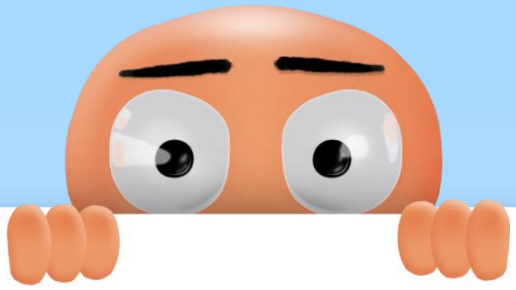
Eerste examinator:	A.A.A.M. Jacobs
Tweede examinator:	R. Ruijsenaars
Begeleider Gamebasics:	G. Meuldijk
School:	De Haagse Hogeschool
Opleiding:	Informatica



Scoutlijst

Afstudeerverslag

06-06-2013 Versie 1.0

Remco Overdevest

10040536

Referaat

Remco Overdevest, afstudeeropdracht, Ontwikkelen van de scoutlijst,
Gamebasics bv., Zoetermeer, Februari 2013

Student Remco Overdevest heeft in de periode van 11 februari 2013 tot en met 7 juni 2013 zijn afstudeeropdracht uitgevoerd bij Gamebasics bv.
Het project 'Ontwikkelen van de scoutlijst' is uitgevoerd in opdracht van Gamebasics B.V.

Descriptoren:

- Online Soccer Manager
- ASP.NET MVC
- Mobile development
- RUP
- UML
- Gamebasics
- Objective C
- Java
- Android
- iOS

Versiebeheer

Versie	Datum	Auteur	Wijziging
0.1	17-3-2013	Remco Overdevest	Eerste opzet afstudeerverslag, inleiding en bedrijf beschreven.
0.2	19-3-2013	Remco Overdevest	Opdracht omschrijving.
0.3	27-3-2013	Remco Overdevest	Glossary en Literatuurlijst toegevoegd.
0.4	19-4-2013	Remco Overdevest	Plan van aanpak beschrijving afgerond.
0.5	22-4-2013	Remco Overdevest	Begin gemaakt aan het beschrijven van de inception fase.
0.6	29-4-2013	Remco Overdevest	Vooronderzoek beschreven en inception fase afgerond.
0.7	7-5-2013	Remco Overdevest	Aanpassingen systeemeisen aan elaboration fase toegevoegd.
0.7	8-5-2013	Remco Overdevest	Elaboration fase verder uitwerken.
0.8	16-5-2013	Remco Overdevest	Construction fase iteratie twee beschrijven.
0.9	17-5-2013	Remco Overdevest	Iteratie twee van construction fase toegevoegd.
0.10	28-5-2013	Remco Overdevest	Feedback verwerkt.
0.11	31-5-2013	Remco Overdevest	Statisch testen verder uitgewerkt. Acceptatietest toegevoegd.
1.0	4-6-2013	Remco Overdevest	Afronden afstudeerverslag

Voorwoord

Dit verslag is geschreven naar aanleiding van het afstuderen bij Gamebasics. Het afstudeerproject is uitgevoerd als onderdeel van mijn opleiding Informatica aan de Haagse Hogeschool.

Hierbij wil ik van de gelegenheid gebruik maken om personen te bedanken die betrokken zijn geweest bij de afstudeeropdracht of mij gesteund hebben tijdens het afstudeertraject.

Als eerste wil ik Jeroen Derwort en Frank Tijhuis bedanken voor de mogelijkheid om het afstudeertraject bij Gamebasics door te lopen.

Gijs Meuldijk wil ik bedanken voor de begeleiding tijdens mijn afstudeerproject.

Als laatste wil ik de ontwikkelaars van Gamebasics bedanken voor hun adviezen en tips tijdens mijn afstudeerproject.

Remco Overdevest,

Zoetermeer, februari 2013

Inhoud

1 Inleiding

In dit hoofdstuk wordt de aanleiding van dit document beschreven, vervolgens wordt er kort beschreven wat er in dit document staat en waar dit terug te vinden is.

1.1 Aanleiding

Het procesverslag is in het kader van het afstudeertraject opgesteld en geeft verantwoording over het proces dat doorlopen is tijdens de afstudeerperiode. Gedurende de afstudeerperiode van februari 2013 tot en met juni 2013 is er bij het bedrijf Gamebasics een afstudeerproject uitgevoerd. Het (her)ontwikkelen van de Online Soccer Manager scoutlijst voor verschillende platformen zal in dit document centraal staan.

1.2 Doelstelling

De doelstelling van het verslag is het inzicht geven in de door de student uitgevoerde werkzaamheden gedurende het afstudeertraject. Op basis van dit verslag kunnen examinatoren een beoordeling geven en bepalen of de student de van te voren aangegeven competenties heeft voldaan.

1.3 Structuur

Aan het begin van dit document is er een beschrijving van het bedrijf te vinden (Hoofstuk 2) en zijn de aanleiding, probleemstelling en doelstelling in hoofdstuk 3 beschreven. Tevens wordt er in hoofdstuk 3 de uitgangspositie beschreven.

Hierop volgt een beschrijving van het plan van aanpak in hoofdstuk 4. Hierbij wordt niet ingegaan op de inhoud van het plan van aanpak, maar de totstandkoming van het document en waarom er is gekozen voor een aantal technieken en methoden.

Vervolgens is dit document ingedeeld in RUP fases die zijn doorlopen tijdens dit project. Waarbij de inception fase, in hoofdstuk 5, als eerst aan bod komt. In deze fase is het project opgestart en zijn de systeemeisen en use-cases opgesteld. In hoofdstuk 6 wordt de volgende fase van RUP beschreven, de elaboration fase. In de elaboration fase zijn de systeemeisen aangepast en zijn de scoutlijst applicaties ontworpen.

Daarna wordt in hoofdstuk 7 een beschrijving gegeven van de construction fase, de fase waarin het ontwikkelen van de scoutlijst applicaties beschreven staan. De laatste fase van RUP, de transition fase, wordt in hoofdstuk 8 beschreven. Hierin vinden de acceptatie en overdracht plaats.

Gereflecteerd wordt er in hoofdstuk 9, een evaluatie over het uitgevoerde proces en de opgeleverde producten. In de evaluatie komen voor de sterke en zwakke punten naar voren en wordt er gekeken naar wat ik de volgende keer anders zou aanpakken.

Aan het einde van dit document is een woordenlijst opgenomen, deze bestaat uit uitleg van technische begrippen. Ook is hier een literatuurlijst te vinden van gebruikte bronnen tijdens dit project. Alle documenten die opgeleverd zijn tijdens dit project zijn te vinden in de bijlagen.

2 Organisatie

Dit hoofdstuk omschrijft de organisatie waar de afstudeerder het afstudeerproject heeft uitgevoerd.

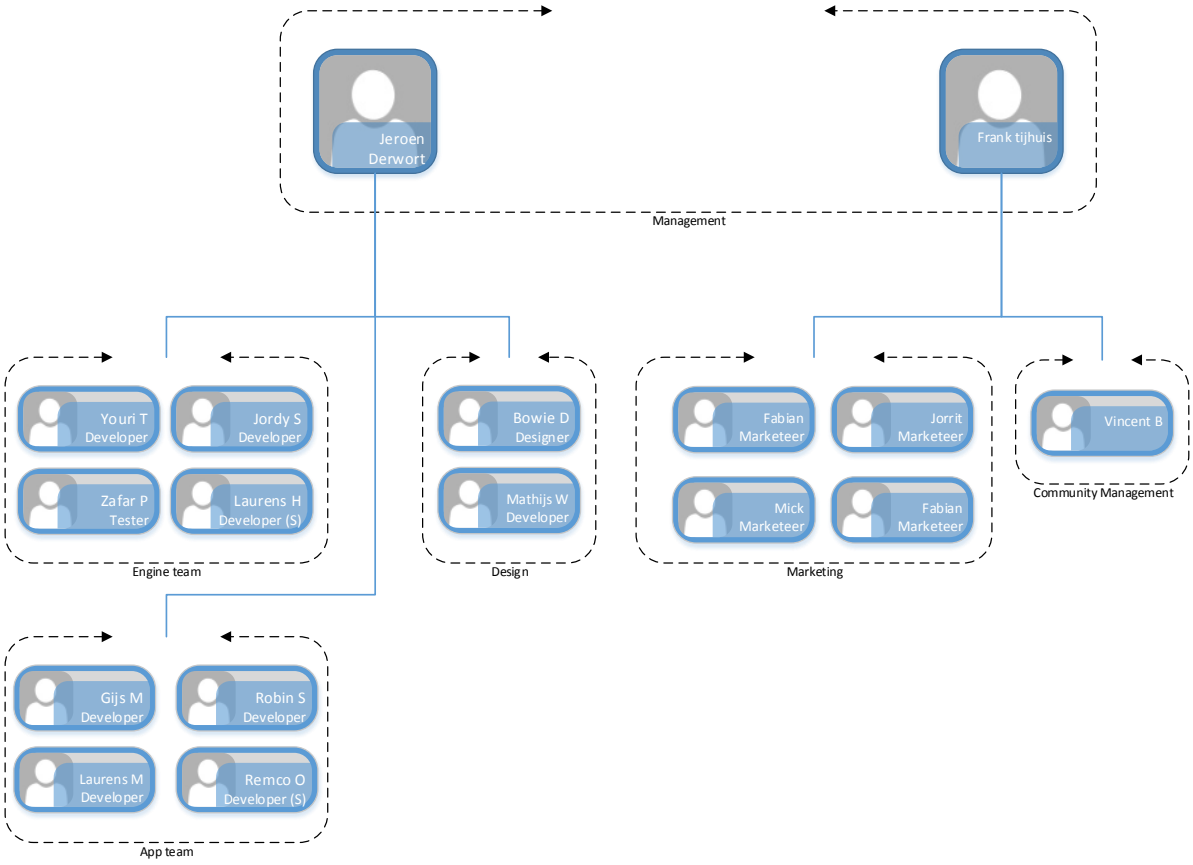
2.1 Organisatie omschrijving

Gamebasics BV ontwikkelt en exploiteert online community games in Nederland en in het buitenland. Het is opgericht in 2004 en is het bedrijf achter de bekende voetbalmanagement game Online Soccer Manager. Gamebasics heeft in het verleden verschillende online manager games ontwikkeld, zoals het Bondscoachspel en Plaza Challenge, maar focust zich de laatste jaren volledig op Online Soccer Manager. De missie van Gamebasics is als volgt:

Gamebasics wil zoveel mogelijk voetballiefhebbers wereldwijd in gelokaliseerde communities 10 minuten per dag het gevoel geven dat zij manager zijn van hún favoriete club. Dit willen we bereiken door te streven naar een optimale spelbeleving, waarbij je met je vrienden, op elk gewenst moment, via elk mainstream platform je voetbalkennis kan etaleren.

2.2 Organisatie structuur

Gamebasics telt op dit moment 18 werknemers. Deze zijn werkzaam in de administratie, marketing en development. De twee huidige directeuren zijn de oprichters van Gamebasics en zijn tevens de bedenkers van Online Soccer Manager.



Figuur 1 – Organigram Gamebasics

3 Opdracht: De Online Soccer Manager Scoutlijst

In dit hoofdstuk wordt de opdracht omschreven die tijdens het afstudeertraject is uitgevoerd.

3.1 Aanleiding

In 2004 is de eerste versie van de Online Soccer Manager scoutlijst ontwikkeld. Deze was oorspronkelijk gebouwd voor een geringe set requirements en voor een klein aantal gebruikers. Door de groei van Online Soccer Manager is er steeds meer vraag naar een scoutlijst met uitgebreide functionaliteiten.

Sinds de zomer van 2012 is er een Online Soccer Manager applicatie voor de twee meest gebruikte mobiele platformen, Android en iOS. Maar liefst 45% van alle actieve managers spelen Online Soccer Manager met behulp van hun mobiele device. Doordat er zoveel gebruikers op de mobiele platformen spelen is er een groeiende vraag naar een uitgebreide scoutlijst applicatie.

3.2 Probleemstelling

Gamebasics heeft besloten om de huidige scoutlijst te herschrijven en te ontwikkelen voor verschillende platformen. Op dit moment is er maar één versie van de scoutlijst die toegankelijk is via desktop computers, smartphones en tablets. Door gebruik te maken van de ASP.NET MVC technologie kunnen er verschillende versies van de scoutlijst worden gemaakt op één code basis. Door dit te combineren met responsive web design is er nog maar één versie van de scoutlijst nodig die werkt op veel verschillende resoluties op één code basis. Daarnaast moet de bestaande API van Gamebasics worden uitgebreid om sturing te geven aan de mobiele applicaties van de scoutlijst.

3.3 Doelstelling

Zoals eerder beschreven wil Gamebasics de oude scoutlijst uit faseren en vervangen met een nieuwe versie die gebruikt maakt van de nieuwste technologieën. De doelstelling van dit project is om met behulp van verschillende technieken de scoutlijst aan te bieden op de meest gebruikte platformen. Voor de desktop versie wordt een ASP.NET MVC project opgezet en ook zal de al in ASP.NET MVC gebouwde API van Gamebasics uitgebreid worden om de mobiele scoutlijst versies te laten voorzien van de benodigde data.

3.4 Scope

Dit project zal zich richten op het ontwikkelen van de scoutlijst op de volgende drie platformen; Web, Android en iPhone. Daarbij moet de generieke oplossing om data naar de externe applicatie te versturen worden uitgebreid. Het aanpassen van de bestaande Online Soccer Manager applicaties, waaronder ook de mobiele applicaties, vallen buiten de scope van dit project. Indien er toch aanpassingen worden gedaan in deze applicatie worden hier aantekeningen van gemaakt.

3.5 Uitgangssituatie

In dit hoofdstuk wordt de huidige situatie beschreven. Omdat de Online Soccer Manager Scoutlijst een hulpmiddel is van Online Soccer Manager wordt er als eerst kort beschreven wat Online Soccer Manager is, om vervolgens de rol van de scoutlijst binnen Online Soccer Manager toe te lichten. Daarna wordt er een blik gericht op de architectuur van de huidige scoutlijst.

3.5.1 Wat is Online Soccer Manager?

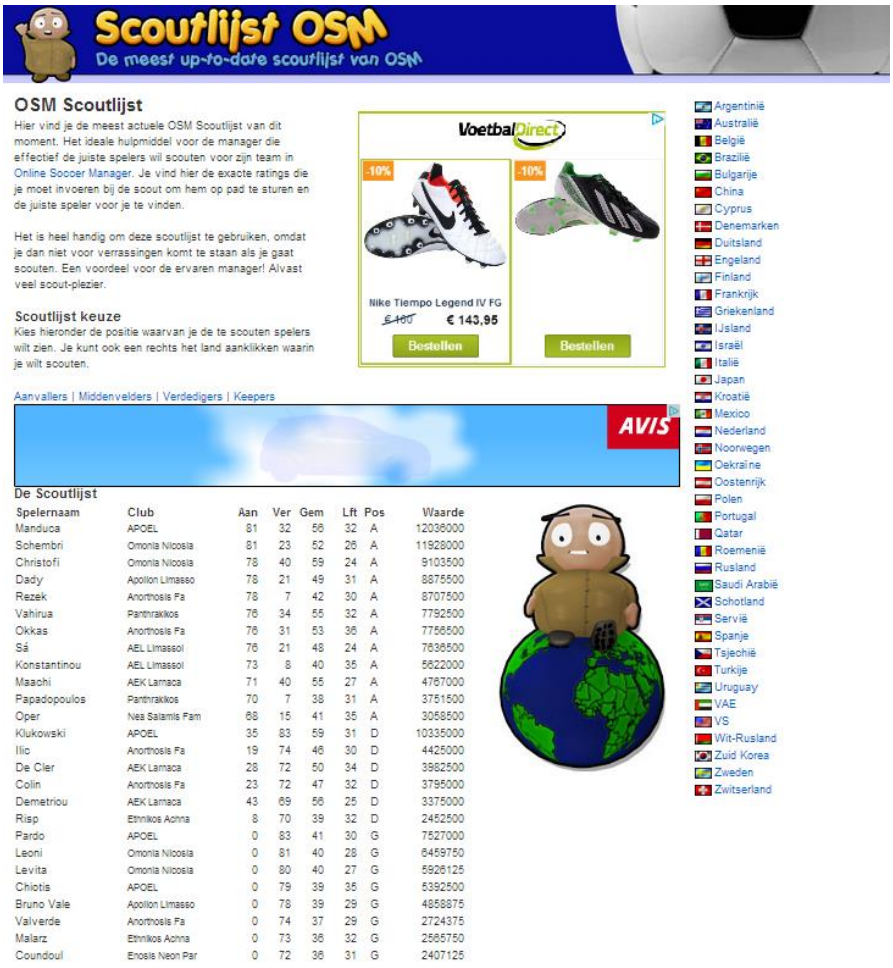
Online Soccer Manager is een online voetbal manager spel, wat gebruikers de mogelijkheid biedt om hun favoriete team te managen. Gebruikers kunnen deelnemen aan reeds bestaande competities of kunnen samen met hun vrienden een nieuwe competitie starten om zo te kijken wie de beste manager is. Gebruikers die eenmaal begonnen zijn om een team te managen, kunnen verschillende manager taken uitvoeren. Zo kan een manager spelers kopen en verkopen, specialisten selecteren, spelers trainen en de tactiek bepalen. In figuur 2 is het startscherm van Online Soccer Manager te zien.

Figuur 2 - OSM Startscherm

Gebruikers hebben de mogelijkheid om voor Online Soccer Manager te betalen en krijgen daardoor toegang tot meerdere functionaliteiten. Stadion upgraden, sub-accounts beheren en het scouten van spelers behoren tot deze extra functionaliteiten. Ook krijgen betaalde gebruikers de mogelijkheid om zelf een competitie te starten, zodat zij ervoor kunnen zorgen dat alleen specifieke vrienden in de betreffende competitie kunnen.

3.5.2 Rol Online Soccer Manager Scoutlijst

Zoals hierboven is beschreven behoort het scouten van spelers bij een betaalde functionaliteit van Online Soccer Manager. Om een speler te scouten dienen er verschillende eigenschappen te worden ingevuld in het scout formulier van Online Soccer Manager. Deze eigenschappen kunnen willekeurig worden ingevuld, maar kunnen ook worden ingevuld om één specifieke speler te vinden. Om de eigenschappen van een betreffende speler te weten is de eerste versie van de scoutlijst gemaakt.



Figuur 3 - De huidige scoutlijst

In de huidige scoutlijst kunnen gebruikers alleen maar spelers zoeken op basis van hun positie of het land van de competitie waar ze in spelen. Ook kan de lijst die getoond wordt niet gesorteerd worden. De eigenschappen van de spelers moeten handmatig ingevoerd worden in het scout formulier van Online Soccer Manager.

4 Plan van aanpak

In dit hoofdstuk wordt de aanpak van het project beschreven. Het doel van het plan van aanpak was om inzicht te krijgen in de uit te voeren werkzaamheden. Ook wilde ik voor alle risico's tijdens dit project een passende oplossing vinden. Uiteindelijk is er op basis van de werkzaamheden en risico's een planning gemaakt. Het complete plan van aanpak is terug te vinden in de bijlagen (Bijlage: Plan van aanpak).

4.1 Project risico analyse

Na het opstellen van de project omschrijving heb ik analyse gemaakt van de projectrisico's. Door het actief inventariseren en bijhouden van mogelijke risico's kunnen tijdig maatregelen bedacht of genomen worden om risico's te verminderen of te vermijden. Uiteindelijk zouden risico's kunnen leiden tot vertraging van het project of zelfs totale mislukking van de opdracht. Een aantal risico's die ik herkend en geanalyseerd heb zijn:

4.1.1 Kennis van programmeertalen

Mijn kennis van verschillende programmeertalen is gering. Door mijn eerder opgedane kennis van de programmeertalen Java en C# ben ik in staat om twee delen van het project naar behoren uit te voren, maar de onvoldoende kennis van Objective-C voor de iPhone applicatie zou eventueel een probleem kunnen worden.

Doordat de Gamebasics ontwikkelaars al eerdere ervaring hebben met het ontwikkelen van een iPhone applicatie met behulp van Objective-C, is het voor mij een oplossing om advies en tips te krijgen van hen. Daarnaast zal ik gedurende het project mijzelf moeten inlezen over het onderwerp via internetbronnen en eventueel aanwezige literatuur.

4.1.2 Kennis van methodieken

Binnen dit project ga ik gebruik maken van de methodieken RUP en UML. Doordat ik alleen kennis van deze methodieken heb vergaard vanuit school, zal ik proberen om mijn kennis aan te vullen door boeken over RUP en UML te lezen die beschikbaar zijn. Eventueel zou ik informatie kunnen vragen aan medewerkers van Gamebasics, omdat zij ervaring hebben in de eerder genoemde methodieken.

4.1.3 Realisatie project

Het project wat ik uit ga voeren is veelomvattend en divers, waardoor het risico bestaat dat niet alle functionaliteiten gerealiseerd kunnen worden. Als oplossing wordt er gebruik gemaakt van de MoSCoW-methode, beschreven in hoofdstuk 4.2.2, zodat in overleg met de opdrachtgever alle eisen een prioriteit toegewezen krijgen. Dit zorgt ervoor dat de belangrijkste eisen als eerst gerealiseerd worden en de minder belangrijke eisen op zich kunnen laten wachten.

4.2 Methode en technieken

4.2.1 Ontwikkelmethodiek

Voor de keuze van een ontwikkelmethodiek is er een overweging gemaakt tussen SCRUM, RUP en XP (eXtreme Programming). De keuze, die uiteindelijk gevallen was op RUP, had te maken met een aantal factoren, namelijk:

1. De RUP methode biedt een uitstekende faseplan, wat er voor zorgt dat er een goede en gedetailleerde planning kan worden gemaakt.
2. Binnen de organisatie is er al enige ervaring wat betreft RUP, hierdoor is de begeleiding vanuit het bedrijf eenvoudiger.
3. RUP maakt gebruik van de UML tekentechniek. Dit is een voordeel omdat ik al kennis heb opgedaan van UML en RUP tijdens mijn schoolperiode. Daarnaast is UML een object georiënteerde tekentechniek, dit is belangrijk omdat de scoutlijst voor zowel de web applicatie als de mobiele applicaties in een object georiënteerde taal worden ontwikkeld.

XP zou daarentegen nog wel in aanmerking kunnen komen als ontwikkelmethodiek voor dit project. De scoutlijst moet namelijk voor drie platformen worden ontwikkeld, wat ervoor zorgt dat er veel tijd nodig is om te ontwikkelen. XP is uiteindelijk alsnog afgevallen door de geringe documentatie die het met zich mee brengt. De opgestelde competenties zijn daardoor lastig aan te tonen.

SCRUM is ook overwogen als ontwikkelmethodiek omdat er binnen Gamebasics gebruik van wordt gemaakt. Bij SCRUM wordt er aangeraden om over minimaal drie team leden te beschikken, omdat dit afstudeerproject individueel wordt uitgevoerd viel scrum al snel af.

4.2.2 MoSCoW

Om eisen een prioriteit te geven is er gekozen voor de MoSCoW-methode. MoSCoW is een acroniem waarvan de letters staan voor:

- **M**ust have - Deze eis *moet* in het eindresultaat terug te vinden zijn.
- **S**hould have - Deze eis is zeer gewenst, maar een vergelijkbare eigenschap is ook goed genoeg.
- **C**ould have - Deze eis mag alleen aan bod komen als het geen invloed heeft op andere delen van het project
- **W**on't have – Deze eis is gewenst voor later, maar valt buiten de scope voor dit project.

Door elke eis te koppelen aan één van deze prioriteiten kan er een duidelijk beeld worden geschetst van de belangrijkste eisen van het product.

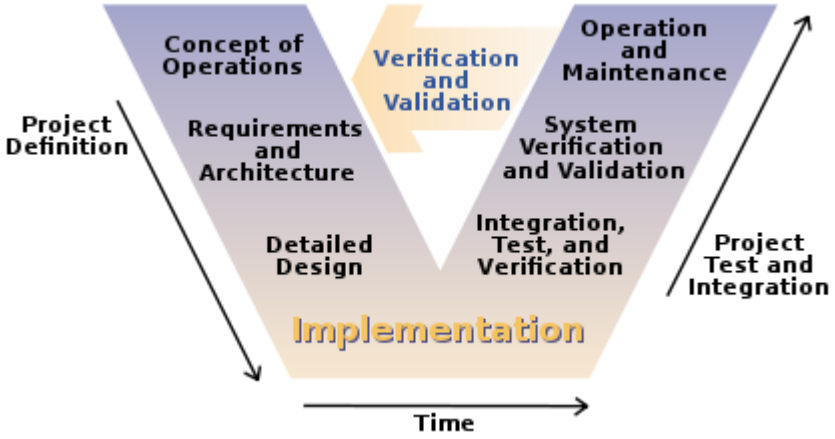
4.2.3 Git

Git is net als Subversion een versiebeheer methode voor de broncode. De keuze van Git boven Subversion is gemaakt omdat Gamebasics voor al zijn producten al gebruikt maakt van Git. Door het aanmaken van een aparte branch zal de development branch van Gamebasics geen last hebben van wijzigingen die doorgevoerd worden tijdens dit project. Zodra de broncode van dit project is goedgekeurd door één van de andere ontwikkelaars, zal de broncode van mijn product toegevoegd worden aan de development branch.

4.2.4 TestFrame

TestFrame is een gestructureerde testmethode waarmee ervoor gezorgd kan worden dat het deel dat op het kritieke pad zit, zo klein mogelijk wordt door de test al voor te bereiden voordat de applicatie wordt opgeleverd om te testen. Dit houdt in dat de testvoorbereiding en testautomatisering tegelijk plaatsvinden met de systeemontwikkeling.

Het V-model, getoond in figuur 4, zal centraal staan tijdens het testen van de applicatie. Zo zal de acceptatietest in een vroeg stadium beginnen met het statisch testen van de gebruikerswensen en functionele eisen.



Figuur 4 - Het V-model

Als alternatief voor testframe had ik ook TMap-next kunnen gebruiken. TMap is vooral gericht op testmanagement, waar TestFrame gaat over het voorbereiden en uitvoeren van testen. TMap is ontstaan vanuit de algemene testtheorie en is erg 'theoretisch' en houd zich dus op een afstand. TestFrame daarentegen gaat veel specifieker in op verschillende zaken.

In overleg met de tester van Gamebasics op basis van bovenstaande argumenten heb ik besloten om TestFrame als uiteindelijke testmethode te hanteren.

4.2.5 UML

Voor het modelleren van verschillende diagrammen is er gekozen voor UML, dit omdat binnen dit project met verschillende object georiënteerde talen gewerkt moet worden en UML is een object georiënteerde modellering techniek. Binnen Gamebasics is er een dot aan ervaring voor het modelleren van UML diagrammen, waardoor het voor mij eenvoudiger kan zijn om hulp te krijgen.

4.3 Activiteiten

Hieronder volgt een beschrijving van de activiteiten die ik zal uitvoeren tijdens het afstudeertraject. In deze activiteiten lijst is vooral het gebruik van RUP door middel van fases terug te vinden.

Plan van aanpak (Inception fase)
Aan het begin van het project stel ik een plan van aanpak op dat duidelijkheid verschaft over de opdracht tussen opdrachtgever en opdrachtnemer. Waarbij de uit te voeren activiteiten centraal zullen staan.

Requirements opstellen (Inception fase)
In overleg met de opdrachtgever worden er eisen opgesteld die gekoppeld worden aan een prioriteit van de MoSCoW-methode, zodat er duidelijkheid ontstaat over wat er geïmplementeerd moet worden en wat niet.

Ontwerp (Elaboration fase)
Tijdens de elaboration fase worden de vastgestelde eisen van de inception fase omgezet naar een ontwerp.

Prototype (Elaboration fase)
Nadat het ontwerp van de elaboration fase is afgerond, wordt er een prototype gemaakt om zo de grens te verleggen van wat technisch haalbaar is en inzicht krijgen in noden van gebruikers onder reële omstandigheden.

Construction iteratie één
De eerste iteratie van de construction fase zal bestaan uit het uitbreiden van de API functionaliteit en het ontwikkelen van het hoogst geprioriteerde applicatie. Tevens zullen er tijdens deze iteratie unittests geschreven worden.

Construction iteratie twee
In de tweede iteratie worden de twee lagere geprioriteerde applicaties ontwikkeld. Ook hier zullen unittests voor geschreven worden.

Acceptatie (Transition fase)
Na het ontwikkelen van de applicaties worden er test beschrijvingen en acceptatietests opgesteld waarmee stakeholders de applicaties kunnen goedkeuren.

4.4 Op te leveren producten

Dit project heeft een aantal op te leveren producten. Hieronder ziet u een overzicht van deze op te leveren producten, onderverdeeld in de vier RUP fases.

Inception fase

- Inception rapport

Elaboration fase

- Prototypes
- Elaboration rapport

Construction fase

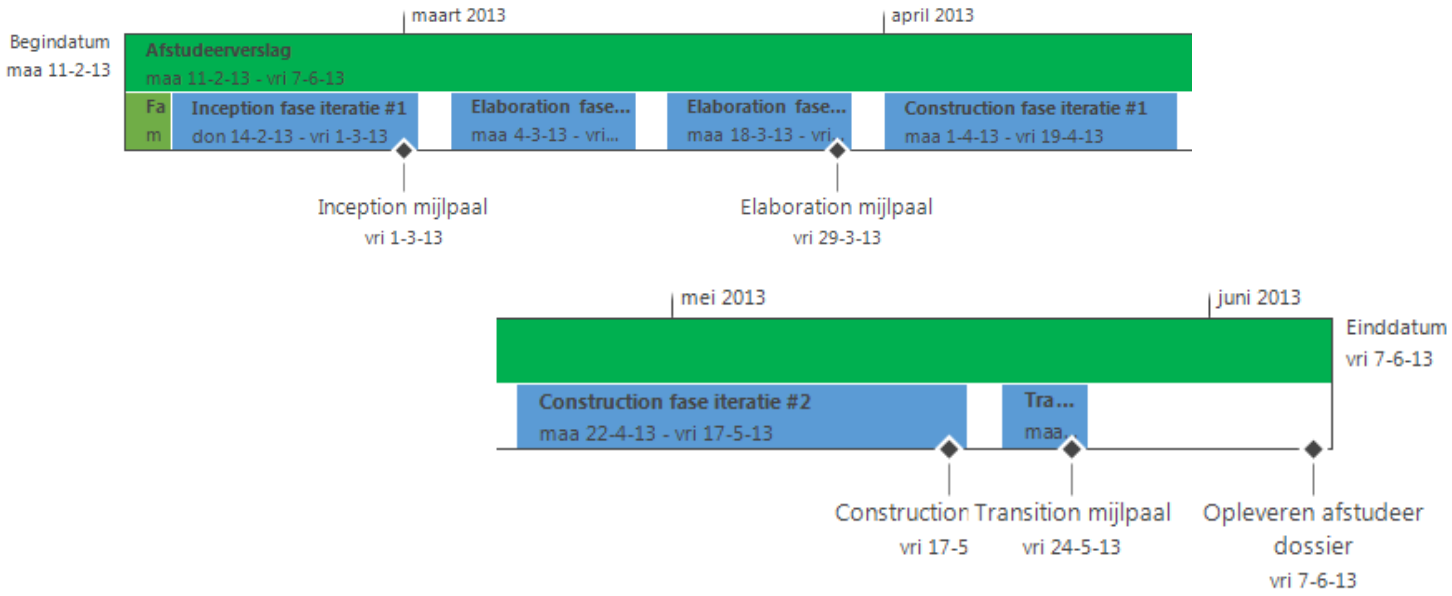
- Web applicatie
- iOS applicatie
- API aanpassing
- Android applicatie
- Construction rapport

Transition fase

- Acceptatietest
- Transition rapport

4.5 Planning

De planning die ik heb opgesteld is gebaseerd op een afstudeertraject van 85 dagen. De begindatum ligt op 11 Februari 2013 en de einddatum op 7 Juni 2013. Met behulp van het programma Microsoft Project geef ik hieronder een tijdlijn weergave van de planning weer.



Figuur 5 - Tijdlijn planning

Ik heb het totale project ingepland tot en met 24 mei 2013, terwijl het afstudeertraject loopt tot en met 7 Juni 2013. Dit heb ik gedaan om eventuele uitloop op te kunnen vangen en extra tijd op te nemen om mijn afstudeerdossier te verbeteren.

Ik heb er voor gekozen om de construction fase in twee iteraties op te splitsen, omdat ik de opdracht zie in twee onderdelen. Het eerste onderdeel is het ontwikkelen van de website en uitbreiden van de API. Het tweede onderdeel is het ontwikkelen van de twee mobiele applicaties die gebruik maken van de API uitbreidingen in iteratie één.

5 Inception fase

De inception fase is de eerste fase die doorlopen is. De inception fase is de begin fase van RUP waarbij analyses van functionaliteit en technische haalbaarheid van het systeem centraal staan. De nadruk zal met name op de requirements liggen.

5.1 Opstellen systeemeisen

Na goedkeuring van het plan van aanpak ben ik begonnen met het opstellen van de systeemeisen. Op het forum van Online Soccer Manager heb ik nieuwe thread aangemaakt over de door mij te ontwikkelen scoutlijst. In deze thread heb ik gevraagd om suggesties en ideeën voor de nieuwe scoutlijst aan te geven. Deze vraag kan alleen worden beantwoordt door spelers van Online Soccer Manager, een representatieve doelgroep. Suggesties van de spelers waren veelal hetzelfde, bijvoorbeeld 'kunnen filteren op spelers eigenschappen' en 'sorteren van de spelers op verschillende eigenschappen' waren vaak gesuggereerd.

Van de opgegeven ideeën en suggesties van ik vervolgens een lijst gemaakt met potentiele systeemeisen. In overleg met de opdrachtgever heb ik een aantal van deze suggesties meegenomen is het opstellen van de lijst met systeemeisen.

Bij het opstellen van de lijst zijn een aantal eigenschappen per systeemeis opgenomen. Omdat ik drie verschillende versies van de scoutlijst moet ontwikkelen (Web, Android en iOS) heb ik ervoor gekozen om een 'Platform' eigenschap bij elke systeemeis op te nemen, zodat ik voor elk onderdeel snel en eenvoudig inzicht kan krijgen in de eisen van het specifieke platform.

Verder zitten systeemeisen gekoppeld aan een MoSCoW prioriteit. Deze prioriteit is toegewezen in overleg met de opdrachtgever. De systeemeisen met hun gekoppelde MoSCoW prioriteit en overige eigenschappen zijn te vinden een de bijlagen (Bijlage 2 Inception rapport). De systeemeisen zijn uiteindelijk samengesteld aan de hand van de gebruikers hun wensen, de opdrachtgever zijn wensen en de wensen van de marketeer. Hieronder staan een aantal belangrijke systeemeisen.

5.1.1 Functionele eisen

Gebruikers moet de scoutlijst kunnen filteren op de volgende eigenschappen:

- Naam
- Competitie land
- Club
- Nationaliteit
- Waarde
- Positie
- Aanvalskwaliteit
- Verdedigingskwaliteit
- Gemiddelde kwaliteit
- Leeftijd

De scoutlijst moet beschikbaar zijn in de volgende talen: Engels, Duits, Deens, Indonesisch, Spaans, Frans, Italiaans, Hongaars, Nederlands, Pools, Portugees, Roemeens, Servisch, Zweeds, Turks, Russisch, Arabisch, Thais en Japans

Wanneer de gebruiker een speler geselecteerd wordt in de lijst, wordt er een pop-up getoond met extra informatie van de speler.

De prijs van een speler moet op basis van de op dat moment ingestelde taal de juiste valuta tonen.

5.1.2 Niet-functionele eisen

Alle spelers die scoutbaar zijn binnen Online Soccer Manager, moeten zichtbaar zijn op de scoutlijst.

De lay-out van de scoutlijst moet schalen op basis van de resolutie van de browser

Het design van de scoutlijst moet in de stijl van Online Soccer Manager

De mobiele scoutlijst applicaties moeten ontwikkeld worden in landscape mode

5.2 Use-cases

5.2.1 Vaststellen actoren

Door te bepalen welke objecten en gebruikers interactie hebben met het systeem, was het vrij eenvoudig om de actoren vast te stellen voor het systeem. Uiteindelijk blijkt er maar één actor, de gebruiker, interactie te hebben met het systeem.

5.2.2 Vaststellen use-cases

De volgende stap die ik heb genomen is het vaststellen van de use-cases. Omdat ik gebruik maak van RUP, en RUP een use-case driven methode is, is het vaststellen van correcte use-cases essentieel. Ik heb daarom ook veel aandacht besteed aan het definiëren van use-cases.

De bedoeling is dat de use-cases de functionele systeemeisen zoveel mogelijk dekken. Uiteindelijk ben ik in deze fase tot vijf use-cases gekomen. Deze use-cases heb ik bepaald aan de hand van de functionele systeem eisen.

- Scoutlijst sorteren
- Speler inzien
- Scoutlijst inzien
- Scoutlijst filteren
- Speler scouten

5.2.3 Ordenen use-cases

Door prioriteiten toe te kennen aan de use-cases kunnen er in een later stadium van het project prioriteiten worden gelegd indien er tijdsgebrek optreed, zodat er duidelijk is welke functionaliteit van het systeem prioriteit heeft.

Ik heb hierbij gekozen om de use-case ‘Scoutlijst inzien’ als belangrijkste use-case te bestempelen, omdat in deze use-case de meeste essentiële informatie getoond wordt. Als deze use-case niet wordt uitgevoerd, zal de rest van de functionaliteiten geen toegevoegde waarde hebben. ‘Scoutlijst filter’ en ‘Scoutlijst sorteren’ zijn de daarop volgende use-cases. De handelingen die in deze use-cases worden uitgevoerd zijn essentieel voor het nut van het systeem.

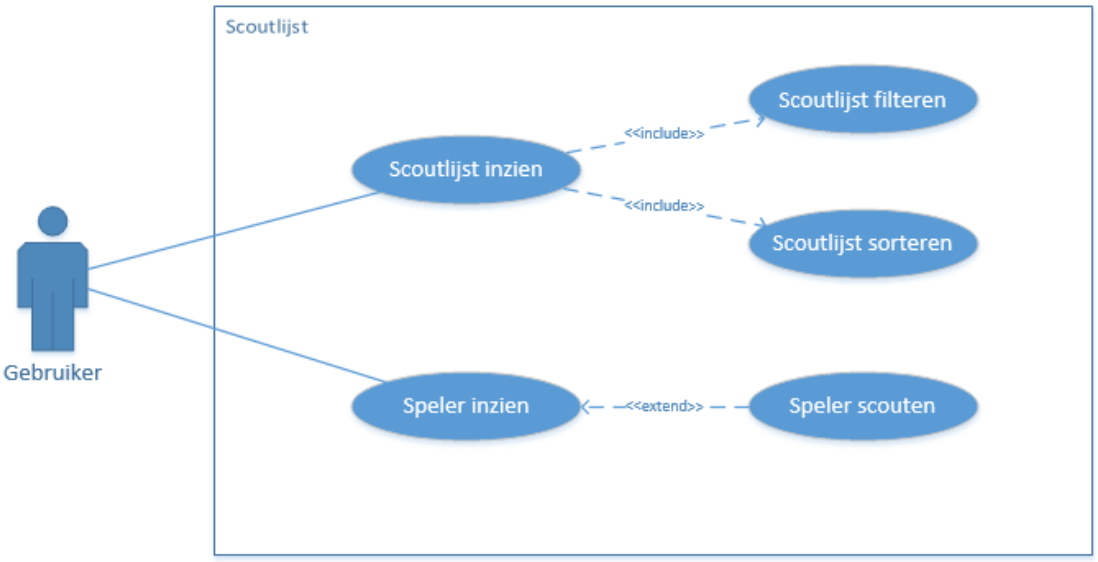
De volgende use-case op de prioriteitenlijst is ‘Speler inzien’. Het inzien van alle eigenschappen van een speler is belangrijk voor een gebruiker om alle gegevens eenvoudig en direct over te kunnen nemen in de scout functionaliteit van Online Soccer Manager. Als laatste op de lijst staat de use-case ‘Speler scouten’, omdat deze functionaliteit het minst essentieel is om tot het doel van de applicatie te komen.

Een overzicht van de geprioriteerde use-case:

1. Scoutlijst inzien
2. Scoutlijst filteren
3. Scoutlijst sorteren
4. Speler inzien
5. Speler scouten

5.2.4 Use-case model

Het doel van het use-case model is om een grafisch beeld te geven van de functionaliteiten van de scoutlijst. Het use-case model is opgesteld op basis van de vastgestelde use-cases en actor.



Figuur 6 - Het use-case model

5.2.5 Beschrijving use-cases

Nu de use-cases zijn geprioriteerd en het use-case model is opgesteld, kunnen er use-case beschrijvingen gemaakt worden. Per use-case heb ik een gedetailleerde beschrijving gemaakt zoals in tabel 1 weergegeven is.

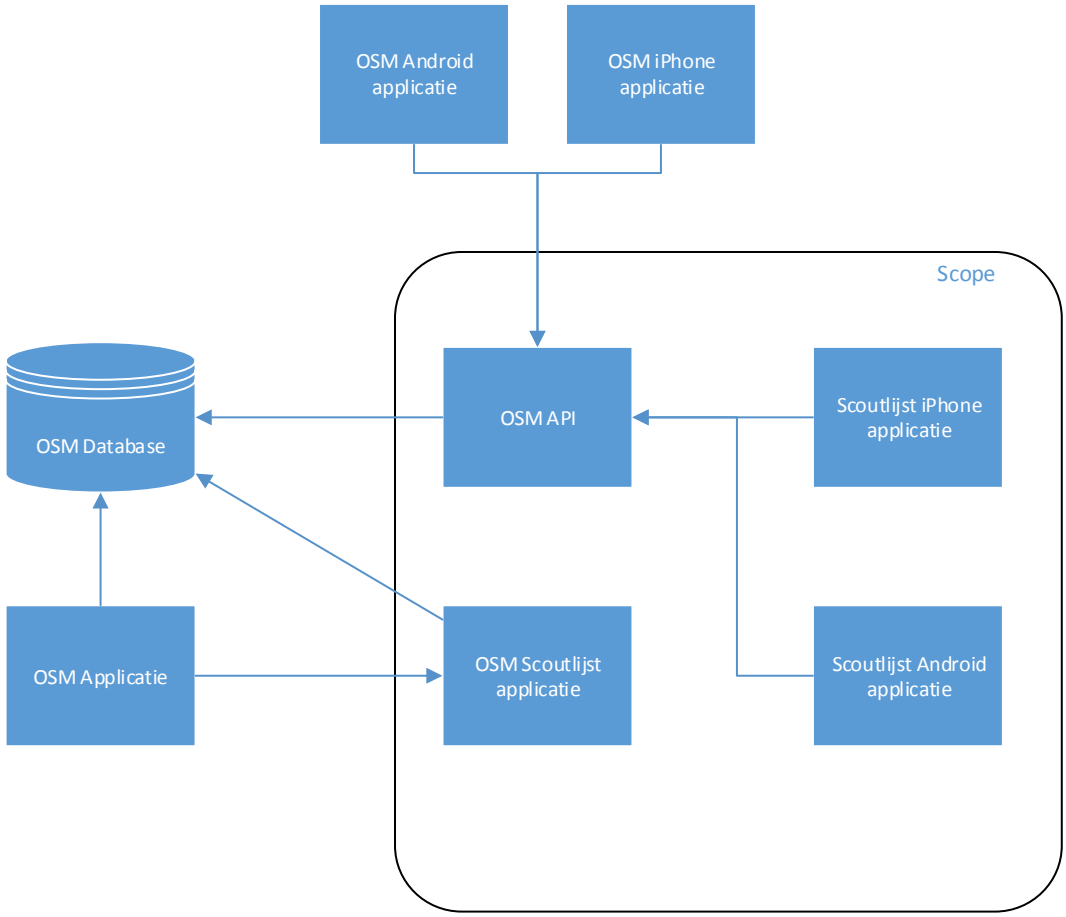
Naam	Scoutlijst inzien
Actor	Gebruiker
Samenvatting	De gebruiker
Aannames	Geen.
Beschrijving	(1) Gebruiker start de applicatie. (2) Systeem laat de lijst met spelers zien die gefilterd en gesorteerd kan worden. De uitzondering [Geen spelers gevonden.] treedt op wanneer er geen spelers gevonden zijn.
Uitzondering	[Geen spelers gevonden.] Het systeem geeft een melding en vraagt Gebruiker of hij/zij de filter instellingen wilt aanpassen. Use-case Scoutlijst filteren dient daarvoor uitgevoerd te worden.
Resultaat	Lijst met spelers wordt getoond op het scherm.

Tabel 1 - Use-case beschrijving

In totaal heb ik vijf use-case beschrijving gemaakt. Deze beschrijvingen bevatten het pad dat de actor doorloopt met de interactie van het systeem. Hierbij zijn ook de uitzonderingen die kunnen optreden genoteerd. De overige use-cases zijn terug de vinden in de bijlage (Bijlage: Inception rapport).

5.3 Scope

Nu de systeemeisen en use-cases opgesteld zijn, kon er een inschatting worden gemaakt over welke onderdelen binnen de scope van het project vallen en welke er buiten. Om dit weer te geven heb ik een model ontworpen die geen gebruik maakt van een vastgestelde techniek of methode. Ik wil hiermee een duidelijk beeld creëren hoe de nieuwe systemen gaan werken met de bestaande systemen van Gamebasics.



Figuur 7 - Opdracht scope

OSM Database - De OSM Database valt volledig buiten de scope van dit project, er staan geen wijzigingen op de planning. Indien er toch wijzigingen gedaan worden zullen hier aantekeningen van gemaakt worden.

OSM API - De bestaande functionaliteit van de API zal niet veranderen, wel zal er een toevoeging worden gedaan om de scoutlijst van de iPhone en Android te ondersteunen.

OSM Scoutlijst - De OSM Scoutlijst wordt volledig herschreven worden en valt binnen de scope van deze opdracht

Scoutlijst iPhone & Scoutlijst Android - De iPhone & Android scoutlijsten worden nieuwe, losstaande applicaties. Data verkrijgen ze via de OSM API.

OSM iPhone & OSM Android - De mobiele applicaties van Online Soccer Manager worden niet aangepast en vallen dus buiten de score van dit project.

5.4 Statisch testen

Voor het statisch testen van de eisen die beschreven staan in hoofdstuk 5.1, heb ik gebruik gemaakt van de techniek Perspective based reading (PBR). Een kenmerk van PBR is het toetsen van een bepaald product vanuit een specifiek oogpunt. Bijvoorbeeld als marketeer, ontwikkelaar, tester en opdrachtgever.

Motivatie gebruik PBR

Het testen van de eisen gebeurt op basis van een review techniek. Een aantal review technieken binnen TestFrame zijn, informele review, walkthrough, technische review en inspectie. Geen van deze technieken voldeed aan de eisen die ik stelde aan een review techniek. De review techniek moest vanuit verschillende oogpunten een aantal eisen kunnen beoordelen en moest gebaseerd zijn op een informele procedure.

De technische review techniek kwam het meest in de buurt om te gebruiken als review techniek. Jammer genoeg wordt een technische review alleen maar uitgevoerd door technische mensen, waardoor de opdrachtgever buiten de reviewers valt. Na verder onderzoek ben ik gaan kijken naar de PBR techniek, die perfect aansloot op mijn eisen van een review techniek. Uiteindelijk heb ik er dus voor gekozen om deze techniek te gaan gebruiken.

5.4.1 Perspective based reading

PBR is uitgevoerd om te controller of de systeemeisen elkaar niet in de weg zitten, of de systeemeisen koppen en om de eisen te valideren en verifiëren. Om deze punten te controleren heb ik de systeemeisen overhandigd aan verschillende personen binnen Gamebasics. Aanpassingen die gedaan moeten worden aan systeemeisen zijn in de elaboration fase doorgevoerd.

De opdrachtgever vond de lijst met verschillende filtermethoden goed, desalniettemin wilde de opdrachtgever het zoeken op spelersnaam loskoppelen van het filteren. De formaten van de banners moesten ook aangepast worden.

5.5 Vooronderzoek

Door de ruime planning van de inception fase in het plan van aanpak, had ik tijd over om een aantal vooronderzoeken te doen over met name MVC en Responsive Web Design. Ik heb ervoor gekozen om dit aan het eind van de inception fase te doen omdat eerst duidelijk moest worden wat in grote lijnen de functionele eisen zijn van de drie applicaties. Zodat er een duidelijk beeld gevormd kon worden welke technische kennis nodig is om de applicaties te kunnen implementeren.

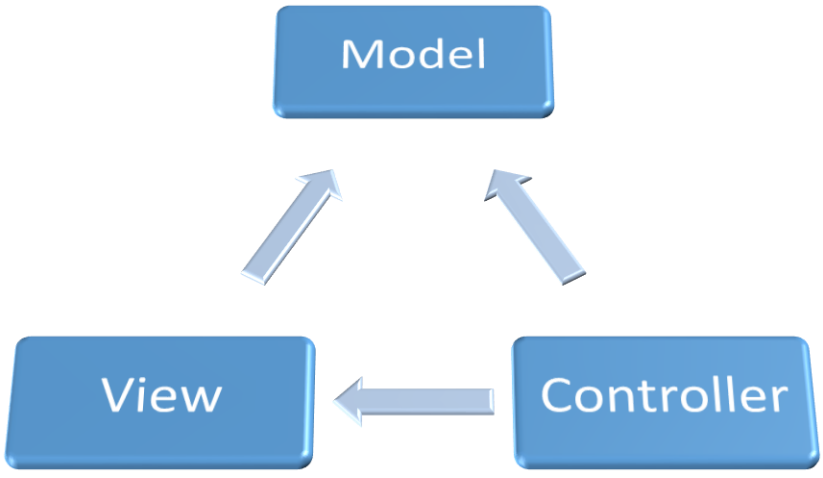
5.5.1 ASP.NET MVC

Eén van de systeemeisen van de scoutlijst was dat het gebouwd moest worden met behulp van de techniek ASP.NET MVC. Door de geringe kennis hiervan, heb ik er voor gekozen om tutorials te volgen. Op de officiële website van ASP.NET MVC staan verschillende tutorials om zo ermee bekend te raken.

MVC

Als eerst was het nodig om te begrijpen wat MVC nou precies inhoud. Het MVC pattern is een architectuur pattern waarbij de applicatie opgedeeld wordt in drie lagen met verschillende verantwoordelijkheden.

- Models (Data laag)
  Models zijn delen van een applicatie die de logica van het domein geïmplementeerd hebben. Vaak staan models het dichtstbij de database.
- Views (Presentatie laag)
  Een view is het component dat verantwoordelijk is voor de user interface. De view vraagt aan het model de data die getoond mag worden en geeft dit weer.
- Controllers (Applicatielogica laag)
  De controller bestaat uit zogenaamde actions die de user interactie afhandelt. Aan de hand van de action worden de juiste modellen en de juiste view opgespoord en gebruikt. De controller zorgt dus voor de communicatie tussen de models en de views.



Figuur 8 - MVC interactie

Het gebruik van MVC zorgt ervoor dat code overzichtelijker en beter te begrijpen is, omdat ieder gedeelte specifieke functionaliteiten bevat. Daarnaast is het mogelijk om code her te gebruiken, doordat er een onderscheid in lagen aanwezig is.

ViewData vs ViewModels

Er zijn twee manier wat betreft het versturen van data van de controller naar de bijbehorende view. ViewData is simpelweg een collectie waaraan waardes toegevoegd kunnen worden. ViewModels daarentegen zijn objecten waarover je zelf de controle hebt en kunnen daarom meerdere types tegelijk bevatten.

Uiteindelijk is de keuze in het voordeel van de ViewModels gevallen, omdat niet alle waardes los aan de collectie toegevoegd hoeven worden, zodat de controller acties overzichtelijker blijven. Ook geeft het veel meer vrijheid wat betreft het opbouwen van verschillende data.

Routing

Routing is een expressie waarin een formaat wordt opgeslagen van een URL waarmee er bepaald kan worden welke acties op welke controllers uitgevoerd moeten worden. In ASP.NET MVC is er standaard een route gedefinieerd die er als volgt uit ziet:

```
routes.MapRoute(
    "Default",
    "{controller}/{action}/{id}",
    new { controller = "Home", action = "Index", id = "" }
);
```

De standaard route zorgt er voor dat bijvoorbeeld http://scoutlijst.com/Home/Index/28, door wordt gestuurd naar de actie ‘Index’ van controller ‘Home’ met als parameter 28. Aan de hand van de parameters kan bijvoorbeeld een bepaalde speler opgehaald en getoond worden.

Bundling and minification

Bundling is een techniek binnen ASP.NET MVC om de laadtijd van een pagina te optimaliseren. Het verbetert de laadtijd door het aantal aanvragen van CSS en JavaScript te verlagen. Bundling maakt het eenvoudig om meerdere bestanden in één bestand om te zetten, zodat deze als enige ingeladen hoeft te worden [1]. Dit zorgt voor minder aanvragen en dus een snellere laadtijd.

```
bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
            "~/Content/themes/base/jquery.ui.core.css",
            "~/Content/themes/base/jquery.ui.resizable.css",
            "~/Content/themes/base/jquery.ui.selectable.css",
            "~/Content/themes/base/jquery.ui.accordion.css",
            "~/Content/themes/base/jquery.ui.autocomplete.css",
            "~/Content/themes/base/jquery.ui.button.css",
            "~/Content/themes/base/jquery.ui.dialog.css",
            "~/Content/themes/base/jquery.ui.slider.css",
            "~/Content/themes/base/jquery.ui.tabs.css",
            "~/Content/themes/base/jquery.ui.datepicker.css",
            "~/Content/themes/base/jquery.ui.progressbar.css",
            "~/Content/themes/base/jquery.ui.theme.css"));
```

Het figuur hierboven laat zien hoe meerdere CSS bestanden toegevoegd worden aan één bundel. Hierdoor hoeft er nog maar één regel toegevoegd te worden in de view, om al deze bestanden in de bundel in de view te laden.

```
@Styles.Render("~/Content/themes/base/css", "~/Content/css")
```

Omdat de scoutlijst web applicatie gebaseerd wordt op responsive web design, is het van belang dat er rekening wordt gehouden met de performance op mobiele apparaten. Bundling zou dus van grote waarde kunnen zijn tijdens het ontwikkelen van de scoutlijst.

Naast het bundelen van CSS een JavaScript bestanden kan er ook minification worden toegepast. Minification doet verschillende code optimalisaties aan CSS en JavaScript, zoals het weghalen van onnodige witregels en commentaar en het inkorten van variabel namen. Een stuk JavaScript kan er bijvoorbeeld als volgt uit zien:

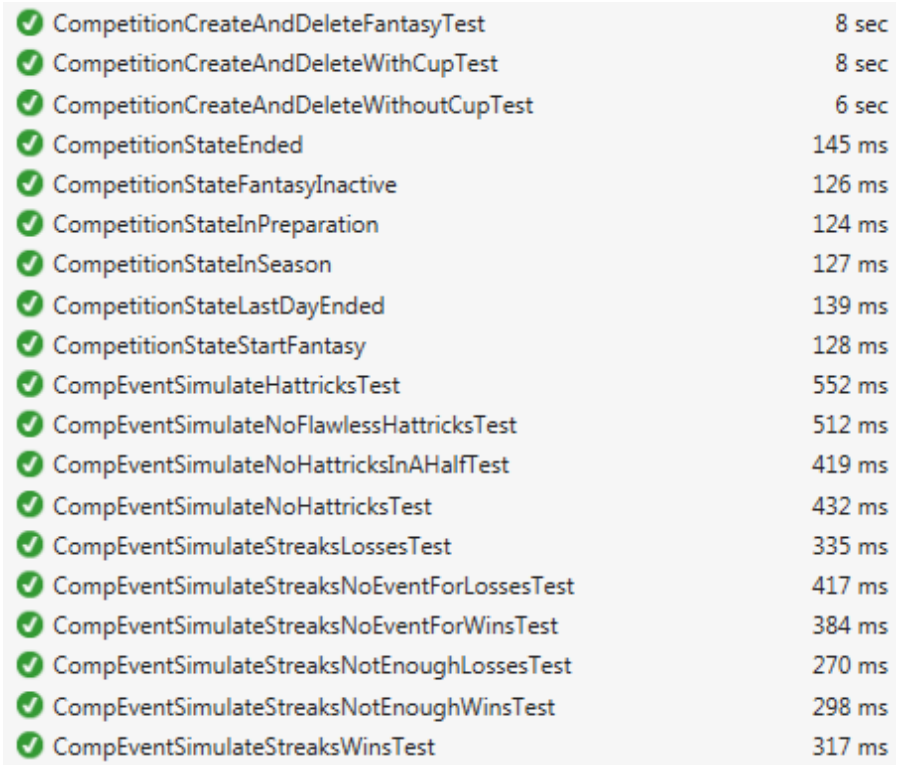
```
AddAltToImg = function (imageTagAndImageID, imageContext) {
    ///<signature>
    ///<summary> Adds an alt tab to the image
    // </summary>
    //<param name="imgElement" type="String">The image selector.</param>
    //<param name="ContextForImage" type="String">The image context.</param>
    ///</signature>
    var imageElement = $(imageTagAndImageID, imageContext);
    imageElement.attr('alt', imageElement.attr('id').replace(/ID/, ''));
}
```

Nadat de minification is toegepast ziet de JavaScript er als volgt uit:

```
AddAltToImg = function (n, t) { var i = $(n, t); i.attr("alt",
i.attr("id").replace(/ID/, "")) }
```

Unittest

Visual Studio biedt de mogelijkheid om snel, eenvoudig en uitgebreide unittests te schrijven, waarbij de resultaten in een duidelijk overzicht getoond worden. Omdat Gamebasics al eerder unittests heeft geschreven, beschikken ze over een test-database waarbij de data altijd opnieuw geinitialiseerd wordt op het moment dat de unittests gerund worden. Dit voorkomt dat de data veranderd en unittests van een eerdere versie omvallen.



Figuur 9 - Unittest overzicht Visual Studio

Het bovenstaande figuur geeft een overzicht van bestaande unittests binnen het Online Soccer Manager project. Wanneer de unittests gestart worden, worden deze één voor één uitgevoerd. Hierbij wordt aan de linker kant een groen icoon weergegeven zodra de unittest geslaagd is en wordt er een rood icoon weergegeven zodra de unittest gefaald is. Aan het einde van de regel staat hoeveel tijd er is verstreken bij het uitvoeren van de betreffende unittest.

5.5.2 Responsive web design

Responsive web design is een ‘nieuwe’ manier van html pagina’s ontwikkelen. Het voordeel ervan is dat webpagina’s op verschillende formaten schermen aangepast worden zodat deze op een zo goed mogelijk manier gepresenteerd worden. Voorbeelden van verschillende resoluties zijn: de desktop computers, tablets en smartphones.

Tegenwoordig voldoet een aparte mobiele website van een applicatie niet meer. En aangezien één van de eisen was om een mobiele versie van de applicatie te maken, heb ik overwogen om gebruik te maken van responsive web design, waarbij er, zoals hierboven beschreven, gebruik wordt gemaakt van één html bestand per pagina.

De content staat centraal met responsive web design, er hoeft geen keuze meer gemaakt te worden welke informatie er gedeeld moet worden en met wie. Responsive web design is daarom ook een voordeel van de online marketeers van Gamebasics. Er wordt gebruik gemaakt van één unieke URL en alles is overzichtelijk in één Google Analytics profiel.

Tijdens dit vooronderzoek ben ik tot de conclusie gekomen dat responsive web design uitermate geschikt is om de scoutlijst web applicatie mee op te bouwen. Ten eerste is het gebruik maken van responsive web design heel efficiënt, er wordt per pagina maar één html bestand gemaakt waardoor ik veel tijd winst kan boeken. Ten tweede is er geen op browser gebaseerde re-direct nodig, dit zorgt voor een kortere ladingstijd van de pagina [3]. Ook zijn op browser gebaseerde re-directs gevoelig voor fouten, er zijn namelijk veel verschillende browser op mobiele platformen beschikbaar.

Figuur 10 - Responsive web design voorbeeld

In figuur 10 wordt een voorbeeld getoond van een applicatie die gebaseerd is op een responsive web design. Op alle toestellen die te zien zijn, is dezelfde URL geopend en wordt er een versie getoond die geoptimaliseerd is voor de specifieke resolutie.

5.5.3 HTML5 versus native

Het onderzoek naar één volledig functionerende HTML5 applicatie die zowel werkt op Android als op iOS versus twee aparte native applicaties heb ik net als het vorige vooronderzoek uitgevoerd in de inception fase, dit was mogelijk doordat er een ruime planning van de inception fase opgesteld was.

Native

Native applicaties bieden de beste bruikbaarheid, de beste eigenschappen en de beste mobiele ervaring [2]. Er zijn een aantal dingen die alleen beschikbaar zijn voor native applicaties:

- Multi touch - Dubbel klik, zoom en andere UI gebaren zijn beschikbaar voor native applicaties.
- Snelle grafische weergave - Het native platform geeft een uitermate soepele grafische weergave.
- Ingebouwde componenten - De camera, adresboek, locatiebepaling en meer functionaliteiten kunnen moeiteloos worden gebruikt in native applicaties.
- Vertrouwd - Het native platform is wat men gewend is, wanneer er gebruik word gemaakt van een native functionaliteit, zal dit bekend voorkomen bij gebruikers.

Native applicaties zijn over het algemeen complexer om te ontwikkelen. Het ervaringsniveau van een ontwikkelaar moet hoger zijn dan bij andere scenario's. Zo kan een ontwikkelaar niet simpelweg Objective-C kopiëren-plakken en verwachten dat het werkt.

Vanaf de eind gebruiker gezien zitten er een aantal voordelen aan native applicaties. Wanneer een native applicatie gestart wordt, start deze onmiddellijk, krijgt de gebruiker snelle prestaties en ziet de gebruiker een consistente interface wat zorgt voor een verhoogde gebruiksvriendelijkheid.

HTML5

Een mobiele HTML5 applicatie bestaat in feite uit één of meerdere webpagina's die zijn gebouwd om op kleine schermen te werken. HTML5 applicaties kunnen op elke moderne mobiele browser geopend worden en omdat het om een webpagina's gaat, is het mogelijk om via zoek engines de applicaties te vinden.

Een belangrijk punt van HTML5 applicaties is het 'on-the-fly' update. Wanneer er aanpassingen of toevoegingen worden gedaan aan de applicatie, zullen alle gebruikers de aanpassing krijgen zonder dat ze er iets voor hoeven doen.

Een voordeel van een HTML5 applicatie is dat de ontwikkelervaring van een ontwikkelaar niet zo hoog hoeft te zijn als bij native applicaties. Voor een nieuwe ontwikkelaar zou het dus makkelijker zijn om met een HTML5 applicatie te beginnen.

Conclusie

Gezien het feit dat de applicatie voorzien moet worden van veel data waar snel en eenvoudig op gefilterd moet worden met behulp van native functionaliteit, liggen twee aparte native applicaties voor de hand. Door de ervaring van het ontwikkelen van native applicaties binnen Gamebasics, maakt deze keuze alleen maar makkelijk.

Notificaties worden niet ondersteund door HTML5 applicaties zoals in tabel 2 te zien is. Op ten duur wil Gamebasics eventueel notificaties inbouwen om gebruikers in te lichten over een nieuwe spelerslijst. Dit is dus één van de criteria waar de applicatie aan moet voldoen. Daar komt nog bij dat de Online Soccer Manager applicatie ook als native applicatie is gebouwd, en deze applicatie aangesproken moet kunnen worden vanuit de scoutlijst.

	Native	HTML5
Applicatie functionaliteiten		
Grafisch	Native API	HTML
Prestaties (snelheid)	Snel!	Langzaam
Interface	Native	Nagebootst
Uitgave	App store	Web
Native toegang		
Camera	Ja	Ja, door gebruik van library
Notificaties	Ja	Nee
Contacten	Ja	Ja, door gebruik van library
GPS	Ja	Ja, door gebruik van library
Ontwikkeltaal	Objective-C, Java	HTML5, CSS, Javascript
Connectie	Online en offline	Online

Tabel 2 - Native vs HTML5

6 Elaboration fase

De tweede fase van de RUP methode is de elaboration fase. Tijdens de inception fase heb ik de opdracht vastgesteld en op een globaal niveau uitgewerkt, zodat ik in de elaboration fase het grootste gedeelte van het systeem kan beschrijven. Als eerst worden de bestaande eisen en use-cases nogmaals bekeken een aangepast indien nodig om vervolgens op basis van deze eisen mockups en ontwerpen ga maken.

6.1 Aanpassingen systeemeisen

In het overleg dat ik had gevoerd aan het einde van de inception fase met de opdrachtgever en marketeers, werden mij nog een aantal aanvullende wensen en eisen duidelijk. Deze elaboration fase geeft de mogelijkheid om aanpassingen door te voeren. Onderstaand de nieuwe, aangepaste en verwijderen eisen.

Nieuwe systeemeis

Op de website moet paginering worden toegepast indien er meer dan vijftien spelers in de lijst getoond moeten worden.

Nieuwe systeemeis

De scoutlijst moet de volgende eigenschappen van een speler tonen in portrait modus

- Naam
- Leeftijd
- Waarde
- Basis kwaliteit

Nieuwe systeemeis

De scoutlijst moet de volgende eigenschappen van een speler tonen in landscape modus

- Naam
- Leeftijd
- Waarde
- Aanvallende kwaliteit
- Club
- Verdedigende kwaliteit

Nieuwe systeemeis

De mobiele applicaties van de scoutlijst moeten ondersteuning krijgen voor zowel landscape modus als portrait modus.

Nieuwe systeemeis

Voor gebruikers moet het mogelijk zijn om op spelersnaam te zoeken.

Aanpassing systeemeis

Gebruikers moet de scoutlijst kunnen filteren op de volgende eigenschappen:

- ~~Naam~~
- Positie
- Competitie land
- Aanvalskwaliteit
- Club
- Verdedigingskwaliteit
- Nationaliteit
- Gemiddelde kwaliteit
- Waarde
- Leeftijd

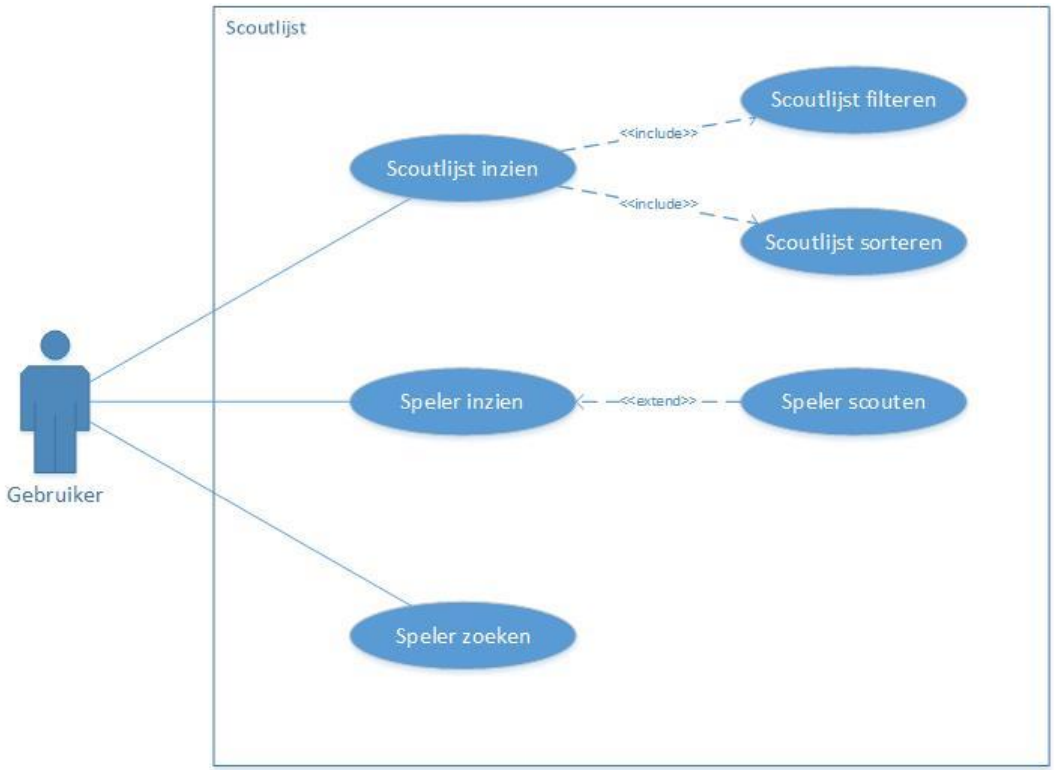
Aanpassing systeemeis

~~De scoutlijst moet voorzien worden van reclame~~ De scoutlijst lijst moet voorzien worden van de volgende formaten reclame banners: twee keer 336x280 en één keer 728x90.

6.2 Aanpassingen use-cases

In het overleg met de stakeholders, kwam naar voren dat het filteren op spelersnaam losgekoppeld moest worden van het filteren van spelers. Ik vond dit erg handig, omdat er dan op elk moment gezocht kan worden op naam, ook als de lijst al gefilterd is op verschillende eigenschappen. In de systeemeisen is deze aanpassing al te vinden en zal dus ook doorgevoerd worden in de use-cases.

Het nieuwe use-case model ziet er dan als volgt uit:



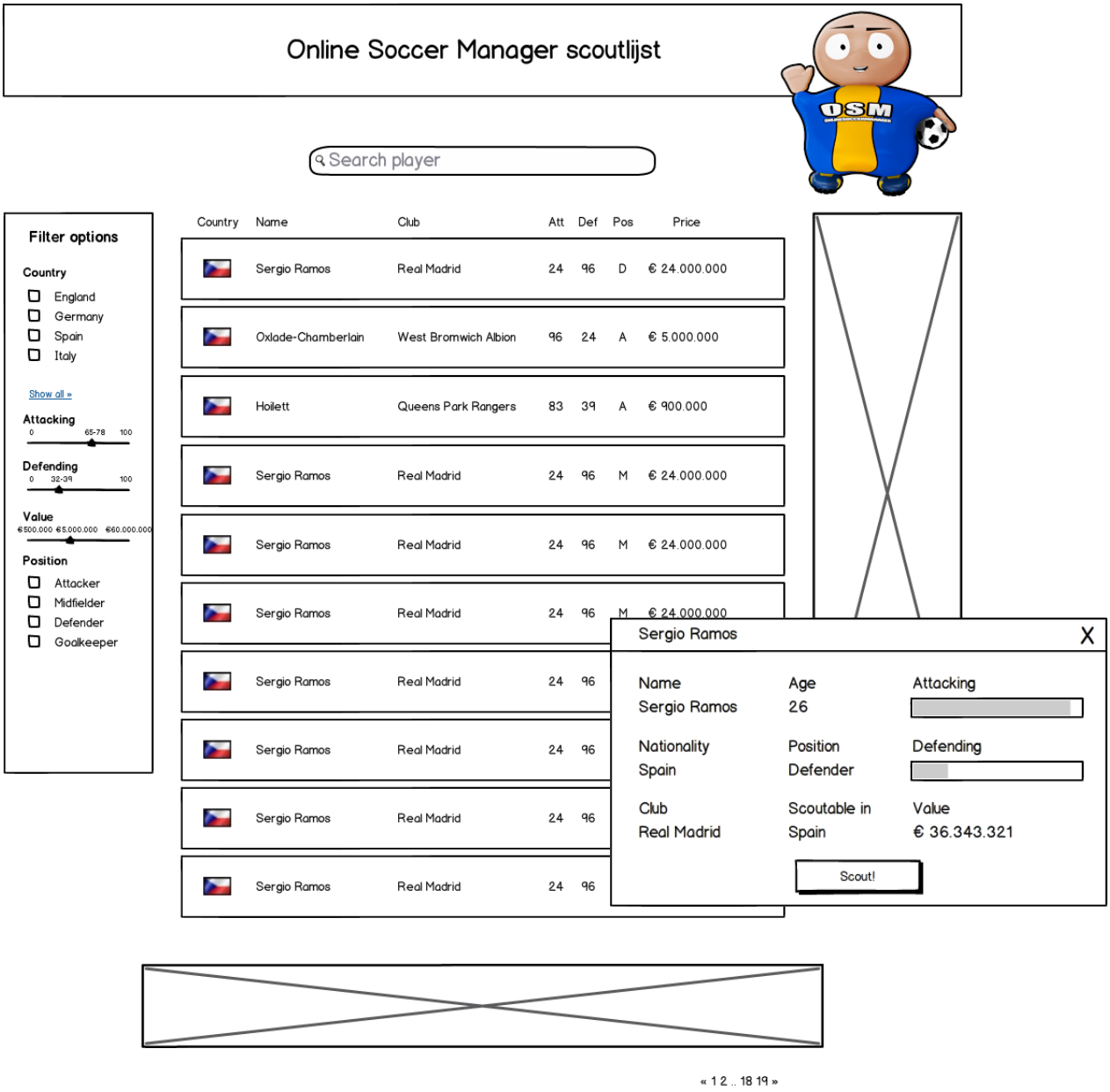
Figuur 11 - Use-case model

6.3 Mockup

Om mijn idee van de scoutlijst over te brengen op de designers van Gamebasics heb ik besloten om tijdens de elaboration fase mockups te maken. De mockups zorgen ervoor dat de designers weten welke informatie waar moet komen voordat ze daadwerkelijk het design opbouwen. Ik heb de mockups voor de web applicatie en de mobiele applicaties gescheiden gehouden.

6.3.1 Web mockup

De mockup voor de web applicatie ziet er als volgt uit:



Figuur 12 - Mockup website

Omdat er veel gebruik gaat maken van het filter heb ik ervoor gekozen om het filter altijd te tonen en niet een uitklapbare variant te maken. De rechthoeken met de kruizen erin staan er ter indicatie van de reclame banners die op de site moeten komen. Tijdens het ontwerpen van de mockup is er rekening gehouden met de systeemeisen, zo komt de informatie die wordt laten zien op het scherm overeen met de informatie die voorgeschreven wordt in de systeemeisen.

6.3.2 Mobiele mockup

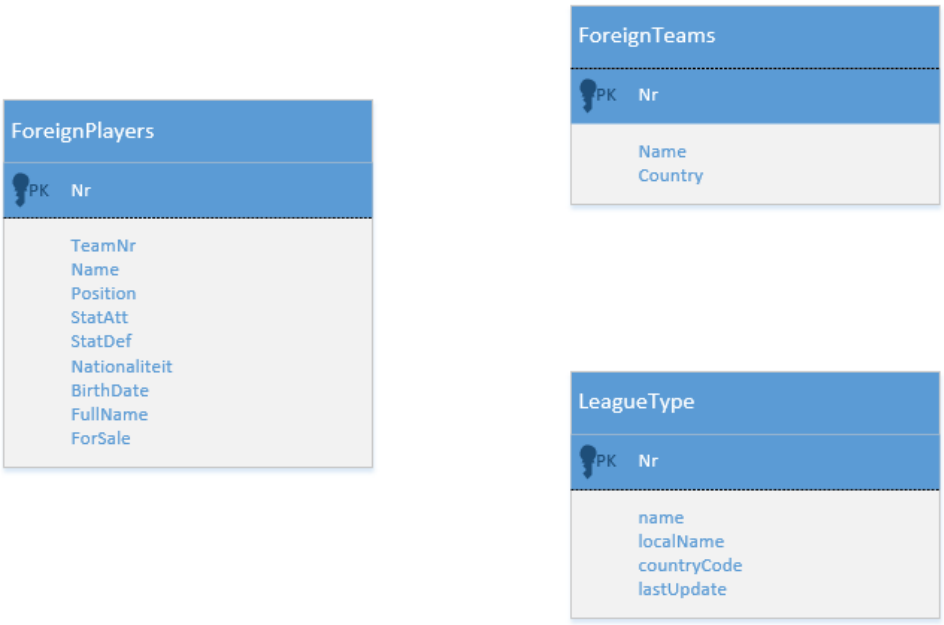
De mockups voor de mobiele versies van de scoutlijst zien er als volgt uit:

Figuur 13 - Mockups mobiel

Er is veel ruimte beschikbaar gemaakt voor de spelerslijst. Het idee is dat het zoekveld verdwijnt zodra er naar beneden gescrold wordt, zodat er nog meer ruimte komt voor de spelerslijst. Het filteren en sorteren wordt gecombineerd in één pagina, zodat de gebruiker altijd weet waar hij of zij heen moet.

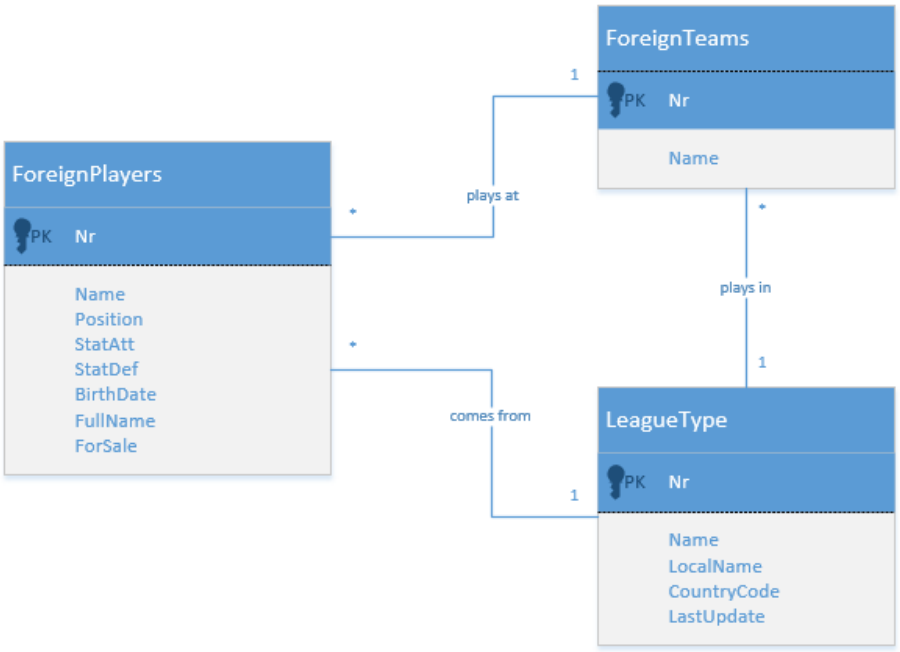
6.4 Entity Relationship Diagram

Tijdens de elaboration fase is er gekeken naar de huidige situatie van de database structuur. De vraag was of de huidige structuur voldeed om data aan te leveren voor drie verschillende applicaties. Om op deze vraag antwoord te geven heb ik een ERD gemaakt voor de huidige situatie. Deze is te zien in figuur 14.



Figuur 14 - Huidige database structuur

De scoutlijst maakt gebruik van dezelfde database als Online Soccer Manager. Omdat Online Soccer Manager lang geleden is gebouwd, draait het nog met oude tabellen zonder vreemde sleutels. In eerste instantie schrok ik hier van, omdat ik mijzelf geen database kan voorstellen zonder vreemde sleutels. Desalniettemin heb ik een ERD opgesteld die ik graag had willen doorvoeren tijdens mijn opdracht. Dit diagram is te vinden in figuur 15.



Figuur 15 - Nieuwe database structuur

Op basis van de nieuwe database structuur is er een overleg geweest, hierbij is er gediscussieerd wat deze wijziging teweeg zal brengen. Tijdens dit overleg kwam naar voren dat er teveel aanpassingen gedaan moeten worden aan Online Soccer Manager, wat buiten mijn scope valt. Ik heb er dus uiteindelijk voor gekozen om deze wijziging niet door te voeren, alhoewel ik dat erg jammer vond.

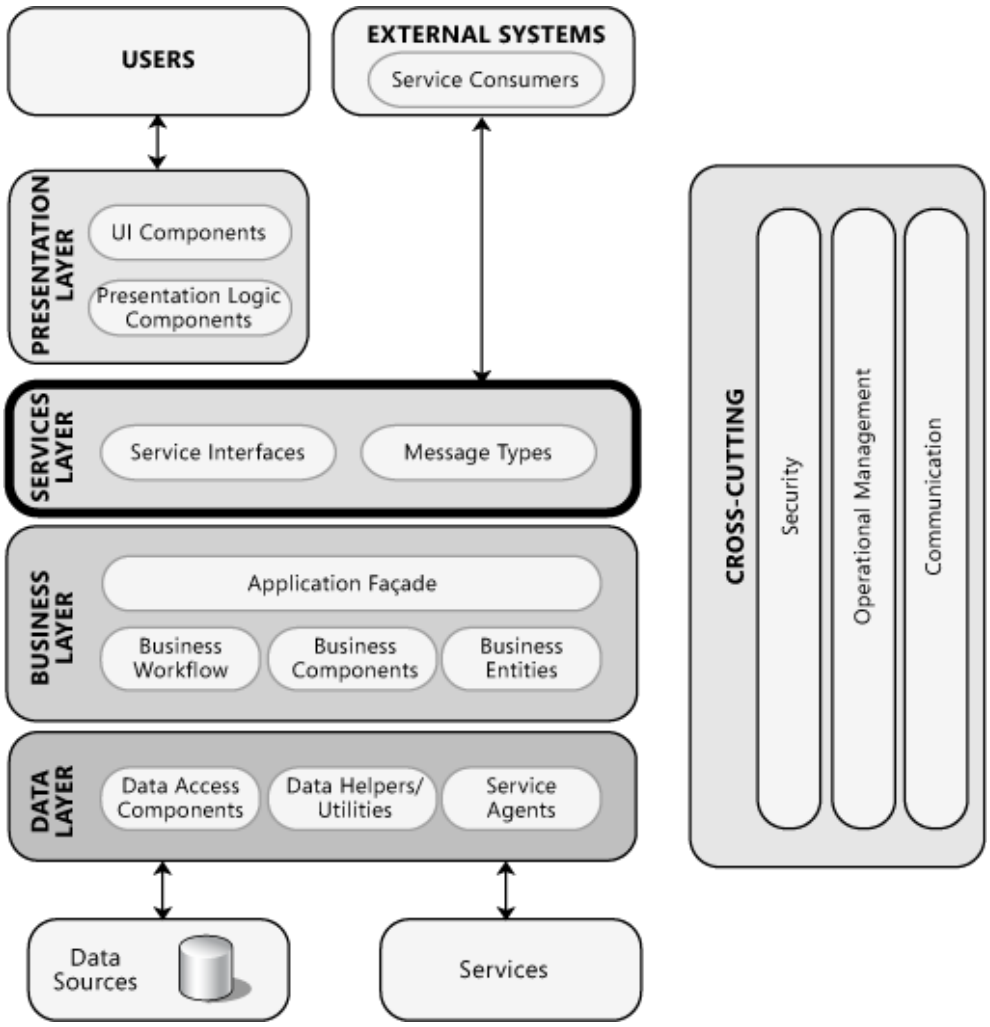
6.5 Ontwerp

Omdat er binnen Gamebasics gebruik wordt gemaakt van een centraal data project om data aan te leveren voor alle producten, moest er een oplossing voor de koppeling van de scoutlijst bedacht worden.

6.5.1 Service Layer

In een vroeger stadium van het Online Soccer Manager project had Gamebasics besloten om gebruik te maken van een zogenaamde Service Layer. Een principe dat afgeleid is uit de Application Architecture Guide van Microsoft [4]. In figuur 16 is te zien hoe verschillende lagen verschillende verantwoordelijkheden hebben. De Service Layer die Gamebasics gebruikt voor het centrale data project is een combinatie van de Business Layer en de Service Layer, wat ertoe leidt dat de Service Layer Business logica bevat. De Service laag is tussen de Controllers en de data laag(Repositories en Models) opgenomen. De verantwoordelijkheden van de verschillende lagen zijn dus als volgt:

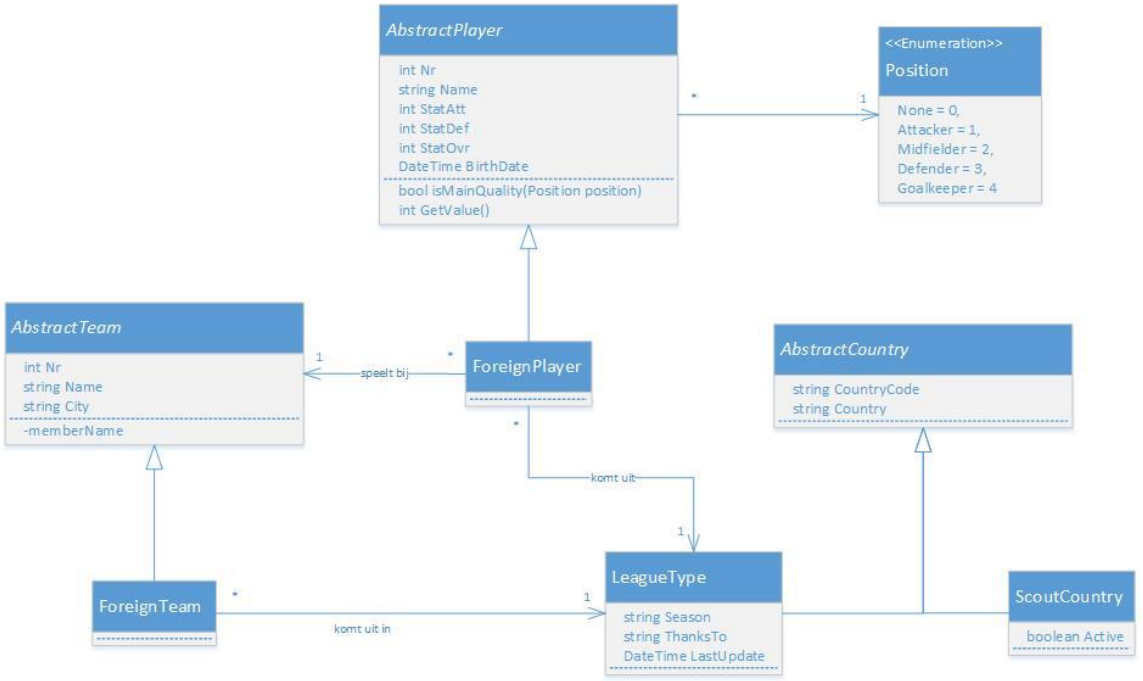
- Controllers - User interactie
- Service Layer - Business logica
- Repository - Database interactie



Figuur 16 - Microsoft Architect Guide voorbeeld

6.5.2 Ontwerp web applicatie

De klassen die benodigd zijn om de scoutlijst te kunnen realiseren, bestaan uit: Player, Team, LeagueType en Position. Om deze klassen te realiseren is er naar het bestaande data project gekeken. In het data project bleken al abstracte klassen aanwezig te zijn waarop ik de klassen van de scoutlijst aan kon sluiten. In figuur 17 is te zien hoe dit is ontworpen.



Figuur 17 - Web klassediagram

Er is voor gekozen om de klassen van de scoutlijst te koppelen aan de al bestaande abstracte klassen omdat deze al grotendeels beschikken over de benodigde attributen. Het zou zonde zijn geweest om nieuwe klassen aan te maken en de attributen nogmaals te definiëren. Ik heb ervoor gekozen om de Service en Repository layer uit dit model te laten, omdat deze lagen aanwezig zijn voor alle concrete klassen.

De models die gedefinieerd zijn in dit klassendiagram zijn niet direct aan elkaar gekoppeld maar worden gekoppeld met behulp van datamodels. Gamebasics heeft sinds de overstap naar ASP.NET MVC gebruik gemaakt hiervan. Datamodels zijn er dus om koppelingen te leggen tussen bepaalde models. Om dit klassendiagram als voorbeeld te nemen; het ForeignPlayer model heeft alleen een verwijzing naar het nummer van een Team. De daadwerkelijk koppeling wordt gemaakt in het ForeignPlayerTeamDataModel. Hierin zijn de twee models ForeignPlayer en Team opgenomen.

Het ontwerpen van de Controllers van de web applicatie heb ik op basis van de eerder opgestelde use-cases gedaan, daarnaast heb ik ook naar de huidige structuur gekeken hoe zij de Controllers opgezet hebben. Uiteindelijk kwam ik tot de conclusie dat ik maar één controller nodig had voor de functionaliteiten van de scoutlijst web applicatie. Deze is hieronder beschreven inclusief zijn bijbehorende acties.

- Player
    - Overview
    - PlayerList
    - PlayerDetails

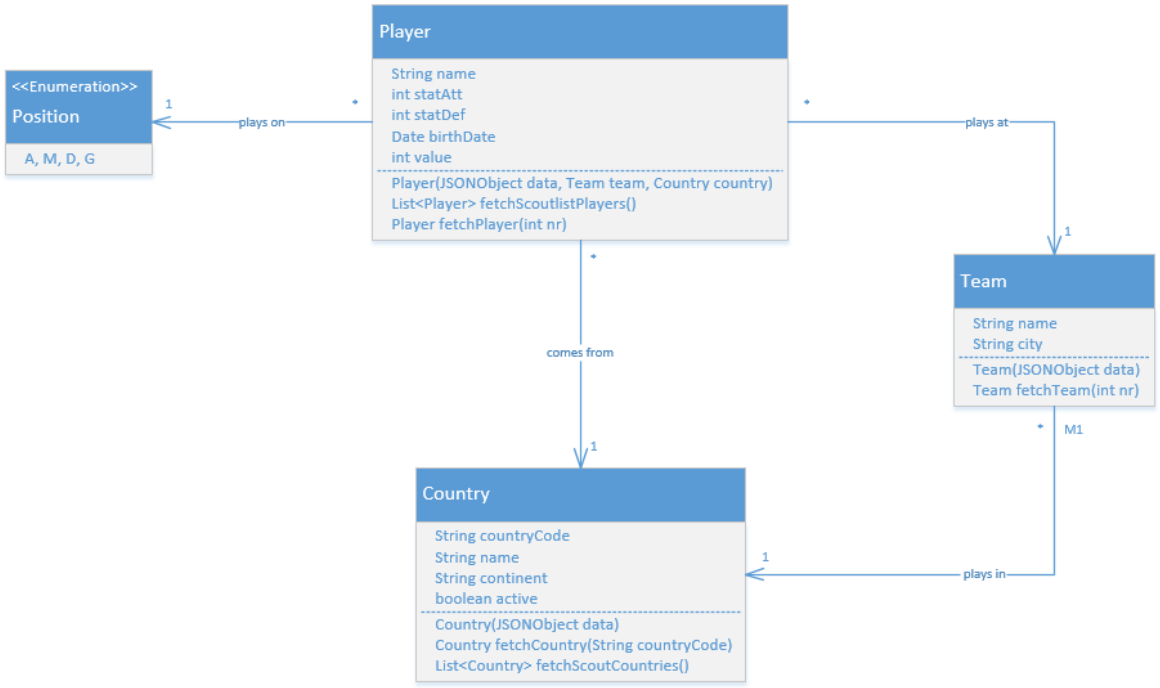
Om de Controllers van de API te ontwerpen heb ik gekeken naar de huidige API controllers. Hierbij kwam naar voren dat zij verschillende 'fetch' methodes gebruiken om lijsten van bepaalde Models op te halen. Ik heb er voor gekozen om voor de spelers data die ik nodig heb ook fetch methodes aan te maken in de Controllers van de Models. Uiteindelijk ziet het er als volgt uit:

- Player
  - FetchPlayers
- Team
  - FetchTeams
- Country
  - FetchScoutCountries
  - FetchLeagueTypes

Ik heb ervoor gekozen om van alle data die ik nodig heb losse API requests te maken, om vervolgens de koppeling te kunnen leggen in de applicatie die de API response ontvangt. Dit heb ik gedaan omdat een team of een country dan maar één keer opgehaald hoeft te worden. Als alternatief had ik kunnen kiezen om één grote API request te maken, dit heb ik dus niet gedaan omdat er meerdere spelers hetzelfde team hebben, waardoor één bepaald team vaker opgehaald wordt. Dit gaat ten koste van de grote van de API request.

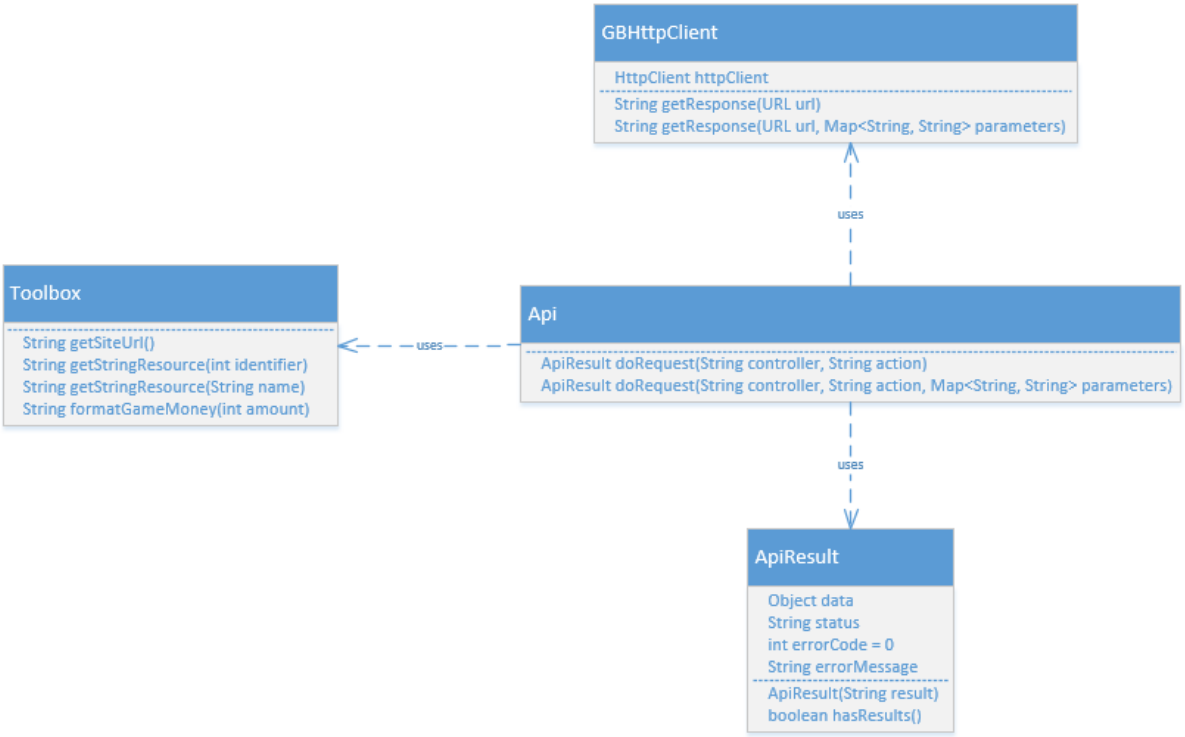
6.5.3 Ontwerp mobiele applicaties

Omdat er voor het mobiele platform twee compleet nieuwe applicaties worden ontwikkeld, kan er geen gebruik worden gemaakt van al bestaande klassen. Het domein model ziet er dus als volgt uit:



Figuur 18 - Mobiel klassediagram

Om een API request te doen op één van de mobiele platformen heb ik een aantal klassen ontworpen die allemaal hun eigen verantwoordelijkheden hebben. Ik heb hiervoor een klassendiagram gemaakt om een beter beeld te geven welke klassen wat doet.



Figuur 19 - Klassendiagram API request

Toolbox - Verantwoordelijk voor kleine zaken binnen de applicatie. De toolbox zorgt ervoor dat de juiste URL wordt gebruik in verschillende API request en dat de waarde van spelers op de juiste wijze getoond wordt.

API - Via de API klasse kunnen er verschillende API request gedaan worden. Met behulp van de doRequest methode wordt er op basis van een controller, action en eventuele parameter een ApiResult teruggegeven.

ApiResult - De ApiResult is een klasse die de binnenkomende JSON afhandelt. Wanneer er een API request gedaan is kan hierin de status teruggevonden worden en de complete set met data indien aanwezig.

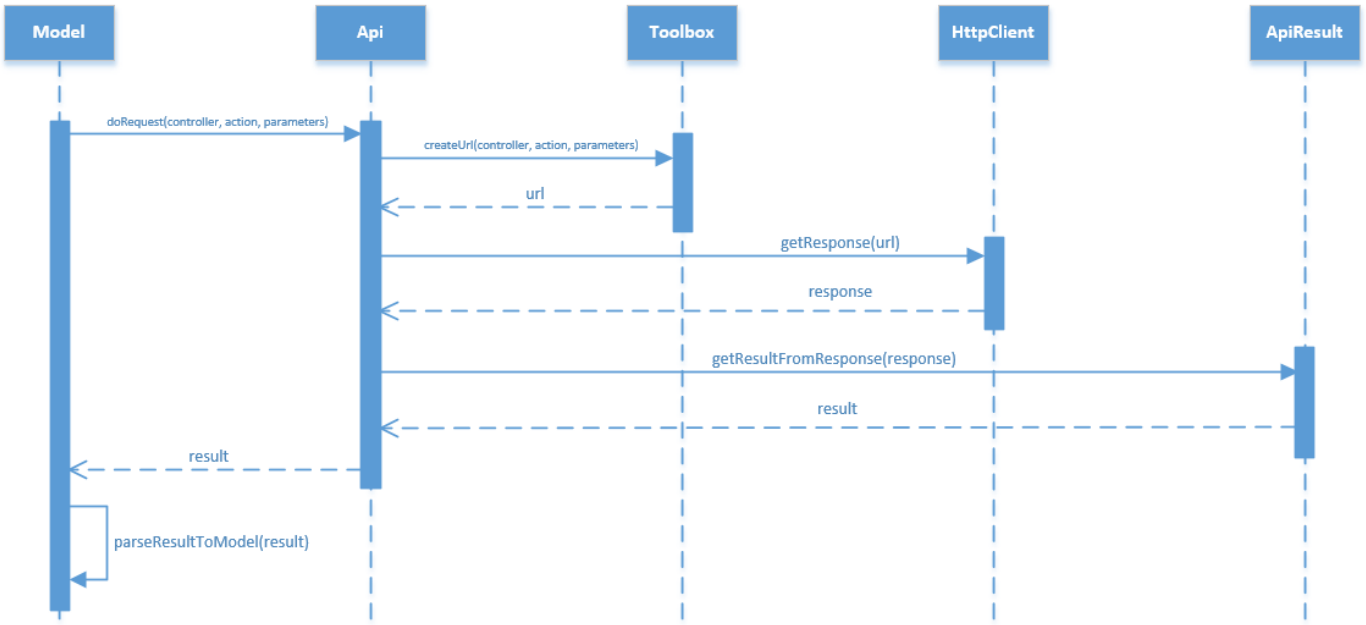
GBHttpClient - De GBHttpClient is een wrapper om de HttpClient van Apache. Hierin worden de daadwerkelijke requests gedaan.

6.6 Sequentiediagram

De volgende stap die ik heb genomen is het opstellen van sequentiediagrammen. Een sequentiediagram geeft een mogelijk scenario weer. Ik vind een sequentiediagram zeer handig bij het helder krijgen van interactie tussen de objecten en klassen. Mijn bedoeling was om de belangrijkste scenario's weer te geven. De belangrijkste scenario's voor de scoutlijst applicaties zijn: Een API request doen en spelers inladen op basis van een ingesteld filter. Deze scenario's worden hieronder weergegeven.

6.6.1 API request

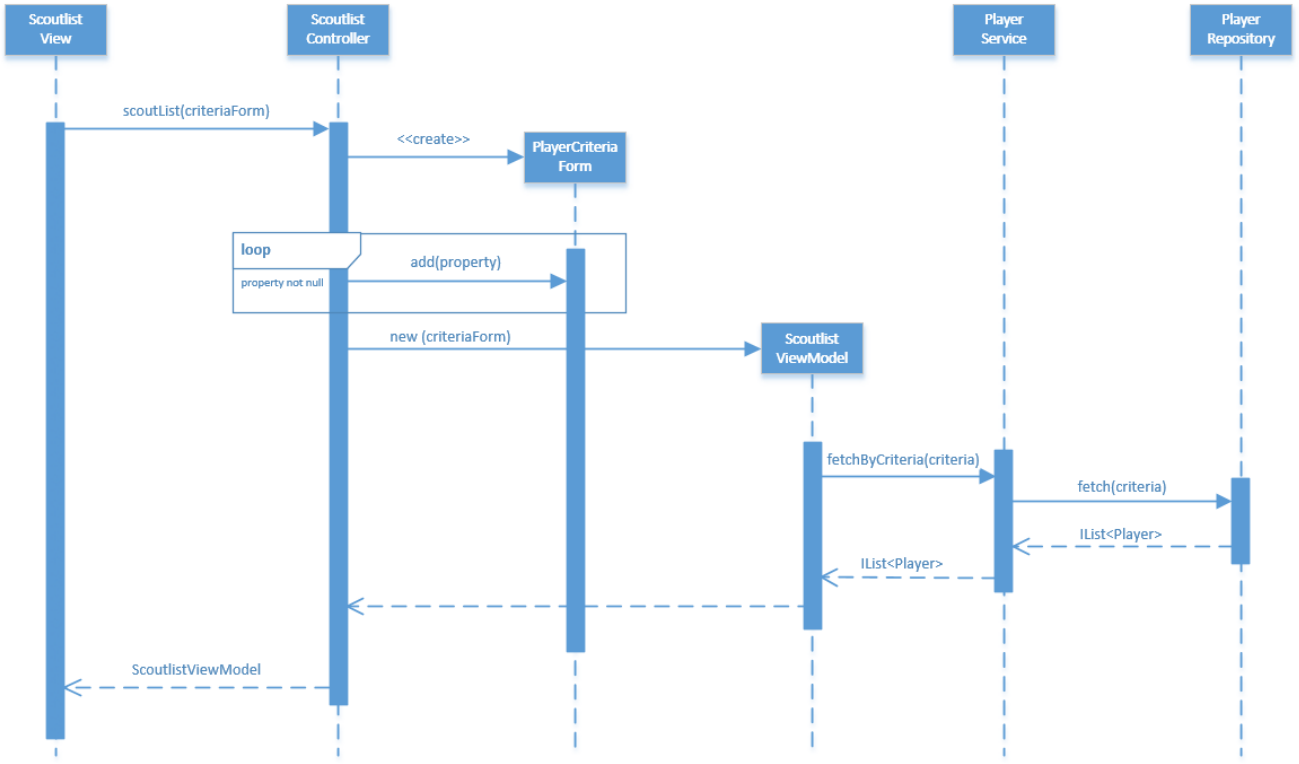
Het sequentiediagram van een API request ziet er als volgt uit:



Figuur 20 – Sequentiediagram API request

1. Model doet een request op de API met drie parameters.
2. Vanuit de API wordt de toolbox aangeroepen om de URL op te bouwen op basis van de taal en parameters. De Toolbox geeft een URL terug.
3. Vervolgens roept de API de getResponse methode aan in de HttpClient. Een response wordt terug gegeven.
4. De ApiResult zorgt er voor dat de response wordt omgezet in een resultaat waar de applicatie mee kan werken.
5. Het model parsed het resultaat om naar een model die gebruikt kan worden voor de applicatie.

6.6.2 Spelers inladen



Figuur 21 - Sequentiediagram spelers inladen

1. Het filter van de scoutlijst wordt aangepast door de gebruiker, de scoutlist view wordt daarbij opnieuw opgebouwd
2. De scoutlist view zorgt ervoor dat de scoutList methode van de controller aangeroepen wordt met de bijbehorende parameter.
3. Binnen de controller wordt de juiste viewmodel gebruikt.
4. De viewmodel zorgt voor alle data op de view en haalt dus alle spelers op via de service.
5. De service vraagt de repository om de beschikbare spelers en filtert deze op basis van de meegegeven criteria.

6.7 Prototyping

Aan het einde van de elaboration fase is er gewerkt aan verschillende prototypes. Het is gebruikelijk dat aan het einde van de elaboration fase een werkend prototype wordt opgeleverd. Ik heb de prototypes vooral gebruikt om te kijken of het project technisch haalbaar is en of het ontwerp goed werkt. Tijdens het ontwikkelen van de prototypes is er gewerkt aan de volgende functionaliteiten:

- Met behulp van AJAX een lijst met spelers ophalen aan de hand van een kleine set criteria zonder dat de pagina ververst.
- Een API request doen vanuit het Android platform.
- Een eenvoudige pagina die gebruik maakt van responsive web design.

Er is gekozen voor deze prototypes omdat dit de kern functionaliteiten van de scoutlijst zijn. Prototyping voor het iOS platform is niet gedaan omdat ik niet de beschikking over een Mac had.

6.7.1 Git

Door mijn geringe ervaring met Git heb ik mijzelf, tijdens het ontwikkelen van de prototypes, ook verdiept in deze versiebeheer methode. Nadat ik een eigen branch had gecreëerd en een aantal functionaliteiten had gecommit, moest de development branch in mijn eigen gecreëerde branch samengevoegd worden. Dit had te maken met de aanpassing van de berekening van de prijs van een speler. Omdat ik veel bestanden uit mijn branch verwijderd had, kreeg ik een aantal conflicten. Omdat ik, zoals eerder beschreven, weinig ervaring had met Git, wist ik niet hoe ik hiermee om moest gaan, waardoor ik een aantal wijzigingen ben kwijt geraakt. Vervolgens had Gijs, mijn begeleider, een uitleg gegeven over het mergen in Git, zodat dit in de toekomst niet meer kon gebeuren. Achteraf ben ik blij dat dit tijdens het ontwikkelen van het prototype gebeurde en niet tijdens het ontwikkelen van de daadwerkelijke applicatie. Uiteraard zou het beter zijn geweest als het helemaal niet was voorgekomen.

6.7.2 Responsive web design

Responsive web design kan worden toegepast met behulp van media queries. Media queries zijn nieuw in CSS3 en worden gebruikt om styling opties voor een specifieke resolutie uit te voeren. De media queries die ik gebruikt heb tijdens het ontwikkelen van de prototypes zijn:

```
@media all and (min-width: 300px) {
    body {
        background-color: blue;
    }
}

@media all and (min-width: 500px) {
    body {
        background-color: beige;
    }
}

@media all and (min-width: 800px) {
    body {
        background-color: yellow;
    }
}
```

De achtergrond kleur wordt geel wanneer het scherm wordt verkleind tot onder de 800 pixel, onder de 500 pixel wordt de kleur beige en de kleur wordt blauw zodra het scherm kleiner is dan 300 pixel. De media queries kunnen zo aangepast worden dat er voor iedere resolutie aangepaste CSS

meegestuurd kan worden. Hiermee kan er dus content verborgen worden als het scherm kleiner wordt.

6.7.3 Spelers ophalen

Voor het ophalen van spelers met behulp van AJAX, moest ik een Controller actie maken met een bijbehorende View. Om de View van data te voorzien maakte ik gebruik van een ViewModel, zoals beschreven in hoofdstuk 5.5.1. Binnen het ViewModel wordt de spelerslijst geladen door gebruik de maken van de PlayerService. In de PlayerService wordt de opgehaalde lijst gefilterd aan de hand van de meegegeven criteria. Hieronder een voorbeeld hoe de filtering toegepast wordt op de lijst.

```
public IList<ForeignPlayerTeamDataModel> FetchByForm(PlayerCriteriaForm form)
{
	return _foreignPlayerRepo.FetchAll()
	.Where(x => x.ForeignPlayer.StatAtt <= form.MaxStatAtt &&
	x.ForeignPlayer.StatAtt >= form.MinStatAtt)
	.Where(x => x.ForeignPlayer.StatDef <= form.MaxStatDef &&
	x.ForeignPlayer.StatDef >= form.MinStatDef).ToList();
}
```

De PlayerRepository wordt aangeroepen om de lijst met spelers op te halen, waarna er met behulp van LINQ statements wordt gefilterd op de meegegeven criteria. In hoofdstuk 7.2.2 leg ik uit waarom er vanuit de service altijd de volledige lijst met spelers uit de repository haalt.

6.7.4 API Request

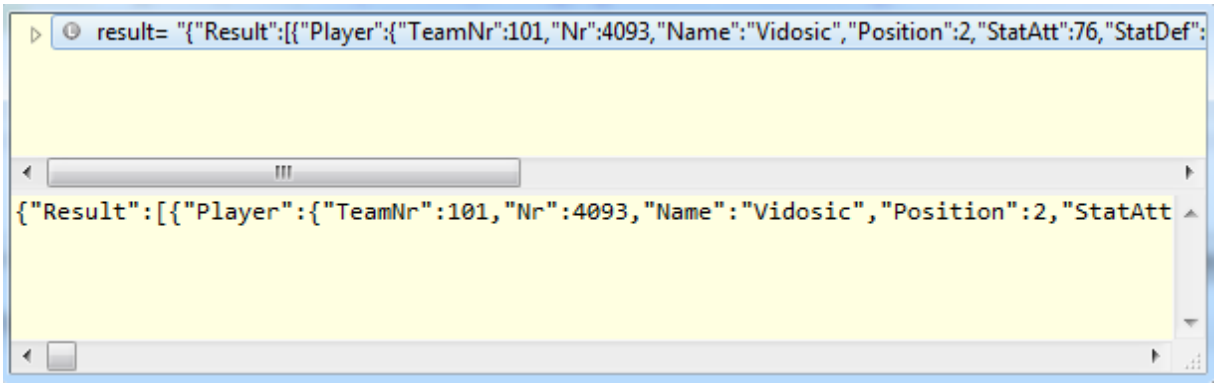
Om een API request te doen op een mobiel platform, heb ik gebruik gemaakt van de HttpClient van Apache. Ik heb deze HttpClient gebruikt om het prototype eenvoudig te houden, wellicht kan er in de construction fase nog een keuze worden gemaakt tussen verschillende HttpClient's.

Om een request te doen heb ik een getResponse methode toegevoegd een het resultaat van een URL omzet in één String. De getResponse methode ziet er als volgt uit:

```
public String getResponse(String apiUrl) {
	URL url;
	HttpResponse httpResponse;
	String response = "";
	try {
		url = new URL(apiUrl);
		HttpRequestBase request = new HttpPost(url.toString());
		httpResponse = httpClient.execute(request);
		HttpEntity entity = httpResponse.getEntity();
		if (entity != null) {
			InputStream instream = entity.getContent();
			Header encoding = entity.getContentEncoding();
			if (encoding != null && encoding.getValue().equals ("gzip")) {
				instream = new GZIPInputStream(instream);
			}
			BufferedReader in;
			in = new BufferedReader(new InputStreamReader(instream));
			String inputLine;
			StringBuilder responseBuilder = new StringBuilder();
			while ((inputLine = in.readLine()) != null)
				responseBuilder.append(inputLine);
			in.close();
			instream.close();
			response = responseBuilder.toString();
			entity.consumeContent();
			return response;
```

```
            }
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (NullPointerException e) {
            e.printStackTrace();
        }
        return "failed";
}
```

Om deze getResponse methode te gebruiken, hoeft er alleen maar een URL van de API meegegeven te worden. Wanneer de getResponse is uitgevoerd en wordt opgevangen in een String object, kan ik, als ik de applicatie debug, het resultaat terug zien in Eclipse. In figuur 22 is de zien dat het String object 'result' is gevuld met een JSON string.



Figuur 22 - API request resultaat

6.8 Testen

De elaboration fase zat er bijna op, maar voordat de volgende fase aan bod komt, moeten de aangepaste requirements en RUP modellen nog statisch getest worden met behulp van Perspective Based Reading.

Het statisch testen van de requirements en RUP modellen is op dezelfde manier gedaan als in de inception fase. Het elaboration rapport is uit verschillende oogpunten bekeken; een marketeer, een ontwikkelaar en de opdrachtgever. Tijdens een feedback moment gaven ze allen te kennen dat het elaboration rapport goed is.

Ook heb ik tijdens de elaboration fase een testplan opgesteld. In dit testplan staat beschreven wanneer unittests geschreven worden, welke testtechniek ik gebruik om de systeemtest uit te voeren en wanneer de acceptatietest wordt uitgevoerd.

Om de systeemtesten uit te voeren heb ik gekozen voor de minder gestructureerde 'Exploratory testing' techniek. Het voordeel hiervan is dat het weinig voorbereidingstijd kost, er hoeft alleen maar aangegeven te worden hoeveel tijd aan welk onderdeel besteed moet worden. Deze techniek heeft wel een nadeel, fouten zouden eventueel niet altijd gereproduceerd kunnen worden. Dit nadeel is deels te verwerpen doordat de drie applicaties vrij klein zijn en uit weinig onderdelen bestaat.

In het testplan heb ik ook een testverslag opgenomen, daarin kan verslag worden gelegd van de uitgevoerde tests en de gevonden incidenten.

7 Construction fase

Nadat de elaboration fase was afgerond ben ik verder gegaan met de construction fase uit RUP. De construction fase bestaat uit, zoals eerder beschreven, twee iteraties. In dit hoofdstuk beschrijf ik hoe ik deze iteraties ben doorlopen.

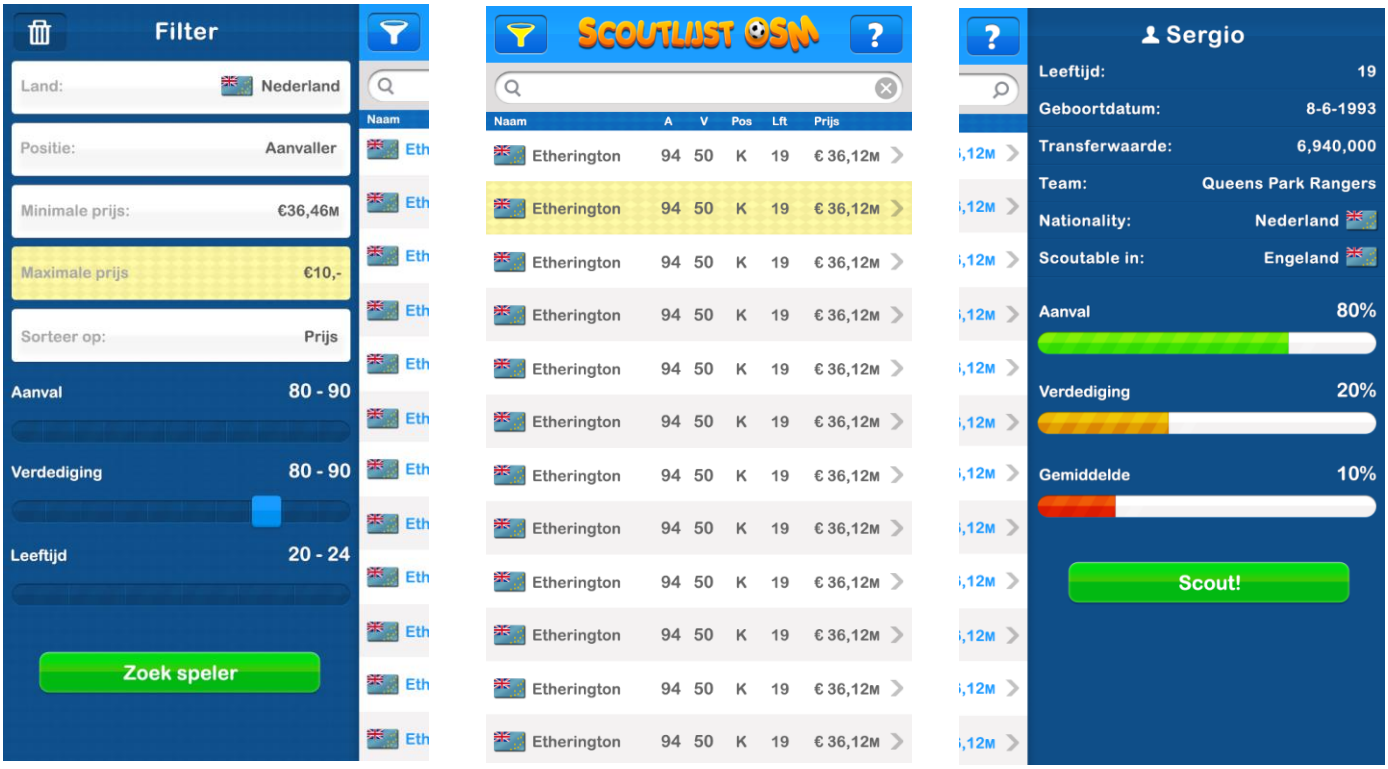
7.1 Ontwerp Scoutlijst

Aan het eind van de elaboration fase heb ik de mockups overhandigd aan de designers van Gamebasics. Zij zouden, zodra ik begon met ontwikkelen, het design van de scoutlijst applicaties aanleveren. Aan de hand van de mockups hebben zij het onderstaande design aangeleverd voor de webapplicatie.

Figuur 23 - Scoutlijst web design

Het design is door de designer verbetert ten opzichte van de mockups. Het filter is bovenaan de pagina geplaatst en de zoek balk dichter bij de lijst geplaatst. Naar mijn mening was de header van de tabel te ver weg geplaatst van de tabel zelf. Ik heb daarom de keuze gemaakt om de advertentie balk boven de header te plaatsen, uiteraard na goedkeuring van de designers.

Er is een apart design gemaakt voor de mobiele variant van de scoutlijst, omdat dit losstaande applicaties zijn. Het design van de mobiele applicaties is hieronder te vinden.



Figuur 24 - Scoutlijst mobiel design

Het design komt bijna overeen met de mockup die ikzelf had ontworpen. Alle knoppen en items zijn iets meer uitvergroot om het te optimaliseren voor het aanraken van het scherm.

7.2 Iteratie 1: Het ontwikkelen van de webapplicatie en uitbreiden van de API

Tijdens deze iteratie is er als eerst gewerkt aan het ontwikkelen van de webapplicatie, omdat de API functionaliteiten pas nodig zijn zodra de webapplicatie af is en er een begin is gemaakt aan de tweede iteratie.

7.2.1 Responsive web design

Om responsive web design toe te passen op de web applicatie, dienen er een aantal keuzes te worden gemaakt. Zo moet er bekeken worden welke items er eventueel zouden kunnen wegvallen op de pagina en moet er een keuze worden gemaakt welke resoluties ondersteund moeten worden.

Uit overleg met de stakeholders bleek dat zij de advertenties en uitleg niet belangrijk vonden in mobiele browsers, omdat dit alleen maar ruimte in beslag neemt op een klein scherm. Zodoende is er voor gekozen om deze items te verbergen wanneer de pagina wordt geopend op een scherm die kleiner is dan 800 pixels.

De tabel kan een aantal spelers eigenschappen verbergen als het scherm wordt verkleind. Zo zijn de spelers eigenschappen ‘Team’ en ‘Country’ niet essentieel. Ook moet het lettertype worden aangepast indien het scherm kleiner wordt dan 450 pixels, zodat de rest van de eigenschappen netjes in de tabel passen.

Op de volgende pagina is te vinden hoe deze keuzes zijn toegepast met behulp van CSS media-queries.

```
@media (max-width:450px) {
    #PlayersTable .avg { display: none; }
    #PlayersTable td { font-size: 9px; }
    #PlayersTable th { font-size: 11px; }
}

@media (max-width:600px) {
    #PlayersTable .club { display: none; }
    #divFooter { font-size: 11px; }
}

@media (max-width:700px) {
    #PlayersTable .country { display: none; }
}

@media (max-width:800px) {
    #divMain { margin: 0; }
    #divWrapper { width: 95%; height: 966px; }
    #divUpperRowContainer { width: 95%; }
    #divHeader { margin-top: 10px; }
        #divHeader img { width: 95%; }
    #divRightColumn, #divLeftColumn { float: left; width: 100%; }
    #divExplain { display: none; }
    #divSquareAd, #divSquareAd2, #divPlayerListAd { display: none; }
    #divLeftColumn #divFilter { height: 400px; }
    #divFilter #divFilterContent { height: 344px; }
    #divFilterLeftColumn { width: 100%; }
    #divFilterRightColumn { padding: 0; width: 100%; height: 190px; }
    .divFilterPart { margin-bottom: 5px; height: 60px; width: 100%; }
        .divFilterPart .divSlider { width: 100%; }
        .divFilterPart label { margin-left: 5px; }
    #divFooter { height: auto; margin-top: 5px; padding-bottom: 5px; }
    #PlayersTable tbody tr:hover { background-color: white; }
    #PlayersTable tbody tr:active { background-color: #fff9ca; }
    #divPositions { margin-top: 15px; }
}
```

7.2.2 Toepassen criteria op spelerslijst

Voor het toepassing van de criteria op de spelerslijst had ik verschillende mogelijkheden. Mogelijkheid één was om een nieuwe methode in de PlayerRepository aan te maken die een query opbouwde met alle meegegeven criteria. Nadeel hiervan is dat er geen caching toegepast kan worden op de gehele lijst met spelers.

Mogelijkheid twee was om vanuit de PlayerService alle spelers op te halen uit de PlayerRepository door middel van de FetchAll methode. Vervolgens kan er in de PlayerService het filter toegepast worden met behulp van LINQ statements. Voordeel hiervan is dat altijd dezelfde query uitgevoerd wordt en dat er eenvoudig caching toegepast kan worden.

Uiteindelijk heb ik er dus voor gekozen om vanuit de PlayerService de gehele spelerslijst op te halen uit de PlayerRepository, omdat dit simpelweg veel sneller is in combinatie met caching. Aangezien de data van de scoutlijst maar één keer in de maand aangepast wordt, is het geen probleem om de spelerslijst voor een aantal uur te cachen.

7.2.3 API uitbreiding

Huidige API

De huidige API van Online Soccer Manager is in een ASP.NET MVC structuur ontwikkeld. Door het gebruik van deze structuur wordt er net als de web applicatie gebruik gemaakt van Controllers. De Controllers van de API kunnen weer gebruik maken van de Service laag binnen het data project. Dit zorgt ervoor dat de API Controller acties dezelfde functies van de Service aan kunnen roepen als dat de web applicatie doet. Hierdoor blijft de data die verstreken wordt altijd consistent.

De API van Online Soccer Manager ondersteunt twee types methoden, namelijk de GET en de POST methoden. De GET methode kan gebruikt worden om informatie op te halen en de POST methode kan gebruikt worden om waardes in de database te veranderen.

De API ondersteunt de volgende formaten; XML, JSON en plist. Het formaat van de output van een API request kan meegegeven worden in de URL. Wanneer dit niet gebeurt wordt er standaard XML teruggegeven. Het specifiëren van de formaten gaat als volgt:

- XML - http://osmapi.onlinefootballmanager.co.uk/auth/initialize/xml
- JSON - http://osmapi.onlinefootballmanager.co.uk/auth/initialize/json
- plist - http://osmapi.onlinefootballmanager.co.uk/auth/initialize/plist

Uitbreidingen

Omdat er geen wijzigingen meer teweeg zijn gebracht aan eisen van de mobiele platformen, kon ik de API uitbreiding ontwikkelen volgens het Controller ontwerp welke opgesteld is in de elaboration fase.

Testen

In de tweede iteratie van dit project worden de mobiele applicaties van de scoutlijst gerealiseerd. Om hier unittests voor te maken heb ik tijdens het uitbreiden van de API enkele extra Controller acties toegevoegd. Deze extra Controller acties geven lijsten van voor gegenereerde Players, Teams of Countries. Deze lijsten zullen dus altijd dezelfde waardes bevatten, waardoor het eenvoudig is om hier op te unittesten binnen de mobiele omgevingen.

7.2.4 Unittests

Nadat de webapplicatie was afgerond, ben ik begonnen aan het maken van unittests. Ik heb ervoor gekozen om de Service laag van de applicatie te unittesten en niet de ViewModels, omdat de Service laag alle logica bevat.

Tijdens het ontwikkelen van de web applicatie van de scoutlijst, heb ik gebruik gemaakt van het centrale data project van Online Soccer Manager. Hier heb ik, zoals eerder beschreven, mijn benodigde Service en Repository aangemaakt. Omdat er voor de Service laag unittests geschreven moeten worden, is er een centraal data test project aanwezig.

Om unittests te maken heb ik in het data test project een PlayerTest klasse aangemaakt waarin de unittests opgenomen kunnen worden. Binnen de PlayerTest klasse heb ik een aantal methoden aangemaakt met de annotatie `[TestMethod]`, dit is om aan Visual Studio te laten zien dat het om een unittest gaat. Hieronder een voorbeeld van één van de unittests die is geschreven tijdens het ontwikkelen van de scoutlijst.

```
[TestMethod]
        public void FetchScoutlistPlayerByCriteria()
        {
            var criteriaForm = new PlayerCriteriaForm()
            {
                Age = 2,
                MinStatAtt = 0,
                MaxStatAtt = 100,
                MaxStatDef = 100,
                MinStatDef = 0,
                MinValue = 0,
                MaxValue = 50000000
            };
            IList<ForeignPlayerTeamDataModel> playerList = new
ForeignPlayerService(TestDataContext.Get).FetchByForm(criteriaForm).ToList();
            Assert.IsNotNull(playerList);
            Assert.AreEqual(playerList.Count, 938);
        }
```

In deze unittest is te zien dat er een criteria model wordt aangemaakt met specifiek ingevulde gegevens. Vervolgens wordt er een Service request gedaan met als parameter het criteria model. Opvallend is dat bij het aanmaken van de Service het TestDataContext object meegegeven wordt. Dit heb ik gedaan om ervoor te zorgen dat de test database van Gamebasics gebruikt wordt, waarbij de data nooit veranderd. Nadat de Service request voltooid is wordt de inhoud van het playerList object gecontroleerd op verwachte data. Als deze data hetzelfde is als de verwachte data, is de unittest geslaagd.

7.3 Iteratie 2: Het ontwikkelen van de mobiele applicaties

Tijdens deze iteratie is er gewerkt aan de mobiele applicaties. De scoutlijst voor het iOS platform heb ik als eerste ontwikkeld, dit was in overleg met de opdrachtgever besloten tijdens het opstellen van de requirements. De mobiele applicaties maken allebei gebruik van de API uitbreiding die is ontwikkeld in iteratie één.

7.3.1 iOS scoutlijst

Om de iOS applicatie te realiseren, moest ik een keuze maken wat betreft de data ophalen via de API. Moet de data iedere keer opnieuw ingeladen worden als de applicatie wordt gestart? Of moet de applicatie de data opslaan en updaten indien de data aangepast is. In dit hoofdstuk wordt er antwoord gegeven op deze vragen.

Data updaten

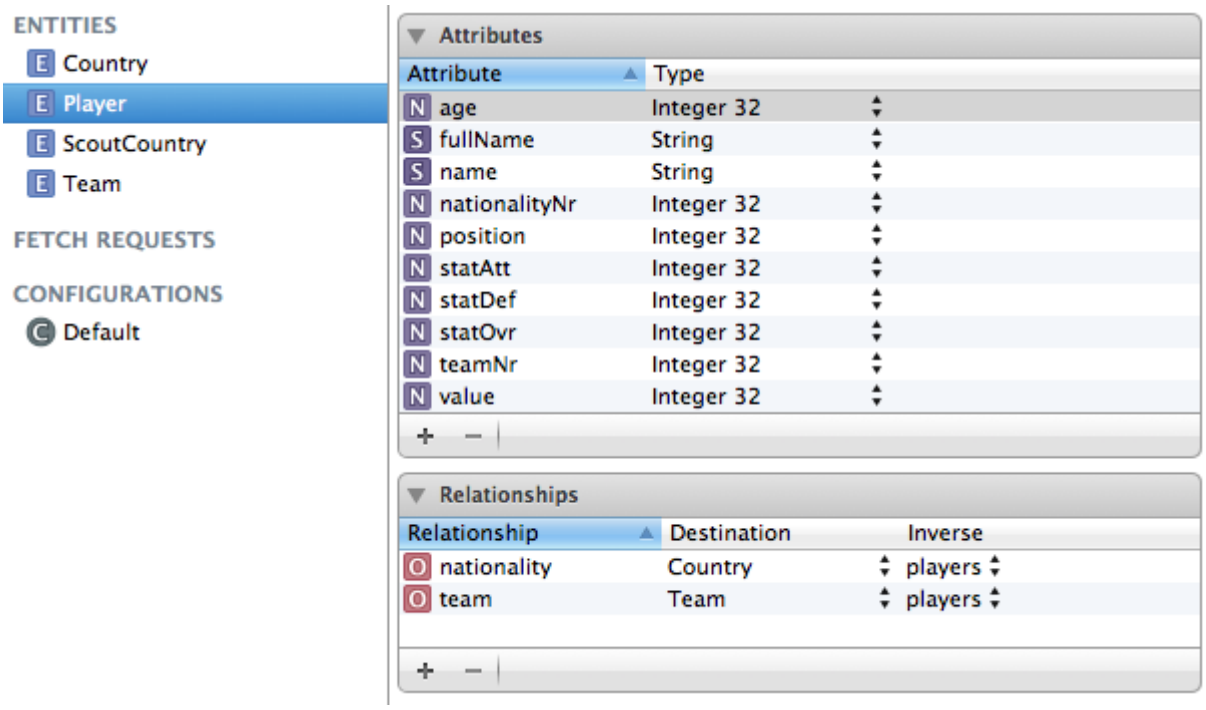
Om een keuze te maken tussen deze twee mogelijkheden, heb ik de Community Manager van Online Soccer Manager geraadpleegd met de vraag ‘Hoe frequent wordt de data van de spelers van de scoutlijst aangepast?’. Hierop kreeg ik het volgende antwoord: ‘De data wordt na transferperiodes, de zomer en de winter, een week lang ge-update.’ Uiteindelijk heb ik de keuze gemaakt om data op te slaan op de telefoon en eens in de zoveel tijd een data update te doen om de volgende redenen.

- De applicatie is offline te gebruiken nadat de applicatie eenmalig met een internet verbinding is opgestart.
- De applicatie start vele malen sneller wanneer er geen API requests gedaan hoeven worden.

CoreData

Wanneer er begonnen wordt met het maken van een iOS applicatie is het mogelijk om gebruik te maken van CoreData. CoreData is een systeem om data mee te organiseren volgens het 'entity-relationshipmodel' om vervolgens te worden geserialiseerd in XML of SQLite-data. De data kan worden aangepast door middel van objecten van een hoger niveau die entiteiten en hun relaties voorstellen.

Voor de scoutlijst applicatie heb ik als eerst een Data Model bestand aangemaakt. Vanuit dit bestand wordt er een interface getoond waarin entiteiten met hun bijbehorende relaties toegevoegd kunnen worden.



Figuur 25 - CoreData interface

In het bovenstaande figuur is de Player entiteit geselecteerd. Aan de rechterkant zijn de attributen en de relaties van de Player entiteit te zien zoals ontworpen in de elaboration fase, hoofdstuk 6.5. Na alle attributen en relaties te hebben toegevoegd, konden de Models worden gegenereerd. Hierdoor ontstaan in de geselecteerde map de klassen van de entiteiten.

Performance

Na het implementeren van Core Data ben ik de API requests in de aplicatie gaan verwerken. Om dit te doen heb ik categories aangemaakt van de door Core Data gegenereerde Models. In de categories heb ik static methoden aangemaakt die na het maken van de API request de JSON omzet naar bruikbare Objective-C objecten.

Nadat de data is opgeslagen in Core Data, moet er door de ontwerp keuze nog een koppeling worden gemaakt tussen Players, Teams en LeagueTypes. Om dat te doen wordt er bij het aanmaken van Players in de lijst met Teams gekeken en een koppeling gemaakt op basis van het TeamNr van de Player. Dit gebeurt ook voor de LeagueTypes om de nationaliteit van de speler te koppelen. Het koppelen van de Teams en LeagueTypes zag er als volgt uit:

```
+ (NSArray *)loadPlayers:(NSManagedObjectContext *)context teams:(NSArray *)teams countries:(NSArray *)countries
{

    NSLog(@"Loading new players");
    NSDictionary *result = [Api doUnauthorizedRequestWithController:@"Scoutlist" action:@"FetchPlayers" parameters:nil];
    NSArray *resultDict = [result valueForKey:@"Result"];

    NSMutableArray* players = [[NSMutableArray alloc] initWithCapacity:[resultDict count]];
    NSLog(@"Create players");
    for(int i = 0; i < [resultDict count]; i++) {

        NSDictionary* playerEntry = [resultDict objectAtIndex:i];
        Player *player = [NSManagedObject createFromDictionaryAndDescription:[playerEntry objectForKey:@"Player"]
                                                                       entity:@"Player"
                                                                      context:context];

        NSPredicate *predicate = [NSPredicate predicateWithFormat:@"(teamNr == team.nr)"];
        player.team = [[teams filteredArrayUsingPredicate:predicate] objectAtIndex:0];

        predicate = [NSPredicate predicateWithFormat:@"(nationalityNr == country.nr)"];
        player.nationality = [[teams filteredArrayUsingPredicate:predicate] objectAtIndex:0];

        [players addObject:player];
    }

    NSLog(@"Finished loading players");

    return players;

}
```

Tijdens het opstarten van de applicatie wordt deze API request uitgevoerd nadat de Teams en LeagueType API requests zijn uitgevoerd, zodat Team en LeagueType direct daarna gekoppeld kunnen worden aan de Player. Performance bleek echter een issue na het implementeren hiervan. Het opzetten van de Player objecten duurde op en iPhone 4 circa 1 minuut en 20 seconde. Dit is natuurlijk veel te lang om een applicatie op te starten. De oorzaak van deze performance issue bleek te komen uit het opzoeken van het bijbehorende Team. Omdat er meer dan 3000 spelers worden opgehaald met de API request en ongeveer 700 teams, moet de applicatie 3000 keer in een lijst met 700 teams zoeken.

Een oplossing hiervoor is om niet vier verschillende API requests te doen, maar één grote API request waar de koppeling van Players met Teams en LeagueTypes al is toegepast. Hierdoor hoeft de JSON alleen maar omgezet te worden naar een bruikbaar object en hoeft er niet meer naar de bijbehorende Teams en LeagueTypes gezocht te worden.

Om deze oplossing te implementeren moest de API uitbreiding in iteratie één aangepast worden. Het nieuwe ontwerp van de Controller met bijbehorende acties zag er vervolgens als volgt uit:

- Player
    - FetchPlayers

- Country
    - FetchScoutCountries

Deze aanpassing had een grote impact op het iOS project. De API request die werden gedaan van de Team categorie en LeagueType categorie konden worden verwijderd en worden verwerkt in de API request van de Players. Het ophalen van Players gebeurde als volgt:

```
+ (NSArray *)loadPlayers:(NSManagedObjectContext *)context
{
    NSLog(@"Loading new players");
    NSDictionary *result = [Api doUnauthorizedRequestWithController:@"Scoutlist" action:@"FetchPlayers" parameters:nil];
    NSArray *resultDict = [result valueForKey:@"Result"];

    NSMutableArray* players = [[NSMutableArray alloc] initWithCapacity:[resultDict count]];
    NSLog(@"Create players");
    for(int i = 0; i < [resultDict count]; i++) {

        NSDictionary* playerEntry = [resultDict objectAtIndex:i];
        Player *player = [NSManagedObject createFromDictionaryAndDescription:[playerEntry objectForKey:@"Player"]
                                                                       entity:@"Player"
                                                                      context:context];

        Team *team = [NSManagedObject createFromDictionaryAndDescription:[playerEntry objectForKey:@"Team"]
                                                                  entity:@"Team"
                                                                 context:context];

        Country *country = [NSManagedObject createFromDictionaryAndDescription:[playerEntry objectForKey:@"Nationality"]
                                                                        entity:@"Country"
                                                                       context:context];

        player.nationality = country;
        player.team = team;

        [players addObject:player];
    }

    NSLog(@"Finished loading players");
    return players;
}
```

7.3.2 Android scoutlijst

Android Studio

In eerste instantie zou ik de Android applicatie gaan ontwikkelen in de ontwikkelomgeving Eclipse. De dag dat ik klaar was met het ontwikkelen van de iOS applicatie werd de eerste versie van Android Studio gereleaset. Android Studio is een ontwikkelomgeving die gebaseerd is op IntelliJ IDEA.

Er is door Gamebasics de keuze gemaakt om de Online Soccer Manager applicatie te porten naar de nieuwe ontwikkelomgeving. Daardoor was ik genoodzaakt om de Android scoutlijst te ontwikkelen binnen Android Studio. Hieronder de voordelen van Android Studio op een rij:

- Android specifieke refactoring en snelle oplossingen.
- Een rijke lay-out editor die previews van meerdere scherm configuraties laat zien.
- Het builden van applicatie die een release nabootst met behulp van ProGuard.
- Preview van Strings / kleuren / plaatjes.

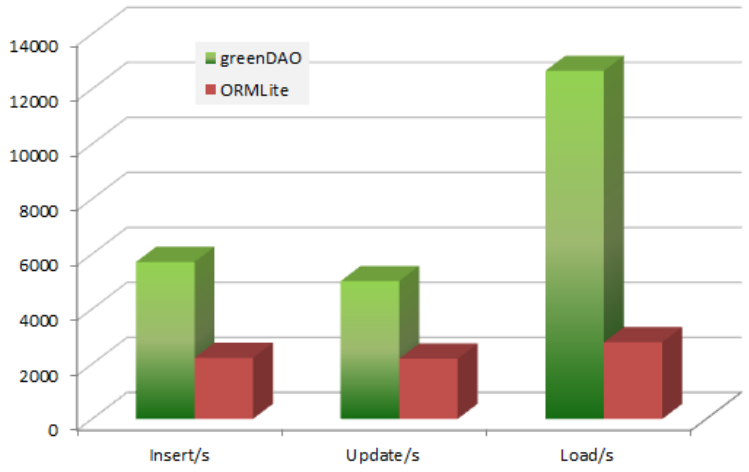
GreenDAO & ORMLite

Na een project te hebben gecreëerd binnen Android Studio ben ik verder gegaan met het uitzoeken van een Object-relational mapping (ORM) voor de scoutlijst.

Om een Android applicatie te bouwen zijn verschillende ORM's beschikbaar. ORMLite en GreenDAO zijn twee veel gebruikte varianten. Om de scoutlijst applicatie te realiseren, dient er dus één van deze twee gekozen te worden. Verschillende criteria hebben daar invloed op, snelheid, gebruiksvriendelijkheid en grootte.

Allereerst kwam ik erachter dat ORMLite gebruikt maakt van annotaties. Annotaties worden vooral gebruikt om het gedrag van bepaalde klassen run-time aan te passen. Vaak zorgt het er ook voor dat de hoeveelheid code verlaagd wordt. GreenDAO gebruikt daarentegen gegenereerde klassen, dit gebeurt al voordat de applicatie gebuild wordt. Op voorhand wint GreenDAO het dus al op snelheid, wat een essentieel punt is op het Android platform.

GreenDAO heeft een onderzoek gedaan naar de snelheid tegenover ORMLite. Hieruit blijkt dat entiteiten opslaan en updaten ruim twee keer zo snel gaat en het laden vier en een half keer zo snel gaat als ORMLite [5].



Figuur 26 - GreenDAO & ORMLite

De scoutlijst applicatie haalt één keer per week de nieuwste spelerslijst op, de andere dagen moet de ORM ervoor zorgen dat de data snel ingeladen wordt. Omdat inladen dus van essentieel belang is voor een goede werking van de scoutlijst applicatie heb ik ervoor gekozen om GreenDAO te gebruiken als ORM voor de Android scoutlijst. Een bijkomend voordeel van GreenDAO was ook dat de library maar 100kb groot is. Waardoor de scoutlijst applicatie nauwelijks groter wordt.

Om GreenDAO te implementeren moest een zoals eerder beschreven een losstaand project gemaakt worden met de GreenDAO Generator. In het generator project heb ik een klasse 'ScoutlistDaoGenerator' gemaakt. Deze klasse bevat per entiteit een methode die de entiteit opbouwt, inclusief de relaties tussen entiteiten. In de main functie worden deze methoden uitgevoerd en worden er klassen in de scoutlijst applicatie gegenereerd.

7.4 Systeemtest

Zoals in de elaboration fase is beschreven ga ik gebruik maken van exploratory testing. Het testverslag dat ik heb opgesteld tijden de elaboration fase heb ik gebruikt om onderdelen van de applicatie te definiëren. Vanuit het gedefinieerde onderdeel worden alle functionaliteiten op elk mogelijke manier doorlopen om te controleren of er zich fouten voordoen.

De incidenten die zich voordeel tijdens het testen van een specifiek onderdeel worden genoteerd. Van deze genoteerde incidenten moest er een keuze worden gemaakt of het incident opgelost moest worden, of het incident later opgelost moet worden of dat het incident buiten de scope van de opdracht valt en dus niet opgelost hoeft te worden.

Uiteindelijk zijn er maar een aantal kleine incidenten gevonden:

Webapplicatie

1. De Arabische versie wordt niet van rechts naar links getoond.
2. Op een mobiel device zijn de knoppen van de slider van het filter niet groot genoeg.
3. Op een klein mobiel device past de tabel van de spelers niet netjes op de pagina

iOS applicatie

1. Profiel aan de rechterzijde kan geopend worden zonder dat er eerst een speler geselecteerd is.
2. De picker om te kunnen filteren op land wordt niet goed gevuld in landscape modus.
3. Wanneer een device in landscape modus zit en vervolgens plat op een tafel neergelegd wordt verdwijnt het team eigenschap uit de spelerslijst van de spelers.

Android applicatie

1. Het filteren van het land gebeurd op de nationaliteit van spelers en niet in welke land de speler speelt.
2. Er zitten geen klik animaties op de knoppen

Nadat al deze incidenten opgesteld waren is er per incident gekeken of deze opgelost moest worden binnen de opdracht. Al de fouten die zijn gevonden waren in feite 'bugs' van de scoutlijst en moesten dus gewoon binnen de opdracht opgelost worden. Nadat ze opgelost waren zijn deze bugs her-test door de testafdeling van Gamebasics. Hier kwam uit dat de bugs waren opgelost.

8 Transition fase

Een systeem is niet af wanneer het niet daadwerkelijk geïmplementeerd is. Hiervoor is in RUP de transition fase voor bedoelt. Deze fase zorgt dus voor de overgang van het software-ontwikkelingstraject naar een werkend geïmplementeerd systeem. In deze transition fase heb ik alleen de acceptatietest uitgevoerd en het systeem voorbereid op implementatie. Het daadwerkelijk implementeren van de applicaties valt buiten de scope van mijn afstudeeropdracht.

8.1 Acceptatietest

Voor het uitvoeren van de acceptatietests van de webapplicatie heb ik gebruik gemaakt van de 'continuous integration server' TeamCity. TeamCity is een tool die code analyses uitvoert, compileert en uiteindelijk een build doet. TeamCity is al in gebruik door Gamebasics om de Online Soccer Manager applicatie te builden. Ik was dus genoodzaakt om hier ook gebruik van te maken.

De OSM iOS applicatie maakt gebruik van TestFlight voor het testen van de API. TestFlight is een 'over-the-air' platform om applicatie in beta te laten testen door teamleden, hierdoor kunnen ontwikkelaars eenvoudig feedback krijgen en is een applicatie eenvoudig te distribueren. Een wens van Gamebasics was dat de scoutlijst hiervan gebruik ging maken om ervoor te zorgen dat er eenvoudig getest kon worden.

Na het opzetten van de omgevingen heb ik de complete marketing afdeling van Gamebasics uitgenodigd om een acceptatietest te komen uitvoeren. De marketeers van Gamebasics speelt zelf dagelijks Online Soccer Manager, waardoor dit een representatieve groep is aan de toekomstige scoutlijst gebruikers.

Om de tests door de marketeers te laten uitvoeren heb ik een acceptatietest document (Bijlage: Acceptatietest) opgesteld. Dit document bevat een beschrijving hoe ze bij de testomgeving kunnen komen en welke functionaliteiten ze moeten testen. Ik heb ervoor gekozen om alleen

functionaliteiten te benoemen in de acceptatietest in plaats van de gebruikelijke stappenplannen om tot een bepaald resultaat te komen, omdat ik tevens wil testen of het gebruik van de scoutlijst applicaties vanzelfsprekend is. Zijn de acties die uitgevoerd moeten worden om een bepaald resultaat te krijgen 'logisch'?

Per functionaliteit heb ik de marketeers een cijfer laten geven op een schaal van één tot en met tien en een veld met opmerkingen waarin de marketeers eventueel onlogische acties kunnen vermelden. Hieronder een voorbeeld van een functionaliteit die beschreven staat in de acceptatietest

Functionaliteit 1

Bij het opstarten worden standaard alle spelers ingeladen en in een lijst getoond. Ik zie voldoende data in de lijst en de lijst scrolt soepel. Ook toont de lijst in landscape modus genoeg data en is er goed door de lijst te scrollen.

Cijfer:

Opmerkingen:

9 Evaluatie

Nadat ik mijn eindproducten had afgerond, heb ik mijn werkwijze geëvalueerd. In dit hoofdstuk blik ik terug op de eindproducten en de manier waarop deze tot stand zijn gekomen. Ik zal me vooral bezig houden met mijne sterke en zwakke punten, over waar het eventueel beter zou kunnen en of ik het de volgende keer op dezelfde manier zou aanpakken.

9.1 Productevaluatie

Aan het einde van mijn afstudeerperiode heb ik de scoutlijst applicaties opgeleverd conform de opgestelde systeemeisen. Bij deze oplevering zat ondersteunende documentatie vanuit RUP, denk hierbij aan een inception rapport, elaboration rapport, testplannen en een construction rapport.

Na een presentatie van de web scoutlijst en de iOS scoutlijst is naar voren gekomen dat Gamebasics content is over de werking van de producten en de koppeling met de al bestaande Online Soccer Manager applicaties.

Ikzelf ben ook tevreden met het behaalde eindresultaten, van te voren zijn er een aantal eisen op tafel gelegd waar ik aan voldaan heb. De producten zijn zo opgezet dat ze eenvoudig aan te passen en uit te breiden zijn. Daarnaast is de code duidelijk geschreven en waar nodig voorzien van commentaar.

Waar ik minder tevreden over ben is het feit dat er maar één API call gebruikt wordt om de mobiele applicaties van data te voorzien. De API call is naar mijn mening te groot waarbij verschillende data meerdere keren wordt opgehaald. Dit komt ten goede op te mobiele platformen, omdat deze de data makkelijker om kunnen zetten naar bruikbare objecten.

Ik ben ook erg tevreden over het gebruik van responsive web design, omdat ik hier nog nooit mee gewerkt had waren enige twijfels aanwezig. Uiteindelijk werkt de web applicaties perfect op alle resoluties en kan dus zonder problemen gebruikt worden op kleine apparaten.

9.2 Procesevaluatie

In deze procesevaluatie zal ik een evaluatie geven van het proces dat ik heb doorlopen om tot de eindproducten te komen. Ik zal hierbij mijn ervaring per fase opstellen.

Inception fase

Het goed uitvoeren van de inception fase bleek later van essentieel belang in dit project, zodat er het gehele project duidelijk is wat er gemaakt moest worden en welke eisen de stakeholders aan het product stelden. De snelle feedback van de opdrachtgever op de systeemeisen heeft hier een grote rol in gespeeld. Tevens ben ik achteraf erachter gekomen dat het ruim inplannen van de inception fase een goed idee was, omdat ik zo genoeg tijd had om voor onderzoek te doen naar voor mij onbekende technieken.

Elaboration fase

In de elaboration fase heb ik aan de hand van overleg met de stakeholders de wensen en eisen aangepast. Daarop konden verschillende ontwerpen voor de scoutlijst worden gemaakt die later enorm hebben bijgedragen aan het ontwikkelen ervan. Ik was niet geheel tevreden met de tijd die ik had voor de prototypes. Ik had weliswaar twee prototypes opgeleverd, één voor de web applicatie en één voor de Android applicatie, maar voor de iOS applicatie had ik niet genoeg tijd, alhoewel dit wel één van mijn grootste struikelblokken zou kunnen worden. Achteraf had ik beter de prototype van iOS als eerst kunnen maken om aan te kunnen tonen dat de technisch opgestelde oplossingen mogelijk waren.

Construction fase

Over het algemeen ben ik tevreden met de construction fase en het opdelen ervan in twee iteraties. De eerste iteratie verliep soepel, de scoutlijst web applicatie kon vlot door worden ontwikkeld en ook de bestaande API was prima uit te breiden. In de tweede iteratie begon ik met het ontwikkelen van de mobiele applicaties. De opdrachtgever had, zoals eerder beschreven, de prioriteit bij het iOS platform gelegd. Ik was hierdoor dus genoodzaakt om te beginnen met het platform waarvan ik de minste kennis bezit. Totaal ben ik ongeveer zeven dagen uitgelopen op mijn planning, waardoor de scoutlijst applicatie voor het Android platform niet voor 100% afgerond kon worden. Desalniettemin stond er een iOS applicatie klaar om in de App store gezet te worden.

Transition fase

Tijdens de transition fase is er gewerkt aan het in productie nemen van de scoutlijst applicatie en iOS applicatie. Het in gebruik nemen van de scoutlijst web applicatie was geen probleem, omdat er al een oude versie van de scoutlijst live draaiden. Voor de iOS applicatie moet er nog een aanvraag worden gedaan bij Apple om het product in de App store te krijgen. Aan het testen van de applicaties ben ik over het algemeen tevreden maar daar zal ik in mijn volgende project nog meer aandacht aan besteden.

9.3 Beroepstaken

9.3.1 Uitvoeren analyse door definitie van requirements

STARR	Uitwerking	Uitvoeren analyse door definitie van requirements
Situatie	Aan het begin van het afstudeertraject is de opdracht vastgesteld, als onderdeel van deze opdracht moeten er requirements worden opgesteld met behulp van meerdere stakeholders.	
Taken	Om de requirements goed op te stellen dienen er verschillende gesprekken met stakeholders gevoerd te worden.	
Actie	Per stakeholder worden de requirements opgesteld en onderverdeeld in functionele en niet-functionele requirements. De requirements worden tevens geprioriteerd met behulp van MoSCoW in overleg met de stakeholders.	
Resultaat	In het inception rapport en elaboration rapport zijn de (aangepaste) requirements te vinden. Aan de hand hiervan kon het project ontwikkeld worden.	
Reflectie	Het opstellen van requirements is ongelofelijk belangrijk voor dit project. De manier waarop de requirements zijn opgesteld aan de hand van de drie applicaties die ontwikkeld moesten worden ben ik tevreden over. Ook de perspective based readings die ik heb uitgevoerd in de inception en elaboration fase ben ik tevreden mee.	

9.3.2 Ontwerpen systeemdeel

STARR	Uitwerking	Ontwerpen systeemdeel
Situatie	Aan de hand van de opgestelde requirements moet er een ontwerp worden gemaakt voor de scoutlijst webapplicatie als de scoutlijst mobiele applicaties.	
Taken	Het ontwerpen van de applicaties met UML	
Actie	Met het ontwerpen van de web applicatie heb ik rekening gehouden met het huidige systeem van Gamebasics. Voor de mobiele applicaties zijn compleet nieuwe ontwerpen bedacht.	
Resultaat	Het elaboration rapport. In dit document staat het ontwerp van de applicaties beschreven. Aan de hand hiervan kon het project geïmplementeerd worden	
Reflectie	Tijdens het ontwerpen van het systeem heb ik het ontwerp prima aangepast aan het huidige systeem, hier ben ik dan ook meer dan tevreden over. De manier waarop een API request gedaan wordt heb ik bewezen in het prototype.	

9.3.3 Bouwen applicatie

STARR	Uitwerking	Bouwen applicatie
Situatie	Volgens de opdracht moeten drie verschillende applicaties gebouwd worden	
Taken	De applicaties moeten gebouwd worden aan de hand van de beschreven ontwerpen en de opgestelde requirements. Tevens moet er gebruik worden gemaakt van een framework, versiebeheer tool en een programmeertool.	
Actie	Na het prototypen te hebben gebouwd en de ontwerpkeuzes heb 'bewezen' heb ik de applicaties gebouwd. Als versiebeheer tool wordt Git gebruikt. Het framework en de programmeertool die gebruikt worden waren per platform verschillend en bestaan uit: ASP.NET MVC, Android Studio, XCode,	
Resultaat	De resultaten van deze competentie zijn de afgeronde applicaties. Per applicatie voldoen ze aan de opgestelde requirements.	
Reflectie	Ik vind zelf dat ik deze competentie heb uitgevoerd, de omschakeling naar verschillende talen en platformen maakte het niet gemakkelijker. Maar al met al zien de applicaties er goed en verzorgd uit.	

9.3.4 Uitvoeren van en rapporteren over het testproces

STARR	Uitwerking	Uitvoeren van en rapporteren over het testproces
Situatie	De drie applicaties die ontwikkeld worden moeten getest worden.	
Taken	Er moet een testframework gekozen worden aan het begin van het afstudeerproject.	
Actie	Er worden statische tests in de inception fase en elaboration uitgevoerd. Daarnaast worden dynamische tests uitgevoerd in de construction fase en transition fase.	
Resultaat	Ik heb tijdens de elaboration fase een testplan opgesteld met een testverslag die gebruikt is om in te rapporteren. Tijdens de ontwikkelingen van de applicaties in de construction fase heb ik unittests geschreven. In de transition fase heb ik een acceptatietest opgesteld en laten uitvoeren door eindgebruikers.	
Reflectie	Het opstellen van de unittests viel niks op aan te merken, de functionaliteiten hebben een goede dekking waardoor de kwaliteit van de producten bewezen worden. Voor de systeemtest heb ik exploratory testing gebruikt wat niet echt goed gedocumenteerde test cases bevat waardoor ik niet kan aantonen dat ik tot in detail goed getest heb.	

9.4 Conclusie

Concluderend kan ik zeggen dat ik een zeer leerzame afstudeerstage heb gehad, waarin ik veel kennis en ervaring heb opgedaan met een groot aantal nieuwe technieken. Hierbij doel ik niet alleen op de programmeertalen Objective-C en C#, maar ook op de verschillende ontwikkelomgevingen als Android Studio, XCode en Visual Studio.

Het doorlopen van dit complete software ontwikkeltraject met het opstellen van de bijbehorende planning is naar mijn mening succesvol verlopen en heeft mij veel inzicht gegeven in hoe een project als deze van begin tot eind succesvol doorlopen kan worden.

Ik ben tevreden met de eindproducten die ik heb opgeleverd en dat ze ook daadwerkelijk in gebruik worden genomen.

10 Glossary

.NET Framework
Applicatie framework voor het Windows platform en zorgt voor een naadloze samenwerking van applicaties die geschreven zijn in verschillende programmeertalen.

API
Een standaard interface waarmee software programma's onderling kunnen communiceren.

Caching
Cache is een geheugen om tijdelijk de meest opgevraagde informatie op te slaat, zodat deze sneller in te laden zijn.

CSS
Afkorting voor Cascading style sheets waarmee het mogelijk om vormgeving van pagina's los te koppelen van hun inhoud en op een centraal punt vast te leggen.

Git Branch
Een branch gebruik je om enkele eigenschappen of gerelateerd werk in te doen.

HTML5
Nieuwste versie van de internettaal HTML.

JavaScript
Een scripttaal die veel gebruikt wordt om web applicaties mee te ontwikkelen en om webpagina's interactief te maken.

Mockup
Een ontwerp of model op schaal of ware grootte van een ontwerp of product

Objective-C
Programmeer taal waarmee applicaties kunnen worden ontwikkeld voor het iOS platform.

ORM
Object-relational mapping is een programmeer techniek voor het converteren van data tussen

PBR
Perspective based reading, een techniek waarbij deelnemers een product vanuit één bepaald gezichtspunt beoordelen.

plist
Formaat gebruikt door de IPhone, de ondersteuning hiervoor bevind zich native op dit platform en het is een variant op XML.

XML
Standaard voor syntax waarmee men gestructureerde gegevens kan weergeven als tekst en daarmee versturen over het internet.

11 Literatuurlijst

[1] http://www.asp.net/mvc/tutorials/mvc-4/bundling-and-minification

[2]http://wiki.developerforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[3] https://developers.google.com/webmasters/smartphone-sites/details

[4] http://msdn.microsoft.com/en-us/library/ee658090.aspx

[5] http://greendao-orm.com/features/