CGI

Experience the commitment®

Gamificatie PDP

Afstudeerscriptie

Naam:	**Sangam K. Gupta**
Studentnummer:	**10081224**
School:	**Academie voor ICT & Media Zoetermeer,**
	De Haagse Hogeschool
Datum:	**03-10-2015**
Versie:	**1.0**

Voorwoord

Graag wil ik CGI bedanken voor het feit dat ze mij een opdracht ter beschikking hebben gesteld, en de volgende personen met de hulp dat zei mij boden. De bedrijfsmentor/opdrachtgever Tjeerd, hij heeft mij erg geholpen gedurende mijn hele afstudeerproces. Ook wil mijn begeleider Nanny bedanken voor haar begeleiding vanuit school.

Zoetermeer, 4 oktober 2015

Sangam Kumar Gupta

Documenthistorie

Versie	Omschrijving	Datum
0.1	Eerste opzet Afstudeerdossier	26-05-2015
0.2	Toevoeging Iteratie 1 construction fase, verbeteren/uitbreiden inception fase en toevoeging bijlages afstudeerplan en Plan van Aanpak.	12-06-2015
0.3	Toevoeging iteratie 2	27-06-2015
0.4	Verbeteren alle hoofdstukken voor 60% inlevering	02-07-2015
0.5	Toevoeging iteratie 3 en verbeterpunten 60% gesprek verwerkt.	08-07-2015
0.6	Toevoeging Iteratie 4 en 5. Verbeterpunten uit het gesprek verwerkt	27-07-2015
0.7	Verbeteren iteratie 3, 4 en 5	19-08-2015
0.8	Toevoeging iteratie 6, verbeteren rest van het document	02-09-2015
0.9	Verwerken 80% feedback	22-09-2015
1.0	Afronding afstudeerverslag	02-10-2015

Inhoudsopgave

Begrippenlijst

Begrip	Definitie/verklaring
Backend	Spring applicatie met een database. Het geeft voornamelijk data door aan de front-end.
CGICraft	De naam van de gebouwde applicatie. Dit omvat zowel de front- en backend. De naam is overgenomen van de mock-up die dit project tot stand heeft gebracht.
DAD	Disiciplined Agile Delivery, ontwikkelmethode.
Front-end	Android app.
Game elementen	Onderdelen/aspecten die veel voorkomen in games.
Gamificatie	Het toevoegen van game elementen aan een niet game context om zo gebruikers betrokkenheid te vergroten.
Kennisdeling	De deling van kennis onderling de medewerkers van CGI.
Kennisontwikkeling	De persoonlijke kennisontwikkeling van de medewerkers van CGI.
Members	De medewerkers worden binnen CGI members genoemd.
PDP	Personal Development Plan, een interne systeem bij CGI met als goal om de medewerkers te ondersteunen/promoten bij hun persoonlijke ontwikkeling.
Practice	Vakgebied e.g. Software Engineer, tester, Manager, etc. CGI kent 10 practices.
Talent	Binnen de vakgebied de expertise/ specialisatie e.g. Java programmeur.
Crafter, Speler, Gebruiker, Medewerker	De gebruikers van het te ontwikkelen systeem. De gebruikers zijn allemaal de medewerkers van CGI. Maar de naamgeving kan verschillen per fase.

1 Inleiding

Het gamificeren (toevoegen van game elementen aan een niet-game context) van alledaagse bezigheden wordt steeds populairder. Door de groeiende game industrie die in de komende jaren een extra boost krijgt door de opkomende mobiele markt [1], heeft gamificatie zijn weg gevonden in het dagelijks leven van iedere gamer en ook niet gamer. Zo is er een fitness tracker die je het gevoel geeft dat je achtervolgd wordt door honderden zombies of een app die je spelenderwijs een nieuwe taal leert.

Gamificatie daar kan men nauwelijks omheen. CGI wil hier de kracht van meten. Binnen CGI wordt elke medewerker gemotiveerd om zichzelf te ontwikkelen en zijn kennis te delen. Dit wordt gedaan met behulp van het Personal Development Plan (PDP). Echter in praktijk valt dit erg tegen. Het PDP stimuleert niet en is niet erg gebruiksvriendelijk.

Er komt een nieuwe gegamificeerde systeem dat ervoor moet zorgen dat het gebruik van het PDP toeneemt. Daarnaast moet het kennisdeling en ontwikkeling stimuleren.

Als eerste wordt in hoofdstuk twee het bedrijf waar de afstudeeropdracht is uitgevoerd besproken. Hierbij wordt duidelijk waar ik binnen de organisatie ben geplaatst.

Vervolgens komt in hoofdstuk drie de opdracht aan bod en bespreek ik de gekozen aanpak voor dit project. In hoofdstuk vier beschrijf ik mijn werkzaamheden tijdens de inception fase, gevolgd door Iteratie 1 t/m 6 in hoofdstuk vijf.

Als laatste vindt u de zelfevaluatie van het proces en product in de hoofdstukken zeven en acht. En tot slot de evaluatie van de beroepstaken.

2 Bedrijf oriëntatie

2.1 Beschrijving

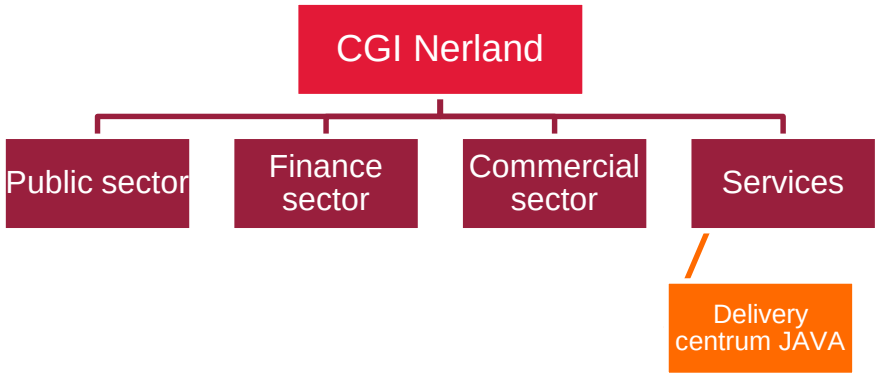
CGI Group Inc. is een multinationaal informatie en technologie consulting group met meer dan 400 vestigingen in 40 landen met zo'n 68.000 werknemers. Nederland kent ook een aantal vestigingen waaronder de grootste in Rotterdam is. Hier vindt mijn afstudeertraject plaats. CGI is opgericht in 1976 door Serge Godin en André Imbeau. De collectieve droom van CGI is:

"To create an environment in which we enjoy working together and, as owners, contribute to building a company we can be proud of."

De structuur van de organisatie is hiërarchisch. De sfeer binnen de organisatie is informeel en de kledingdracht is business casual.

2.2 Organisatie

De organisatie van CGI Nederland ziet er als volgt uit:



Figuur 1: organogram CGI Nederland

In figuur 1 is te zien dat CGI drie sectoren kent: financieel, publiek(overheid) en commercieel. Services ondersteunen deze sectoren. Binnen elke sector werken mensen met kennis van dat vakgebied. Elke sector is vervolgens verder verdeeld in afdelingen, deze zijn wegelaten om het overzicht in het diagram te behouden.

Gedurende mijn afstudeerstage werk ik binnen het delivery centrum JAVA als een Software Engineer. Voor dit project kom ik in contact met de gamificatie expert, de opdrachtgever (hij is tevens mijn mentor) en een architect.

Voor vragen of opmerkingen kan ik direct terecht bij mijn opdrachtgever/mentor indien hij niet bezig is. Voor specifieke vragen over gamificatie kan ik terecht bij gamificatie expert.

3 Afstudeeropdracht

3.1 Probleemstelling

De probleemstelling is te vinden in bijlage B, hoofdstuk 2.2. Ter ondersteuning, het PDP is een intern systeem waar de medewerkers van CGI zijn goals en doelstellingen kunnen opnemen. De goal van het systeem is om kennisdeling en ontwikkeling te stimuleren. Dit systeem wordt tevens gebruikt bij de beoordeling van een medewerker. In praktijk wordt het niet of nauwelijks gebruikt.

In het verleden zijn er meerdere projecten uitgevoerd met als goal kennisdeling en ontwikkeling te verhogen. Deze projecten zijn min of meer mislukt. In plaats van een nieuw systeem wil CGI het huidige PDP aantrekkelijker maken en ervoor zorgen dat dit PDP op vrijwillige basis gebruikt gaat worden met behulp van gamificatie.

Gamificatie volgens het Oxford woordenboek [2] is:

“The application of typical elements of game playing (e.g. point scoring, competition with others, rules of play) to other areas of activity, typically as an online marketing technique to encourage engagement with a product or service:”

Gamificatie is kort door de bocht het toevoegen van game elementen aan een niet game context. Om zo de gebruikers betrokkenheid te bevorderen. In hoofdstuk 4.1.1 wordt er dieper in gegaan op gamificatie en in het gamedesigndocument bijlage C vind je extra uitleg m.b.t. gamificatie en al mijn bevindingen.

3.2 Doelstelling

De opdracht is om de mock-up (zie hoofdstuk 4.1.4) dat voor de gamificatie van kennisopbouw/deling is ontwikkeld verder uit te werken en hier een eerste versie van te implementeren. Dit omvat het ontwerpen van de hiervoor in te zetten koppelingen met gerelateerde systemen, targetplatform(s) en bepalen van de voor de ontwikkeling te gebruiken tools.

Binnen Nederland is gamificatie nog niet erg bekend of populair, daarnaast heeft CGI Nederland nog nooit gewerkt met gamificatie. Het gaat dus om een innoverende opdracht.

Uit gesprekken met de opdrachtgever zijn voor deze opdracht de volgende doelstellingen opgesteld:

- Stimuleren kennisdeling, de kennis van een medewerker beschikbaar stellen voor de rest van CGI.
- Stimuleren kennisontwikkeling, persoonlijke kennisontwikkeling
- Verhogen gebruik PDP
- Het te bouwen systeem plaatsen binnen de huidige architectuur van CGI

De primaire doelstelling is het gebruik van het PDP verhogen, dit heeft raakvlakken met Stimuleren kennisdeling en ontwikkeling. De doelstellingen zijn niet SMART geformuleerd. Dat komt omdat persoonlijke kennisontwikkeling en deling heel lastig zijn om te meten. Wel wordt de applicatie bij een testgroep ingezet om zo feedback te verzamelen.

3.3 Resultaat

Na het afronden van de opdracht heeft CGI een gegamificeert systeem voor kennisontwikkeling en deling die al enkele iteraties heeft ondergaan. Het systeem is aangesloten op de relevante servers en databases van CGI en data wordt real time heen en weer gestuurd. Met andere woorden, de gamificatie is “playable”. Informatie van medewerkers kan snel geladen worden en de gamificatie systemen werken goed. Het systeem wordt door een testgroep gespeeld en er is (hopelijk) een toenemende trend te zien bij het gebruik van PDP.

3.4 Aanpak

De gekozen aanpak voor dit project is Disciplined Agile Delivery (DAD). Binnen het Delivery Center Java is dit een veel gebruikte ontwikkelmethodiek. DAD is een hybride ontwikkelmethode die gebouwd is op bestaande softwareontwikkelmethodes zoals SCRUM, RUP en Extreme Programming (XP).

Om exact te zijn is DAD:

“The Disciplined Agile Delivery (DAD) decision process framework is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value delivery lifecycle, is goal-driven, is enterprise aware, and is scalable.” [3]

DAD kent explicit drie fases afgeleid uit RUP zoals getoond in figuur twee:

- Inception
- Construction
- Transition

Per fase geef ik een korte omschrijving en leg ik uit hoe het is toegepast voor dit project.

Inception fase

Inception is de initialisatie fase, binnen SCRUM wordt dit ook wel gezien als sprint 0. Gedurende deze fase worden de wensen en eisen van de opdrachtgever in kaart gebracht, hierbij worden de juiste oplossingen bedacht. In bespreking met mijn stagebegeleider is er voor dit project gekozen voor een periode van drie weken. Tijdens de inception fase ga ik:

- de requirements identificeren en verwerken;
- een Plan van aanpak opstellen;
- verdiepen in de huidige architectuur;
- verdiepen in gamificatie;
- aan de hand van de bovenstaande taken bepalen wat voor applicatie er gebouwd gaat worden.

Construction fase

Construction, deze fase staat voor de ontwikkeling van de applicatie. Gedurende deze periode wordt er iteratief ontwikkeld aan het product. Elke iteratie levert een potentieel opleverbaar product op. Voor dit project is de construction fase opgedeeld in zes iteratie van ieder twee weken. Ieder Iteratie begint met een iteratie planning sessie en eindigt met een iteratie review & retrospective, waarin ik mijn voortgang laat zien en terugblik op de iteratie. De algemene taken die ik ga uitvoeren tijdens de iteraties zijn:

- een applicatie bouwen, aan de hand van de work items;
- unit testen;
- ontwerpen maken van de te ontwikkelen applicatie;
- documentatie schrijven/ bijwerken.

Aan het einde van de construction fase is er een eerste versie van het product.

Transition fase

Tijdens de transition fase wordt het product in gebruik genomen. De transition fase overlapt met de construction fase vanwege tijd tekort. Tijdens de construction fase wordt de deployment uitgevoerd, terwijl dit gewoonlijk een onderdeel is van de transition fase. Tevens zal er een kennisoverdracht plaats vinden van de opdracht. Bij overdracht worden alle relevante documenten overgedragen aan het bedrijf. Zodat zij eventueel verder kunnen met het project. Tevens moet het product voldoende zijn gedocumenteerd volgens de DC Java richtlijnen.

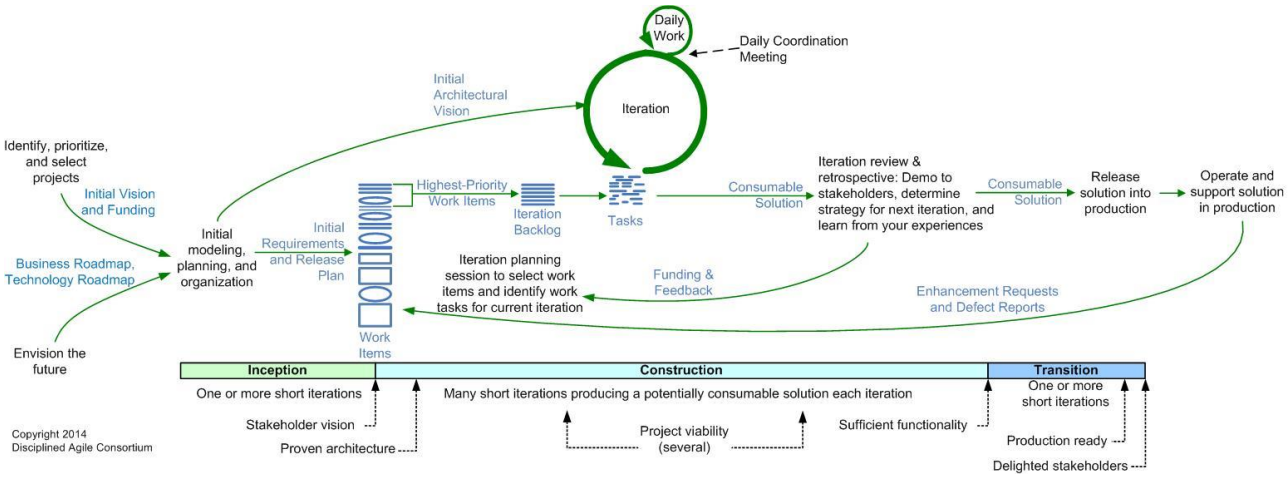
De laatste twee weken zijn voor eventuele uitloop en het schrijven van het afstudeerverslag.

Naast de planning zijn er ook een aantal aanpassingen gemaakt in de aanpak zodat het beter past bij dit project. De DAD methode staat dit toe als de situatie hier om vraagt.

Ten eerste begint dit project met de inception fase: “Initial modeling, planning, and organization” zoals te zien in figuur 2. Alles hiervoor is niet van toepassing op dit project, het is namelijk al uitgevoerd voorafgaande aan mijn afstuderen.

Ten tweede valt ook “Operate and support solution in production” na de transition fase buiten de scope van dit project. Support is niet nodig gezien de schaal van dit project.

Als laatste zijn er geen Daily Coordination Meetings omdat het team hiervoor te klein is. In plaats daarvan worden er wekelijks gesprekken met de mentor ingepland om zo voortgang en problemen toch nog tijdig te bespreken.



Figuur 2: basis levenscyclus [4], uitbreiding op SCRUM

4 Inception fase

4.1 Opstellen plan van aanpak

Bij het opstellen van het PVA (zie bijlage b), heb ik als eerst gekeken naar de inhoud. Hierbij heb ik bepaald welke informatie ik noodzakelijk vind voor dit project.

In het PVA wordt de volgende informatie opgenomen:

- Contact gegevens
- Doel en probleemstelling
- Organisatie
- Aanpak en risico's

Dit heb ik vervolgens besproken met de opdrachtgever. Hierbij heeft de opdrachtgever mij aangeraden om ook de belangrijke onderdelen uit het DAD (voor de inception fase) te verwerken in het PVA. Het gaat om de volgende onderdelen:

- Visie
- Scope
- Technische strategie

Ik licht kort toe wat de bovenstaande punten inhouden.

Visie

De visie (zie bijlage b hoofdstuk 3) bespreekt de doelstellingen van de opdrachtgever en hoe deze mogelijk kunnen worden gehaald. De doelstellingen zijn al bekend, maar hoe gamificatie kan helpen om deze te behalen is nog niet duidelijk. Om deze vraag te beantwoorden heb ik me eerst verdiept in gamificatie. In hoofdstuk 4.1.1 ga ik daar verder op in.

Uit een gesprek met de gamificatie expert bleek dat ik het beste al mijn bevindingen kan verwerken in een gamedesigndocument. Dit wordt gedurende het afstudeertraject een levend document. Het wordt continue aangepast en het is pas af als het product ook zo goed als af is.

Scope

De scope (zie bijlage b hoofdstuk 4) bevat een door de opdrachtgever goedgekeurd initiële lijst met user stories en features. De initiële lijst van user stories is gemaakt met de kennis die is opgedaan tijdens het maken van de visie (verdiepen gamificatie) en de technische strategie (architectuur). Dit proces wordt verder uitgelegd in hoofdstuk 4.1.3.

Technische strategie

De technische strategie gaat over de architectuur. Dit wordt in hoofdstuk 4.1.2 besproken.

Planning

De planning is opgesteld aan de hand van de user stories. Hierbij hebben we gelet op prioriteit en hebben we bedacht om zo snel mogelijk een eerste versie uit te brengen.

De planning ziet er als volgt uit:

Week	Periode	Fase	Activiteiten
1 t/m 3	11 mei – 29-mei	Inception	- Plan van aanpak schrijven
			- Gamedesign document opstellen
			- Requirements identificeren
			- Verdiepen huidige architectuur en keuze maken over type applicatie
4 t/m 15	1 jun – 21 aug	Construction / Transition	- Iteratie 1 t/m 6 PDP ontwikkelen - Design maken - Documentatie bijhouden/bijwerken - Testen - PDP deployen naar testomgeving
16 t/m 17	24 aug – 4 sep	Afronding	- Extra ruimte voor uitloop - Afronden afstudeerverslag

Tabel 1: planning voor dit project

4.1.1 Verdiepen gamificatie

Ik heb me verdiept in gamificatie doormiddel van gesprekken met zowel de opdrachtgever als gamificatie expert. Verder heb ik een gamificatie workshop gevolgd, scripties, blogs en bestaande frameworks doorgenomen. De verschillende ideeën die ik heb bedacht m.b.t. gamificatie heb ik gepitcht bij collega's en de bij opdrachtgever om te peilen wat zij ervan vinden. Overigens heb ik ook tijdens de workshop gevraagd om ideeën. Mijn bevindingen heb ik voor een deel opgenomen in het game designdocument bijlage C. de rest wordt hieronder besproken om een globaal beeld te schetsen.

Gamificatie is oud onderwerp [5] dat vrij recent [6] in de spotlight is komen te staan. Het is tevens een heel breed onderwerp, dat ik niet volledig in dit verslag kwijt kan. Extra informatie is opgenomen in het game designdocument bijlage C.

In de laatste jaren is gamificatie erg gegroeid en zo zijn er steeds meer frameworks ontstaan. Ik heb voor dit project niet gekozen om een framework te gebruiken, maar in plaats daarvan heb ik meerdere frameworks geanalyseerd. Hieruit heb ik bepaald welke onderdelen relevant zijn voor dit project.

De relevantie heb ik bepaald met behulp van de gamificatie expert, hierbij hebben we gekeken naar: wat, hoe en waarom er bepaalde game elementen het PDP kunnen ondersteunen.

Als eerste bespreek ik wat gamificatie inhoudt en hoe het herkend kan worden. Vervolgens komt de user experience lifecycle [7] aan bod. Als laatste beschrijf hoe ik dit allemaal heb gecombineerd voor het PDP.

Gamificatie is het toevoegen van game elementen aan een niet game context. De game elementen kunnen van alles zijn, een aantal voorbeelden van veel gebruikte game elementen zijn:

- Leaderboards
- Badges
- Punten
- Achievements

Door game elementen toe te voegen wordt het voor de gebruiker 'leuker' [8] om een bepaalde taak uit te voeren. Een valkuil bij gamificatie is, dat er goed nagedacht moet worden over de keuze van game elementen. Games worden nagenoeg altijd vrijwillig gespeeld, als men het spel goed ontwerpt, door bijvoorbeeld een goede balans te vinden tussen interne en externe motivatie, wordt het spel sneller geaccepteerd door de gebruikers. Het toevoegen van game elementen zonder hier eerst over na te denken zal het gebruik van de applicatie niet doen toenemen.

In tabel 2 zijn de verschillende fases in een lifecycle van user experience beschreven.

Fase	Omschrijving
Discovery Phase	Iemand ontdekt je spel, hoe prikkel je hem om te spelen?
Onboarding Phase	Iemand is geprikkeld om je spel uit te proberen, hoe overtuig je hem dat hij hier meer tijd in moet stoppen?
Scaffolding Phase	Iemand is al wat langere tijd bezig met je spel. Hoe hou je hem bezig? Wat is zijn vooruitgang? Hoe laat je weten dat hij goed of slecht bezig is?
Endgame Phase	Iemand is een veteraan in je spel. Hoe zorg jij ervoor dat hij zich niet gaat vervelen?

Tabel 2: de verschillende fases in een lifecycle van user experience volgens

Gedurende de verschillende fases wordt er gespeeld met de intrinsieke en extrinsieke motivatie van de gebruiker. Het is lastig om het verschil tussen intrinsieke en extrinsieke motivatie te noemen. Dit kunnen onderzoek onderwerpen op zichzelf zijn.

Om dit beter toe te passen voor dit project verkleinen we de scope en kijken we naar interne, externe motivatie en motivatie op lang en korte termijn.

Motivatie op korte termijn prikkel je, door bijvoorbeeld badges te geven nadat de speler een goal heeft gehaald. Dit zijn externe prikkels.

Lange termijn motivatie aan de andere kant is wat lastiger. Hoe zorg je ervoor dat een speler terug blijft komen nadat hij of zij bekend is met het spel (na de onboarding fase). Als we specifiek kijken naar het PDP kan de motivatie zijn dat de speler aan zijn persoonlijke ontwikkeling bouwt. Als dit ook erkend wordt door bijvoorbeeld zijn teamleader, dan blijft de speler het spel gebruiken omdat het ook daadwerkelijk toegevoegde waarde heeft voor zijn carrière. Zo heb je een mix van interne (drang om te groeien) en externe (erkend worden door je teamlead) dat samen in theorie ervoor zorgen dat spelers op lange termijn gemotiveerd worden.

Voordat ik uitleg hoe ik het PDP ga gamificeren moet er eerst gekeken worden naar het DAD. Volgens het DAD moeten alle functionaliteiten in het te realiseren systeem een of meerdere doelstellingen ondersteunen. Als dat niet het geval is heeft het ook weinig toegevoegde waarde om de functionaliteit te implementeren, omdat de opdrachtgever daar geen vraag naar heeft. Met dit in gedacht ben ik gaan kijken hoe ik elke doelstelling kan realiseren met behulp van gamificatie.

Het gaat voor dit project om de volgende doelstellingen:

- Stimuleren kennisdeling
- Stimuleren kennisontwikkeling
- Verhogen gebruik PDP

Het verbruik van het PDP wil ik verhogen door de game ‘leuk’ te maken. Daarnaast moet het overal beschikbaar zijn omdat de kracht van gamificatie dan beter tot zijn recht komt. Als de gebruikers de game vrijwillig en overal kunnen spelen is de kans groter dat het spel aanslaat. Om een groter publiek aan te trekken moet de applicatie het liefst platform en locatie onafhankelijk zijn. Als het spel ‘leuk’ is, worden de medewerkers bewust van het PDP en stimuleert dit hun om het PDP te gebruiken.

De volgende punt is om de deling en ontwikkeling van kennis te bevorderen. Hierbij heb ik het volgende bedacht. Een random gegenereerde quiz ofwel in het spel ‘battle’ genoemd. De quiz wordt gebouwd door de community. De community bestaat uit de medewerkers van CGI. Elke medewerker heeft de mogelijkheid vragen te bedenken en deze toe te voegen aan het spel.

De vragen kunnen vervolgens tijdens de quiz beoordeeld en besproken worden door anderen. Met dit aspect van de game willen we de kennisdeling en ontwikkeling bevorderen. De gebruikers worden geprikkeld om nieuwe vragen te bedenken en bestaande vragen kritische te beoordelen (intrinsiek).

Verder krijgt de quiz een singleplayer en een multiplayer (speler tegen speler) component. De battle zou eventueel ook met een team gespeeld kunnen worden. Na het spelen van de battle krijgen de spelers een aantal punten en eventuele unlockables. De punten zorgen ervoor dat een speler kan stijgen en dalen op de leaderboard en doormiddel van unlockables willen we de spelers de mogelijkheid geven om het spel te personaliseren. Met unlockables bedoel ik achievements, badges, etc. De unlockables zijn bedoeld om de spelers op korte termijn vast te houden.

Daarnaast komt er een level en progressie systeem. De speler kan zijn level verhogen door zich te verdiepen in nieuwe vakgebieden. Hierbij wordt de progressie bijgehouden. Uiteindelijk is het de bedoeling dat de medewerkers gestimuleerd worden om zich te verbreden en specialiseren op nieuwe vakgebieden.

Als laatste is er nagedacht om een apart element voor teamleaders te bouwen waarin zij de statussen van zijn medewerkers kunnen bijhouden.

Om een beter overzicht te geven is alles gecombineerd in tabel 3.

Fase	Doelstelling	Motivatie	Game elementen
Discovery Phase	Ontdekking van het spel. Aantrekken om het te 'uit te proberen'.	Extern, kort termijn motivatie	• Naamgeving • Look and Feel • Korte termijn unlockables • Singleplayer
Onboarding Phase	Stimuleren kennisontwikkeling. Verhogen gebruik PDP.	Korte termijn motivatie	• Level systeem • Ranking systeem • Punten systeem • Unlockables • Multiplayer • Singleplayer
Scaffolding Phase	Stimuleren kennisdeling. Stimuleren kennisontwikkeling. Verhogen gebruik PDP.	Korte termijn met een overgang naar lange termijn motivatie	• Mogelijkheid tot toevoegen nieuwe vragen (feedback systeem), • Alle bovenstaande elementen • Betere unlockables • Team systeem
Endgame Phase	Stimuleren kennisdeling. Stimuleren kennisontwikkeling. Verhogen gebruik PDP.	Voornamelijk lange termijn	• Feedback system (vragen toevoegen/beoordelen, een soort van admin) • Alle bovenstaande elementen

Tabel 3: overzicht gamificatie van het PDP

4.1.2 Architectuur

Het laatste onderdeel van de PVA is de Technische strategie (bijlage B hoofdstuk 5). Binnen de technische strategie bespreek ik de architectuur en neem ik een besluit over het type applicatie dat ik ga ontwikkelen. De meeste informatie heb ik verkregen middels gesprekken met de opdrachtgever.

4.1.2.1 Context

Welke data moet er worden opgehaald

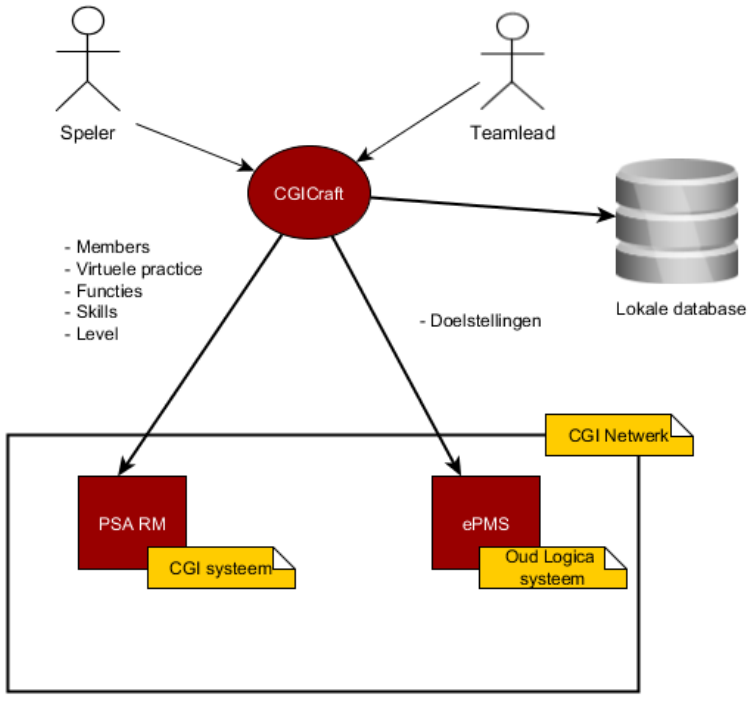
Ik ben begonnen met het vastleggen van alle data dat nodig is voor CGICraft en uit welke systemen ik deze kan ontsluiten. Aan de hand van de initiële user stories en gesprekken met de opdrachtgever is bepaald dat het gaat om de volgende data:

- **Members**: werknemers gegevens
- **Virtuele Practices**: Vakgebieden binnen CGI zoals software engineer, tester, architect, etc.
- **Functies**: Baan binnen een practice zoals, Junior Java Software Engineer (JAVA programmeur).
- **(PMS/OMS) levels**: Niveau van een medewerker. Een Level 1 heeft weinig ervaring en een level 5 heeft de veel ervaring.
- **Hard en soft skills**: Skills waar de member over moet beschikken binnen zijn functie.
- **Doelstellingen**: Wat de member wil bereiken. E.g. van Level 2 naar level 3 binnen 5 jaar.

De data die ontsloten wordt is gerelateerd met members. De bovenstaande data zal de basis worden voor CGICraft. Wat ik daarmee wil zeggen is dat we de medewerkers op een van deze vakken willen stimuleren.

Aan de hand van gesprekken met de opdrachtgever is vastgesteld waar de bovenstaande gegevens beschikbaar zijn. Het gaat om de volgende twee systemen ePMS en PSA RM. In figuur 3 is een context diagram geschetst, die de communicatie met deze systemen weergeeft.

Communicatie



Figuur 3: context diagram

De communicatie verloopt in een richting. Dat wil zeggen dat CGICraft alleen data ophaalt en geen data verstuurd naar de interne systemen. CGICraft heeft geen relevante data voor ePMS of PSA RM. Daarnaast gaat het om oude systemen. Het aanpassen van deze systemen zou een grote klus worden waarbij de vraag is of het de kosten en tijd waard zijn.

Tevens wordt de opgehaalde data niet opgeslagen in een eigen database. In plaats daarvan wordt het direct verwerkt tot relevante data voor de front-end. Dit wordt gedaan om de kwaliteit van het systeem op de volgende punten te verbeteren:

Onderhoudbaarheid, als de data van ePMS of PSA RM wordt opgeslagen in de database van CGICraft ontstaat er dataduplicatie. Het bewaren van de data integriteit wordt dan lastiger omdat er rekening moet worden gehouden met synchronisatie. Als de data real time wordt opgehaald kunnen deze problemen niet ontstaan.

Betrouwbaarheid, de data is als altijd correct (als in meest recent) omdat het real time wordt opgehaald.

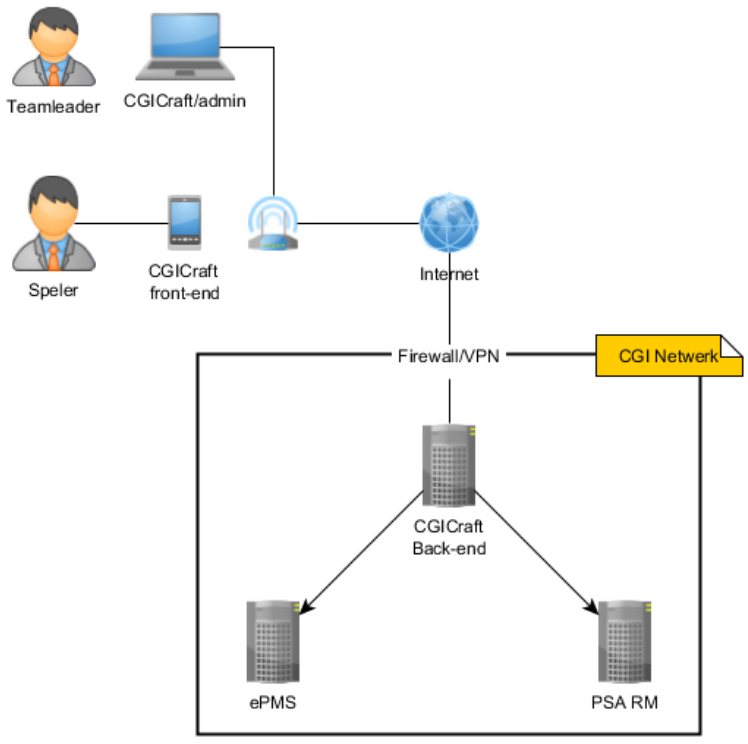
Security, de gedachtegang was als volgt: hoe minder interne data van CGI er wordt opgeslagen des te minder (data) er beveiligd moet worden.

Overigens zijn er ook nadelen. Een lokale database is sneller dan data real time ophalen uit een extern systeem. Hierbij kan het netwerk of de snelheid van het externe systeem een bottleneck zijn.

4.1.2.2 Deployement

Figuur 4 toont de architectuur voor CGICraft zoals hiervoor besproken. Hierbij haalt de backend real time de data op uit relevante systemen en wordt deze vertaalt voor de front-end. In de database worden alleen CGICraft specifieke gegevens opgeslagen.

Verder staat de backend binnen het CGI netwerk maar kan het wel van buiten worden benaderd. De front-end kan zowel een web app of mobiele app zijn.



Figuur 4: deployment diagram

Om te bepalen of de bovenstaande architectuur mogelijk is heb ik een gesprek gevoerd met een architect. Hieruit is duidelijk geworden dat deze architectuur niet mogelijk is. Er zijn namelijk een aantal beperkingen.

Het interne netwerk is niet direct beschikbaar van buiten. Als de backend binnen het CGI netwerk wordt geplaatst dan kan ik hier niet mee communiceren totdat ik ben ingelogd op het interne netwerk.

Daarnaast was het nog steeds niet duidelijk of CGICraft met de interne systemen mag communiceren, maar hier staan ook strikte eisen tegenover. Zo mag de data van de interne systemen niet naar buiten e.g. via een mobiele app.

4.1.2.3 Alternatieve architectuur

Na het gesprek heb ik alternatieve soorten architecturale oplossingen bedacht waarbij ik rekening houd met security eisen:

ID	Omschrijving
1	Een standalone applicatie waarbij de twee interne systemen gemocked worden zodat het niet conflicteert met security. Zo wordt de app ook future proof indien de organisatie beslist om de app intern te implementeren. Zowel de mock als CGICraft worden in dit geval buiten het netwerk geplaatst.
2	Een applicatie binnen het netwerk van CGI waarbij de gebruikers het spel alleen kunnen spelen als ze verbonden zijn op het interne netwerk van CGI (tijdens werk).
3	Een standalone applicatie waarbij de backend binnen het netwerk staat en deze verbonden is met de relevante systemen. De front-end staat buiten het netwerk en ontvangt alleen real time gemanipuleerde data (met behulp van een proxy) die beter past bij CGICraft, hiermee wordt de integriteit van de interne data bewaard.
4	Een standalone applicatie waarbij de gebruikers moeten registreren. Dit staat geheel buiten CGI en kan alleen problemen opleveren vanuit het marketing oogpunt.

Tabel 4: verschillende soorten architecturale oplossingen

Interne plaatsing

In principe moet ik eerst kiezen of ik de applicatie intern of extern geplaatst wordt. Bij optie 2 en 3 wordt CGICraft binnen het interne netwerk geplaatst. Het voordeel hiervan dat de applicatie officieel deel uitmaakt van CGI en ook aan haar eisen voldoet. Daarnaast is er communicatie mogelijk met andere interne system.

Mijn voorkeur gaat uit naar optie 3. Optie 3 is de situatie zoals al eerder besproken. Het verschil is, dat de front-end ook van buiten het CGI netwerk beschikbaar is (met behulp van een proxy).

Het nadeel is dat er tijdens de ontwikkeling rekening moet worden gehouden met security, dit kan erg limiterend zijn voor de creativiteit. Daarnaast is het niet duidelijk of een proxyverbinding mogelijk is, hierdoor kan CGICraft alleen tijdens werk gespeeld worden. Dit is niet gunstig.

Externe plaatsing

Dat brengt ons bij extern. Als CGICraft in zijn geheel buiten het netwerk van CGI wordt geplaatst, ontstaat er meer vrijheid met wat er mogelijk is. Bij optie 4 en 1 wordt CGICraft extern geplaatst. Optie 4 heeft een aantal voordelen ten opzichte van optie 1.

Ten eerste moet er minder ontwikkeld worden. Het mocken van twee systemen kan erg lastig zijn en zelfs onnodig. Daarnaast kost het veel tijd om de twee systemen te mocken terwijl de toegevoegde waarde hiervan laag is. In plaats van mocken kun je beter rekening houden dat er een externe verbinding kan komen. Als het design goed is uitgedacht, dan zou dit geen groot probleem moeten zijn.

Ten tweede kun je veel meer creativiteit kwijt bij optie 4 omdat je niet gebonden bent aan de eisen van de mock of CGI. Het nadeel is echter dat als CGICraft verplaatst wordt naar het intern netwerk, dan kan dit conflicten opleveren vanuit het security oogpunt. Dat gezegd hebbende wegen de nadelen niet op tegen de

voordelen. Doordat optie 4 een groter raakvlak heeft is de kans ook hoger dat dit project een succes wordt. Met een groot raakvlak bedoel ik het aantal mensen, dat dit systeem in de eerste instantie kunnen uitproberen.

Keuze

Uiteindelijk is ervoor gekozen om eerst een standalone applicatie op te zetten (optie 4) en deze eventueel in de toekomst te verbinden met de relevante systemen. Hier is voor gekozen omdat er tegen het einde van de inception fase nog steeds onduidelijk was of er nog een mogelijkheid is om vanaf het internet verbinding te maken met het interne netwerk en of de systemen ePMS en PSA RM benaderd mogen worden. Hierdoor vallen veel mogelijkheden weg. De opdrachtgever is nagegaan of er alternatieven zijn zodat de applicatie alsnog verbinding mag/kan maken met ePMS en PSA RM, alleen is de communicatie vanuit security en de betrokkenen langzaam verlopen. Op een standalone applicatie heeft CGI weinig invloed en kan ik gewoon beginnen aan de construction fase. Als de applicatie al bij een aantal medewerkers aanslaat kan er uiteindelijk voor gekozen worden om het te integreren in het interne netwerk.

Het nadeel is echter dat de data handmatig moet worden ingevuld. Voor de eerste versie hebben we besloten om alleen de practice ‘Software engineer’ uit te werken en vandaar uit langzaam op te bouwen.

Type applicatie

Naast de architectuur heb ik ook bepaald wat voor type applicatie er gebouwd gaat worden. De keuzes waren:

- Web – applicatie (website)
- Mobiele – app
  - Cardova
  - Phone-gap
  - Unity
  - Native Android
  - Native IOS

Ik heb desktopapplicaties niet opgenomen als potentiële optie omdat deze erg ouderwets zijn, en niet passen bij dit project.

Bij het kiezen van het type applicatie heb ik gelet op de volgende punten:

- Look and feel, de app is modern en kan bijvoorbeeld animaties bevatten. De gekozen platform moet dit ondersteunen. De medewerkers moeten geprikkeld worden om de applicatie uit te proberen.
- Gebruikerservaring, het gekozen platform moet een fijne gebruikerservaring leveren. Een medewerker kan bijvoorbeeld zonder al te veel moeite gebruik maken van de applicatie.
- Beschikbaarheid, het liefst is de applicatie platform onafhankelijk, zodat zoveel mogelijk medewerkers de applicatie kunnen gebruiken.

Webapplicatie

Ten eerste kijk ik naar webapplicaties. Beide type applicaties hebben zijn voor en nadelen. Een webapplicatie is makkelijker te ontwikkelen en is overal beschikbaar, dus van mobiel tot tablet en pc tot Mac. En tegenwoordig kan er ook aardig wat animaties uitgevoerd worden met behulp van HTML5. Het nadeel is echter dat je altijd een internetverbinding nodig hebt en er dus geen mogelijkheid is om offline te spelen.

Mobiele applicatie

Een alternatief is mobiel. Net als bij webapplicatie is de app overall beschikbaar, maar ook zonder internetverbinding. Echter hangt dit wel af van het ontwerp. Het voordeel van een mobiele app is dat de gebruikerservaring beter is dan met een webapplicatie. Daarnaast is de instapdrempel voor apps erg laag dus mensen zijn sneller geneigd een app te downloaden dan zicht te registreren voor een website. Dit is te zien in snelle groei van de mobiele markt. Een webapplicatie kan ontworpen worden als een app, maar de native ervaring is altijd beter, vooral bij spellen.

Keuze

Ik heb gekozen voor mobiele app op basis van de context. In veel situaties is een webapplicatie een beter alternatief dan een mobiele app. Puur omdat het makkelijker is om te ontwikkelen en simpeler is om te beheren met dezelfde functionaliteiten. Maar dat is voor dit project niet het geval, er zullen vrijwel zeker animaties/game elementen toegevoegd worden. De native ervaring van een mobiele applicatie zorgt voor een betere gebruikerservaring.

Daarnaast heb ik bedacht waar de applicatie hoogstwaarschijnlijk gespeeld gaat worden. Ook hierbij is een app de betere keuze. De app of webapplicatie zal niet tijdens werk gespeeld worden, omdat men dan aan het werk is. De app is meer casual en heeft een lagere instapdrempel, je kunt de app op de bank of in de trein spelen op je mobiel, terwijl je een webapplicatie liever op een desktop of laptop opent. Veel medewerkers werken de hele dag achter de pc, een mobiele app is dan een prettig alternatief.

Als laatste wordt een app sneller geassocieerd met een spel dan een web app. De tijd dat web games populair waren is al lang voorbij e.g. flash games. Wel zijn er nog casual games op dit gebied als je kijkt naar Facebook. Maar vergeleken met de mobiele markt als gaming platform is het een stuk minder [9].

Door associatie met werk zo laag mogelijk te houden wil ik de medewerkers stimuleren om de app uit te proberen. De associatie bestaat wel, maar alleen binnen de app.

Platform

Wat betreft het platform (Xamarin, Cardova of native) is er gekozen voor een native app omdat dit simpelweg een betere ervaring levert (mede dankzij de applicatie gezien kan worden als game). Cardova gebruikt HTML5 voor crossplatform dit maakt het managen van de applicatie lastig is. Daarnaast zit er een zekere leercurve aan Cardova.

Xamarin is wel native maar kost geld. CGI heeft nog geen contract met Xamarin (daar waren ze wel mee bezig, maar dat kregen ze niet op tijd geregeld) en dus valt Xamarin af.

Verder hadden we niet gekeken naar andere alternatieven. Ik had de meest populaire platforms gepakt omdat hierbij de bugs bekend zijn en veel informatie hierover te vinden is. Wat ervoor moet zorgen dat de development soepeler verloopt.

IOS is de gewenst platform omdat het veiliger is dan Android en om deze reden wordt het ook officieel ondersteund door CGI. Dat wil niet zeggen dat dit de enige platform is dat gebruikt wordt door de medewerkers van CGI.

Ik heb uit eindelijk gekozen voor Android omdat ik geen IOS device of Macbook tot mijn beschikking heb wat wel nodig is voor IOS development daarnaast heb ik ook geen ervaring met IOS development.

In tegenstelling tot IOS heb ik wel een Android device en ook ervaring met Android development. Hierdoor kan ik snel beginnen en eventueel kwalitatief een beter product leveren. Als de Android app een succes wordt, kan er altijd voor gekozen worden om een IOS versie te bouwen.

Back-end

Voor de back-end is gekozen voor Spring. Spring is een veelgebruikte framework en is een standaard binnen DC Java.

4.1.3 Identificeren Requirements

De requirements zijn opgenomen in bijlage b hoofdstuk 4. Het identificeren van de requirements was lastig. Dit komt omdat we op dat moment niet wisten wat voor applicatie er gebouwd gaat worden en hoe gamificatie hierbij toegepast kan worden. Voor de requirements is het volgende uitgevoerd.

Ten eerste hebben we doelstellingen opgesteld. Vervolgens hebben we gekeken hoe de doelstelling bereikt kunnen worden doormiddel van gamificatie. Om dit te beantwoorden heb ik mij verdiept in gamificatie. Daarnaast heb ik ook gekeken naar het type applicatie en de architectuur. Hieruit is een lijst met potentiele game elementen geproduceerd die ook rekening houdt met de architectuur.

De game elementen zijn abstract omdat we niet precies weten hoe we dit goed moeten implementeren. De game elementen heb ik vervolgens omgezet naar user stories. Dit heb ik gedaan doormiddel van brainstormsessies met de opdrachtgever en gamificatie expert. Als voorbeeld pakken we het game element *level systeem* dit element heb ik gekozen om de spelers een gevoel van vooruitgang te geven die tevens het level systeem binnen CGI representeert. Het level systeem is erg abstract hierbij hebben we de volgende vragen proberen te beantwoorden: wat willen we met het level systeem bereiken en hoe ziet de koppeling met andere game elementen eruit? Hieruit zijn de user stories US3.4 en US3.5 ontstaan.

Ik ben voorzichtig geweest met het specificeren van de user stories en heb deze abstract gehouden omdat zowel de opdrachtgever en ik niet weten wat zal werken. Na het deployen van de app zal aan de van de gebruikers feedback de applicatie worden aangepast. De kans is dus groot dat de user stories zullen veranderen. Het idee is om de specificatie van de functionaliteiten over te laten aan de gebruikers maar ze wel alvast een globaal beeld te geven. Verder worden user stories per iteratie bekeken en eventueel verbeterd indien dat nodig is. Met de gedachte dat we dan meer ervaring hebben.

Tevens heeft elke user story een prioriteit, zodat er bepaald kan worden welke functionaliteiten cruciaal zijn voor een succesvol product. De prioriteit is bepaald aan de hand van de MoSCoW-methode. De verschillende niveaus zijn toegekend aan de user stories aan de hand van de kennis die is opgedaan uit de meetings met de opdrachtgever. MoSCoW kent vier niveaus:

Must: deze user stories moeten terugkomen in het eindresultaat en zijn dus cruciaal voor het succesvol afronden van het project.

Should: deze user stories zijn gewenst maar het product kan er ook zonder.

Could: deze user stories zullen aan bod komen als er genoeg tijd is.

Would: deze user stories zullen in dit project niet aan bod komen, maar kunnen interessant zijn bij een vervolgproject.

ID	Doelstelling	Omschrijving	Prioriteit
DS1	**Stimuleren kennisdeling**		
US1.1		Als een gebruiker kan ik nieuwe vragen en antwoorden toevoegen aan het systeem.	M
US1.2		Als een gebruiker kan ik mijn profiel aanpassen en delen, zodat ik mijn voortgang kan laten zien aan mijn collega's.	M
US1.3		Als een gebruiker kan ik zoeken naar een team, waar ik lid van wil worden.	S
US1.4		Als een gebruiker kan ik lid worden van een team, om teambattles te spelen.	S
DS2	**Stimuleren kennisontwikkeling**		
US2.1		Als een gebruiker kan ik een nieuwe (quiz)battle starten, zodat ik kan stijgen/dalen in ranking.	M

ID	Doelstelling	Omschrijving	Prioriteit
US2.2		Als een gebruiker kan ik zien hoeveel spelers een reactie hebben geplaatst bij een vraag, zodat ik een beeld krijg bij de populariteit van de vraag.	M
US2.3		Als een gebruiker kan ik een reactie plaatsen of reageren op een bestaande reactie, zodat ik de vraag kan bespreken.	M
US2.4		Als een gebruiker kan ik de vraag beoordelen met een 'like', zodat de vraag vaker terugkomt in battles voor andere spelers.	M
US2.5		Als een gebruiker kan ik uit een overzicht van goals kiezen, zodat ik mijn eigen goals kan bepalen.	M
DS3	**Verhogen gebruik PDP**		
US3.1		Als een gebruiker kan ik mijn ranking zien op de leaderboard, zodat ik kan zien waar ik sta tegenover de rest.	M
US3.2		Als een gebruiker kan ik de globale ranking zien.	C
US3.3		Als een gebruiker kan ik badges verdienen door battles succesvol af te ronden, zodat ik mijn badges kan toevoegen aan mijn avatar.	M
US3.4		Als een gebruiker wil ik mijn avatar level verhogen door middel van battles, zodat ik nieuwe badges en nieuwe goals kan unlocken.	M
US3.5		Als een gebruiker kan ik mijn individuele skills levelen door middel van battles, zodat ik mijn speler level kan verhogen.	M
US3.6		Als teamleader kan ik de profielen van mijn members (spelers) zien zodat ik met deze gegevens rekening kan houden tijdens een evaluatiegesprek met de member.	C
US3.7		Als een teamleader kan ik een overzicht zien van al mijn members.	C
US3.8		Als een team kunnen we een nieuwe battle starten, zodat we gezamenlijk in rank kunnen stijgen of dalen.	S
US3.9		Als gebruiker kan ik stijgen en dalen op de leaderboard.	M
US3.10		Als gebruiker kan ik inloggen, zodat ik het spel kan spelen.	M

Tabel 5: initiële lijst van user stories

Tot slot heb ik samen met de opdrachtgever alle user stories geverifieerd. Daarnaast heb ik een kopie meegegeven aan de gamificatie expert om te kijken of de functionaliteiten logisch zijn en passen bij de bijbehorende game element.

Naast user stories zijn er ook niet functionele requirements opgenomen in de vorm van constraints. Ook bij dit onderdeel heeft gamificatie en architectuur veel invloed gehad.

4.1.4 Huidige mock-up

De huidige mock-up is gemaakt door een medewerker van CGI met Unity. De mock-up was bedoeld om te laten zien dat gamificatie toegepast kan worden voor het promoten van het PDP systeem. De mock-up is gebaseerd op het spel World of Warcraft (WoW). Dit was gepitchet bij de opdrachtgever en heeft geleid tot dit project.

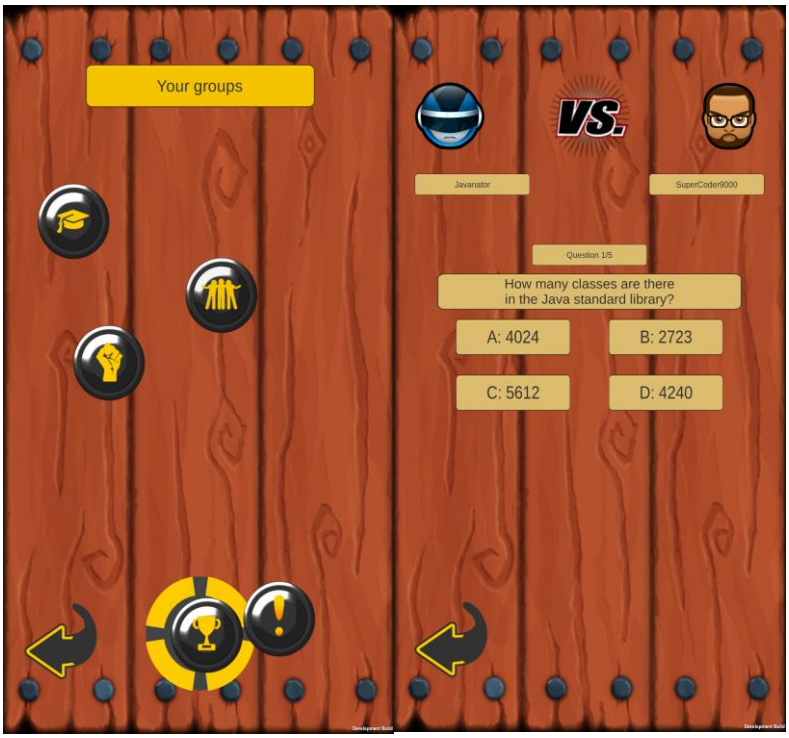
De mock-up bevat geen functionaliteiten. Er kan alleen genavigeerd worden door de verschillende schermen, hierbij was het doel om een gevoel te creëren van wat er allemaal mogelijk is. De mock-up laat tevens de verschillende onderdelen van het PDP terugkomen in het spel zodat de opdrachtgever een koppeling kan leggen tussen het spel en het PDP.

In figuur 5 en 6 is de mock-up te zien. Het eerste wat opvalt, is dat de look and feel sterk op een game lijkt. Dit was nodig om de kracht van gamificatie beter tot zijn recht te laten komen. De opdrachtgever

moet geprikkeld worden door de mock-up. Als dat niet het geval is, wordt er ook geen project gestart. In dat opzicht heeft de mock-up zijn doel bereikt. Echter, de opdrachtgever en ik zijn beide erover eens dat de mock-up niet representatief is voor het uiteindelijke systeem. Het is te cartoony, hierdoor past het niet goed bij onze doelgroep en CGI in het algemeen.

Figuur 5: selecteer Practice (links), Start screen (rechts)

Als we kijken naar de doelgroep wordt het snel duidelijk dat iedereen gemotiveerd en erg bezig is met zijn carrière. De meeste medewerkers zijn naar verwachting ook geen gamers. De uiteindelijke game zal wat serieuzer uitzien maar zal nog steeds game elementen bevatten.



Figuur 6: Home en Battle scherm

Naast de look and feel zullen er een aantal elementen uit de mock-up terugkomen in de applicatie. De naam van de mock-up is CGICraft, dit wordt overgenomen zodat er een duidelijk onderscheid is tussen het PDP en de game. Daarnaast wordt er verder gebouwd op de game elementen die zijn geïntroduceerd binnen de mock-up zoals: battle(quiz) en ranking.

Qua design zal er niet veel worden gedaan. Ik heb geen designers tot mijn beschikking en mijn kennis met design tools is erg minimaal. Dit is een beperking voor dit project. Ik ben immers geen designer maar een engineer.

5 Construction fase

5.1 Iteratie 1

5.1.1 Iteratie planning sessie

De iteratie begon met een iteratie planning meeting. De goal van de meeting was om te bepalen welke taken er uitgevoerd gaan worden voor deze iteratie. Hierbij is rekening gehouden met het DAD methode en de initiële work item list dat opgesteld was tijdens de inception fase.

Volgens het DAD methode is het belangrijk om de architectuur zo vroeg mogelijk te bewijzen, zo voorkom je namelijk problemen in de toekomst bijvoorbeeld: een verbinding met een externe systeem dat niet mogelijk blijkt te zijn. Voor dit project bewijs ik de architectuur door een connectie te leggen tussen de database, front- en backend. De backend kan nog eventueel verbinding maken met andere interne systemen. Dit heb ik voorgesteld aan de opdrachtgever en hij was er mee eens.

Daarnaast hebben we tijdens de meeting besloten om als eerste een registratie en inlog pagina te bouwen. Voor de eerste versie willen we de data dat binnen CGI gebruikt wordt implementeren. De gebruiker kan tijdens de registratie zijn practice en functie kiezen en vervolgens zijn member gegevens invoeren. Practice, Functie en Member data vormen de basis applicatie.

Verder moet ik ook de ontwikkelomgeving opzetten. In tabel 6 vind je een globale lijst met taken en de bijbehorende omschrijving.

Taak	Omschrijving
Architectuur bewijzen	Verbinding maken tussen de database, front- en backend - Eerste versie bouwen van de front-end - Eerste versie bouwen van de backend - Eerste versie bouwen van de database
Ontwikkel omgeving opzetten	- Laptop configureren - Benodigde software instaleren - Repository opzetten
Registratie/login bouwen	Een nieuwe speler moeten zich kunnen registreren en bestaande spelers moeten kunnen inloggen.

Tabel 6: samenvatting taken iteratie 1

5.1.2 Opzetten ontwikkelomgeving

De laptop die ik van CGI heb gekregen komt standaard geïnstalleerd met een aantal software zoals Microsoft Office. Vanwege security eisen raad CGI mij aan om het Software Approval Register (SAR) raad te plegen bij het installeren van nieuwe software. SAR is een portal waarin alle goedgekeurde software zijn opgenomen.

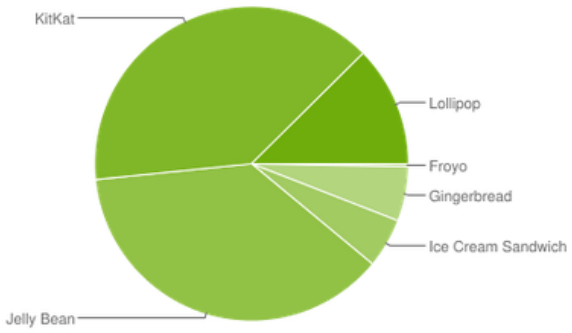
De front-end ontwikkel ik in Android en de backend in Spring.

Spring kent verschillende soorten projecten. Ik heb gekozen voor Spring Boot, zodat ik direct aan de slag kan. Spring boot is redelijk nieuw en heeft als doel om het opzetten en configureren van nieuwe projecten te automatiseren zonder dat het ten koste gaat van de functionaliteit.

Android biedt ook een aantal keuzes tijdens maken van een nieuw project. De belangrijkste hiervan is: welke API wil ik ondersteunen? Op het moment van schrijven zag de distributie van de verschillende Android versies er als volgt uit:

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

Data collected during a 7-day period ending on June 1, 2015.
Any versions with less than 0.1% distribution are not shown.



Figuur 7: distributie Android versies

Ik heb gekozen voor een minimale API van 15 en een target API van 22. Met minimale API van 15 heb ik een dekking van 90%. Alles wat voor API 15 valt, is erg verouderd. Google heeft een grote omslag gemaakt van Gingerbread naar Ice Cream Sandwich vandaar ook de grote sprong in versie nummers.

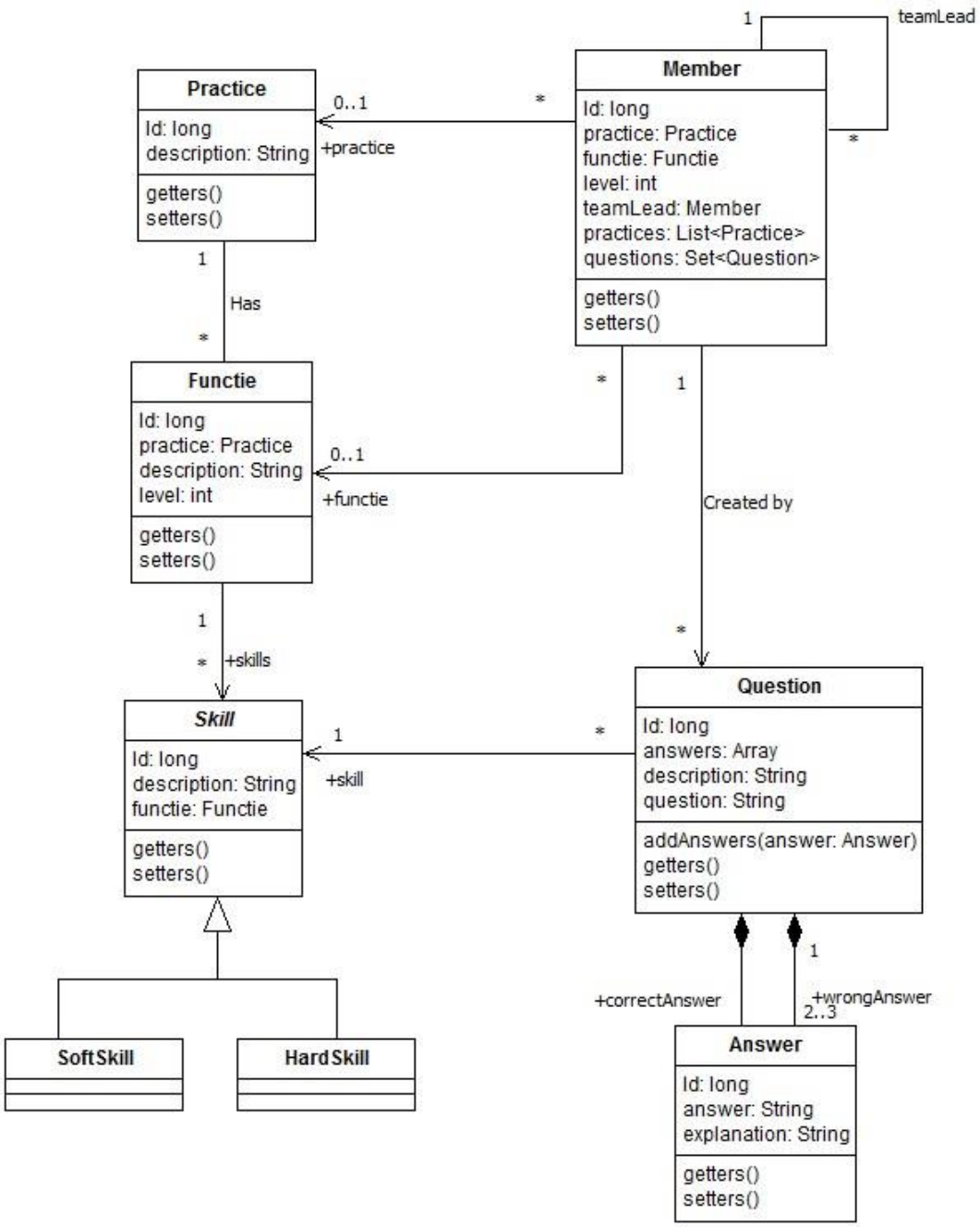
Voor de database heb ik gekozen voor MySQL omdat ik daarmee bekend ben. De onderliggende database heeft in mijn ogen weinig invloed op de applicatie, aangezien er een persistentie laag tussen zit. Daarnaast heeft MySQL geen grote na- of voordelen om hierover te discussiëren, ook zijn er voor de database geen eisen opgelegd vanuit CGI. Wel werd MySQL erkend als een van de database die kan gebruiken.

Naast de database heeft een admin een repository opgezet voor dit project, waar ik gebruik van mag maken. Het betreft een SVN repository die intern draait. Als laatste heb ik een aantal ondersteunende tools geïnstalleerd zoals Staruml. Alle tools heb ik voor het installeren gecontroleerd in SAR.

5.1.3 Eerste opzet CGICraft

De applicatie kent een database, front- en backend. In figuur 8 zie je het klassendiagram van de front-end voor iteratie één. Het klassendiagram is gemaakt met de kennis die ik heb opgedaan tijdens de inception fase. Hierbij heb ik alle informatie die nodig is voor een initiële opzet toegevoegd. Tevens heb ik het klassendiagram laten controleren door de opdrachtgever om te kijken of de koppeling tussen de verschillende entity's klopt.

De extra klassen Skill en Question worden tijdens deze iteratie niet geïmplementeerd, maar zijn alvast opgenomen om een beter beeld te creëren voor de volgende iteratie.



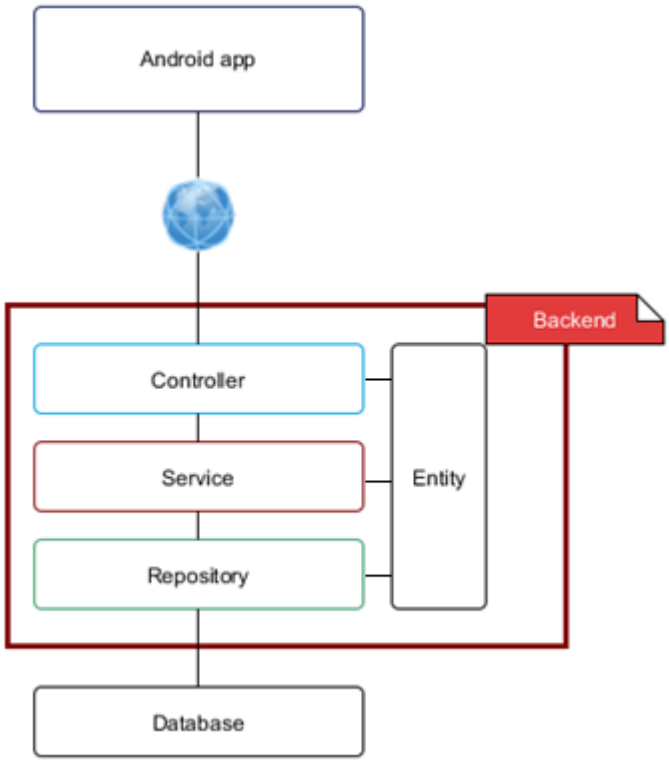
Figuur 8: klassendiagram iteratie één

Het klassendiagram representeert in die zin ook het domein. Ik had hiervoor een aparte domein model kunnen maken maar in mijn optiek zou dat geen extra toegevoegde waarde hebben. Het domein voor dit project is niet al te groot en is dus duidelijk genoeg met alleen het klassendiagram.

Het eerste wat een nieuwe gebruiker doet is zichzelf registreren. Hierbij heb ik bedacht om de registratie onder te verdelen in drie schermen om zo beter het overzicht te bewaren voor de gebruiker. In het eerste scherm kiest de gebruiker zijn practice, in het tweede scherm zijn functie en in het laatste scherm vult hij zijn member/gebruikers gegevens in. Voor de registratie moet de front-end, practice en functie gegevens ophalen uit de backend.

Aan het einde van de inception fase werd het duidelijk dat communicatie met interne systemen niet mogelijk was, daardoor moet de database voorlopig handmatig worden gevuld. In de eerste instantie hebben de opdrachtgever en ik besloten om alleen de practice ‘software engineer’ uit te werken. De database heb ik gebouwd aan de hand van het klassendiagram. Hierbij heb ik drie tabellen gemaakt met de bijbehorende attributen. Het gaat om de tabellen Functie, Practice en Member, deze heb ik vervolgens gevuld met test data dat de practice ‘software engineer’ representeert. Dit is nodig zodat de gebruiker ook daadwerkelijk wat kan kiezen tijdens de registratie zoals te zien in figuur 10. Hetzelfde heb ik uitgevoerd voor functie.

Voor de backend heb ik spring boot gebruikt. Dit zorgde ervoor dat de opzet vrij soepel verliep. De architectuur van de backend ziet er als volgt uit:



Figuur 9: architectuur back-end

Dit is de standaard architectuur van Spring. Functie, Practice en Member zijn allemaal een entity. Deze zijn gekoppeld aan een tabel op basis van de naam. Een klasse met de naam Practice wordt gekoppeld aan een tabel practice. De namingconventie wordt afgehandeld door Spring. Elke entity heeft een service, repository en controller. De app communiceert met de backend doormiddel van een Rest API die aangeboden wordt door de controller. Er is gekozen voor Rest omdat dit simpel, leesbaar en lichtgewicht is vergeleken met bijvoorbeeld SOAP.

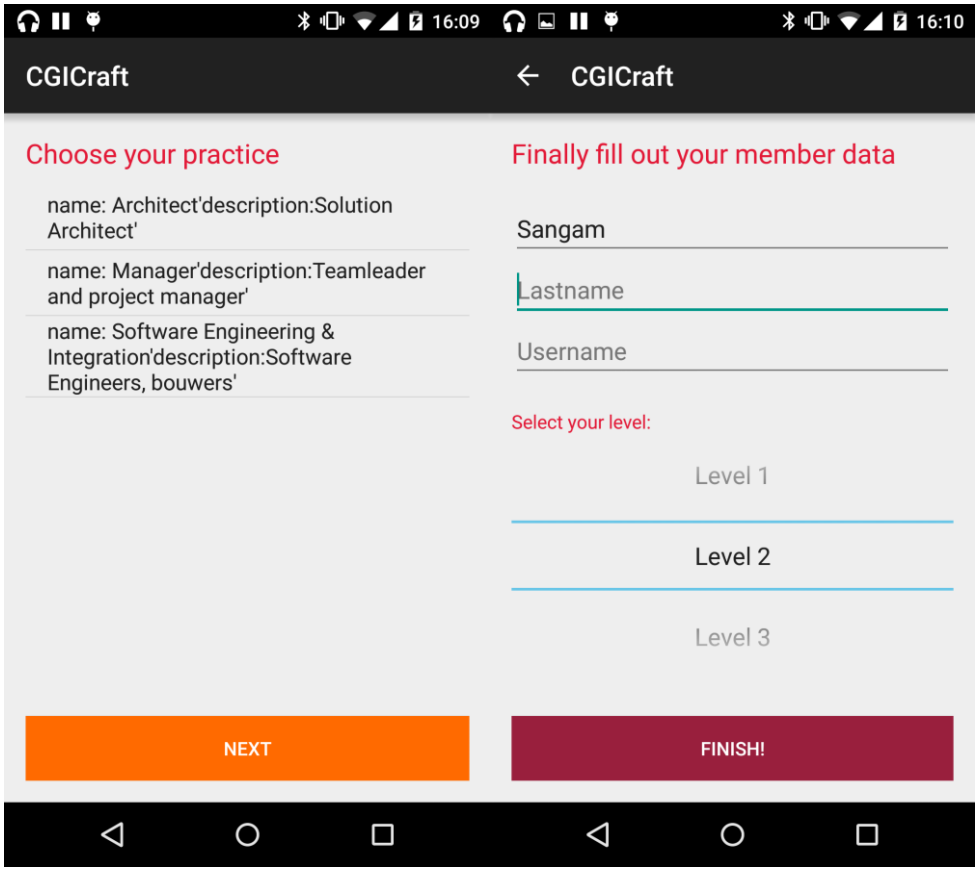
De app is gemaakt volgens het klassendiagram. Voor de communicatie van de app naar de backend wordt gebruik gemaakt van de ingebouwde Http client. Http client is een onderdeel van het Android SDK maar is sinds de nieuwste API depracted en wordt dus niet meer ondersteund door Google. Voor iteratie één heb ik het gelaten zoals het is. Vanwege tijd tekort.

Voor het parsen van JSON maak ik gebruik van Jackson. Jackson staat bekend om zijn hoge performance [10] bij grote file sizes, maar het verschil zal in de werkelijkheid niet merkbaar zijn. Gson een andere populaire parser zou een alternatief kunnen zijn voor Jackson. Ik heb gekozen voor Jackson omdat ik daar

beter bekend mee ben dan Gson, daarnaast zijn de verschillen tussen Gson en Jackson niet relevant voor dit project. Het gaat om een klein project en de keuze zou geen invloed moeten hebben.

Jackson en Gson kennen wel beide een aantal voordelen over handmatig parsen. Ten eerste is het sneller [11]. Ten tweede wordt de parsing van JSON naar Object automatisch geregeld, wat ertoe leidt dat ik aanzienlijk minder code hoef te schrijven. Dit is goed voor de onderhoudbaarheid en maakt testen ook eenvoudiger.

Een probleem dat ik tegenkwam is dat mijn mobiele telefoon op een ander netwerk zit als mijn laptop. Communicatie tussen de server en app is dus niet mogelijk. Dit heb ik opgelost met behulp van Chrome doormiddel van port forwarding. De poort waar de server op draait wordt opengezet op mijn mobiel via een USB verbinding. Dit is een tijdelijke oplossing, je bent namelijk afhankelijk van een USB kabel. Hierdoor kan de opdrachtgever de app niet testen op zijn eigen mobiel.



Figuur 10: registratie schermen

5.1.4 Testen

Gedurende deze periode heb ik gebruik gemaakt van exploratory testing. Tijdens het programmeren heb ik constant getest of de applicatie doet wat het moet doen. Voor deze iteratie gaat het om een eerste opzet waarbij ik de architectuur bewijs. Er zijn dus geen expliciete game elementen toegevoegd. De gebruiker kan zich registreren en zijn gegevens vervolgens inzien in de app. Ik heb dit getest door een aantal nieuwe gebruikers aan te maken en te kijken of de data (correct) wordt opgeslagen in de database. Vervolgens heb ik getest of de data wordt opgehaald tijdens het inloggen. Het inloggen bevat geen security dit is bedoeld voor latere iteraties. Het heen en weer sturen van data tussen de app, backend en de database bewijst dat de gekozen architectuur werkt.

5.1.5 Reflectie

Deze reflectie gaat zowel over iteratie 1 als de inception fase. Hierbij heb ik een aantal opmerkingen.

Ten eerste, de gekozen platform is helaas niet IOS. Eén van de reden hiervoor was omdat er geen IOS-device beschikbaar was of een Macbook waar ik op zou kunnen programmeren. Dit is vreemd omdat IOS een ondersteund platform is. Je zou zeggen dat er Macbooks en IOS devices beschikbaar zouden zijn. Dit was jammer, want bij IOS zou ik direct goede ondersteuning krijgen vanuit CGI terwijl het nu buiten CGI valt. Aan de andere kant is het ook een beetje raar als ik met C zou werken in DC JAVA. De software ontwikkelstraat is opgezet voor JAVA en Windows. Ik kan hier dus sneller mee aan de slag. Aangezien het om een klein project gaat en ik dit meer zie als een proof of concept heeft het voorlopig geen grote impact, maar dan nog lag de voorkeur bij IOS.

Ten tweede, CGI is opgedeeld in 10 practices. Deze practices bevatten vervolgens allemaal functies en skills. Het probleem hierbij is dat de practices, functies en skills niet zijn gestandaardiseerd. Elke practice wordt op een iets andere manier beschreven en omvat weer iets andere data. Dit zorgt ervoor dat je niet snel kunt overstappen tussen practices wat normaal ook niet de bedoeling is maar bij CGICraft wel. Een medewerker kan nieuwe practices ‘uitproberen’. Daarnaast zijn niet alle practices even goed uitgewerkt/ actief. Dit is een gemiste kans in mijn ogen. Als de practices op een gestandaardiseerde manier werden beschreven, dan weet je welke informatie je kan verwachten. Dit is een probleem voor dit project als we kijken naar de database. We weten niet welke informatie belangrijk is en hoe we dat allemaal goed kunnen opslaan. CGI is overigens wel bezig met een omzetslag ze willen alles standaardiseren. Dit leidt mij tot het volgende punt, als CGI bezig is met een omzetslag is de vraag hoe lang en of dit project nuttig zal zijn. De omzetslag kan overigens wel jaren duren. Het gaat immers om een groot bedrijf met veel data.

Verder mis ik een definition of done (DoD). Dit is op het moment impliciet maar het is beter om dit explicitiet op te nemen in het PVA. Op het moment zijn er wel acceptatie criteriums echter zijn deze voor het project en niet toepasbaar per iteratie. Met een DoD heb ik ook een deel kwaliteitsbewaking.

Naast het ontwikkelen vind ik de gekozen ontwikkelmethode verassend goed werken voor dit project. Ik heb wel een aantal opmerking over deze methode. De eerste is dat het DAD aangepast kan worden als de context erom vraagt. Dit maakt de methode enorm flexibel. Hierbij vraag ik mij dan af hoe specifiek de methode is voor softwareontwikkeling. Verder wordt er veel nadruk gelegd op architectuur. Dit onderdeel is voor dit project vrij klein uitgevallen. En als verder kijken leent het ontzettend veel onderdelen van SCRUM. Dit werkt goed omdat ik elke game element goed kan indelen in iteraties. Zo is er een duidelijk plan wat betreft deadlines en milestones. Als laatste vallen veel onderdelen weg. Dan vraag ik me af een ontwikkelmethode echt nodig is bij een één man project. Dat wil niet zeggen dat er geen onderdelen geleend kunnen worden uit bestaande ontwikkelmethodes. Maar eerder dat je het geen naam geeft waardoor er verwarring kan ontstaan.

Tot slot vond ik het jammer dat we geen verbinding kunnen maken met het huidige PDP vanwege security. Het gaat om een relatief groot bedrijf en beveiliging is belangrijk, hier kan ik niet omheen en dit was ook bekend. Als er iets moet worden uitgevoerd dan moet dat door meerder mensen worden goedgekeurd.

5.2 Iteratie 2

5.2.1 Iteratie planning sessie

Iteratie één eindigde met een meeting waarin we een aantal zaken hebben besproken.

Terugkoppeling iteratie één

De meeting begon met een demo en terugblik op iteratie één. Wat ging er goed en wat ging er fout? Om dit kort samen te vatten, iteratie één verliep over het algemeen goed. Er waren geen grote problemen en alle vooraf gestelde goals had ik gehaald.

De demo was ook goed ondanks het feit dat er weinig te zien was. Een al eerder besproken punt is dat de app niet kan communiceren met de server tenzij de mobiele telefoon verbonden is met de laptop waar de server op draait. Dit heeft als gevolg dat de opdrachtgever de app niet zelf kan testen. Het nadeel hiervan is dat de opdrachtgever geen of weinig feedback kan geven over de app zelf. Om dit probleem op te lossen gaat de opdrachtgever kijken of er een server beschikbaar is waarop de backend kan draaien, zodat hij ook toegang heeft tot de app.

Taken iteratie twee

Voor iteratie één lag de nadruk op het bewijzen van de architectuur en met iteratie twee is de nadruk gelegd om iets speelbaars te bouwen. De conclusie die ik al snel trok (en waar de opdrachtgever ook mee eens was), is om een singleplayer element te bouwen. Met singleplayer wordt bedoeld dat de speler tegen zichzelf speelt. In iteratie twee wordt er dus een singleplayer battle gebouwd.

Naast battle moet ik ook kijken naar de navigatie en de look and feel van de app. De mock-up zoals te zien in figuur 6 is niet representatief voor de doelgroep, waardoor er enige verwarring is ontstaan bij mij. Zo is het niet duidelijk wat voor sfeer we willen creëren voor de speler en welke schermen er komen (denk hierbij aan ranking, profiel, home). Dit is wel belangrijk om te weten zodat ik met meer zekerheid kan zeggen dat de opdrachtgever en ik op een lijn zitten. Daarnaast zorgt een navigatie mock-up ervoor dat de opdrachtgever iets ‘tastbaar’ heeft zodat hij eventueel beter kan bedenken wat hij wil zien. Ofwel nieuwe ideeën opdoen.

Verder moet ik kijken naar een potentiele punten systeem en profiel /homepagina. Daarnaast moet ik refactoren waar nodig is en indien ik tijd over houd unit testen.

Taak	Omschrijving
Singleplayer Battle	Een quiz voor één speler
Refactoren	
Profiel/home pagina	
Potentiele punten systeem	Wat krijgt de speler na het afronden van de battle?
Testen	Unit testen
Navigatie mock-up	Bedenken op wat voor manier de speler navigeert door de app.

Tabel 7: samenvatting taken iteratie 2

5.2.2 Refactoren

Ik heb besloten om als eerste te refactoren voor iteratie twee, zodat de nieuwe functionaliteiten gelijk de juiste code base hebben.

In iteratie één had ik ervoor gekozen om een de standaard http client van Android in combinatie met Jackson mapper te gebruiken voor de communicatie met de rest API (backend). De reden dat ik in iteratie één gekozen had om de standaard http client te gebruiken was omdat ik mijn basiskennis wou verbeteren zodat ik in de toekomst beter kan debuggen. De meeste open source libraries zijn namelijk gebouwd op dezelfde

principe. En in de tweede instantie had ik te weinig tijd om me te verdiepen welke library het beste past bij dit project.

De Http client van Android is depracted oftewel Google ondersteund het niet meer. De kans is groot dat de http client in de nieuwe versies van het SDK verwijderd wordt. Om de app zo goed mogelijk future proof te maken moet deze vervangen worden door een alternatief. Ik heb niet onderzocht waarom het depracted is, omdat ik dat niet relevant vind voor het project.

Het alternatief dat Google aanbiedt voor zijn http client is openConnection(). OpenConnection heeft weinig out of the box functionaliteiten maar is daardoor wel licht. Daarnaast is het een nieuw onderdeel van het SDK, toegevoegd in API 22. Hoe goed het werkt is nog niet bekend. Een algemene verbeterpunt is dat de API duidelijker is.

De standaardoplossing is lichtgewicht en kan ook redelijk veel, maar dat moet je zelf bouwen en configureren. Een alternatief hiervoor is om een ‘populaire’ open source libraries te gebruiken. De reden dat ik populaire heb toegevoegd als eis is omdat deze libraries bewezen zijn te werken door meerdere personen in verschillende projecten. Er staat dus een community achter de library die ik kan raadplegen indien ik vragen heb. De libraries waar ik mij in verdiep zijn Retrofit en Volley. Beide libraries zijn in mijn ogen beter dan de standaard optie die Google aanbiedt, omdat ze simpelweg veel meer kunnen met minder code. Dit zorgt ervoor dat de app veel beter onderhoud kan worden. Daarnaast is het sneller en duidelijker dan de standaard optie.

In tabel 8 staan de voor- en nadelen opgenomen van beide libraries specifiek voor dit project.

Volley	Retrofit
Voordelen	
Caching	Veel documentatie
Ingebouwde imageloader	Eenvoudiger om te implementeren in het huidige project.
Memory en error handling	Standaard automatische parsing van JSON naar model.
Krachtig, als in meer functionaliteiten	Performance, zie figuur 12.
	Goede ondersteuning.
	Betere integratie met andere libraries.
Nadelen	
Groot, volley is groot in size	Geen error en memory handling.
Weinig documentatie	Geen image loading.
Slecht ondersteund	

Tabel 8: voor- en nadelen Volley versus Retrofit

	One Discussion	Dashboard (7 requests)	25 Discussions
AsyncTask	941 ms	4,539 ms	13,957 ms
Volley	560 ms	2,202 ms	4,275 ms
Retrofit	312 ms	889 ms	1,059 ms

Figuur 11: performance, Async (Http Client) versus Volley Versus Retrofit [12]

Volley is gemaakt door Google en Retrofit is gemaakt door Square. Google ondersteunt Volley wel, maar doet dat niet heel goed. Zo staat Volley niet in de Maven repository en moet het dus handmatig toegevoegd worden aan het project. Daarentegen is Retrofit wel opgenomen in de Maven repository, wat versie beheer erg versimpeld. Het grootste voordeel dat ik in Volley zie, is dat het standaard al plaatjes kan downloaden met caching. Voor Retrofit is het grootste voordeel dat het JSON parsing automatisch afhandelt. Beide libraries

passen bij dit project. De voor en nadelen hadden mij nog niet overtuigt. Ik heb uiteindelijk een app geschreven waarin ik beide libraries test. Hierbij heb ik gekeken naar de implementatie, dus met welke library hoeft er minder code geschreven te worden en is beter te integreren met dit project?

De conclusie die ik trok, was dat Retrofit beter past bij dit project dan Volley, dit komt voornamelijk door de out of the box JSON naar model mapping. Met als gevolg dat de testbaarheid en onderhoudbaarheid van de code hoger is met Retrofit, omdat er minder complexe code geschreven moet worden voor dezelfde functionaliteiten. Daarnaast stuurt de app constant kleine API calls naar de back-end. Retrofit werkt beter met veel kleine API calls terwijl Volley grote API calls beter afhandelt. Voor image loading heb ik naast Retrofit gekozen om Picasso te implementeren. Dit is een library specifiek voor het downloaden van plaatjes. Picasso is net als Retrofit gebouwd door Square en heeft veel documentatie met goede ondersteuning. De combinatie van Retrofit en Picasso is sterker dan Volley.

Implementatie

De implementatie verliep soepel met dank aan de duidelijke documentatie. Ten eerste heb ik een RestAdapter geïnitialiseerd zoals te zien in figuur 12. De restadapter is memory intensief en hierbij wordt aangeraden om het singleton pattern toe te passen, zodat er een instantie gebruikt wordt door de gehele applicatie. De restAdapter genereert een implementatie van een interface zoals die in figuur 13.

```
private static final String BASE_URL = "http://localhost:8080/api";
private static RestAdapter restAdapter = null;

private RestAdapterProvider() {
}

public synchronized static RestAdapter getRestAdapter() {
    if(null == restAdapter){
        restAdapter = new RestAdapter.Builder()
                .setLogLevel(RestAdapter.LogLevel.FULL)
                .setEndpoint(BASE_URL)
                .setConverter(createJacksonConverter())
                .build();
    }
    return restAdapter;
}
```

Figuur 12: RestAdapter initialisatie

In de interface worden de verschillende API calls gedefinieerd met behulp van annotaties. Als getAllPractices wordt uitgevoerd, haalt het de JSON data op uit:

“http://localhost:8080/api/practices/”.

Door een Callback mee te geven weet Android dat het asynchroon moet worden uitgevoerd.

```
public interface PracticeService {
    @GET("/practices/")
    void getAllPractices(Callback<List<Practice>> callback);
}
```

Figuur 13: interface voor practice

Vrijwel elke model heeft een eigen interface, veel tutorials en code voorbeelden stoppen alle API calls in een interface. Ik heb gekozen om de interfaces te scheiden gebaseerd op de backend. Kortom als een model een eigen controller heeft in de backend, dan krijgt het ook een eigen interface in de front-end, zodat het overzicht voor toekomstige developers beter wordt bewaard.

Net als de RestAdapter moeten ook de interfaces Singleton zijn. Om de implementatie van de interfaces te genereren wordt gebruik gemaakt van een wrapper klasse. PracticeApi is de wrapper klasse voor PracticeService zoals te zien in figuur 14. Tijdens de initialisatie van PracticeApi wordt de implementatie van PracticeService gegenereerd.

```
private PracticeService practiceService;

public PracticeApi() {
    RestAdapter restAdapter = RestAdapterProvider.getRestAdapter();
    practiceService = restAdapter.create(PracticeService.class);
}
public void getAllPractices(Callback<List<Practice>> callback) {
    practiceService.getAllPractices(callback);
}
```

Figuur 14: een wrapper voor de service om de interface te initialiseren

De PracticeApi wordt geïnitialiseerd met behulp van een enum factory zoals te zien in figuur 15. De enum factory creëert de services pas bij de eerste API call. De code is gescheiden en maakt toekomstige uitbreidingen simpeler wat niet goed mogelijk was met de standaardoplossing.

```
PRACTISE_SERVICE(new PracticeApi()),
TALENT_SERVICE(new TalentApi()),
QUESTION_SERVICE(new QuestionApi());

private RestApi INSTANCE;

ApiTypesEnum(RestApi INSTANCE) { this.INSTANCE = INSTANCE; }

public RestApi getApiService() { return INSTANCE; }
```

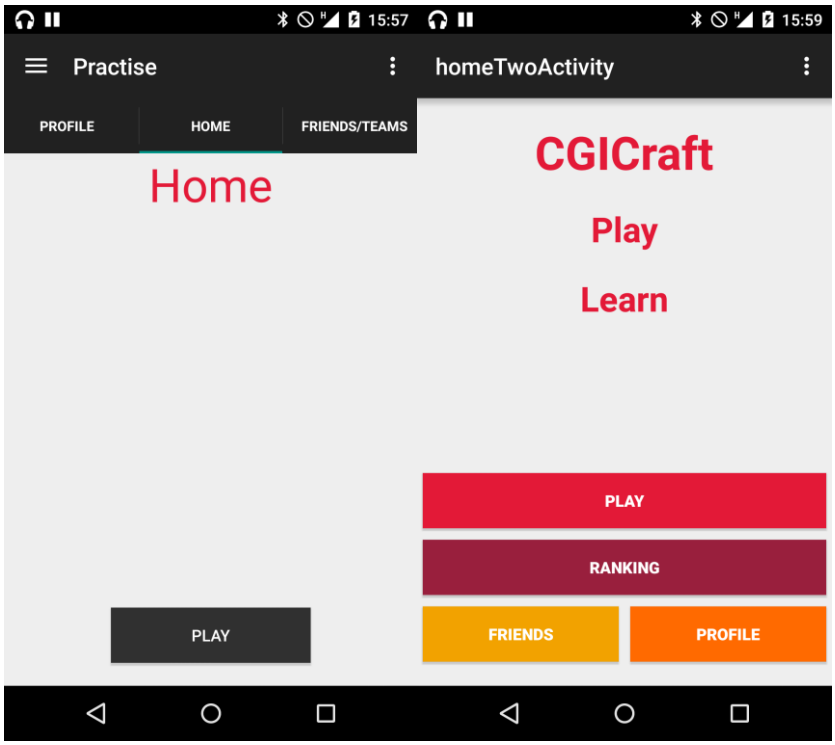
Figuur 15: enum factory methode

5.2.3 Navigatie mock-up

Zoals kort toegelicht in hoofdstuk 5.2.1 is op het moment de mock-up uit figuur 6 het uitgangspunt voor de app. Zowel de opdrachtgever en ik zijn er mee eens dat het design van de app niet past bij onze doelgroep. Het probleem is dat we niet weten hoe de applicatie er dan wel uit komt te zien. Op het moment worden er functionaliteiten toegevoegd zonder rekening te houden met het uiterlijk of plaatsing. Dit kan een valkuil zijn. Bijvoorbeeld als de navigatie inconsistent is, of bepaalde data op de verkeerde locatie staat. Dit leidt allemaal tot verwarring bij de gebruiker. Omdat het gaat om een app is consistentie en gebruikerservaring veel belangrijker dan bij een traditionele CRUD applicatie op een desktop. Dat gezegd hebbende is het doel van de navigatie mock-up, puur bedoeld voor de navigatie. Dus waar worden de schermen geplaatst, welke schermen worden er toegevoegd en welke data wordt hier getoond? Er wordt geen focus gelegd op het design buiten de standaard designelementen die het Android SDK aanbiedt.

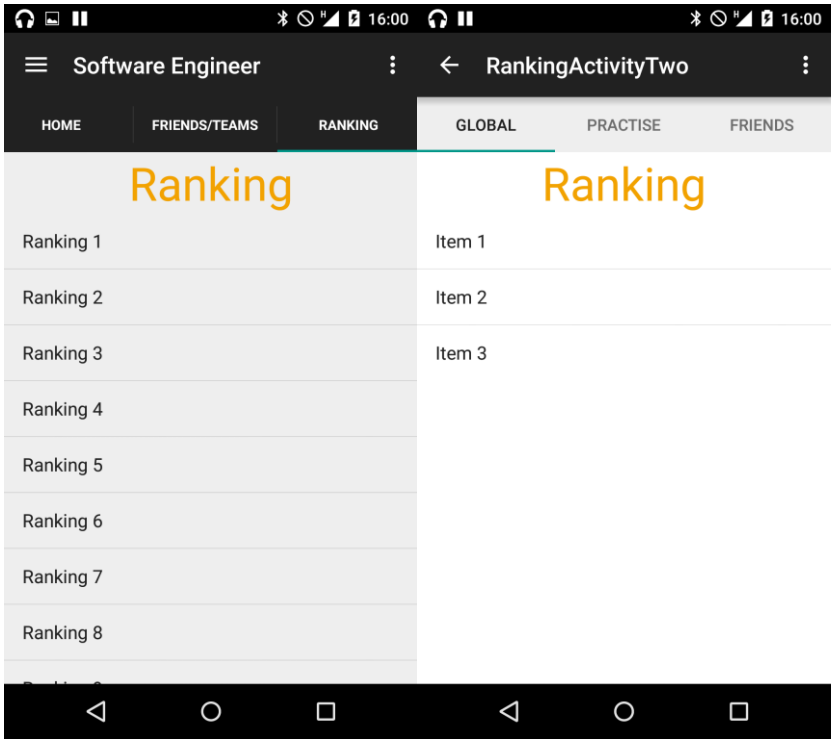
Voor het bepalen van de navigatie heb ik inspiratie opgedaan uit apps uit de Playstore en heb ik de design guide Lines van Android gevolgd. Hieruit heb ik de volgende twee manieren van navigeren bedacht:

- Optie 1: alles op een pagina met weinig detail.
- Optie2: alles op aparte pagina's met veel detail.



Figuur 16: links optie 1 elke pagina direct bereikbaar en rechts optie 2 elke onderdeel heeft een eigen pagina('s)

Zoals in figuur 16 te zien is, kan de gebruiker bij optie één heel snel veranderen tussen pagina's maar kan er op de pagina zelf minder data kwijt. Bij optie twee heb je aparte pagina's voor elke onderdeel. Die kunnen weer onderverdeeld zijn in meerdere pagina's waardoor je bij optie twee op een nette manier meer informatie kwijt kan. Optie één daarentegen is meer gebruiksvriendelijk.



Figuur 17: optie twee kan meer detail kwijt dan optie één

De keuze die gemaakt wordt zal geen invloed hebben op de functionaliteiten maar zorgt ervoor dat er een beter beeld ontstaat van het uiteindelijke product. De schermen bestaan en ik moet ze alleen nog vullen met data en functionaliteiten.

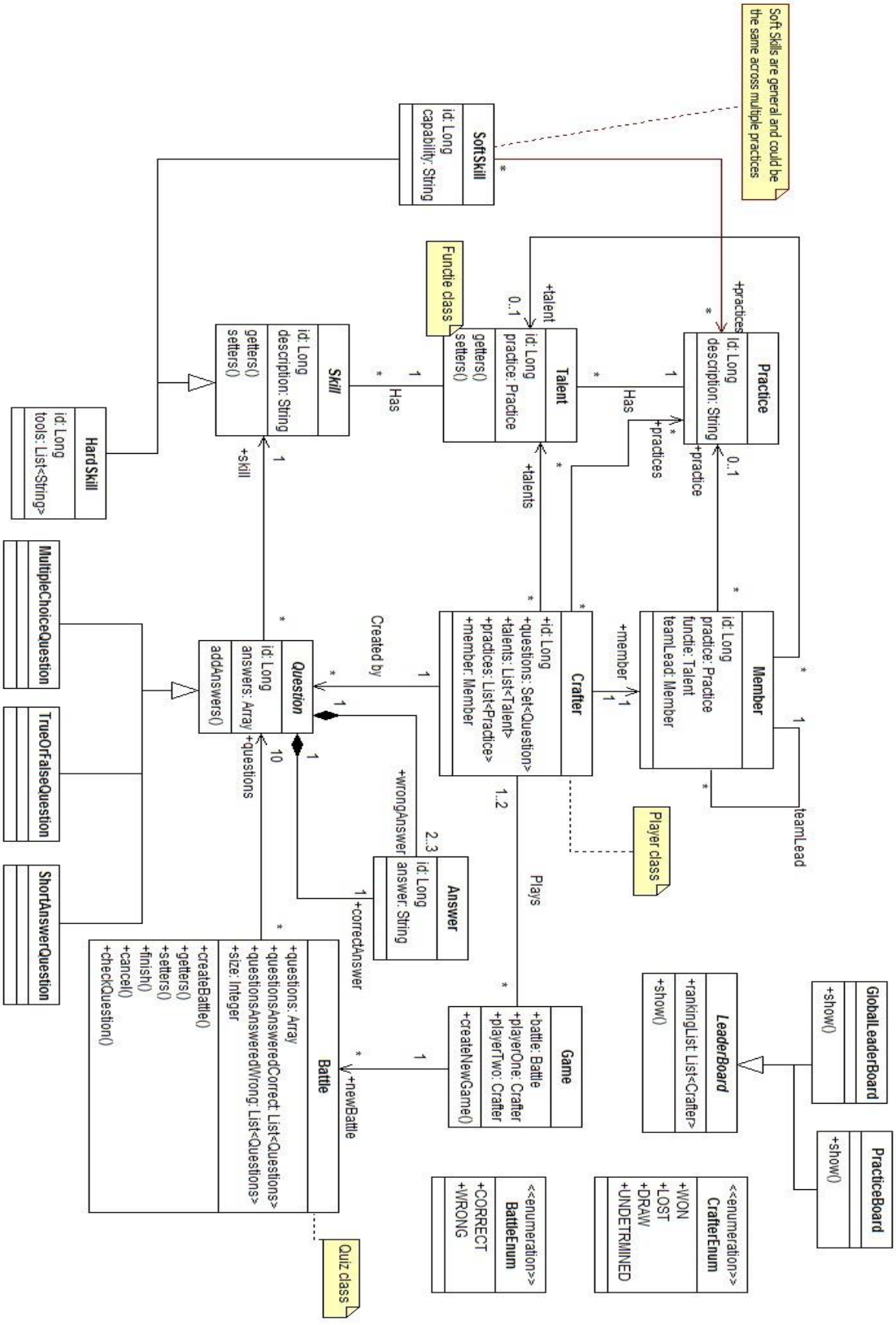
5.2.4 Battle bouwen

In figuur 18 is het klassendiagram te zien dat gemaakt is in iteratie twee. Binnen het klassendiagram zijn niet alle klassen opgenomen denk hierbij aan netwerk en parsing klassen. Dit is gedaan om het overzicht te bewaren. Daarnaast heb ik ook niet alle variabelen opgenomen omdat het diagram dan te groot zou worden voor dit document.

Er zijn een aantal veranderingen ten opzichte van het klassendiagram van iteratie één. Zo heb ik functie hernoemd naar talent zodat het consistent is met de rest van de gebruikte termen. Functie is een Nederlands woord en binnen de app worden alleen Engelse termen gebruikt (dit was een eis van de opdrachtgever) de vertaling van functie zou zijn Job. Job past niet goed bij het spel. Met gamificatie wil je spelers prikkelen, dit kan bijvoorbeeld door aparte namen te gebruiken die je niet dagelijks tegenkomt. Job is een term dat sterk gerelateerd wordt met werken. Dat is niet de ervaring die we willen creëren. Vandaar dat functie vertaald is naar talent. Het woord talent komt voort uit de originele mock-up.

Naast talent is er ook een nieuwe klasse Crafter bijgekomen. Tijdens de inception fase werden er verschillende namen gegeven aan CGICraft. Waarvan gegamificeerde PDP het meest gebruikte was. De verschillende namen zorgde voor verwarring, vandaar dat ik ervoor koos om de mock-up naam CGICraft over te nemen en dit te gebruiken tijdens de ontwikkeling. CGICraft komt voort uit World of Warcraft waar het originele idee uit is opgebouwd. De term Crafter slaat op de spelers van het spel. Met Crafter wordt bedoeld: een bouwer en in dit geval is dat een speler die aan zijn carrière bouwt. Crafter mag dan ook meer practices en talents hebben dan een member. De crafter wordt gemotiveerd om meerdere practices en talents uit te proberen en beoefenen (kennisontwikkeling).

Verder zijn er een aantal relaties veranderd. Ten eerste, soft skills zijn algemeen en zijn dus niet specifiek gebonden aan een talent zoals dat wel het geval is voor hard skills. Een soft skill komt terug in meerdere practices. Ten tweede is de relatie tussen question en answer verbeterd. Het houdt nu zowel zijn foute als goede vragen bij, terwijl daar in iteratie één geen duidelijk onderscheid tussen zat. Daarnaast wordt er nu onderscheid gemaakt tussen de verschillende soorten vragen. Verder zijn er game, battle en enum klassen toegevoegd. De enum klassen houden de staat van het spel en de speler bij. Als laatste is het leaderboard alvast opgenomen maar heeft deze nog geen relaties. Tot slot komt er een punten systeem, maar dat valt voorlopig buiten deze iteratie.



Figuur 18: klassendiagram iteratie 2

Implementatie

Als eerste heb ik de database gebouwd omdat het toevoegen van spring componenten geen nut heeft zonder een database. Spring (Hibernate) geeft compile time errors als de entities niet gemapped kunnen worden met een tabel.

De entities in Spring, in dit geval alle model klassen in het klassendiagram zijn een op een met de database tabellen. Met deze gedachte heb ik het klassendiagram gebruikt voor het ontwerpen van de database.

Een factor waar ik wel rekening mee moet houden zijn overervingen. In het klassendiagram is te zien dat er twee klassen zijn die gebruik maken van overerving binnen CGICraft. De eerste is skills en de tweede is questions (Leaderboard is niet relevant voor deze iteratie).

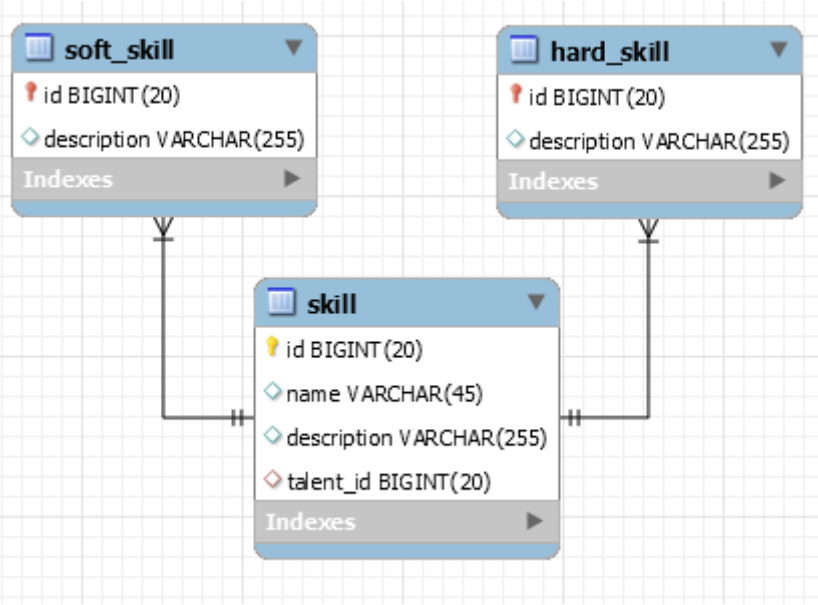
Overervingen in een applicatie worden anders afgehandeld dan in een database. In de database kan overerving op meerdere manieren worden geïmplementeerd. Ik heb hier drie verschillende overervings technieken voor gevonden:

- Table-Per-Type (TPT)
- Table-Per-Hierarchy (TPH)
- Table-Per-Concrete (TPC)

Ik zal kort toelichten wat de verschillende overervings technieken inhouden en welke ik heb toegepast voor skill en question.

TABLE-PER-TYPE (TPT)

Table per Type heb ik toegepast voor skill. TPT houd in dat zowel de parent als child een eigen tabel hebben. Alle generieke attributen staan in het parent tabel ‘skill’ en alle specifieke attributen van soft- en hard skills staan in de tabellen ‘soft- en hard skill’. De child tabellen hebben een primaire sleutel dat tegelijk een vreemde sleutel is naar zijn parent, zoals te zien in figuur 19.



Figuur 19: TPT toegepast voor skill

TPT is goed te onderhouden en houd dezelfde structuur als de applicatie aan. Tevens is het de meest genormaliseerd techniek waardoor de leesbaarheid hoog is. Dit maakt de implementatie van TPT makkelijker ten opzichte van de rest. Het nadeel is echter dat er joins en unions geschreven moeten worden als ik bepaalde gegevens wil ophalen. Dit gaat ten koste van de performance.

TPT past goed bij skills omdat skills niet gestandaardiseerd zijn. Er zijn in totaal 10 practices, elke practice kent meerdere talents, elke talent heeft verschillende skills. Deze zijn allemaal op andere wijze opgenomen binnen CGI. Er is dus geen gestandaardiseerd formaat voor skills, echter zijn wel alle skills onderverdeeld in soft- en hard skills. Doordat het formaat niet gestandaardiseerd is, weet ik niet hoeveel verschillende soorten skills er zullen ontstaan en welke attributen deze bevatten. Met TPT vang ik dit probleem beter op.

TABLE-PER-HIERARCHY (TPH)

Table per Hierarchy heb ik geïmplementeerd voor question. TPH kent maar één tabel waar alle attributen worden opgenomen. Het onderscheid tussen de verschillende soorten questions wordt gemaakt met behulp van een discriminator kolom.

Het voordeel van TPH is dat er geen joins geschreven hoeven te worden en dus daardoor sneller is dan de rest. Het nadeel is dat alles in een tabel staat, dit leidt ertoe dat veel kolommen een null waarde zullen bevatten. Om de integriteit te bewaren, moeten er checks en constraints geschreven worden. Echter is dit geen probleem voor question. De child klassen van question hebben op het moment geen eigen attributen en zo zijn er dus ook geen null waardes mogelijk. Als ik in dit geval ervoor kies om TPT toe te passen dan krijg ik twee lege tabellen met alleen een primaire/vreemde sleutel en met TPC zou ik drie exact dezelfde tabellen krijgen.

TABLE-PER-CONCRETE (TPC)

TPC heb ik niet toegepast. Bij het laatste type ‘TPC’ worden alle attributen in zowel de parent als de child tabellen opgenomen. Er zijn ook geen relaties tussen de child en parent tabellen. Het voordeel is dat alle data direct bij de corresponderende tabel beschikbaar is. Het nadeel is echter dat de onderhoudbaarheid van de database erg omlaag gaat. Er moet constant gecheckt worden of er iets veranderd in de parent tabel dat invloed zou hebben op de child tabel en vice versa. Als dit niet goed wordt uitgevoerd komt de integriteit van de data in gevaar. TPC is hierdoor het meest lastig om te implementeren.

Uitbreiden backend

Bij de uitbreiding van de backend heb ik rekening gehouden met de implementatie van de overervingen en veel op veel relatie tussen de crafter naar talent en crafter naar practice. Dus hoe wordt de veel op veel relatie gemanaged. Dit is goed op te lossen in spring met behulp van JsonBackReference en JsonManagedReference. Dit geeft aan wie de eigenaar is bij een veel op veel relatie binnen de applicatie.

Naast de mapping heb ik ook onderzocht wat de beste manier is om plaatjes op te slaan. Tijdens een quiz worden er plaatjes gebruikt ter ondersteuning van de vraag. Dit is niet altijd het geval maar is wel mogelijk. Zo kan er een plaatje van een stukje code getoond worden en daarbij een vraag bedacht worden.

Het probleem is echter hoe worden de plaatjes opgeslagen? Hierbij zijn twee oplossingen te bedenken:

- Oplossing 1: Opslaan van plaatjes in de database.
- Oplossing 2: Opslaan van plaatjes op een server met de locatie naar de plaatjes in de database.

Beide opties hebben zijn voor en nadelen. Oplossing één is sneller maar zorgt ervoor dat database snel groeit in opslaggrootte. Database opslag is duurder dan systeem opslag. Oplossing twee is beter schaalbaar maar langzamer. Oplossing één is beter als de plaatjes vaak veranderen en er een geschiedenis van moet worden bijgehouden of als de meta data om de plaatjes heen erg belangrijk is. Het bewaren van de integriteit is namelijk eenvoudiger als de plaatjes direct in de database worden opgeslagen.

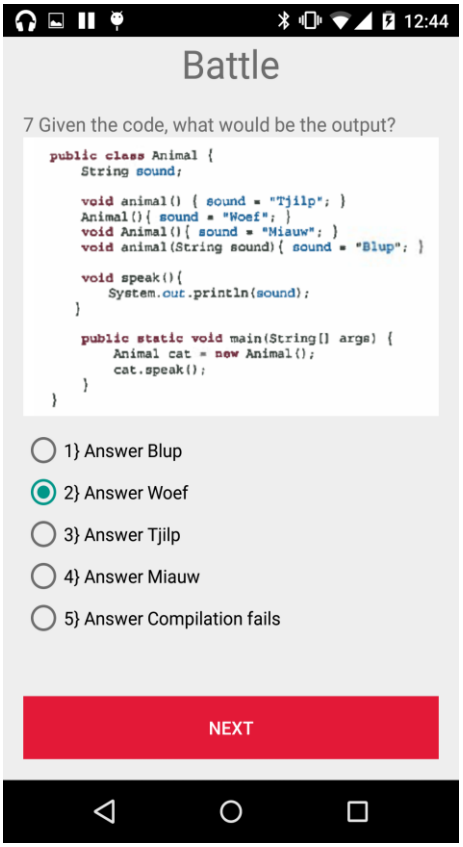
Geen van de bovenstaande voordelen zijn van toepassing voor plaatjes in CGICraft. De plaatjes worden hoogstwaarschijnlijk een keer aangemaakt en zullen niet veranderen. Daarnaast is schaalbaarheid erg belangrijk. Uiteindelijk is de bedoeling dat elke member vragen kan toevoegen, dit kan leiden tot een enorme groei in plaatjes. Hierbij komt schaalbaarheid goed van pas en is oplossing twee dus beter.

Uitbreiden mobiele app

Het bouwen en uitbreiden van de app verliep soepel er waren geen grote problemen. Het enige probleem dat ik tegenkwam is dat je snel de neiging krijgt om alles binnen de activities uit te voeren. In het klassendiagram zijn twee klassen toegevoegd dat geen modellen zijn, namelijk: battle en game. Game is voor het bepalen van het type, dus een multiplayer of een singleplayer. Game is gekoppeld aan een battle. De battle haalt random vragen op vanuit de server en stuurt dit naar activity.

Dit is echter niet mogelijk en wordt zelfs afgeraden vanwege de memory leaks die kunnen ontstaan. In mijn geval haalt activity alle vragen op en stuurt deze door naar de battle klassen (, dus de werking is omgedraaid). Vervolgens vraagt de Activity continue voor updates aan battle. Zoals, controleren of de vraag goed is beantwoord. Activities draaien op de main thread, normale Java klassen draaien op een aparte thread. Dit maakt de communicatie tussen de twee lastig, tenzij activity een referentie heeft naar de Java klasse.

In figuur 20 zie je hoe een battle eruitziet. De vraag staat bovenaan met daaronder een plaatje ter ondersteuning gevolgd door de bijbehorende antwoorden. Er is geen focus gelegd op design, alleen de standaardkleuren van CGI zijn overgenomen.



Figuur 20: singleplayer battle scherm

5.2.5 Testen

Ik heb de functionaliteiten die er zijn gebouwd getest doormiddel van error guessing en exploratory testen. Ik heb de user story US2.1 uitgewerkt en deze getest. Tijdens de bouw ben ik voor elke functionaliteit nagegaan of het werkt zoals beschreven.

Het gaat hier om de battles spelen. Om de battles te kunnen spelen heb ik test data toegevoegd. Deze data heb ik verkregen uit de entree toets van CGI. De entree toets is een simpele algemene software engineer (Java) toets voor nieuwe medewerkers. De vragen van de entree toets heb ik toegevoegd in de database met de bijbehorende code snippets. Vervolgens heb ik de quiz meerdere malen gespeeld en zelfs toekomstige gebruikers de quiz laten spelen om te kijken of de vragen en gekozen antwoorden goed worden geregistreerd.

5.2.6 Reflectie

Iteratie 2 ging over het algemeen goed. Ik heb en klein deel gerefactored maar daar was ik al van op de hoogte tijdens iteratie 1. Overigens heb ik wel een aantal opmerkingen over de battle.

Ik heb gemerkt dat het bouwen van de app lastig is omdat veel elementen gekoppeld worden aan de view. De view ofwel de UI heeft dus invloed op hoe de functionaliteiten worden geïmplementeerd. Op het moment benader ik de applicatie door puur functionaliteiten toe te voegen maar dit kan een probleem zijn. Het is geen vereiste voor de medewerkers om dit systeem te gebruiken. De manier waarop een app ervaren kan worden is dan ook belangrijk e.g. UI. Het beste zou zijn om dit over te laten aan iemand die hier expertise over heeft, zoals een designer. Dit had ik gemeld bij de opdrachtgever. De opdrachtgever zei dat er geen designers zijn binnen CGI. Wat ik merkwaardig vond gezien de grootte van CGI. Natuurlijk zijn functionaliteiten het belangrijkste maar hoe verder je komt in het project hoe belangrijker het wordt om een goed design te hebben. Tevens wordt het lastiger om de UI aan te passen des te verder je in het ontwikkeltraject zit.

Verder moet ik een nieuwe manier vinden om data toe te voegen. Het originele idee om gebruikers vragen toe te laten voegen via de app is te lastig. Het schrijven van vragen op je mobiel werkt niet fijn. Hierbij is bedacht om een aparte webapplicatie ter ondersteuning te bouwen die puur bedoeld is voor de teamleader en het toevoegen van content.

5.3 Iteratie 3

5.3.1 Iteratie planning sessie

Iteratie twee eindigde met een meeting waarin we een aantal zaken hebben besproken.

Terugkoppeling iteratie twee

Als eerste hebben we besproken hoe iteratie twee is verlopen. Hierbij hebben we gekeken in hoeverre ik alle besproken taken heb uitgevoerd. Daarnaast heb ik mijn problemen/bevindingen besproken. Gevolgd door een demo van het product en de nieuwe functionaliteiten. Over het algemeen ging de meeting goed. Ik had voldaan aan alle taken en loop nog goed volgens de planning. Daarnaast ben ik geen grote problemen tegengekomen.

Er zijn twee onderdelen die ik mis in het project. Dat zijn een ontwerper en een expert op het gebied van Java. De designer zou me kunnen helpen met het indelen van de app en de programmeur zou kunnen kijken naar mijn code. Dit heb ik laten weten aan de opdrachtgever.

Er zijn geen designers binnen CGI dus hier moet ik omheen werken. De expert kan een van mijn collega's zijn. Ik moet hiervoor een afspraak maken.

Als laatste heb ik de navigatie demo apk (install file) gemaild naar de opdrachtgever zodat hij dit kan testen en een keuze kan maken.

Taken iteratie drie

Voor iteratie drie hebben we bedacht om te testen. Zowel de front- als de back-end zijn nog niet getest. Tijdens iteratie 1 en 2 heb ik geen tijd gehad om officieel te testen. Ik heb alleen gebruik gemaakt van informele testen zoals error guessing. Om te voorkomen dat hier een groot achterstand ontstaat heb ik ervoor gekozen om deze iteratie alles te testen. De opdrachtgever vond dit prima maar gaf wel de suggestie dat we na deze iteratie zo snel mogelijk moeten werken naar een eerste versie. Deze kunnen we vervolgens testen bij een kleine groep users.

Naast testen wilt de opdrachtgever ook dat CGICraft aan Sonar wordt toegevoegd. Hiervoor moet eerst gedeployed worden naar een build server. Ook moeten de profiel en homepagina verder worden uitgewerkt. De profiel pagina moet aanpasbaar zijn en de battle moet gebaseerd zijn op de data in de profiel pagina. Zo krijgt de gebruiker alleen vragen over zijn practices, talents, etc.

Als laatste heb ik de opdrachtgever laten weten dat ik achterloop qua documentatie en dat ik eventueel tijd uittrek om dit bij te werken. Halverwege de iteratie is namelijk de 60% oplevering van mijn verslag.

In tabel 9 is een korte opsomming van de taken te zien:

Taak	Omschrijving
Singleplayer Battle personaliseren	De quiz moet bouwen aan de hand van de practices en talents die de speler heeft gekozen.
Refactoren	
Profiel/home pagina	Een gebruiker kan zijn profiel pagina aanpassen. Eg. Een nieuwe practice kiezen.
Testen	Unit testen front- en back-end volgens DC Java richtlijnen.
Deployen	Deployen naar Jenkins build server en toevoegen aan Sonar.
Documentatie bijwerken	Werken aan afstudeerverslag en de documenten die hierin komen, zoals UML diagrammen.

Tabel 9: samenvatting taken iteratie 3

5.3.2 Testen

Voordat ik daadwerkelijk begin met testen had ik de guide Lines die zijn vastgesteld door CGI opgezocht.

5.3.2.1 DCJava guidelines

De relevante onderdelen uit het DC Java guideline met betrekking tot testen (van CGICraft) kunnen als volgt worden opgesomd:

- Code duplicatie moet minder zijn dan 1%.
- De code moet voldoen aan de volgende randvoorwaarden zoals berekend in Sonar:
    - Blocker violations: 0.
    - Critical violations: ≤ 0.5 per 1000 lines of code.
    - Major violations: ≤ 5 per 1000 lines of code.
    - Minor violations: ≤ 10 per 1000 lines of code.
    - Info violations: ≤ 20 per 1000 lines of code.
- Meer dan 95% van de publieke API moet relevante JavaDoc bevatten. Het gaat hierbij alleen om belangrijke methodes dus geen getters, setters, etc.
- De bovenstaande punten worden ofwel opgesomd in de Technical Dept Ratio(TDR). Deze mag nooit hoger zijn 5%. Hieronder een overzicht van de nummers:
    - TDR < 2,5%: Good.
    - 2,5% ≤ TDR < 5%: Sufficient.
    - 5% ≤ TDR < 10%: Insufficient.
    - 10% ≤ TDR < 15%: Bad.
    - 15% ≤ TDR: Very bad, seriously consider throw away and rebuild from scratch.
- De Test coverage voor unit testen moet zijn LineCoverage ≥ 80% en TestSuccess = 100%. Verder moet elke test methode op zijn minst een assertion bevatten.

Tijdens het testen heb ik gestreefd om aan de guideline te voldoen. Ik kom met de meeste regels pas in aanmerking nadat ik het project in Sonar plaats. Dit is echter niet ideaal. De deployment wordt uitgevoerd samen met collega/admin die al ervaring heeft met Jenkins en Sonar maar dit is pas tegen het einde van de iteratie. Tot die tijd heb ik gebruik gemaakt van de ingebouwde tools in Intellij om mijn test coverage te meten. Uit ervaring weet ik dat dit redelijk overeenkomt met Sonar. Zo heb ik tijdens het testen een soort van overzicht totdat het project in Sonar staat.

5.3.2.2 Back-end unit testen

Testen kan opgedeeld worden in twee onderdelen. Het testen van de front-end en het testen van de back-end. Ik ben begonnen met het unit testen van de back-end omdat dit belangrijker is. Hoewel het grootste gedeelte van de business logica verwerkt is in de front-end, kan ik er niet omheen dat de back-end open staat voor het publiek (REST API). Het is dan ook belangrijk dat de back-end goed getest is omdat iedereen hierbij kan. Als de applicatie uitgebreid wordt dan is de back-end de basis van de applicatie.

Tools

Om te kunnen Unit testen heb je een aantal tools/libraries nodig. De basis voor unit testen is Junit. Om de individuele stukjes code (unit) te testen is het belangrijk om de invloed van buiten te verkleinen. Dit wordt meestal gedaan met behulp van stubs, mocks, etc.

Hiervoor zijn libraries beschikbaar. De meest populaire library is Mockito daarnaast heb je alternatieven zoals Easymock, Powermock en Jmockit. Ieder framework heeft een ander structuur. Dit maakt de overgang tussen frameworks lastig. Voor het kiezen van de frameworks heb ik me verdiept in elke framework.

Uiteindelijk heb ik mijn testgroep verkleind tot Jmockit en Mockito.

Beide frameworks verschillen enorm. De syntax voor beide frameworks is anders. Dat gezegd hebbende past de syntax van Mockito zich aan de test case terwijl de syntax van Jmockit consistent blijft.

Hoewel Jmockit erg populair is gaat de voorkeur van de meeste developers uit naar Jmockit van wat ik heb gelezen. Beide libraries hebben ook verschillende filosofieën dit zorgt ervoor dat beide frameworks aparte functionaliteiten bevatten.

Uiteindelijk wordt de keuze gebaseerd op de specifieke project, de eisen die eraan zijn gesteld en structuur van de code. Voor CGICraft heb ik gekozen voor Jmockit. Welke framework ik kies heeft weinig invloed voor het project. Daarnaast is het geen groot project en zijn er geen specifieke eisen vanuit de organisatie. Het komt dus uit op persoonlijk voorkeur. Ik heb al eens eerder gewerkt met Jmockit en was dus hiermee bekend.

Testen

Ik heb als eerste alle models getest, gevolgd door de services en controllers. De model test ziet er als volgt uit:

```
@Test
public void createPractice(){
    Practice entity = new Practice();
    entity.setDescription("desciription test");
    entity.setName("unit test");
    Assert.assertEquals("constructing Practice","desciription test", entity.getDescription());
    Assert.assertEquals("constructing Practice","unit test", entity.getName());
}
```

Figuur 21: unit test model practice

In figuur 21 is het model test van practice te zien. Dit is een hele simpele unit test aangezien de models voornamelijk variabele met getters en setters methodes bevatten. In figuur 21 worden de constructor, getter en setter getest.

```
@Test
public void getAllPracticesTest() throws Exception {
    final List<Practice> resultList = new ArrayList<>();
    final Practice p = new Practice();
    p.setName("test");
    p.setDescription("desc");
    resultList.add(p);
    new NonStrictExpectations(){{
        practiceService.getAllPractices(); result = resultList;
    }};

    mockMvc.perform(get("/practices/")).andExpect(status().isOk())
            .andExpect(jsonPath("$..name").value("test"))
            .andExpect(jsonPath("$..description").value("desc"));
    new Verifications(){{
        practiceService.getAllPractices();
    }};
}
```

Figuur 22: controller test practice

In figuur 22 is de controller test voor practice te zien. Ten eerste wordt er mock request gebouwd en uitgevoerd met behulp van mock MVC, hierbij wordt aangegeven wat ik als resultaat verwacht. In dit geval is dat een naam en een omschrijving. De controllermethode roept vervolgens de service klasse aan. De Service klasse is een mock waarbij het resultaat van de methode een resultList is. In verifications kijk ik of de juiste methode ook daadwerkelijk wordt aangeroepen. De assertion is in dit geval de andExpect.

```
@Test
public void findPracticeTest(){
    final Practice resultPractice = new Practice();
    new Expectations(){{
        practiceRepository.findOne(1L); result = resultPractice;
    }};
    Assert.assertEquals("find practice", practiceService.findPractise(1L), resultPractice);
}
```

Figuur 23: Service test Practice

Doordat de back-end weinig logica bevat zijn ook de services makkelijk te testen. Net als bij de controller check of de Service klasse de juiste methodes aanroept en of de resultaat van de methodes overeenkomt met wat ik had verwacht (assert).

De repository klassen kon ik niet testen omdat het allemaal interfaces zijn die gebruik maken van namedqueries JPA. Dit houdt in dat de query wordt gebouwd aan de hand van de naam van de methode.

findByName(String name)

Zou resulteren in een query gelijk aan

SELECT * FROM Practice WHERE name = ‘’;

(* vervangen door alle kolomnamen).

Deze worden compile time gebouwd. Als er een fout is in de mapping of als de naam niet klopt dan wordt dat tijdens het compilen opgevangen. Wel heb ik alvast een in memory database geïmplementeerd voor het geval dat ik custom query’s moet schrijven. De in-memory database is nog niet gevuld. Hiervoor ga ik gebruik maken van een populator klasse. Deze vult de database tijdens de initialisatie met test data.

Front-end unit testen

Unit testen van Android is lastiger dan de back-end. De Android app bevat zowel Java klassen zonder enige spoor van Android en klassen specifiek van Android. Er wordt dus onderscheid gemaakt tussen Android specifieke klassen zoals activities en fragments en simpele Java klassen.

De manier waarop de front-end is gebouwd zorgt ervoor dat de meeste logica in Java klassen staat, met een deel in activities. (Dit kan niet anders en komt door de structuur van Android). De Java klassen kunnen getest worden met Junit. De Android klassen worden getest door specifieke interfaces te implementeren die aangeboden worden door het Android SDK.

Testen in Android is lastig omdat het nog een vrij jong onderdeel is, dat recent is uitgebreid. De IDE die ik gebruik bied dan ook geen goede ondersteuning om Junit testen uit te voeren. Dit was veel gedoe om het werkend te krijgen. Om unit testen uit te voeren moet de bouw variant veranderd worden. Hierdoor kunnen unit testen en Android testen niet tegelijkertijd worden uitgevoerd. Wat testen in Android erg omslachtig maakt.

```
@Test
public void getNextQuestionTest(){
    battle = new Battle(new BattleOptions(2, 2));
    battle.setQuestions(questionList);
    Assert.assertEquals(battle.currentBattleState(), BattleStateEnum.STARTED);
    Assert.assertEquals("", battle.getNextQuestion());
    battle.setState(BattleStateEnum.PLAYING);
    Assert.assertEquals(battle.getNextQuestion(), questionList.get(1).getQuestion());
}
```

Figuur 24: test voor volgende vraag methode

In de figuur 24 is te zien hoe de methode getNextQuestion wordt getest. Deze methode heeft als taak de volgende vraag te returnen maar dat doet de methode alleen als de staat van het spel playing is. Zodra ik de staat verander haal ik de tweede vraag op uit een lijst die ik al eerder meegaf. De eerste vraag wordt overgeslagen. Maar dat doet hier niet toe.

Ik heb geen Android specifieke testen geschreven omdat deze weinig toegevoegde waarde hebben. Een voorbeeld van zo'n test is bijvoorbeeld, checken of de tekst in en tekstveld is veranderd.

5.3.3 Deployment CGICraft back-end

DC Java Guideline

Voor de deployment van de back-end heb ik een guideline gevolgd van DC Java. Binnen CGI draait alles op een proxy waardoor elke laptop geconfigureerd moet worden als men gebruik wil maken van de interne tooling. Voordat de back-end toegevoegd kan worden aan Jenkins moet het ten eerste opgenomen worden in Nexus, een interne repository.

De back-end staat al in Subversion maar nog niet in Nexus. Om de back-end te deployen in Nexus moeten de Maven settings worden aangepast. Ik heb hierbij de Maven settings gevraagd aan mijn collega's.

Nadat alles opgezet was heb ik samen met een collega een build script aan Jenkins toegevoegd. De build lukte. Dus de testen slaagde maar de build werd niet goed gepakt door Sonar. Ik heb hier hulp gevraagd aan mijn collega's maar niemand wist er raad mee. Zelf wist ik ook niet waar het aan lag. Ik had het voorlopig gelaten voor wat het is. Ik ga dit in de toekomst debuggen door een build server en Sonar lokaal te draaien.

5.3.4 Reflectie

Hoewel de deployment niet is gelukt heb ik volgens Intellij wel een hoger testcoverage dan 95%. Tevens heb ik de test laten controleren door een collega. Hij heeft snel gekeken of de testen logisch zijn (en doen wat ze moeten doen). Voor de front-end heb ik alleen maar de business logica getest en geen activities, views, etc. De code coverage is ongeveer 40% de precieze code coverage kan ik niet berekenen omdat ik gebruik maak van twee technieken. Junit en Android instrument testing.

De unit testen op zichzelf zeggen niet veel over de kwaliteit van het systeem. De testen kunnen slecht zijn geschreven. Natuurlijk heb ik dit elke iteratie de user stories zo goed mogelijk proberen door te nemen met behulp van exploratory testing en error guessing. Unit testen is immers maar een onderdeel van het testing framework. Verder hoor ik unit testen gewoon bij te houden. Dit heb ik helaas niet gedaan.

Daarnaast wil je elke iteratie nieuwe functionaliteiten toevoegen. Dat is deze iteratie minder gebeurd doordat ik moest testen. Aan de andere kant is het beter vroeg dan laat. Tot iteratie 3 heb ik een aantal functionaliteiten toegevoegd. Door deze goed te testen in plaats van verder uitstellen kan ik met zekerheid zeggen dat ik t/m iteratie 3 voldoe aan de eisen van CGI. Ook kan ik hier nu verder in het project rekening mee houden.

5.4 Iteratie 4

5.4.1 Iteratie planning sessie

Iteratie drie eindigde met een meeting waarin we een aantal zaken hebben besproken.

Terugkoppeling iteratie drie

De meeting ging goed. Ik loop nog steeds op planning en de afgesproken taken heb ik uitgevoerd. Ik heb een minimale test coverage van 95% voor de back-end.

De front-end heeft een lager percentage. Dit had ik gemeld bij de opdrachtgever. Hij vond dit niet erg, als de back-end maar voldoet aan de dc Java guidelines. Wel heb ik voor de front-end een groot gedeelte van de belangrijke functionaliteiten getest, dus geen views maar business logica. Bovendien is de vraag of 80% testen nodig is voor de app. We hebben het gelaten voor wat het is en zijn verder gegaan met de planning.

Vervolgens heb ik aangegeven dat deployment niet is gelukt en waarom het is gefaald. Hij vond het niet erg. Wel hebben we besproken om alsnog te deployen, maar dit heeft niet de hoogste prioriteit. Tijdens deze iteratie zal ik werken aan deployment als ik tijd overhoud tussen grote taken.

Verder kon ik tijdens de meeting weinig nieuws laten zien aangezien ik deze iteratie vrijwel alleen heb getest en het overige deel daarvan aan documentatie is opgegaan. Deze moest bijgewerkt worden voor een 60% gesprek tijdens iteratie 3.

Tot slot heb ik besproken hoe mijn 60% gesprek was verlopen.

Taken iteratie vier

Aan de hand van de planning (en meeting) worden er in iteratie vier veel nieuwe functionaliteiten toegevoegd. Dit is een van de grootste en belangrijkste iteraties. We hebben tijdens de meeting besproken dat er aan het einde van iteratie 4 een potentieel leverbaar product moet zijn.

Deze zou dan uitgebracht worden bij een kleine test groep. De feedback die ik krijg van de test groep kan ik in iteraties vijf en zes verwerken. Om dit te realiseren moet er een server klaargezet worden. De opdrachtgever gaat rondvragen of er een interne server beschikbaar is en anders kijken of er een externe server gehuurd kan worden. Als dat niet mogelijk blijkt te zijn, kan ik de server bij mij thuis op mijn laptop draaien.

Voor deze iteratie moeten de volgende functionaliteiten worden toegevoegd:

- Level systeem, aan het einde van elke battle krijgt de speler een bepaald aantal punten toegekend afhankelijk van hoe hij het spel heeft gespeeld. Na een bepaald aantal punten stijgt de speler in rang (level).
- Comment/feedback systeem, na het beantwoorden van een vraag kan de speler een opmerking (comment) toevoegen aan de vraag. Daarnaast kan de gebruiker bestaande vragen beoordelen en kan hij op bestaande comments reageren (reply).
- Profiel pagina, de profiel pagina die in iteratie 3 is gebouwd laat een aantal basis gegevens zien. Voor deze iteratie krijgt de speler de mogelijkheid om zijn profiel aan te passen. Tevens moet de profiel (gekozen practices, talents, etc.) invloed hebben op de battle. Een speler krijgt alleen vragen over zijn of haar gekozen vakgebied.
- Rewards/unlockables, een speler kan aan het einde van elke battle een random reward verdienen. Eventueel worden er ook unlockables toegevoegd. Deze kunnen vrijgespeeld worden door een bepaald hoeveelheid punten te halen.

Voor extra informatie zie bijlage C.

De opdrachtgever en ik waren er beide mee eens dat de bovengenoemde onderdelen nog niet volledig zijn uitgewerkt. Wat ik hiermee bedoel is dat de eisen niet vast staan. Als voorbeeld pak ik het reward systeem. Hoe wordt bepaald of een speler een reward verdient? Wat voor reward krijgt hij dan? Op dit soort vragen weten we niet de juiste antwoord. We verwachten hier meer inzicht over te krijgen aan de hand van de user feedback. Vandaar dat het erg belangrijk is dat er aan het einde van deze iteratie iets speelbaars is, zodat het spel kan worden afgestemd.

Tot slot zal er ook het een en ander worden gerefactored.

In tabel 10 is een korte opsomming van de taken te zien:

Taak	Omschrijving
Comment systeem	De gebruiker kan opmerkingen plaatsen bij een vraag.
Level systeem	Na een battle krijgt de speler een aantal punten. Door een bepaald aantal punten te halen stijgt de speler in rang.
Rewards	Aan het einde van elke battle heeft de speler een kans om reward te verdienen.
Ranking	De speler kan stijgen in rang en dit is te zien op een leaderboard.
Profiel pagina	De speler moet zijn profiel kunnen aanpassen en de profiel moet invloed hebben op de battles
Refactoren	

Tabel 10: samenvatting taken iteratie 4

5.4.2 Refactoren

Origineel was ik niet begonnen aan refactoren maar al snel kwam ik tijdens de iteratie achter dat wel nodig was.

Huidige probleem

Op het moment worden er voor netwerkcalls veel boilerplate code geschreven. Hierdoor worden de activities groot en onleesbaar. Dit komt niet zo zeer door de manier waarop ik Retrofit heb geïmplementeerd. Maar dit is mede dankzij dat activities god objecten zijn binnen Android en alle netwerk calls op een aparte thread uitgevoerd worden. Tevens zijn er geen richtlijnen voor het bouwen van REST applicaties in Android.

Om duidelijk te maken wat de huidige restrictie zijn, pakken we als voorbeeld de profiel pagina en zijn respectieve activity.

```
private void retrievePractices() {
    PracticeApi service = (PracticeApi) ServiceProviderEnum.PRACTISE_SERVICE.getApiService();
    service.getAllPractices(new Callback<List<Practice>>() {
        @Override
        public void success(List<Practice> practices, Response response) {

        }

        @Override
        public void failure(RetrofitError error) {

        }
    });
}
```

Figuur 25: ophalen alle practices

Om de complexiteit te verlagen is de implementatie binnen succes en failure weggelaten

In figuur 25 is een methode te zien voor het ophalen van alle practices. Dit wordt gebruikt als de gebruiker van practice wil veranderen. Deze methode is een verdere uitbouw van de uitleg in hoofdstuk 5.3.1.

Er zijn een aantal problemen met deze implementatie. Ten eerste, de code geeft bij succes een lijst met practices terug. Als de lijst met practices aangepast moet worden zodat het beter past bij de activity, wordt dit uitgevoerd in activity klassen. Dus activity klassen voert manipulaties van data uit.

Ten tweede het chainen van netwerkcalls is erg lastig bij deze implementatie. Het zorgt ervoor dat de code erg complex wordt omdat er veel error handling moet worden toegevoegd. Met chainen bedoel ik dat de netwerk calls synchroon achter elkaar worden aangeroepen. Dit is normaal niet mogelijk omdat elke netwerk call op een aparte thread wordt uitgevoerd.

De primaire taak van de activity is het updaten/managen van de views. De activity klassen is vergelijkbaar met Controller binnen het MVC pattern. De activity manipuleert op dit moment de data en dit hoort niet bij zijn taken. Het liefst wordt dit verplaatst naar aparte klassen.

Oplossing

Ik heb gekozen om de netwerkcalls naar service klassen te verplaatsen. Ik heb hiervoor gekozen omdat de front- en back-end dan gelijk zijn in structuur. De front en back-end developers kunnen op deze manier de applicatie sneller lezen en begrijpen. Tevens kunnen ze hier makkelijker over discussiëren omdat de structuur gelijk is.

Om dit te realiseren moeten de service klassen de activities op de hoogte brengen als de netwerk calls klaar zijn. Zodat de activity zijn view kan updaten. Op dit moment is dat niet mogelijk omdat de netwerkcalls asynchroon lopen met de Activity. De activity wacht dus niet tot de netwerkcall klaar is.

Dit kan op een aantal manieren worden opgelost. Ten eerste, je kan een referentie naar de Activity opslaan in de service klassen. Als de call klaar is dan kan de service de juiste methode in activity klassen aanroepen. Dit is echter bad practice volgens Google omdat het memory leaks kan veroorzaken.

Ten tweede kan er gebruik worden gemaakt van een callback. Ook dit is niet de beste manier omdat Retrofit zelf al een callback gebruikt. Een callback zou dan een tweede callback aanroepen. Dit is geen slechte oplossing maar ook niet de beste.

De laatste oplossing die ik heb bedacht is om een event/message bus te implementeren. Als de service klaar is wordt er een event verstuurd naar de main thread. Alle luisterende methodes krijgen dan vervolgens het event binnen. Dit zorgt voor een lage zichtbare koppeling maar heeft als nadeel dat het er ook een single point of failure is. Ik heb gekozen voor deze oplossing omdat dit vergeleken met de rest het beste is. Een lage zichtbare koppeling zorgt voor verbeterde leesbaarheid. Dit lost het originele probleem het best op.

Voor de event/message bus heb ik gekozen om een open source library te gebruiken omdat dit simpelweg beter zal werken dan mijn eigen implementatie en waarom zou je het wiel opnieuw uitvinden. Hierbij heb ik gelet op populariteit en of ze goed passen bij mijn usecase.

Ik heb gekeken naar de volgende twee libraries:

- Otto
- EventBus

Ik heb gekozen voor Otto. Otto is een library gemaakt door Square dit zijn dezelfde makers als Retrofit. Otto gaat goed samen met Retrofit en heeft goede documentatie. De verschillen tussen EventBus en Otto zijn ook vrij klein.

Het alternatief EventBus is een library van Greenrobot. Ik heb voor de vergelijking tussen de twee de documentatie doorgelezen. Het grootste verschil dat ik tegen kwam was dat Otto op annotaties is gebaseerd en EventBus niet. EventBus schijnt hiermee een betere performance te hebben volgens de GitHub van

EventBus. Echter laat zo'n onderzoek niet zien of er bij normaal gebruik ook daadwerkelijk een performance verschil wordt opgemerkt.

Uiteindelijk is het uitgelopen op de "ease of use" zodat ik de library sneller kan implementeren. Betere documentatie is dan ook erg handig.

Implementatie

Voor de implementatie wordt er een singleton gebruikt. Net als bij Retrofit wordt dit gedaan wegens de performance. Elke keer opnieuw een bus aanmaken kost veel rekenkracht wat heel schaars is bij mobiele apparaten, daarnaast gaat dit ten koste van de batterij en verhoogt het de memory footprint.

Als voorbeeld pakken we weer Profiel activity.

In figuur 26 is te zien hoe de activity zichzelf registreert aan de bus.

```
@Override
protected void onStart() {
    super.onStart();
    bus.register(this);
}
```

```
@Override
protected void onStop() {
    super.onStop();
    bus.unregister(this);
}
```

Figuur 26: how to register (left) and unregister (right) to a bus

Vervolgens In figuur 27 is te zien hoe de activity de service klassen aanroept.

```
private void retrievePractices() {
    PracticeApi service = (PracticeApi) ServiceProviderEnum.PRACTISE_SERVICE.getApiService();
    service.getAllPractices();
}
```

Figuur 27: practice service call to retrieve all practices

PracticeService bevat de methode getAllPractices die origineel in activity stond. Bij succes post het een LoadPracticeEvent en bij falen een failed event. Om events te posten hoeft de service klassen zich niet te registreren dit is alleen nodig als het luistert.

```
public void getAllPractices() {
    practiceService.getAllPractices(new Callback<List<Practice>>() {
        @Override
        public void success(List<Practice> practices, Response response) {
            bus.post(new LoadPracticeEvent(practices));
        }

        @Override
        public void failure(RetrofitError error) {
            error.getMessage();
            bus.post(new FailedToLoadModelEvent(error.getMessage()));
        }
    });
}
```

Figuur 28: get all practices methode

Doormiddel van subscribe en de parameter LoadPracticeEvent zoals in figuur 29 geef je aan dat de activity luistert naar een LoadPracticeEvent.

```
@Subscribe
public void onPracticeLoaded(LoadPracticeEvent event) {
    allPractices = event.getPractices();
    buildPopup(convertPractice(allPractices), "Practice", practiceListener);
}
```

Figuur 29: event on practices loaded methode

Er worden nu meerdere stappen uitgevoerd en dit zorgt voor minimale code duplicatie. De klassen zijn leesbaarder, omdat ze eenduidiger en korter zijn. Ook kunnen de errors nu op een generieke manier worden afgehandeld.

Klassendiagram

Ik heb als eerste een klassendiagram gemaakt. Deze is te zien in figuur 30.

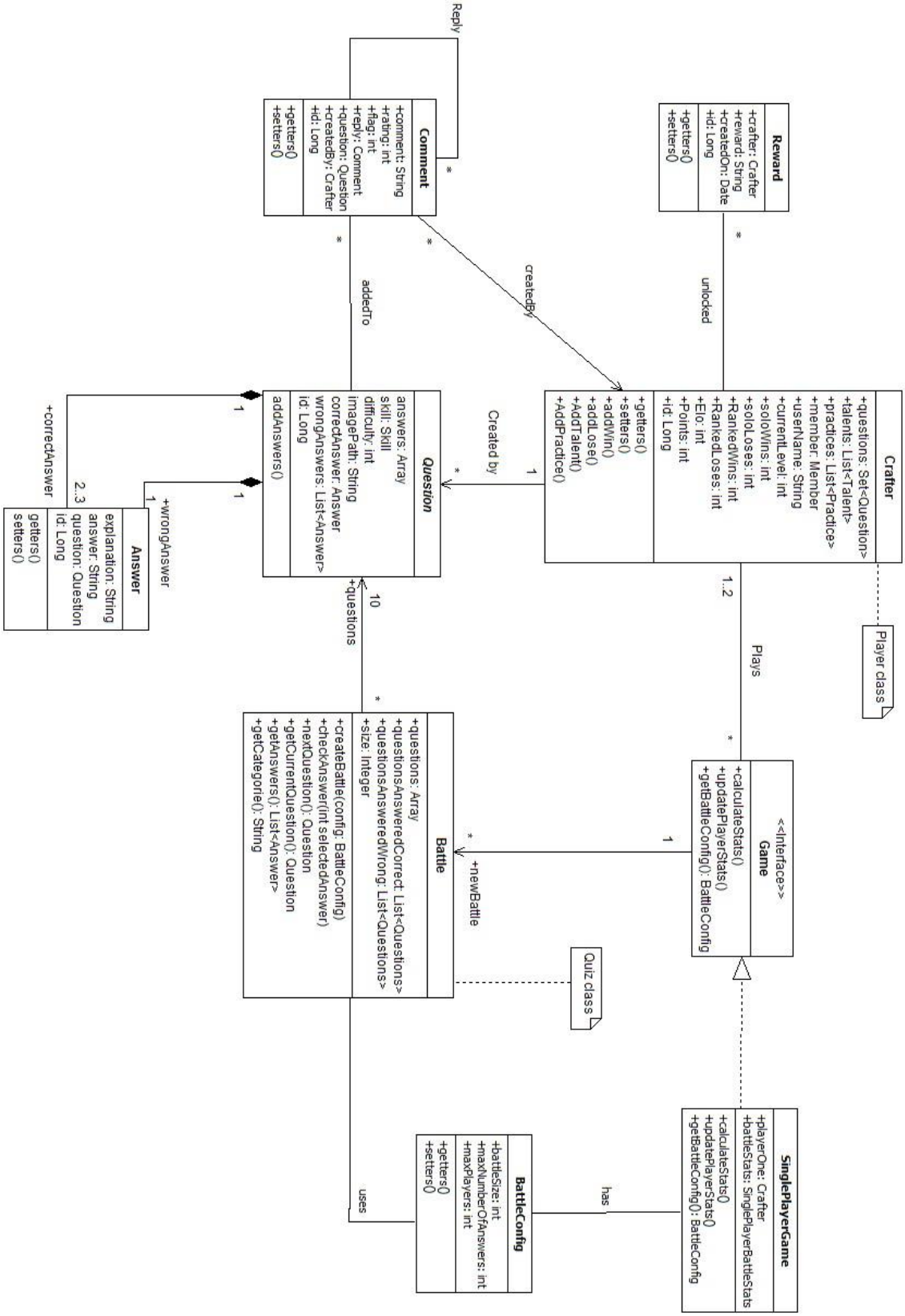
Het klassendiagram geeft een deel van de front-end weer. Ik heb geen klassen diagram voor de back-end gemaakt omdat deze vrijwel identiek is aan de front-end qua data. De front-end heeft echter ook klassen zoals battle en game.

In het klassen diagram zijn er een aantal verschillen te zien ten opzichte van het diagram uit iteratie 2. Game klassen is nu een interface en geen concrete klasse. SinglePlayerGame implementeert deze game klassen. Ik heb gekozen om Game een interface te maken om een aantal redenen.

Ten eerste, volgende iteratie staat multiplayer op de planning. Ik verwacht een klasse MultiplayerGame. Deze klasse zal ook Game implementeren. Dit doe ik zodat beide game klassen een vast aantal methodes hebben die gebruikt worden door de game activity. Dus de klassen moeten alle interface methode implementeren.

Ten tweede gebruik ik een interface in plaats van abstracte klassen omdat de implementatie volledig wordt overgelaten aan de klassen die de interface implementeren. Waarbij je bij abstracte klassen vast zit aan een klassen die je in zekere zin instructies geeft voor wat je moet uitvoeren.

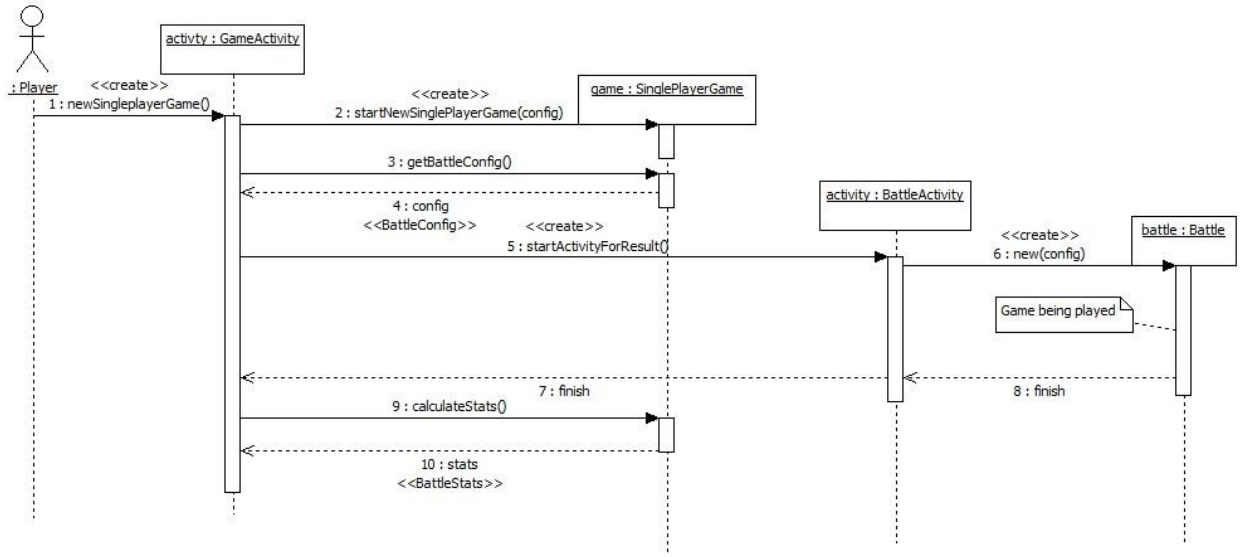
Ik verwacht dat SinglePlayerGame meerder interfaces zal implementeren. De naam game kan hierbij tot verwarring leiden. Ik had het eventueel anders kunnen noemen, zoals playable, maar ook daar kan over worden gediscussieerd.



Figuur 30: klassendiagram iteratie 4

BattleConfig

Singleplayer game heeft een BattleConfig. Dit is simpelweg de configuratie voor een battle. De battle klasse zou zichzelf kunnen bouwen als het weet waar hij op moet letten, zoals het aantal vragen. Dit is ook nodig omdat game klasse niet een battle klasse kan/mag aanmaken. Dat komt omdat Battle en Game een eigen activity (scherm) hebben. Een normale Java object mag/kan geen activity creëren. Alleen activities mogen elkaar starten.



Figuur 31: sequence diagram, create new singleplayer battle

In het sequence diagram is dit goed te zien. Je ziet hoe de activity de Java objecten creëren. In het diagram zijn stappen zoals netwerk calls (services), BattleConfig weggelaten om het overzicht te bewaren. Het is meer bedoeld om te laten zien wat voor rol activity heeft aangezien ik dat niet in me klassendiagram heb verwerkt.

```
public void startNewSinglePlayerBattle(View view) {
    multiplayer = false;
    game = new SinglePlayerGame(crafter);
    BattleConfig config = game.getBattleConfig();
    Intent intent = new Intent(this, BattleActivity.class);
    intent.putExtra("multiplayer", multiplayer);
    intent.putExtra("config", config);
    startActivityForResult(intent, 1);
}
```

Figuur 32: methode voor het creëren van een singleplayer battle

Iets wat opvalt in figuur 32 is dat ik startActivityForResult() gebruik in plaats van startActivity. Als battleActivity klaar is dan keer je terug naar GameActivity. Game activity wordt in de tussentijd in de memory opgeslagen zodat de staat niet verloren gaat. Dit is nodig zodat de game gegevens niet verloren gaan. Normaal gesproken zou de activity destroyed worden.

Als alternatief had ik game als global variabel kunnen opnemen maar dat is minder netjes.

Comment en reward

Als laatste zijn de twee nieuwe klassen comment en reward toegevoegd. Wat de rewards precies zijn is nog niet duidelijk. Voorlopig heb ik als placeholder gekozen om kleuren als reward te gebruiken. Dit is tijdelijk oplossing en veranderd als het plan concreter wordt. De kleur wordt als hex code string opgeslagen. De reward wordt random gegenereerd aan het einde van een battle. Dit wordt gedaan in de back-end zodat er niet vals gespeeld kan worden.

Verder is het commentsysteem erg simpel en ziet het er als volgt uit:

Figuur 33: comment pop-up

5.4.3 Testen

Ik heb gedurende deze iteratie een aantal unit testen geschreven en veel exploratory testen en error guessing uitgevoerd. De unit testen hoor ik goed bij te houden alleen ben ik daar niet aan toe gekomen. Alles wat ik heb gebouwd is in principe gebaseerd op de user stories.. Hierbij heb ik per onderdeel gekeken of het systeem doet wat het moet doen. Zo heb ik het comment systeem getest door comments toe te voegen, hierop te reageren. Vervolgens heb ik gekeken of de comments ook goed worden opgeslagen in de database en goed worden opgehaald. Verder zou ik ook acceptatie testen willen schrijven. Deze ga ik uitvoeren zodra de applicatie wordt gedeployed.

5.4.4 Reflectie

Deze iteratie ging minder goed dan verwacht. Doordat ik weer opnieuw een groot gedeelte van de applicatie ben gaan refactoren is de code slordig. Hoewel alle functionaliteiten het gewoon doen had ik nog steeds het gevoel dat het beter kon. En dus was ik ook niet klaar met refactoren.

Ik heb gekozen voor iteraties van twee weken. Hierbij werk steeds een klein onderdeel van het systeem uit. Hoewel dit goed werkt voor dit project zijn hier ook valkuilen aan verbonden. Zo kan ik snel het globale beeld verliezen. Doordat ik elke iteratie een klein stuk van het systeem in UML uitwerk en de interactie tussen de verschillende onderdelen mis.

Verder heb ik een opmerking over het gekozen proces. Op het moment heb ik geen issue systeem. De bugs die ik tegenkom kan ik nergens goed opnemen. Dit maakt het bijhouden en managen van de issues lastig. Het liefst wil je alle issues koppelen aan een userstorie waarbij elke commit die ik uitvoer gekoppeld is aan een issue. Zo weet ik zeker dat alle problemen zijn vastgelegd. Dit is vooral belangrijk als het systeem groeit. De programmeurs die later met dit project aan de slag gaan, kunnen dan terugzien welke issues het systeem had/heeft en hoe deze zijn opgelost. Zover ik weet heeft CGI hier geen omgeving voor en ik heb dat ook nergens kunnen lezen/vinden. Wel hadden we een scrumboard opgezet voor dit project, hierbij was ook het idee om alle user stories per iteratie op te nemen en deze uit te werken indien dat nodig blijkt te zijn. Deze hebben we echter snel achterwege gelaten gezien de moeite die erin gestoken moest worden in vergelijking met de grootte van het project.

Tot slot is deployment heel belangrijk en dat is helaas deze iteratie niet gelukt. Dit is wel cruciaal voor dit project. Ik heb voor de deployment een server nodig waar de backend op kan draaien. Als dit niet gebeurt kan ik ook niet de functionaliteiten testen en meten om te kijken of de gekozen methode werkt en wat er kan worden verbeterd.

5.5 Iteratie 5

5.5.1 Iteratie planning sessie

Iteratie vier eindigde met een meeting waarin we een aantal zaken hebben besproken.

Terugkoppeling iteratie vier

Iteratie vier liep goed. De meeste geplande functionaliteiten waren in zijn basis geïmplementeerd. Het was echter niet gelukt om een server op te zetten. Hierdoor konden we aan het einde niet deployen. De app was dus niet speelbaar. De opdrachtgever zou kijken naar alternatieven zoals een server huren.

Taken iteratie vijf

De taken voor iteratie vijf zijn als volgt:

Ten eerste moet het game gedeelte van de app gerefactored worden zodat het multiplayer beter ondersteund. Hierbij wordt ook de navigatie demo geïmplementeerd. De opdrachtgever heeft gekozen voor optie 1 zoals te zien is in figuur 16.

Ten tweede wordt multiplayer toegevoegd. Dit stond gepland voor deze iteratie. Multiplayer is een battle waarbij er twee spelers tegen elkaar strijden. Naast multiplayer kijk ik ook naar de ranking en de login systeem.

Als laatste worden alle functionaliteiten die tot nu toe zijn geïmplementeerd verbeterd en verfijnd voor zover dat kan.

Tot slot heb ik ook afgesproken om de app te deployen indien dat mogelijk blijkt te zijn (server beschikbaar, gehuurd of intern).

In tabel 11 is een korte opsomming van de taken te zien:

Taak	Omschrijving
Comment systeem	Verbeteren
Level systeem	Verbeteren
Rewards	Verbeteren
Ranking	De speler kan stijgen in rang en dit is te zien op een leaderboard.
Profiel pagina	Verbeteren
Refactoren	Er moet nog veel gerefactored worden, voordat multiplayer geïmplementeerd kan worden.
Navigatie app wijzigen	De simpele demo app die ik had gebouwd in iteratie 2 moet geïmplementeerd worden.
Multiplayer	De spelers kunnen tegen elkaar spelen.

Tabel 11: samenvatting taken iteratie 5

5.5.2 Navigatie demo

In iteratie twee had ik een navigatie demo gemaakt. Dit is belangrijk omdat het invloed heeft op de interne structuur van de app.

De opdrachtgever heeft gekozen voor optie 1 zoals te zien in hoofdstuk 5.2.3. Hij vond dit fijner werken dan optie 2.

5.5.2.1 Implementatie

Bij optie één gebruik je tabs. Het komt erop neer dat elke scherm een eigen fragment heeft die allemaal door een activity worden beheerd.

Voor de navigatie demo had ik gebruik gemaakt van een actionbar met tabs. Dit was de standaard voor API 21. Echter is dit sinds API 21 deprecated. Voor de demo had dit geen invloed omdat looks niet veranderen.

Voor CGICraft is het wel belangrijk om een alternatief te gebruiken. Depracted is in principe een waarschuwing dat zegt dat er betere alternatieven zijn. Google heeft actionbar vervangen door toolbar omdat dit beter past bij material design. Daarnaast is het ook flexibeler. Toolbar kan namelijk als een view worden toegevoegd. De plaatsing van toolbar is dus vrij om te kiezen. Bij actionbar was dit niet het geval. Het zat standaard bovenaan het scherm en was moeilijk aanpasbaar.

Toolbar is toegevoegd in API 21, oudere API's hebben dus geen ondersteuning voor toolbar. Voor backwards compatibility biedt google een support library. Dit bevat belangrijke componenten van nieuwe API's die ook ondersteund worden door oudere API's waarbij de functionaliteiten voor een groot deel hetzelfde zijn. Voor dit project heb ik de nieuwste support library gebruikt, dat was op het moment 22.

De implementatie was simpelweg een aantal methodes refactoren zodat ze gebruik maken van de support library. Voor de implementatie van de navigatie demo heb ik de bestaande activities omgezet naar fragments. Dit was goed te doen omdat de lay-out in XML wordt gedefinieerd binnen Android. Dit kon ik dus hergebruiken.

Naast de navigatie demo heb ik ook de rest van de applicatie omgezet zodat het gebruik maakt van de support library. Dit zorgt voor een betere backwards compatibility.

5.5.3 Bouwen Multiplayer

5.5.3.1 Gameplay

Multiplayer is lastig. Zowel de opdrachtgever en ik weten niet precies hoe de gameplay (manier waarop de gebruikers interacteren met de wereld) eruit komt te zien. De gamificatie expert was op vakantie en hem kon ik dus niet raadplegen. Daarnaast heb ik ook geen guidelines of regels gevonden voor multiplayer op het web.

Als eerste heb ik bestaande multiplayer spellen bekeken om zo ideeën op te doen. Echter geen enkele app paste goed bij onze use case. Uiteindelijk heb ik gekozen om de gameplay over te laten aan de gebruikers. Hierbij wordt als eerste een basis (multiplayer) spel gebouwd, waar ik vervolgens feedback/ideeën over zal vraag.

De gameplay heb ik opgenomen in het game designdocument bijlage C.

5.5.3.2 Architectuur multiplayer

Omdat ik over geen enkele kennis beschikte met betrekking tot multiplayer heb ik hier eerste onderzoek naar gedaan. Hierbij heb ik gezocht naar bestaande architecturale oplossingen zodat ik een global beeld kan schetsen voor de communicatie tussen de clients en/of server. Dus hoe communiceren twee spelers met elkaar of met de server?

Ik heb twee oplossingen gevonden die ik kan toepassen voor dit project. Ten eerste kun je gebruik maken van een server. Beide clients communiceren met de server. De server managet vervolgens de staat van beide clients en geeft zo nodig messages aan elkaar door. De server fungeert hier als een manager/doorgeefluik.

De tweede oplossing is een peer to peer connectie. Hierbij valt de server weg en communiceren de twee clients direct met elkaar.

Ik heb gekozen voor client server architectuur omdat dit al voor een groot deel al is gerealiseerd. Hierdoor is de implementatie simpel, er hoeft weinig aangepast te worden voor multiplayer als we kijken naar de huidige

architectuur. Daarnaast is het goed schaalbaar en is security eenvoudig om te implementeren. Het nadeel is dat er een server gehuurd moet worden en dit kost geld.

Peer to peer in tegenstelling tot client server is erg lastig om goed te realiseren. Er moet rekening worden gehouden met discovery (dus hoe vinden de spelers elkaar), dit wordt alleen maar erger zodra de speler achter een firewall zit. Er moet rekening worden gehouden met onstabiele internetverbindingen en langzame internetverbindingen. Verder is security lastig om te implementeren. Daarnaast is het moeilijker om vals spelen tegen te gaan bij peer to peer.

Een alternatief zou zijn om de Cloud te gebruiken. Dit was misschien wel de simpelste en beste optie omdat het standaard ingebouwd zit in Android doormiddel van Google Play service. Echter was dit niet mogelijk omdat de app alleen intern gedraaid wordt en CGI niks in de Cloud wil hebben.

Huidige situatie

In de huidige situatie gebruik ik REST (http) als communicatieprotocol. De communicatie verloopt van client naar server. De client stuurt een request naar de server, de server reageert hierop met een response. De server kan geen messages versturen naar de client zonder een request.

Dit is niet handig voor multiplayer waarbij de server de client op de hoogte moet brengen als de staat van het spel veranderd. Bijvoorbeeld hoe laat de server de client weten als er een tegenstander is gevonden? Dit zou je kunnen oplossen doormiddel van polling. In dit geval controleert de client met een vast interval of er een tegenstander is gevonden. Constante polling is niet gunstig voor smartphones. Rest messages zijn relatief zwaar en gebruiken dus veel data. Bovendien gaat deze oplossing ten koste van de accu omdat de client constant netwerk calls maakt. Tevens is het ook zwaarder voor de server. Er worden constant verbindingen geopend en gesloten.

Voor een multiplayer game gebruik je het liefst een lichte full-duplex communicatieprotocol. De huidige communicatieprotocol is dus niet geschikt voor multiplayer.

TCP en UDP

In vergelijking met http zijn TCP en UDP een stuk lichter. Ze hebben weinig overhead dus kost het de user weinig data. Bij TCP en UDP wordt er een poort opengezet. Dit zorgt voor een full-duplex verbinding. Hierdoor kan zowel de client als de server tegelijkertijd messages versturen. Iedereen die naar dat poort luistert ontvangt messages. De client hoeft niet constante calls te maken dit scheelt data en accu.

Het verschil tussen TCP en UDP is dat TCP meer robuust is. Dit komt doordat het controleert of elke package goed en in de juiste volgorde is aangekomen bij de client. Het heeft dus errorhandling. UDB heeft dat niet maar is daardoor sneller.

Ik heb gekozen voor TCP. Performance is niet relevant voor dit project. Het gaat om een quiz, snelle communicatie is geen vereiste zoals dat wel het geval is bij veel multiplayer spellen. Het is wel handig als de communicatie robuust is. Zo wordt het makkelijker om error handeling uit te voeren. Dit is nodig omdat mobiele verbindingen minder stabiel zijn dan bij een pc met LAN kabel. Met TCP kan ik onstabiele smartphone verbinding beter opvangen.

WebSocket

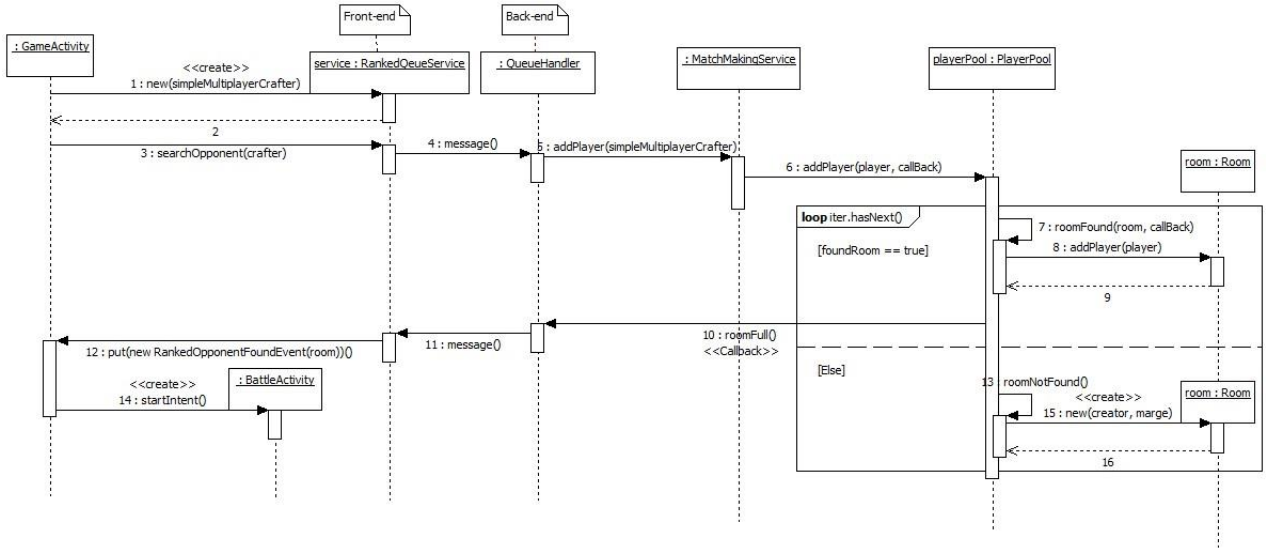
Voor de daadwerkelijke connectie maak ik gebruik van WebSocket. WebSocket maakt verbinding met de server middels http en zorgt doormiddel van een upgrade dat het wordt omgezet naar een TCP connectie. Dit is handig omdat backend alleen de porten 80 (http) en 443 (https) open heeft staan (i.v.m. security).

5.5.3.3 Implementatie

Multiplayer kan opgedeeld worden in twee onderdelen. Het eerste onderdeel is het zoeken naar een tegenstander en het tweede onderdeel is het spel zelf. De gedacht was om dit te splitsen zodat er een duidelijk onderscheid is tussen alles wat voor het spel gebeurt en alles wat tijdens het spel gebeurt.

Matchmaking

De matchmaking kan gezien worden als de game klassen in het klassendiagram figuur 40. Als er een multiplayer game gestart wordt zoekt de speler eerst naar tegenstanders van zijn niveau. Dit wordt gedaan doormiddel van de queueservice (wat overigens na iteratie 5 vertaald is naar MatchmakingService, zodat het beter aansluit op het doel van de klasse).



Figuur 34: sequence diagram, creëren multiplayer battle

In sequence diagram 34 is te zien welke stappen er voor matchmaking ondernomen worden. GameActivity maakt een nieuwe RankedQueueService aan en geeft hierbij als argument een SimpleMultiplayerCrafter(SMC) instance mee. Dit is een model class gebaseerd op crafter. Deze class bevat alleen gegevens die relevant zijn voor multiplayer om zo het dataverbruik zo min mogelijk te houden.

De rankedQueueService opent een WebSocket connectie zoals te zien is in figuur 35 en verstuurd de SMC gegevens als een JSON naar de server.

```
public void searchOpponent() {
    AsyncHttpClient.getDefaultInstance().websocket(RANKED_QEUE_URL,
            null,
            new AsyncHttpClient.WebSocketConnectCallback() {

                @Override
                public void onCompleted(Exception ex, WebSocket socket) {
                    if (ex != null) {
                        Log.v(TAG, "WebSocket connection failed...", ex);
                        return;
                    }
                    webSocket = socket;
                    webSocket.send(crafterToJsonString(crafter));

                    webSocket.setStringCallback(onResultListener);
                    webSocket.setClosedCallback(onConnectionClosedListener);
                }
            });
}
```

Figuur 35: methode om connectie te leggen met de server

De JSON komt binnen als TextMessage bij de QueueHandler en wordt terug geparsed naar SMC. (Dit is weggelaten in het sequence diagram om het overzicht te behouden).

De speler wordt aan MatchmakingService toegevoegd. Deze voegt vervolgens de speler aan een playerpool toe. De playerpool bevat alle kamers waar alle spelers over zijn verdeeld. Een kamer is een klasse waar alle spelers die samen een spel gaan spelen zijn verbonden. Het idee van een kamer is dat het als wachtruimte fungeert. Elke kamer heeft een ranking range. Als een speler zich wil aansluiten bij een kamer dan moet zijn ranking binnen de range van de kamer vallen. Tevens heeft elke kamer een unieke id. Dit id is een gecombineerde id van alle spelers. Zo kunnen er nooit twee kamers met dezelfde id tegelijkertijd bestaan. Zodra de speler is toegevoegd aan de playerpool wordt er gezocht naar een kamer waar hij zich kan aansluiten.

Als de speler geen enkele kamer vindt wordt er een nieuwe kamer aangemaakt, hierbij wordt de ranking van de speler de basis ranking van de kamer.

Indien de speler wel een kamer vindt (en de kamer vol is wat altijd het geval is voor twee spelers) wordt de QueueHandler op de hoogte gebracht doormiddel van een callback zodat het de kamer gegevens kan doorsturen naar alle (aangesloten) spelers.

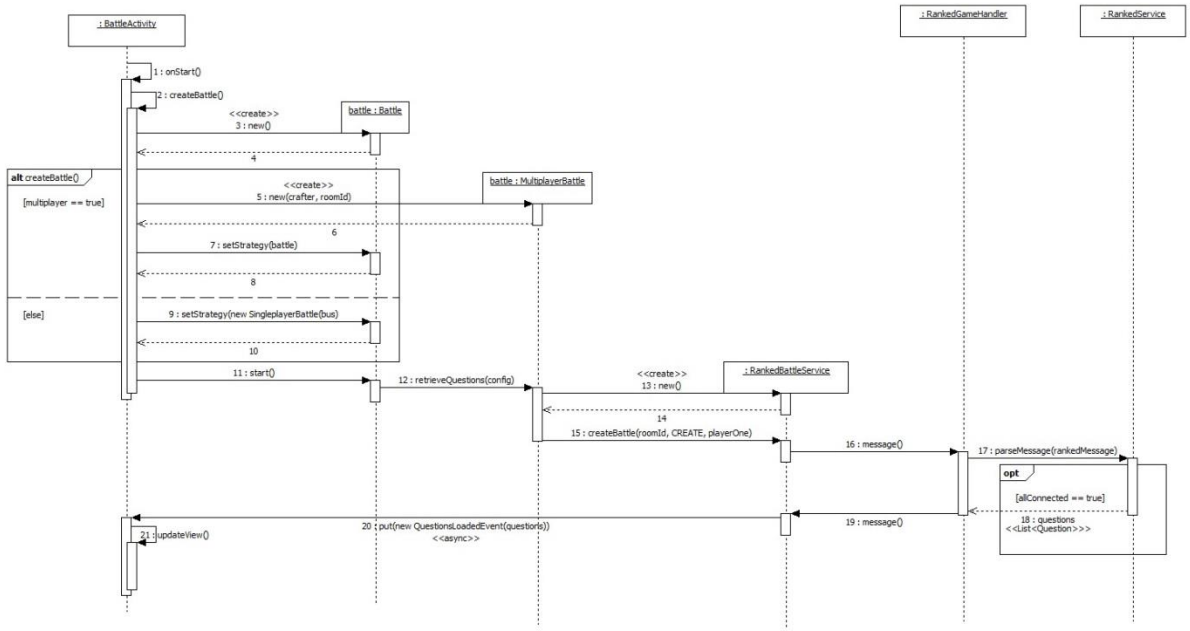
```
@Override
public void onMessageReceived(String message) {
    room = jsonStringToRoom(message);
    bus.post(new RankedOpponentFoundEvent(room));
    close();
}
```

Figuur 36: methode, die berichten ontvangt van de server

De QueueRankedService (app) ontvangt de message in de bovenstaande methode. De string wordt omgezet naar een Kamer klasse en dit wordt gepost als event. De GameActivity luistert naar dit event. Zodra het de event ontvangt wordt het tweede onderdeel gestart, BattleActivity.

Create Battle

Nadat de spelers elkaar hebben gevonden wordt het spel gecreëerd zoals te zien is in sequence diagram 38.



Figuur 37: sequence diagram, ophalen vragenlijst

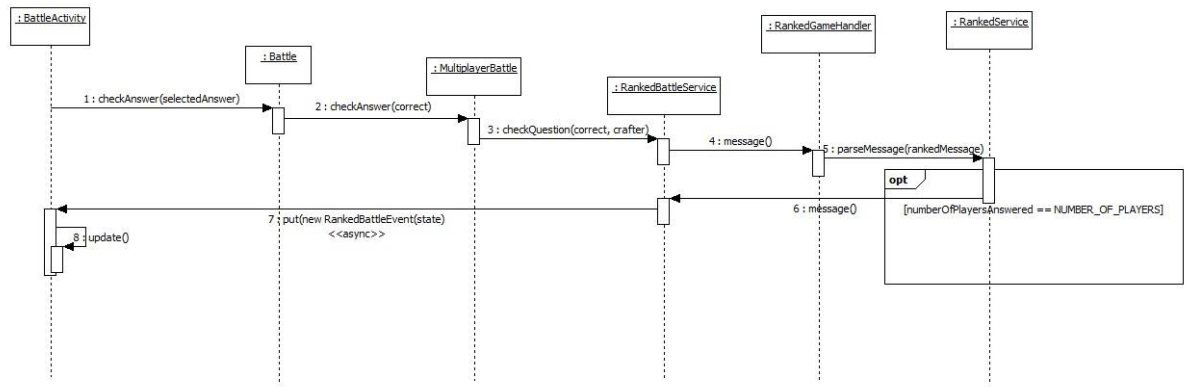
In het klassendiagram gaat het hier om een strategie pattern. De battle kan run time tussen single en multiplayer switchen. In praktijk wil men alleen van multiplayer naar singleplayer. Hiervoor heb ik gekozen met de gedachte dat dit een betere gebruikerservaring levert voor de speler. Hij kan zelf kiezen of hij de quiz afmaakt of stopt. Als de speler de quiz afmaakt wordt het gerekend als een singleplayer game. Als hij stopt dan zijn daar geen consequenties aan verbonden. Een alternatief zou zijn om hem terug te sturen naar de homescreen.

Afhankelijk van strategie worden de vragen opgehaald doormiddel van REST of WebSocket. In het sequence diagram laat ik alleen multiplayer zien.

Voor de creatie wordt de status CREATE gebruikt. De server weet dan dat het spel nog niet is gestart en voegt de speler toe aan een match. Match is weggelaten in het sequence diagram. Het zou geplaatst worden na RankedService. Ik heb het weggelaten omdat het niet heel relevant is voor het verhaal. Alle spelers worden toegevoegd aan een match gebaseerd op hun roomId. Zodra alle spelers zijn toegevoegd wordt er een random vragenlijst gegenereerd. Deze vragenlijst wordt verstuurd naar de alle spelers in de match.

BattleService weet dat het een vragenlijst zal ontvangen omdat de status CREATE is. Na het ontvangen van de vragenlijst stuurt BattleService dit verder door naar BattleActivity d.m.v. een event. BattleActivity luistert naar dit event en update de view zodra het de lijst ontvangt.

Play Battle



Figuur 38: sequence diagram, controleren spelers

In sequence diagram 38 is te zien wat er tijdens het spel gebeurd. Elke keer als de speler een vraag beantwoord en deze wil controleren wordt het volgende uitgevoerd. RankedBattleService stuurt een RankedMessage (als JSON) met de status UPDATE. Hierbij neemt hij ook op of de vraag goed is beantwoord. De back-end ontvangt deze message en parsed het zoals in figuur 39.

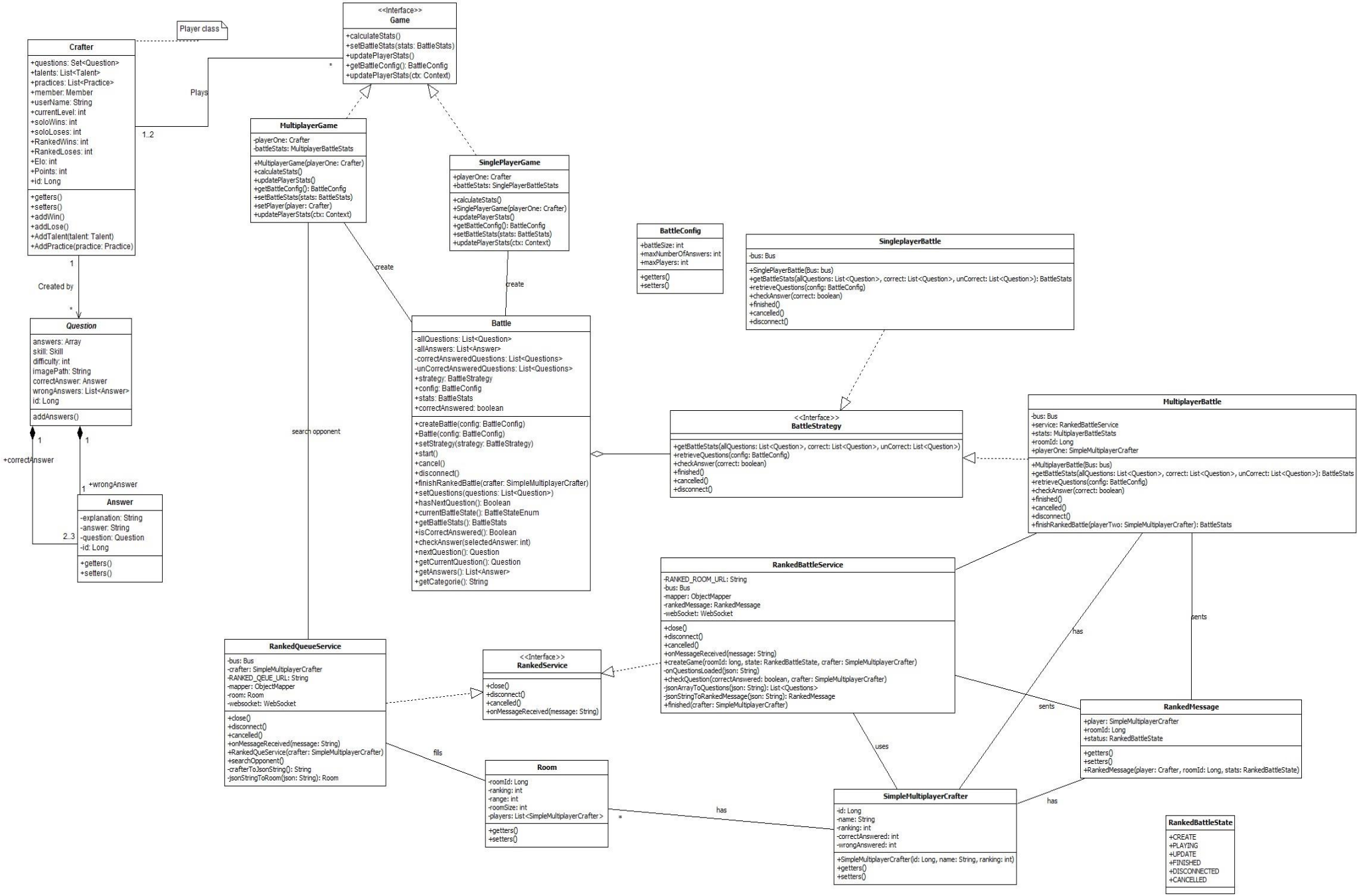
Als alle spelers de vraag hebben beantwoord krijgen ze een RankedMessage terug met PLAYING als status. Deze status geeft aan dat er verder gespeeld mag worden e.g. volgende vraag wordt getoond.

Dit blijft zo door gaan tot er geen vragen meer zijn. Dan wordt als laatste de status FINISHED doorgegeven. Bij FINISHED worden de statistieken van de match berekend en opgeslagen in de database. Deze worden vervolgens getoond in de app.

Als de speler connectie verliest of de match cancelled dan wordt de match verwijderd en alle overige spelers die daar nog in zitten worden op de hoogte gebracht. De battle wordt vervolgens voor deze spelers omgezet naar singleplayer. Er is op het moment geen verschil tussen cancelled of disconnect.

```
public void parseMessage(RankedMessage message) {
    switch (message.getStatus()) {
        case CREATE:
            addMatch(message);
            break;
        case DISCONNECTED:
            removeMatch(message);
            break;
        case CANCELLED:
            removeMatch(message);
            break;
        case UPDATE:
            checkQuestion(message);
            break;
        case FINISHED:
            finish(message);
            break;
        default:
            removeMatch(message);
            break;
    }
}
```

Figuur 39: methode voor het parsen van RankedMessage



Figuur 40: klassendiagram multiplayer

5.5.4 Testen

De multiplayer en singleplayer heb ik getest met behulp van exploratory testing, error guessing en unit testen.

Ik heb tijdens deze iteratie veel aangepast. Dit was nodig omdat ik mijn strategie die ik eerder had bedacht moest veranderen zodat het beter aansluit bij multiplayer. Dit was niet erg maar zorgde ervoor dat ik singleplayer opnieuw moest testen.

Ik heb multiplayer en singleplayer functioneel getest door het spel te spelen en verschillende scenario's na te bootsen. Hierbij was de uitgang van de test de user stories. Zo heb ik getest wat er gebeurt als een speler het spel verlaat of zijn internet uitvalt. Scenario´s waarbij netwerk een rol speelt waren erg lastig om te testen omdat het netwerk variabel is.

Verder heb ik met behulp van testen een race conditie ontdekt in multiplayer. Tijdens de tweede queue als alle spelers aan een match verbinden, is er een kans dat de spelers die op exact dezelfde moment verbinding maken met het systeem beide een match genereren en dus nooit het spel spelen omdat de kamer nooit vol raakt. Het maken van de match wordt asynchroon uitgevoerd vanwege efficiëntie. Ik wist niet hoe ik dit probleem het beste kon oplossen.

Ik heb getest met behulp van verschillende Android devices, versies en emulators. Dit was moeizaam want twee devices tegelijkertijd besturen zonder een server dus doormiddel van twee usb kabels zorgt niet voor de beste test ervaring. Het heeft niet direct invloed op het testen maar het is minder efficiënt.

Daarnaast had ik geprobeerd om de WebSocket te mocken en deze te gebruiken voor unit testen maar dat is mij niet gelukt. Ik had ook collega´s om hulp gevraagd maar die wisten er ook geen raad mee. Ik heb het voorlopig gelaten voor wat het is omdat ik er niet te lang aan wil vast zitten. Ik schiet tekort in kennis op dit gebied.

5.5.5 Reflectie

Dit was de lastigste iteratie tot nu toe om een aantal redenen. Ik had geen ervaring met multiplayer. Ik wist niet wat de beste architectuur zou zijn. Ik wist niet welke technieken er gebruikt worden en hoe deze goed kunnen worden toegepast. Daarnaast kon ik weinig informatie vinden op dit gebied. Dit heeft bij mij tot de vraag geleidt of WebSocket wel de beste oplossing is voor dit project. Ik had liever een bestaande oplossing gebruikt zoals google play service dan weet ik zeker dat het werkt. Aan de andere kant zou ik dan minder hebben geleerd. Daarnaast weet ik ook niet of de WebSocket implementatie goed is. Het werkt maar is het uitbreidbaar, managebaar en kan je het beveiligen. Ik kon hier geen duidelijke antwoorden op vinden. Overigens gaat het wel om een klein project en weet ik niet hoe relevant deze punten nou echt zijn. Ik moet me hier meer in verdiepen.

5.6 Iteratie 6

5.6.1 Iteratie planning sessie

Terugkoppeling iteratie vijf

Iteratie vijf verliep goed. Ik heb op ranking na alles wat op de planning stond uitgevoerd. Ik heb ranking niet kunnen realiseren omdat ik meer tijd kwijt was aan multiplayer dan gepland. Dit was niet erg, aangezien ik ranking al eens eerder heb onderzocht en alles wat ik nodig heb al bestaat e.g. database, models, etc. Het enige wat ik mis is de berekening en het tonen van de ranking na een multiplayer game. De opdrachtgever vond dit overigens niet erg.

Verder hebben we multiplayer besproken gevolgd door een demo. Hierbij heb ik uitgelegd wat, hoe en waarom ik bepaalde keuzes heb gemaakt zoals WebSocket. Tijdens de demo kreeg de opdrachtgever ook de nieuwe navigatie van de app te zien. De opdrachtgever had hier geen opmerkelijke opmerkingen over.

Taken iteratie zes

Iteratie 6 (en ook 7) zullen afwijken van de standaard. Hiermee wil ik zeggen dat er geen grote functionaliteiten meer worden toegevoegd. Tijdens iteratie 6 licht de focus voornamelijk op het verbeteren van de code en de documentatie.

Om dit punt te ondersteunen wordt tijdens iteratie 6 mijn code bekeken door een architect (code review). Hierbij bepaalt hij de kwaliteit van de code en geeft mij richtlijnen/feedback om deze te verbeteren. Ook het verbeteren van de code staat gepland voor iteratie 6. Bovendien zal hij ook mijn verslag/documentatie lezen. Verder zal ik multiplayer verbeteren zodat het beter bestendig is tegen uitval en tot slot wordt ranking toegevoegd.

5.6.2 Ranking

Ik heb gekozen voor het ELO rating systeem. Dit systeem geeft doormiddel van een getal de sterkte van een speler weer. Het wiskundige systeem is bedacht voor schaken door de Amerikaanse natuurkundige en schaker Árpád Élő.

Ik heb gekozen voor het ELO rating systeem omdat dit goed schaalbaar (m.b.t. aantal spelers) en goed aanpasbaar is. Zodat het snel ingezet kan worden voor CGICraft. Daarnaast is het een bekende rating binnen de gaming wereld. Zo maakt het meest populaire spel [13] op dit moment gebruik van het ELO rating.

Het mooie van ELO rating is, dat het niet kijkt (tijdens het uitdelen van punten) naar de sterkte van een speler maar naar het percentage kans dat de speler heeft om te winnen. De sterkte van een speler is lastig om vast te leggen. Bijvoorbeeld: we hebben een speler x en een speler y. Beide hebben een standaard rating van 1200. In praktijk is speler y beter dan speler x. Het systeem houdt hier niet rekening mee maar kijkt puur naar de kans om te winnen. Dat 50% is voor beide spelers. Aan de hand hiervan worden de punten berekend. Als de speler goed is zal hij automatisch stijgen en vice versa.

Voor de realisatie heb ik simpelweg de volgende formule geïmplementeerd.

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}.$$
$$R'_A = R_A + K(S_A - E_A).$$

```
private double calculateNewElo(final double rankingToCalculate,
                               final double rankingOpponent,
                               final double score,
                               final double kFactor) {
    double expectedScore = 1 / (1 + Math.pow(10, (rankingOpponent - rankingToCalculate) / 400));
    return rankingToCalculate + Math.round(kFactor * (score - expectedScore));
}
```

Figuur 41: ELO formule (boven), implementatie ELO formule (beneden)

5.6.3 Code review

De code review was niet samen. De architect heeft de code in zijn eigen tijd doorgenomen. Ik heb alleen het repository URL doorgegeven.

De architect had geen opmerkelijke of foute code gezien. Wel had hij een aantal opmerkingen die veel tijd zouden kosten om te verbeteren, waaronder error handling en Javadoc.

Voor error handling had ik op veel plaatsen e.printStackTrace() gebruikt. Dit was volgens de architect geen goede oplossing. In plaats daarvan moet ik de errors loggen (zodat ze niet verloren gaan) en afhandelen. Om error handling goed af te handelen heb Ik zowel binnen CGI als op het web gezocht naar best practices op het gebied van error handling. Vervolgens heb ik deze zo goed mogelijk toegepast.

De tweede opmerking was dat ik geen Javadoc heb geschreven. Dit klopt en moet dus nog gebeuren.

Verder had de architect geen opmerkelijke of foute code gezien. Maar om de code voor de zekerheid te controleren moest ik Sonar installeren. Dit heb ik dan ook gedaan. Sonar gaf een aantal criticals. En mijn technical debt was 5%. Volgens CGI is 5% net nog goed. Het liefst wil je bij een klein project een zon laag mogelijk technical dept omdat dit lastiger zal zijn om later te verbeteren, de technical dept moest dus worden verbeterd. Ik heb de criticals en andere violations opgelost. Na mijn verbeteringen was mijn technical dept 0.4% hier was de opdrachtgever blij mee. Een voorbeeld van een critical: ik had Equals overridden zonder Hashcode.

Om de code verder te verbeteren heb ik naast Sonar ook PMD gebruikt. PMD is volgens deze paper [14] de beste tool als het gaat om code kwaliteit. Ik kreeg in PMD ook een stuk meer violations dan in Sonar. Ongeveer 1500. Ik heb alle violations doorgenomen en verbeterd indien dat nodig was. Een valkuil van PMD is dat je vrijwel alleen bezig bent met het verbeteren van violations, daar is PMD echter niet voor bedoeld het is meer een guideline.

5.6.4 Security

Het laatste wat er gepland stond voor deze iteratie was Security. Op het moment zit er geen security laag tussen de front en back-end. De gebruikers hoeven dus niet in te loggen om gebruik te maken van de applicatie. Ik heb gekozen om spring security te implementeren. Spring security is een securityservice voor J2EE Enterprise applicaties. Ik heb dit gekozen omdat het goed geïntegreerd kan worden met mijn bestaande spring back-end. Daarnaast is het goed aanpasbaar en werkt het out of the box.

Voor de implementatie zou ik eerst een workshop volgen over Spring security. Deze workshop werd binnen CGI gegeven en was op de ene laatste dag van iteratie 4. Mijn plan was om de workshop te volgen en vervolgens Spring security te implementeren. Echter was de workshop niet goed en ik had er bijna niets geleerd. Ik koos ervoor om security door te schuiven naar de volgende iteratie omdat ik geen halve implementatie wou schrijven.

5.6.5 Reflectie

Deze iteratie heb ik weinig keuzes gemaakt en is mijn reflectie erg klein vergeleken met de rest. Een deel is ook opgenomen in de hoofdstukken 7 en 8.

Ik ging ervan uit dat ik veel zou leren tijdens de workshop wat niet het geval was en dit was een valkuil. Ik heb uiteindelijk security niet kunnen implementeren. Wat ik had moeten doen was alvast security implementeren en dan vervolgens vragen stellen tijdens de workshop over problemen die ik tegenkwam.

6 Transition fase

De transition fase bestaat uit de laatste twee weken. In de planning staat dit beschreven als uitloop. Voor deze fase was er een kort gesprek met de opdrachtgever. Tijdens het gesprek heb ik gemeld dat ik vrijwel alleen mijn verslag ga werken en de overige openstaande taken ga afmaken. De opdrachtgever was hier mee eens.

Het gaat hier om de volgende taken:

- Javadoc schrijven
- Testen

Er was geen tijd meer om te deployen. Dus kon ik ook niet meer verder werken aan het product. Ik heb deze iteratie vrij weinig te vertellen met betrekking tot ontwikkelen.

7 Reflectie product

Ik heb gemixte gevoelens en meningen over het product. Enerzijds ben ik blij met het product omdat ik zeker weet dat ik alle opties en keuzes bewust heb gemaakt. Dit heeft in mijn ogen een goed product opgeleverd. Het product voldoet aan de eisen van CGI maar schiet tekort op het gebied van testen en functionaliteit. Met testen bedoel ik dat ik uiteindelijk minder heb getest dan ik had gehoopt. Dit kon veel meer maar ik had bewust de prioriteit gelegd bij documentatie.

Qua functionaliteiten schiet het tekort omdat ik uiteindelijk de applicatie niet heb kunnen deployen. Origineel had ik gepland om de app te deployen en de functionaliteiten te verfijnen/verder uitwerken. Dit is helaas niet gelukt. Wat gewoon ontzettend jammer is. Ik weet dus niet of de functionaliteiten goed zijn. Natuurlijk heb ik de functionaliteiten naar mijn beste kunnen geïmplementeerd zoals ik ze heb beschreven in de user stories, maar hierbij waren de user stories juist abstract. Echter zou ik wel willen melden dat de opdrachtgever mij al vrij vroeg had verteld dat niet alles mogelijk zal zijn. Ik had het dus zien aankomen maar ik heb daar helaas niks aan kunnen doen.

Qua inhoudelijk kwaliteit zit ik volgens mij goed. De code is doorgenomen door een architect. Daarnaast heb ik Sonar en PMD gebruikt om de code kwaliteit te bewaken. Dit is in mindere mate geldig voor de Android app, maar ook daar heb ik vertrouwen dat de code kwaliteit voldoende tot goed is. Zover ik weet is de opdrachtgever ook blij met het product. Verder ben ik blij dat ik gebruik heb gemaakt van libraries. Tot een jaar geleden gebruikte ik vrij weinig libraries. Natuurlijk heb ik hierbij wel gewoon gekeken naar de libraries die het beste passen bij dit project en/of ze nuttig zijn.

Conclusie

Het product heb ik zo goed als ik dat kan ontwikkeld. Hierbij heb ik vrijwel alle user stories geïmplementeerd, maar het product mist verfijning. Het is nu abstract waardoor het lijkt dat het weinig kan, terwijl je met relatief weinig moeite de applicatie zou kunnen uitbreiden.

Als ik het project opnieuw zou uitvoeren zou ik ook de nadruk leggen op IOS. Ik kwam er namelijk achter dat er alleen een interne app store is voor IOS apps. Daarnaast zou het makkelijker zijn om CGI te overtuigen om een server te huren voor een IOS project. De organisatie had wat dat betreft veel invloed. Daarnaast zou ik nog steeds kiezen voor een app en nog steeds dezelfde technieken gebruiken. Natuurlijk kun je dit project zien als een proof of concept en het verder uitwerken van deze app zou op zich geen probleem zijn. Als het werkt dan kan het er een IOS-versie gemaakt worden alleen vergt dat wat overtuiging.

8 Reflectie proces

Ik was over het algemeen tevreden over het proces, maar ook hier zaten er een aantal leerpunten. Mijn reflectie over het proces kan beïnvloed zijn doordat ik CGICraft nooit heb kunnen deployen. Per proces bespreek ik wat er beter had gekund.

De requirements heb ik opgenomen als user stories deze zijn abstract en zijn verder niet uitgebreid/ verder gespecificeerd. Voor een volgende keer zou ik veel meer nadruk leggen op het specificeren van de requirements. Dit zou ik doen door de inception fase voor dit project verder uit te rekken. Drie weken was naar mijn mening niet genoeg om een goed beeld te schetsen. Daarnaast zou ik per iteratie meer focussen op het verder uitwerken van de requirements hierbij zou ik de opdrachtgever beter bij kunnen betrekken. Een valkuil zou kunnen zijn dat ik hier dan weer te veel nadruk op leg, terwijl ik niet weet of de user story ook gaat werken.

Dat brengt me bij de tweede punt. Gamificatie. Ik geloof dat dit project beter geschikt was als ik het als een onderzoek had benaderd. Ik heb tijdens dit project wel vragen gesteld maar deze niet vastgelegd. Bij een onderzoek zou ik kritischer hebben gekeken naar dit project, gamificatie en de nut hiervan. Hierbij zou ik een prototype of een Proof of Concept schrijven waarmee ik mijn theorie ondersteun/test. Mijn gedachtegang zou dan ook anders hebben gewerkt. Nu dacht ik voornamelijk aan het bouwen van een applicatie. Dit komt mede doordat ik onder de indruk was dat ik een bestaand systeem zou aanpassen. Daarentegen zou ik bij een onderzoek meer de nadruk leggen op het uitwerken van de game elementen en deze zo snel mogelijk te testen door het te deployen bij de gebruikers. Met een juiste balans tussen functionaliteiten toevoegen en verfijnen had ik een product kunnen ontwikkelen waar CGI ook daadwerkelijk iets aan heeft. Dat wil niet zeggen dat dit project nutteloos is. Verre van zelfs. Het mist alleen deployment. In principe is het zoals hierboven beschreven uitgevoerd alleen niet volgens een officiële onderzoekmethode.

Ik heb vrijwel elke iteratie ontworpen. Ik heb zelfs onderdelen ontworpen die niet terug te zien zijn in het verslag omdat ik ze uiteindelijk niet heb geïmplementeerd. Hierbij heb ik mij zo goed mogelijk aan de UML-regels gehouden. Verder had ik wat meer ontwerpen kunnen maken, voor mij zou het geen invloed hebben maar het had wel een toevoeging kunnen zijn voor de developers die na mij aan CGICraft zullen werken. Zo heb ik nu geen entity diagram voor de database omdat het klassendiagram dit al redelijk goed representeert. Hier ben ik mij van op de hoogte maar dat hoeft niet het geval te zijn voor de toekomstige developers.

Het ontwikkelen van de applicatie ging ook goed. Hierbij heb ik best practices van CGI zo goed mogelijk gevolgd. Daarnaast was de kwaliteit ook goed. Ik had wat minder kunnen refactoren maar dat is iets waar niet omheen kon.

Verder ben ik wat minder blij over testen. Ik heb alles getest door de functionaliteiten door te lopen alleen mis ik nog aantal unit testen. Ik had ook acceptatie testen willen toevoegen maar dat is niet gelukt omdat de applicatie nooit is gedeployed. Dit is echter niet goed genoeg en ik had het nog steeds acceptatietesten kunnen uitvoeren maar dan met een persoon. Dit heb ik ook gedaan maar niet op een officiële manier. Zo heb ik gedurende het project de app regelmatig laten spelen door een aantal collega's achteraf gezien kon ik dit ook officieel vastleggen. Een gemiste kans.

Tot slot mis ik een Definition of Done. Hier werd gedurende het project impliciet aan gehouden. Wat ik daarmee wil zeggen dat ik tijdens elke iteratie op een vaste wijze na ging of de iteratie een succes was en of ik besproken onderdelen wel goed heb geïmplementeerd. Dit had ik achteraf gezien beter expliciet kunnen beschrijven in het PVA. Zodat ik aan het einde van elke iteratie een duidelijk checklist had van wat ik allemaal minimaal had moeten uitvoeren. Daarnaast is het ook een vorm van kwaliteitsbewaking.

Conclusie

Over het algemeen is het project goed uitgevoerd alleen ben ik er niet achter gekomen of dit project nou nuttig was/ werkt. Nuttig omdat CGI met een omslag bezig is, er geen directe koppeling is met het PDP en of het überhaupt gebruikt zal worden. Werkt omdat ik niet weet of de gekozen game elementen inderdaad doen waar ze voor zijn bedoeld.

Verder zijn er een aantal onderdelen impliciet uitgevoerd. Het zou beter zijn geweest als ik ze had vastgelegd. Daarnaast kan mijn requirements fase wat beter door nog kritischer te kijken naar wat de opdrachtgever wil. Hoewel dat voor dit project redelijk lastig was.

Voor een vervolgopdracht zou ik de applicatie verder uitwerken. Hierbij raad ik aan om als eerste de huidige app te deployen en te meten hoe de app wordt ervaren. Aan de hand van de feedback zou ik de app verder uitwerken. Er kan nog genoeg toegevoegd worden. Deployen is cruciaal omdat je anders werkt aan een product waarvan je niet weet of het gaat werken. Een alternatief zou zijn om opnieuw te beginnen echter raad ik dit niet aan. Ik denk namelijk dat je meer kunt leren als je dit project eerst goed afrond.

9 Competenties

1.4 Uitvoeren analyse door definitie van requirements

Gedurende het project heb ik op verschillende wijze de requirements ontsloten. Ik heb de opdrachtgever geïnterviewd, workshops gevolgd/ gevraagd tijdens workshop wat de gebruikers willen zien in de applicatie. Ik heb mij verdiept op het gebied van gamificatie om zo de opdrachtgever te begeleiden met dit project. Ik heb gesproken met een gamificatie expert en zelfs een interview gehad met een twee PHD studenten die onderzoek doen over gamificatie. Dit allemaal heeft nog steeds geleidt tot abstracte requirements (user stories) deze zouden uitgewerkt worden aan de hand van de gebruikers feedback. Alleen is dat niet gebeurt.

Naast gamificatie heb ik ook onderzoek/verdiept in de architectuur van CGI hierbij heb ik gekeken naar de bestaande systemen en hoe ik hiermee zou kunnen communiceren. Daarnaast heb ik gekeken naar de data dat deze systemen bevatten en hoe dit gecombineerd/ bruikbaar kan zijn voor dit project. Dit heb ik gedaan door de informatie op het intranet door te nemen en door een architect te interviewen.

3.2 Ontwerpen systeemdeel

Voorafgaande elke bouw heb ik met behulp van UML het te bouwen stuk ontworpen. Hierbij heb ik geprobeerd om de juiste technieken te gebruiken. Elke iteratie waar dat nodig was heb ik een UML diagram gemaakt. Het gaat hierbij om klassendiagram en sequence diagrammen. Naast UML heb ik ook de design guidelines van Android gevolgd, hierbij heb ik schetsen gemaakt van de app en deze uitgewerkt in een mock-up. De mock-up heb ik getoond aan de opdrachtgever. Voor de database heb ik guidelines gevolgd die beschreven zijn door Microsoft. Dit heb ik gedaan omdat CGI hier geen eisen bij had.

3.3 Bouwen applicatie

Ik heb twee systemen gemaakt die samen een systeem moeten vormen. De back-end is geschreven in Spring en heeft een MySQL database. De front-end is een Androidapplicatie die ondersteunt wordt doormiddel van verschillende libraries. De front- en back-end communiceren met elkaar doormiddel van REST en WebSocket. Gedurende het project heb ik ook de kwaliteit bewaakt door mij zo goed mogelijk aan de richtlijnen van CGI te houden. Tijdens de bouw heb ik alle functionaliteiten die zijn beschreven tijdens de requirements en analyse fase zo goed mogelijk gerealiseerd.

3.5 Uitvoeren van en rapporteren over het testproces

Voor testen heb ik elke iteratie functionele testen uitgevoerd. Hierbij heb ik gekeken of de functionaliteit doet wat het moet doen. Naast dat heb ik ook unit testen gemaakt. Deze testen heb ik ook volgens de DCJava richtlijnen uitgevoerd. Ter verdere kwaliteit controle heeft er ook een code review plaats gevonden en heb ik de testen laten doornemen door een college. Daarnaast heb ik gebruik gemaakt van Sonar en PMD om de kwaliteit te bewaken.

Literatuurlijst

[1] Gaudiosi, J. (2015, January 15). Mobile game revenues set to overtake console games in 2015. Retrieved July 10, 2015.

[2] Definition of gamification in English:. (n.d.). Retrieved September 11, 2015.

[3] Lines, M., & Ambler, S. (2012). Disciplined Agile Delivery a practitioner's guide to agile software delivery in the enterprise (p. 543). S.l.: IBM Press.

[4] Full Agile Delivery Lifecycles. (2014, May 6). Retrieved June 11, 2015, from http://www.disciplinedagiledelivery.com/lifecycle/

[5] A Brief History of Gamification Infographic - e-Learning Infographics. (2014, August 18). Retrieved June 11, 2015, from http://elearninginfographics.com/brief-history-of-gamification-infographic/

[6] Google Trends - Web Search interest - Worldwide, 2004 - present. (2015, June 11). Retrieved June 11, 2015.

[7] Chou, Y. (2014, September 3). Gamification Design: 4 Phases of a Player's Journey. Retrieved September 11, 2015, from http://www.yukaichou.com/gamification-examples/experience-phases-game/#.VfK0FxHtlBd

[8] Muntean, C. I. (2011, October). Raising engagement in e-learning through gamification. In Proc. 6th International Conference on Virtual Learning ICVL (pp. 323-329).

[9] Global Games Market Will Grow 9.4% to $91.5Bn in 2015 | Newzoo. (2015, April 22). Retrieved June 11, 2015, from http://www.newzoo.com/insights/global-games-market-will-grow-9-4-to-91-5bn-in-2015/

[10] Dreyfuss, J. (2015, May 28). The Ultimate JSON Library: JSON.simple vs GSON vs Jackson vs JSONP | Takipi Blog. Retrieved July 11, 2015.

[11] Saloranta, T. (2011, March 10). Upgrade from org.json to Jackson, piece by piece, using jackson-module-org-json. Retrieved September 11, 2015, from http://www.cowtowncoder.com/blog/archives/2011/03/entry_447.html

[12] Android Async HTTP Clients: Volley vs Retrofit -. (2013, December 9). Retrieved September 11, 2015.

[13] Most played PC games 2015 | Statistic. (n.d.). Retrieved August 11, 2015.

[14] Rutar, N., Almazan, C. B., & Foster, J. S. (2004, November). A comparison of bug finding tools for Java. In Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on (pp. 245-256). IEEE.

Experience the commitment®

Bijlagen

Bijlage A – Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2015-1.2 (start uiterlijk 11 mei 2015)
Startdatum uitvoering afstudeeropdracht: 11-05-2015
Inleverdatum afstudeerdossier volgens jaarrooster: 5 oktober 2015

Studentnummer: 10081224
Achternaam: dhr. Gupta
Voorletters: S.K
Roepnaam: Sangam
Adres: Azuurblauw 28
Postcode: 2718JG
Woonplaats: Zoetermeer
Telefoonnummer: 079-3613054
Mobiel nummer: 06-11214837
Privé emailadres: Sangamgupta10@gmail.com

Opleiding: Informatica
Locatie: Zoetermeer
Variant: voltijd

Naam studieloopbaanbegeleider: Tim Cocx
Naam begeleidend examinator: Nanny Jacobs
Naam tweede examinator: Arno Nederend

Naam bedrijf: CGI
Afdeling bedrijf:
Bezoekadres bedrijf: George Hintzenweg 89
Postcode bezoekadres: 3068 AX
Postbusnummer:
Postcode postbusnummer:
Plaats: Rotterdam
Telefoon bedrijf: +31 (0)88 564 0000
Telefax bedrijf:
Internetsite bedrijf: http://www.cginederland.nl/

Achternaam opdrachtgever: dhr. Zijsling
Voorletters opdrachtgever: T
Titulatuur opdrachtgever:
Functie opdrachtgever: Project Manager
Doorkiesnummer opdrachtgever: +31 6 51 38 94 21
Email opdrachtgever: tjeerd.zijsling@cgi.com

Achternaam bedrijfsmentor: dhr. Zijsling
Voorletters bedrijfsmentor: T
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Project Manager
Doorkiesnummer bedrijfsmentor: +31 6 51 38 94 21
Email bedrijfsmentor: tjeerd.zijsling@cgi.com
NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder: 06-11214837
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Een nieuwe versie van PDP met game elementen ontwikkelen bij CGI.

Opdrachtomschrijving

1. Bedrijf

CGI groep is een multinationaal bedrijf met vestigingen in 40 landen verspreid over America, Europa en Azië. CGI omvat zo'n 68.000 professionals. CGI is toegewijd met het helpen van zijn klanten met hun zakelijke doelstellingen; om de professionals binnen CGI te belonen met een carrière; en de aandeelhouders superieure rendementen over tijd aan te bieden. CGI is een bedrijf dat resultaat levert.

Binnen CGI werk ik als software engineer in het innovatiecentrum van het Delivery Center Java. In het innovatiecentrum worden projecten uitgevoerd waarbij nieuwe ideeën en technieken vertaald worden naar nieuwe producten en diensten, betere processen of zelfs totaal nieuwe markten.

2. Probleemstelling

Voor CGI is het erg belangrijk dat haar medewerkers over voldoende relevante en actuele kennis beschikken. Dit betekent dat een medewerker continue moet blijven leren en zich verbreden om inzetbaar te blijven bij haar klanten. Binnen CGI wordt een PDP gebruikt om een medewerker te stimuleren nieuwe skills te leren en zijn huidige skills te verbeteren. Dit PDP vormt ook de basis voor de beoordeling of een medewerker zich voldoende heeft ingezet om zijn kennis op pijl te houden.

In de praktijk wordt het personal development plan onvoldoende voor dit doel gebruikt. Het systeem waarin dit PDP is vastgelegd is weinig gebruikersvriendelijk, en stimuleert niet. De gewenste kennisontwikkeling bij medewerkers blijft hierdoor achter. Omdat CGI op centraal niveau geen inzicht heeft in hoe medewerkers van plan zijn zich te ontwikkelen, is het lastig om hier trainingsprogramma's voor op te zetten. Ook ontbreekt een duidelijk en toegankelijk overzicht van de skillsets van medewerkers, wat het delen van kennis tussen mensen die zich in verschillende onderdelen van de organisatie bevinden lastig maakt.

Het is belangrijk voor CGI als bedrijf om het gebruik van het PDP aantrekkelijker te maken en er voor te zorgen dat dit PDP op vrijwillige basis gebruikt gaat worden. Om dit te stimuleren wil CGI gamificatie gebruiken (het gebruiken van game elementen om user engagement te verhogen). Door het PDP te ondersteunen met een gegamificeerde versie verwacht CGI dat medewerkers uit zichzelf meer skills en tools gaan leren waardoor ze als persoon waardevoller worden (voor zichzelf en voor het bedrijf). Een gegamificeert PDP zou moeten zorgen voor toegenomen betrokkenheid en actuelere kennis, waardoor de CGI medewerkers breder inzetbaar zijn en aanwezige kennis beter kan worden ontsloten.

3. Doelstelling van de afstudeeropdracht

De opdracht is om het idee dat voor de gamificatie van kennisopbouw/deling is ontwikkeld verder uit te werken en hier een eerste versie van te implementeren. Dit omvat het ontwerpen van de hiervoor in te zetten koppelingen met gerelateerde systemen, targetplatform(s) en bepalen van de voor de ontwikkeling te gebruiken tools. Het gaat om een innoverende opdracht, die ook de kracht van gamificatie aan de rest van CGI kan laten zien.

Na het afronden van de opdracht is het de bedoeling dat CGI de gamificatie inzet bij een testgroep en daarmee duidelijke metrics te verzamelen over het toegenomen gebruik van het PDP.

4. Resultaat

Na het afronden van de opdracht heeft CGI een gegamificeert systeem voor kennisontwikkeling en -deling die al enkele iteraties heeft ondergaan. Het systeem is aangesloten op de relevante servers en databases van CGI en data wordt real time heen en weer gestuurd. Met andere woorden, de gamificatie is “playable”. Informatie van medewerkers kan snel geladen worden en de gamificatie systemen werken goed. Het systeem wordt door een testgroep gespeeld en er is (hopelijk) een toenemende trend te zien bij het gebruik van PDP (het liefst meer dan 10%).

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

Ik ga werken met SCRUM omdat dit de standaard softwareontwikkelmethode is binnen CGI.

Mijn afstudeerstage heeft een looptijd van 17 weken. Hiervan trek ik drie weken voor sprint nul gevolgd door zes sprints van twee weken. Ik houd dan twee weken over voor het schrijven van het afstudeerverslag, oplevering van producten en eventuele uitloop.

Binnen sprint nul wordt het plan van aanpak geschreven waarbij de nadruk wordt gelegd op een globale planning zodat de opdrachtgever en ik op een lijn zitten. Ook wordt er een product backlog opgesteld en worden de requirements vastgesteld. Deze requirements krijg ik doormiddel van interviews met de stakeholders. Vervolgens ga ik prioriteiten vaststellen aan de requirements met behulp van MoSCoW.

Verder wordt er georiënteerd wat voor type applicatie (mobiel, webapp, desktop) er ontwikkeld moet worden en welke tools ik nodig heb om dit te realiseren. Daarnaast wordt het term gamificatie beter uitgewerkt. Tevens zal ik gedurende deze periode kennis maken met het bedrijf.

Sprint nul wordt opgevolgd door sprint 1 t/m 6. Ieder sprint ga ik in principe analyseren, ontwerpen, bouwen en testen.

In sprint één en twee ontwikkel ik een basis applicatie die connectie legt met de benodigde API’s. De sprints die hierop volgen zullen nieuwe features toevoegen. Verder verwacht ik dat er in een latere sprint gedeployed gaat worden naar een testomgeving voor eventuele acceptatie testen.

De tools en technieken die ga gebruiken bestaan uit de basis tools van CGI en de gekozen tools die afhankelijk zijn van de type applicatie die er ontwikkeld gaat worden. Wel gaat de neiging uit naar een JAVA gebaseerd applicatie. De volgende lijst geeft een overzicht van tools die gebruikt zullen/kunnen worden.

- JAVA
- JUnit
- Wicket
- Subversion
- UML tool
- Eclips

6. Op te leveren (tussen)producten

De producten die ik ga opleveren zijn:

- PVA

- Requirements document
- Analyse & Design documenten/diagrammen zoals klassendiagram of architectuur document
- Het gerealiseerde systeem en source code met commentaar.
- Testen waaronder een testontwerp, test implementatie en een testrapport.
- Eventueel Scrum artifacts zoals product of sprint backlog

7. Te demonstreren competenties en wijze waarop

1.4 Uitvoeren analyse door definitie van requirements
Het opstellen van requirements door stakeholders te interviewen en daarna prioriteiten vaststellen.

3.2 Ontwerpen systeemdeel
PDP ontwerpen voorafgaand aan de bouw met behulp van UML.

3.3 Bouwen applicatie
PDP ontwikkelen in JAVA.

3.5 Uitvoeren van en rapporteren over het testproces
Ik ga JUnit tests uitvoeren en mijn bevindingen beschrijven in een testrapportage.

Bijlage B – Plan van Aanpak

Plan van Aanpak

Gameficatie PDP

Naam:	**Sangam K. Gupta**
Studentnummer:	**10081224**
School:	**De Haagse Hogeschool**
Datum:	**29-05-2015**
Versie:	**1.0**

Document historie

Versie	Omschrijving	Datum
0.1	Eerste opzet PVA	12-05-2015
0.2	Verbeteringen doorgevoerd en nieuwe hoofdstukken visie en scope toegevoegd.	20-05-2015
0.3	User stories verbeterd, constraints toegevoegd.	21-05-2015
0.4	Verbetering spellingsfouten, uitbreiding planning & aanpak en toevoeging technische strategie	28-05-2015
1.0	Verbetering spellingsfouten/ structuur	29-05-2015

Inhoudsopgave

Experience the commitment®

1 Inleiding

Games worden steeds populairder en dat zie je veel terug in het dagelijks leven. Zo bestaat er een taalapp die je spelenderwijs een nieuwe taal leert [1] of een fitness tracker die je het gevoel geeft dat je achtervolgd wordt door honderden zombies [2]. Het toevoegen van game elementen aan alledaagse taken om de gebruikers betrokkenheid te bevorderen wordt steeds populairder [3].

CGI wil hier gebruik van maken. Eén van de hoofdoelen van CGI is dat haar medewerkers voldoende gemotiveerd zijn om zichzelf te ontwikkelen zowel in hun vakgebied als in persoonlijke vaardigheden. Dit wordt gedaan met behulp van het personal development plan (PDP). Het huidige PDP systeem is niet erg gebruiksvriendelijk en wordt nauwelijks gebruikt. Door middel van gamificatie wil CGI het gebruik van het PDP stimuleren en kennisdeling tussen medewerkers bevorderen.

Dit document is bedoeld om het project en de bijbehorende werkzaamheden te verhelderen. In hoofdstuk twee wordt de opdracht toegelicht gevolgd door de visie in hoofdstuk drie. Vervolgens komt in hoofdstuk vier de scope van het project aan bod, hierbij zijn de globale requirements gespecifieerd. In hoofdstuk vijf wordt kort de architectuur besproken. Daaropvolgend komen de hoofdstukken zes en zeven betreffende de projectorganisatie en de risico's die komen kijken bij dit project. Als laatste wordt de gekozen aanpak en de globale planning besproken in hoofdstuk acht.

2 Opdrachtomschrijving

2.1 Bedrijf

CGI groep is een multinationaal bedrijf met vestigingen in 40 landen verspreid over Noord-, Zuid-Amerika, Europa en Azië. CGI omvat zo'n 68.000 professionals. CGI is toegewijd met het helpen van zijn klanten met hun zakelijke doelstellingen; om de professionals binnen CGI te belonen met een carrière; en de aandeelhouders superieure rendementen over tijd aan te bieden. CGI is een bedrijf dat resultaat levert.

Een onderdeel dat hoog op de agenda staat binnen CGI is innovatie. Zo worden er in het innovatiecentrum binnen het Delivery Centrum Java niet alleen Java engineers opgeleid maar wordt er ook gepushte om nieuwe ideeën en technieken uit te werken en deze te vertalen naar nieuwe producten en diensten, betere processen of zelfs totaal nieuwe markten.

Binnen het innovatiecentrum hangt een informele sfeer en voor vragen of problemen kan ik terecht bij mijn collega's.

2.2 Probleemstelling

Voor CGI is het erg belangrijk dat haar medewerkers over voldoende relevante en actuele kennis beschikken. Dit betekent dat een medewerker continue moet blijven leren en zich verbreden om inzetbaar te blijven bij haar klanten. Binnen CGI wordt een persoonlijk ontwikkelplan (PDP) gebruikt om een medewerker te stimuleren nieuwe skills te leren en zijn huidige skills te verbeteren. Dit PDP vormt ook de basis voor de beoordeling of een medewerker zich voldoende heeft ingezet om zijn kennis op pijl te houden.

In de praktijk wordt het persoonlijk ontwikkelplan onvoldoende voor dit doel gebruikt. Het systeem waarin dit PDP is vastgelegd is weinig gebruikersvriendelijk, en stimuleert niet. De gewenste kennisontwikkeling bij medewerkers blijft hierdoor achter. Omdat CGI op centraal niveau geen inzicht heeft in hoe medewerkers van plan zijn zich te ontwikkelen, is het lastig om hier trainingsprogramma's voor op te zetten. Ook ontbreekt een duidelijk en toegankelijk overzicht van de skillsets van medewerkers, wat het delen van kennis tussen mensen die zich in verschillende onderdelen van de organisatie bevinden lastig maakt.

Het is belangrijk voor CGI als bedrijf om het gebruik van het PDP aantrekkelijker te maken en er voor te zorgen dat dit PDP op vrijwillige basis gebruikt gaat worden. Om dit te stimuleren wil CGI gamificatie gebruiken (het gebruiken van game elementen om user engagement te verhogen). Door het PDP te ondersteunen met een gegamificeerde versie verwacht CGI dat medewerkers uit zichzelf meer skills en tools gaan leren waardoor ze als persoon waardevoller worden (voor zichzelf en voor het bedrijf). Een gegamificeert PDP zou moeten zorgen voor toegenomen betrokkenheid en actuelere kennis, waardoor de CGI medewerkers breder inzetbaar zijn en aanwezige kennis beter kan worden ontsloten. De gegamificeerde versie van het PDP krijgt hierna de ontwikkelnaam CGICraft, zodat er een duidelijk onderscheid is tussen het huidige PDP systeem en de nog te ontwikkelen PDP game (CGICraft).

2.3 Doelstelling

De opdracht is om de mock-up die voor de gamificatie van kennisopbouw/deling is ontwikkeld verder uit te werken en hier een eerste versie van te implementeren. Dit omvat het ontwerpen van de hiervoor in te zetten koppelingen met gerelateerde systemen, targetplatform(s) en bepalen van de voor de ontwikkeling te gebruiken tools. Het gaat om een innoverende opdracht, die ook de kracht van gamificatie aan de rest van CGI kan laten zien.

Na het afronden van de opdracht is het de bedoeling dat CGI de gamificatie inzet bij een testgroep om daarmee duidelijke metrics te verzamelen over het toegenomen gebruik van het PDP.

2.4 Resultaat

Na het afronden van de opdracht heeft CGI een gegamificeert systeem voor kennisontwikkeling en deling die al enkele iteraties heeft ondergaan. Het systeem is aangesloten op de relevante servers en databases van CGI en data wordt real time heen en weer gestuurd. Met andere woorden, de gamificatie is “playable”. Informatie van medewerkers kan snel geladen worden en de gamificatie systemen werken goed. Het systeem wordt door een testgroep gespeeld en er is (hopelijk) een toenemende trend te zien bij het gebruik van PDP (het liefst meer dan 10%).

3 Visie

Zoals besproken in hoofdstuk 2.3 is de opdracht om een gegamificeerde versie van het PDP te maken. CGICraft moet een aantal doelstellingen verwezenlijken, namelijk:

- Stimuleren van kennisdeling.
- Stimuleren van kennisontwikkeling.
- Verhogen gebruik van het PDP.
- CGICraft plaatsen binnen de huidige architectuur van CGI.

Stimuleren van kennisdeling en kennisontwikkeling gaan hand in hand. De core van CGICraft wordt een community driven quiz ofwel battle genoemd. Met community driven quiz wordt bedoeld dat de vragen en antwoorden bedacht en gemaakt worden door de gebruikers van CGICraft. Deze kunnen vervolgens beoordeeld en besproken worden tijdens de quiz. Dit zorgt ervoor dat de kennisdeling/ontwikkeling wordt gestimuleerd, de gebruikers willen namelijk een goede vraag bedenken en doen dan hiervoor de nodige research. Indien de vraag of antwoord niet correct of onduidelijk is, dan kan hierover worden gediscussieerd, wat weer leidt tot kennisdeling. Naast de core zijn er ook alternatieve manieren die gebruikers motiveren om kennis te ontwikkelen/delen. De idee is dat de game elementen bedoeld zijn ter ondersteuning van deze twee doeleinden en zo hopend door de gamificatie ook het gebruik van het PDP te verhogen door het ‘leuker’ te maken. De volledige lijst met initiële features wordt besproken in het volgende hoofdstuk.

Met behulp van gamificatie is het ook de bedoeling om het gebruik van het PDP zelf te verhogen. Er is een aantal zaken waar rekening mee moet worden gehouden wat betreft gamificatie. Het simpelweg toevoegen van game elementen aan de nu al weinig gebruiksvriendelijke applicatie zal het gebruik niet bevorderen. Gamificatie is meer dan alleen game elementen toevoegen. Er komt veel sociale psychologie bij kijken [5]. Dit valt buiten de context van dit document en hier zal niet verder op in worden gegaan. Hoe het PDP gegamificeerd gaat worden met de achterliggende gedachte, hoe er voor wordt gezorgd dat de gebruikers blijven terugkomen, etc. worden in meer detail besproken in het gamedesigndocument.

Als laatste is het de bedoeling dat CGICraft goed aansluit bij CGI. Het gaat hier om het design en de architectuur. Hierbij moet niet alleen goed gekeken worden naar de huidige architectuur en of er überhaupt de mogelijkheid is om data te ontsluiten uit bestaande systemen of databases, maar ook hoe de applicatie eruit komt te zien. Denk hierbij bijvoorbeeld aan lettertypes en kleuren.

4 Scope opdracht

De scope voor dit project is bepaald door te kijken naar de doelstellingen van de opdrachtgever en hoe deze ondersteund kunnen worden met een game. Elke feature moet dan ook één of meerdere doelstellingen ondersteunen. De features zijn in kaart gebracht middels brainstorm sessies met de opdrachtgever. Ook zijn de features grotendeels uitbreidingen op de mock-up die dit project in gang heeft gezet. Deze zijn opgenomen in tabel één in de vorm van user stories (work items). De user stories uit tabel één worden in de komende iteraties verder uitgewerkt.

Tevens heeft elke user story een prioriteit, zodat er bepaald kan worden welke functionaliteiten cruciaal zijn voor een succesvol product. De prioriteit is bepaald aan de hand van de MoSCoW-methode. De verschillende niveaus zijn toegekend aan de user stories aan de hand van de kennis die is opgedaan uit de meetings met de opdrachtgever. MoSCoW kent vier niveaus:

- **M**ust: deze user stories moeten terugkomen in het eindresultaat en zijn dus cruciaal voor het succesvol afronden van het project.
- **S**hould: deze user stories zijn gewenst maar het product kan er ook zonder.
- **C**ould: deze user stories zullen aan bod komen als er genoeg tijd is.
- **W**ould: deze user stories zullen in dit project niet aan bod komen, maar kunnen interessant zijn bij een vervolg project.

ID	Doelstelling	Omschrijving	Prioriteit
DS1	**Stimuleren kennisdeling**		
US1.1		Als een gebruiker kan ik nieuwe vragen en antwoorden toevoegen aan het systeem.	M
US1.2		Als een gebruiker kan ik mijn profiel aanpassen en delen, zodat ik mijn voortgang kan laten zien aan mijn collega's.	M
US1.3		Als een gebruiker kan ik zoeken naar een team, waar ik lid van wil worden.	S
US1.4		Als een gebruiker kan ik lid worden van een team, om teambattles te spelen.	S
DS2	**Stimuleren kennisontwikkeling**		
US2.1		Als een gebruiker kan ik een nieuwe (quiz)battle starten, zodat ik kan stijgen/dalen in ranking.	M
US2.2		Als een gebruiker kan ik zien hoeveel spelers een reactie hebben geplaatst bij een vraag, zodat ik een beeld krijg bij de populariteit van de vraag.	M
US2.3		Als een gebruiker kan ik een reactie plaatsen of reageren op een bestaande reactie, zodat ik de vraag kan bespreken.	M
US2.4		Als een gebruiker kan ik de vraag beoordelen met een 'like', zodat de vraag vaker terugkomt in battles voor andere spelers.	M
US2.5		Als een gebruiker kan ik uit een overzicht van goals kiezen, zodat ik mijn eigen goals kan bepalen.	M
DS3	**Verhogen gebruik PDP**		
US3.1		Als een gebruiker kan ik mijn ranking zien op de leaderboard, zodat ik kan zien waar ik sta tegenover de rest.	M
US3.2		Als een gebruiker kan ik de globale ranking zien.	C

ID	Doelstelling	Omschrijving	Prioriteit
US3.3		Als een gebruiker kan ik badges verdienen door battles succesvol af te ronden, zodat ik mijn badges kan toevoegen aan mijn avatar.	M
US3.4		Als een gebruiker wil ik mijn avatar level verhogen door middel van battles, zodat ik nieuwe badges en nieuwe goals kan unlocken.	M
US3.5		Als een gebruiker kan ik mijn individuele skills levelen door middel van battles, zodat ik mijn speler level kan verhogen.	M
US3.6		Als teamleader kan ik de profielen van mijn members (spelers) zien zodat ik met deze gegevens rekening kan houden tijdens een evaluatiegesprek met de member.	C
US3.7		Als een teamleader kan ik een overzicht zien van al mijn members.	C
US3.8		Als een team kunnen we een nieuwe battle starten, zodat we gezamenlijk in rank kunnen stijgen of dalen.	S
US3.9		Als gebruiker kan ik stijgen en dalen op de leaderboard.	M
US3.10		Als gebruiker kan ik inloggen, zodat ik het spel kan spelen.	M

Tabel 1: initiële lijst van user stories

Naast de user stories is er ook een aantal niet functionele requirements geïdentificeerd. Deze zijn in tabel twee opgeschreven in de vorm van constraints.

ID	Constraint
CON1	CGICraft moet een speelse sfeer (look and feel) hebben.
CON2	CGICraft moet een lage leercurve hebben. De gebruikers moeten binnen een uur de applicatie kunnen gebruiken.
CON3	CGICraft moet overal gebruikt kunnen worden.
CON4	CGICraft moet veilig zijn, volgens de regels van CGI.
CON5	CGICraft moet aangesloten zijn op relevante systemen en databases.
CON6	CGICraft moet passen (qua stijl) bij CGI, zoals kleur en lettertype keuze.

Tabel 2: constraints voor dit project

Het uiteindelijke systeem is pas af (wordt gezien als succes) als het voldoet aan de acceptatie criteria in tabel drie. Dit wordt gedaan om misverstanden tussen het team en de opdrachtgever over het eindproduct te minimaliseren.

ID	Acceptatie Criteria
AC1	Alle user stories met een hoge prioriteit (M) moeten zijn geïmplementeerd.
AC2	Alle functionaliteiten moeten zijn getest (unit testen).
AC3	Het PDP heeft een verhoging in gebruik van liefst 10%.

Tabel 3: definition of done

Verder worden de volgende (tussen)producten verwacht:

- Plan van Aanpak, dit bevat tevens een lijst met initiële requirements (work items).
- Gamedesign document, dit wordt een levend document. Dit document bevat ook UML diagrammen en de requirements.
- Architectuur document. Ook het architectuur document bevat de nodige UML diagrammen.

- Te realiseren applicatie met source code, inclusief unittests.
- Testrapportage.

5 Technische strategie

Binnen dit hoofdstuk wordt beschreven hoe de architectuur eruit komt te zien als CGICraft gerealiseerd wordt.

Om vast te stellen met welke systemen CGICraft gaat communiceren, moet er eerst bepaald worden welke data er ontsloten moet worden en in welke systemen deze data dan te vinden is.

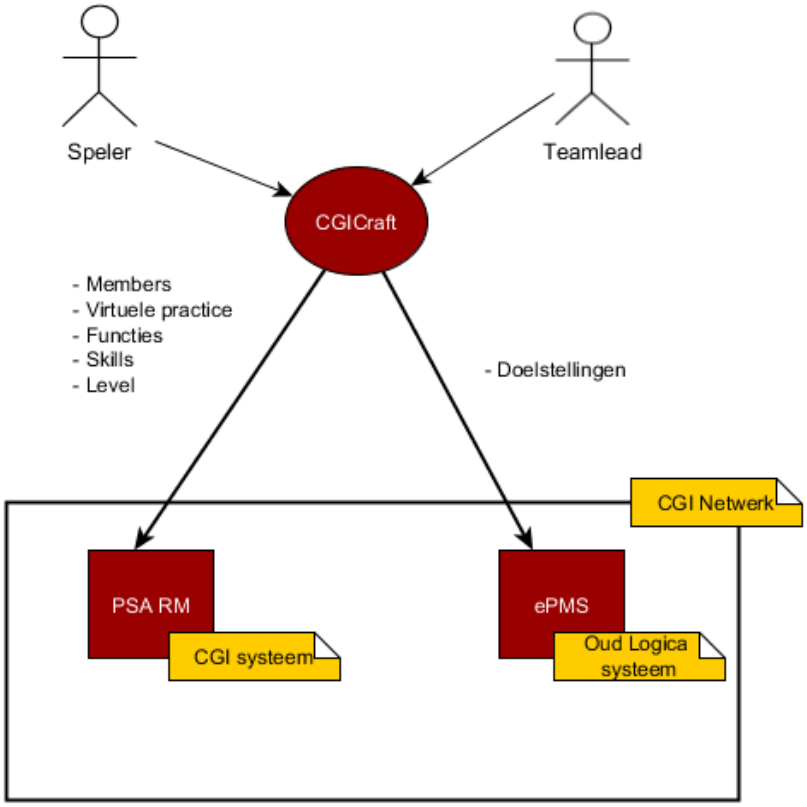
Aan de hand van de initiële user stories is bepaald dat het gaat om de volgende data:

- Members
- Virtuele Practices
- Functies
- (PMS/OMS) levels
- Hard en soft skills
- Doelstellingen

CGICraft draait om de medewerkers van CGI. De data die ontsloten wordt gaat dan ook voornamelijk over personeelsgegevens. Wat de data inhoud en de koppeling hiervan wordt in meer detail besproken in het game design document.

5.1 Context diagram

De bovenstaande gegevens zijn beschikbaar in de volgende twee systemen ePMS en PSA RM.



Figuur 1: context diagram

In het context diagram wordt het eerste beeld van het systeem geschapen. Hiermee wordt aangetoond hoe het systeem communiceert en de verwachte dataflow. PSA RM is een CGI systeem en ePMS is een oud Logica systeem. CGICraft haalt uit beide systemen personeelsgegevens op. De communicatie verloopt in één richting. Dat houdt in dat CGICraft alleen data ophaalt en niets opstuurt. De data wordt tijdens het ophalen ook verwerkt zodat het beter aansluit bij CGICraft.

Naast de dataflow valt op dat CGICraft buiten het CGI netwerk is geplaatst. Dit is gedaan vanwege concern drie “CON3: CGICraft moet overal gebruikt kunnen worden“. Dit is cruciaal voor gamificatie. Het spel moet vrijwillig gespeeld worden en de instapdrempel om het spel te spelen is lager als het overal beschikbaar is, in plaats van alleen tijdens werk. De meeste medewerkers hebben dan ook geen tijd om het spel te spelen. Als het spel niet overal beschikbaar is neemt de kracht van gamificatie ook af, wat niet de bedoeling is.

Dit levert een aantal problemen. Vanuit het security oogpunt kan de data niet benaderd worden buiten het netwerk en mag de data niet meegenomen worden naar buiten e.g. via mobiel. Er zijn hier een aantal architecturale oplossingen voor te bedenken.

ID	Omschrijving
1	Een standalone applicatie waarbij de twee interne systemen gemocked worden zodat het niet conflicteert met security. Zo wordt de app ook future proof indien de organisatie beslist om de app intern te implementeren. Zowel de mock als CGICraft worden in dit geval buiten het netwerk geplaatst.
2	Een applicatie binnen het netwerk van CGI waarbij de gebruikers het spel alleen kunnen spelen als ze verbonden zijn op het interne netwerk van CGI (tijdens werk).
3	Een standalone applicatie waarbij de back-end binnen het netwerk staat en deze verbonden is met de relevante systemen. De front-end staat buiten het netwerk en ontvangt alleen realtime gemanipuleerde data die beter past bij CGICraft, hiermee wordt de integriteit van de interne data bewaard.
4	Een standalone applicatie waarbij de gebruikers moeten registreren. Dit staat geheel buiten CGI en kan alleen problemen opleveren vanuit het marketing oogpunt.

Tabel 4: mogelijke architectuur oplossingen

De kans op succes is bij punt één het kleinst. Voor het mocken van de twee interne systemen heb je gegevens nodig over het origineel. De kans is groot dat dit niet mogelijk is vanuit het security oogpunt. Tevens is het qua planning haast niet mogelijk om twee systemen goed na te bootsen en deze te implementeren.

Punt twee heeft een kans om te slagen indien security het ontsluiten van data toelaat, maar dit gaat ten koste van gamificatie, aangezien het alleen intern beschikbaar is en er vrijwel zeker regels gebonden zullen zijn aan waar de applicatie aan moet voldoen.

Punt drie is in principe de beste oplossing omdat de data de werkelijkheid reflecteert maar zo nodig gemanipuleerd is dat het niet ten koste gaat van de interne integriteit. Ook hierbij moet security toegang geven tot de data.

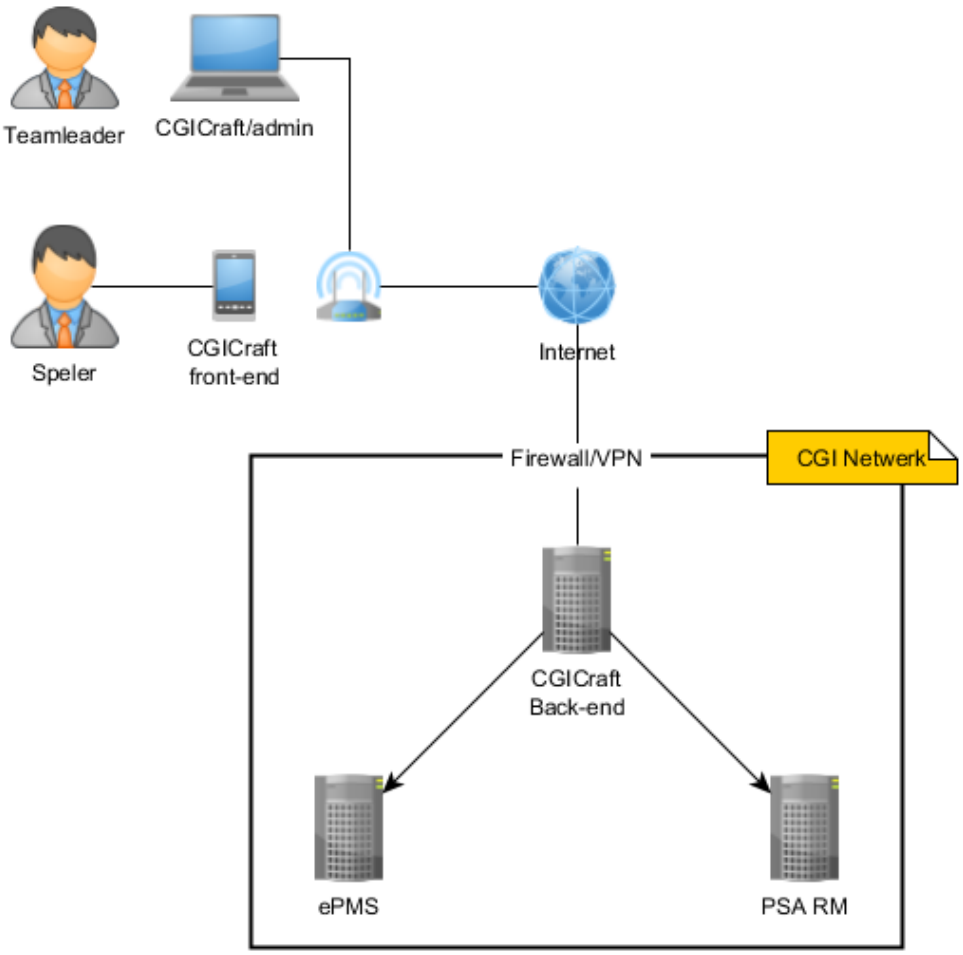
Punt vier heeft de grootste kans om te slagen doordat er geen problemen zullen ontstaan m.b.t. security. Het systeem is niet gebonden aan de interne systemen of het CGI netwerk. Vanuit gamificatie oogpunt wordt er ook niet geremd op creativiteit. Vanuit marketing oogpunt kan het echter wel problemen met zich meebrengen in verband met eventuele negatieve naam/merk associatie.

5.2 Deployment diagram

Het deployment diagram in figuur 2 is gebaseerd op de ideale situatie, punt drie in tabel vier. Er wordt naar gestreefd dat dit de uiteindelijke oplossing is. Wel is er aan het einde van dit hoofdstuk een alternatief aangeboden.

Voor de realisatie moet gelet worden op twee punten:

- Er kan/mag data opgehaald worden door de CGICraft back-end uit ePMS en PSA RM.
- Data uitwisseling is toegestaan/mogelijk tussen de front- en back-end van CGICraft door de VPN van CGI.



Figuur 2: deployment diagram

Een sterke alternatief voor figuur 2 is punt vier uit tabel vier. Er zijn geen grote security aspecten waar rekening mee moet worden gehouden en de data wordt zelf opgegeven door de gebruiker. Een nadeel is dat er niet gecontroleerd kan worden of de gebruiker alles naar waarheid heeft ingevuld tijdens de registratie. Verder ontstaat er dubbele data, op dit moment is dat niet veel maar dit kan een problemen worden in de toekomst. Daarnaast is de applicatie minder future proof dan zijn alternatieven. De applicatie staat niet in verbinding met de interne systemen en bij veranderingen in structuur zou alles handmatig aangepast moeten worden. De content is niet dynamisch ingeladen. Dit hoeft geen groot probleem te zijn maar hier moet wel rekening mee worden gehouden.

6 Projectorganisatie

De rolverdeling binnen het project is als volgt:

Naam	Functie/Rol	Contactgegevens
Sangam Gupta	Software Engineer / afstudeerder	Sangamgupta10@gmail.com +31611214837
Tjeerd Zijsling	Opdrachtgever / bedrijfsmentor	tjeerd.zijsling@cgi.com +31651389421
Alexander Chatzizacharias	Software Engineer / expert gamificatie	alexander.chatzizacharias@cgi.com
Nanny Jacobs	Begeleidende examinator	A.A.A.M.Jacobs@hhs.nl
Arno Nederend	Tweede examinator	A.a.Nederend@hhs.nl

Tabel 5: rolverdeling

7 Risicoanalyse

In dit hoofdstuk worden de risico's die kunnen optreden geïdentificeerd en in kaart gebracht.

Risico	**R1 – Geen PDP data**
Omschrijving	Huidige PDP data is niet beschikbaar vanwege de afgesloten architectuur. De database is niet toegankelijk en er is geen API.
Gevolg	Geen realtime gegevens over de huidige werknemers. Ieder werknemer die gebruik wil maken van het gegamificeerde PDP moet zich registreren.
Risiconiveau	Hoog
Mitigerende maatregel	Vermijden is lastig omdat het buiten de invloed van het team valt. Wel kan het tijdig ontdekt worden, tijdens de inception fase.
Compenserende maatregel	Een eigen database maken voor het opslaan van werknemers, deze moet dan gesynchroniseerd worden. Dit valt wel buiten de scope van het project.

Risico	**R2 – PDP voldoet niet aan security eisen**
Omschrijving	Nieuwe versie van PDP voldoet niet aan de security eisen van CGI of het huidige PDP is niet beschikbaar vanwege strikte security eisen.
Gevolg	PDP is niet inzetbaar op de productieomgeving binnen de CGI architectuur en er kan geen data worden opgevraagd/gedeeld.
Risiconiveau	Hoog
Mitigerende maatregel	Het is niet duidelijk wat de eisen zijn en of het überhaupt mogelijk is om data te ontsluiten uit de huidige PDP en de betrokken systemen.
Compenserende maatregel	PDP wordt een standalone applicatie. Die hopelijk alsnog de kracht van gamificatie laat zien en zodoende de doelstellingen haalt.

Risico	**R3 – Bijhouden documentatie & afstudeerverslag**
Omschrijving	Tijdens de construction fase moet ik de documentatie goed bijhouden. Dit kan zich voordoen als ik teveel nadruk leg op ontwikkelen.
Gevolg	Documentatie is niet consistent en niet goed vanuit het oogpunt van school.
Risiconiveau	Gemiddeld
Mitigerende maatregel	Vaste hoeveelheid tijd besteden aan het maken/bijwerken van documentatie.
Compenserende maatregel	Uitloop besteden aan het bijwerken van documentatie.

Risico	**R4 – Uitloop planning**
Omschrijving	Tijdens de construction fase is door het team de hoeveelheid tijd dat iedere user story gaat kosten verkeerd ingeschat.
Gevolg	Het team loopt achter op planning of krijgt de game onvoldoende af
Risiconiveau	Gemiddeld
Mitigerende maatregel	Goed bespreken met de opdrachtgever/developers of de planning per sprint realistisch is. Aangezien het team nog geen ervaring heeft en ze eerst een gevoel moeten krijgen voor de tijd die iedere taak kost. ("velocity")
Compenserende maatregel	Planning voor komende sprints aanpassen zodat het realistischer is met de capaciteit van het team.

Tabel 6: geïdentificeerde risico's

8 Planning en aanpak

Binnen het Delivery Center Java wordt gewerkt met de software ontwikkelmethode Disciplined Agile Delivery (DAD). DAD is een hybride ontwikkelmethode die gebouwd is op bestaande software ontwikkelmethodes zoals SCRUM, RUP en Extreme Programming (XP).

DAD kent explicit drie fases afgeleid uit RUP zoals getoond in figuur drie:

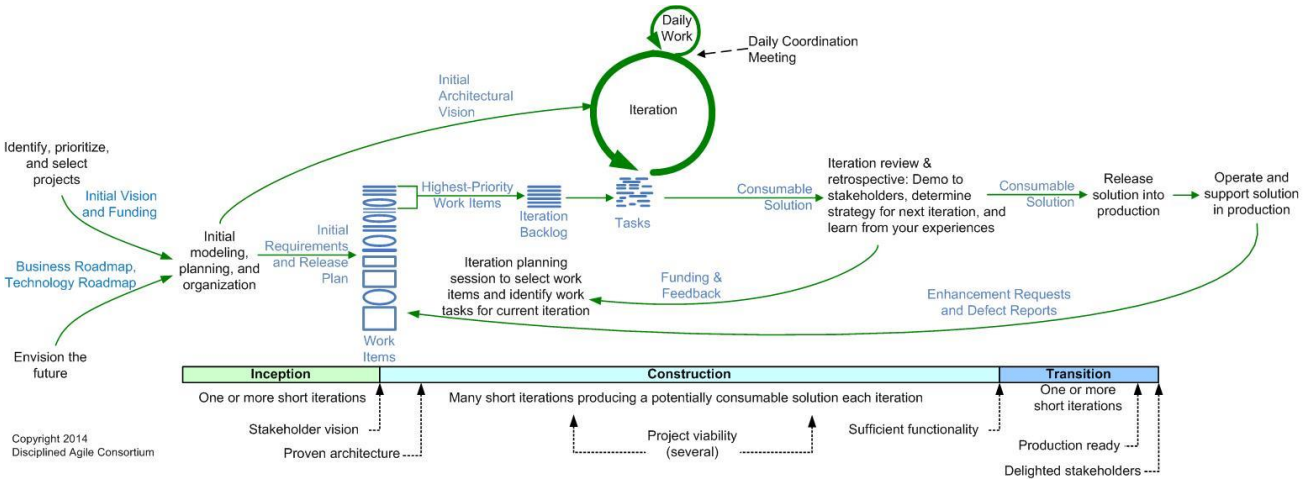
- Inception fase: de initialisatie van het project. Gedurende deze fase worden de wensen en eisen van de opdrachtgever in kaart gebracht, hierbij worden de juiste oplossingen bedacht. Voor dit project wordt dit vastgelegd in een Plan van Aanpak. Deze is voor dit project zodanig aangepast zodat het beter aansluit op DAD.
- Construction, ontwikkeling applicatie. Gedurende deze periode wordt er iteratief ontwikkeld aan het product. Elke iteratie levert een potentiëel opleverbaar product op. Naast ontwikkelen worden er ook unit testen uitgevoerd tijdens deze periode en de documentatie bijgewerkt.
- Transition, tijdens de transition fase wordt het product in gebruik genomen. Voor dit project is dat de afronding en het deployment naar de productieomgeving.

Elke fase is onderverdeeld in één of meerdere iteraties. Er is een aantal aanpassing gemaakt in het DAD zodat het beter past bij dit project. De DAD methode staat dit toe als de situatie hier om vraagt.

Ten eerste begint dit project met de inception fase: “Initial modeling, planning, and organization” zoals te zien in figuur drie. Alles hiervoor is niet van toepassing op dit project, het is namelijk al uitgevoerd voorafgaande aan mijn afstuderen.

Ten tweede valt ook “Operate and support solution in production” na de transition fase buiten de scope van dit project. Tevens is na de release support niet nodig gezien de schaal van dit project.

Als laatste zijn er geen Daily Coordination Meetings omdat het team hiervoor te klein is. In plaats daarvan worden er wekelijks gesprekken met de mentor ingepland om zo voortgang en problemen toch nog tijdig te bespreken.



Figuur 3: basis levenscyclus [4], uitbreiding op SCRUM

Per fase licht ik kort toe wat er wordt uitgevoerd en hoe deze eruit komt te zien.

Week	Periode	Fase	Activiteiten
1 t/m 3	11 mei – 29-mei	Inception	- Plan van aanpak schrijven
			- Gamedesign document opstellen
			- Requirements identificeren
			- Verdiepen huidige architectuur en keuze maken over type applicatie
4 t/m 15	1 jun – 21 aug	Construction / Transition	- Iteratie 1 t/m 6 CGICraft ontwikkelen - Design maken/bijwerken - Documentatie bijhouden/bijwerken - Testen - CGICraft deployen voor een testgroep
16 t/m 17	24 aug – 4 sep	Afronding	- Extra ruimte voor uitloop - Afronden afstudeerverslag

Tabel 7: planning voor dit project

In tabel zeven is er nog geen duidelijk onderscheid gemaakt tussen de transition en de construction fase. Het is namelijk mogelijk dat er halverwege de construction fase gedeployed wordt voor een testgroep, om zo de voortgang te meten en eventuele fouten of verbeteringen te ontdekken. Tijdens de ontwikkeling zal worden gekeken of zo'n extra deployment gewenst is.

Verder kent de construction fase zes iteraties van ieder twee weken lang. Ieder Iteratie begint met een iteratie planning sessie en eindigt met een iteratie review & retrospective waarin ik mijn voortgang laat zien en terugkijk op de sprint. Per iteratie planning sessie worden de taken van de huidige sprint bepaald.

9 Literatuurlijst

ID	Auteur	Datum	Titel	Type	Beschikbaarheid
1	Duolingo	2015, Mei 18	*Voorbeeld: gamificatie*	[Online]	https://www.duolingo.com/
2	Zombie run game	2015, Mei 18	*Voorbeeld: gamificatie*	[Online]	https://www.zombiesrungame.com/
3	Google Trends	2015, Mei 18	*Gamification Google Trends*	[Online]	http://www.google.com/trends/explore#q=gamification
4	Disciplined Agile Delivery	2014	*Full Agile Delivery Lifecycles*	[Online]	http://www.disciplinedagiledelivery.com/lifecycle/
5	Hamari, J., & Koivisto, J., Sarsa, H.	2014	*Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification*	[Online]	http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6758978

Tabel 8: literatuurlijst

Bijlage C – Game design PDP

Game design PDP

CGICraft

Name:	**Sangam K. Gupta**
Student number:	**10081224**
Institute:	**The Hague University of applied sciences**
Date:	**26-05-2015**
Version:	**0.3**

Design history

Version	Description	Date
0.1	First draft of the game design document	26-05-2015
0.2	Added multiplayer segment	12-08-2015
0.3	Improved grammar and added references	28-09-2015

Table of contents

10 Introduction

CGI is a big international ICT company. CGI finds it important that his members (employees) keep learning and sharing knowledge throughout their career, so they can grow as a person and be more valuable for CGI itself.

This is done through the personal development plan (PDP) system. The employees write their goals, motivation all to stimulate learning. The PDP is also used during assessment of the employees.

In reality the PDP is not used and unfriendly. To stimulate this and make it a more pleasant we want to use gamification. Gamification is the use of game elements in a not-game context. In recent years gamification has regained popularity [1]. CGI wants to measure the power of this new medium to increase their member engagement.

11 Gamification

I did not use an official research method. Instead I did lots of desk research. In this chapter I will describe what I found and learned.

11.1 What is gamification

Gamification has gained popularity in recent years [1]. It is only a matter of fact before everyone wants to try this innovative way of building system where a good user experience is key.

Gamification according to the oxford dictionary is [2]:

"The application of typical elements of game playing (e.g. point scoring, competition with others, rules of play) to other areas of activity, typically as an online marketing technique to encourage engagement with a product or service:"

To simplify it means by adding game elements to a not-game context we can increase user engagement. The next question I wanted to answer was "How does it work?"

11.2 How does gamification work

How does adding game elements to a non-game context increase user engagement? To answer this question I did some research by looking through various papers, reading blogs and learning about the different frameworks as well as talking to the gamification expert inside CGI, who initiated this project. I listened to his reasoning and used that to combine different elements of gamification to determine what the best way is to support the PDP, which I will describe in the next chapter.

To get back to the question of How? Gamification uses motivators to get humans to perform a task and make it "fun" while doing it. (I used this [3] paper for the basis of my research). In principle it is the same concept as games use. You bring a challenge to the table and make it just hard enough. When the task is performed the user experiences a feeling of accomplishment and Joy. [4]

What I also noticed is that according to the expert gamification uses intrinsic and extrinsic motivation [5] to keep users engaged through different lifecycles of user experience.

Gamification as the name says it uses game like elements. There is no standard for the game elements that should be used as far as I could find. But still widely used and popular game elements include:

- Progress mechanics (points/badges/leaderboards, or PBL's)
- Narrative
- Player control
- Immediate feedback
- Opportunities for collaborative problem solving
- Scaffolded learning with increasing challenges
- Opportunities for mastery, and levelling up
- Social connection

11.3 How we use gamification

To link gamification with PDP and our project we looked at what we had right now. The mock-up is built with the idea that the core of the app will be a quiz, where players fight against or with each other to gain points and rank their skills. With core of application we mean that it will be a quiz which will be supported with game elements.

While all these game elements are used to engage players and keep them engaged it is by no means directly connected to the PDP. So the link the PDP with a gamified system I searched for ideas. I looked at various applications that use gamification primarily for e-learning. I listened to talks about gamification in the hope to find ideas that could work in our use case. I also talked to Employees of CGI where I pitched my ideas. I came to the conclusion that linking PDP with an application will be hard and we won't do it. Because the link isn't clear and could potentially lead to misinterpretation of the application. People will think it is part of the PDP and thus won't use it.

So we decided to use no direct link between the PDP and the game. There will be no direct 'physical' link between the two systems. Instead we want to encourage the employees and motivate them enough so they will work on their career be that with the PDP or self-study for example. We also want to give positive feedback for employees who are actively working on themselves. For example, the teamleader could check the accounts of their subordinates. They can use the information that the game gives about their employees to further encourage new practices and talents. People also can try new field/expertise in a playful way without being judged. In short we use gamification to motivate players to take initiative and improve their self.

11.3.1 What is PDP

So now we discussed how we want to achieve our goals we will discuss what technics we are going to use. Meaning what game elements are we going to use to motivate players and how it all ties together.

Let's start by the different phases of user experience. In table 1 you can find the four lifecycles of an user experience.

Phase	Description
Discovery Phase	How to promote the game and attract users.
Onboarding Phase	We have the attention of an user, how can we keep them engaged and make them try out more (of the game).
Scaffolding Phase	The players gained experience. How do we keep these players engaged? What progression-benefits do they get by playing this game?
Endgame Phase	During this phase the player is basically a veteran. How can we promote him to improve the game and attract new users?

Table 1: different lifecycles of user experience

In the next table we have the different game elements combined with motivation and user experience lifecycle. The link between this is by no means scientifically supported. It is based on assumptions.

Phase	Goal	Motivation	Game elements
Discovery Phase	Discovery of the game. Attract to ' try out'.	External, short term motivation	• Name • Look and Feel • Short term unlockable • Singleplayer
Onboarding Phase	Stimulate learning. Indirectly increase use of PDP.	short term motivation	• Level system • Ranking system • Point system • Unlockable • Multiplayer • Singleplayer
Scaffolding Phase	Stimulate knowledge sharing. Stimulate learning. Indirectly increase use of PDP.	Short with a transition to long term motivation	• Ability to add new questions, • All the elements described above • Better unlockable • Team system
Endgame Phase	Stimulate knowledge sharing. Stimulate learning. Indirectly increase use of PDP.	Mostly long term	• Feedback system(add/improve/remove questions and answers) admin • All of the above

Table 2: overview gamificatie van het PDP

We came up with the different game elements by looking at other gamified systems. We specifically looked that e-learning system. Here we found that many successful gamified systems like Duolingo use quiz like gameplay to test the player's knowledge.

Next up we tried to combine different gam elements to find what would build a coherent system. I also asked the employees whether they liked the idea. Besides that, I looked at the mock-up and how it possibly could be improved.

Next we describe the elements we came up with.

Singleplayer

Singleplayer will be a quiz consisting of ten questions. Each question has multiple answers. By answering these questions correctly the player can earn points. If the player answers an x number of questions correctly he wins the quiz. If he doesn't he loses. X will be calculated based on the difficulty of the quiz. If the quiz is relatively easy the player will need to answer more questions correctly and vice versa.

After the quiz finishes the player gets a chance to win a reward. What the reward will be has yet to be determined.

We chose for various reasons that the core of the game will be a quiz. First of all, it is dynamic and can be tailored for each practice and player. Second, quizzes are a standard during a learning process. It can be seen as an exam, but that is not what we are aiming for. It should be more fun. Next, is easy to add various elements out of table 2 to support the quiz. Lastly we couldn't find anything that would work as a casual game.

Multiplayer

Multiplayer is basically the same as singleplayer the biggest difference is that you play against another player. Multiplayer is one versus one. Both players receive the same quiz. They then continue to answer each question. At the end the player with the most correct answers wins the game.

Multiplayer is expandable. It can be played with more than two players. But for the short term and this project we will keep it at two players. It can also be improved for example by adding a timer to each question. The players will need to answer the question within the time limit. This could potentially add an element of excitement to the quiz.

Ranking system and leaderboard

Each player has a numerical ranking. The base ranking is 1200. The players can increase their ranking by winning games and decrease it by losing. During multiplayer the system puts players with roughly the same ranking against each other. This way the players are evenly matched. The players who play multiplayer will also be listed on a leaderboard. The leaderboard shows players how they stack up against other players.

The leaderboard is also meant to motivate players to play more. We want players who are good at the game and earn lots of points and be recognized in the long term. Other players can maybe use this leaderboard system to find people they need for project in real world situations.

For the ranking system we chose elo. The elo system is a widely used system in chess world. This system has since then also been implemented in many games, where one of the most played game in the world ‘League of Legends’ proved that this system can work for games. Elo system is customizable and scalable. It can be tailored for this game.

Unlockable/rewards and level system

Each player has a level. The level a player has indicates his skill and knowledge. Players start at 1 and as they progress through the game they earn points towards their level. The level can be used to unlock certain functionality or rewards.

The rewards are generated randomly after each game if the player won. The rewards are for short term motivation. They also can be used to customize their in game persona. By allowing customization we want to give players the ability to make their persona unique.

Also the level system has many subsystems. CGI has practices, talents soft and hard skills. Inside the game each element will have progression and level of their own. All of this will add up to level of the player. There is a separation between these elements so the player can get insight in her strong and weak points. The players can also try out new practices, talents and skills with which they are not familiar with. The questions the player gets will be based on their selected skills, practices and talents.

Feedback system

The feedback system is given to the players when they achieve a certain level. The feedback system is basically the ability to remove, add and edit certain answers. They can be seen as an admin.

Whether this will work is to be determined. We don’t know if this will catch on, and will have a big enough user group to become self-sustaining. This is needed because the core of the game is a quiz. A quiz becomes repetitive fast when the users will answer the same questions over and over. By giving the ability to users to create their own questions and add this to database we can keep the quizzes dynamic and fresh. This way the more players you have the better the system will work.

12 Game Overview

12.1 Game Concept

The main concept is to motivate players to learn and share knowledge this will be achieved with core of the game which is based around a quiz. This will be called battle. The battle that takes place is the main way a user can level his account and earn unlockable or badges.

The user will create and account or use an existing account from the database inside CGI. Either way the player will be a digital copy of himself.

Next he can play solo battles. Which I described in chapter 2. After a certain level he can also play ranked (multiplayer battle). By winning battles the player earns rewards and gains points for their level.

12.2 Feature Set (requirements)

The initial feature set will be included in the Plan of Approach.

12.3 Genre

There are two themes. We have business casual and the second one is the total opposite. It is cartoony and will be closer to a real game. The reason why we chose these themes is because we don't want to distract the players with too many visuals and graphical content. Also because the target audience is mature and probably never plays games on regular bases it will be better to be subtle and not introduce the stigma that is usually associated with gamers.

The two themes are really abstract and can mean anything. With CGICraft we don't want to aim for a specific genre, we want it to be learning platform. The closest genre to this is puzzle.

12.4 Target Audience

The Target audience are the employees of CGI. CGI is a technical (ICT) company. The atmosphere inside the company is business casual. Majority of the employees are over the age of 21. The demographic is old (estimation 25 and higher) and hints towards 'serious'. Assumption is also that many people don't game or keep it at a casual/social level.

This was not officially determined by research but more by experience by talking to different employees.

Furthermore the company has many types of people with different fields of specialty. This introduces a big amount of differences among the employees and therefore should be accounted for when making the game.

There are a total of ten different practices inside CGI each practice has many fields. Each field has a specialized goals and skillsets.

For the proof of concept the practice we are targeting is Java Software Engineer. We are targeting this practice because that's the practice we are most familiar with. This has the advantage that we can produce results faster and potentially more accurate.

12.5 Game Flow Summary

The player starts by making a virtual avatar of him. He or she then uses their respective field and skillsets. They can afterward join a team or play individually. Each player will be ranked based on their wins and losses.

After the player created an account he or she can play games get insight in their profile and also unlock new skills, practices and talents.

The players increase their level by training their soft and hard skills. This is done by winning a battle.

By winning a lot a player can unlock new skills to practice and this way determine where her or his interest lies. Also if the player wins a lot in their respective field they will gain recognition. Recognition can be useful to others if they need help.

12.6 Look and Feel

The setting for the game should be focused on casual gaming or social gaming. It won't have hard core game vibe and should be simple in design. The other point that should be accounted for is the CGI standards. CGI has standardized their theme and this theme should be part of the game so it feels like it is built for CGI.

12.7 Project Scope

The scope of the project will be added to the Plan of Approach.

12.7.1 Number of badges, skills, etc

To be determined.

13 Gameplay and Mechanics

13.1 Gameplay

It's a 2d game. The gameplay is simple and can be played with only a mouse or fingers. The core of the game is a quiz. After the player logs in he or she can start a new quiz. The quiz varies from the number of question it will ask or consist of. The player will answer each question the number of right answers will determine the points he or she received based on this number he or she can also earn a badge. The chance is based on the points the player received. So for the most part it is click and play.

13.1.1 Game Progression

The player progresses like he or she would progress inside the company. This of course has a deeper meaning. With same progression as inside the company the player can get a feel for what they can achieve in real life based of the game. This is partly because the information and skills levels that the player wants to gain are mostly one to one. So there is some kind of progression but it is not relevant.

13.1.2 Mission/challenge Structure

Most missions are the quizzes that are made by the community. There can be an expansion where the community build challenges that players can do. For example, win 10 times in a row!

13.1.3 Puzzle Structure

There two types of puzzle in this game. One is team based puzzle and the other is individual puzzle. The team based puzzle has not been discussed because that will not be implemented in the first version. In principle both are based on the same mechanics. One is multiple players versus multiple players and the other is one versus one. The quiz will be generated based of the selected skills, practices and talents of the player. If you do a team battle then it will be based of the current skills of the team. This way every party gets a customized quiz.

The puzzle will give points. The points can be lower or higher depending on the difficulty of the battle.

After each answer a bubble will appear and the answer you chose will be highlighted red or green based on wrong or right. The bubble will function as portal to a discussion. You can up vote the question and rate it on difficulty. When clicking on the bubble you will go to the discussion page which is a simple comment and reply system.

After you finish the quiz there will be a stats screen where you can see various stats.

13.1.4 Objectives

The objective for the player is too learning trough playing. The incentive here is that the player can use this strategy to promote himself. This in turn can lead to a better evaluation. The player doesn't only learn new skills and knowledge but also earns recognition. This recognition can for example lead to promotion or finding new projects to work on.

13.1.5 Play Flow

13.2 Screen Flow

13.2.1 Screen Flow chart

13.2.2 Screen Descriptions

13.2.2.1 Main Menu Screen

13.2.2.2 Options Screen

There is no options menu. It would have no impact on the user experience and was not considered for this project. For future functionalities it is recommended to include an option screen.

The option screen should include a textbox containing the URL to the server. This is needed because the server can change a lot, for example from external to internal hosting. When this happens the users can manually change their server URL. So there will be no need to update the app. updating the app can be experienced as cumbersome. Because it will only run internal and thus won't be part of play store. This means there won't be a centralized place to push updates to the users. CGI doesn't have an internal app store.

14 Interface

14.1 Visual System

14.1.1 HUD - What controls

14.1.2 Menus

14.1.3 Rendering System

14.2 Control System – How does the game player control the game? What are the specific commands?

14.3 Audio

14.4 Music

14.5 Sound Effects

14.6 Help System

15 Artificial Intelligence

16 Technical

16.1 Current architecture

The current architecture will be described in Plan of Approach. Further architecture will be described in architectural document.

16.2 Type of application

The type of application the PDP has to become has yet to be determined. The choice has been narrowed down to the three types of application best suited for the PDP:

- Native mobile application
- Unity mobile application
- Web application

In the following table each type of application will be listed together with its pros and cons.

Application type	Benefits	Consideration
Native mobile application	- Support for handheld devices. - Ease of use and availability out of office - Can be both offline as online - Low overhead - Free	- Limited screen area - Development cost and time - Limited by device - Limited to one OS
Unity mobile application	- Cross platform development - Easy to program - Fast results and guaranteed quality	- High overhead - Can be slow - License fee
Web application	- Supported by all devices even mobile if designed right. - Ease of deployment	- Game like design harder to implement - Always online no offline available

Table 3: pro's and con's different platforms

There are two big choices to be made. One is whether we want to have a mobile application or a desktop application. Second choice arises when we choose for mobile. Do we want cross platform or not.

We chose for mobile. The assumption is that the app won't be used during workhours. The employees will be busy.

The application is more likely to be picked up after work for example during a trip in the train or at home. Mobile phones are better in this aspect because it supports offline capability. Mobile apps are also easier to pick up and more associated with games and less with work.

Webapps on the other hand are easier to develop and have a broader audience. Besides that it is better for the architecture. A webapp is more secure than a mobile platform. On mobile platform you don't have power to see how the player uses their data e.g. like sharing it with friends. Also updates for webapp are easier to push compared to mobile app.

We chose for mobile mainly because it is easier to pick up and also more personal. Besides that the application will have some game like animation/elements which deliverer a better experience on a mobile application.

We also chose for native instead of cross platform. The main reason for this is because there is no good development environment for cross platform development. Xamarin is seen as the best development SDK but at CGI we don't have a license for it. A webapp is not suitable for animation and doesn't deliver the best experience. Besides that I have experience in native Android development. With native I can focus more on developing the application instead of learning a new language (IOS). I expect the product that I make will be superior on Android than on IOS or on cross platform application. And also I don't have an IOS device for testing or a MacBook for developing and neither does CGI.

Although, after all of this IOS is still the preferred platform because it is supported by CGI. In the future if the proof of concept is successful an IOS version can be build.

16.3 Target hardware

The target hardware will be Android devices. The API we chose will be a minimum of 15. With a minimum API of 15 we have an audience of 90% as seen in figure 1.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

Data collected during a 7-day period ending on June 1, 2015.
Any versions with less than 0.1% distribution are not shown.

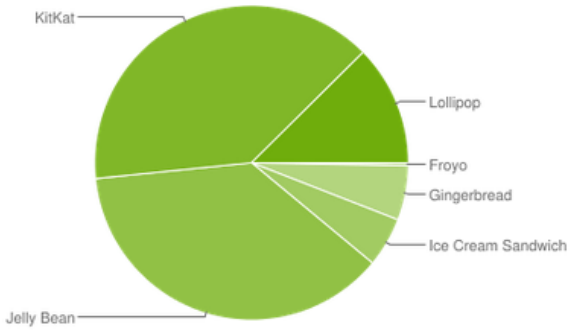


Figure 1: distribution Android API

Although we chose for Android we don't know the percentage of Android users inside CGI Netherlands. Also IOS is the only officially supported mobile operating system. But because I can produce better and faster results in Android. I chose for Android.

16.4 Development hardware and software

For back-end we chose Spring. Spring is the standard software used inside DC Java. DC Java is the department I will be working at. The hardware I will use during development will be the laptop I will receive from CGI.

The front-end will be an Android app. For this the hardware I will use will be my own phone a nexus 5.

For the database I chose MySQL. The database should have no impact on the application because of the size of this project and because there is a persistent layer between the back-end and the database, swapping type of database will have no effect on the back-end. Also I have experience with MySQL and can produce results faster.

16.5 Development procedures and standards

I will follow the development standards set by CGI. These standards can be found on the Intranet of CGI. When there are no standards set by CGI I will use best practices as found on the web.

16.6 Design documents

The design documents can be found in my graduate report**.**

Bibliography

ID	Auteur	Date	Title	Type	Availability
1	Google Trends	2015, May 18	*Gamification Google Trends*	[Online]	http://www.google.com/trends/explore#q=gamification
2	Oxford Dictionary	2015, May 18	*Oxford Dictionary*	[Online]	http://www.oxforddictionaries.com/definition/english/gamification
3	Hamari, J., Koivisto, J., & Sarsa, H.	2014, January	*Does Gamification Work? — A Literature Review of Empirical Studies on Gamification*	[Online]	http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6758978
4	Yu-kai Chou	2015, January 3	*What is Gamification*	[Online]	http://www.yukaichou.com/gamification-examples/what-is-gamification/#.Ve198hHtmkp
5	Muntean, C. I.	2011, October	*Raising engagement in e-learning through gamification*	[Online]	http://icvl.eu/2011/disc/icvl/documente/pdf/met/ICVL_ModelsAndMethodologies_paper42.pdf

Table 4: Bibliography

Bijlage E – Voortgangsverslag

Voortgangsverslag

Auteur: Sangam Gupta
Studentnummer: 10081224
Opleiding: Informatica Zoetermeer
Opleidingsinstituut: De Haagse Hogeschool
Stagebegeleider: Nanny Jacobs
Versie: 1.0
Begeleider Bedrijf: Tjeerd Zijsling

Inhoud

1. Inleiding

Dit document beschrijft mijn huidige voortgang van de afstudeerstage bij CGI. Dit voortgangsverslag is bedoeld voor de begeleidend docent. Het document bestaat uit de volgende onderwerpen: Behaalde mijlpalen, obstakels tijdens de doorlopen periode en de resterende planning.

2. Voortgang

In dit hoofdstuk beschrijf ik mijn huidige voortgang. Hierbij bespreek ik de bereikte mijlpalen, de obstakels en de belangrijke besluiten.

2.1 Bereikte mijlpalen

De planning zoals beschreven in het Plan van aanpak ziet er als volgt uit:

Week	Periode	Fase	Activiteiten
1 t/m 3	11 mei – 29-mei	Inception	- Plan van aanpak schrijven
			- Gamedesign document opstellen
			- Requirements identificeren
			- Onderzoeken huidige architectuur en keuze maken over type applicatie.
			- Gamificatie onderzoeken
4 t/m 15	1 jun – 21 aug	Construction / Transition	- Iteratie 1 t/m 6 CGICraft ontwikkelen - Design maken/bijwerken - Documentatie bijhouden/bijwerken - Testen - CGICraft deployen voor een testgroep
16 t/m 17	24 aug – 4 sep	Afronding	- Extra ruimte voor uitloop - Afronden afstudeerverslag

Tabel 1: planning uit het PVA

Op het moment van schrijven zit tegen het einde van iteratie twee. Tot nu toe loopt alles volgens de planning. Op basis van het product backlog dat was opgesteld tijdens de inception fase wordt met het huidige tempo verwacht dat ik voor kom te lopen op de planning waardoor de opdracht eventueel uitgebreid zou kunnen worden.

Het product backlog ziet er als volgt uit:

ID	Doelstelling	Omschrijving	Prioriteit
DS1	**Stimuleren kennisdeling**		
US1.1		Als een gebruiker kan ik nieuwe vragen en antwoorden toevoegen aan het systeem.	M
US1.2		Als een gebruiker kan ik mijn profiel aanpassen en delen, zodat ik mijn voortgang kan laten zien aan mijn collega's.	M
US1.3		Als een gebruiker kan ik zoeken naar een team, waar ik lid van wil worden.	S
US1.4		Als een gebruiker kan ik lid worden van een team, om teambattles te spelen.	S
DS2	**Stimuleren kennisontwikkeling**		
US2.1		Als een gebruiker kan ik een nieuwe (quiz)battle starten, zodat ik kan stijgen/dalen in ranking.	M
US2.2		Als een gebruiker kan ik zien hoeveel spelers een reactie hebben geplaatst bij een vraag, zodat ik een beeld krijg bij de populariteit van de vraag.	M

ID	Doelstelling	Omschrijving	Prioriteit
US2.3		Als een gebruiker kan ik een reactie plaatsen of reageren op een bestaande reactie, zodat ik de vraag kan bespreken.	M
US2.4		Als een gebruiker kan ik de vraag beoordelen met een 'like', zodat de vraag vaker terugkomt in battles voor andere spelers.	M
US2.5		Als een gebruiker kan ik uit een overzicht van goals kiezen, zodat ik mijn eigen goals kan bepalen.	M
DS3	**Verhogen gebruik PDP**		
US3.1		Als een gebruiker kan ik mijn ranking zien op de leaderboard, zodat ik kan zien waar ik sta tegenover de rest.	M
US3.2		Als een gebruiker kan ik de globale ranking zien.	C
US3.3		Als een gebruiker kan ik badges verdienen door battles succesvol af te ronden, zodat ik mijn badges kan toevoegen aan mijn avatar.	M
US3.4		Als een gebruiker wil ik mijn avatar level verhogen door middel van battles, zodat ik nieuwe badges en nieuwe goals kan unlocken.	M
US3.5		Als een gebruiker kan ik mijn individuele skills levelen door middel van battles, zodat ik mijn speler level kan verhogen.	M
US3.6		Als teamleader kan ik de profielen van mijn members (spelers) zien zodat ik met deze gegevens rekening kan houden tijdens een evaluatiegesprek met de member.	C
US3.7		Als een teamleader kan ik een overzicht zien van al mijn members.	C
US3.8		Als een team kunnen we een nieuwe battle starten, zodat we gezamenlijk in rank kunnen stijgen of dalen.	S
US3.9		Als gebruiker kan ik stijgen en dalen op de leaderboard.	M
US3.10		Als gebruiker kan ik inloggen, zodat ik het spel kan spelen.	M

Tabel 2: work items(backlog) zoals opgenomen in het PvA

De backlog is ondertussen wel verder gespecificeerd.

De globale planning voor de construction fase ziet er als volgt uit:

Iteratie	Week	Taken
Iteratie 1	Week 4 t/m 5	- Eerste opzet App en server - Architectuur bewijzen - Inschrijven/login - Database
Iteratie 2	Week 6 t/m 7	- Quiz bouwen en bespelen(singleplayer) - Navigatie uitwerken/implementeren - HttpClient kiezen en implementeren (Communicatie server/ app)
Iteratie 3	Week 8 t/m 9	- Testen volgens richtlijnen CGI - Profiel en home pagina - Categorieën kiezen - Quiz personaliseren - Punten systeem

Iteratie 4	Week 10 t/m 11	- Punten systeem - Rewards en andere gamificatie elementen - Leaderboard en ranking - Feedback (Community elementen) - Vragen toevoegen - Multiplayer
Iteratie 5	Week 12 t/m 13	- Multiplayer - Deployen bij testgroep
Iteratie 6	Week 14 t/m 15	- Feedback verwerken - Nog in te vullen
Afronding/uitloop	Week 15 t/m 17	- Afstudeerverslag schrijven

Tabel 3: globale planning Construction fase

2.2 Obstakels

Er waren niet al te veel obstakels. De meeste obstakels zijn opgelegd vanuit CGI. Zo kan ik de interne systemen niet benaderen. Ook is er geen mogelijkheid om interne systemen van buiten het CGI netwerk te benaderen. Een grotendeel van de obstakels zijn vastgelegd in de vorm van risico's tijdens de inception fase in het Plan van Aanpak. Dus ik was er bekend mee.
Het feit dat het interne netwerk van CGI niet benaderd kan worden zorgt er tevens voor dat de server(backend) niet intern gedraaid kan worden. Hier wordt naar gekeken bv. door eventueel een externe server te huren en gebruiken. Op het moment is er voor gekozen om tijdelijke server bij mij thuis op te zetten. Zodat de app ook buiten de productieomgeving getest kan worden door de opdrachtgever. Tevens is dit nodig voor iteratie 5 zodra de app gedeployed gaat worden voor een testgroep.
Verder is er een obstakel bijgekomen die ik niet had verwacht. Dat is namelijk het ontwerpen van de applicatie. Ik ben geen designer en er zijn ook geen interne designers die mij kunnen helpen bij dit onderdeel. Er is dan ook voor gekozen om het design pas op het einde uit te voeren. Tot die tijd worden er voornamelijk functionaliteiten toegevoegd. De app kan gezien worden als een game vandaar dat dit element wel belangrijk is.

2.3 Belangrijke besluiten

Belangrijke besluiten zijn voor een groot deel genomen tijdens de inception fase het gaat hierbij om de architectuur en hoe het PDP gegamificeerd gaat worden. De keuzes zijn gemaakt met behulp van experts op het gebied van architectuur en gamificatie.
De besluiten die zijn genomen tijdens de construction fase betreffen voornamelijk design keuzes zoals opbouw, welke library ik kies, hoe ziet het datamodel eruit etc.

3. Resterende planning

De rest van de planning is te zien in tabel drie. Het is belangrijk dat we zo snel mogelijk een app uitbrengen die we kunnen testen bij een testgroep. Afhankelijk van de feedback bepalen we wat er verder gaat gebeuren.

Bijlage D – Tussentijds assessment

Bespreking concept	**Tussentijds assessment**	Eerste beoordeling

Formulier tussentijds assessment

Student:	Sangam K. Gupta	**Studentnummer:**	10081224
Datum:	21 september 2015	**eerste / tweede TTA:**	eerste

Tijdens het tussentijds assessment is het volgende geconstateerd:		**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*	X	
b	*Het afstudeerdossier is digitaal beschikbaar*	X	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*	n.v.t.	
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	X	
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	

Aanpak	**O**	**T**	**V**	**G**
Passend			X	
Theoretisch verantwoord			X	
Samenhang uitvoering beroepstaken			X	

Beroepstaken op afgesproken niveau uitgevoerd?		**O**	**T**	**V**	**G**
1	Uitvoeren analyse door definitie van requirements			x	
2	Ontwerpen systeemdeel			x	
3	Bouwen applicatie			x	
4	Uitvoeren van en rapporteren over het testproces		x		

Producten	**O**	**T**	**V**	**G**
Tussenproducten			x	
Eindproducten		x		

Effectief communiceren	O	T	V	G
Binnen afstudeerbedrijf			x	
Afstudeerdossier			x	

Reflectie	O	T	V	G
Inzicht in eigen functioneren	x			
Inzicht in eigen leerproces	x			

Toelichting per beoordelingscriterium

Aanpak
De keuze voor DAD wordt deels verklaard vanuit de wens vanuit de organisatie, maar er ontbreekt nog een gedegen onderbouwing vanuit de student. De aanpak wordt wel gedurende het project zichtbaar gehanteerd.

Beroepstaken op afgesproken niveau uitgevoerd?
Uitvoeren analyse door definitie van requirements : in de bijlage zijn de requirements te zien. Het tot stand komen van de requirements en de eventuele aanpassingen zijn minder zichtbaar. De relatie tussen het doel van gamification en de requirements blijft onduidelijk. **Ontwerpen systeemdeel : er is een ontwerp zichtbaar. Bij de gebruikte techniek is het onduidelijk welk klassendiagram er gebruikt is (design of analisys.)** **Bouwen applicatie : het is duidelijk dat er software geproduceerd is, maar de op grond van de huidige beschrijving is er ook voldoende complexiteit zichtbaar.** **Uitvoeren van en rapporteren over het testproces : is nog aan de magere kant. De koppeling met de requirements lijkt te ontbreken, door een onduidelijke Defintion Of Done.**

Producten
Het uiteindelijke software product is nog onduidelijk. De opgeleverde tussenproducten geven helaas nog een onvolledig beeld van het gewenste eindproduct.

Effectief communiceren

De afstudeerder lijkt de opdrachtgever en andere stakeholders in voldoende mate betrokken te hebben bij zijn project, maar een onafhankelijk oordeel hierover van de partijen ontbreekt nog.

Reflectie
Is nog niet beschikbaar, hoewel iedere iteratie wel een reflectie bevat is het nog onduidelijk of de afstudeerder voldoende lerend vermogen laat zien bij de reflectie.

Advies

Aankruisen welk advies de student heeft gekregen

X	Inleveren **(bindend advies)**
	Verlengen **(vrijblijvend advies)**
	Stoppen **(vrijblijvend advies)**

Besluit student

Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

	Afstudeerdossier wordt op afgesproken datum ingeleverd	Inleverdatum:
	Afstudeerperiode wordt verlengd	Inleverdatum:
	Student stopt met afstudeeropdracht	

Naam begeleidend examinator: Nanny Jacobs
Naam tweede examinator: Arno Nederend
Datum: 21 september 2015

Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de coördinator van At Work faculteit ITD (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.