden.

Bijlage B: Plan van Aanpak

Het ontwikkelen van een geautomatiseerde testapplicatie

Project:	Het ontwikkelen van een geautomatiseerde testapplicatie	
Bijlage:	Plan van aanpak	
Auteur:	Floris Otto	11073551
Bedrijf:	Crave Interactive B.V.	
Opdrachtgever:	Dhr. G. van der Kruijk	
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg	
Tweede examinator:	Dhr. P.R. Smit	
Datum:	10 februari 2014	
Versie:	001	

Documenthistorie

Versie	Revisiedatum	Wijzigingen
001	10-02-2014	Initiële versie

Inhoud

Verklarende woordenlijst

Woord	Betekenis
Afstudeerder	De student die de opdracht uit gaat voeren; Floris Otto.
Client	Een fysiek apparaat waar software van Crave Interactive B.V. op geïnstalleerd is.
Crave Interactive B.V.	Het bedrijf waar de afstudeeropdracht uitgevoerd wordt.
Database	Een gegevensbank voor het opslaan van data.
Device	Een fysiek apparaat.
Haagse Hogeschool	De organisatie waar de opleiding Informatica gegeven wordt aan de afstudeerder.
Product backlog	Complete lijst met functionaliteiten voor het product.
Sprint	Een afgebakende periode van +- 4 weken.
Sprint backlog	Lijst met functionaliteiten om te implementeren in een sprint.
Timeboxing	Het vooraf bepalen van de lengte van periodes.
Webservice	Een laag tussen de clients en webservice die het communiceren met de database mogelijk maakt.

1. Achtergronden

1.1. Crave Interactive B.V.

Crave Interactive is een internationaal bedrijf dat innovatieve oplossingen biedt voor de bediening in de horeca. Door gebruik te maken van E-menu's kunnen klanten bijvoorbeeld vanaf hun tafel bestellingen plaatsen. Naast restaurants is Crave Interactive vooral gericht op hotels waar vanuit kamers roomservice besteld kan worden.

Het bedrijf is opgesplitst in twee onderdelen. Het management en het testteam bevindt zich in Milton Keynes, Engeland en het ontwikkelteam in Naaldwijk, Nederland. Door goed en vaak te communiceren tussen test- en ontwikkelteam tracht Crave Interactive een kwalitatief hoog en gebruiksvriendelijk product te ontwikkelen. Het complete team in Engeland bestaat uit ongeveer 13 personen, waarvan 3 mensen constant bezig zijn met het testen van de software. Het ontwikkelteam in Nederland bestaat uit 5 personen.

1.2. De student

Na de afronding van mijn vierjarige MBO opleiding applicatieontwikkeling aan het Mondriaan college te Delft ben ik, Floris Otto, begonnen aan de HBO opleiding Informatica. Om naast mijn studie tegelijkertijd ook ervaring op te doen in het bedrijfsleven, heb ik er voor gekozen om de deeltijd variant van de opleiding te volgen. Op dit moment ben ik een derdejaars student en heb ik zowel de hoofdminoren als de keuzeminoren van de opleiding afgerond, waarna ik ben gestart aan het afstudeerproces.

Binnen Crave Interactive ben ik vanaf 2011 werkzaam als software ontwikkelaar. Naast het volgen van Informatica opleiding kon ik 32 uur per week bij het bedrijf aan de slag. Tijdens mijn werk heb ik al veel ervaring op kunnen doen. Dagelijks ben ik bezig met het implementeren van nieuwe functionaliteit en het oplossen van fouten in de programmatuur. Er wordt binnen het bedrijf voornamelijk geprogrammeerd in C# .NET en Java.

2. Afstudeeropdracht

2.1. Probleemstelling

Na elk sprint wordt een nieuwe versie gemaakt van de structuur waarna het testteam begint met testen. Het testen van het systeem wordt op dit moment handmatig gedaan met behulp van test management software. Hierin kunnen test cases aangemaakt worden. Een test case beschrijft welke stappen doorlopen moeten worden in het programma zodat objectief gevalideerd kan worden of er nog steeds aan het verwachte resultaat voldaan wordt. De stappen in de test cases moeten hierbij dus handmatig doorlopen worden. Voor elke nieuwe versie moeten alle test cases succesvol getest zijn alvorens de versie in productie mag worden genomen.

Bij elke release wordt een nieuwe versie van de webservice opgeleverd, waar nieuwe functionaliteit in kan zitten. Er moet echter wel rekening worden gehouden met backwards compatibiliteit. De clients die op dat moment live draaien, moeten immers nog steeds zonder problemen kunnen functioneren als de webservice ge-update wordt.

Nu de ontwikkeling van de mobiele applicatie bijna afgerond is, moet er rekening gehouden worden met een toename in dataverkeer. Als het gebruik van de applicatie aanslaat bij de consument kan dit leiden tot een grotere belasting van de webservice. Er worden op dit moment nog geen stress testen uitgevoerd. Het is daarom dus ook niet duidelijk wat voor belasting de webservice precies aan kan.

Als er op dit moment een fout gevonden wordt, moet er vaak een device geconfigureerd worden voor een bepaald bedrijf om de situatie na te bootsen. De webservice houdt er rekening mee wat voor device een aanvraag doet en past de business logica hier op aan. Een device moet zich authentiseren in elke webservice methode alvorens de methode wordt uitgevoerd. Als een device opstart, wordt een device aangemaakt in de database op basis van zijn MAC-adres en het type device. Bij het uitvoeren van methoden wordt het MAC-adres van het device meegegeven zodat bepaald kan worden om wat voor type device het gaat. Het handmatig configureren van een device om de fout te kunnen reproduceren, neemt veel tijd in beslag. In veel gevallen blijkt achteraf dat de fout zich niet in de client applicatie bevond, maar in de webservice.

De clients bestaan op het moment uit een aantal verschillende devices. Hieronder vallen Samsung tablets, Otoucho schermen (kleine pc's) en sinds kort ook mobiele devices. De devices communiceren of direct, of via een On-site server met de database. Via de methodes wordt bijvoorbeeld een menu met producten opgehaald, of worden bestellingen geplaatst.

2.2. Opdracht

Maak een testapplicatie die de diverse problemen kan verhelpen. De testapplicatie zal drie belangrijke taken moeten verrichten die het ontwikkel- en testteam kunnen ondersteunen bij hun werkzaamheden.

1. Het geautomatiseerd testen van de webservice methoden

In de testapplicatie kan een aantal test cases geselecteerd worden die uitgevoerd moeten worden. In deze test cases worden verschillende webservice methoden aangeroepen om te valideren of alles nog naar behoren werkt. Voor test cases die data versturen naar de webservice moet ook gevalideerd worden of de data ook daadwerkelijk in de database terecht is gekomen. Indien er fouten optreden, wordt dit gerapporteerd aan de gebruiker. Een ontwikkelaar kan vervolgens de gevonden fouten alsnog oplossen. Door de testapplicatie direct na een sprint te laten testen, kunnen fouten nog voor een update opgelost worden. Zo wordt een groot deel van de fouten preventief opgelost, waardoor er minder terugkoppeling nodig is vanuit het testteam in Engeland.

Een deel van de test cases zal bij het maken van de testapplicatie geïmplementeerd moeten worden. Deze test cases zullen betrekking hebben tot het bestellen van producten. Dat is namelijk het belangrijkste gedeelte van de webservice. Deze test cases zullen aantonen dat het concept werkt. Indien er aan het eind van het project nog tijd beschikbaar is, kan de testapplicatie eventueel aangevuld worden met extra test cases.

Als er in de toekomst nieuwe test cases toegevoegd moeten worden, schrijft een ontwikkelaar deze test case in de programmatuur van de test applicatie. Zodra de test case werkt en getest is, zal er een nieuwe versie van de testapplicatie worden gemaakt. De testapplicatie wordt vervolgens beschikbaar gesteld aan zowel het testteam, als het ontwikkelteam. Ontwikkelaars kunnen zo de test cases schrijven in de programmeertaal waar zij al jaren mee bekend zijn.

2. Het testen van device-gerelateerde logica in de webservice

De testapplicatie zal ieder device na moeten kunnen bootsen zodat elke aanvraag snel en gemakkelijk gemaakt kan worden. Het type device zal namelijk instelbaar zijn per aan te roepen test case, of voor alle test cases. Bij het authentiseren van de client, zal het type device meegegeven worden. Zo wordt de testapplicatie geregistreerd als een ander type device. Dit zal de ontwikkelaars in de toekomst een hoop tijd besparen.

3. Het bepalen van de stabiliteit van de webservice

De testapplicatie zal stress testen uit moeten kunnen voeren op de webservice. Met behulp van de testapplicatie kan ingesteld worden welke test cases uitgevoerd moeten worden en hoe vaak dit achter elkaar moet gebeuren. Er zal ook een mogelijkheid zijn waarbij het aantal automatisch opgehoogd zal worden. Zo kan bepaald worden op welk punt de webservice bezwijkt onder het aantal aanvragen. Potentiële problemen kunnen zo preventief opgelost worden. Zo kan er vroegtijdig worden besloten om een extra server toe te voegen.

Door testapplicaties met elkaar te verbinden, zullen test cases tegelijkertijd vanaf verschillende PC’s uitgevoerd kunnen worden. Op deze manier wordt de productomgeving nagebootst. In de productieomgeving zijn er ook meerdere devices die communiceren met de webservice.

Naast het uitvoeren van test cases vanaf meerdere testapplicaties, zullen de testapplicaties gebruik maken van multithreading. Gezien het feit dat huidige PC’s tegenwoordig standaard voorzien van meerdere cores, kan een enkele testapplicatie gelijktijdig meerdere test cases uitvoeren. Door hier

gebruik van te maken, zijn er minder PC's nodig om een afspiegeling van de productieomgeving te repliceren.

2.3. Ontwikkelmethodiek

Voor de ontwikkeling van de testapplicatie zal gebruik worden gemaakt van de principes van de agile ontwikkelmethodiek Scrum. Dit betekent normaliter dat er met behulp van timeboxing periodes van ongeveer 4 weken worden ingedeeld; zogeheten sprints. Gezien het feit dat de doorlooptijd van het afstudeertraject niet zo lang is, zullen er voor het project sprints gedefinieerd worden van 2 weken.

Voorafgaand aan de ontwikkeling zal een product backlog opgesteld worden waarin de functionaliteiten van de te bouwen testapplicatie worden opgedeeld. Met behulp van het MoSCoW principe zal per functionaliteit de prioriteit bepaald worden in samenspraak met de opdrachtgever.

Op basis van de product backlog zal voor elke sprint een sprint backlog opgesteld worden. De functionaliteiten met de hoogste prioriteit zullen als eerste geïmplementeerd worden, waarna vervolgens minder belangrijke punten opgepakt kunnen worden. Zo wordt voorkomen dat er aan het eind van het project een niet werkende applicatie wordt opgeleverd. Daarnaast kan door het gebruik van Scrum worden ingespeeld op veranderingen in de requirements.

3. Doelstellingen

De opdracht heeft de volgende doelstellingen:

1. Het besparen van tijd door het testen van de webservice methoden te automatiseren.

De afstudeeropdracht zorgt ervoor dat het testen van de webservice methoden sneller verloopt, zonder dat hierbij het kwaliteit van het testen achteruit gaat. Door gebruik te maken van een geautomatiseerde testapplicatie wordt er tijd bespaard. Er zijn twee belangrijke uitgangspunten waar rekening mee gehouden moet worden.

- Ontwikkelaars moeten snel en eenvoudig test cases op kunnen stellen.

De ontwikkelaars zullen de test cases gaan schrijven. Bij het maken van de testapplicatie zal een deel van deze test cases al geïmplementeerd worden. De test cases zullen geschreven worden in C#. Hier wordt namelijk dagelijks al mee gewerkt. Zo hoeven de ontwikkelaars geen nieuwe programmeertaal te leren of uit te zoeken hoe een bestaande tool gebruikt moet worden. Hier wordt veel tijd mee bespaard. Eventuele nieuwe test cases zullen alleen wel handmatig toegevoegd moeten worden, waarna een nieuwe versie van de testapplicatie uitgebracht zal worden.

- Gebruikers moeten de testapplicatie eenvoudig kunnen gebruiken, zonder achterliggende technische kennis.

De testapplicatie zal vervolgens door zowel het testteam als de ontwikkelaars gebruikt worden. Doordat de test cases vooraf geprogrammeerd zijn, hoeft de gebruiker geen verstand te hebben van de programmatuur, of van het inrichten van de test cases. Dit betekent een forse besparing op de inwerktijd voor de gebruikers.

2. Het besparen van tijd door het testen van device-gerelateerde logica in de webservice.

Het is tijdsintensief werk om een client te configureren. Vaak wordt na het configureren pas duidelijk dat een fout niet in de programmatuur van de client zit, maar in de business logica van de webservice.

3. Het bepalen van de stabiliteit van de webservice.

Zodra het aantal clients toeneemt, zal het dataverkeer van en naar de webservice ook toenemen. Het is belangrijk om periodiek te controleren of de webservice deze toename wel aan kan. Hierdoor kunnen problemen met de webservice preventief verholpen worden.

4. Projectgrenzen en randvoorwaarden

4.1 Projectgrenzen

Startdatum uitvoering project:	10 februari 2014
Inleverdatum afstudeerdossier:	6 juni 2014
Project duur:	17 weken

Wat wordt opgeleverd:

- De testapplicatie bevat de volgende functionaliteiten:
    a. De testapplicatie bevat genoeg test cases om het concept van geautomatiseerd testen te kunnen valideren. De test cases zullen gericht zijn op het bestellen van producten.
    b. De testapplicatie kan gedistribueerd testen d.m.v. een master-slave architectuur.
    c. De testapplicatie maakt gebruik van multithreading.
    d. De testapplicatie kan verschillende typen devices na bootsen.
        i. Samsung Galaxy Tab (Crave Emenu)
        ii. Mobile device (Crave World)
        iii. Otoucho (Flex)

Wat wordt niet opgeleverd of is optioneel:

- De testapplicatie bevat de volgende functionaliteiten:
    a. De testapplicatie bevat test cases om alle webservice methoden te kunnen testen.

4.2 Randvoorwaarden

Vereisten aan het product vanuit opdrachtgever:

- Ontwikkelaars moeten snel en eenvoudig test cases op kunnen stellen.
- Gebruikers moeten de testapplicatie eenvoudig kunnen gebruiken, zonder achterliggende technische kennis

4.3 Risicofactoren

Bij het ontwikkelen van de testapplicatie zal een aantal test cases geïmplementeerd worden om te valideren of het concept werkt. Deze test cases zullen voor het prototype vooralsnog alleen gericht zijn op het bestellen van producten. Zonder een goede afbakening zou het aantal test cases voor het prototype kunnen groeien, waardoor de rest van het project achterstand oploopt.

Het is voor een goed verloop van het project noodzakelijk om een goede communicatielijn te onderhouden tussen opdrachtgever en afstudeerder. Uiteindelijk moet het product worden zoals de opdrachtgever voor ogen heeft. Door wekelijks af te spreken zal de opdrachtgever betrokken blijven bij elke stap van het project. Als dit niet gebeurt, loopt het project het risico om een verkeerde weg in te slaan.

4.4 Benodigde mensen / middelen

Mensen

- Floris Otto — Afstudeerder
- Dhr. G. van der Kruijk — Opdrachtgever
- Mevr. A.M.J.J. Lousberg — Begeleidend examinator
- Dhr. P.R. Smit — Tweede examinator

Middelen

- Microsoft Visual Studio 2012
- Microsoft SQL Server 2012
- TortoiseSVN
- PivotalTracker
  Gedurende het project zal er gebruik gemaakt worden van een project management tool. Met behulp van de tool kunnen er user stories aangemaakt worden. Alle punten van de product backlog kunnen hierin verwerkt worden.

5. Projectactiviteiten

Om het project gestructureerd te laten verlopen, wordt er vooraf een aantal activiteiten gedefinieerd. Deze activiteiten vormen de basis voor de planning in bijlage A.

- Plan van aanpak schrijven
- Requirements opstellen
- Product backlog opstellen
- Sprint backlog opstellen
- Vaststellen van lijst met test cases voor prototype
- Ontwerpen van de architectuur
- Ontwerpen van de testapplicatie
- Ontwikkelen van de testapplicatie
- Testen van de testapplicatie
- Opstellen handleiding
- Overdracht aan ontwikkelteam
- Afstudeerdossier opstellen

6. Op te leveren producten

Tijdens de uitvoering van de afstudeeropdracht zal een aantal producten opgeleverd worden. Deze producten zullen voortkomen uit de activiteiten van hoofdstuk 5. Onderstaande producten worden opgeleverd aan het eind van het project.

- Plan van aanpak
- Rapport met requirements
  Het requirementsrapport zal een lijst bevatten met zowel gebruikers- als systeemrequirements. Daarnaast wordt onderscheid gemaakt tussen functionele en non-functionele requirements.
- Rapport met test cases
  Er wordt een rapport opgesteld met alle test cases die uiteindelijk in de testapplicatie geïmplementeerd dienen te worden. Het implementeren van die test cases is optioneel voor het project. Als er aan het eind van het project nog voldoende tijd beschikbaar is, dan zullen de overige test cases geïmplementeerd worden.
- Testapplicatie
  Aan het eind van het project wordt een werkende testapplicatie opgeleverd aan de opdrachtgever. Hoofdstuk 4 bevat de voorwaarden aan de testapplicatie.
- Testrapport
  Zodra de testapplicatie gemaakt is, zal deze getest worden. De resultaten van de uitgevoerde tests zullen gebundeld worden tot een testrapport.
- Architectuurrapport
  De testapplicatie zal gebruik maken van een master-slave architectuur. Hoe die communicatie precies verloopt tussen de testapplicaties zal te vinden zijn in het architectuurrapport.
- Handleiding
  Voor de oplevering aan het test- en ontwikkelteam wordt een handleiding opgesteld over het gebruik van de testapplicatie.
- Afstudeerdossier

7. Kwaliteit

Om de kwaliteit van het eindproduct te waarborgen, zal er een wekelijks gesprek ingepland worden om de voortgang van het project te bespreken. In dit gesprek kan de opdrachtgever feedback geven op opgeleverde producten en is er een mogelijkheid voor de afstudeerder om vragen te stellen. Door veel te communiceren, wordt getracht een kwalitatief hoogwaardig product te ontwikkelen. Elk document zal door de opdrachtgever worden gecontroleerd alvorens het wordt doorgestuurd naar een begeleider vanuit school.

Naast de wekelijkse besprekingen, wordt er elke ochtend een stand-up meeting gehouden. Dit gebeurt met het gehele ontwikkelteam. Tijdens deze meeting wordt er verteld wat er afgelopen dag is gedaan en worden eventuele problemen besproken. Zo houdt het team elkaar scherp en worden problemen op de best mogelijke manier opgelost.

Om te valideren of de testapplicatie werkt, zal de testapplicatie getest moeten worden. Zodra er een test case uitgevoerd wordt waarbij data naar de database wordt verstuurd, valideert de testapplicatie achteraf of de data ook daadwerkelijk in de database terecht is gekomen. De uitvoer van een test case wordt zo dubbel bevestigd; eenmaal door de webservice en eenmaal door de database.

Aan de hand van de vooraf opgestelde requirements kan achteraf gevalideerd worden of de testapplicatie aan de gewenste functionaliteit voldoet. Deze requirements zullen een handvat bieden voor de tests in het testrapport. De resultaten van de uitgevoerde tests zullen ook toegevoegd worden in het testrapport.

Bijlage A: Planning

Planning

	Weken																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Activiteiten					Sprint 1		Sprint 2		Sprint 3		Sprint 4						
Plan van aanpak schrijven																	
Requirements opstellen																	
Product backlog opstellen																	
Vaststellen van lijst met test cases voor prototype																	
Ontwerpen van de architectuur																	
Ontwerpen van de testapplicatie																	
Ontwikkelen van de testapplicatie																	
Testen van de testapplicatie																	
Opstellen handleiding																	
Overdracht aan ontwikkelteam																	
Afstudeerdossier opstellen																	
Sprint backlog opstellen (Periodiek 1 dag)																	

Bijlage C: Requirementsrapport

Het ontwikkelen van een geautomatiseerde testapplicatie

Project:	Het ontwikkelen van een geautomatiseerde testapplicatie	
Bijlage:	Requirementsrapport	
Auteur:	Floris Otto	11073551
Bedrijf:	Crave Interactive B.V.	
Opdrachtgever:	Dhr. G. van der Kruijk	
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg	
Tweede examinator:	Dhr. P.R. Smit	
Datum:	17 mei 2014	
Versie:	001	

Inhoud

1. Inleiding

Aan het begin van het project zijn er requirements opgesteld. De requirements geven inzicht in de eisen en de wensen van de gebruikers ten opzichte van het systeem. De requirements zijn verzameld door verschillende gebruikers te interviewen. Zo zijn zowel de opdrachtgever als een ontwikkelaar geïnterviewd. De opdrachtgever nam hierbij de rol van tester op zich. Tijdens de interviews zijn de volgende vragen gesteld:

- Welke knelpunten in het huidige testproces zorgen op dit moment voor de meeste vertraging?
- Welke aspecten worden op dit moment nog onderbelicht in het huidige testproces?
- Welke functionaliteiten moet de testapplicatie bevatten om het huidige testproces te kunnen verbeteren?

Op basis van deze vragen is een aantal belangrijke punten naar voren gekomen. Deze punten zijn weergegeven in hoofdstuk 2.

De resultaten van de interviews resulteerden uiteindelijk in functional user requirements. Dit zijn de requirements vanuit het oogpunt van de gebruiker gezien. Zij beschrijven wat de gebruiker moet kunnen doen met het systeem. De functional user requirements zijn weergegeven in hoofdstuk 3.

De functional user requirements zijn vervolgens gebruikt om de functional system requirements op te stellen. Deze requirements richten zich meer op de functionaliteit van het systeem en worden dan ook opgesteld vanuit het oogpunt van het systeem. De requirements zijn gedetailleerder dan de function user requirements. De functional system requirements zijn weergegeven in hoofdstuk 4.

Uiteindelijk zijn er uit de interviews ook een aantal kwaliteitskenmerken naar voren gekomen waar het systeem aan zou moeten voldoen. Deze requirements vallen onder de titel non-functional requirements en zijn weergegeven in hoofdstuk 5.

In overleg met de opdrachtgever is vervolgens een product backlog samengesteld. Deze punten zijn opgesteld vanuit de functional system requirements. De product backlog is weergegeven in hoofdstuk 6.

Uiteindelijk is in hoofdstuk 7 weergegeven welke features zijn ontwikkeld per sprint. Vanuit de product backlog zijn elke sprint een tweetal features verplaatst naar de sprint backlog alvorens deze geïmplementeerd konden worden. De tabel in hoofdstuk 7 geeft weer welke features dit waren.

2. Interviewresulaten

Uit de interviews is een aantal belangrijke punten naar voren gekomen over een aantal problemen, die momenteel beter kunnen in het huidige testtraject. Deze punten staan hieronder beschreven.

1. **Geautomatiseerd testen van oude webservices**
   Actor: *Ontwikkelaar*
   Na elke release wordt een nieuwe webservice versie van de mobile webservice en de crave webservice gemaakt. Met behulp van TestRail voert het testteam handmatig test cases uit waarbij de nieuwe webservices getest worden. Hierbij wordt niet getest of de nieuwe versies er voor gezorgd hebben dat de oudere versies niet meer werken. Met het oog op backwards compatibility zouden na elke release de methoden uit de oude webservices getest moeten worden.

2. **Stresstesten van de webservice**
   Actor: *Opdrachtgever*
   Sinds een aantal weken staat de eerste versie van de mobiele applicatie in zowel Google Play, als de App Store. Als er in de toekomst ineens een forse toename in het gebruik van de mobiele applicatie plaatsvindt, is niet duidelijk of de webservices hier wel tegen bestand zijn. Er is op dit moment niet bekend wat de capaciteiten zijn van de server. Dit wordt simpelweg nog niet getest. Een toename in dataverkeer zou er voor kunnen zorgen dat het systeem traag wordt, of zelfs dat het systeem het in zijn geheel niet meer aan kan. Op dit moment wordt er niet getest wat de webservices aan kunnen wat betreft het aantal aanvragen.

   Om dezelfde situatie als de productieomgeving te simuleren is door de opdrachtgever aangedragen om multithreading te gebruiken zodat er meerdere tests asynchroon uitgevoerd kunnen worden. Zo kan een enkele testapplicatie zich voordoen als meerdere clients.

   Naast het gebruik van multithreading wordt ook geopperd om verschillende testapplicaties met elkaar te kunnen verbinden door middel van een client-server architectuur. De server kan hierbij tests delegeren aan verbonden client testapplicaties waarna zij gezamenlijk een webservice met aanvragen kunnen bestoken. De combinatie van multithreading en client-server architectuur zou ervoor kunnen zorgen dat er een realistischer test scenario opgezet kan worden.

3. **Gemakkelijk uitvoeren van bestaande tests zonder achterliggende technische kennis**
   Actor: *Opdrachtgever, Ontwikkelaar, Tester*
   Met het oog op punt 1 moeten met behulp van de testapplicatie tests uitgevoerd kunnen worden. De tests moeten gecombineerd kunnen worden tot een test run zodat er een aantal tests direct achter elkaar uitgevoerd kunnen worden. In een test run moet bijvoorbeeld getest kunnen worden of voor elke bedrijf het menu opgehaald kan worden voor een

bepaalde versie van de webservice. Deze test run moet opgeslagen kunnen worden zodat het proces in de toekomst herhaald kan worden. Een tester hoeft dan alleen maar de test run te laden en op een knop te drukken waarna het proces automatisch uitgevoerd wordt. Na de uitvoering van de test run moeten de resultaten van de uitgevoerde tests getoond kunnen worden.

Tijdens het interview kwam hierbij een tegenstrijdig belang naar boven tussen de opdrachtgever en tester. Een tester wil namelijk over een GUI beschikken om tests gemakkelijk uit te kunnen voeren, terwijl het voor de opdrachtgever in eerste instantie belangrijker is dat de testapplicatie functioneel goed werkt. Voor de diepgang van het afstudeerproject is het niet van belang dat er een GUI aanwezig is. Deze kan in een later stadium nog gemaakt worden. In overleg met de opdrachtgever is daarom besloten om er in eerste instantie voor te zorgen dat de testapplicatie functioneel werkt via een console applicatie. Deze tegenstrijdigheid komt ook terug in de prioritering van de requirements UR03 en UR15 in de tabel van hoofdstuk 3.

4. **Tests schrijven in reeds bekende programmeertaal**
   Actor: *Opdrachtgever, ontwikkelaar*
   Een belangrijke wens van de opdrachtgever en de ontwikkelaar is om ervoor te zorgen dat ontwikkelaars gemakkelijk nieuwe tests toe kunnen voegen aan de testapplicatie. Het liefst moet dit kunnen gebeuren door de test te kunnen schrijven in de programmeertaal waar zij al dagelijks mee werken. Zo wordt er geen inwerktijd verspeeld aan het leren beheersen van een nieuwe taal, of het begrijpen van een bestaande tool. Een nadeel is wel dat zodra er een nieuwe test geïmplementeerd wordt, er een nieuwe versie van de testapplicatie gemaakt zal moeten worden.

3. Functional user requirements

De volgende functional user requirements zijn tot stand gekomen naar aanleiding van de interviews die gehouden zijn met zowel de opdrachtgever als een ontwikkelaar. Deze requirements zijn vanuit het oogpunt van de gebruiker gespecifieerd.

Id	Requirement	Actor	Interview punt	Prioriteit
UR01	De gebruiker moet een device type kunnen selecteren waarmee een test case wordt uitgevoerd.	Gebruiker	1	M
UR02	De gebruiker moet een versie van de webservice kunnen selecteren waarop een test case uitgevoerd wordt.	Gebruiker	1	M
UR03	De gebruiker moet een test run uit kunnen voeren vanaf de command line.	Gebruiker	3	M
UR04	De gebruiker moet een test run met één test case uit kunnen voeren.	Gebruiker	3	M
UR05	De gebruiker moet een test run met meerdere test cases uit kunnen voeren.	Gebruiker	3	M
UR06	De gebruiker moet test cases kunnen combineren tot een test run.	Gebruiker	3	M
UR07	De gebruiker moet parameters kunnen selecteren waarmee een test case wordt uitgevoerd.	Gebruiker	3	M
UR08	De gebruiker moet een test run op kunnen slaan zodat de test run herhaald kan worden.	Gebruiker	3	M
UR09	De gebruiker moet testapplicaties kunnen verbinden binnen een netwerk.	Gebruiker	2	S
UR10	De gebruiker moet in kunnen stellen hoe vaak een test run achter elkaar uitgevoerd moet worden.	Gebruiker	2	S
UR11	De gebruiker moet kunnen selecteren of meerdere test runs naast elkaar uitgevoerd moeten worden.	Gebruiker	2	S
UR12	De gebruiker moet een verbinding tussen twee testapplicaties kunnen verbreken.	Gebruiker	2	S
UR13	De gebruiker moet een test run uit kunnen laten voeren door één of meerdere verbonden testapplicaties.	Gebruiker	2	S
UR14	De gebruiker moet de resultaten van een uitgevoerde test run kunnen bekijken.	Gebruiker	3	C
UR15	De gebruiker moet een test run uit kunnen voeren vanaf de GUI.	Gebruiker	3	C

4. Functional system requirements

De functional system requirements zijn voortgekomen uit de functional user requirements en op basis van de gehouden interviews.

Id	Requirement	Actor	User requirement	Prioriteit
SR01	Het systeem moet van webservice kunnen wisselen afhankelijk van het gekozen device type.	Systeem	UR01	M
SR02	Het systeem moet van webservice kunnen wisselen afhankelijk van de gekozen webservice versie per test case.	Systeem	UR02	M
SR03	Het systeem moet een test run uit kunnen voeren vanaf de command line.	Systeem	UR03	M
SR04	Het systeem moet een test run met één test case uit kunnen voeren.	Systeem	UR04	M
SR05	Het systeem moet een test run met meerdere test cases uit kunnen voeren.	Systeem	UR05	M
SR06	Het systeem moet test cases kunnen combineren tot een test run.	Systeem	UR06	M
SR07	Het systeem moet na uitvoer van een test case de resultaten kunnen verifiëren met data in de database.	Systeem	-	M
SR08	Het systeem moet kunnen verbinden met een server testapplicatie binnen een netwerk.	Systeem	UR09	S
SR09	Het systeem moet een test run meerdere keren achter elkaar uit kunnen voeren.	Systeem	UR10	S
SR10	Het systeem moet test runs parallel uit kunnen voeren.	Systeem	UR11	S
SR11	Het systeem moet een verbinding met een opgezette verbinding met een testapplicatie kunnen verbreken.	Systeem	UR12	S
SR12	Het systeem moet een test run uit kunnen laten voeren door één verbonden client testapplicatie.	Systeem	UR13	S
SR13	Het systeem moet beschikbare test cases weer kunnen geven afhankelijk van het gekozen device type.	Systeem	UR01	C
SR14	Het systeem moet een aantal parameter velden tonen afhankelijk van de gekozen test case.	Systeem	UR04, UR05	C
SR15	Het systeem moet beschikbare webservice versies kunnen tonen.	Systeem	UR02	C
SR16	Het systeem moet een test run met geselecteerde test cases op kunnen slaan.	Systeem	UR08	C
SR17	Het systeem moet de uitvoer van een test run loggen.	Systeem	UR14	C
SR18	Het systeem moet de resultaten van een test run kunnen tonen.	Systeem	UR14	C
SR19	Het systeem moet een test run uit kunnen voeren vanaf de GUI.	Systeem	UR15	C

5. Non-functional requirements

Tot slot zijn er een tweetal non-functional requirements opgesteld, de kwaliteitseisen van het systeem. Ook deze zijn uit de interviews naar voren gekomen.

Id	Requirement	ISO 9126 / kwaliteitskenmerk
NF01	Het systeem moet te gebruiken zijn door gebruikers zonder beschikking over achterliggende technische kennis.	Gebruiksvriendelijkheid
NF02	De uitgevoerde test cases moeten geverifieerd worden met data in de database.	Juistheid

6. Product backlog

Op basis van de requirements is in overleg met de opdrachtgever een product backlog opgesteld. De product backlog bevat features die geïmplementeerd kunnen worden in periodes van een week. Om de traceerbaarheid van de features te behouden is de referentie met de requirements toegevoegd in de laatste kolom.

Id	Functionaliteit	Prioriteit	Requirement
P01	Detach legacy models/converters from webservice project	Must have	SR01, SR02
P02	Setting up a general structure	Must have	SR01, SR02, SR03, SR19
P03	Test cases for prototype (Ordering)	Must have	SR04, SR05, SR06
P04	Implement test runs	Must have	SR03, SR17, NF01
P05	Create webservice classes automatically with the CraveBuilder	Must have	-
P06	Validate result of test cases with data in database	Must have	SR07, NF02
P07	Distributed testing	Should have	SR08, SR11, SR12
P08	Determine breaking point of webservice	Should have	SR09, SR10, SR12
P09	Multithreading	Should have	SR10
P10	Error reporting	Could have	SR18
P11	GUI	Could have	SR13, SR14, SR15, SR16, SR19
P12	All test cases	Would have	-

7. Sprint backlog

Aan het begin van elke sprint zijn features vanaf de product backlog verplaatst naar de sprint backlog. Dit gebeurde deels op basis van de prioriteit van de features, en deels in overleg met de opdrachtgever. Uiteindelijk zijn de volgende sprints doorlopen, en zijn daarin de achterstaande features verwerkt:

Sprint	Id	Functionaliteit
Sprint 1	P01	Detach legacy models/converters from webservice project
	P02	Setting up a general structure
		Release 1
Sprint 2	P03	Test cases for prototype (Ordering)
	P04	Implement test runs
		Release 2
Sprint 3	P04	Implement test runs
	P05	Create webservice classes automatically with the CraveBuilder
		Release 3
Sprint 4	P06	Validate result of test cases with data in database
	P10	Error reporting
		Release 4

Bijlage D: Test cases document

Het ontwikkelen van een geautomatiseerde testapplicatie

Project:	Het ontwikkelen van een geautomatiseerde testapplicatie	
Bijlage:	Test cases document	
Auteur:	Floris Otto	11073551
Bedrijf:	Crave Interactive B.V.	
Opdrachtgever:	Dhr. G. van der Kruijk	
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg	
Tweede examinator:	Dhr. P.R. Smit	
Datum:	14 april 2014	
Versie:	001	

1. Test cases voor afstudeerproject

Een test case wordt in het project gebruikt om een enkele webservice methoden te kunnen testen. Tijdens het project zijn een select aantal test cases geïmplementeerd in de test applicatie. Dit is in overleg met de opdrachtgever besloten om genoeg tijd over te houden voor de ontwikkeling van de rest van de testapplicatie. De test cases die in het project al geïmplementeerd zijn, hebben allen betrekking op het bestellingsgedeelte van de webservice. Deze methode zijn immers het belangrijkst voor Crave Interactive B.V.

De test cases die geïmplementeerd zijn tijdens het project:

Crave webservice

Id	Test case	Webservice methode
M01	Lijst met namen en ids van bedrijven ophalen.	GetCompanies
M02	Alle informatie van een bedrijf ophalen.	GetCompany
M03	Menu ophalen voor een kamergroep.	GetMenu
M04	Bestelpunten voor een tafelgroep ophalen.	GetDeliverypoints
M05	Een product bestellen.	SaveOrder
M06	Ophalen/aanmaken van een client voor de testapplicatie.	GetClient

Mobile webservice

Id	Test case	Webservice methode
C01	Lijst met namen en ids van bedrijven ophalen.	GetCompanyList
C02	Alle informatie van een bedrijf ophalen.	GetCompany
C03	Menu ophalen voor een kamergroep.	GetMenu
C04	Een product bestellen.	SaveOrder
C05	Ophalen van een customer voor de testapplicatie.	GetCustomer
C06	Aanmaken van een customer voor de testapplicatie.	CreateCustomer

2. Test cases voor het eindproduct

Naast de implementatie van de test cases voor het prototype is er een lijst opgesteld met alle test cases die uiteindelijk geïmplementeerd gaan worden. Hoogstwaarschijnlijk zullen deze test cases pas na het afstudeerproject verwerkt kunnen worden.

De totale set aan test cases die de test applicatie uiteindelijk zal bevatten:

Crave webservice

Id	Test case	Webservice methode
M01	Lijst met namen en ids van bedrijven ophalen.	GetCompanies
M02	Alle informatie van een bedrijf ophalen.	GetCompany
M03	Menu ophalen voor een kamergroep.	GetMenu
M04	Bestelpunten voor een tafelgroep ophalen.	GetDeliverypoints
M05	Een product bestellen.	SaveOrder
M06	Ophalen/aanmaken van een client.	GetClient
M07	Ophalen van alle advertenties voor een client.	GetAdvertisements
M08	Ophalen van geplande announcements voor een client.	GetAnnouncements
M09	Ophalen van clients van een bedrijf.	GetClients
M10	Ophalen van entertainment voor een client.	GetEntertainment
M11	Ophalen van alle berichten voor een client.	GetMessages
M12	Ophalen van verwerkte orders voor een client.	GetOrderHistory
M13	Ophalen van lopende orders voor een client.	GetOrders
M14	Ophalen van informatie over de laatste release voor een client.	GetRelease
M15	Ophalen van informatie over de laatste release voor een client en de locatie van de release.	GetReleaseAndDownloadLocation
M16	Ophalen/opslaan van de status voor een client.	GetSetClientStatus
M17	Ophalen/opslaan van de status voor een terminal.	GetSetTerminalStatus
M18	Ophalen van de socialmedia berichten voor een client.	GetSocialmediaMessages
M19	Ophalen van de enquêtes voor een client.	GetSurveys
M20	Opslaan van enquêteresultaten voor een client.	SaveSurveyResults
M21	Ophalen van beschikbare terminals voor een bedrijf.	GetTerminals
M22	Ophalen van een specifieke terminal.	GetTerminal
M23	De status van een order op ‘verwerkt’ zetten.	ProcessOrders
M24	Een bestelpunt voor een client instellen.	SetClientDeliverypoint

Mobile webservice

Id	Test case	Webservice methode
C01	Lijst met namen en ids van bedrijven ophalen.	GetCompanyList
C02	Alle informatie van een bedrijf ophalen.	GetCompany
C03	Menu ophalen voor een kamergroep.	GetMenu
C04	Een product bestellen.	SaveOrder
C05	Ophalen van een customer.	GetCustomer
C06	Aanmaken van een customer.	CreateCustomer
C07	Ophalen van de access codes voor een customer.	GetAccessCodes

C08	Ophalen van verwerkte orders voor een customer.	GetOrderHistory
C09	Ophalen van de points of interest voor een customer.	GetPointOfInterest
C10	Ophalen van een site voor een customer.	GetSite
C11	Een wachtwoord reset email opvragen voor een customer.	RequestPasswordReset
C12	Het resetten van het wachtwoord voor een customer.	ResetPassword
C13	Het updaten van gegevens voor een customer.	UpdateCustomerDetails

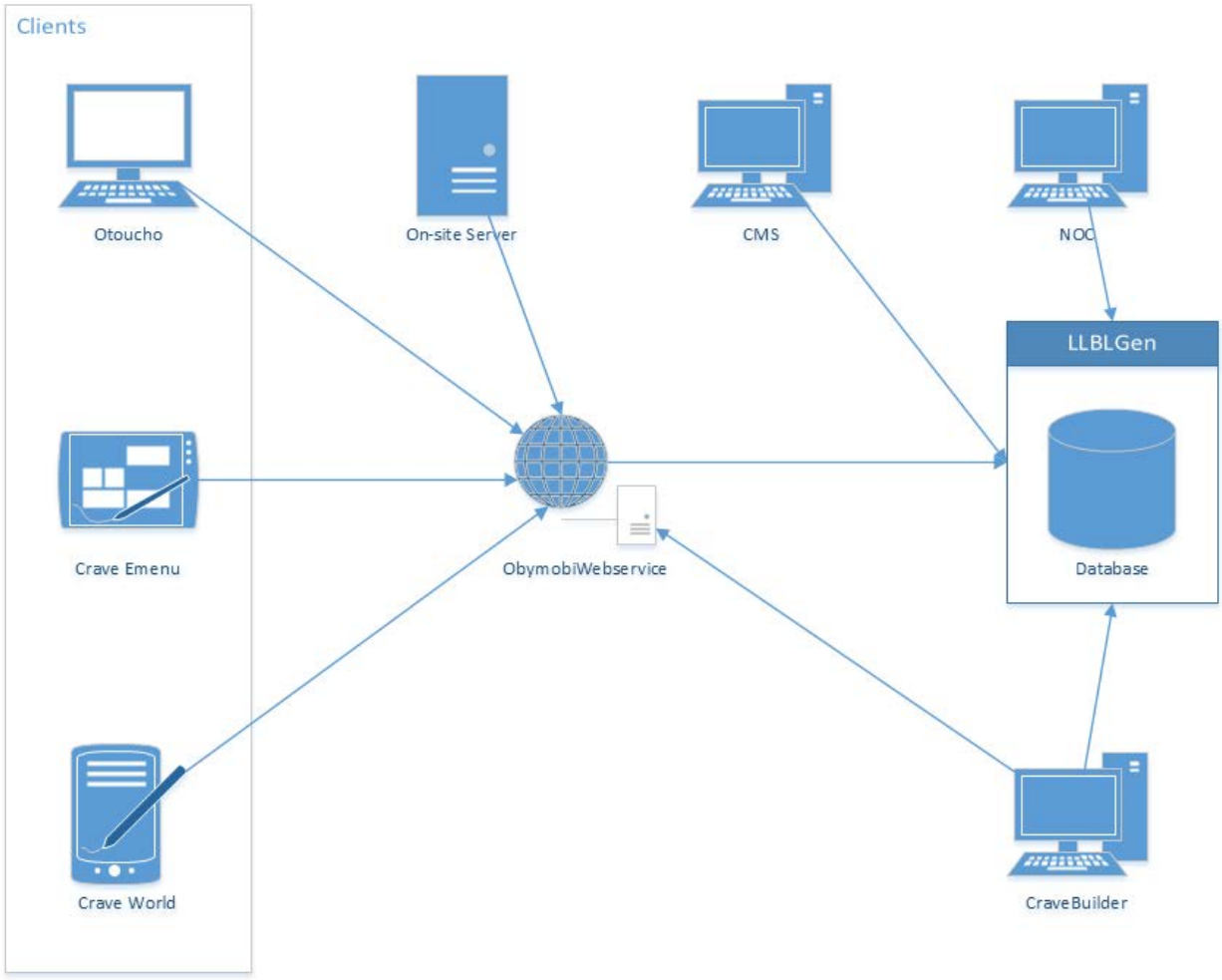
Bijlage E: Architectuurrapport

Het ontwikkelen van een geautomatiseerde testapplicatie

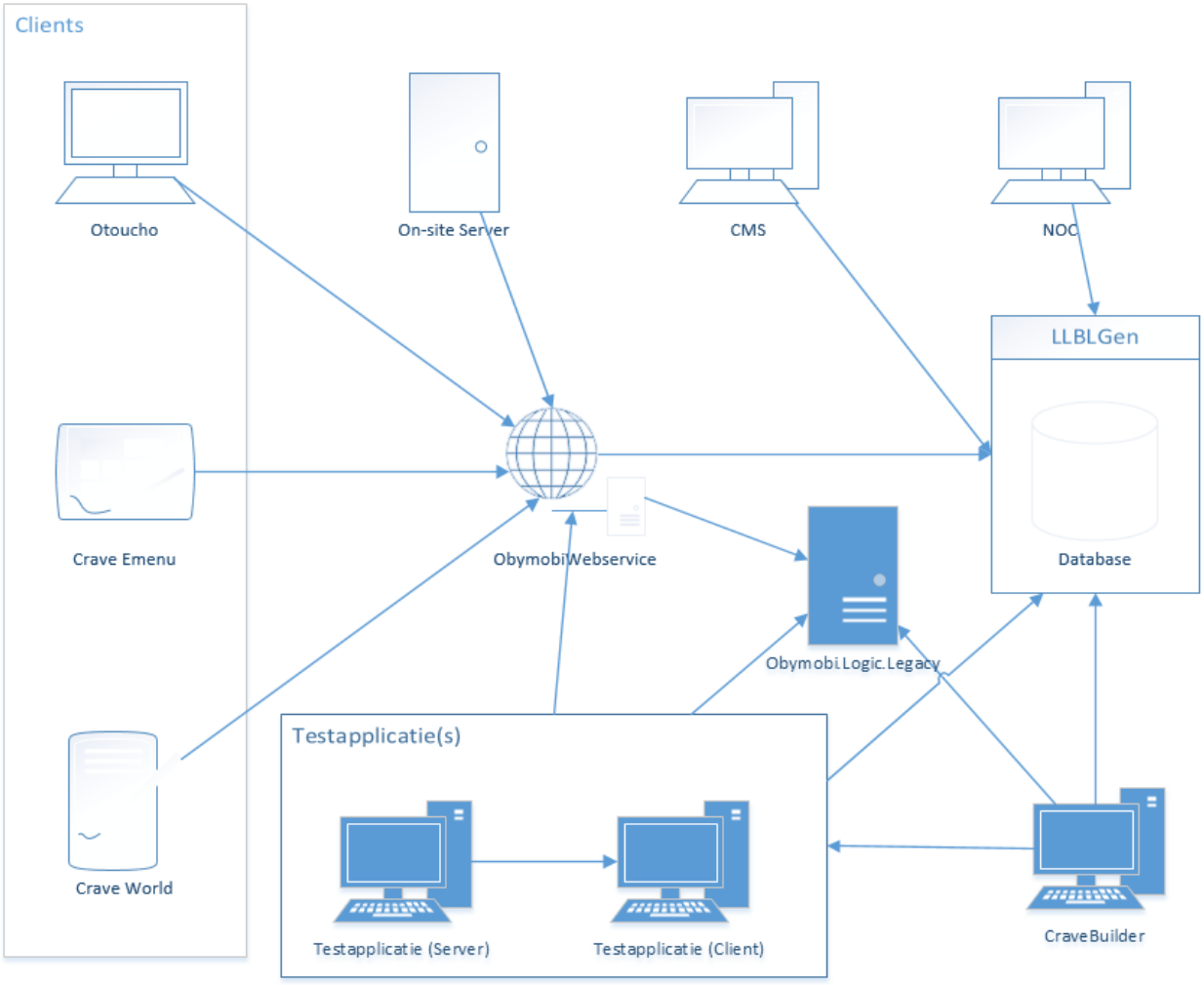
Project:	Het ontwikkelen van een geautomatiseerde testapplicatie	
Bijlage:	Architectuurrapport	
Auteur:	Floris Otto	11073551
Bedrijf:	Crave Interactive B.V.	
Opdrachtgever:	Dhr. G. van der Kruijk	
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg	
Tweede examinator:	Dhr. P.R. Smit	
Datum:	10 maart 2014	
Versie:	001	

Inhoud

1. Architectuur oude situatie

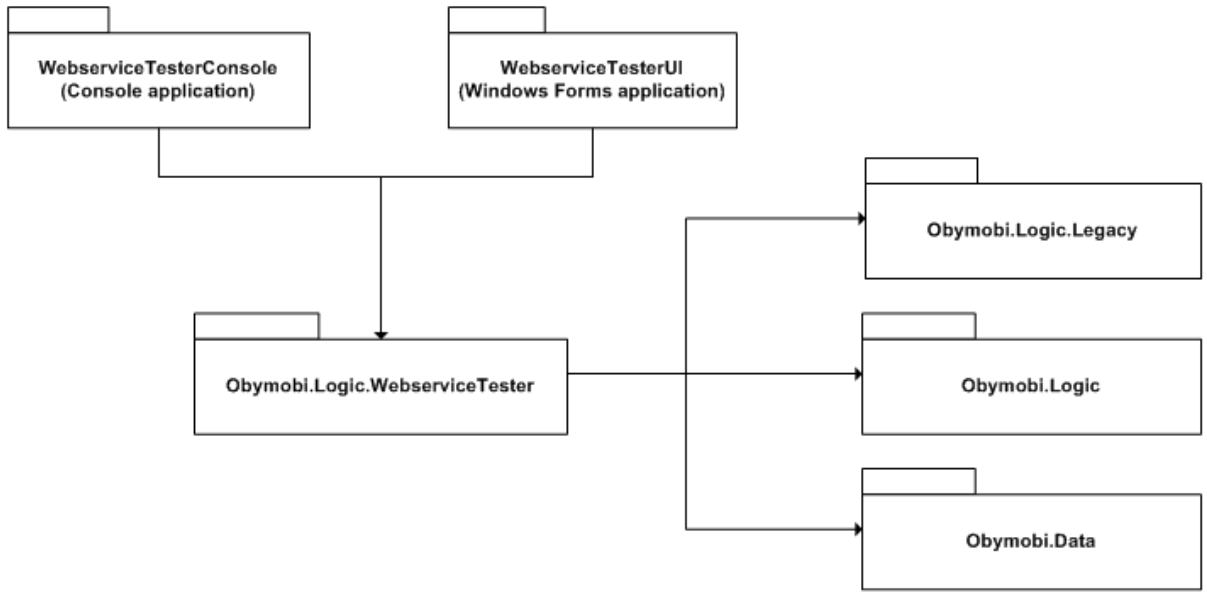


2. Architectuur nieuwe situatie

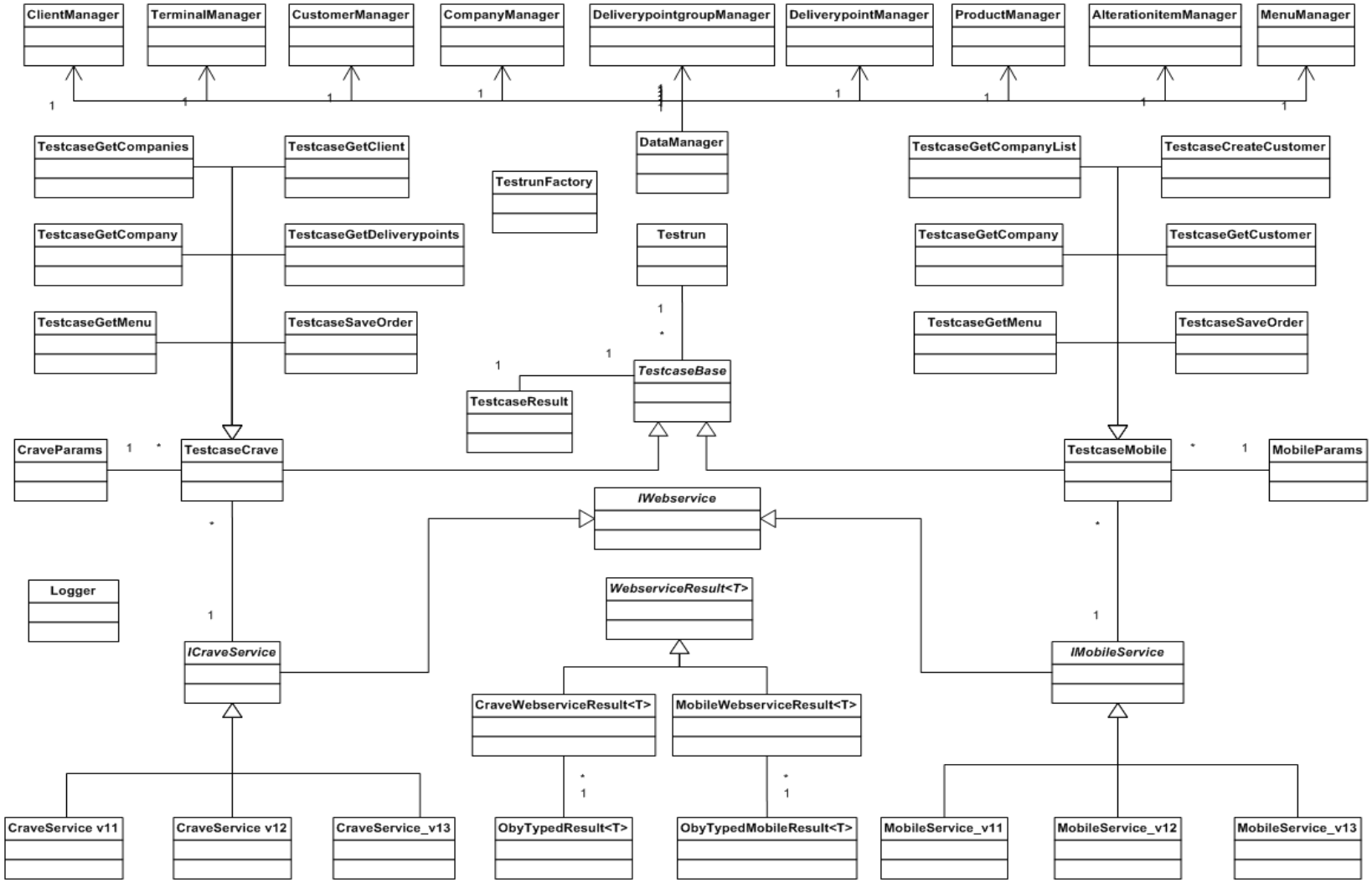


3. Interne architectuur testapplicatie

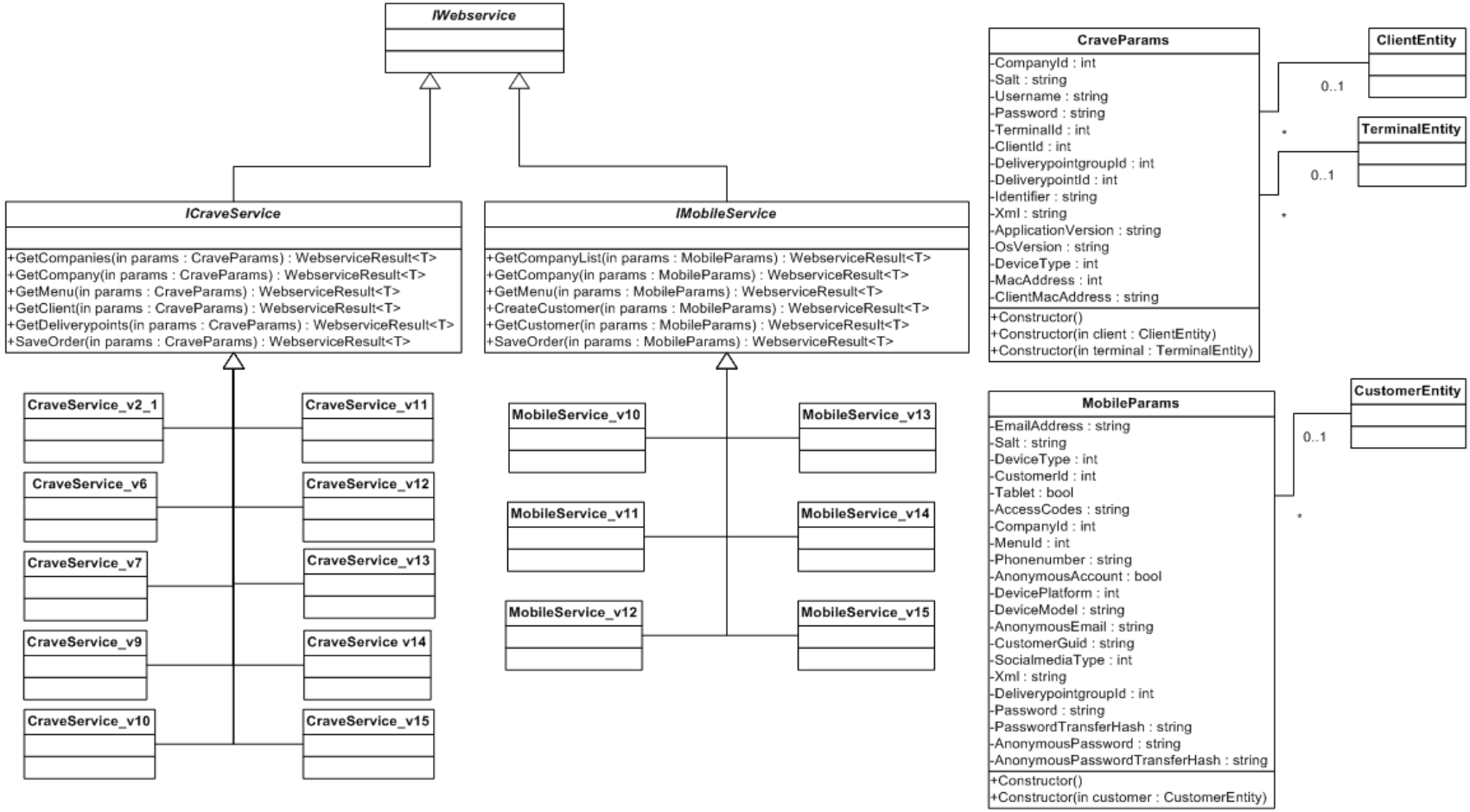
3.1 Componenten



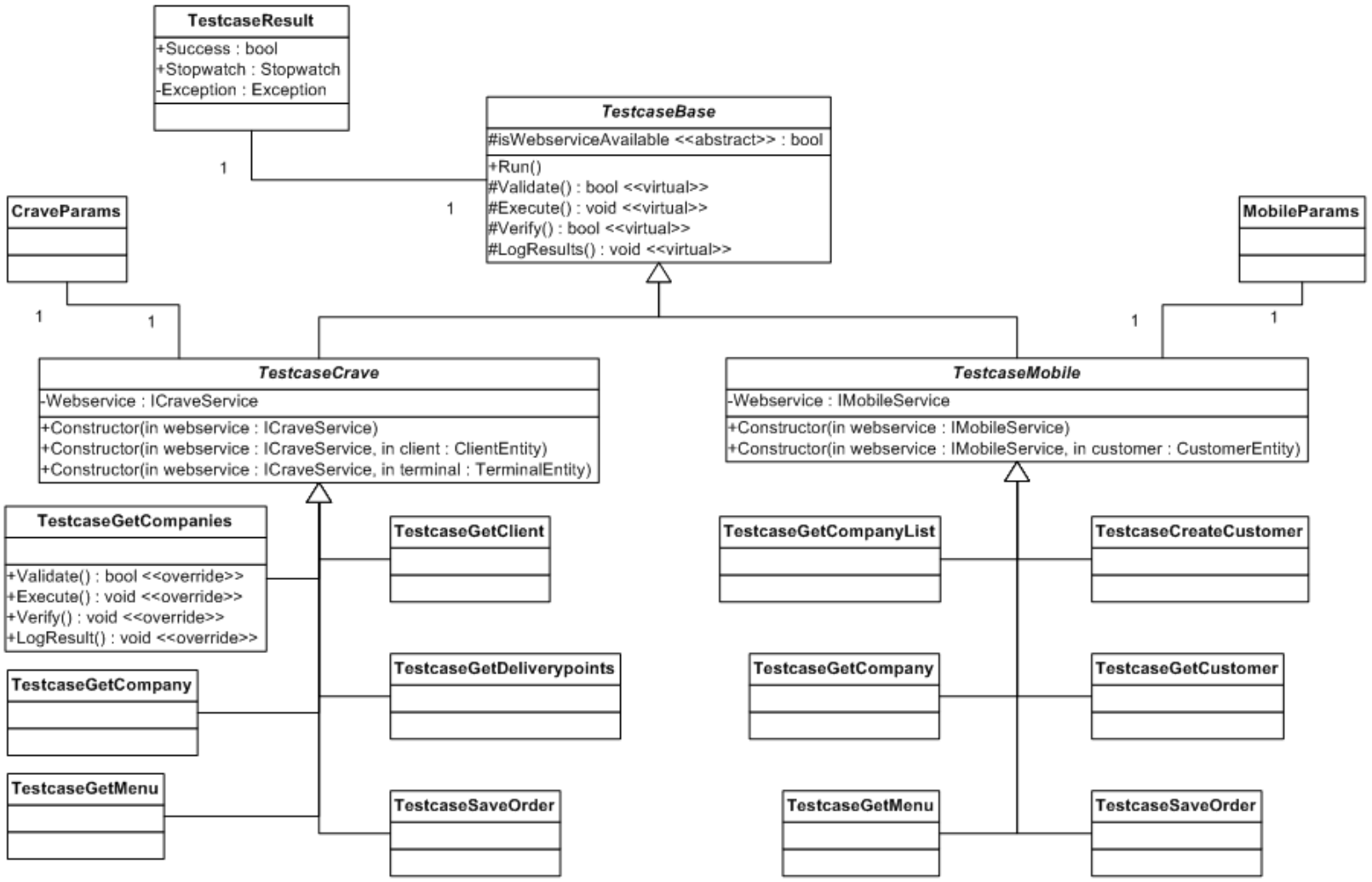
3.2 Conceptueel klassendiagram



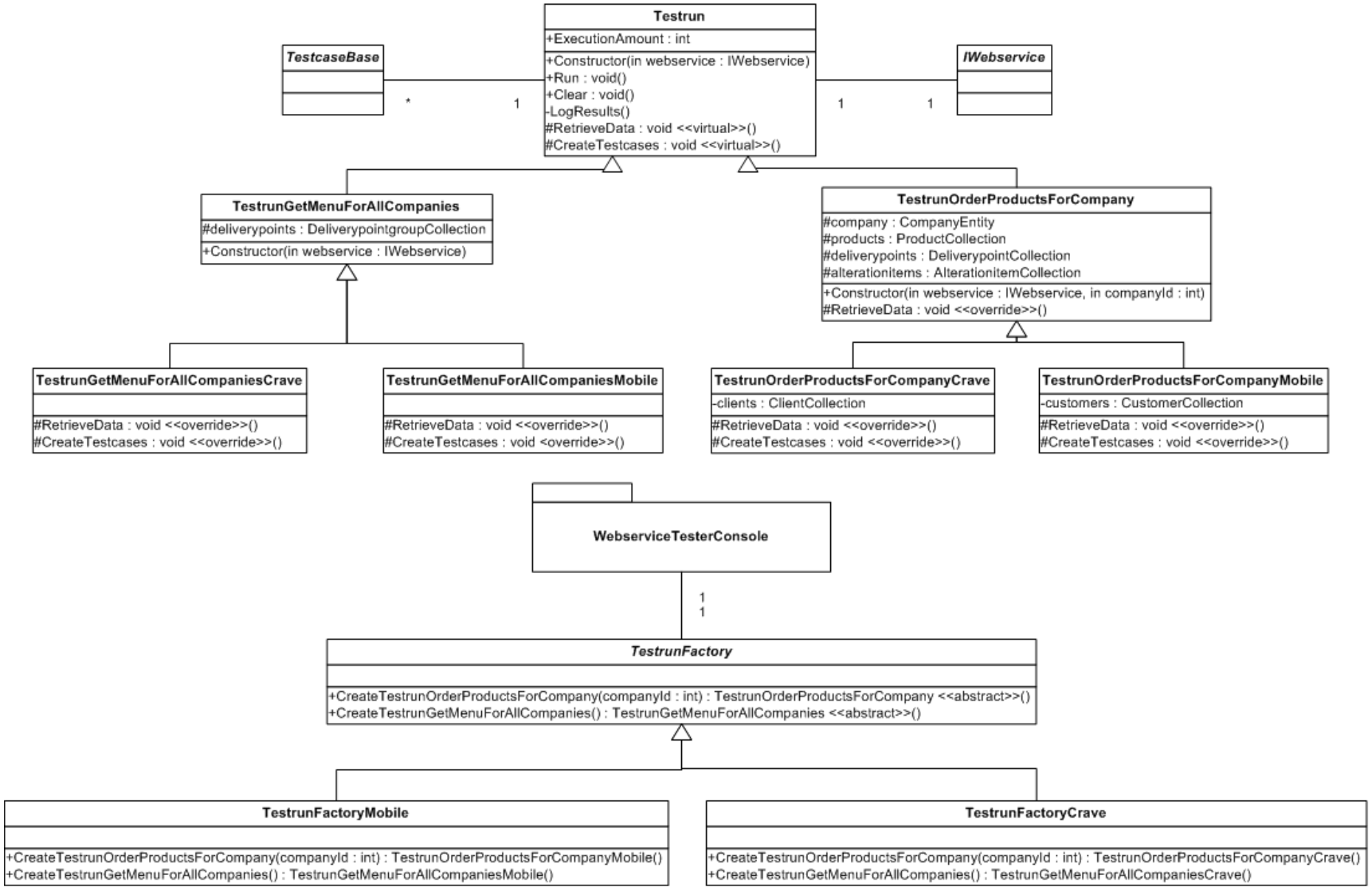
3.3 Klassendiagram: Webservices



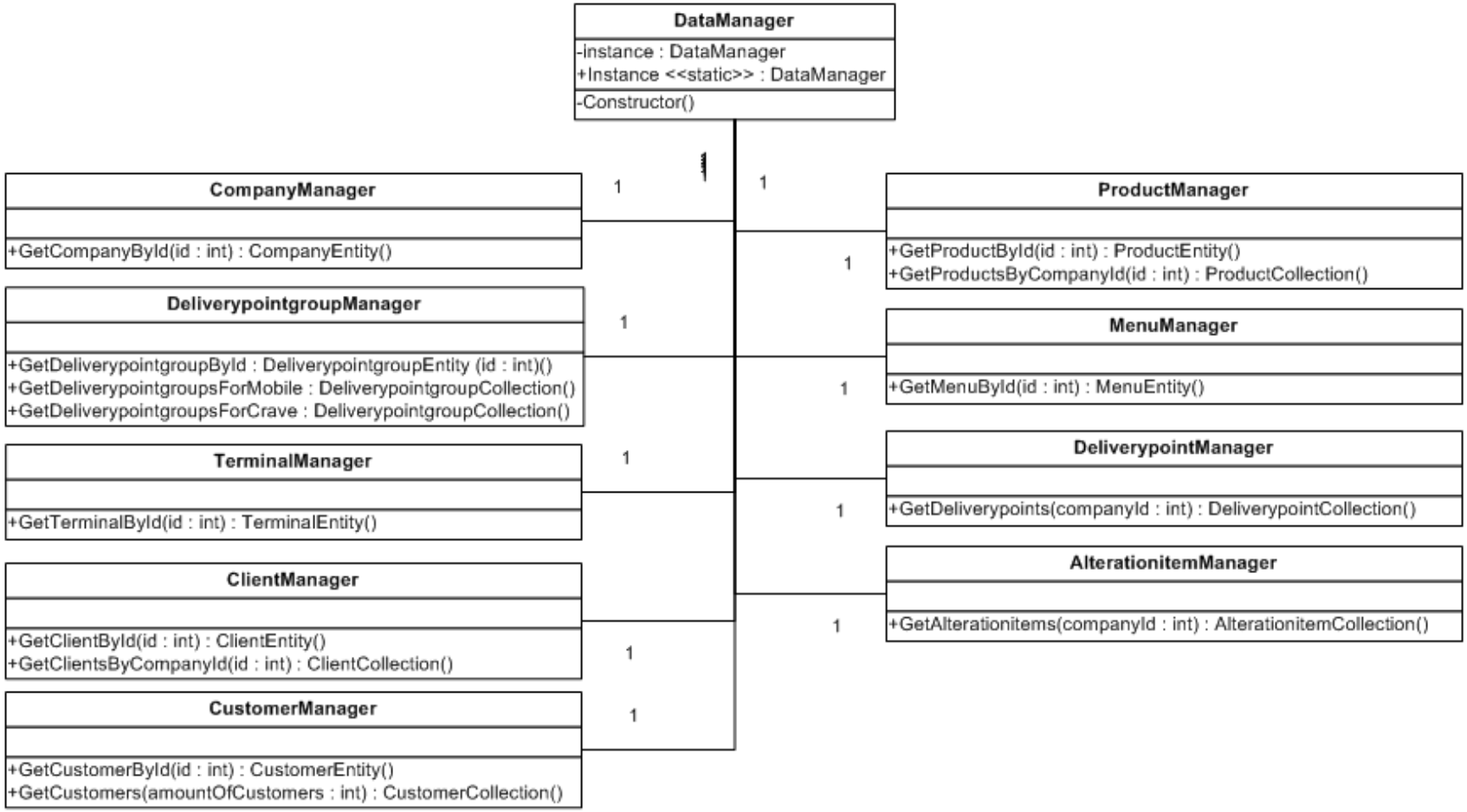
3.4 Klassendiagram: Testcases



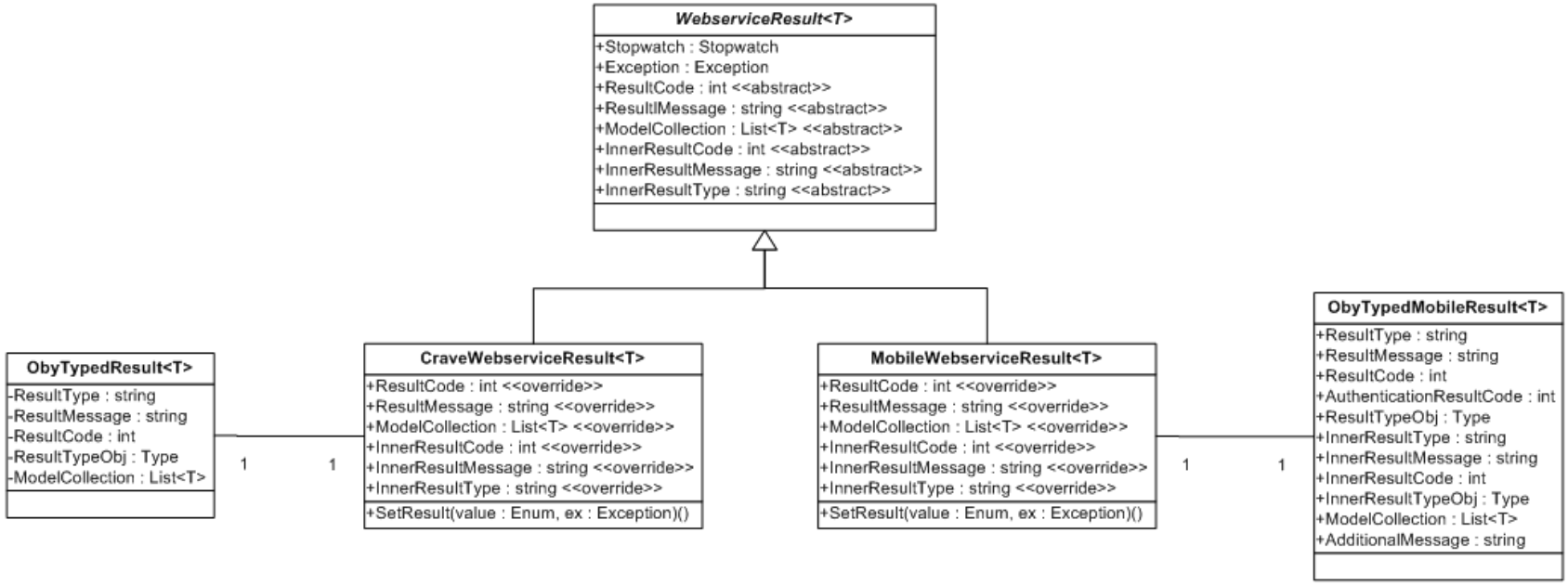
3.5 Klassendiagram: Test runs



3.6 Klassendiagram: Validating test results



3.7 Klassendiagram: Error reporting



Bijlage F: Testrapport

Het ontwikkelen van een geautomatiseerde testapplicatie

Project:	Het ontwikkelen van een geautomatiseerde testapplicatie	
Bijlage:	Testrapport	
Auteur:	Floris Otto	11073551
Bedrijf:	Crave Interactive B.V.	
Opdrachtgever:	Dhr. G. van der Kruijk	
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg	
Tweede examinator:	Dhr. P.R. Smit	
Datum:	10 maart 2014	
Versie:	001	

crave

DE HAAGSE HOGESCHOOL

Inhoud

1. Opgestelde tests

Dit hoofdstuk bevat alle tests die zijn opgesteld om de testapplicatie te kunnen testen. Na afronding van elke feature zijn, indien mogelijk, tests opgesteld om de feature te kunnen testen. De tests zijn in het Engels aangezien het testteam van Crave Interactive B.V. zich in Engeland bevindt en de tests mogelijk in de toekomst ook uitgevoerd gaan worden door dit testteam. Tijdens het project zijn de tests door de afstudeerder uitgevoerd.

1.1 Test runs

ID:	T01
Name:	Execute test run *TestrunGetMenuForAllCompanies* for the latest version of the CraveService.
Preconditions:	• Console version of the testapplication is installed. • At least one company available for usage with the CraveService.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunGetMenuForAllCompanies –webserviceType craveservice –webserviceVersion <latest version>* 4. Replace *<latest version>* with the latest version of the CraveService. (e.g. 15.0) 5. Press enter.
Expected result:	The testapplication is creating *TestrunGetMenu* test cases for all the available companies after which the test cases are executed on the latest CraveService. The process is shown in the console.

ID:	T02
Name:	Execute test run *TestrunGetMenuForAllCompanies* for an old version of the CraveService.
Preconditions:	• Console version of the testapplication is installed. • At least one company available for usage with the CraveService.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunGetMenuForAllCompanies –webserviceType craveservice –webserviceVersion <old version>* 4. Replace *<old version>* with an older version number of the CraveService. (e.g. 10.0) 5. Press enter.
Expected result:	The testapplication is creating *TestrunGetMenu* test cases for all the available companies after which the test cases are executed on the chosen version of the CraveService. The process is shown in the console.

ID:	T03
Name:	Execute test run *TestrunGetMenuForAllCompanies* for the latest version of the MobileService.
Preconditions:	• Console version of the testapplication is installed. • At least one company available for usage with the MobileService.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunGetMenuForAllCompanies –webserviceType mobileservice –webserviceVersion <latest version>* 4. Replace *<latest version>* with the latest version of the MobileService. (e.g. 15.0) 5. Press enter.
Expected result:	The testapplication is creating *TestrunGetMenu* test cases for all the available companies after which the test cases are executed on the latest version of the MobileService. The process is shown in the console.

ID:	T04
Name:	Execute test run *TestrunGetMenuForAllCompanies* for an old version of the MobileService.
Preconditions:	• Console version of the testapplication is installed. • At least one company available for usage with the MobileService.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunGetMenuForAllCompanies –webserviceType mobileservice –webserviceVersion <old version>* 4. Replace *<old version>* with an older version number of the MobileService. (e.g. 10.0) 5. Press enter.
Expected result:	The testapplication is creating *TestrunGetMenu* test cases for all the available companies after which the test cases are executed on the chosen version of the MobileService. The process is shown in the console.

ID:	T05
Name:	Execute test run *TestrunOrderProductsForCompany* for the latest version of the CraveService.
Preconditions:	• Console version of the testapplication is installed. • A company is available for usage with the CraveService. • The company has a number of products available for ordering.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType craveservice –*

	webserviceVersion <latest version> -companyId <companyId> 4. Replace *<latest version>* with the latest version of the CraveService. (e.g. 15.0) 5. Replace *<companyId>* with the companyId of the company as mentioned in the preconditions. 6. Press enter.
Expected result:	The testapplication is creating *TestrunSaveOrder* test cases with random products of the company after which the test cases are executed on the latest CraveService. The process is shown in the console.

ID:	T06
Name:	Execute test run *TestrunOrderProductsForCompany* for an old version of the CraveService.
Preconditions:	• Console version of the testapplication is installed. • A company is available for usage with the CraveService. • The company has a number of products available for ordering.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType craveservice –webserviceVersion <old version> -companyId <companyId>* 4. Replace *<old version>* with an older version number of the CraveService. (e.g. 10.0) 5. Replace *<companyId>* with the companyId of the company as mentioned in the preconditions. 6. Press enter.
Expected result:	The testapplication is creating *TestrunSaveOrder* test cases with random products of the company after which the test cases are executed on the chosen version of the CraveService. The process is shown in the console.

ID:	T07
Name:	Execute test run *TestrunOrderProductsForCompany* for the latest version of the MobileService.
Preconditions:	• Console version of the testapplication is installed. • A company is available for usage with the MobileService. • The company has a number of products available for ordering.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType mobileservice –webserviceVersion <latest version> -companyId <companyId>* 4. Replace *<latest version>* with the latest version of the MobileService. (e.g. 15.0) 5. Replace *<companyId>* with the companyId of the company as mentioned in the preconditions.

	6. Press enter.
Expected result:	The testapplication is creating *TestrunSaveOrder* test cases with random products of the company after which the test cases are executed on the latest version of the MobileService. The process is shown in the console.

ID:	T08
Name:	Execute test run *TestrunOrderProductsForCompany* for an old version of the MobileService.
Preconditions:	• Console version of the testapplication is installed. • A company is available for usage with the MobileService. • The company has a number of products available for ordering.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType mobileservice –webserviceVersion <old version> -companyId <companyId>* 4. Replace *<old version>* with an older version number of the MobileService. (e.g. 10.0) 5. Replace *<companyId>* with the companyId of the company as mentioned in the preconditions. 6. Press enter.
Expected result:	The testapplication is creating *TestrunSaveOrder* test cases with random products of the company after which the test cases are executed on the chosen version of the MobileService. The process is shown in the console.

ID:	T09
Name:	Execute a test run a single time.
Preconditions:	• Console version of the testapplication is installed. • At least one company available for usage with the CraveService.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType craveservice –webserviceVersion <latest version> -executionAmount 1* 4. Replace *<latest version>* with the latest version number of the CraveService. (e.g. 15.0) 5. Press enter.
Expected result:	The testapplication is creating *TestrunGetMenu* for the available companies after which the test cases are all executed one time on the CraveService. The process is shown in the console.

ID:	T10
Name:	Execute a test run multiple times.
Preconditions:	• Console version of the testapplication is installed.

	• At least one company available for usage with the CraveService.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType craveservice –webserviceVersion <latest version> -executionAmount 2* 4. Replace *<latest version>* with the latest version of the CraveService. (e.g. 15.0) 5. Press enter.
Expected result:	The testapplication is creating *TestrunGetMenu* test cases for the available companies after which the test cases are all executed two times on the CraveService. The process is shown in the console.

1.2 CraveBuilder (to be tested by a developer)

ID:	T11
Name:	Create a new crave webservice.
Preconditions:	• Latest version of the CraveBuilder installed. • Obymobi.WebserviceTester solution checked out from SVN at: "D:\Development\DotNet\Obymobi.WebserviceTester".
Steps:	1. Start the CraveBuilder. 2. Specify the correct dotnet folder. 3. Click “Tools” in the top bar and click “Create new webservice version…”. 4. Enter the correct version number for the new CraveService and click “Create new webservice version”.
Expected result:	1. A new CraveService is created in the *ObymobiWebservice* project. 2. A set of models and converters has been created for the previous CraveService version in the *Obymobi.Logic.Legacy* project. 3. A new communication class has been created for the testapplication and is located at: D:\Development\DotNet\Obymobi.WebserviceTester\Obymobi.WebserviceTester\WebServices\<new version number>\CraveService.cs 4. The communication class of the previous CraveService has been modified for the testapplication and is now using Converters for that particular version.

ID:	T12
Name:	Create a new mobile webservice.
Preconditions:	• Latest version of the CraveBuilder installed. • Obymobi.WebserviceTester solution checked out from SVN at: "D:\Development\DotNet\Obymobi.WebserviceTester".
Steps:	1. Start the CraveBuilder. 2. Specify the correct dotnet folder. 3. Click “Tools” in the top bar and click “Create new webservice version…”. 4. Enter the correct version for the new MobileService and click “Create new

	webservice version".
Expected result:	1. A new MobileService is created in the *ObymobiWebservice* project. 2. A set of models and converters has been created for the previous MobileService version in the *Obymobi.Logic.Legacy* project. 3. A new communication class has created for the testapplication and is located at: D:\Development\DotNet\Obymobi.WebserviceTester\Obymobi.Webservice Tester\WebServices\<new version number>\MobileService.cs 4. The communication class of the previous MobileService has been modified for the testapplication and is now using Converters for that particular version.

1.3 Validating test results

ID:	T13
Name:	Validate test results with data in database
Preconditions:	• Console version of the testapplication is installed. • A company is available for usage with the MobileService. • The company has a number of products available for ordering.
Steps:	1. Open the command prompt. 2. Navigate to the same directory as the testapplication. 3. Enter the following line in the command prompt: *WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType craveservice –webserviceVersion <latest version> -companyId <companyId>* 4. Replace <*latest version*> with the latest version of the CraveService. (e.g. 15.0) 5. Replace <*companyId*> with the companyId of the company as mentioned in the preconditions. 6. Press enter.
Expected result:	The *TestcaseSaveOrder* test cases are created with random products for the company after which the test cases are executed. In the result message of the test case a message is displayed if the data is successfully verified with data in the database. *"The received orders has been successfully verified with orders in the database".*

2. Testresultaten

In dit hoofdstuk worden de resultaten van de uitgevoerde tests getoond. Aan het eind van elke sprint zijn alle, indien beschikbare, tests uitgevoerd om te kunnen verifiëren of het product nog naar behoren functioneerde. Hierbij zijn dus ook features van de vorige sprints getest om te controleren of de implementatie van nieuwe features er niet voor gezorgd heeft dat bestaande functionaliteit is aangetast.

Test ID	Geslaagd	Werkelijke resultaat
T01	Ja	-
T02	Ja	-
T03	Ja	-
T04	Ja	-
T05	Ja	-
T06	Ja	-
T07	Ja	-
T08	Ja	-
T09	Ja	-
T10	Ja	-
T11	Ja	-
T12	Ja	-

Tabel 2.1: Uitslagen van tests na afronding van sprint 3.

Test ID	Geslaagd	Werkelijke resultaat
T01	Ja	-
T02	Ja	-
T03	Ja	-
T04	Ja	-
T05	Ja	-
T06	Ja	-
T07	Ja	-
T08	Ja	-
T09	Ja	-
T10	Ja	-
T13	Ja	-

Tabel 2.2: Uitslagen van tests na afronding van sprint 4.

Bijlage G: Handleiding

Het ontwikkelen van een geautomatiseerde testapplicatie

Project:	Het ontwikkelen van een geautomatiseerde testapplicatie	
Bijlage:	Handleiding	
Auteur:	Floris Otto	11073551
Bedrijf:	Crave Interactive B.V.	
Opdrachtgever:	Dhr. G. van der Kruijk	
Begeleidend examinator:	Mevr. A.M.J.J. Lousberg	
Tweede examinator:	Dhr. P.R. Smit	
Datum:	19 mei 2014	
Versie:	001	

Manual

This manual describes the functionality of the console version of the testapplication for automated testing of the webservice. The testapplication makes it possible to execute different webservice calls automatically without the need of setting up a device to test with. Both the CraveService as the MobileService can be tested. The console version of the application currently contains two predefined test runs which can be used for both webservices. The amount of test runs will increase in time as more test runs will be implemented in the near future. The GUI version of the testapplication will be able to create test runs dynamically by giving the user the ability to add test cases and saving them for future purposes. This version will also be released in the near future.

Installation

1. Copy the following directory to the desired location on your computer:

   "O:\Software Development\Releases\WebserviceTesterConsole\"

2. Modify the WebserviceTester.config file to test on the desired environment:
   a. Set the cloud environment
   b. In case of manual cloud: Set the ManualWebserviceBaseUrl

Available parameters

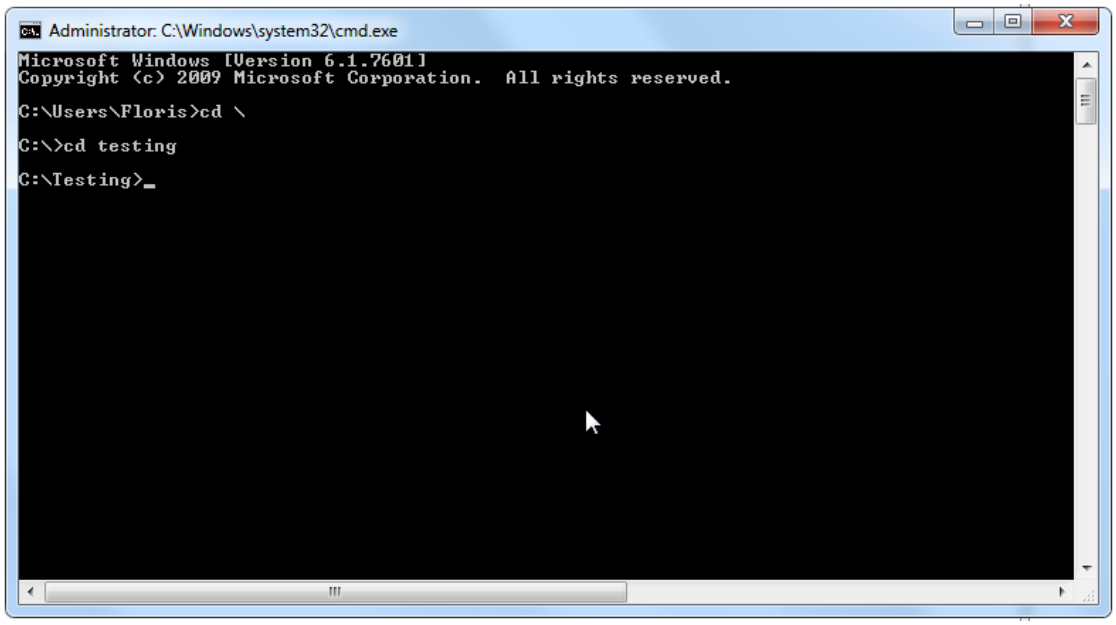
• Testruns	(e.g. TestrunOrderProductsForCompany)	Mandatory
• WebserviceType	(e.g. CraveService, MobileService)	Mandatory
• WebserviceVersion	(e.g. 14.0, 15.0)	Mandatory
• ExecutionAmount	(e.g. 1, 2)	Optional
• CompanyId	(e.g. 199, 235)	Optional

Available test runs with necessary parameters

- TestrunOrderProductsForCompany
  - –testruns
  - –webserviceType
  - –webserviceVersion
  - –companyId
- TestrunGetMenuForAllCompanies
  - –testruns
  - –webserviceType
  - –webserviceVersion

Executing a test run

1. Open the command prompt by clicking start and type “CMD” after which you press enter.
2. Navigate to the same directory as the testapplication.



The testapplication is located at "C:\Testing\" in the example above.

3. Enter one of the following lines to execute a test run. The parameters to use can differ between the test run to execute. For example:

TestrunGetMenuForAllCompanies

WebserviceTesterConsole.exe –testruns TestrunGetMenuForAllCompanies –webserviceType craveservice –webserviceVersion 15.0

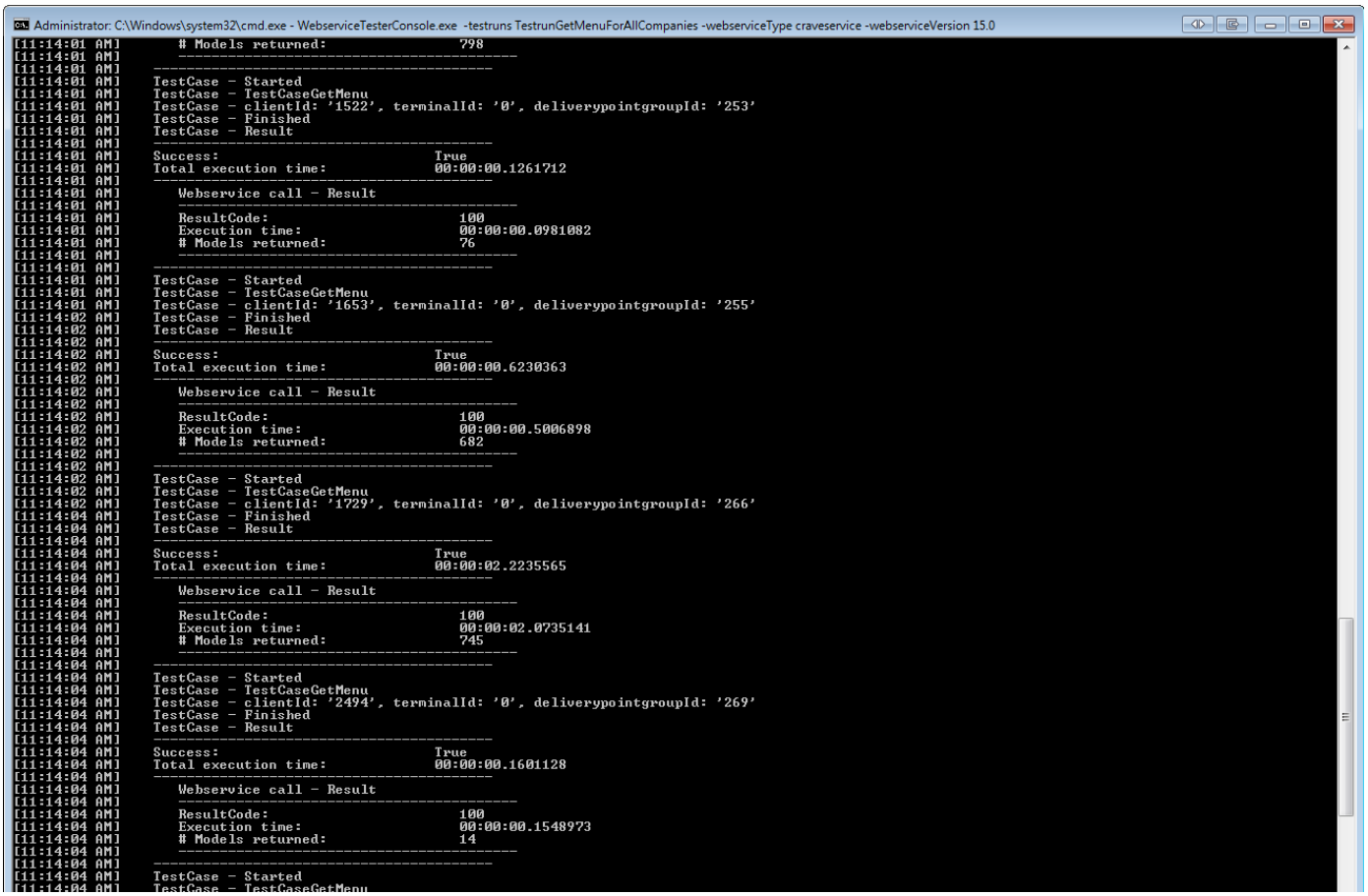
TestrunOrderProductsForCompany

WebserviceTesterConsole.exe –testruns TestrunOrderProductsForCompany –webserviceType mobileservice –webserviceVersion 14.0 –companyId 199

4. The values behind the parameters, the ones starting with -, can be modified according to what needs to be tested.

About to test the TestrunGetMenuForAllCompanies test run for version 15.0 of the CraveService.

5. After entering the desired test run to test, press enter. The testapplication will start creating test cases, after which he'll execute them one after the other.



The TestcaseGetMenu test cases are being executed successfully.

6. The results are being saved as log files per day. The directory of the log files can be found in the same directory as where the testapplication is installed.