# Architectuur-beschrijving

## MMORPG ARCHITECTUUR – 2D TOPDOWN MEDIEVAL

Finn Alberts, Aaron Dang, Laurent Dassen en Peter Derks
ZUYD HOGESCHOOL | HBO ICT

# Inhoudsopgave

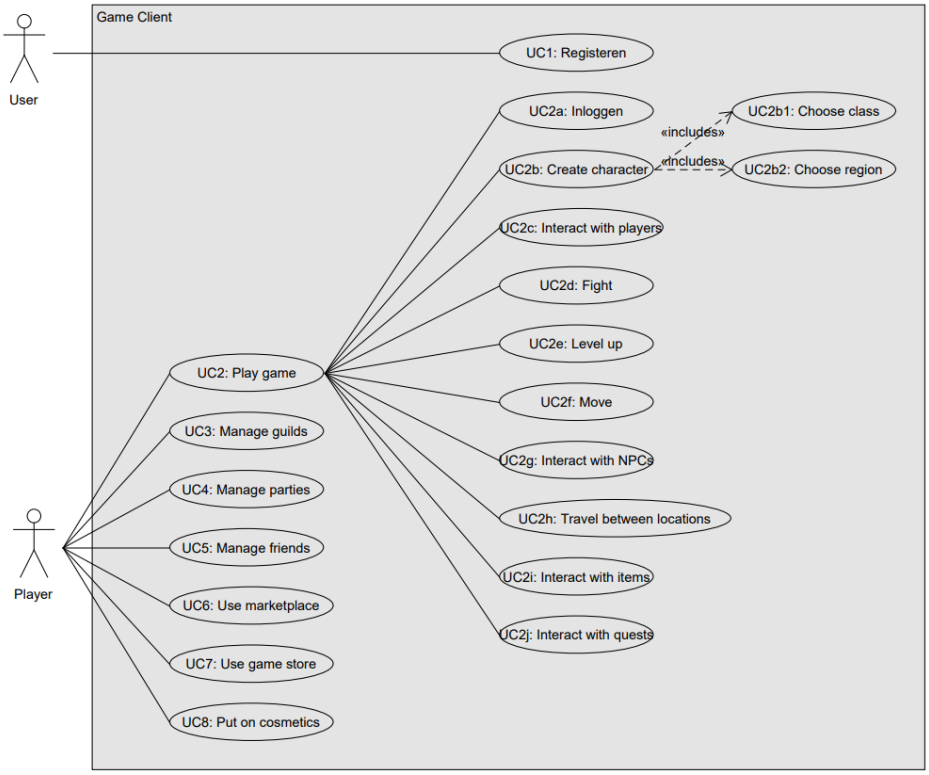
# 1. Beschrijving van de MMORPG

Starfall Online is een top-down view 2D pixelart open world MMORPG die plaats vindt in een middeleeuwse fantasiewereld Astaria waarin mensen, elven, dwergen, feeën en veel meer leven. Astaria wordt bedreigd door duistere sterren die zomaar uit de lucht vallen. Deze vallende sterren bevatten een onbekende zwarte magie dat alles wat levend is corrupt. De corrupte wezens zijn veel sterker dan hun normale tegenhangers en vallen alles wat in hun zicht is aan. De taak van de speler is om Astaria te beschermen tegen deze vallende sterren. De speler kan kiezen uit één van de vijf klassen namelijk warrior, ranger, mage, supporter en tamer om tegen de corrupte wezens te vechten en de vallende ster te vernietigen.

## Vallende ster

Het vallende ster event binnen de game werkt als volgt. Op willekeurige momenten vallen er een aantal vallende sterren in willekeurige locaties en die locatie worden corrupt. De spelers moeten vechten tegen een aantal waves van vijanden waarvan de laatste wave een baas is. Als de baas verslagen is, verdwijnt de ster en wordt de locatie weer normaal.

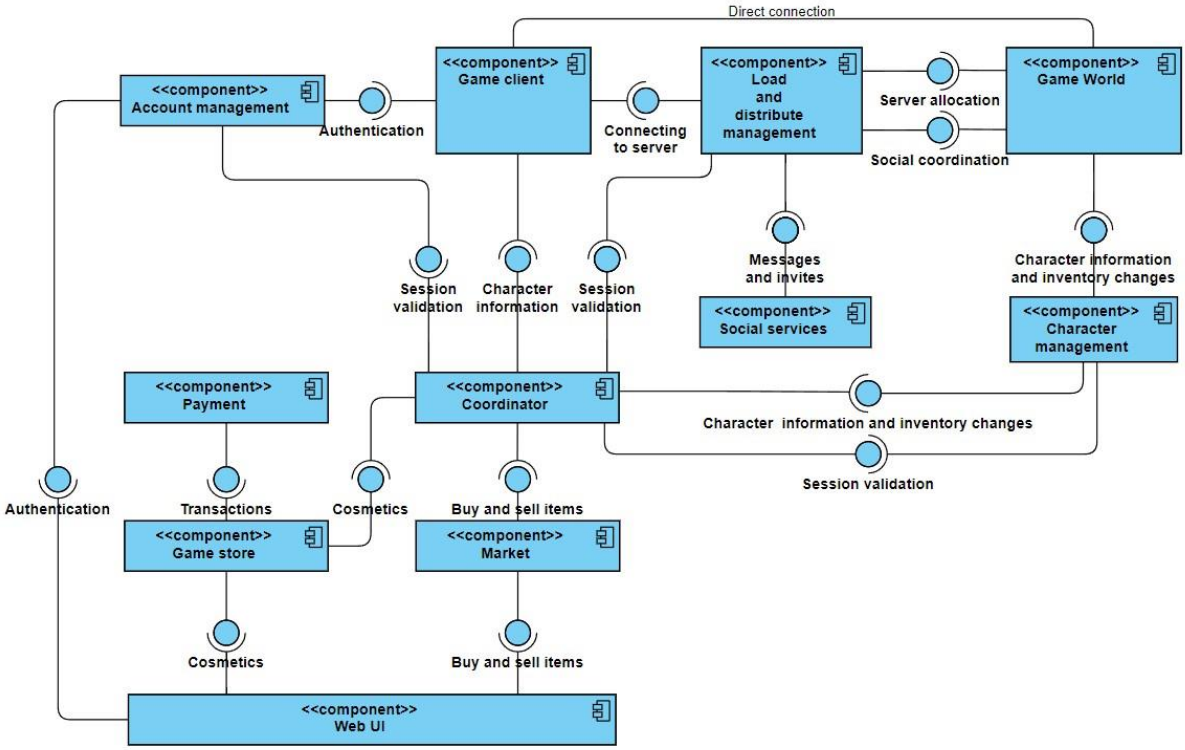
# 2. Scenario's




*Figuur 1 Use-case diagram*

In Figuur 1 zijn de verschillende acties te zien die een gebruiker kan uitvoeren. Dit diagram bevat alleen de acties die een gebruiker zou mogen uitvoeren. Binnen (online) spellen wordt echter vaak door spelers valsgespeeld. Met deze valsspelers moet ook rekening worden gehouden.

# 3. Development view




*Figuur 2 Component diagram*

## 2.1 Componenten

### Account management

Dit component wordt gebruikt voor alle account-gerelateerde functionaliteiten. Dit omvat registreren, inloggen, authenticatie, enzovoorts.

### Game client

De game client draait lokaal op de apparatuur van de gebruikers. In de client zullen de spelers inloggen en een regio kiezen. Vervolgens zullen de spelers een karakter maken en deze karakters in de spelwereld laden. De game client handelt de lokale inputs van de speler af en stuurt deze door naar de server. Ook ontvangt de client informatie over de andere spelers en server entiteiten van de server en verwerkt deze lokaal.

### Game world

Binnen het game world component draait de spelwereld. De spelwereld bestaat uit meerdere verschillende locaties, die ieder een spelerslimiet kunnen hebben. Indien deze limiet wordt bereikt, wordt een extra instantie van de locatie gestart. In dit component worden alle locaties, monsters, voorwerpen en world events gehost. De game world heeft in beginsel een controlerende taak: de game client geeft aan een actie te hebben uitgevoerd en de game world controleert of deze actie mocht worden uitgevoerd. Wanneer dit niet het geval is, wordt de actie op de game client teruggedraaid. Wanneer de actie wordt goedgekeurd, wordt deze gecommuniceerd naar alle verbonden clients.

### Character management
Binnen de character management worden alle spelergegevens, zoals het huidige HP, level en class, verwerkt en (indien nodig) opgeslagen. Ook handelt dit component alle zaken met betrekking tot de inventaris en het huidige in-game saldo van de karakters af.

### Social services
Het social services component wordt gebruikt voor alle sociale interacties tussen spelers, zoals chatten en het opzetten van parties of guilds.

### Load and distribute management
Dit component is verantwoordelijk voor het verdelen van de spelers over de servers. Wanneer een speler een locatie in de spelwereld wil betreden, wordt deze door het load and distribute management aan de juiste server verbonden. Het component zet dus een directe verbinding op tussen de game client en game world. De component is ook altijd op de hoogte van met welke server iedere speler verbonden is, en op welke locatie dit is.

Daarnaast is dit component verantwoordelijk voor het op- en afschalen van de servercapaciteit, afhankelijk van de drukte op dat moment.

Verder is de load and distribute management ook verantwoordelijk voor het verzorgen van sociale interacties tussen verschillende servers (zoals een chatbericht van de ene naar de andere server).

### Coordinator
Omdat de game client een beveiligingsrisico vormt (spelers kunnen namelijk lokaal data manipuleren en deze naar de server doorgeven), staat deze in nauw contact met coordinator. De coordinator zorgt ervoor dat alle verzoeken vanuit de client (die niet te maken hebben met de gamewereld of sociale berichten) bij het juiste component terecht komen. De component wordt voornamelijk gebruikt om communicatie met de market en de game store centraal te laten verlopen.

### Market
De market is het component waarbinnen de in-game market wordt afgehandeld. Hierbinnen kunnen karakters voorwerpen te koop aanbieden of kopen van andere spelers. De market is ook toegankelijk via een web UI.

### Game store
In de game store kunnen door spelers cosmetics (skins) worden gekocht. Deze worden gekocht met echte valuta, zoals de euro of dollar. De gekochte cosmetics geven spelers nooit een in-game voordeel. Er is dus geen pay-to-win. Net als de market is de game store ook via een web UI toegankelijk.

### Payment
Het payment component handelt alle transacties voor echte valuta (zoals de euro of dollar) af via een derde partij. Dit component is dus extern.

## 2.2 Interfaces en verbindingen

### Authentication
Deze interface geeft de mogelijkheid tot het inloggen en registreren. Deze verbinding is synchroon voor het inloggen en voor het opvragen van de accountgegevens is de communicatie asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **Login()** | Username en Password | Sessie ID | Inloggen van een gebruiker. |
| **ResetUsername()** | Email | bool | Het resetten van een gebruikersnaam. |
| **ResetPassword()** | Email | bool | Het resetten van een wachtwoord. |

### Session validation

Deze interface wordt gebruikt om te valideren of een sessie geldig is. De verbinding is asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **ValidateSession()** | SessionID | bool | Valideert of een sessie geautoriseerd is. |

### Connecting to server

Deze interface zorgt ervoor dat de direct connection tussen de game client en game world wordt opgezet. Deze verbinding is asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **TravelToLocation()** | SessionID, Location, CharacterID, ActiveParty? | Server adres | Het verbinden van een speler met een locatie. |

### Direct connection

Deze verbinding is de directe verbinding tussen de game client en game server. Over deze verbinding worden alle uitgevoerde acties van de speler gestuurd door de game client en worden vanuit de game server alle acties van de andere spelers (binnen dezelfde locatie) verstuurd. Deze communicatie is event-based. Een actie op de client triggert een event, wat door de game world wordt afgehandeld. Andersom kan de game world een event triggeren (zoals een monster wat beweegt), wat door alle verbonden game client (binnen dezelfde locatie) wordt verwerkt. De communicatie is asynchroon, om lag te voorkomen. Bepaalde acties hebben een hogere prioriteit dan andere acties. Dat wil zeggen: bepaalde acties moeten eerder worden afgehandeld dan andere acties. Iedere actie heeft dus een bepaalde (gehardcode) prioriteit. Een voorbeeld hiervan is dat bewegen een lagere prioriteit heeft dan vechten met andere spelers.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **DealDamage()** | SessionID, CharacterID, Target, Damage | bool | Het toebrengen van schade. |
| **UseAbility()** | SessionID, CharacterID, Abillity, Target? | bool | Het gebruiken van een ability. |
| **PickUpItem()** | SessionID, CharacterID, DroppedItemID | bool | Het oppakken van een voorwerp. |
| **Move()** | SessionID, CharacterID | bool | Bewegen in de spelwereld. |

| | | | |
|---|---|---|---|
| **InteractWithNPC()** | SessionID, CharacterID, NPCID | bool | Interacties met een NPC uitvoeren. |
| **InteractQuest()** | SessionID, CharacterID, QuestID, InteractionType | bool | Een interactie uitvoeren met een quest. |
| **UpdateChar()** | SessionID, CharacterID | Updated character variables, values | Het updaten van de waarden van de character in de game client o.b.v. de waarden van de character in de game world. |
| **SendMessage()** | SessionID, Message, CharacterID, Type, GroupID? | bool | Het versturen van een bericht. Het type geeft aan of het een DM, party, guild, local of global bericht is. GroupID geeft het ID van de party/guild/ locatie/vriend aan. |
| **RedirectMessage()** | SessionID, Message, CharacterID, Type, ReceiverID | bool | Een verwerkt chatbericht doorsturen naar de ontvanger(s). |
| **PartyInvite()** | SessionID, PartyID, CharacterID | bool | Het versturen van een uitnodiging voor een party. |
| **AcceptPartyInvite()** | SessionID, PartyID, CharacterID | bool | Het accepteren van een uitnodiging voor een party. |
| **LeaveParty()** | SessionID, PartyID, CharacterID | bool | Het verlaten van een party. |
| **GuildInvite()** | SessionID, GuildID, CharacterID | bool | Het versturen van een uitnodiging voor een guild. |
| **AcceptPartyGuild()** | SessionID, GuildID, CharacterID | bool | Het accepteren van een uitnodiging voor een guild. |
| **LeaveGuild()** | SessionID, GuildID, CharacterID | bool | Het verlaten van een guild. |
| **FriendInvite()** | SessionID, FriendshipID, CharacterID | bool | Het versturen van een vriendschaps-verzoek. |
| **RemoveFriend()** | SessionID, FriendshipID, CharacterID | bool | Het verwijderen van een vriend. |

### Server allocation

Deze verbinding zorgt voor het op- en afschalen van de capaciteit van de servers. Zodra een in-game locatie de ingestelde limiet heeft bereikt, wordt via deze verbinding ervoor gezorgd dat een nieuwe instantie van deze locatie wordt gestart. De communicatie in synchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **HostNewLocation()** | Location | bool | Het hosten van een nieuwe locatie. |

### Character information and inventory changes

Via deze interface biedt character management de informatie over karakters en hun inventaris aan. Tevens kunnen via deze interface wijzigingen, zoals een nieuw voorwerp in de inventaris, worden verwerkt. De communicatie is asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **AddItem()** | SessionID, CharacterID, ItemID, Quantity | bool | Het toevoegen van een voorwerp aan de inventaris. |
| **RemoveItem()** | SessionID, CharacterID, ItemID, Quantity | bool | Het verwijderen van een voorwerp uit de inventaris. |
| **AddCurrency()** | SessionID, CharacterID, CurrencyType, Quantity | bool | Het toevoegen van in-game valuta aan een character. |
| **RemoveCurrency()** | SessionID, CharacterID, CurrencyType, Quantity | bool | Het verminderen van in-game valuta van een character. |
| **GetCharInfo()** | SessionID, CharacterID | Character data | Het ophalen van informatie over een character. |
| **UpdateCharInfo()** | SessionID, CharacterID, Property, Value | bool | Het bijwerken van een eigenschap van een character. |

### Messages and invites

Omdat het mogelijk is dat chat berichten verstuurd worden door iemand op server A en worden ontvangen door iemand op server B, wordt social services via deze interface verbonden met load and distribute management. Alle chatberichten, invites en andere sociale interacties gaan dus via deze interface. De communicatie is asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **SendPartyMessage()** | Message, PartyID, CharacterID | bool | Het versturen van een bericht in een party. |

| | | | |
|---|---|---|---|
| **SendGuildMessage()** | Message, GuildID, CharacterID | bool | Het versturen van een bericht in een guild. |
| **SendFriendMessage()** | Message, FriendshipID, CharacterID | bool | Het versturen van een DM. |
| **SendLocalMessage()** | Message, LocationID, CharacterID | bool | Het versturen van een bericht in een locatie. |
| **SendGlobalMessage()** | Message, CharacterID | bool | Het versturen van een global bericht. |
| **RedirectMessage()** | Message, CharacterID, Type, ReceiverID | bool | Het verwerkte bericht doorsturen naar de ontvanger(s). |
| **PartyInvite()** | PartyID, CharacterID | bool | Het versturen van een uitnodiging voor een party. |
| **AcceptPartyInvite()** | PartyID, CharacterID | bool | Het accepteren van een uitnodiging voor een party. |
| **LeaveParty()** | PartyID, CharacterID | bool | Het verlaten van een party. |
| **GuildInvite()** | GuildID, CharacterID | bool | Het versturen van een uitnodiging voor een guild. |
| **AcceptPartyGuild()** | GuildID, CharacterID | bool | Het accepteren van een uitnodiging voor een guild. |
| **LeaveGuild()** | GuildID, CharacterID | bool | Het verlaten van een guild. |

### Social coordination

Omdat de sociale interacties via de load and distribute management gaan, wordt middels deze interface de functionaliteit aangeboden om deze acties naar andere spelers te communiceren. De communicatie is asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **SendMessage()** | Message, CharacterID, Type, GroupID? | bool | Het versturen van een bericht. Het type geeft aan of het een DM, party, guild, local of global bericht is. GroupID geeft het ID van de party/guild/ |

| | | | |
|---|---|---|---|
| | | | locatie/vriend aan. |
| **PartyInvite()** | PartyID, CharacterID | bool | Het versturen van een uitnodiging voor een party. |
| **AcceptPartyInvite()** | PartyID, CharacterID | bool | Het accepteren van een uitnodiging voor een party. |
| **LeaveParty()** | PartyID, CharacterID | bool | Het verlaten van een party. |
| **GuildInvite()** | GuildID, CharacterID | bool | Het versturen van een uitnodiging voor een guild. |
| **AcceptPartyGuild()** | GuildID, CharacterID | bool | Het accepteren van een uitnodiging voor een guild. |
| **LeaveGuild()** | GuildID, CharacterID | bool | Het verlaten van een guild. |
| **FriendInvite()** | FriendshipID, CharacterID | bool | Het versturen van een vriendschaps-verzoek. |
| **RemoveFriend()** | FriendshipID, CharacterID | bool | Het verwijderen van een vriend. |
| **RedirectMessage()** | Message, CharacterID, Type, ReceiverID | bool | Een verwerkt chatbericht doorsturen naar de ontvanger(s). |

### Buy and sell items

Binnen de market kunnen voorwerpen gekocht en verkocht worden. Deze wijzigingen worden via deze interface aan de coordinator doorgegeven. Tevens zorgt deze interface ervoor dat de market weet welke spullen een karakter in zijn inventaris heeft (en dus kan verkopen). Om te voorkomen dat een voorwerp twee keer kan worden gekocht, is de communicatie synchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **BuyItem()** | Buyer, Seller, ItemID, Quantity, Price | bool | Het kopen van een voorwerp op de market. |
| **SetItemForSale()** | CharacterID, ItemID, Quantity, Price | bool | Het te koop aanbieden van een voorwerp op de market. |
| **RemoveSale()** | CharacterID, OfferID | bool | Het annuleren van een verkoop op de market. |
| **GetMarketData()** | Region | Market data | Het ophalen van data van de market van een |

| | | | |
|---|---|---|---|
| | | | bepaalde regio (zoals Europa). |

### Cosmetics

Via deze interface kan (door de coordinator) worden gecommuniceerd met character management, om zo aangekochte cosmetics toe te kunnen voegen aan de inventaris. De communicatie is asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **BuyCosmetic()** | CharacterID, CosmeticID | bool | Het kopen van een cosmetic in de store. |
| **GetGameStoreData()** | - | Game store data | Het ophalen van data van de game store. |

### Character information

Tijdens het verbinden met de game world door de game client, kan een karakter worden gekozen, waarmee de speler wil spelen. Via deze interface geeft de coordinator informatie over de karakters van de speler, die vanuit character management worden opgehaald. De communicatie is asynchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **SendCharData()** | CharacterID | Character data | Het ophalen van informatie over een character. |
| **SendMarketData()** | Region | Market data | Het ophalen van data van de market van een bepaalde regio (zoals Europa). |
| **SendGameStoreData()** | - | Game store data | Het ophalen van data van de game store. |
| **BuyCosmetic()** | CharacterID, CosmeticID | bool | Het kopen van een cosmetic in de store. |
| **BuyItem()** | Buyer, Seller, ItemID, Quantity, Price | bool | Het kopen van een voorwerp op de market. |
| **SetItemForSale()** | CharacterID, ItemID, Quantity, Price | bool | Het te koop aanbieden van een voorwerp op de market. |
| **RemoveSale()** | CharacterID, OfferID | bool | Het annuleren van een verkoop op de market. |

### Transactions

Via deze interface worden de betalingen voor de game store afgehandeld. De communicatie is synchroon.

| Functie | Parameters | Return-waarde | Beschrijving |
|---|---|---|---|
| **HandlePayment()** | Currency, Amount | bool | Het afhandelen van een betaling. |

### Web UI

De web UI is een user interface, waar via de game store en market toegankelijk zijn.

## 2.3 Gevolgen van wijzigingen

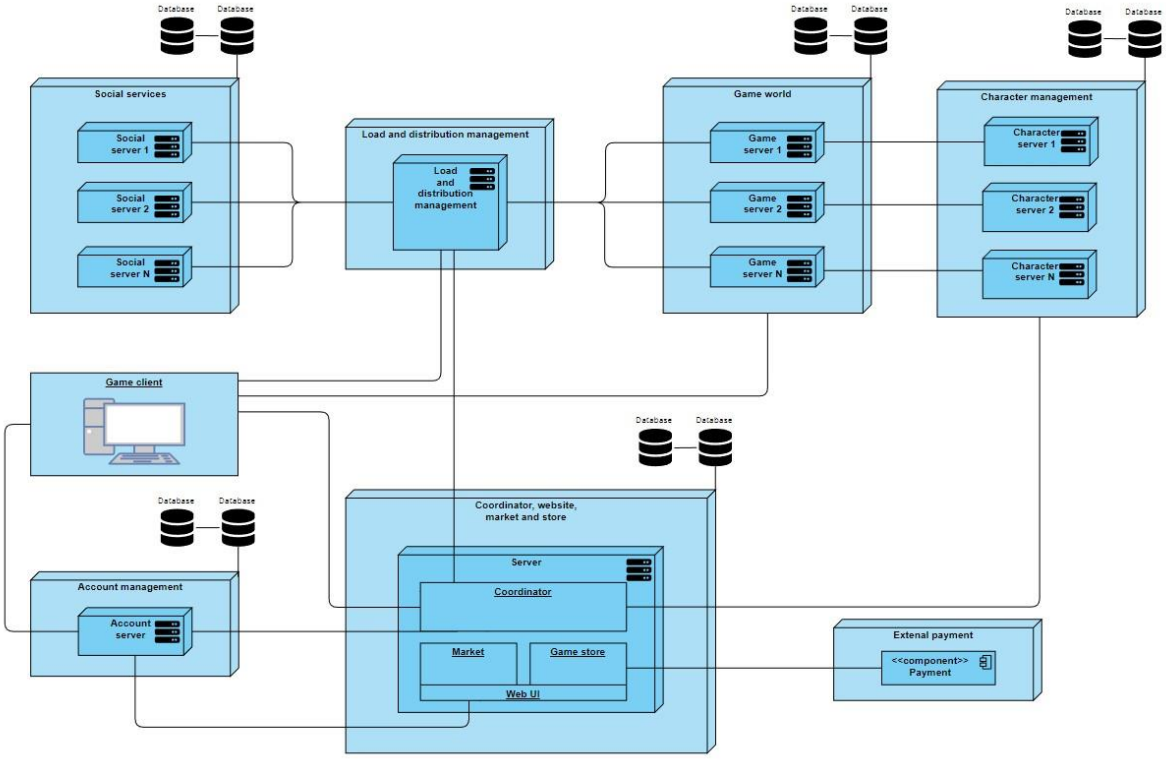
### Vervangen van een component

Binnen deze architectuur kan elk component vervangen worden zonder wijzigingen toe te passen bij andere componenten, zolang de interfaces tussen de componenten constant blijven. Wanneer de interfaces wel veranderen, zullen er wijzigingen moeten worden aangebracht aan de componenten die gebruik maken van deze interfaces.

### Toevoegen van een component

Wanneer een component toegevoegd wordt binnen de architectuur moeten nieuwe interfaces gemaakt worden, waarmee het nieuwe component communiceert met andere componenten. Bestaande interfaces moeten eventueel omgeleid worden naar het nieuwe component. Alle componenten die een nieuwe interface hebben met het nieuwe component moeten aangepast worden, om met deze interface te werken.

# 4. Physical view




*Figuur 3 Deployment diagram*

### Social services

Social services bestaat uit een n aantal servers om bottlenecking te voorkomen. Het moet niet mogelijk zijn om de social servers te overbelasten waardoor communicatie tussen spelers beperkt

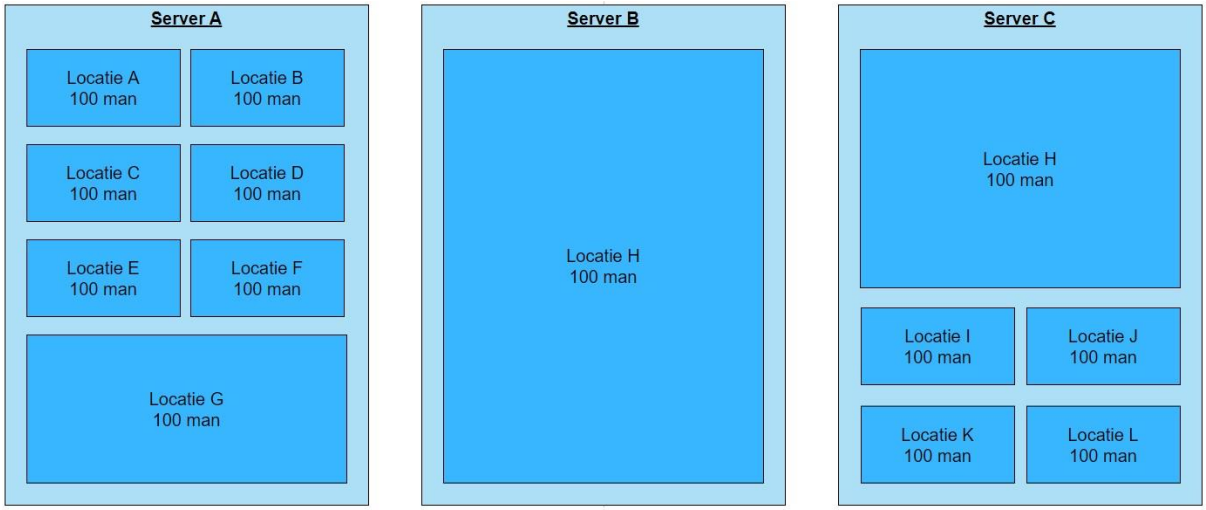
wordt. De servers zijn verbonden aan een database die een log bijhoudt van alle communicatie. Deze data bestaan uit tekstberichten en guild informatie. Tekstberichten worden 2 uur bewaard. De database is verbonden aan een extra database om zo voor redundantie te zorgen.

### Load and distribute management

De load and distribute management bestaat uit één server. Deze server heeft geen database nodig, omdat deze geen data opslaat.

### Game world

Game world bestaat uit een n aantal game servers. Deze servers hosten de verschillende locaties van de game wereld. De locaties en de karakters daarbinnen worden geregeld door de load and distribute management server. Iedere instantie van een locatie wordt afzonderlijk door één server afgehandeld. Het is dus niet mogelijk dat meerdere servers acties afhandelen voor dezelfde instantie van een locatie. Wel is het mogelijk dat een locatie meerdere keren wordt gehost op verschillende servers. Het is mogelijk dat een server meerdere locaties tegelijkertijd host. Zie voor een visuele weergave Figuur 4.




*Figuur 4 Locaties per server*

De game servers zijn verbonden aan een database. Aan deze database zit nog een database die voor redundantie zorgt.

### Character management

Character management bestaat ook uit een n aantal character servers. Deze servers zorgen er samen voor dat de druk kan worden verdeeld. Ieder game server staat in verbinding met een character server.

De character servers zitten verbonden aan een database. Deze database is verbonden met een extra database. Dit zorgt voor redundantie.

### Game client

De game client wordt lokaal door de speler gedraaid en staat dus niet op een server. Dit maakt de game client wel kwetsbaar. Daarom gaat informatie nooit door de client heen en bevat deze ook niet ‘de waarheid’. Dat wil zeggen: als de server iets anders zegt als de client, heeft de server gelijk.

### Account management

Account management bestaat uit een enkele server, omdat gebruikers alleen op het begin hoeven in te loggen en deze server daarom niet snel overbelast zal raken.

De servers zijn verbonden met een database. Deze is tevens verbonden met een extra database voor redundantie.

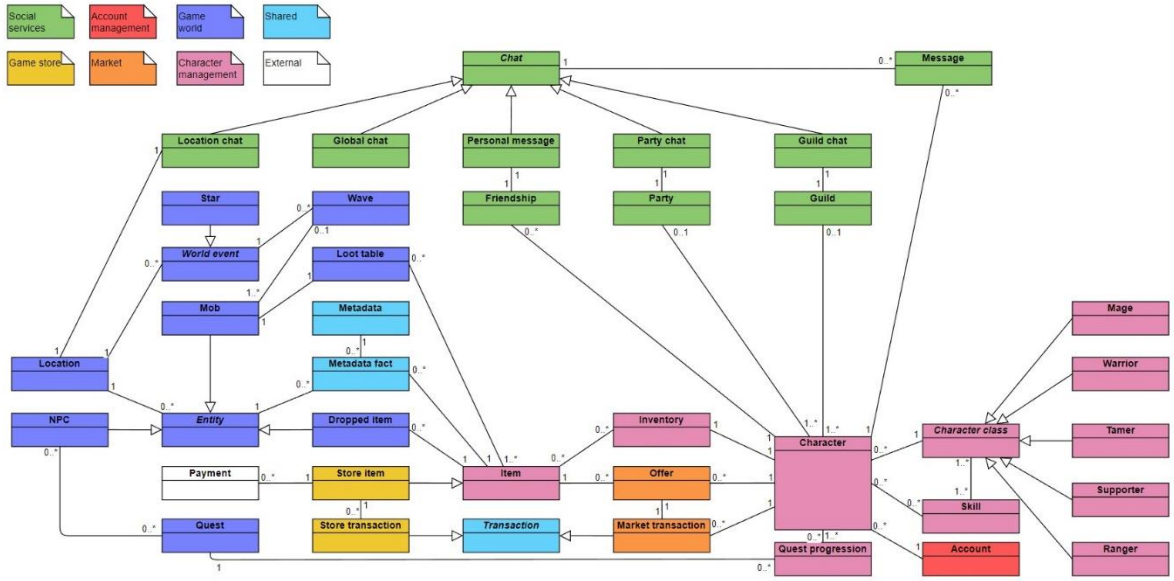
### Coordinator, website, market and game store

Deze server bestaat uit vier onderdelen, namelijk coordinator, website, market en game store. De game store en market draaien beiden op dezelfde server. Via deze server zijn beide ook toegankelijk middels een web UI. Om communicatie centraal te laten verlopen, is er een coordinator die in verbinding staat met de game client en character management.

### External payment

Betalingen worden door een externe provider afgehandeld.

## 5. Logical View




*Figuur 5 Conceptueel klassendiagram*

### Account

Class voor het account van de speler. Een account kan meerdere karakters hebben

### Character

Class voor het karakter van de speler. Deze class bevat alle gegevens van het karakter.

### Inventory

Class met de lijst van de items die het karakter bezit.

### Character class

Abstracte class voor character classes van de Character waarvan Mage, Warrior, Tamer, Supporter en Ranger overerven. Classen geeft de karakter verschillende stats en toegang tot bepaalde skills.

### Skill

Class voor skills te maken.

### Friendship
Class voor het vaststellen van een vriendschap tussen twee karakters.

### Party
Class voor een party.

### Guild
Class voor een guild.

### Chat
Abstracte class voor de chat in de game wereld waarvan Location chat, Gobal chat, Personal message, Party chat en Guild chat overerven. Deze class bevat de lijst van Messages an de chat.

### Message
Class voor de berichten in de chat. Deze class bevat het bericht en de eigenaar van het bericht.

### Location chat, Gobal chat, Personal message, Party chat en Guild chat
Deze classes erven over van de generieke Chat class en bevat logica specifiek voor de use case van de corresponderende class.

### Item
Class voor de items in het spel. Deze class bevat de data van alle verschillende items in het spel.

### Quest progression
Class voor het bijhouden welke karakters hoe ver zijn met welke quests.

### Quest
Class voor de quests die de speler kan doen. Quests kunnen gebonden zijn aan NPC's.

### World event
Abstracte class voor grote events binnen de game wereld waarvan Star overerft. Deze class bevat de logica van een event die in een locatie kan plaats vinden. Een World event heeft eventuele Waves van vijanden.

### Star
Deze class wordt gebruikt voor het Star event die op een locatie kan plaats vinden.

### Wave
Class om een wave van vijanden te maken. Deze class bevat welke vijanden en hoeveel daarvan in een wave zitten.

### Entity
Abstracte classen voor alle entiteiten in de game waaronder NPC , Mob en Dropped item die overerven van deze class. Deze class bevat algemene data die elke entiteit nodig heeft.

### Dropped item
Class voor gedropte items. Gedropte items zijn instanties van de items in de spelwereld.

### NPC
Class voor de NPC's binnen het spel. Deze class bevat de logica van de NPC.

### Mob
Class voor bevechtbare wezens binnen het spel. Deze class bevat de logica van een wezen.

### Loot table
Deze class bevat de items en de kans van die items die een Mob kan droppen.

### Location
Class voor locaties in het spel.

### Metadata en Metadata fact
Entities en Items kunnen afhankelijk van het specifieke Item of de specifieke Entity andere eigenschappen hebben. Hiervoor wordt gebruik gemaakt van Metadata in combinatie met een Metadata fact. In de Metadata wordt de eigenschap vastgesteld en in de Metadata fact wordt deze gekoppeld aan het betreffende Item/Entity.

Voorbeeld: er is een Item voor een zwaard. Het zwaard doet een aantal punten damage. Niet ieder Item doet damage, dus wordt er gebruik gemaakt van Metadata. In Metadata wordt een property gemaakt voor attack damage. In Metadata fact wordt de attack damage gekoppeld aan het zwaard.

### Store item
Deze class wordt gebruikt voor Items die worden verkocht in de Game store. Dit zijn items die met echte valuta (zoals euro of dollar) kunnen worden gekocht.

### Offer
Deze class wordt gebruikt voor Items op de in-game market. Karakters kunnen hiermee spullen kopen en verkopen van elkaar met in-game valuta.

### Transaction
Abstracte class voor transacties. Transacties loggen de (ver)koop van een item. Market transaction en Store transaction erven van deze klasse over.

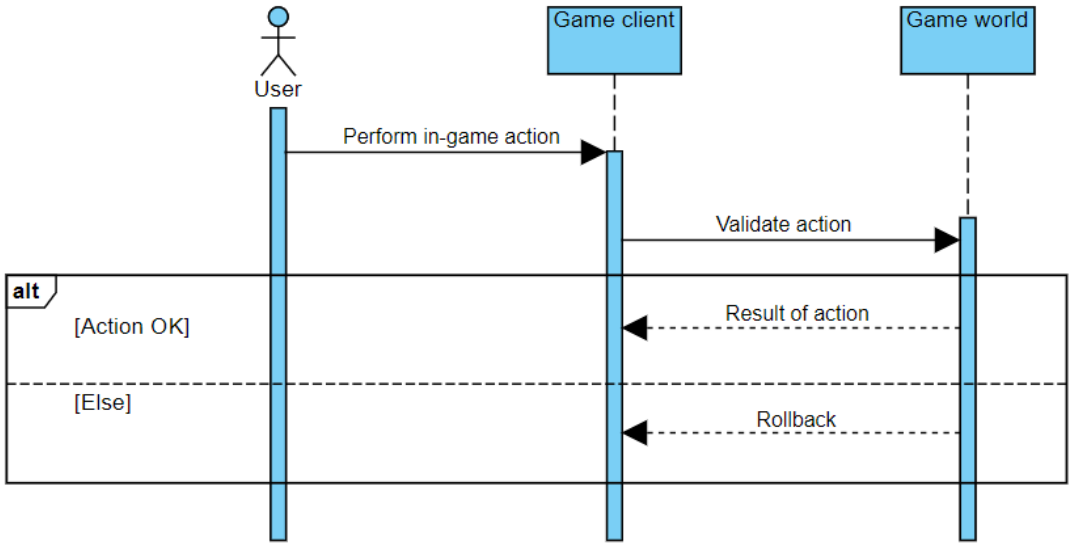
### Market transaction
Deze class logt een (ver)koop van een Item op de in-game market.

### Store transaction
Deze class logt een verkoop van een Item in de Game store.

### Payment
Class voor betalingen met echte valuta, zoals de euro of dollar.

# 6. Process view

## 5.1 Uitvoeren van acties




*Figuur 6 Sequentiediagram voor het uitvoeren van acties*

Communicatie tussen de Game client en Game world werkt event-driven. Wanneer een gebruiker in het spel een actie uitvoert, wordt deze actie client-side uitgevoerd (inclusief alle bijbehorende berekeningen). Vervolgens wordt de actie gevalideerd door de Game world. Dat wil zeggen: de Game world controleert of de actie kon en mocht worden uitgevoerd. Indien de actie akkoord is, wordt het resultaat van deze actie naar alle Game clients gecommuniceerd (dus niet alleen de Game client vanaf waar de actie werd uitgevoerd). Indien de actie niet akkoord is, wordt een rollback uitgevoerd binnen de Game client die de actie uitvoerde.

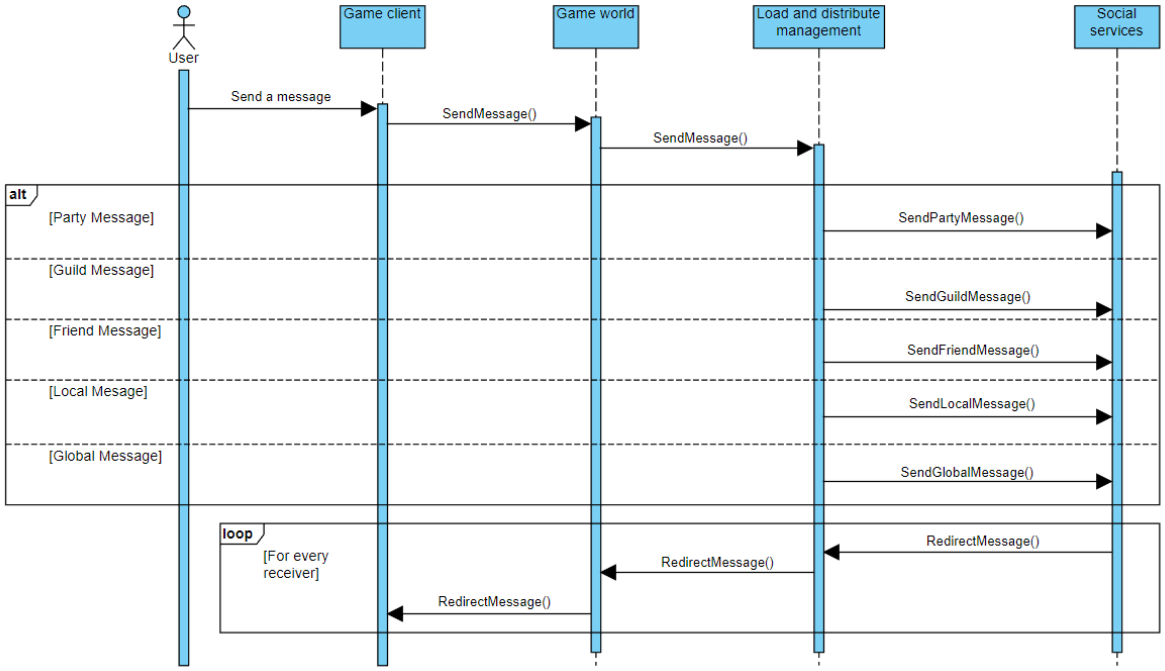
De Game world heeft dus een controlerende taak en spreekt altijd ‘de waarheid’.

Voorbeeld: een gebruiker slaat met een zwaard een zombie. De Game client geeft dit visueel weer ene berekent de schade die aan de zombie wordt toegebracht. De actie met bijbehorende resultaat worden naar de Game world gestuurd voor validatie. De Game world controleert of de actie en het resultaat kloppen. Deze kloppen beiden en de actie wordt gecommuniceerd naar de andere Game clients.

Er geldt een uitzondering op bovenstaand proces, wanneer er een random-factor in een actie zit. Een voorbeeld hiervan is de kans dat een wapen extra damage doet. Acties waarbij deze random-factor aanwezig is worden op een andere manier afgehandeld, vanwege het risico op cheaten, door clients die de kans op deze extra damage verhogen. In deze situaties wordt de actie niet alleen gevalideerd door Game world, maar wordt ook het resultaat op de Game world bepaald. De Game world heeft in deze gevallen dus niet alleen een controlerende taak. Omdat het niet wenselijk is dat gebruikers lang moeten wachten op antwoord van de Game world, worden deze acties met extra prioriteit uitgevoerd door de Game world.

Een alternatieve oplossing voor het oplossen van acties met een random-factor, is het toevoegen van een anti-cheat laag tussen de Game client en Game world. Hierbij wordt met behulp van AI gekeken of er verdachte acties zijn, waarbij een gebruiker bijvoorbeeld erg vaak 'geluk' zou hebben. Deze implementatie brengt echter performanceverlies met zich mee, wat niet wenselijk is, zeker aangezien deze acties met een random-factor vaak in gevechten zullen voorkomen, waarbij performance extra belangrijk is. Er is daarom niet voor deze oplossing gekozen.

## 5.2 Versturen van chatberichten




*Figuur 7 Sequentiediagram voor het versturen van chatberichten*

Wanneer een gebruiker een bericht verstuurd, wordt dit bericht via de Game client naar de Game world verstuurd. De Game world stuurt dit bericht door naar de Load and distribute manager. De Load and distribute manager controleert het type bericht en roept op basis hiervan de juiste functie aan in Social services. Social services verwerkt vervolgens het bericht en bepaald de ontvanger(s). Voor iedere ontvanger wordt vervolgens het bericht doorgestuurd naar Load and distribute management. Load and distribute management weet in welke Game world de ontvanger zit en stuurt het bericht naar deze Game world. De Game world stuurt het bericht vervolgens naar de Game client. Iedere functieaanroep ontvangt als antwoord *True* als het succesvol kon worden uitgevoerd of *False* als iets mis ging.

# 7. Server hosting

Voor server hosting van een MMORPG zijn twee opties: zelf hosten of de cloud hosting. Voorbeelden van cloud hosting zijn Microsoft Azure BRON of Amazon AWS BRON. Beiden hebben een aantal voor- en nadelen.

Een voordeel van de cloud is onderhoud. Door server hosting uit te besteden worden alle serveronderhoudswerkzaamheden uit handen gegeven (Andrés, 2021). Dit zorgt voor minder onderhoudskosten en tijdsbeslag. Daarnaast is de beveiliging van de cloud vaak goed geregeld en worden beveiligingsrisico's snel aangepakt (Andrés, 2021). Hetzelfde geldt voor updates, deze worden direct door de cloudleverancier geïnstalleerd wanneer nodig (Andrés, 2021). Een ander

belangrijk voordeel van de cloud is schaalbaarheid (CodeFirst, 2016). Waar bij zelf hosten de beschikbare hardware een beperkende factor is, kan via cloud opgeschaald worden wanneer dit nodig is. Betrouwbaarheid is ook stukken hoger bij cloud hosting, wanneer vergeleken met zelf hosten (CodeFirst, 2016). Als in de cloud een server uit de lucht gaat, kan snel een nieuwe instantie worden gestart. Wanneer zelf wordt gehost moet eerst het probleem worden verholpen voordat de hardware weer beschikbaar is.

Een nadeel van de cloud is dat het ten kosten gaat van flexibiliteit (Andrés, 2021). De mogelijkheden worden beperkt tot wat de leverancier aan mogelijkheden heeft en aanbied. Een ander nadeel is dat dat je zelf geen eigenaar bent van de data (Andrés, 2021). Dit betekent dat bij verlies van toegang tot het platform, de data (tijdelijk) verloren is.

Zelf hosten geeft meer flexibiliteit (Andrés, 2021). Waar de cloud beperkt tot de mogelijkheden die worden aangeboden, zijn de mogelijkheden bij zelf hosten eindeloos (Andrés, 2021).

Een nadeel van zelf hosten is dat er veel tijd en geld in onderhoud gaat zitten (Andrés, 2021). Hierbij ontstaat tevens het risico dat bij een gebrek aan expertise onnodige risico's ontstaan op het gebied van (onder andere) beveiliging en dataverlies door bijvoorbeeld een tekort aan goede back-ups (Andrés, 2021). Ook is, zoals eerder genoemd, schaalbaarheid bij zelf hosten een probleem (CodeFirst, 2016). Het is niet mogelijk verder op te schalen als alle servers in gebruik zijn. Dit geldt ook andersom: als er weinig resources nodig zijn, zijn er wel nog steeds onderhoudskosten voor de ongebruikte servers.

Een laatste grote factor die van invloed kan zijn op de keuze zijn de kosten. Het is echter lastig inschatten welk van de twee goedkoper is, zowel in opstartkosten als in kosten op de lange termijn. Dit komt door een groot aantal factoren, waaronder de benodigde rekenkracht, onderhoudskosten, eventuele extra personeelskosten, enzovoorts (Rędzioch, 2018).

Omdat het inschatten van de kosten voor beide opties lastig is, moet vooral worden gekeken naar de verschillende voor- en nadelen. Gekeken naar de hierboven benoemde punten, valt de keus voor een MMORPG voor de cloud. Hierbij is vooral de schaalbaarheid de grootste invloedsfactor. Daarnaast is de inlevering van flexibiliteit niet dermate beperkend dat dit problemen oplevert.

# 8. Verwijzingen


Andrés, R. (2021, maart 26). *Cloud vs. Self-hosting: which one to choose?* Retrieved from Angry Ventures Blog : https://angry.ventures/blog/cloud-vs-self-hosting-which-one-to-choose/

CodeFirst. (2016, juli 21). *Self-Hosted or Cloud Hosted? Pros and Cons*. Retrieved from CodeFirst: https://www.codefirst.co.uk/blog/self-host-or-cloud-host/

Rędzioch, S. (2018, oktober 5). *TCO: is self-hosting really cheaper?* Retrieved from LinkedIn: https://www.linkedin.com/pulse/tco-self-hosting-really-cheaper-szczepan-r%C4%99dzioch/