Virtuele beurzen

Onderzoek naar het maken van een platform voor virtuele beurzen



Scriptie
03-05-2021 tot 04-02-2022

Naam:	W.G.R. Beunk
Studentnummer:	408323
Opleiding:	HBO-ICT
Onderwijsinstelling:	Saxion
Bedrijf:	Virtual Builders
Begeleider:	Arne Lubbers
Technische begeleider:	Wouter Versloot
Begeleider school:	Erik van der Arend

Virtuele beurzen

Onderzoek naar het maken van een platform voor virtuele beurzen

Versies

Versienummer	Datum	Veranderingen
V0.1	19 mei 2021	Eerste opmaak van het verslag
V0.2	7 juli 2021	Deelvragen 1 t/m 3 beantwoord, begonnen met deelvraag 4. Interviews toegevoegd. Gebruikte technieken toegelicht.
V0.3	6 september 2021	Feedback verwerkt, onderzoek afgerond, conclusie toegevoegd, use case toegevoegd, functioneel ontwerp use case afgerond, wireframes gemaakt, testplan gestart, API ontwerp gestart
V0.4	3 oktober 2021	Feedback verwerkt, architectuurplaat toegevoegd, database ontwerp toegevoegd, requirements toegevoegd
V0.5	15 november 2021	Implementatie toegevoegd, feedback verwerkt
V0.6	8 december 2021	Flow Diagrams toegevoegd, feedback verwerkt, bronnen toegevoegd
v0.7	16 januari 2022	Feedback verwerkt, implementatie Docker toegevoegd, implementatie videobel api verwerkt, reflectie toegevoegd, aanbevelingen toegevoegd

Extra informatie

Er zijn twee begeleiders vanuit het bedrijf. De eerste begeleider is Arne Lubbers en hij is tevens de opdrachtgever. Arne Lubbers is de eigenaar van het bedrijf Virtual Builders, het bedrijf waar de afstudeeropdracht wordt gelopen. Arne biedt voornamelijk ondersteuning bij het controleren van het werk en de documenten die worden gemaakt. Arne is te bereiken via de mail: arne@virtualbuilders.nl.

De tweede begeleider vanuit het bedrijf is Wouter Versloot. Wouter ondersteunt vooral bij het technische aspect van de opdracht. Zo kan Wouter advies geven en helpen bij eventuele problemen die kunnen ontstaan tijden het programmeren. Wouter is te bereiken via de mail: wouter@virtualbuilders.nl.

Vanuit school is Erik van der Arend de begeleider. Erik houdt zich voornamelijk bezig met het controleren van de student. Dit houdt in dat Erik kijkt of de student nog wel op schema ligt en

controleert de documentatie die wordt geschreven. Erik is te bereiken via de mail: h.a.vanderarend@saxion.nl.

De opdracht wordt uitgevoerd door Rutger Beunk (studentnummer 408323). Rutger studeert HBO-ICT uitstroomprofiel software engineer te hogeschool Saxion in Enschede. De student voert de opdracht uit bij het bedrijf Virtual Builders in Rijssen. Rutger is te bereiken via de e-mailadressen: 408323@student.saxion.nl en rutger_beunk@hotmail.com

Voorwoord

Tijdens deze afstudeerperiode heb ik mogen werken aan een interessante opdracht. Door deze opdracht heb ik nieuwe inzichten gekregen en veel geleerd over de manier van denken van de klant. Hierbij wil ik mijn dank uitbrengen aan Arné voor deze mogelijkheid. In deze gekke tijd was het lastig om een plek te vinden en ik ben dankbaar dat hij voor mij een plekje had. Daarnaast wil ik Erik van der Arend ook bedanken voor de begeleiding vanuit school en het helpen meedenken.

Samenvatting

In de corona-periode zijn veel evenementen niet doorgegaan. Dit geldt ook voor de beurzensector. Aan de student is gevraagd om te onderzoeken hoe een online beurs zou kunnen gaan werken en wat hiervoor nodig is. Hiervoor is het belangrijk om rekening te houden met de schaalbaarheid van het platform en de veiligheid.

Om hierachter te komen is eerst onderzoek gedaan. Voor het onderzoek is gebruikt gemaakt van verschillende methodes die samenvallen onder het DOT-framework. Onder dit framework valt het zoeken naar bestaande technieken die gebruikt kunnen worden en het zoeken naar nieuwe mogelijkheden om zo tot een goed ontwerp te kunnen komen. Ook is er gesproken met een aantal stakeholders om erachter te komen wat zij willen van het project.

Een van de vereisten van het project is dat er gewerkt wordt met een Unreal Engine applicatie. Het grote nadeel van deze applicatie is dat er sterke hardware nodig is om deze applicatie te kunnen draaien. Niet iedereen heeft deze hardware tot de beschikking en het is dus nodig om een platform te ontwerpen dat hiermee rekening houdt. Door gebruik te maken van een cloud omgeving is het niet nodig dat de gebruiker zelf de hardware bezit. Om ervoor te zorgen dat er niet te veel servers nodig zijn in de cloud kan er ook gebruikt gemaakt worden van Docker om zo meerdere instances van de applicatie op dezelfde server te kunnen draaien.

Uiteindelijk is er een ontwerp gekomen dat rekening houdt met de schaalbaarheid van het platform en dat het platform veilig is. Uit dit ontwerp is een proof of concept gekomen om de mogelijkheden te laten zien. Dit houdt in dat het geen product is dat klaar is voor productie en waarvoor nog aanpassingen nodig zijn.

Inhoudsopgave

Virtuele beurzen **2**

Versies **2**

Extra informatie **2**

Voorwoord **4**

Samenvatting **5**

Inhoudsopgave **6**

Inleiding **10**
- Aanleiding 10
- Leeswijzer 10

Achtergrond **11**
- Virtual Builders 11
- Organisatieomschrijving 11
- Missie en visie 11
- Kernactiviteiten 11
- Huidige situatie 11

Onderzoeksopzet **13**
- Onderzoeksvragen 13
- Gebruikte methodieken 13
- Deelvraag 1 14
- Deelvraag 2 14
- Deelvraag 3 14
- Deelvraag 4 15

Onderzoeksresultaten **16**
- Leeswijzer onderzoek 16
- Resultaten 16
  - Wat kunnen use cases zijn voor het platform? 16
  - Hoe moet de data beveiligd worden om het platform veilig te maken? 17
    - Data at rest 18
    - Data in transit 19
    - Data in use 19
    - Rollen en encryptie 20
  - Welke videobel API past het beste bij dit platform? 21
    - WebRTC 21
    - STUN server 21
    - TURN server 21
    - MCA (Multi Criteria Analyses) 22

Welke architectuur past het beste bij een platform met Unreal Engine? 25
Pixel Streaming 25
Unreal Engine with Docker 26
AWS 26
PureWeb 27
Architectuurplaat 28
Conclusies 28
Conclusie deelvraag 1: Wat kunnen use cases zijn voor het platform? 28
Conclusie deelvraag 2: Hoe moet de data beveiligd worden om het platform veilig te maken? 29
Conclusie deelvraag 3: Welke videobel API past het beste bij dit platform? 29
Conclusie deelvraag 4: Welke architectuur past het beste bij een platform met Unreal Engine? 29
Slotconclusie 30

Ontwerp 31
Functioneel ontwerp 31
Use case diagram 31
Requirements 31
Wireframes 32
Registreren 32
Inloggen 33
Beurs scherm 33
Overzicht bezoekers 34
Stand aanmaken/aanpassen 35
Product aanmaken/aanpassen 35
Beginscherm organisator 36
Testplan 36
Website pagina’s 36
API calls 37
Acceptatietest 37
Technische ontwerp 37
Architectuurplaat 37
Database 38
Flow Diagrams 39
Gebruiker inloggen 40
Registreren 40
Standhouder 41
Organisator 41
API ontwerp 41
Error codes 42

Implementatie 44

Unreal Engine 44
Database 45
Docker 46
Docker met Node.js en Mysql 46
Node.js 46
Mysql 46
Docker compose (Node.js en MySQL) 47
Videobellen 49
Mesibo 49
Videosdk 50
Virtuele beurs via API 50
API test 52

Conclusie & aanbevelingen 54
Authenticatie 54
Videobel applicatie 54
Database ontwerp 54

Literatuurlijst/Bibliografie 55

Bijlage 1: Interview met Leonie Pamboer 58

Bijlage 2: Interview met Arne 64

Bijlage 3: Reflectie 69
Planning 69
Wat was de situatie? 69
Wat was je taak? 69
Hoe heb je het aangepakt? 69
Heeft het gewerkt? 69
Wat heb je er van geleerd? 69
Requirements 70
Wat was de situatie? 70
Wat was je taak? 70
Hoe heb je het aangepakt? 70
Heeft het gewerkt? 70
Wat heb je ervan geleerd? 70
Communicatie met de leraar 70
Wat was de situatie? 70
Wat was je taak? 71
Hoe heb je het aangepakt? 71
Heeft het gewerkt? 71
Wat heb je ervan geleerd? 71
Vastzitten met een probleem 71
Wat was de situatie? 71

Wat was je taak? 71
Hoe heb je het aangepakt? 71
Heeft het gewerkt? 72
Wat heb je hiervan geleerd? 72

1 Inleiding

1.1 Aanleiding

Door de coronacrisis heeft het leven een tijd stilgelegen. Veel evenementen zijn niet doorgegaan en bedrijven hebben dicht gezeten. Op het moment van schrijven is de coronacrisis nog steeds niet voorbij. Ook de beurzensector heeft hier veel last van. Deze heeft in anderhalf jaar tijd geen beurzen meer kunnen houden. Hierdoor is deze sector op zoek gegaan naar een alternatieve oplossing waardoor er wel weer beurzen mogelijk kunnen zijn. Daardoor is deze opdracht ontstaan.

De beurzensector is op zoek gegaan naar een alternatieve oplossing. Een oplossingen waarnaar wordt gekeken is om een virtuele beurs online te houden. Dit houdt in dat bezoekers naar een website kunnen gaan om zo alsnog een beurs te kunnen bekijken. Deze virtuele beurs wordt gemaakt in Unreal Engine. Deze wordt gemaakt door een andere afstudeerder. Unreal engine is niet voor elke PC beschikbaar aangezien er vrij stevige hardware nodig is om die applicatie te kunnen draaien. Daarom wordt er onderzocht of het mogelijk is om hier een platform omheen te bouwen zodat het voor elke gebruiker toegankelijk wordt. In dit platform moet er rekening worden gehouden met schaalbaarheid en veiligheid van het platform.

1.2 Leeswijzer

Eerst wordt de achtergrond behandeld. In dit hoofdstuk wordt de achtergrond behandeld van het bedrijf. Hierbij wordt besproken wat het bedrijf doet en waar het bedrijf voor staat. Vervolgens wordt de onderzoeksopzet besproken. Hierin wordt besproken hoe het onderzoek is opgezet en welke technieken er zijn gebruikt. In het volgende hoofdstuk worden de resultaten besproken van het onderzoek. Hierin worden de deelvragen beantwoord en worden er conclusies getrokken. In het volgende hoofdstuk wordt het ontwerp van het platform besproken. Eerst wordt er gekeken naar de use case diagram, waaruit de requirements worden opgesteld. Vervolgens komen de wireframes voor de website pagina’s die laten zien hoe het platform eruit komt te zien. Als laatst in dit hoofdstuk wordt er gekeken naar hoe er getest wordt. Vervolgens wordt het technisch ontwerp behandeld. In dit hoofdstuk wordt de architectuurplaat, de database en hoe de api werkt, behandeld. In het hoofdstuk daarop wordt er gekeken naar de implementatie. Hoe zijn bepaalde onderdelen geïmplementeerd en hoe werken onderdelen met elkaar samen. In het laatste hoofdstuk worden conclusies en aanbevelingen gedaan. Het gaat hierbij vooral om wat er achteraf gezien beter kan en wat er beter anders gedaan kan worden.

2 Achtergrond

In dit hoofdstuk wordt de achtergrond van Virtual Builders nader toegelicht. Het gaat over wat het bedrijf doet, wat de kernactiviteiten zij en wat de status is van het product voor het onderzoek.

2.1 Virtual Builders

In dit hoofdstuk wordt er wat meer verteld over het bedrijf waar de stage wordt gelopen. Wat is de kernactiviteit van het bedrijf en wat doet het bedrijf nog meer. Daarnaast wordt er nog wat vertelt over de grootte van het bedrijf. Als laatst wordt de visie van het bedrijf en de huidige situatie van het bedrijf behandeld.

2.1.1 Organisatieomschrijving

De opdracht wordt uitgevoerd bij en voor Virtual Builders. Dit bedrijf heeft zich vooral gespecialiseerd in het bouwen van real-time 3D visualisaties en applicaties. Het is nog een relatief klein bedrijf dat is opgericht door Arne Lubbers en is gevestigd in Rijssen. Virtual Builders heeft 3.6 FTE en er werken op het moment van schrijven 3 stagiairs. De IT afdeling is 1.5 FTE groot.

2.1.2 Missie en visie

Het is vaak lastig in te beelden of een meubel er goed uit zal zien in de kamer en of het mooi past. Hiermee helpt Virtual Builders door deze modellen in een virtuele omgeving te plaatsen. Dit heeft als voordeel dat een consument makkelijker kan zien of dat meubel wel past in de kamer.

Daarnaast helpt Virtual Builders ook met het visualiseren van vastgoedpresentaties. Ook hierbij is het vaak lastig om in te beelden hoe bijvoorbeeld een huis er uit komt te zien. Hierbij helpt Virtual Builders door een 3D weergave te maken. Deze weergave is interactief en fotorealistisch. Dit helpt klanten met het visualiseren hoe het eindproduct eruit zal zien.

2.1.3 Kernactiviteiten

Meestal krijgen consumenten een 2D tekening voorgeschoteld en is het lastig om het te kunnen visualiseren hoe het er in het echt uit zal zien. Virtual Builders helpt met het omzetten van een 2D tekening naar een 3D fotorealistische weergave waar een klant doorheen kan lopen. Hierbij gaat het vaak om interieur gerelateerde zaken.

Ook is het bedrijf bezig met een nieuw product zodat consumenten via hun eigen computer hun eigen kamer kunnen namaken en zo kunnen zien hoe bijvoorbeeld een meubel er uit zal zien in die kamer. Dit doen ze door gebruik te maken van Unreal Engine. Ze maken hiermee een soort game met de componenten van klanten die ze in die game zetten. Zo kunnen bedrijven hun klanten dit product laten zien en krijgen de klanten een beter idee hoe hun meubel er in hun huis er uit ziet.

2.1.4 Huidige situatie

Op het moment van de start van het afstuderen ligt er een wensenlijst voor een virtuele beurs (zie bijlage 3). Deze lijst is opgesteld door virtual builders aan de hand van een proof of concept van een andere virtuele beurs. Deze beurs was nog niet in 3D en er waren wat technische problemen zoals het niet werken van de database. Aan de hand van deze wensenlijst is er een afstudeerstudent in de

richting van game development begonnen met een afstudeerproject om deze wensen om te zetten in een Unreal Engine game. Voor het begin van het project ligt er verder geen requirement vast, alleen dat de game applicatie via een webbrowser beschikbaar moet zijn. Daarnaast is het doel van het project om te kijken of dit mogelijk is en of het schaalbaar kan worden gemaakt. Aan de hand van het onderzoek moet er een MVP worden gemaakt zodat het bedrijf kan kijken of het voldoet aan de verwachtingen en hiermee eventueel verder gaat. Verder is er geen voorkeur aan welke taal het platform moet gebruiken en is dus open voor de student. Verder heeft het bedrijf zelf geen cloud omgeving.

3 Onderzoeksopzet

In dit hoofdstuk wordt gekeken naar de opzet van het onderzoek. Dit houdt in dat er eerst wordt besproken wat de hoofdvraag is en wat de deelvragen zijn. Vervolgens wordt uitgelegd welke methode is gebruikt om de informatie te verzamelen om de vraag te beantwoorden.

3.1 Onderzoeksvragen

De doelstelling van dit onderzoek is om te kijken hoe er een platform kan worden gemaakt van dit idee. Hoe kan dit schaalbaar en veilig worden opgezet. Deze oplossing is niet alleen bedoeld als tijdelijke oplossing maar er wordt ook gekeken of deze kan worden ingezet als vervanging of als verrijking van het huidige organiseren van beurzen.

Voor dit onderzoek is er een hoofdvraag opgesteld. Deze hoofdvraag is de belangrijkste vraag om te zien hoe deze opdracht tot een platform kan worden gemaakt. Daarnaast zijn er nog een aantal deelvragen opgesteld om te kijken hoe deze opdracht tot een platform kan worden gemaakt.

Hoofdvraag: **Hoe kan je een platform voor virtuele beurzen ontwikkelen met een Unreal Engine applicatie in een web omgeving?**
Deelvragen:

1. Wat kunnen use cases zijn voor het platform?
2. Welke data moet er beveiligd worden om het platform veilig te maken?
3. Er moet een videobellen integratie komen in het platform, vanuit het bedrijf is in eerste instantie gekozen voor Agora, zijn er hiervoor andere en misschien betere oplossingen?
4. Welke architectuur past het beste bij een platform met Unreal Engine?

Voor dit onderzoek wordt er gebuikt gemaakt van triangulatie. In dit geval houdt dat in dat er verschillende methodes worden gebruikt om zo de verschillende resultaten van het onderzoek te onderzoeken. (The DOT framework, 2021)

3.2 Gebruikte methodieken

Voor onderzoeken in de ICT wordt vaak gebruik gemaakt van het DOT-framework (Development Oriented Triangulation). In dit framework wordt er gebruik gemaakt van verschillende methodes om zo tot een antwoord te kunnen komen. Dit framework baseert zich op het onderzoek naar bestaand werk en werk wat nog zelf gemaakt moet worden. Het framework is opgedeeld in 5 verschillende strategieën, namelijk: biebstrategie, veldstrategie, werkplaatsstrategie, labstrategie en showroomstrategie. (The DOT framework, 2021)

De eerste strategie is de bieb strategie. Deze strategie wordt ook wel deskresearch genoemd. Dit houdt in dat er wordt gezocht naar bestaande oplossingen via het internet of andere documenten. Aan de hand van deze informatie wordt er gekeken of deze oplossingen kunnen bijdragen aan de oplossing van het vraagstuk.

De tweede strategie is de veldstrategie. Dit houdt in dat er aan mensen gevraagd wordt wat zij ervan vinden en hun input kunnen geven. Voor deze strategie zijn er een aantal technieken die kunnen worden gebruikt bij het onderzoek.

De derde strategie is de werkplaats strategie. Deze strategie houdt in dat er bijvoorbeeld een aantal kleine programma's worden gemaakt om zo de verschillende oplossingen met elkaar te vergelijken. Het gaat er bij deze strategie dus om dat er wat wordt gemaakt om zo te toetsen of het nuttig is voor het eindproduct of niet.

De vierde strategie is de labstrategie. Deze strategie houdt in dat er ideeën worden getest om zo te kijken of het gebruikt kan worden voor het product. Deze ideeën worden meestal getest bij gebruikers. Ook worden deze gebruikt om te toetsen of wat er geleerd is uit het onderzoek ook zo werkt als verwacht wordt dat het werkt.

Als laatste is er de showroomstrategie. In deze strategie wordt er gekeken of de ideeën die er zijn bedacht ook daadwerkelijk werken bij het bestaande werk. Dit wordt gedaan door bijvoorbeeld een prototype te laten zien aan een expert.

3.3 Deelvraag 1

Voor het beantwoorden van de eerste deelvraag is vooral gebruik gemaakt van de methode 'explore user requirements' (HBO-I, 2018). Deze methode houdt in dat er onderzocht wordt wat een gebruiker verwacht van een systeem. De gebruikers van een systeem kunnen verschillende stakeholders zijn die te maken hebben met het systeem of de eindgebruiker. Door in gesprek te gaan met deze gebruikers kan een goed beeld ontstaan hoe een systeem eruit zou moeten zien. De manier om hier data te verkrijgen wordt vooral gedaan door interviews. Aan de hand van deze interviews kunnen de requirements van het systeem worden opgesteld. Dit is vooral handig bij het beantwoorden van deelvraag 1, aangezien het platform, wat ontwikkeld moet worden, vrij nieuw is. Hierdoor is er nog weinig bekend over wat een gebruiker van het systeem verwacht.

3.4 Deelvraag 2

Voor het beantwoorden van de tweede deelvraag is vooral gebruik gemaakt van 'deskresearch' (HBO-I, 2018). Deskresearch houdt in dat er gezocht wordt naar bruikbare bronnen en met deze bronnen uiteindelijk de vraag te kunnen beantwoorden. Voor deze deelvraag is dit een goede methode omdat er veel te vinden is over het versleutelen van informatie. Daarnaast moet er ook rekening worden gehouden met de AVG. Deze informatie is vrij makkelijk te vinden via deze methode. De informatie wordt vooral verkregen via het internet.

3.5 Deelvraag 3

Bij deelvraag 3 is er een combinatie gebruikt van een aantal methodes. De eerste die is gebruikt is de biebstrategie (HBO-I, 2018). Er is een gedeelte op het internet opgezocht om de werking van het videobellen uit te lichten. Hiervoor is dus de 'deskresearch' gedaan. Om bestaande programma's te vinden is er een 'available product analyses' (HBO-I, 2018) gedaan. Hierbij is gekeken naar wat de meest bekende en gebruikte producten zijn voor het videobellen. Met deze producten is er een 'multi criteria analyses' gedaan. Hierin zijn de producten vergeleken met elkaar aan de hand van een paar criteria's van het product. De 'multi criteria analyses' hoort bij de werkplaatsstrategie.

3.6 Deelvraag 4

Bij deelvraag 4 is er gebruik gemaakt van deskresearch (HBO-I, 2018). Er is op internet gezocht naar geschikte manieren om te kijken hoe het product online kan worden gehost. Aan de hand daarvan is er een architectuurplaat gemaakt om te laten zien wat de beste manier is om een Unreal Engine applicatie te hosten.

4 Onderzoeksresultaten

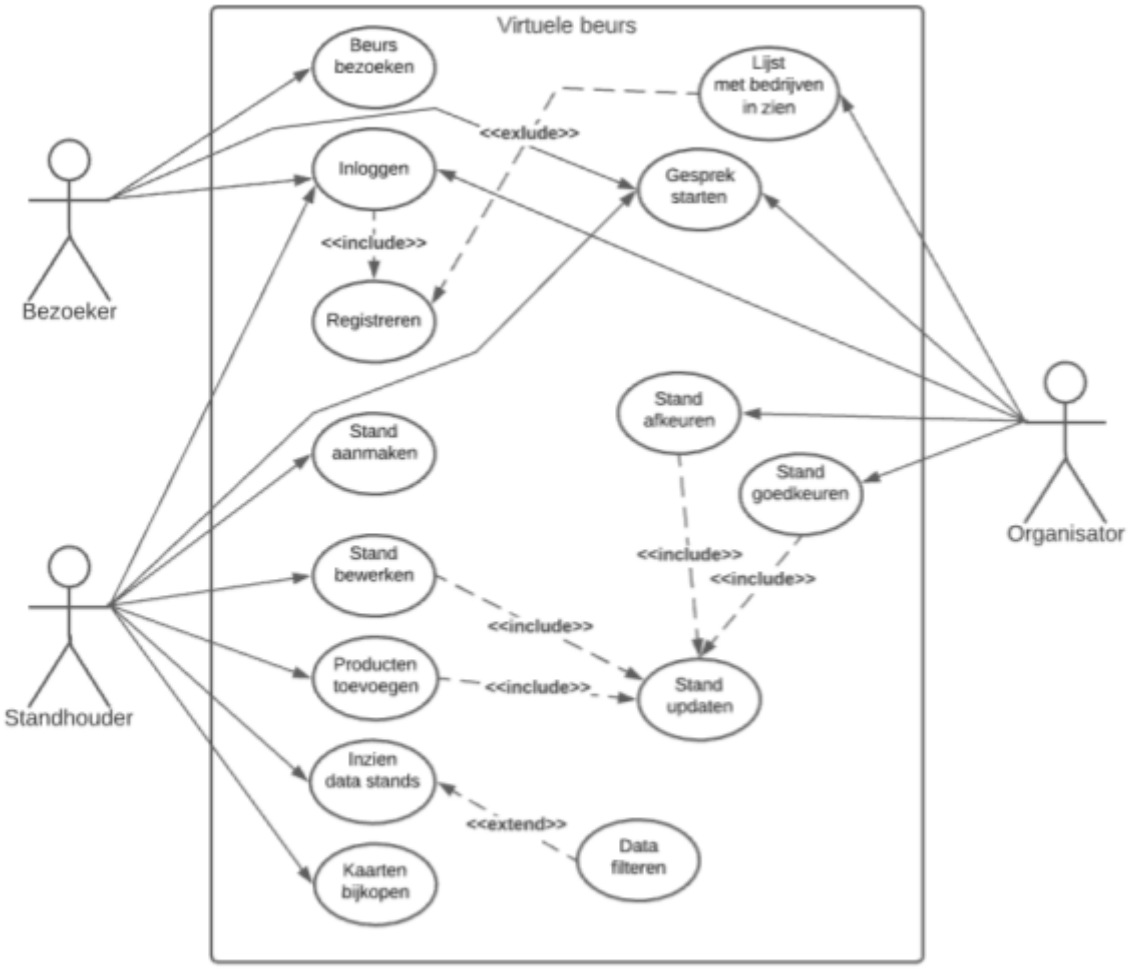
4.1 Leeswijzer onderzoek

De resultaten beginnen met de use cases. Hieruit komt een use case diagram. Vervolgens wordt er gekeken hoe de data kan worden beveiligd. Daarna wordt er gekeken welke videobelapplicatie het beste past bij het project. Als laatste wordt er gekeken welke architectuur het best past bij dit project.

4.2 Resultaten

4.2.1 Wat kunnen use cases zijn voor het platform?

Om er achter te komen wat use cases kunnen zijn voor het platform zijn er verschillende interviews gehouden met verschillende stakeholders. De stakeholders in dit geval waren iemand uit de beurzen sector Leonie en de eigenaar van het bedrijf Arne. Daaruit is duidelijk geworden wat er verwacht wordt van het systeem. Hiermee is een use case diagram opgesteld (zie afbeelding 1 hieronder) waarin duidelijk wordt welke interactie de verschillende gebruikers kunnen hebben met het platform. Zo is er bijvoorbeeld uitgekomen dat een organisator een lijst wil hebben met bedrijven zodat de organisator contact kan opnemen met het betreffende bedrijf (zie bijlage 1: Interview met Leonie Pamboer).



Afbeelding 1: Use case diagram

4.2.2 Hoe moet de data beveiligd worden om het platform veilig te maken?

Het beveiligen van data voor een gebruiker bestaat al sinds het gebruik van de computer. Vroeger werd er echter nog niet veel over nagedacht en was het meer een wens van de klant dan dat het mee werd genomen in het ontwerp. Dit kwam vooral omdat de computer en het gebruik daarvan nog in trek waren was en niet toegankelijk voor iedereen. Ook was er geen regelgeving voor het verwerken van gebruikersdata.

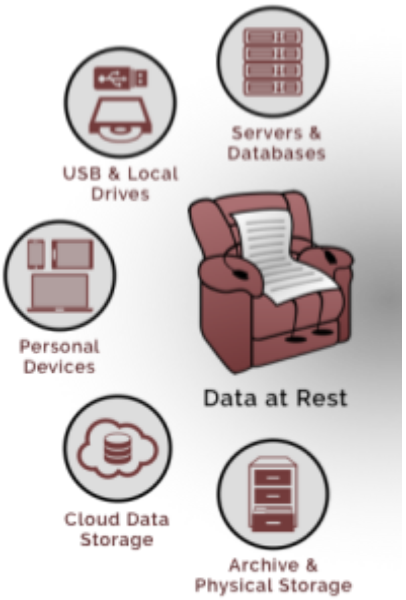
Met het toenemende internetverkeer en het gebruik van de computer en tegenwoordig ook de smartphones, is het steeds belangrijker om ook de data van applicaties te beveiligen. Computers worden tegenwoordig zo sterk dat het steeds makkelijker wordt om bepaalde encrypties te ontcijferen. Daarnaast worden er ook wetten gemaakt die reguleren welke data verwerkt mag worden en op welke manier.

Het beschermen van data gaat op verschillende lagen en op verschillende manieren. Zo kan de data beschermd worden aan de serverkant maar ook aan de kant van de applicatie. Ook kan de data

worden beveiligd op het moment dat de data over het internet gaat. Voor elke plek voor het beveiligen van data, zijn er manieren om dit te doen.

4.2.2.1 Data at rest

Data at rest houdt in dat data ergens is opgeslagen en niet wordt gebruikt of verzonden naar iemand. Dit kan data zijn op een USB-stick maar ook op een lokale schijf of in de cloud of op een server. Eigenlijk overal waar data opgeslagen kan worden, zowel op fysieke wijze als op logische wijze.



Afbeelding 2: voorbeelden data at rest

Voor het beveiligen van data in rest zijn een aantal mogelijkheden. De eerste is 'Full-disk encryption'. Tegenwoordig hebben de meest operating systems, denk hierbij aan Windows of Linux, dit al wel ingebouwd in hun servers die ze aanbieden. Bij full-disk encryption worden de schijven, waar de data opstaan, in zijn geheel versleuteld. Dit wordt vaak gedaan met een vorm van encryptie. De vorm van encryptie die in dit geval vaak wordt gebruikt is een symmetrische encryptie waarbij een beheerder een sleutel of wachtwoord moet invoeren om toegang te krijgen tot de server. Ook moet de beheerder deze sleutel of wachtwoord invoeren om de data te versleutelen. Met full-disk encryption hoeft de software op de server niets bijzonders te doen om op een normale manier te kunnen werken. De OS biedt transparante toegang tot de versleutelde data. Dit gebeurt met een minimaal verlies in prestatie van de server. Het is wel belangrijk om te versleutelen voordat er gewerkt gaat worden met data. Dit komt omdat voor de initiële versleuteling de data wordt verwijderd. Daarom wordt er aangeraden om dit te doen op het moment dat de server wordt uitgerold.

Een andere manier om data in rest te versleutelen is 'file system encryption'. Meestal zijn de schijven waar data wordt opgeslagen onderverdeeld in mappen (files). Een beheerder kan bij deze encryption kiezen welke mappen met data versleuteld moeten worden. In tegenstelling tot de full disk encryption, wordt bij file system encryption alleen de mappen die zijn geselecteerd versleuteld. Door gebruik te maken van file system encryption kan er een server worden gemaakt die geen sleutel of wachtwoord nodig heeft om op te starten. De sleutel of wachtwoord zijn dan alleen nodig als er toegang nodig is tot die data en dit kan terwijl de server al aan het draaien is. Net zoals bij full disk encryption is toegang tot de data voor applicaties transparant. Hierbij moeten gebruikers van de server oppassen dat ze geen gevoelige informatie zetten in de mappen die niet versleuteld zijn.

Daarnaast heb je ook nog de 'database encryption'. Deze methode zorgt voor bescherming van data at rest op databaseniveau. Database software (zoals Oracle, MySQL) kunnen op een applicatie niveau versleuteling bieden. Net zoals bij OS niveau versleuteling, moet er gebruik gemaakt worden van een sleutel of wachtwoord. Deze moet worden ingevoerd tijdens het opstarten van de server. Na het invoeren van de sleutel of wachtwoord heeft de beheerder transparante toegang tot de data. (*Encrypting Data at Rest on Servers: What Does It Get You? – HLN Consulting, LLC*, 2016)

Voor het platform is de beste optie om data te beveiligen in de database: ‘database encryption’. Deze manier van encryptie is een goede manier om de data te beveiligen. Er wordt wat data van een gebruiker verzameld, daarom is het verstandig om de data te beveiligen. Aangezien het gaat om niet al te gevoelige data is het niet noodzakelijk om de data nog veel zwaarder te beveiligen. Daarnaast wordt bij veel gehoste servers al een ‘full-disk-encryption’ aangeboden die er voor zorgt dat de data nog een keer wordt encrypt. Deze twee manieren gecombineerd moeten voldoende zijn om de data van de gebruiker veilig te maken.

4.2.2.2 Data in transit

Met data in transit wordt bedoeld dat de data in beweging is. De data gaat dan van de ene locatie naar de andere locatie. Deze beweging gaat dan of over het internet of over een privé netwerk. Een groot probleem met data in transit is dat het onbeschermd beweegt.



Afbeelding 3: voorbeelden data in transit

Een manier om deze data te kunnen beschermen is ‘end-to-end encryption’. Dit houdt in dat de data wordt versleuteld bij het beginpunt en ontsleuteld alleen bij het eindpunt. Er wordt een tekst versleuteld. Deze tekst wordt dan verzonden over het internet of een privé verbinding en is niet te lezen door andere, als de tekst is aangekomen wordt deze ontsleuteld.

‘Data Loss Prevention’ is een systeem om data te beveiligen op het moment dat de data in transit is. Dit systeem wordt gebruikt binnen netwerken en systemen. Het systeem scant dan op onregelmatigheden in de data die wordt verzonden of ontvangen. Tijdens het scannen wordt er gekeken naar een paar afgesproken regels voor het gebruik van de data. Bijvoorbeeld het systeem scant een e-mail van een medewerker die klanteninformatie deelt met een burgerservicenummer. Een DLP-systeem kan dit dan blokkeren of dit gedeelte versleutelen. (*Protecting the Three States of Data*, n.d.)

De meest gangbare manier om data in transit te beveiligen is om gebruik te maken van ‘end-to-end encryption’. Dit is waarschijnlijk ook de beste manier om de data van de gebruiker voor dit platform te beveiligen. Deze encryptie voldoet voldoende om de gegevens van de gebruiker te beveiligen wanneer deze in transit is.

4.2.2.3 Data in use

Als laatst is er nog de term ‘data in use’. Dit houdt in dat de data niet passief is opgeslagen op bijvoorbeeld een server, maar in gebruik is in het systeem. Dit kan zijn dat de data wordt gegenereerd of dat de data wordt verwijderd. Dit soort handelingen met het systeem is dus ‘data in use’. Veel van deze data wordt bewaard in het random-access-memory (oftewel RAM) van het systeem. Hierdoor is het systeem vatbaar voor een cold boot attack. Dit houdt in dat een hacker fysiek aanwezig moet zijn bij het systeem. Vervolgens zorgt de hacker ervoor dat het systeem kan worden opgestart met een USB en vervolgens zet de hacker het systeem uit. Omdat data nog een

kleine tijd in het RAM blijft staan kan de hacker nu het systeem weer opstarten met bijvoorbeeld een USB-stick en kan dan zo de informatie van het RAM halen. Hierin kunnen bijvoorbeeld encryptie sleutels staan of andere zaken die belangrijk kunnen zijn voor een hacker.

Ook voor data in use wordt er gebruik gemaakt van verschillende soorten encryptie om de data te beveiligen. Een van die vormen is 'full memory encryption'. Met full memory encryption wordt de data die in het RAM staat versleuteld. Deze vorm van encryptie is wat lastiger te implementeren aangezien dit moet worden gedaan in de CPU. Er zijn nog niet veel CPU's die dit kunnen. Echter is AMD Epyc de eerste processor die deze vorm kan ondersteunen. AMD gebruikt hiervoor hun eigen versie van full memory encryptie. AMD noemt het 'Secure Memory Encryption (SME)'. Hierbij kunnen softwarematig pagina's worden aangewezen om versleuteld te worden. Deze pagina's worden automatisch versleuteld en ontsleuteld bij het lezen en schrijven naar deze pagina's. De vorm van encryptie die hiervoor wordt gebruikt is de 128-bit AES encryptie. De sleutel voor deze encryptie wordt willekeurig gegenereerd door de hardware. Een nieuwe sleutel wordt bij elke keer bij het opstarten opnieuw gegenereerd.

Een andere manier om dit probleem op te lossen is om de sleutels niet in het RAM te bewaren, maar in de CPU zelf. Er zijn twee bedrijven die hier een oplossing voor hebben gevonden, namelijk TRESOR en Loop-Amnesia. Beide bedrijven veranderen de kernel van een OS zo dat de sleutels kunnen worden bewaard in de CPU registers. De sleutels die op deze manier worden bewaard worden meteen verwijderd uit het register op het moment dat het systeem plotseling wordt afgesloten. In beide gevallen moeten de sleutels tijdens het gebruik worden aangemaakt in verband met het weinige geheugen op het register. Daarnaast zorgen beide bedrijven ervoor dat er geen interrupts komen tijdens het versleutelen en het ontsleutelen. Dit zorgt ervoor dat de sleutels niet door kunnen lekken naar andere plekken in het geheugen. Daarnaast zorgen beide bedrijven ervoor dat het niet mogelijk is om gebruik te maken van de debug optie op de CPU, zodat ook hierbij niet naar een sleutel kan worden gezocht.

De laatste status waarin data zich kan bevinden is wat overdreven om voor dit platform te beveiligen. De data die in het RAM van de gebruiker staat, staat daar waarschijnlijk al veilig genoeg voor eventuele aanvallen. Deze aanvallen zullen fysiek moeten worden gedaan en dat is vrij veel werk om er achter te komen welke producten een gebruiker interessant vindt. Dus het is voor deze status van de data niet relevant om hier rekening mee te houden.

4.2.2.4 Rollen en encryptie

Naast de data te beveiligen in de verschillende staten is er ook nog sprake van veiligheid in de vorm van privacy. Voor het platform worden er ook persoonsgegevens verwerkt. Denk aan bijvoorbeeld e-mailadressen (*Autoriteit persoonsgegevens*, 2021). Daarnaast wordt ook data gebruikt van gebruikers met betrekking tot de stands. Dus welke producten hebben ze aangeklikt, hoe lang zijn ze in de stand geweest. Ook voor deze data is het niet gewenst dat die terecht komt bij de verkeerde standhouder. Daarom is het ook van belang dat niet alleen de data wordt beveiligd maar dat er ook rollen komen voor de verschillende gebruikers zodat de data van de gebruikers niet in de verkeerde handen vallen.

Door gebruik te maken van rollen kan bij het inloggen gekeken worden welke gebruiker het is en welke rol deze gebruiker heeft. Daardoor kan er later gekeken worden bij welke pagina's de gebruiker

wel toegang heeft en bij welke schermen niet. Ook kan er hierdoor gekeken worden welke gebruiker toegang heeft tot de juiste data van bijvoorbeeld zijn stand. Hierbij krijgen verschillende gebruikers dus ook verschillende rollen. Zo heb je een bezoeker die alleen over de beurs kan lopen. Daarnaast heb je de organisator die kan kijken welke bedrijven er zijn en hoe de stands van de standhouders er gaan uitzien. Als laatste zijn er nog de standhouders. De standhouders kunnen de data inzien van de stand die zij hebben aangemaakt. Daarom is het dus ook belangrijk dat er rollen bestaan zodat de ene standhouder geen toegang heeft tot de data van de stand van een andere.

4.3 Welke videobel API past het beste bij dit platform?

Één van de requirements voor het systeem is om een communicatiemiddel te hebben tussen de gebruikers. Voor B2B (business to business) is het gewenst om te kunnen videobellen. Hiervoor zijn meerdere mogelijkheden. Een van de mogelijkheden is om het zelf te maken en te ontwerpen. Het nadeel hiervan is dat er veel moet worden uitgezocht over de werking van videobellen en dat het veel tijd kost om het zelf te maken. De voornamelijkste reden is dat het werkt met WebRTC en dit maakt gebruikt van peer to peer (p2p) connecties. Daarvoor moet er gewerkt worden met een STUN of TURN server.

4.3.1.1 WebRTC

WebRTC staat voor Web Real-Time Communication. Het is een open source protocol voor het p2p streamen van o.a. spraak en videogesprekken. Het protocol is bedoeld voor het web zodat er geen extra plug-ins of software vereist zijn. Doordat er een rechtstreekse verbinding is met de andere gebruiker kan er audio, video en data real-time worden onderling gedeeld. Om ervoor te zorgen dat er een stabiele verbinding kan worden opgezet met een andere gebruiker, moet er gebruik gemaakt worden van een STUN/TURN server.

4.3.1.2 STUN server

STUN staat voor ‘Session Traversal Utilities for NAT’. Dit is een gestandaardiseerde set aan methodes om voorbij de NAT (Network Address Translator) gateways te komen om een verbinding te kunnen opzetten met een andere gebruiker. Hierdoor kun je real-time video, voice en andere data streamen tussen de gebruikers. Een NAT zorgt ervoor dat er niet steeds een nieuw IP adres hoeft te geven als een gebruiker of server verplaatst wordt. Dit zorgt er echter wel voor dat je een programma niet weet welk IP de andere gebruiker heeft. Daardoor wordt het niet mogelijk om P2P data te verzenden aangezien er niet bekend is welke PC die data nodig heeft. Het programma heeft in dat geval het daadwerkelijke IP adres nodig van de gebruiker. Om dit probleem op te lossen is er een STUN server. De client die data wil gaan sturen, stuurt dan een request naar een STUN server op het publieke internet. Daarna geeft de STUN server een response met het IP adres en poortnummer van de client die de STUN server kan zien. Het resultaat is dus het publieke IP adres van de client, het type NAT waar de client achter zit en de poort waarop de client draait. Hiermee kan er een UDP verbinding worden opgezet met een andere client. Een verbinding met UDP is echter niet beveiligd en er moet dus worden gezorgd dat de data beveiligd wordt.

4.3.1.3 TURN server

TURN staat voor ‘Traversal Using Relays around NAT’ en is een protocol dat helpt met een verbinding van twee gebruikers die achter een symmetrische NAT zitten. Dat is ook meteen het grootste verschil. Als een gebruiker achter een symmetrische NAT zit kan er geen gebruik worden gemaakt van

een STUN server. De gebruiker maakt dan een ‘allocate’ request naar een STUN server. Dit request houdt in dat de gebruiker vraagt om een stukje resource om zo een verbinding te kunnen opzetten met iemand anders. Als er een stukje resource beschikbaar is, stuurt de server een adres terug. Daarna stuurt de gebruiker een ‘CreatePermission request’ om er zo voor te zorgen dat als er een communicatie opgezet kan worden, de TURN server weet dat de communicatie tussen de gebruikers gevalideerd is. Nadat de toestemming is gegeven heeft de gebruiker twee keuzes om data te gaan verzenden: gebruik maken van het ‘Send mechanism’ of een reservering maken voor een channel. Het nadeel van het Send mechanism is dat de header vrij groot is (namelijk 36 bytes). Dit vereist dat er een goede internetverbinding nodig is aangezien dit zorgt voor een vrij grote bandwidth. Het voordeel is dat het meer rechttoe rechtaan is. Het voordeel van het reserveren van een channel is dat het maar 4 bytes groot is. Het nadeel is alleen dat het om de zoveel tijd ververst moet worden. Net zoals bij een STUN server: als er een verbinding is met een andere gebruiker (met welke van de twee methodes dan ook) wordt de data verstuurd over UDP. De TURN server relays de verbinding naar de andere gebruiker.

4.3.1.4 MCA (Multi Criteria Analyses)

De keuze is gemaakt om het videobellen niet zelf te maken maar gebruik te maken van een bestaande API. De reden waarom hiervoor is gekozen omdat er tegenwoordig bestaande API’s zijn die goed te gebruiken zijn. Een ander reden is dat het lang duurt om een videobel API zelf te maken.

Om te kijken welke API’s bruikbaar zijn is er op het internet gezocht naar de meest voorkomende API’s. Uit dit onderzoek is een long list gekomen.

Pubnub	TokBox	Agora
CometChat	Contus Fly	SendBird
Twilio	Apphitect	Applozic
Quickblox	Smooch	Mesibo
Sinch	ChatCamp	Firebase + WebRTC Codelab

Tabel 1: lijst met mogelijke videobel api's

Uit deze lijst zijn er 5 API’s gekozen om verder met elkaar te vergelijken. De reden waarom deze vijf zijn gekozen is omdat deze er op het eerste oog het beste eruit zagen. Hiermee wordt bedoeld dat de vijf keuzes voor het web bedoeld zijn en niet alleen voor android of iOS. De vijf die zijn gekozen zijn te vinden in de tabel hieronder.

Er is samen met de product owner gekeken wat de criteria’s moeten zijn voor de videobel applicatie. Deze zijn gezamenlijk opgesteld zodat de beste applicatie kan worden gekozen voor zowel het bedrijf als voor de programmeur. De criteria voor de videobel applicatie zijn hieronder in de tabel te vinden te vinden.

	Prijs	Chat optie	Schaalbaar	Documentatie	Gebruik telefoon
Agora	€3.35/1000 minuten of €0.00335/minuut (video) 1e 1000 maandelijkse gebruikers gratis daarboven €12.17/1000 gebruikers (chat)	ja	ja	++	nee
Twilio	lokale gesprekken: €0.01257/min om te maken en €0.008381 om te ontvangen (**Totaal: €0.020951**) Bericht sturen: €0.07709 om te sturen en €0.006285 om te ontvangen (**Totaal: €0.083375**)	ja	ja	+	ja
SendBird	VIDEO: €0.002941/minuut per gebruiker CHAT: €335.22/maand per 5000 gebruikers	ja	ja	+/-	nee
Applozic	VIDEO: €0.001272/minuut per gebruiker CHAT: €251.18/maand tot 10000 gebruikers	ja	ja	++	nee
Mesibo	Totaal: €91.60 per maand (bij 1000 gebruikers) of $0.01 of €0.0084031 per gebruiker	ja	ja	++	nee
WebRTC	Gratis	nee	nee	-	nee

Tabel 2: lijst van producten met waardering

De tabel hieronder vertelt wat over hoe de producten zich verhouden ten opzichte van de andere producten. Deze hebben een waarde gekregen van 0 tot 1. Zo werd er bijvoorbeeld gekeken hoe de prijs zich uit ten opzichte van de ander. Deze is gedaan met de formule:

$$waarde = 1 - \frac{prijs - laagste\ prijs}{hoogste\ prijs - laagste\ prijs}$$

Hierbij zijn de prijs per gebruiker gelijk gesteld om zo een goede vergelijking te kunnen maken. Bij Mesibo is het videobellen gratis dus vandaar dat deze meteen een 1.00 heeft gekregen en is niet opgenomen in de laagste prijs. Dit geld ook voor WebRTC. Om dezelfde reden heeft Applozic niet het volle punt gekregen aangezien er nog wel betaald moet worden maar hij is wel het goedkoopst. Er is ook gekeken in hoeverre de chatfunctie wordt ondersteund en of daarbij een telefoon nodig is. Daarna is er gekeken of het schaalbaar is vanuit de API. Dus kan de API meerdere gebruikers aan. Vervolgens is er gekeken of de documentatie duidelijk is. Als laatste is er gekeken of er een telefoon

moet worden gebruikt. Er wordt 0 punten gegeven als er wel een telefoon moet worden gebruikt. Dit omdat het niet handig is voor de gebruiker om ook nog een telefoonnummer te moeten registreren.

	Prijs videobellen	Prijs chatten	Chat optie	Schaalbaar	Documentatie	Gebruik Telefoon
Agora	0.89	0.95	0.75	1.00	1.00	1.00
Twilio	0.00	0.00	0.25	1.00	0.75	0.00
SendBird	0.92	0.05	0.75	1.00	0.50	1.00
Applozic	0.99	1.00	0.75	1.00	1.00	1.00
Mesibo	1.00	1.00	1.00	1.00	1.00	1.00
WebRTC	1.00	0.00	0.00	0.00	0.25	1.00

Tabel 3: tabel verhouding van producten ten opzichte van andere producten

Hieronder is een tabel waarin wordt gekeken hoe zwaar de ene criteria weegt ten opzichte van de andere criteria. Zo weegt de schaalbaarheid zwaarder dan de prijs. Daarna worden de waardes per rij bij elkaar opgeteld om zo tot een rij totaal te komen. Vervolgens worden de rijtotalen bij elkaar opgeteld om tot een totale weging te komen. Om tot een weegfactor te komen worden de rijtotalen gedeeld door het totaal. Hierdoor is er een weegfactor gekomen en kunnen de criteria's worden vergeleken met elkaar. De zwaarte van de weging is in overleg geweest met Arne.

	Prijs	Chat optie	Schaalbaar	Documentatie	Gebruik telefoon	Rijtotaal	Weegfactor
Prijs	1.00	0.50	0.34	2.00	0.25	4.09	0.16
Chat optie	2.00	1.00	0.67	2.00	0.50	6.17	0.17
Schaalbaar	3.00	1.50	1.00	4.00	1.00	10.50	0.30
Documentatie	0.50	0.50	0.25	1.00	0.25	2.50	0.07
gebruik telefoon	4.00	2.00	1.00	4.00	1.00	12.00	0.34

Totaal: 35.26

Tabel 4: Tabel met weegfactor

Onderstaande tabel gaat uitslag geven over welke API de beste optie is. De waardes uit tabel 2 worden vermenigvuldigd met de weegfactor. Daarna worden de uitkomsten bij elkaar opgeteld. De API met de hoogste waarde is dus de beste optie om te gebruiken.

	Weging	Agora	Twilio	Sendbird	Applozic	Mesibo	WebRTC
Prijs	0.16	0.2944	0	0.1552	0.3184	0.32	0.16
Chat optie	0.17	0.1275	0.0425	0.1275	0.1275	0.17	0
Schaalbaar	0.30	0.30	0.30	0.30	0.30	0.30	0
Documentatie	0.07	0.07	0.0525	0.035	0.07	0.07	0.0175
Gebruik telefoon	0.34	0.34	0	0.34	0.34	0.34	0.34
Totaal		1.1319	0.3775	0.9577	1.1559	1.2	0.5175

Tabel 5: tabel met de uitslag van de weegfactoren

Uit de tabel hierboven is dus gebleken dat Mesibo de beste optie is om te gebruiken. Deze heeft op alle vlakken het hoogst gescoord en zal dus ook de aanbeveling zijn aan het bedrijf om deze te gaan gebruiken.

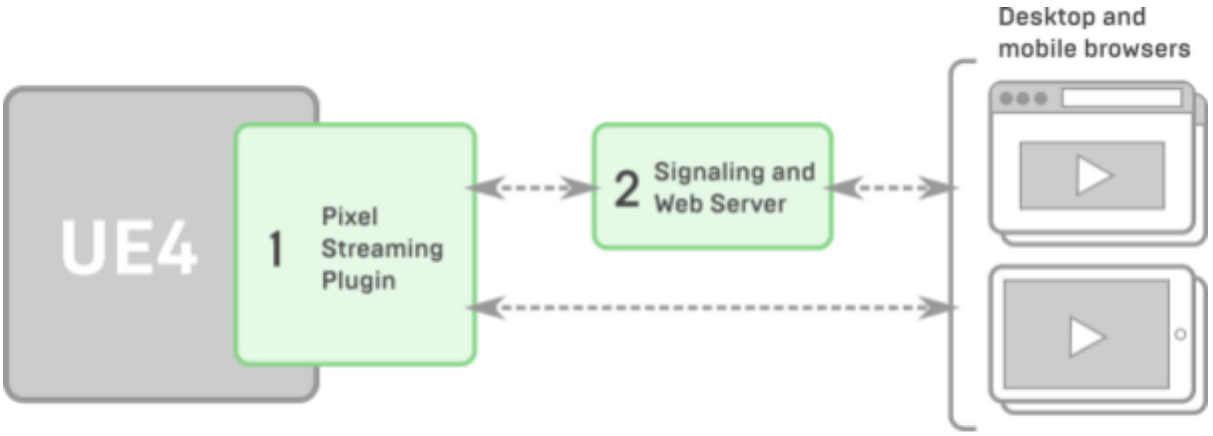
4.3.2 Welke architectuur past het beste bij een platform met Unreal Engine?

Er is een kleine test gedaan om te kijken hoe een Unreal Engine applicatie kan worden gestreamd op een website. Daaruit is gebleken dat een applicatie die op Unreal Engine draait meerdere instances nodig heeft om voor elke gebruiker een eigen beeld te hebben. De game wordt gestreamd via het WebRTC protocol en dit is meestal een peer to peer verbinding. Hierdoor krijgen de gebruikers hetzelfde beeld te zien en als het ene scherm beweegt, beweegt het scherm van de andere gebruiker ook. Dit is niet wenselijk en daarom moet er gekeken worden naar een oplossing die ervoor zorgt dat elke gebruiker zijn eigen beeld te zien krijgt.

4.3.2.1 Pixel Streaming

Normaal gesproken wordt een game applicatie van Unreal Engine op de hardware van de gebruiker zelf gerund. Oftewel de PC van de gebruiker rendered alle grafische beelden en berekent alles. Unreal Engine had de mogelijkheid om alles in een HTML5 packet te bouwen, waardoor de game via een webbrowser kon worden gestreamd. Echter werden hierbij alle berekeningen en logica alsnog op de PC van de gebruiker gedaan. Daardoor was het nog zwaar voor de meeste laptops/PC's. Daarom heeft Epic de Pixel Streaming gemaakt. Hiermee wordt de hardware van de host gebruikt om alle logica te berekenen (deze hardware kan een gewone PC zijn of een virtual machine in de cloud). Daarnaast wordt elk frame van de game gerenderd die ook wordt ge-encoded om vervolgens deze te streamen via een lichtgewicht webservice. De eindgebruiker kan dan de stream bekijken op zijn eigen standaard webbrowser. Hierdoor lijkt het alsof er wordt gekeken naar YouTube of Netflix maar door middel van Pixel Streaming kan de input weer terug worden gestuurd naar de Unreal Engine

applicatie waardoor er interactie mogelijk is.



Afbeelding 4: Communicatie tussen UE4 en de gebruiker via Pixel Streaming

Om gebruik te maken van Pixel Streaming heb je eigenlijk maar twee componenten nodig, namelijk de 'Pixel Streaming plugin' en een 'Signalling and Web Server'. (*Pixel Streaming Overview*, n.d.) De Pixel Streaming plugin is nodig om de gerenderde frames te encoden met de H.264 video compression en deze met de game audio samen te voegen in een media stream. Deze plugin runt in de Unreal Engine applicatie. Daarnaast is er een Signalling and Webserver nodig. Deze is nodig om de verbinding te maken tussen de Unreal Engine en de webbrowsers van de gebruiker. Daarnaast zorgt de web server ervoor dat de gebruiker de juiste HTML en JavaScript krijgt om de media stream te kunnen bekijken.

4.3.2.2 Unreal Engine with Docker

Een veel gebruikte tool in de software ontwikkeling is het gebruik maken van Docker. Docker kan een software applicatie samenvoegen samen met zijn dependencies om zo een virtuele container te maken. Deze containers kunnen dan weer makkelijk worden gehost in bijvoorbeeld de cloud. Het grote voordeel van deze containers is dat de containers geïsoleerd zijn van elkaar. Dus elke container heeft zijn eigen software en dependencies. Daarnaast zijn de docker containers lichtgewicht en zorgen er dus voor dat er meerdere op een server kunnen draaien. (https://www.docker.com/, 2021)

In het algemeen zijn er eigenlijk twee soorten images, namelijk een development image en een runtime image. Het grote verschil zit hem eigenlijk al in de naam. De development image is een image die is gemaakt voor het bouwen van de software. De runtime image is een image die wordt gebruikt voor het runnen van het daadwerkelijke gemaakte programma. Bij een runtime image is het vaak gebruikelijk dat er zo weinig mogelijk extra software wordt gebruikt voor het runnen van een image. Bij een development image wordt er veelal gebruik gemaakt van extra tooling om te helpen bij het bouwen van de software. Deze kunnen vaak vrij groot worden en zijn niet meer nodig bij het runnen van een programma. Daarom is het niet nodig om deze mee te nemen in de runtime image, want dan wordt de image weer groter en kost meer ruimte om deze te draaien.

Ook voor Unreal Engine is er een open source Docker image. Voor deze image zijn er eveneens twee soorten containers (namelijk de development image en de runtime image) (https://unrealcontainers.com/docs/concepts/image-types, 2021).In de Unreal Engine development container zit zowel de Engine build tools als de Unreal Engine Editor. In de runtime container zitten

alleen de gezipte Unreal projects en de runtime dependencies. Voor de Unreal Engine containers is het onderscheid tussen de twee extra belangrijk. Dit komt eigenlijk door twee dingen. Ten eerste komt het omdat de build tools en Editor van de Unreal Engine ervoor zorgen dat de development image vele gigabytes aan data is en daardoor ongeschikt wordt om te runnen als runtime image. Ten tweede komt het door de EULA van Unreal Engine. Deze EULA is niet van toepassing op gezipte projecten, waardoor er een legaal verschil is tussen de twee images.

4.3.2.3 AWS

Er is gekozen om alleen naar AWS te kijken uit eigen ervaring. Amazon is al vrij lang voorloper op het gebied van servers aanbieden voor games. Zo heeft Amazon o.a. een eigen game die ze aan het bouwens zijn voor multiplayers (New World). In deze game kunnen 2000 gebruikers op een server spelen. Deze game wordt natuurlijk gehost op de servers van Amazon zelf. Uit eigen ervaring speelde deze game soepel met 2000 man op de server. Daarnaast maakt Epic games zelf ook gebruik van AWS servers. Microsoft en Google bieden dit ook aan maar zij hebben aanzienlijke minder ervaring met het hosten van games. Daarom is ervoor gekozen om alleen te kijken naar AWS (*Press Release*, 2018). Daarnaast maakt het bedrijf zelf ook gebruik van AWS servers.

Het runnen van een game applicatie is een zware taak voor een gemiddelde computer. Vaak is hiervoor zware hardware voor nodig. Deze games kunnen ook worden gehost in de cloud bij bijvoorbeeld AWS. Deze cloud servers hebben stevige hardware nodig om de games te laten runnen. Hiervoor heeft AWS een server instance genaamd EC2 G3. Daarnaast heeft AWS een server genaamd Amazon GameLift. Deze server is op dit moment helaas niet geschikt voor de applicatie aangezien de game (nog) niet bedoeld is voor multiplayer en deze server is bedoeld voor multiplayer.

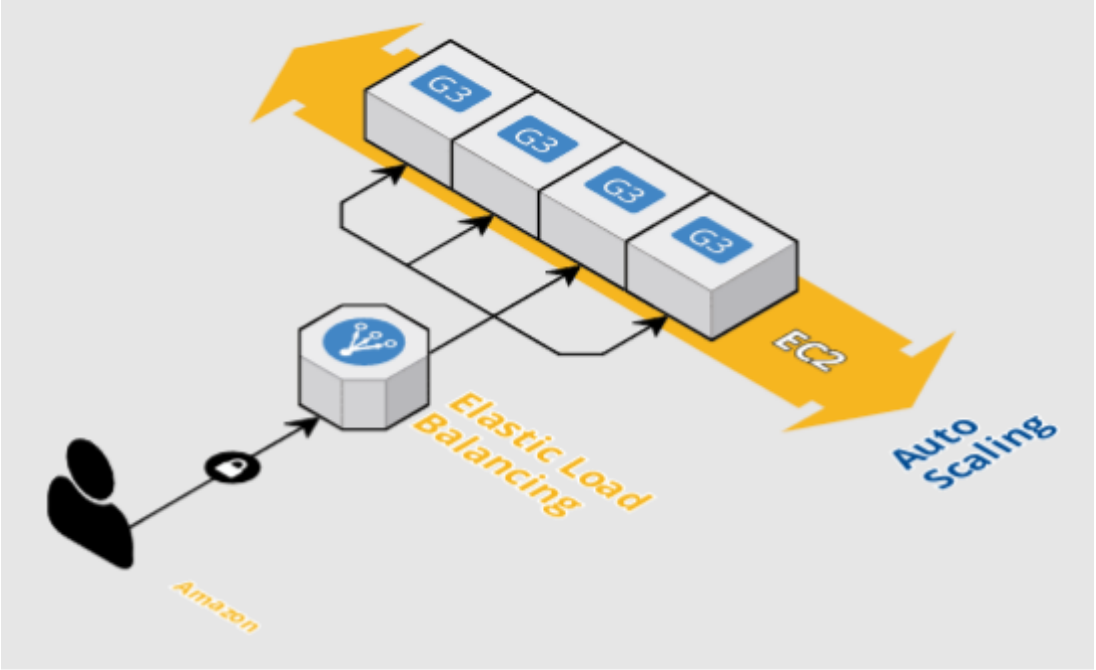
Een EC2 G3 is de laatste generatie van de EC2 GPU grafisch intensieve instances dat een sterke combinatie levert CPU, RAM en GPU capaciteit samen. De G3 servers worden veel gebruikt voor grafisch intensieve applicaties zoals: 3D visualisaties, mid tot high-end virtual werkomgevingen, virtuele applicaties, 3D rendering, applicatie streaming, video encoding en gaming. G3 servers hebben toegang tot de NVidia Tesla M60 GPU. Elke GPU bevat: 2048 parallelle processing cores, 8GB GPU memory en hardware encoder die tot 10 H.265 (HEVC) 1080p met 30 frames kan ondersteunen. Doordat de nieuwste drives worden gebruikt kunnen deze GPU's ondersteuning bieden voor OpenGL, DirectX, CUDA en OpenCL. (*Amazon EC2 G3 Instances — Amazon Web Services (AWS)*, n.d.)

4.3.2.4 PureWeb

PureWeb heeft een platform gemaakt voor het streamen van 3D applicaties. Dit platform houdt in dat er een bestaande infrastructuur is in de cloud die wordt beheerd en kan dynamisch geschaald worden. Deze servers draaien op AWS servers die de juiste configuraties, beveiliging en management hebben en daarom dus dynamisch schaalbaar zijn. Daarbij maakt PureWeb gebruik van de Pixel Streaming service van Unreal Engine. Het enige wat wordt aangeleverd is een link naar de server en hiermee kan de stream bekeken worden. Deze link zou dus ook in een iFrame geplaatst kunnen worden zodat er nog een pagina omheen kan worden gebouwd. Deze oplossing zou toegepast kunnen worden in het platform (*Interactive Cloud Streaming Platform - PureWeb*, n.d.).

4.3.2.5 Architectuurplaat

Het is goed mogelijk om zelf een server op te zetten voor een Unreal Engine applicatie. Zoals gezegd in 4.2.4.3 heeft AWS een G3 server die beschikt over de nodige hardware. Hierdoor kan de game worden gestreamd naar de gebruiker. Echter heeft elke gebruiker zijn eigen instance nodig om door de game heen te gaan. Hiervoor kan er gebruik gemaakt worden van auto scaling zodat elke gebruiker zijn eigen instance krijgen. Als er geen server meer beschikbaar is, worden er nieuwe servers vrijgemaakt. Om voor elke gebruiker een nieuwe server aan te maken is vrij duur. Om dit te verminderen kan er gebruik gemaakt worden van een Docker container. Hierdoor wordt er automatisch een container aangemaakt voor de gebruiker en kunnen er meerdere containers op de server zelf worden gezet. Dit heeft als resultaat dat er meerdere gebruiker op één server kunnen en de kosten gedrukt kunnen worden. Hieronder de architectuurplaat van een auto scaling server van AWS met een autobalancer.



Afbeelding 5: Architectuurplaat van UE4 applicatie gehost op AWS server

4.4 Conclusies

4.4.1 Conclusie deelvraag 1: Wat kunnen use cases zijn voor het platform?

Uit de interviews met Leonie en Arne zijn een aantal use cases gekomen (zie afbeelding 1). Deze zijn te vinden in de afbeelding hieronder. Hieruit zijn een paar use cases gekomen voor de bezoeker, de standhouder en voor de organisator. Ook zijn er een paar overlappende use cases die voor elke actor hetzelfde zijn. Deze zijn daarom belangrijk om te implementeren.

4.4.2 Conclusie deelvraag 2: Hoe moet de data beveiligd worden om het platform veilig te maken?

Het is belangrijk om gevoelige data van gebruikers te beschermen. Voor dit project wordt er ook data bewaard die gelinkt kan worden aan een gebruiker. Daarom is het ook belangrijk om voor dit project de data te beveiligen. De data gaat beveiligd worden op verschillende manieren op verschillende plekken. De eerste plek waar de data wordt beveiligd is op de database. Met een database encryption. Dit zou voldoende moeten zijn om de data op de database veilig te maken. Daarnaast wordt de data die over het internet gaat versleuteld. Hiervoor wordt een end to end encryption gebruikt.

4.4.3 Conclusie deelvraag 3: Welke videobel API past het beste bij dit platform?

Uit de multi criteria analyses is gebleken dat voor dit project de videobel API van Mesibo de beste blijkt te zijn. Deze API is het goedkoopst en biedt alle opties die dit project nodig heeft. Ook zorgt het ervoor dat er geen tijd hoeft te worden gespendeerd aan het zelf ontwerpen en maken van de videobel functionaliteit.

4.4.4 Conclusie deelvraag 4: Welke architectuur past het beste bij een platform met Unreal Engine?

De beste architectuur die past bij het streamen van de unreal engine applicatie naar een website is een oplossing in de cloud. De meest gebruikte oplossing is om een server van AWS te gebruiken en hierop de applicatie te streamen. Dit kan op verschillende manieren. De eerste manier is om de game zelf te gaan hosten op de servers van AWS (een EC2 instance bijvoorbeeld). Dit heeft als grote nadeel dat maar één applicatie per gebruiker op een server kan worden gehost. Ook is het mogelijk om een Docker container te gebruiken. Die zorgt ervoor dat er meerdere gebruikers gebruik kunnen maken van dezelfde server doordat elke gebruiker een eigen container krijgt op de dezelfde server. Daarom hoeven er minder servers te worden gebruikt. Als laatst is het mogelijk om een bedrijf in te huren die dit allemaal regelt. In dit geval wordt er alleen een link naar de server geleverd en heeft iedere gebruiker zijn eigen instance. Het grote nadeel is dat dit minder schaalbaar is en veel geld kan kosten. Daarom is er gekozen om zelf een AWS server te hosten met Docker container images van de applicatie. Door Docker te gebruiken kunnen er meerdere gebruikers op dezelfde server. Waardoor er minder servers nodig zijn om het platform te draaien. Daarnaast is het makkelijk te schalen en hoeft er niemand naar om te kijken. Zie afbeelding 5 (4.3.2.5 Architectuurplaat).

4.5 Slotconclusie

Om een virtuele beurs tot een platform te maken zijn er een aantal zaken nodig. Ten eerste is het belangrijk dat de veiligheid van de gebruiker wordt gewaarborgd. Dit wordt gedaan door middel van encryptie en rollen zodat de data niet bij de verkeerde persoon terecht komt en de data beveiligd is.

De encryptie gaat op verschillende plekken plaatsvinden. De data wordt zowel encrypt op de database en end to end encrypt.

Ten tweede is er voor het contact tussen de verschillende gebruiker van de beurs ook een middel van communicatie nodig. Dit gaat door middel van chatten en videobellen. Het tot stand brengen van deze communicatie wordt niet volledig zelfstandig gedaan maar er gaat gebruik gemaakt worden van Mesibo. Dit is een videobel API die ook het chatten ondersteund.

Als laatste is er een server nodig die de Unreal Engine applicatie voor de gebruikers kan hosten. De beste oplossing om de game te kunnen streamen is om dit te doen met AWS. AWS heeft goede opties om servers te hosten met sterke hardware zoals een GPU. Daarnaast wordt de game applicatie in een webomgeving getoond via een iframe. Hierdoor kan er om de applicatie heen nog een website worden gebouwd. Op de servers worden Docker containers gebruikt om zo veel mogelijk ruimte te benutten per server.

Als laatste wordt er in het platform data verzameld en verwerkt. De data die wordt gebruikt voor de game en voor de website worden bewaard in een SQL database. De data wordt vervolgens aangeleverd door een API die alle dataverkeer verwerkt. Dit alles bij elkaar maakt het weer mogelijk om beurzen te houden.

5 Ontwerp

In dit hoofdstuk worden zowel het functioneel ontwerp als het technische ontwerp behandeld. In het functioneel ontwerp wordt vooral gesproken over hoe het systeem eruit gaat zien en welke wensen van de klant er worden verwerkt. In het technisch ontwerp wordt vooral gekeken naar hoe het systeem werkt.

5.1 Functioneel ontwerp

5.1.1 Use case diagram

Uit het onderzoek is een use case diagram gekomen (zie afbeelding 1) door naar de wensen te luisteren van de geïnterviewde Leonie en Arne. Het begint met de bezoeker. De bezoeker moet zich kunnen registreren en vervolgens kunnen inloggen. Na het inloggen moet de gebruiker de beurs kunnen bezoeken. Tijdens het bezoeken van de beurs moet de bezoeker ook de mogelijkheid hebben om contact te kunnen opnemen met de standhouder. Dit kan worden gedaan door te videobellen of via de chat.

De standhouder moet zich ook kunnen registeren en vervolgens kunnen inloggen. Na het inloggen moet de standhouder een stand kunnen aanmaken die op de beurs komt te staan. Vervolgens moet de standhouder deze stand kunnen inzien en eventueel kunnen aanpassen. Daarnaast moet de standhouder ook producten kunnen toevoegen aan de stand om deze te kunnen laten zien. Ook moet de standhouder de data van zijn stand kunnen inzien aan de hand van wat gebruikers hebben gedaan met de stand. Denk hierbij aan bijvoorbeeld hoe lang zijn ze op de stand geweest? Of op welke producten hebben de bezoekers geklikt? Daarnaast moet de standhouder nog de mogelijkheid hebben om kaarten bij te kopen en deze te kunnen sturen naar relaties. Als laatst is er ook de mogelijkheid om in contact te kunnen komen met zowel de bezoekers al de organisatie.

De organisator moet ook kunnen inloggen. Na het inloggen moet de organisator een lijst kunnen zien met de bedrijven die aanwezig zijn op de beurs. Daarnaast moet de organisator ook stands kunnen inzien om deze goed te keuren of eventueel af te keuren als deze niet voldoen aan de wensen van de beurs. Als laatst moet de organisator ook contact kunnen opnemen met de standhouders.

5.1.2 Requirements

In het onderzoek zijn er een aantal interviews gehouden om er achter te komen wat de gebruikers van het systeem verwachten. Hieruit is een use case diagram gemaakt om de interacties met de gebruikers te beschrijven. Vanuit die use case zijn de requirements opgesteld van de gebruikers waar de interviews mee zijn gehouden. Deze zijn te vinden in de tabel hieronder. De requirements zijn MoSCoW geordend en SMART geformuleerd.

Business/ User	Requirement	MoSCoW
Business	Gebruikers moeten onderling kunnen communiceren door middel van chatten of videobellen	Must

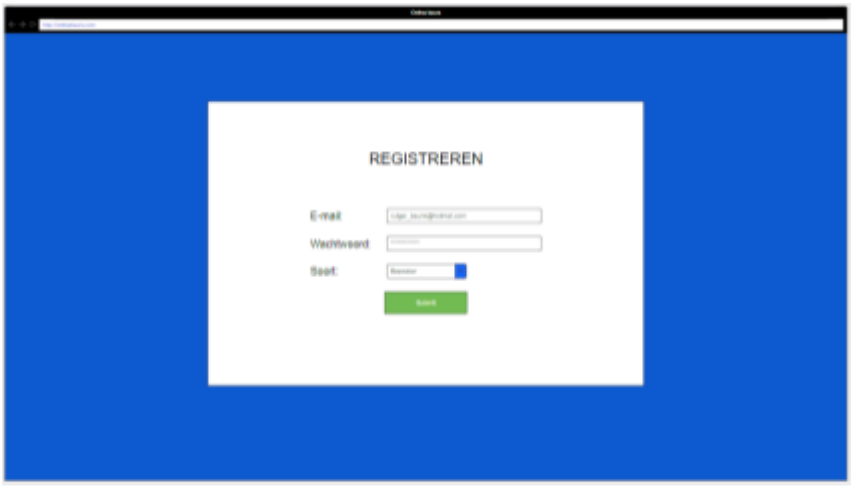
Business	De oplossing moet schaalbaar zijn zodat er meer gebruikers in het systeem kunnen komen naarmate er meer vraag is.	Must
System	Het platform moet een Unreal Engine versie 4.26 of hoger applicatie kunnen draaien	Must
User	De gegevens van de gebruiker moeten veilig worden bewaard	Must
User	Een gebruiker zonder sterke hardware moet ook op de beurs kunnen rondlopen	Must
User	Een standhouder moet een stand kunnen aanmaken en aanpassen	Must
User	Een bezoeker moet de beurs kunnen bekijken	Must
Business	Alleen organisatoren kunnen een stand goedkeuren	Should
Business	De standhouder heeft toegang tot de data die de applicatie van zijn stand heeft verzameld	Should
User	De organisatie heeft overzicht van de bedrijven die op de beurs staan	Should
Business	De standhouder heeft mogelijkheid tot het bijkopen van kaarten	Would
User	De gebruiker kan inloggen met OAUTH	Would

Tabel 6: Requirements

5.1.3 Wireframes

In dit hoofdstuk worden de schermen benoemd. Er wordt uitgelegd per scherm wat het scherm inhoudt, hoe het scherm eruit ziet en wat de gebruiker met het scherm kan doen.

5.1.3.1 Registreren



Afbeelding 6: wireframe registreren

Dit is het scherm voor de registratie. Hier kan de gebruiker een account aanmaken en daarna inloggen. Ook kan de gebruiker kiezen wat voor soort account hij gaat aanmaken. De website heeft verschillende functies voor zowel de gebruikers als de standhouder en de organisator.

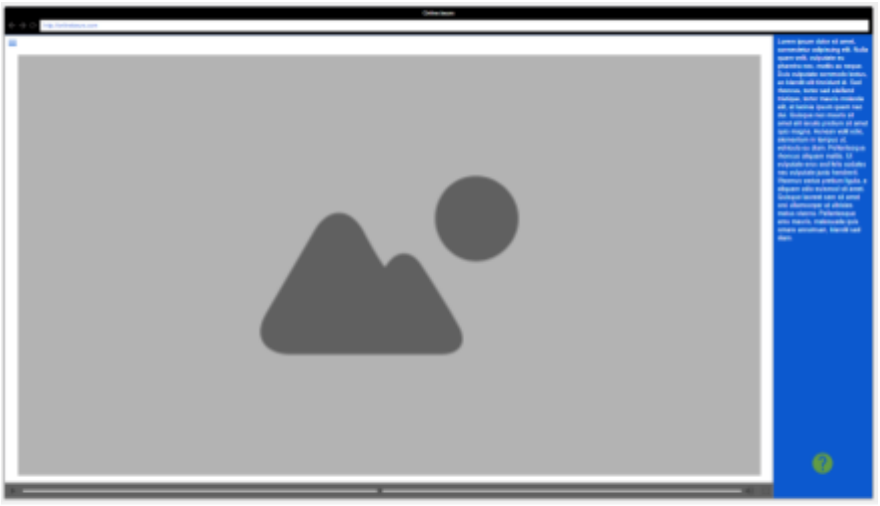
5.1.3.2 Inloggen



Afbeelding 7: wireframe inloggen

Dit is het scherm waarop de gebruiker komt als deze naar de website gaat. Op dit scherm kan de gebruiker inloggen of naar het registratie scherm gaan als de gebruiker nog geen account heeft. Na het inloggen wordt de gebruiker doorgestuurd naar een volgend scherm. Het scherm waar de gebruiker naar toe wordt gestuurd hangt af van welk soort gebruiker er is ingelogd.

5.1.3.3 Beurs scherm

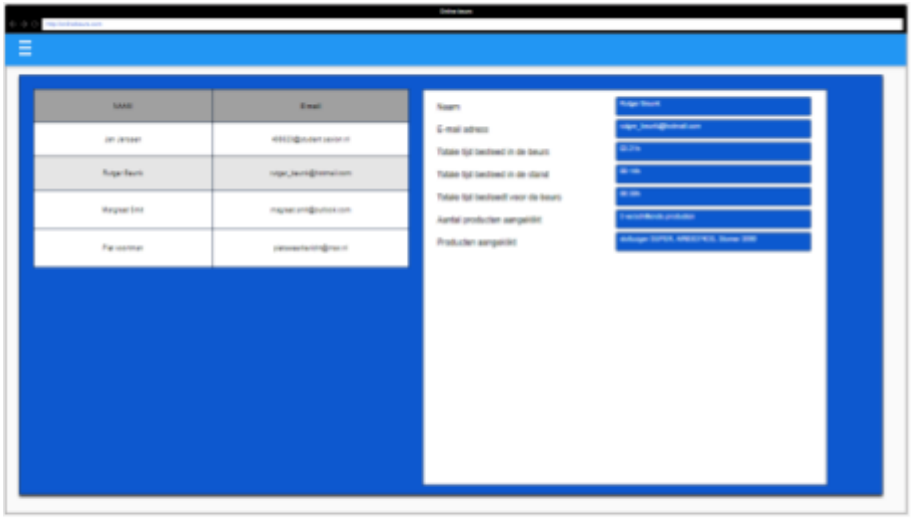


Afbeelding 8: wireframe bezoeker

Dit is het scherm waar de bezoeker opkomt na het inloggen. Op dit schermt komt een iframe met de gestreamde beurs. De reden dat er is gekozen voor een iframe is omdat er naast de game ook extra data wordt weergeven op het scherm. Dit kan dus het beste gedaan worden door een iframe te gebruiken en de game hier naartoe te streamen. Naast de beurs scherm komt een stuk tekst te staan

waar de informatie van de beurs en eventueel informatie van de stand staat die de gebruiker op dat moment bezoekt. Daaronder is een knop om in gesprek te kunnen komen met bedrijven.

5.1.3.4 Overzicht bezoekers



Afbeelding 9: wireframe standhouder

Dit is het scherm waar de standhouder op komt nadat de standhouder is ingelogd. Op dit scherm heeft de standhouder een overzicht van de gebruikers die zijn stand hebben bezocht. Links op het scherm is de lijst met gebruikers die de stand hebben bezocht. De lijst bestaat uit de bezoekers namen en hun e-mail adres zodat de standhouder snel contact kan opnemen met de bezoeker. Als de standhouder op een naam klikt komt er een lijst aan de rechterkant te voorschijn. In de lijst staat een overzicht met de informatie over de gebruiker met betrekking tot de stand. Hier kan de standhouder bijvoorbeeld zien hoelang de bezoeker in zijn stand is geweest en op welke producten de bezoeker heeft geklikt.

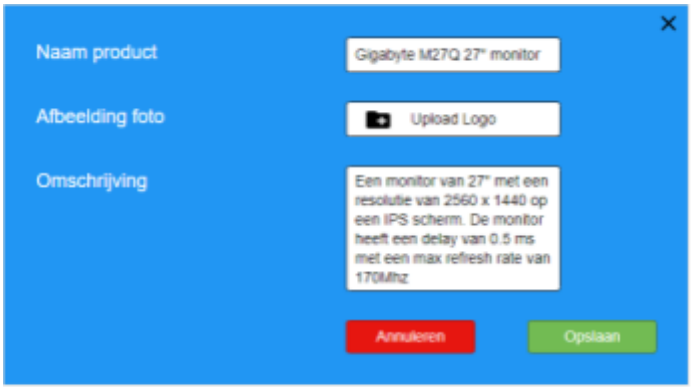
5.1.3.5 Stand aanmaken/aanpassen



Afbeelding 10: wireframe stand toevoegen

Dit is een model die op het scherm komt als een gebruiker (standhouder) een stand wil aanmaken of aanpassen. De gebruiker kan hier een stand aanmaken aan de hand van een paar criteria die moeten worden ingevuld. Er moet een type stand worden gekozen, een logo worden geüpload, een banner worden geüpload, het aantal vertegenwoordigers worden ingevoerd aan de hand van hoeveel er beschikbaar zijn tijdens de beurs, een (korte) video worden geüpload en er kunnen producten worden aangemaakt (Zie hoofdstuk ‘product aanmaken/aanpassen’). De producten kunnen vervolgens aangepast of verwijderd worden als deze zijn aangemaakt. Aangezien een standhouder maar een stand kan hebben is het niet nodig om een lijst te maken voor de stands en kan de stand dus meteen ingeladen worden als deze al bestaat. Vervolgens kan de gebruiker de stand ook verwijderen als de gegevens niet meer kloppen. Elke verandering moet worden goedgekeurd door de organisatoren (zie hoofdstuk ‘beginscherm organisatie’).

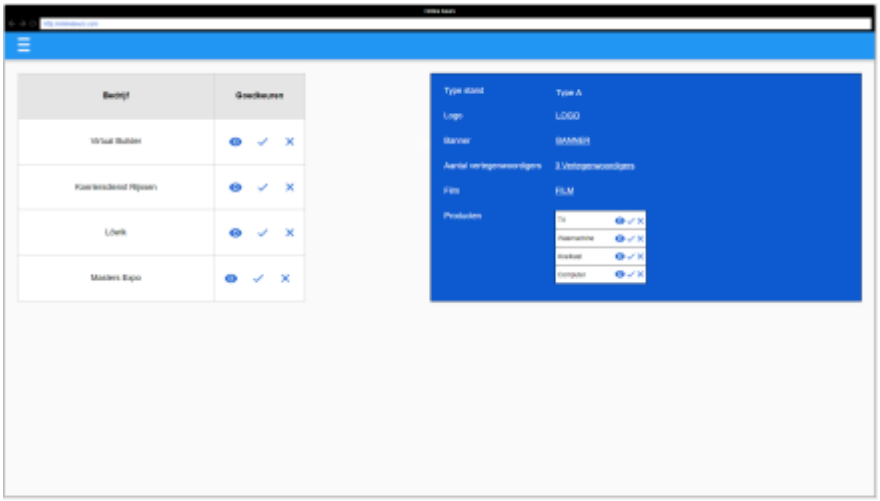
5.1.3.6 Product aanmaken/aanpassen



Afbeelding 11: wireframe product aanmaken

Dit is het modal die de gebruiker (standhouder) krijgt als hij op de knop “nieuw product” klikt. Op dit scherm kan de gebruiker een nieuw product toevoegen aan de stand. Hiervoor heeft de gebruiker een naam nodig van het product, een afbeelding van het product en een korte omschrijving van het product. De gebruiker kan deze dan opslaan of annuleren. Als de gebruiker al een bestaand product heeft en deze wil aanpassen komt hetzelfde scherm ervoor, maar dan is het scherm al ingevuld met wat er bekend is van het product en anders is het scherm leeg.

5.1.3.7 Beginscherm organisator



Afbeelding 12: wireframe organisator

Dit is het scherm voor de organisator als deze inlogt. De gebruiker (organisator) krijgt een overzicht met een lijst van de bedrijven. In de lijst kan de organisator vinden welk bedrijf een verandering heeft gedaan aan zijn online stand. De lijst met ingevulde velden van een bedrijf krijgt de gebruiker te zien nadat de gebruiker klikt op de naam van het bedrijf. Naast de lijst krijgt de gebruiker een scherm te zien waar hij deze veranderingen kan inzien. In het scherm kan de gebruiker bijvoorbeeld een logo downloaden van de gebruiker of een lijst zien met de producten die worden aangeboden. Bij de lijst met producten die worden aangeboden kan de gebruiker ook inzien welke producten het zijn en wat de omschrijvingen er zijn. Deze kan de gebruiker accepteren of weigeren als deze niet goed zijn. Als alles goed is kan de gebruiker de stand goedkeuren of afkeuren. Bij het afkeuren krijgt de gebruiker de reden voor afkeuren zodat het bedrijf ook weet wat er fout aan is. Ook kan vanuit het scherm de informatie van een bedrijf worden bekeken.

5.1.4 Testplan

Het platform moet ook getest gaan worden. Dit wordt gedaan op verschillende manieren. De onderdelen die getest zullen worden zijn de website pagina’s, de API calls die worden gemaakt en de query's met de database.

5.1.4.1 Website pagina’s

De website pagina’s die worden gemaakt zullen getest worden door middel van een review. Als een website pagina gemaakt is wordt deze pagina getest door de begeleider. Deze moet de pagina goedkeuren voordat de pagina wordt gezien als voltooid. Mocht er iets niet goed zijn dan kan de

begeleider het afkeuren en worden de veranderingen doorgevoerd. Dit zorgt ervoor dat er een goed product komt en voldoet aan de wensen van het bedrijf.

5.1.4.2 API calls

De API calls zijn getest door middel van tests die kunnen worden gemaakt in de IDE. Het programma waarin ontwikkeld is Intellij. Intellij heeft een tool ingebouwd waarmee calls kunnen worden aangemaakt. Dus er kan bijvoorbeeld een call worden gedaan om gebruikers in te loggen. Deze call kan vervolgens ook getest worden. De tool kan checken wat het antwoord is van deze call en wat er in de body staat. Door deze te vergelijken met wat er verwacht wordt kan er gecheckt worden of de call naar behoren werkt.

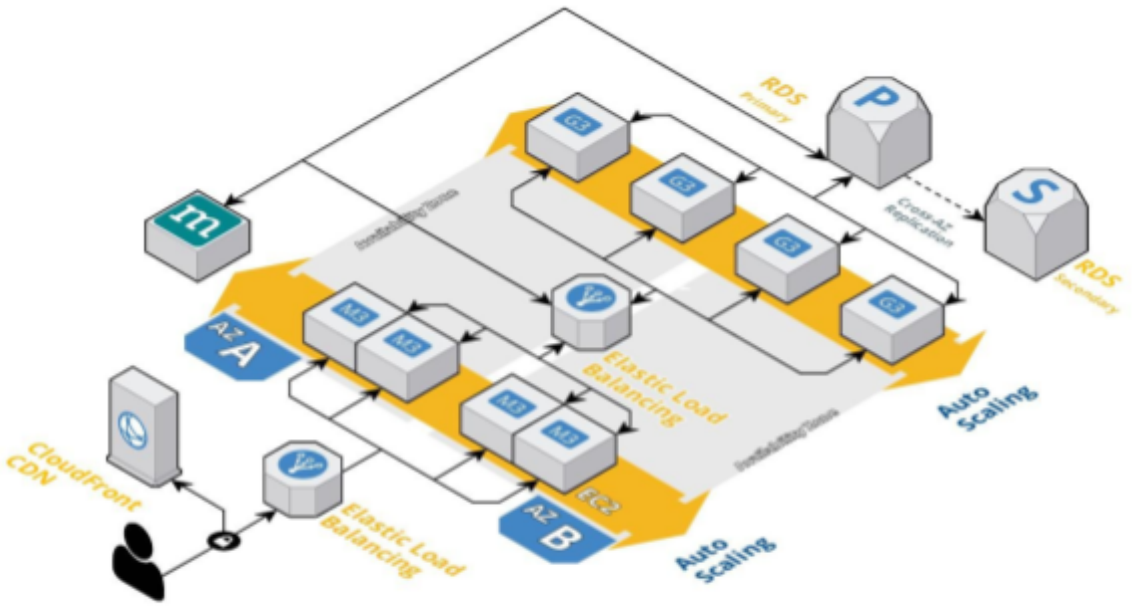
5.1.4.3 Acceptatietest

Om te controleren of de klant tevreden was met het product, is er aan het einde nog een acceptatietest gehouden. Dit houdt in dat het eindproduct is gepresenteerd aan de begeleider van het werk. Vervolgens heeft de begeleider zijn akkoord gegeven met aanvullende feedback. Deze feedback is vervolgens in het eindproduct verwerkt.

5.2 Technische ontwerp

5.2.1 Architectuurplaat

In het ontwerp is er aandacht geweest om te kijken hoe de unreal engine applicatie tot een platform kon worden gemaakt. Hiervoor was een AWS server nodig met Docker containers die automatisch schalen (zie hoofdstuk: conclusie deelvraag 4). Naast de Unreal Engine applicatie war er een platform met een website waarop de Unreal Engine applicatie werd gestreamd.



Afbeelding 13: architectuurplaat platform voor online beurzen

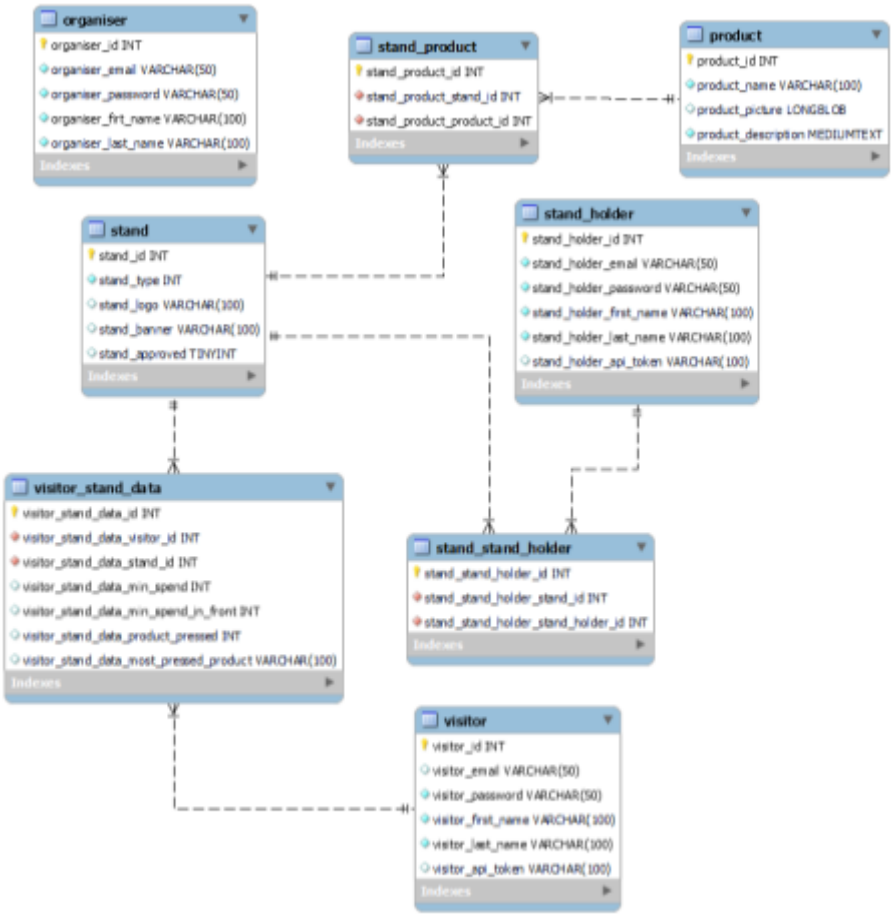
In de afbeelding hierboven wordt het gehele systeem weergeven. Het begint bij de gebruiker die via een load balancer bij het systeem komt. De eerste rij na de load balancer is de M3 server voor de website en de API. Daarop kan de gebruiker inloggen en met het systeem werken. Vervolgens is er

nog een load balancer voor de Unreal Engine applicatie: de G3 server. Voor de bezoekers is er na het inloggen een scherm waarop de beurs wordt gestreamd maar omdat elke gebruiker zijn eigen instance nodig heeft wordt er gebruik gemaakt van een load balancer om er voor te zorgen dat iedere gebruiker zijn eigen instance krijgt en dat de server kan auto scalen. Vervolgens heeft de webserver toegang tot de database met een slave voor het geval er te veel data wordt verwerkt. Dit zal in het begin nog niet meteen nodig zijn maar bij grote aantallen gebruikers zou dit wel wenselijk worden. Ook heeft de applicatie toegang tot de API van Mesibo. Dit is een externe server van Mesibo zelf. Door middel van deze API kunnen de gebruikers met elkaar videobellen en chatten. Als laatst is er nog een CDN server waarop alle extra libraries komen te staan zoals: Bootstrap, JQuery enz.

5.2.2 Database

Er moet ook data worden bewaard. Hiervoor is een ontwerp gemaakt voor een database met relaties. De reden dat er is gekozen voor een SQL database is omdat deze gemakkelijk te onderhouden is, aangezien alles in tabellen staat en geordend is. Daarnaast kan een SQL database gemakkelijk data updaten, invoegen en verwijderen en is de data ook gemakkelijk en snel op te zoeken omdat alles geordend is.

Er is een tabel voor de bezoeker, de standhouder en de organisator. Dit zijn de verschillende rollen die de website kan hebben. Zo kan er gecontroleerd worden wie er inlogt en welke data de gebruiker tot zijn beschikking heeft. Er zijn tabellen voor de producten en de stands. Daarnaast zijn er nog tabellen die data met elkaar verbinden, zoals de producten die met de stands te maken hebben. Hierbij wordt een tabel bijgehouden met de id van een stand en de id van een product. Zo komt er een lijst met producten die bij een stand hoort. Dit geldt ook voor data van een bezoeker van een stand en een lijst met stand en de standhouders. In de data van de stand staat ook een kolom met de data of de stand is goedgekeurd of niet. Als een stand wordt aangemaakt krijg deze kolom een waarde van 2. Zo weet de API dat deze stand nog gekeurd moet worden. De organisatie kan dan vervolgens via de website de stand afkeuren, deze stand krijgt dan een waarde van 0, of goedkeuren, deze stand krijgt dan de waarde 1. Het ontwerp van de database is te vinden in de afbeelding hieronder.

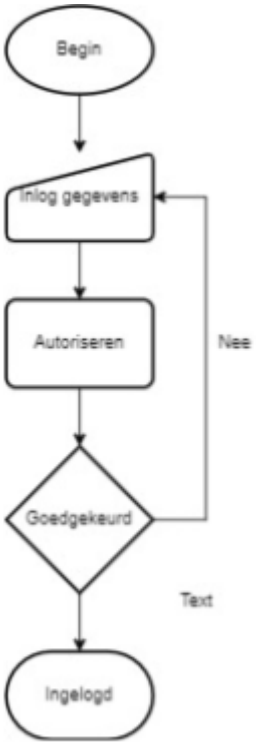


Afbeelding 14: ontwerp database

5.2.3 Flow Diagrams

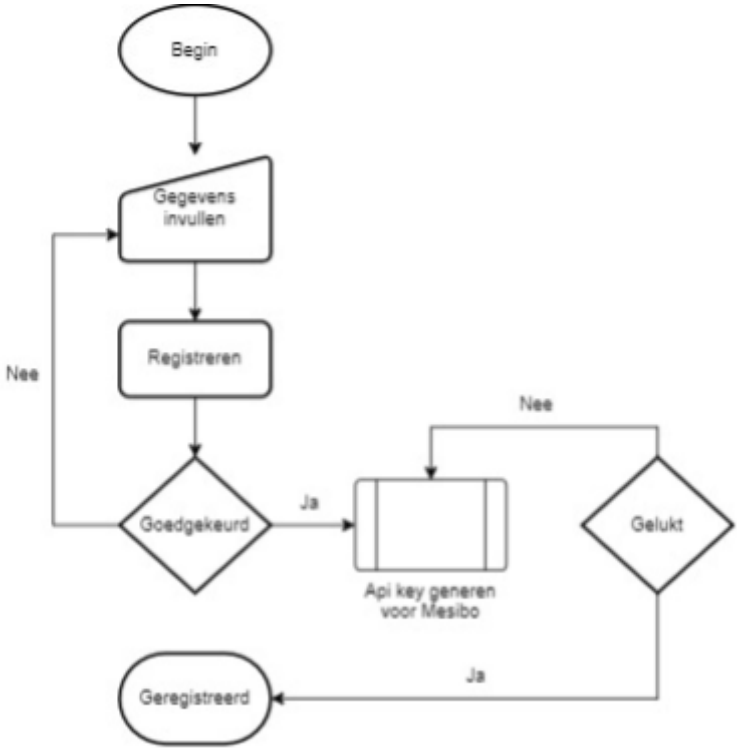
In dit hoofdstuk wordt gekeken naar de flow van het platform. Dit wordt weergegeven in een flow diagram. De flow diagrammen geven aan welke gebruiker door de verschillende pagina's van het platform gaat en welke acties de gebruiker kan uitvoeren.

5.2.3.1 Gebruiker inloggen



Afbeelding 15: flowdiagram gebruiker inloggen

5.2.3.2 Registreren



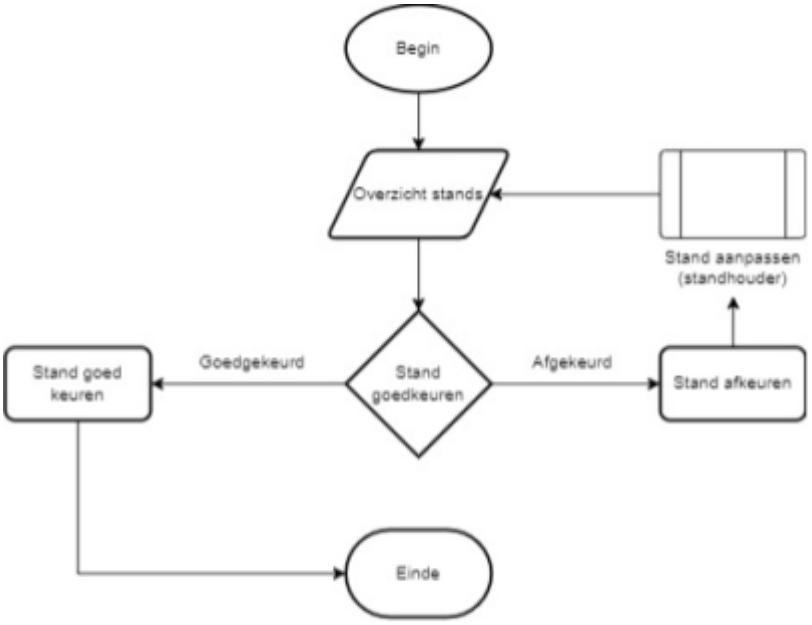
Afbeelding 16: flowdiagram gebruiker registreren

5.2.3.3 Standhouder



Afbeelding 17: flowdiagram standhouder

5.2.3.4 Organisator



Afbeelding 18: flowdiagram organisator

5.2.4 API ontwerp

Hieronder staat een lijst met alle API calls die kunnen worden gedaan met het platform. In de eerste kolom staat het type call. Daarnaast de call die moet worden gedaan. Vervolgens komt er een response status uit die laat zien welke statuscode er terugkomt als het goed gaat. In de laatst kolom staat wat in de body van het request staat als de call is goedgekeurd. Voor de login is gekozen om met een JSON token te werken. De reden hiervoor is omdat er ervaring is met JSON web. Enkele voordelen van JSON web token zijn dat deze stateless is. De token bevat alle informatie die nodig is en het is te gebruiken over meerdere platformen. Mocht er in de toekomst toch een andere front end taal worden gekozen, dan hoeft deze token niet te veranderen.

Type	API call	response status code	body
GET	.../login?id=1	200	JSON token
POST	.../register	201	JSON token
GET	.../visitors	200	JSON lijst met bezoekers
GET	.../stand_data?visitor_id=1&stand_id=1	200	JSON lijst met data van bezoeker voor stand
GET	.../companies	200	JSON lijst met bedrijven
GET	../show_stand?id-1	200	JSON object met stand data
POST	.../make_stand	201	JSON object met stand data
POST	.../make_product	201	JSON object met product data
PUT	.../update_stand	200	JSON object met updated stand data
PUT	.../update_product	200	JSON object met updated product data
PUT	../aprove_stand?stand_id=1	200	N.V.T.
PUT	.../reject_stand?stand_id=1	200	N.V.T.
PUT	.../approve_product?product_id=1	200	N.V.T.
PUT	.../reject_product?product_id=1	200	N.V.T.
API voor Unreal Engine			
GET	…/get_products	200	JSON lijst met producten
POST	.../update_stand_info	200	N.V.T.

Tabel 7: overzicht response code

5.2.5 Error codes

Hieronder is een tabel met de error codes die kunnen voorkomen tijdens het gebruik van het platform en met een mogelijke reden. Daarnaast is uitgelegd wat de code inhoudt.

Error code	Reden	Omschrijving
401	Als een gebruiker een pagina intypt waar hij geen toegang mag	De pagina die de gebruiker probeert te zoeken beschikt over data die niet voor deze gebruiker bedoeld is
404	Zelf een pagina ingetypt	De pagina die de gebruiker zoekt bestaat niet
405	De gebruiker probeert zelf een query te doen	Er wordt een call gedaan om een REST verzoek naar de database te doen maar de methode klopt niet (bijvoorbeeld GET terwijl er een POST moet worden gedaan)
500	De gebruiker klikt op een knop die een verzoek probeert te doen	Er is iets verkeerd gegaan en er kan geen response worden gegeven

Tabel 8: Overzicht fout codes

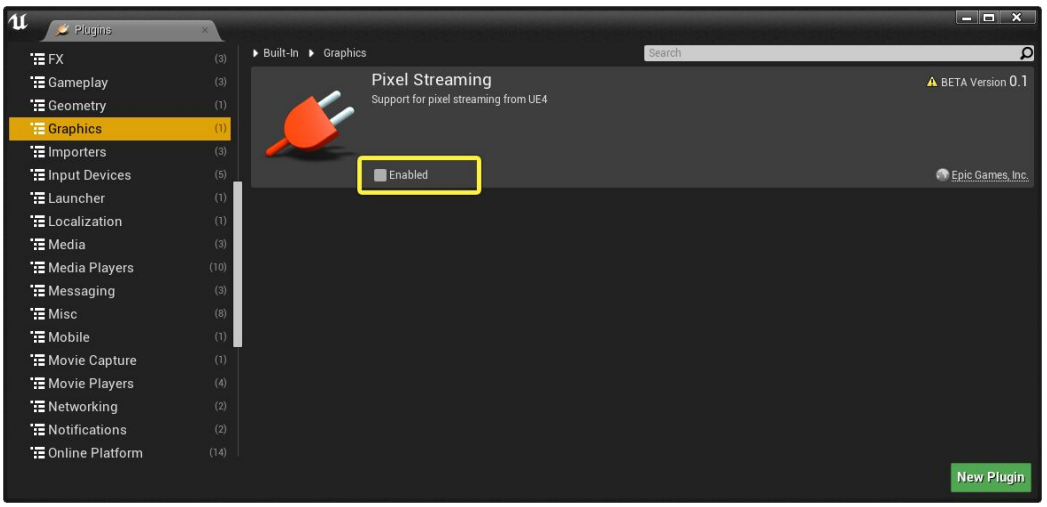
6 Implementatie

In dit hoofdstuk wordt er uitgelegd welke uitdagingen er waren tijdens de implementaties en hoe deze zijn opgelost.

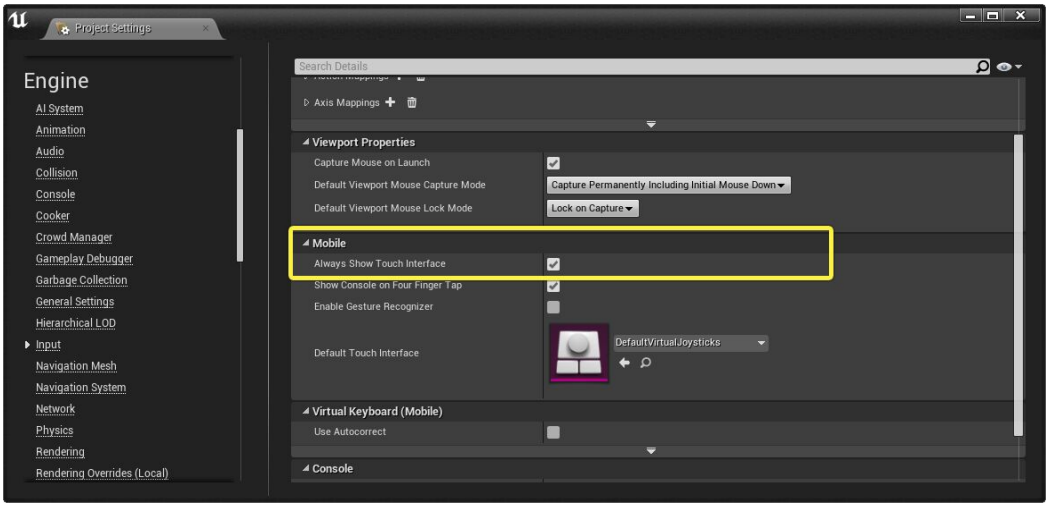
6.1 Unreal Engine

Voor dit project moest er een Unreal Engine applicatie gezien laten worden in het platform. Om dit te realiseren is er gebruik gemaakt van Pixel Streaming van Unreal Engine zelf. Om dit werkend te krijgen moet er in de applicatie eerst zelf opties worden aangevinkt.

Ten eerste moet de Pixel Streaming Plugin worden aangevinkt om er gebruik van te kunnen maken. Vervolgens moet het project opnieuw worden opgestart.

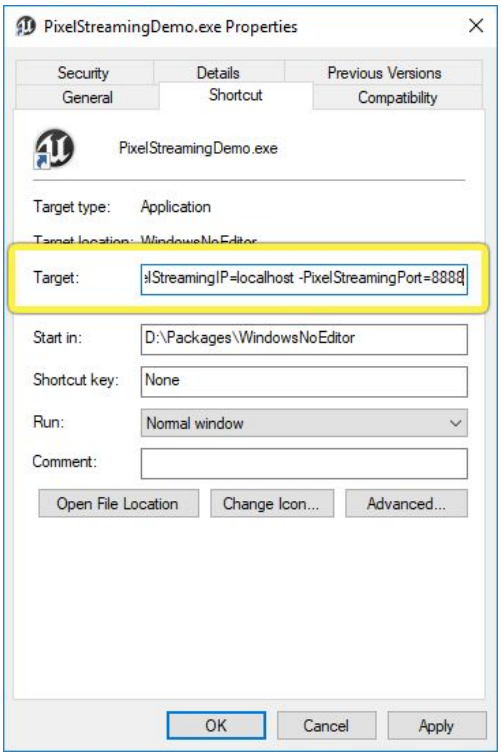


Daarna moet er in de settings de setting worden aangevinkt voor mobile interface. Dit houdt in dat de touch interface altijd te zien is voor mobiele schermen. Als dit niet wordt gedaan is het onmogelijk om voor de mobiel de applicatie te kunnen gebruiken. Dit is echter niet verplicht.



Als laatste instelling is het belangrijk om de volgende parameters mee te geven voor het opstarten van de game: `-AudioMixer -PixelStreamingIP=localhost -PixelStreamingPort=8888.` Vervolgens moet de game worden gepackaged worden voor een

windows (64-bit) systeem. Dit geeft een folder met de naam ‘WindowsNoEditor’ en in deze folder staat de game met Pixel Streaming. Van de game.exe moet er vervolgens een shortcut worden gemaakt zodat de `-AudioMixer -PixelStreamingIP=localhost -PixelStreamingPort=8888` ook bij het opstarten van de .exe kan worden uitgevoerd. Deze tekst wordt achter de target field gezet.



Als laatste moet er een webserver worden gestart zodat er een peer-to-peer connectie kan worden opgezet. Bij het installeren van de Unreal Engine komt standaard een Signaling Server. Deze is gebruikt voor het testen van de Unreal Engine applicatie. Nadat de server en vervolgens de game zijn opgestart, kan de game worden gespeeld via de localhost. Door dit IP adres in een iframe element te stoppen, heb ik in mijn platform een Unreal Engine applicatie gekregen.

6.2 Database

Voor het verwerken van data is gebruik gemaakt van een MySQL database (zie hoofdstuk: 5.2.2 Database). Om de data te benaderen en te wijzigen is er binnen de database gebruik gemaakt van ‘stored procedures’. De reden waarom er voor stored procedures is gekozen: om het onderhoud van de query's te vergemakkelijken en makkelijker om zo de data op te vragen. Tijdens het programmeren was het in het begin lastig om sommige data te verkrijgen. Vooral bij de query’s met meerdere waardes. De waardes waren variabel en werden met een ‘?’ in de query aangegeven om vervolgens de waardes mee te geven in de statement.

```
database.query("INSERT INTO visitor SET?", [{
    visitor_email: email,
    visitor_password: password,
    visitor_first_name: firstName,
    visitor_last_name: lastName,
    visitor_api_token: token
}],
```

Dit zorgde er geregeld voor dat de data die binnen kwam niet overeen kwam met de data die de query verwachtte. Door gebruik van stored procedures te maken was dit verholen. Dit komt voornamelijk omdat er aan de stored procedures data mee kan worden gegeven en deze vervolgens direct in de query wordt gebruikt.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetStandholderByEmail`(IN email varchar(255))
BEGIN
	SELECT * from stand_holder
  WHERE stand_holder_email = email;
END
```

Zoals in dit voorbeeld wordt het e-mailadres meegegeven in de stored procedure. Hierdoor is het makkelijker om de data mee te geven.

Daarnaast is het ook eenvoudiger om de code bij te houden. Als er iets veranderd moet worden, hoeft niet alles in de code te worden aangepast maar alleen in de stored procedure. De manier waarop de stored procedure wordt aangeroepen blijft hetzelfde.

6.3 Docker

Om het platform schaalbaar te maken is er gebruik gemaakt van Docker.

6.3.1 Docker met Node.js en Mysql

6.3.2 Node.js

Om ervoor te zorgen dat het platform schaalbaar kan worden gemaakt is het handig om gebruik te maken van Docker. Door Docker te gebruiken kan er een image worden gemaakt die de website kan maken en installeren. Hierdoor hoeft niet handmatig een nieuwe website te worden gehost maar kan docker dat voor de beheerder doen. Om Docker te gebruiken is een file nodig die zegt wat Docker moet doen als die wordt aangeroepen.

```
FROM node:12.18.1
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
CMD [ "node", "server.js" ]
```

Zo ziet de Dockerfile eruit voor de Node.js applicatie. De eerste regel haalt de Node.js image versie 12..18.1 op. Vervolgens wordt er gedefinieerd op welke plek de app moet gaan draaien. In dit geval is dat de app map die wordt aangemaakt. Vervolgens wordt de package.json file gekopiëerd. In deze file staat welke dependencies Docker moet downloaden en welke file Docker moet worden aangeroepen om de Node.js applicatie te starten. Vervolgens worden alle dependencies gedownload en geïnstalleerd. De ‘Copy’ regel kopieert de rest van de bestanden om de applicatie te kunnen draaien. Hieronder vallen de javascripts, HTML pagina’s en de API’s. Daarna wordt de port 5000 opengezet zodat de applicatie hierop te vinden is. Als laatste wordt de regel ‘node server.js’ uitgevoerd die de applicatie opstart.

6.3.3 Mysql

Om de applicatie te kunnen gebruiken is er ook een database nodig. In dit geval wordt MySQL gebruikt. De voornaamste reden waarom MySQL wordt gebruikt is omdat hiermee de meeste ervaring is.

Ook voor MySQL is er een Dockerfile nodig om de database image op te halen en te draaien:

```
FROM mysql
COPY ./testdump.sql /docker-entrypoint-initdb.d/
EXPOSE 3306
```

In de eerste regel wordt de MySQL image opgehaald en geïnstalleerd. Vervolgens wordt de file testdump.sql gekopieerd in de 'docker-entrypoint-initdb.d. Dit houdt in dat de regels die worden beschreven in de testdump file worden uitgevoerd als de database wordt aangemaakt. In deze file staat de sql code die ervoor zorgt dat de database gevuld wordt met tables en stored procedures. Ook kan er alvast wat data worden meegegeven.

```
CREATE TABLE `product` (
                        `product_id` int unsigned NOT NULL
AUTO_INCREMENT,
                       `product_name` varchar(100) NOT NULL,
                       `product_picture` longblob NOT NULL,
                      PRIMARY KEY (`product_id`)
);
```

Dit is een voorbeeld van een table die wordt aangemaakt en de producten bevat.

```
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` PROCEDURE
`GetStandholderByEmail`(IN email varchar(255))
BEGIN
 SELECT * from stand_holder
 WHERE stand_holder_email = email;
END ;;
DELIMITER ;
```

Dit is een voorbeeld van een stored procedure die een standhouder teruggeeft aan de hand van een e-mail.
Als laatst wordt de database op port 3306 gedraaid.

6.3.4 Docker compose (Node.js en MySQL)

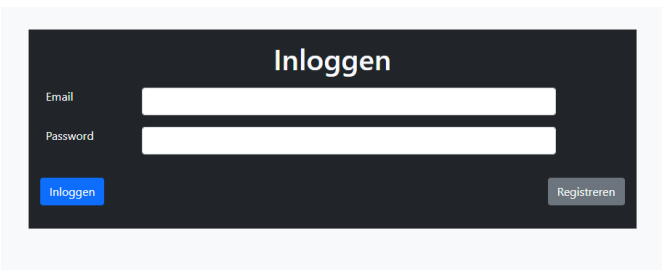
Er zijn nu twee losse docker containers die draaien. Echter kan het wenselijk zijn dat ze allebei worden gestart op hetzelfde moment. Dit wordt mogelijk met een Docker compose file. Deze file zorgt ervoor dat er meerde Dockerfiles kunnen worden gestart en dus meerder containers tegelijk kunnen worden gestart.

```
services:
 db:
   build: ./db
   command: --default-authentication-plugin=mysql_native_password
   environment:
```

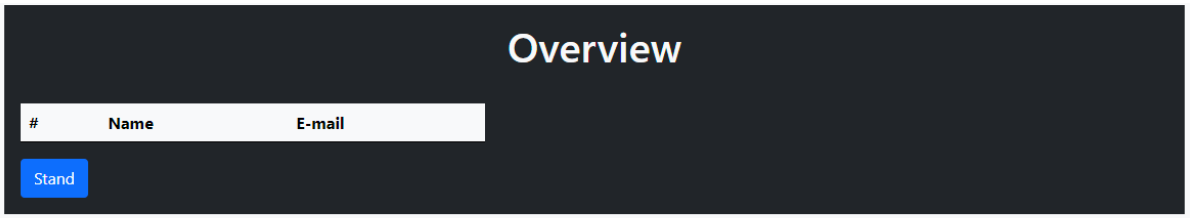
```
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: test
    restart: always
    ports:
      - "3306:3306"
    expose:
      - 3306
  web:
    build: ./
    environment:
      MYSQL_DATABASE: test
      MYSQL_USER: root
      MYSQL_PASSWORD: password
      MYSQL_HOST: db
    ports:
      - "5000:5000"
    depends_on:
      - db
```

In de eerste regels wordt vastgelegd wat er gedaan moet worden voor de database. In de volgende regel wordt er verwezen naar de folder waarin de Dockerfile staat voor de database. In dit geval staat de database in de folder ‘db’. De volgende regel zorgt ervoor dat er een connectie kan worden opgezet in een Docker container. De nieuwste database (die wordt opgehaald in de Dockerfile) ondersteunt nog niet de manier van authenticatie die in Docker wordt gebruikt. Door deze regel te gebruiken kan er toch data worden verzonden op de ‘oude’ manier. De volgende regel geeft de inloggegevens weer van de database. Vervolgens wordt er in de volgende regel gezegd dat de database opnieuw moet opstarten, mocht er wat zijn gebeurd. Dit zorgt ervoor dat de data toegankelijk blijft mocht er iets mis zijn gegaan. Vervolgens wordt de port ‘3306’ van de server opengezet naar ‘3306’ van de Docker container zodat er toegang is tot de database.

De regel ‘web:’ geeft weer wat Docker moet doen voor de web applicatie. Ook hier geeft de volgende regel aan waar de Dockerfile staat; in dit geval voor de Node.js applicatie. In dezelfde map staat de Dockerfile als docker-compose.yaml. Vervolgens worden de enviroment waardes gezet zodat de applicatie weet welke waardes er nodig zijn om in te loggen. Door het op deze manier te doen, hoeven er geen daadwerkelijke inloggegevens te worden gebruikt in de code van de Node.js applicatie. Het resultaat is dat een dataminer niet meteen een string ziet met de inloggegevens. Vervolgens wordt de poort ‘5000’ van de server opengezet voor de poort ‘5000’ van de Docker container. Hierdoor kan een gebruiker naar poort 5000 van de server gaan om de website bezoeken. De laatste regel geeft aan dat de website afhankelijk is van de database er dat er dus gewacht moet worden totdat de database is gemaakt.



Dit is uiteindelijk de inlogpagina, zoals deze wordt getoond op: 85.147.240.108:5000. Dit is dus gedaan door de Docker container. Vervolgens kun je inloggen en verschijnt er dit scherm



Dit geeft dus aan dat de applicatie kan samenwerken met de database en dat het werkt.

6.4 Videobellen

Een onderdeel van het platform is dat er communicatie nodig is tussen de gebruikers. Dit wordt gedaan doormiddel van een videobel API. In eerste instantie was er voor gekozen om Mesibo te gebruiken maar tijdens het programmeren van de API bleek dat deze toch niet goed werkt. Er kon steeds geen verbinding met de API worden gemaakt en op de dashboard kwam hetzelfde voor. Telkens kwam er een bericht dat er geen verbinding kon worden gemaakt en dat de verbinding moest worden gecheckt. Dit heeft meerdere dagen geduurd en er kwamen nog steeds dezelfde meldingen. Daarom is toch besloten om te zoeken naar een bestaande applicatie die wel werkt. Uiteindelijk is er voor gekozen om videosdk.live te gebruiken. Het grootste voordeel van deze API is dat er weinig aanpassingen nodig zijn.

6.4.1 Mesibo

Mesibo is na het onderzoek gekozen om te gebruiken. Voor deze API moest er eerst een gebruiker worden aangemaakt om een token voor deze gebruiker te genereren. Met deze token kon de gebruiker vervolgens gaan videobellen en chatten. Om een token te kunnen genereren moest er een API call worden gemaakt naar Mesibo. Deze zag er als volgt uit:

```
https://api.mesibo.com/api.php?token=cn9cvk6gnm15e7lrjb2k7ggggax5h90n5x7dp4sam6kwitl2hmg4cmwabet4zgdw&op=useradd&appid=com.mesibo.firstapp&addr=123
```

In deze URL staat een token en deze token is van te voren aangemaakt en opgeslagen in de applicatie. Vervolgens is er een appid, deze appid kan elke naam bevatten die de gebruiker wil gebruiken. Als laatst staat er in de URL een addr en deze kan van alles zijn om een gebruiker te kunnen identificeren. In dit geval is er voor gekozen om het e-mail te gebruiken aangezien dit de identificatie is van de gebruiker. De API call werd aangemaakt door middel van een request te maken, deze zag er als volgt uit:

```
request(uri, {json: true}, function (err, respond, body) {
    if (err) {
        // Something went wrong in requesting a key for the user
        res.sendStatus(401);
    }
    var userToken = body.user.token;
});
```

Vervolgens komt er een JSON response uit die een token bevat voor de gebruiker en deze zag er zo uit:

```
{
"user": {
      "uid":"5302",
"token":"cn9cvk6gnm15e7lrjb2k7ggggax5h90n5x7dp4sam6kwitl2hmg4cmwabet4zgdw"
},
"op":"useradd",
"result":true
}
```


De token werd eruit gehaald en vervolgens meegenomen in de query om de gebruiker te registreren in het systeem. Helaas kwam er tijdens het testen hiervan voren dat de API niet goed werkte. Bij het maken van de API call kwam er telkens een error dat er geen verbinding kon worden gemaakt. Als er werd gekeken op het dashboard van Mesibo kwam daar dezelfde melding. Daarom is er gezocht naar een andere oplossing en de keuze is gevallen op videosdk.live

6.4.2 Videosdk

Om gebruik te maken van Videosdk is er een lege html pagina aangemaakt zodat de sdk deze pagina kan vullen. Vervolgens is er een js file aangemaakt om hier de skd in te laden. Daarna is er een script toegevoegd die ervoor zorgt dat de videobel pagina wordt geladen. Ook worden er een paar configs gezet voordat de pagina wordt geladen. Dit is in de vorm van een JSON bestand. Een paar variabelen die door de applicatie zelf worden ingesteld zijn de naam van de gebruikers en de terugstuur link. De naam wordt uit de local storage gehaald en gezet op de volgende manier: `name:`

```
JSON.parse(localStorage.getItem("User")).first_name + " " +
JSON.parse(localStorage.getItem("User")).last_name,
```

Door middel van een terugstuur link is er een optie om de gebruiker door te sturen naar een scherm als de gebruiker de pagina verlaat. De gebruiker wordt in dit geval weer teruggestuurd naar de login pagina.

```
redirectOnLeave: "http://85.147.240.108:3000"
```

Op de login pagina is een functie ingebouwd die de gebruiker naar de juiste pagina stuurt als deze al is ingelogd. Dus als een bezoeker die is ingelogd naar deze pagina gaat, wordt de bezoeker doorgestuurd naar de pagina met de virtuele beurs. Als laatste wordt de de pagina geïnitialiseerd met de configuratie die hiervoor is gemaakt.

```
meeting.init(config);
```

Daarna wordt de pagina toegevoegd aan de lege html pagina:

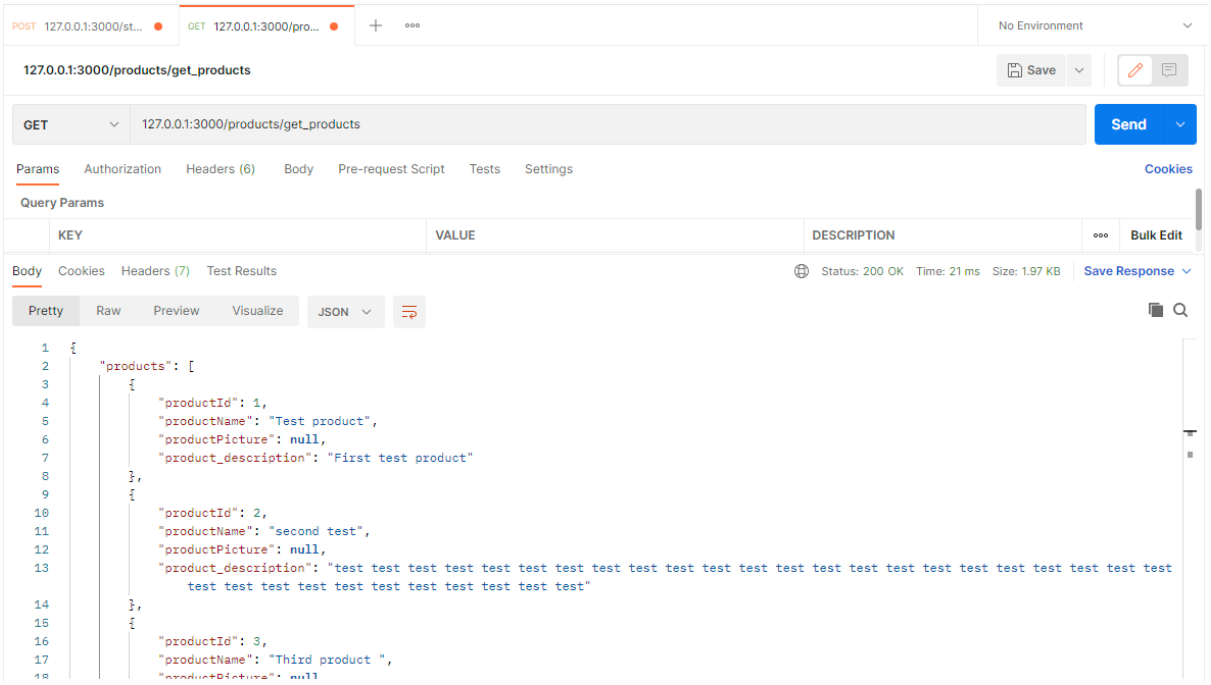
```
script.src =
"https://sdk.videosdk.live/rtc-js-prebuilt/0.1.23/rtc-js-prebuilt.
js";
document.getElementsByTagName("head")[0].appendChild(script);
```

Als dit is gedaan kunnen verschillende personen met elkaar videobellen.

6.5 Virtuele beurs via API

De virtuele beurs moet zowel gegevens kunnen ophalen als verzenden naar de API. Dit is mogelijk door een paar API calls te maken naar de API. Zo kan de API de producten laten zien de de beurs moet inladen. Dit wordt gedaan door de call: /stands/get_stands. Door deze call uit te voeren kan de API alle beschikbare producten ophalen en deze later inladen. De uitkomst van deze call is een JSON

lijst met de data van de producten, zoals: product id, product naam, product afbeelding en product beschrijving. Hieronder een afbeelding met de API call in Postman die laat zien wat eruit komt.


127.0.0.1:3000/products/get_products
GET
127.0.0.1:3000/products/get_products
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 21 ms
Size: 1.97 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"products": [
{
"productId": 1,
"productName": "Test product",
"productPicture": null,
"product_description": "First test product"
},
{
"productId": 2,
"productName": "second test",
"productPicture": null,
"product_description": "test test test

6.6 API test

Om ervoor te zorgen dat de API's werken zijn er test geschreven. Deze test zijn gedaan in Intellij door middel van een http file. In deze file kunnen API calls worden gezet om deze vervolgens uit te voeren. Daarnaast kan er ook een test worden geschreven om te kijken of de uitkomst klopt met wat er verwacht wordt. Dus als er een stand wordt aangemaakt dat de response code ook code 201 (created) is. Deze test zien er als volgt uit:

```
###
## Login test succes
POST http://localhost:3000/auth/login
Content-Type: application/json

{
 "email": "qq@qq.qq",
 "password": "123"
}
> {%
client.test("Request executed successfully", function() {
 client.assert(response.status === 200, "Response status is not
200");
});
%}
```

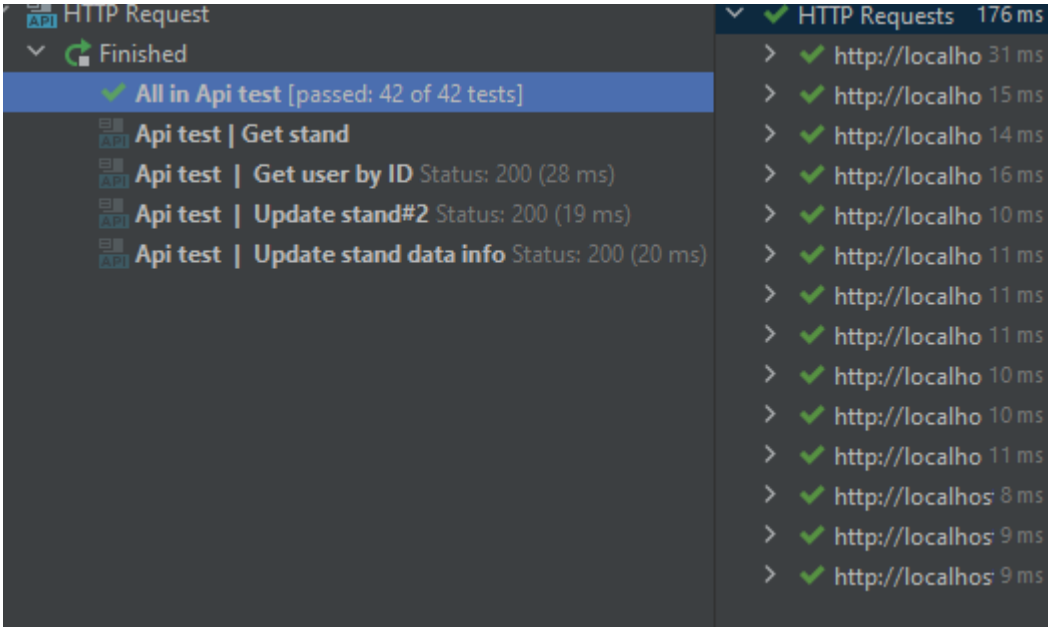
Dit is een test om te kijken of een gebruiker kan inloggen. Eerst wordt er een API call gemaakt. Vervolgens wordt de content-type gezet. Dit hoeft alleen gezet te worden bij POST en PUT calls. Daaronder kunnen de waardes worden meegegeven die nodig zijn om de call te maken. In dit geval dus de gebruikersnaam en wachtwoord van de gebruiker. Vervolgens wordt er in de test gekeken of de response code ook daadwerkelijk de code is die er verwacht kan worden. Als de log in slaagt wordt er een 200 code verwacht en is de test geslaagd. Zo kan er ook getest worden op calls die niet goed zijn. Dus als er geen gebruiker is gevonden met deze inloggegevens geeft de server een 401 terug. Als er in de test dan ook een 401 wordt verwacht is de test alsnog geslaagd.

Op deze manier kunnen ook GET calls worden getest. Deze worden op dezelfde manier opgesteld en getest. Deze testen zien er als volgt uit:

```
### Get Products
## Get all products
GET http://localhost:3000/products/get_products

> {%
client.test("Request failed", function () {
   client.assert(response.status === 200, "Response status is not
200");
})
%}
```

Hiermee worden dus de tests uitgevoerd en gecontroleerd of de API calls werken. Hieronder de uitslag van de testen voor het platform:



Hieruit is op te maken dat alle test zijn geslaagd en dus de API calls naar behoren werken.

7 Conclusie & aanbevelingen

7.1 Authenticatie

In dit project is er gebruik gemaakt van een zelfgemaakte authenticatie met JSON webtoken. Hiervoor moet de gebruiker eerst zichzelf registreren. Als dit PoC een product zou worden is het makkelijker als er ook gebruik gemaakt kan worden van OAuth. OAuth staat voor Open Authorization en hiermee kan de gebruiker ook inloggen door gebruik te maken van de inloggegevens van andere applicaties zoals Google of Facebook. Door OAuth te gebruiker hoeft de gebruiker zichzelf niet te registreren en kan de gebruiker de inloggegevens gebruiken van de andere applicaties.

7.2 Videobel applicatie

Tijdens het maken van de videobel integratie is er achter gekomen dat Mesibo toch niet goed werkte. Als er een API call werd gemaakt werd de connectie geweigerd omdat er geen internetverbinding was. Deze melding kwam ook op het dashboard van Mesibo zelf. Vervolgens is er een andere applicatie gebruikt om dit te verhelpen, maar deze is niet geschikt voor een productie klaar product. Het grote nadeel van deze applicatie is dat het een applicatie is die al gemaakt is en weinig aanpassingen kan bieden. Hierdoor past het niet bij het uiterlijk van de rest van de applicatie. Daarnaast wordt het lastig om één op één gesprekken te houden tussen de gebruikers aangezien deze applicatie alleen de mogelijkheid biedt om in groepen te bellen.

7.3 Database ontwerp

In het ontwerp van de database worden de afbeeldingen en video's opgeslagen in de database. Dit kan werken voor een kleine applicatie zoals een PoC maar voor een grotere applicatie is het verstandig om dit anders op te lossen. Een goede oplossing kan zijn om de bestanden te uploaden naar een server en vervolgens het pad naar dit bestand opslaan in de database. Hierdoor wordt de database aanzienlijk kleiner dan als de bestanden worden bewaard in de database. Daardoor blijft de database sneller werken aangezien de database dan minder data hoeft op te zoeken en te versturen. De applicatie kan dan weer het bestand downloaden van de server en laten zien aan de gebruiker.

Literatuurlijst/Bibliografie

ArunKumar@TWC. (2021, September 1). What is a Cold Boot Attack and how can you stay safe? The Windows Club. https://www.thewindowsclub.com/cold-boot-attack

Development images vs. runtime images | Unreal Containers. (n.d.). Unreal Containers. Retrieved July 20, 2021, from https://unrealcontainers.com/docs/concepts/image-types

Encrypting Data at Rest on Servers: What does it get you? – HLN Consulting, LLC. (2016, July 25). Www.Hln.Com.

https://www.hln.com/encrypting-data-at-rest-on-servers-what-does-it-get-you/index.html

Interactive Cloud Streaming Platform - PureWeb. (n.d.). PureWeb. Retrieved July 28, 2021, from https://www.pureweb.com/interactive-3d-cloud-streaming#Tech

Pixel Streaming Overview. (n.d.). Unreal Engine Documentation. Retrieved July 8, 2021, from https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/PixelStreaming/PixelStreamingOverview/

Press release. (2018, July 17). Amazon.Com, Inc. - Press Room. Retrieved November 29, 2021, from https://press.aboutamazon.com/news-releases/news-release-details/epic-games-goes-all-aws

Pricing. (n.d.). Mesibo. Retrieved June 30, 2021, from https://mesibo.com/pricing/#what

R. (2019, March 25). Encryption in-transit and Encryption at-rest – Definitions and Best Practices. Ryadel.
https://www.ryadel.com/en/data-encryption-in-transit-at-rest-definitions-best-practices-tutorial-guide/
S. (2020, July 9). Protecting the three states of data. Sealpath.
https://www.sealpath.com/protecting-the-three-states-of-data/

Secure Memory Encryption (SME) - x86 - WikiChip. (n.d.). Wikichip. Retrieved June 10, 2021, from https://en.wikichip.org/wiki/x86/sme#Secure_Memory_Encryption

Sendbird Pricing. (2021, March 26). Sendbird. https://sendbird.com/pricing

Sutevski, D. (2019, December 23). Data at Rest vs. Data in Transit: What's the Difference? Entrepreneurship In A Box.
https://www.entrepreneurshipinabox.com/19570/data-at-rest-vs-data-in-transit-whats-the-difference/

Techopedia. (2013, August 15). Data in Use. Techopedia.Com. https://www.techopedia.com/definition/29515/data-in-use

Wat zijn persoonsgegevens? (n.d.). Autoriteit Persoonsgegevens. Retrieved September 8, 2021, from https://autoriteitpersoonsgegevens.nl/nl/over-privacy/persoonsgegevens/wat-zijn-persoonsgegevens

Wikipedia contributors. (n.d.-a). Docker (software). Wikipedia. Retrieved July 8, 2021, from https://en.wikipedia.org/wiki/Docker_(software)

Wikipedia contributors. (n.d.-b). Email encryption. Wikipedia. Retrieved June 3, 2021, from https://en.wikipedia.org/wiki/Email_encryption

Wikipedia contributors. (n.d.-c). Encryption. Wikipedia. Retrieved June 3, 2021, from https://en.wikipedia.org/wiki/Encryption

Wikipedia contributors. (n.d.-d). Traversal Using Relays around NAT. Wikipedia. Retrieved October 3, 2021, from https://en.wikipedia.org/wiki/Traversal_Using_Relays_around_NAT

Wikipedia contributors. (2021, October 2). Unreal Engine. Wikipedia. https://en.wikipedia.org/wiki/Unreal_Engine

Wikipedia-bijdragers. (n.d.). WebRTC. Wikipedia. Retrieved June 17, 2021, from https://nl.wikipedia.org/wiki/WebRTC

Wolfrey, T. (n.d.). STUN servers: A Quick Start Guide. Singlecomm. Retrieved June 17, 2021, from https://help.singlecomm.com/hc/en-us/articles/115007993947-STUN-servers-A-Quick-Start-Guide

Bijlage 1: Interview met Leonie Pamboer

Leonie Pamboer werkt bij MASTERS HQ en is hier werkzaam als Head of events. Zij organiseert beurzen zoals Masters of LXRY, een groot event voor luxe producten. Met haar is een interview gehouden over hoe zij zou denken hoe een virtuele beurs eruit zou moeten zien.

1. **Hoeveel personen komen er gemiddeld bij een beurs die u organiseert?**
   Bij de 'Masters of LXRY' en de Masters Expo die wij organiseren komen er ongeveer tussen de vijftig en de 55 duizend bezoekers over de gehele beursperiode. Dat kun je natuurlijk vertalen naar een gemiddelde over de hele dag. Het is zo dat bijvoorbeeld een maandag en een vrijdag rustiger is dan een zaterdag of zondag.
2. **Over hoeveel gebruikers moet je gemiddeld spreken op de drukste dag?**
   Dan zou ik zeggen vijftienduizend bezoekers.
3. **En is dat dan over de hele dag?**
   Nee dat komt en gaat.
4. **Een online platform, heeft u daar al een idee bij?**
   Ja, wat wij eigenlijk maken met Arne (eigenaar bij het bedrijf waar stage wordt gelopen) is een inbetween, hè. Waarmee we een virtuele plattegrond maken en niet een volledige virtuele beurs dat al beschikbaar is. Als ik dat wel zou willen dan is het denk ik, hoe makkelijk het is om informatie te krijgen per standhouder, heel belangrijk.
5. **Dat staat dus eigenlijk centraal, de data die een stand kan genereren?**
   Ik denk wel als je een echte digitale beurs organiseert, dus eentje waar niet iedereen fysiek aanwezig is, dat dat belangrijk is maar is dat eigenlijk de vraag? Is het echt een digitale beurs want er is geen fysieke of is het en/en?
   Nou voor de opdracht gaat het echt om een virtuele beurs omdat het nog niet bekend is hoelang deze crisis nog doorgaat. Dus dan gaat het echt om dat je gewoon van je computer er doorheen kan lopen en bij stands kan binnenlopen en dan op dingen kan gaan klikken.
   Ja, dan is het echt het allerbelangrijkste dat je eigenlijk de exposanten (de standhouders) de mogelijkheid hebben om hun content erin te kunnen zetten op een makkelijke manier en dat een bezoeker op een hele makkelijke manier daar toegang tot heeft en misschien wel dingen kan opslaan of naar zich kunnen mailen.
6. **Dus eigenlijk is het meer dat ze ook feedback kunnen geven aan een bezoeker die er is geweest en van ze hebben gezien, dat hierop hebt geklikt. Wil je daar meer informatie over?**
   Ja dat zou kunnen, maar bijvoorbeeld ook gewoon dat ik, als bezoeker, daar ben geweest digitaal, een leuk gesprek heb gehad dat ze zeggen: een knopje hier links is een brochure, die kunt u opslaan en thuis nog een keer bekijken, bijvoorbeeld.
7. **En voor z'n standhouder zijn daar meestal voorgebouwde stands voor, zodat ze maar een paar dingen hoeven aanpassen of moeten ze de hele stand zelf kunnen aanpassen. Wat zou de voorkeur hebben?**
   Het allerliefst customizen ze hun stand, uiteraard. Want je wilt opvallen ten opzichte van een ander. Er zijn ook heel veel grote bedrijven die hebben een brandbox,

waarin precies staat hoe, wat op welke manier gebruikt mag worden. Waar bijvoorbeeld stomme dingen instaan zoals: van die kleur groen moet je absoluut nooit gebruiken, want dan lijkt het teveel op de concurrent. Of dit lettertype en geen andere lettertype. Dus ja, veel merken hebben daar wel regels over. Daarom, hoe meer je kunt customizen hoe beter het is. Want ik denk ook namelijk als je kijkt naar een soort uniforme bouw zou doen waar alleen de logo's de kleur en de filmpje wisselt. Dan ben je na acht stands uitgekeken. Dan interesseert het je niet meer, dus ja, als er dan 75 stands zijn, dan is het wel een opgave.

8. **Dus het is wel belangrijk dat bedrijven zo veel mogelijk opties kunnen bieden om zo veel mogelijk te kunnen customizen?**
   Ja, of je moet misschien denken in modules. Of modulair, dus dat ze dan pakket een en vier kunnen kiezen en vier is dan uitgebreid en een is basic. Maar met een digitale beurs moet het in een oogopslag interessant zijn en moet het er mooi uitzien en anders zijn als de voorganger.
9. **Wat zijn dan de meest voorkomende dingen die ze dan customizen, ze hebben een eigen logo, een eigen brochure, wat kan ik er nog meer bij verwachten?**
   Promofilmpje en als dat niet werkt, misschien twee plekken waar ze een advertentie visual kunnen plaatsen. Zodat ze in een oogopslag een sfeerbeeld kunnen creëren, zoals in een advertentie in een magazijn plaatst.
10. **Wat zijn zoal de gemiddelde gegevens van een bedrijf welke opgeslagen moeten worden?**
    Als ik denk aan een digitale beurs dan moet er ergens een informatieknop zitten waar gewoon je basic info staat. Dus: je zes regels over het bedrijf, een website, een e-mail adres en een contactpersoon. Dat en daarnaast zie ik dan voor me dat er ergens een brochure te openen is of iets te openen is waarmee meer informatie te verkrijgen is over welke producten er zijn of een product catalogus of een knopje waar de lancering van het nieuwe product in zit, waar misschien wel een aftel in zit tot de lancering.
11. **Voor de gebruiker zelf moet die zich meestal aanmelden voor z'n beurs als bezoeker zijnde of gaat het op een uitnodiging of hoe moet ik dat zien?**
    Dat verschilt heel erg per beurs. Bij onze beurs zou het zijn dat 95% van de kaarten worden uitgegeven door onze standhouders, dus bij hun stand krijgen ze een kaartenpakket en daarmee nodigen zij hun relaties uit. En vijf procent van de kaarten worden nog gewoon verkocht aan de deur.
12. **Van een normale bezoeker, verzamelen jullie daar ook gegevens over?**
    Nee, minimaal. De vijf procent die bij ons wat koopt, dat is allemaal e-tickets dus die gegevens hebben wij wel. Een e-mailadres in ieder geval. En voor onze standhouders, zij nodigen hun relatie uit en hebben geen toegang tot die informatie.
13. **Dus als een bezoeker gewoon standaard gegevens hoeft in te vullen en daarmee toegang krijgt op de site zou dat in principe voldoende moeten zijn?**
    Ja, je zou bijvoorbeeld met codes kunnen werken i.p.v. e-tickets. Dat zou ik wel doen want het komt vaak voor dat een bedrijf extra kaarten bij willen kopen, ze krijgen namelijk kaarten bij hun stand. Maar het moet wel een verkoopbaar iets zijn. Een soort unieke registratie per kaart.
14. **En voor jullie als organisatoren, lopen jullie dagelijks rond of wat zijn jullie (normale) taken tijdens een beurs?**

Naast dat wij eigenlijk alle klanten, dus alle standhouders, willen spreken: gaat alles goed? Mis je iets? Dan hebben wij ook nog altijd een uitgebreid entertainment programma die wij moeten verzorgen: artiesten, bands, opbouw, afbouw. We hebben nog evenementen tijdens de beurs: een zakenlunch, een netwerkmoment, dat soort dingen. De roldeur gaat niet meer omhoog, een hostess is ziek, de scanner bij de entree doet het niet meer, grote klant is z'n kaartje kwijt. Ja, verzin het maar.

15. **Dus voor een digitale beurs zouden jullie eigenlijk ook een stukje beheer willen hebben?**
    Als wij een digitale beurs zouden verzorgen moet er iemand 24/7 stand-by staan die dingen kan oplossen.
16. **U had het net ook over communicatie met klanten, hoe ziet u dit voor zich in een virtuele beurs?**
    Ik denk dat het heel erg afhangt van de gemiddelde leeftijd van de bezoeker. Dus bijvoorbeeld bij onze beurs is het leeftijdsgemiddelde iets hoger. Deze zit boven de 40 jaar. Dat zullen mensen zijn die eerder een chat app zullen gebruiken dan een videobel app. Onder de 40 is het misschien een ander verhaal. Ik denk dat het een goede feature zou zijn als je bijvoorbeeld een echte B To B beurs (business to business) hebt en alle vertegenwoordigers elkaar kennen. Dat is het een goed idee, maar ik denk als je veel B to C (business to consumer) hebt dat je beter niet kan doen. Dat is wat laagdrempeliger.
17. **Er moet dus een optie zijn voor het videobellen?**
    Nou nee, niet voor alle beurzen. Dat denk ik echt niet, dat het voor alle beurzen geschikt is. De bezoeker, consument, hoe je het ook wilt noemen, moet je het zo makkelijk mogelijk maken en als ik een vraag wil stellen, dan wil ik even een vraag stellen. Als ik daarvoor moet inbellen vind ik het al een gedoe. Ik wil gewoon even iets vragen en antwoord kunnen krijgen. Dat is het meest ideale voor een bezoeker.
18. **Dus dan is eigenlijk het minimale dat je kunt chatten, maar het zou mooi zijn voor sommige beurzen dat er een uitbreiding is met videobellen.**
    Ja. Ik kan me ook voorstellen dat het misschien relaxed is dat je bijvoorbeeld op een stand komt en dat er een echt filmpje of een digitaal in elkaar geknutseld dingetje is, waarin je kort, namens de standhouder, welkom wordt geheten.
19. **Een bezoeker, hoeft die eigenlijk alleen maar een scherm te zien waarmee hij kan registreren, inloggen en dan meteen al naar de beurs kan gaan? Of moet er ook nog iets van een overzicht komen over welke bedrijven er zijn, of wordt dat alweer teveel?**
    Nee, dat is voor een bezoeker wel handig. Je hebt zeker bijvoorbeeld bij een beurs, zoals die van ons, waar 95% wordt uitgenodigd door een exposant, als zij dan bij de beurs komen zullen ze waarschijnlijk ook naar die exposant willen. Dus in plaats van een hele beurs door te zoeken is het wel ideaal als zij gewoon even in een zoekbalk of wat dan ook, binnen een categorie lijstje, de exposant kunnen uitzoeken die hun uitgenodigd heeft en dan rechtstreeks daar naartoe gaan.
20. **Zou alleen een plattegrond ook al genoeg kunnen zijn zodat ze snel weten waarheen ze moeten lopen of zou een link ernaar toe veel beter zijn?**
    Ja dat zou ik wel adviseren. Je weet zelf dat mensen digitaal lui zijn. Ik wil gewoon naar het bedrijf en als ik daar ben geweest ga ik eens kijken om mij heen. Ik ga

bijvoorbeeld per categorie nog even kijken welke merken zijn er en dan klik ik iets aan. Dat zou ik als bezoeker wel het handigst vinden.

21. **Voor een bedrijf, wat voor soort schermen zouden die kunnen verwachten? Ik neem aan ook gewoon een inlogscherm en dan denk ik ook zelf aan een pagina waar ze de stand kunnen aanmelden met een formulier van: voeg hier je logo in en dat soort dingen. Daarnaast denk ik aan een scherm waarin ze artikelen kunnen laden. En dan misschien nog een scherm met een overzicht van de gebruikers die bij hun zijn geweest met de gegevens van de stand (waarop hebben ze geklikt enz.). Mis ik daar nog wat?**
    Ja, voor een exposant moet het mogelijk zijn om hun eigen content up te loaden. Dat is belangrijk. Ze willen inderdaad toegang tot de gegevens van de mensen die hun stand bezocht hebben. Ik denk dat dat eigenlijk de twee belangrijkste dingen zijn.
22. **Voor jou zelf als organisator, denk ik: alleen een lijst met bedrijven die er zijn en ook dat je daarmee kunt communiceren. Wilt u dan nog meer informatie zien of andere pagina's? Of zijn dat wel de hoofdzaken?**
    Ik denk dat voor de organisator het belangrijk is dat we de mogelijkheid hebben om bij noodgevallen de content plat te kunnen leggen van standhouders. Dus stel je voor dat er iemand voor kiest om een gek filmpje te plaatsen, dat wij dat kunnen stoppen of kunnen verwijderen. Dat is niets anders dan met een fysieke beurs. Standhouders leveren ook bij ons standontwerpen in en wij keuren dit goed. Dus als wij zeggen dit is niet mooi of dit past niet, dan sturen wij dit terug. En bij de digitale variant zou je dat eigenlijk ook wel moeten kunnen. Verder denk ik dat het voor de organisator wel handig is als er een soort algemene plek, lounge, of centrale knop zit, waar wij bezoekers meer informatie kunnen vinden over de beurs zelf. Dus hoe zit het met parkeren, wat is het entertainment programma, wat is het nieuwtje van de dag, dat soort informatie. Dat is echt wel belangrijk dat wij dat ergens kunnen doen. En dat het proces makkelijk verloopt dat je op de dag zelf mensen nog toegang kan geven, of als er mensen nog uitgenodigd kunnen worden. of ja, dat het een doorlopend process is dat als er iemand nu nog zegt: ja ik wil nog wel graag een kaartje kopen en ik wil graag erop, dat dat kan.
23. **Dus de kaartjes hoeven niet tijdgebonden te zijn, dus alleen van het begin van de beurs tot het einde van de beurs?**
    Nee. Normaal werk je met bijvoorbeeld dagkaarten. Dus je hebt 1 dag toegang en dat zou je hier kunnen doen als je met een maximale capaciteit zit. Je zegt: nou er kunnen maximaal tien duizend man tegelijkertijd aanwezig zijn. Dan kan je dagkaarten doen. Als het niet uitmaakt zou ik zeggen, mensen kopen een kaart en kunnen de hele week rond lopen. Dat maakt dan niet uit. Dat ligt dan natuurlijk aan de capaciteit als er extra kosten aan gebonden zijn voor de organisator.
24. **Nog even over dat ingrijpen. Als het daadwerkelijk zo is dat er ingegrepen moet worden dan kan het zijn dat de beurs opnieuw moet worden opgestart en dit kan eventjes duren. Kunt u dit voor lief nemen als het echt noodzakelijk is of is het echt een groot probleem?**
    Opnieuw opstarten, hoe lang lig je er dan uit?
    Dat hangt af van de applicatie, dat kan ik niet zo zeggen, maar meestal is een server wel weer binnen een minuut opgestart.

Ja het is niet ideaal want je wilt niet dat er net je grootste klant rondloopt en dat ie opeens een glitch ziet en vast zit. Dat wil je natuurlijk niet. Je zou nog kunnen kijken of je bijvoorbeeld eigenlijk alleen de beheerder de mogelijkheid heeft om content te blokkeren, dat zou ik ook nog wel een prima optie vinden. Dat je als organisator iemand kan zeggen, luister stand 313 heeft een filmpje draaien die stopgezet moet worden. Dan kunnen wij even met hun bellen dat dit niet oké is en dat ze iets nieuws moeten verzinnen. Het liefst laat je de rest dan wel gewoon lopen.

25. **Is het misschien dan ook een mogelijkheid om hetgeen dat wordt geüpload wordt gecontroleerd door een organisator voordat het wordt verwerkt in het systeem? Dan heb je wel een stukje beveiliging, maar bij elke verandering moet dit wel geaccepteerd worden.**
    Dat kan wel, maar dan kies je er eigenlijk voor dat zij geen toegang hebben tot hun eigen back end stand. Want als ze het daarna nog kunnen veranderen dat heeft het voor mij geen zin om het te controleren. Dan zou je het moeten doen dat wij het moeten checken en dat vervolgens dan erin wordt gezet en dat zij als standhouders niet zelf meer kunnen aanpassen, tenzij dat via ons loopt. Maar liever, als organisator natuurlijk, heb jij dat standhouders gewoon zelf uploaden en spullen aanleveren zonder dat wij daar tussen zitten. Dat is gewoon een hels karwij. Het zijn twee verschillende manieren, denk ik.
    ja het is inderdaad kiezen tussen twee verschillende manieren. De ene zorgt ervoor dat je opnieuw moet opstarten omdat iemand de randen opzoekt en bij de andere zorg je ervoor dat dat niet voorkomt maar heb je meer werk aan.
    Het heeft zeker zin met die controle als zij daarna geen toegang meer hebben tot de achterkant van hun stand. Dan is het een goede optie. Dan check je in een keer wat er op moet en dan krijg je natuurlijk wel net voor de beurs nog 36 stands die zeggen dat er nog iets aanpassen. Ik wil toch nog een filmpje dat is wel een gedoe en gezeur. En andersom natuurlijk, als zij dat zelf er allemaal op kunnen zetten en kunnen bijhouden, dan hoeven wij natuurlijk alleen maar in te grijpen als we berichten krijgen van dat er iets geks is. Ja, dat zullen er 1, max 2 zijn. Dat is wel minder werk dan andersom.
    Dat kan, maar dan is er wel een risico dat het systeem opnieuw moet worden opgestart voor alle gebruikers op dat moment.
    Je zou het wel zo moeten maken dat je bijvoorbeeld zegt dat exposanten wel de mogelijkheid hebben om per dag nieuwe content aan te leveren. Ik zeg maar even wat. Alleen dat wordt dan verwerkt tussen twee uur s' nachts en zes uur 's ochtends. Dan zitten er misschien twee mensen op en voor die twee mensen is het niet zo'n punt dat het een minuutje opstart.
26. **Zijn er nog meer verwachtingen van het systeem waarvan je denkt: als je hier eens naar gaat kijken zou dat mooi zijn of misschien is dit nog wel een leuk idee.**
    Ja, ik zou zeker kijken of je, als je via gaming door z'n beurs heen loopt, wel de sfeer van de beurs kan neerzetten. Ik zou dan bijvoorbeeld misschien kijken of je verschillende soorten muziekjes kunt integreren. Bijvoorbeeld in hal 1 is dat het achtergrond muziekje maar dat je naar hal 2 loopt door de doorgang, dan sterft de ene uit en de andere start. Wat je ook in een fysieke beurs zou hebben. Dus zoiets zou ik ontwikkelen. En misschien een hostess poppetje waarop je kan klikken of hoe

dan ook, die jouw makkelijk terug leidt naar het hoofdmenu of die bijvoorbeeld de chatfunctie is met de organisatie. Dat zou natuurlijk ook kunnen want de organisatie heeft natuurlijk geen stand ofzo, dus wij hebben misschien een centrale infopunt. Dat zou natuurlijk wel kunnen. En ja, verlichting zou ik misschien toch over nadenken. Dus als ik overdag erover heen loop wil ik misschien wel een dag setting en als ik er ‘s avonds op zit wil ik misschien wel een avond setting.

Bijlage 2: Interview met Arne

Arne is de eigenaar van het bedrijf waar stage wordt gelopen. Tevens is Arne ook de product owner van het project. Het interview is gehouden om duidelijker te krijgen wat de eisen zijn aan het project vanuit het oogpunt van de product owner.

1. **Wat voor interactie zou jij verwachten wat een gebruiker zou hebben met een beurs?**
   In ieder geval in eerste instantie dat die kan inloggen, zodat wij weten wie het is en welke gegevens aan deze gebruiker gekoppeld is. Het authenticatie is belangrijk zodat wij ook kunnen tracken wie het is. Vervolgens verwachten wij dat die gaat rondlopen net zoals op een normale beurs. Hij kan een stand oplopen en op het moment dat die een stand oploopt kan hij ook interacties doen. Hij kan dingen aanklikken, hij kan her en der een spelletje spelen. Het draait vooral om digitale informatie verschaffing en die data proberen wij te verzamelen en te analyseren en deze bij de standhouder te brengen.
2. **Voor een gebruiker is het dus voldoende om een inlogscherm en registratiescherm te hebben en dan meteen over de beurs te lopen?**
   Ja en hij een video chat starten, informatie bekijken, filmpjes bekijken en trigger volumes in de game door een popup te krijgen of ergens op te klikken. Maar dit staat nog open.
3. **In een andere interview had een organisator het over dat het een uniek verkoopbaar iets moet zijn om toegang te krijgen. Ze had het dan over unieke codes of e-tickets.**
   Ja, je wilt net als in de echte wereld wel een ticket verkopen en daarmee verkoop je eigenlijk een login/registratie. En die is gekoppeld met de gegevens van die klant. Zodra die klant inlogt weten wij ook wie het is en daardoor kunnen wij de standhouders de informatie verschaffen over de gebruikers die hun stand hebben bezocht.
4. **Dus een unieke code zou al voldoende moeten zijn om te kunnen registreren?**
   Ja, een digitale ticket of hoe je het ook wilt noemen, een uitnodigingsmail, het kan op heel veel manieren. Het is gewoon het authenticatie gedeelte eigenlijk.
5. **Voor een standhouder komt er meer interactie aan te pas, wat zou jij verwachten wat een bedrijf voor schermen moet hebben?**
   Ja een standhouder heeft meer interactie. De standhouder heeft een eigen back-end waarin hij uiteraard zijn eigen data kan inzien en filteren. Hij kan dan zien welke gebruiker er is geweest, waarop hij heeft geklikt en die data kan hij ook filteren. Hij kan ook een chat starten met iemand die op zijn stand staat (dit wordt gedaan door een trigger volume). Er komen ook bezoekers op de stand en die moeten ook een chat kunnen starten met de standhouders die wordt beantwoord vanuit de backend.
6. **Dus de gegevens die hij ontvangt over de gebruiker, moet hij dus ook op een overzichtelijke manier terug zien?**
   Correct, het moet op lange termijn ook een export kunnen uitdraaien, bijvoorbeeld een CSV export. Maar het gaat erom dat hij overzichtelijk kan zien welke gebruiker

op zijn stand is geweest en waarop hij heeft geklikt, hoe lang zij op de stand zijn geweest, allemaal van dat soort data wat wij eigenlijk allemaal registreren met een game data analytics plugin die dan waarschijnlijk via een API de data naar een back end pushed waar het bedrijf de data kan filteren en bekijken. Ook kan hij dan de e-mail eruit trekken om zo een bezoeker te benaderen.

7. **Moet het dan een rauwe lijst zijn met data of is het bijvoorbeeld ook handig om een grafiek te hebben met zo lang hebben bezoekers gemiddeld op de stand gestaan enz.**
   Dat is ultiem, maar het is meer een wens. Het is niet het minimum vereiste. Het zou echt een hele mooie kers op de taart zijn, maar het gaat er in de eerste instantie om de basis informatie en om de contactgegevens van de mensen die op de stand zijn geweest.
8. **Er wordt waarschijnlijk bijzondere data verzameld van gebruikers, heeft u er al over nagedacht hoe deze data beschermd gaat worden i.v.m de nieuwe wetgeving?**
   Dat is een lastige, daar heb ik niet genoeg kennis van. In mijn optiek moet het heel duidelijk zijn dat bij het registreren van een account dat de gebruiker weet waarvoor zijn data voor gebruikt zou worden. Dus als een gebruiker een stand bezoekt, dat het ook heel duidelijk is dat op het moment dat die een stand bezoekt dat zijn data naar de standhouder gaat. Maar het moet wel heel duidelijk zijn en ik vind het heel belangrijk dat de data veilig opgeslagen wordt. De standhouder krijgt alleen de data die van zijn stand belangrijk is en niet van andere.
9. **Hoe zie jij het voor je met de stands, moeten die al van te voren zijn gemaakt en kunnen ze kiezen uit stands, of moeten ze kiezen uit modules?**
   Hoe ik het voor me zie zijn een aantal basis stands waaraan een paar parameters zitten, zoals bijvoorbeeld een logo, die je kunt veranderen. Daarnaast is er altijd een maatwerk optie, ze zullen dan contact krijgen met het huidige bedrijf en wij zullen dan de maatwerk stand moduleren.
10. **Hoe moet ik me dit voor me zien? Zijn de stands al voor gemoduleerd en Unreal Engine hoeft alleen maar de modules in te laden?**
    Om het schaalbaar en flexibel te houden zijn ze al van te voren gemoduleerd en dat je alleen aan Unreal Engine hoeft door te geven welke stand bij welke plek is gekozen. Dus het is al in-game content en wat Unreal Engine doet is alleen ophalen welke stand op welke plek moet worden ingeladen.
11. **En hoe zie je het beheer voor je?**
    De beheerdersrol zou eigenlijk bij de organisators moeten liggen. Het mooiste zou dan zijn dat ze een backend hebben waarin ze bepaalde basis dingen beheren. Als ze echt een bedrijf eruit zouden willen hebben, zouden ze deze moeten kunnen blokkeren en dan zou de plaats bijvoorbeeld leeg zijn. Ze zouden ook bijvoorbeeld content moeten keuren zodat er geen gekke dingen tussen zitten.
12. **Dus je wilt eigenlijk het beheer bij de organisator leggen?**
    Ja, het is een platform dat wij aan de organisator voorleggen en wij komen eigenlijk alleen in beeld als er maatwerk nodig is, maar dat is het ideale plaatje. Je blijft natuurlijk ook zitten met de standaard stands en de standaard content die in de betreffende stand staan en de producten willen toevoegen, is ook een uitdaging. Je kunt die producten natuurlijk ook niet standaard fixen. Standaardisatie klinkt heel

mooi maar op lange termijn maakt het dan ook schaalbaar. Er zitten nog wel wat haken en ogen aan. Hoe ik het in de eerste instantie voor me zag was dat wij de hele beurs gewoon opbouwen, en elke stand, wel met standaard vormpjes, zelf hard coded in de game zetten en niet flexibel. Daarin nemen wij dan de producten op van de standhouders en vervolgens gooien we de beurs live. Dan heb je het niet flexibel, standaard en schaalbaar, maar dan heb je een MVP verhaal om mee te starten. Het flexibele is ook nog niet meteen nodig en ik denk dat je dan te veel op de nek haalt en pas later op zou moeten pakken.

13. **Wat zijn (een beetje) de eisen aan de communicatie tussen de gebruikers (het zal waarschijnlijk gaan via videobellen of chatten)?**
    De communicatie gaat via 1 op 1 en je hebt dan alle aandacht van degene waaraan je de vraag stelt.
14. **Het kan dan zijn de mensen in de wacht komen te staan of iets in die richting?**
    Ja, ik denk zeker dat je een soort wachtrij moet hebben want je kunt er niet honderd man neerzetten. Bij een internationale beurs kan het zo zijn dat er honderd man tegelijk zijn en je hebt er dan geen 100 verkopers tegelijk zitten.
15. **De grote van de beurs: waaraan moet ik dan een beetje denken? Van de kleinste beurs tot de grootste gekste beurs?**
    Ik heb daar nog niet echt een idee over maar ik denk dat het voor kleine beurzen niet echt interessant is en ook niet voor hele goedkope producten waarschijnlijk. Het zou waarschijnlijk om grotere beurzen gaan. Die nu ook bijvoorbeeld in de RAI Amsterdam 3 hallen hebben. Dit zijn al serieuze beurzen, die wij dan digitaal maken. Digitaal is natuurlijk alles enorm op te pompen en is niet gebonden aan hoe groot een gebouw is en hoe groot de stand is.
16. **Nee, in dat opzicht niet maar het is meer met de load waarmee je te maken kunt krijgen op een server. Want er is wel een verschil of je nou 1000 man op een server hebt of ineens 10.000 man op een server.**
    Correct, dat is wel een groot verschil inderdaad. Wat dat betreft zou je best een grote beurs kunnen organiseren, maar dan meer gespreid. Je krijgt dan je ticket bijvoorbeeld wel voor een bepaalde dag in een bepaalde tijd range. Het is dan over de hele week verspreid en dan krijgt Pietje Puk op maandag en Henkie op vrijdag. Het is dan gekoppeld aan de ticket zeg maar, om zo de load te balance op de server. Als je in een keer 10.000 bezoekers krijgt dan heb je 10.000 servers nodig en dat zou niet goed komen.
17. **Dan kom je natuurlijk bij het prijs gedeelte uit. Het moet natuurlijk altijd zo goedkoop mogelijk, maar kan zo zijn dat product A goedkoper is dan B maar B kan wel meer functies ondersteunen. Zit in de prijs nog speling, of is het goedkoopste is de beste?**
    Nee absoluut niet. Als het ene beter is dan het andere maar is wat duurder is dit niet erg. Het moet stabiel zijn en goed werken. Dit staat niet altijd in relatie met de prijs zeg maar. Niet altijd het slechtste is het goedkoopste en niet het duurste is het beste. We moeten echt kijken naar functies en niet naar de prijs.
18. **Dit geldt dan ook voor servers?**
    De servers is eigenlijk bijna wel zeker dat dit met PureWeb gaat gebeuren. Die draaien op Amazon. PureWeb is een Canadese organisatie die eigenlijk al heel lang bezig is met streaming van evenementen. Ze hebben ook de ‘Burning man’

gestreamd. Dit is een een groot festival in Amerika. Dit is allemaal goed gegaan. Het is schaalbaar en ze zijn al járen bezig met schaalbaarheid vooral. Ze kunnen heel veel users tegelijk aan. Zij hebben daarvoor een heel stabiel platform staan. We hebben zelf ook een platform ontwikkeld, maar dat is maar heel klein in vergelijking met hun, dus als we iets zouden uitrollen dan zouden wij dat met hen doen.

19. **Zit er ook iets van microservices in, want daar ga je naartoe als je schaalbaarheid wilt. Je wil dan je database ergens anders en je website weer ergens anders.**
    Zij doen alleen het Unreal Engine gedeelte. Dus de zaken waar die andere dingen op draaien, dat is allemaal eigen beheer. De server opzetten en draaien van een Unreal server doen zij en deze praat dan met de server van ons. Daar staat verder niets op alleen de game als ware.
20. **Zijn er nog andere zaken waarvan jij denkt, als er nog tijd over is, is het mooi als je daar nog naar kijkt?**
    Tja lastig, ik vind vooral het schaalbare heel interessant. Kijk hoe kunnen we shiften naar een platform i.p.v dat wij heel druk zijn met content en als je een platform hebt dan betekent dat eigenlijk dat alles flexibel en geregeld is. Dat producten bijvoorbeeld zelf ingelanden worden, dat wij er helemaal niet meer tussen zitten. Alleen voor extreme maatwerk dingen. Dan kun je het ook als platform in de markt zetten. Dan wordt het bedrijf ook meer waard. Dan ga je van dienstverlening naar platform dat is nogal een verschil zeg maar.Het is in ieder geval wel interessant om te onderzoeken in hoeverre je onafhankelijk kan zijn van ingrepen in de 3D kant. Je zou uiteindelijk ernaar toe willen dat een organisator zelf alles kan inplannen en erin zetten als een soort editor. Misschien wil je wel dit uiteindelijk bij de klant hebben, zodat zij zelf de potjes en pannen kunnen inladen en neerzetten. Maar ik denk dat je dit bij de organisator wilt hebben.
21. **Er is voor beide kanten natuurlijk wel wat te zeggen. De organisator wil niet alles controleren. Die vinden dat heel veel werk en willen dit eigenlijk al bij een ander leggen.**
    Ja, normaal is dat de rol van de standbouwers, die ontwerpt en bouwt alles. Er zou dus eigenlijk misschien nog een rol weggelegd zijn voor een bedrijf om dit te doen. Er zou een rol kunnen ontstaan maar ik vraag me af of een organisator zit te wachten op het inrichten van stands. Ik denk dat die rol eerder bij bedrijven ligt. Ja dat denk ik wel, al pratende. Het bedrijf kan dan zelf een stand ontwerpen, zelf de producten inladen, maar dan krijg je weer de vraag wat jij net ook al zei: de vraag of ze alles willen controleren? Ja dat moeten ze wel, dat moeten ze nu ook. Er moet dan wel een blok komen dat ze voor die tijd alles hebben veranderd, daarna is de sluitingsdatum en dan gaan ze alles een keer checken en als het dan goed is, is het goed. Dan kom je er ook niet meer aan. Als je dan iets wilt veranderen gaat het via ons. Iets in die trant.
22. **Ja dat zou een goede oplossing kunnen zijn. De organisatoren vonden dit ook wel een goede oplossing. Ze waren alleen bang dat het teveel tijd zou kosten, omdat iedereen op het laatste moment iets wil veranderen, dat gebeurt nu ook al.**

Ja, dat is maar net hoe streng je bent als organisatie. Als je gewoon zegt van, tot die tijd kan je veranderen en daarna niet meer. Als je dan toch wilt veranderen dan heb je pech. Het moet in ieder geval wel makkelijk zijn zodat het ook snel kan.

23. **Hoe ik het dan voor me zie is, dat je dan een scherm hebt met de veranderingen. En dat je dan op een knop kan klikken om het goed te keuren en je hebt een knop om het af te wijzen met een text optie om aan te geven wat er niet goed aan is. Het bedrijf krijgt hierop dan weer feedback.**
Ja zoiets klinkt goed.

Bijlage 3: Reflectie

De reflectie is geschreven in de STARR methode. Met deze methode is er teruggekeken op situatie. Dit wordt gedaan door een aantal vragen te beantwoorden in de volgende categorieën: situatie, taak, actie, resultaat en reflectie.

1. Planning

Wat was de situatie?

Ik heb het project alleen moeten maken. Dit was de eerste keer dat ik een groot project zelf moest gaan maken. Het project bestond uit verschillende delen: de documentatie en de code. Om het project te structureren heb ik een planning gemaakt.

Wat was je taak?

Mijn taak was om te zorgen voor een duidelijke planning en mij daaraan te houden. De planning had ik opgedeeld in twee delen. Onder het documentatie planning zat het plan van aanpak en de scriptie. Onder de andere planning zat het schrijven van de code. Hieronder zat bijvoorbeeld de API en de front-end.

Hoe heb je het aangepakt?

Ik had de planning gemaakt in tijdvakken. Ik werkte alleen dus om volledig scrum te gebruiken vond ik wat overdreven maar ik heb de planning wel in weken opgedeeld. Dit heb ik gedaan om het toch te kunnen structureren. Elke tijdvak duurde twee weken en ik die twee weken plande ik wat ik af wou hebben. Ik had eerst de planning gemaakt voor de documentatie en het onderzoek. Daarna had ik een planning gemaakt voor het schrijven van de code.

Heeft het gewerkt?

Deze manier van plannen heeft niet gewerkt voor mij. Ik had moeite om mijzelf te kunnen motiveren door het thuiswerken en daardoor heb ik vertraging gekregen in mijn afstuderen. Ik heb de planning niet gehaald omdat ik vaak tegen problemen aanliep die ik niet had voorzien. Dit zorgde ervoor dat ik weer vertraging kreeg voor de volgende planning. Ook door het niet uit te spreken van de planning kon er ook niemand mij aan houden, behalve ikzelf. Telkens dacht ik wel dat het zou gaan lukken en maakte ik mij er niet druk over. Dit samen met de motivatie problemen hebben ervoor gezorgd dat ik de planning niet heb gehaald en te weinig tijd had om de code te schrijven en heb er een extra periode aan moeten plakken.

Wat heb je er van geleerd?

Ik heb geleerd om de planning ook uit te spreken aan iemand. Dit kan een leraar zijn of een begeleider. Hierdoor dwing ik mijzelf om ook aan de planning te houden omdat ik verantwoording moet gaan afleggen aan iemand anders. Dit zorgt voor wat druk en kan er ook voor zorgen dat ik me dan wel aan de planning kan houden.

2. Requirements

Wat was de situatie?

Aan het begin van het project was er nog veel onduidelijk over hoe het PoC eruit zou moeten zien. Er was een kleine lijst met wensen van een klant die eerder al een eerder was opgesteld.

Wat was je taak?

Aan mij was het de taak om erachter te komen wat er precies nodig was. Het ging hierbij vooral om de wensen van een aantal stakeholders.

Hoe heb je het aangepakt?

Ik had aan Arne gevraagd of hij iemand wist die mij zou willen helpen bij het onderzoek naar de wensen. Hij wist iemand en met diegene heb ik een interview gepland om zo te kijken welke wensen voor haar belangrijk waren. Daarnaast heb ik aan Arne gevraagd of hij aan een interview wou meedoen om zo zijn wensen naar voren te krijgen. Van tevoren had ik voor beide interviews vragen voorbereid. Deze interviews heb ik opgenomen zodat ik deze later kon opschrijven. Dit heb ik gedaan zodat ik tijdens het interview kon focussen op de inhoud van de vragen. Later heb ik het interview uitgetypt en aan de hand daarvan kon ik de wensen van de stakeholders halen om te gebruiken in mijn onderzoek.

Heeft het gewerkt?

Dit was een effectieve manier om de wensen te krijgen. Door deze manier te gebruiken heb ik een goede use case diagram kunnen maken om zo duidelijk te kunnen laten zien wat er verwacht wordt van het systeem. Deze heb ik aan Arne laten zien om te kijken of hij het ermee eens was. Hij had geen verdere aanmerkingen op de resultaten, dus het onderzoek was goed gelukt.

Wat heb je ervan geleerd?

Ik heb niet vaak interviews gedaan en ik ben erachter gekomen dat je hiermee veel informatie kan vergaren over het onderwerp. Ook ben ik tevreden met de manier waarop ik het heb gedaan. Door het op te nemen en later uit te schrijven heb ik ervoor gezorgd dat ik veel informatie had. Het zorgde wel voor extra werk. Vooral het uittypen kostte meer tijd dan ik had verwacht maar die extra tijd was het wel waard want hierdoor kon ik mij beter focussen op het gesprek en kon ik beter vragen stellen. De volgende keer zou ik het weer op deze manier doen.

3. Communicatie met de leraar

Wat was de situatie?

Door de gehele afstudeerperiode heb ik regelmatig contact gehad met de leraar. Tussen de online gesprekken door heb ik ook een aantal keer via de mail een vraag gesteld. Hierop heb ik regelmatig pas vlak voor de bespreking antwoord gekregen.

Wat was je taak?

Ik moest de leraar uitnodigen voor de vervolggesprekken. Daarnaast heb ik ook een paar keer gevraagd naar extra uitleg.

Hoe heb je het aangepakt?

De communicatie tussen mij en de leraar liep via de mail. Dit was de makkelijkste manier en meestal werkt deze manier van communicatie ook. Als ik geen antwoord kreeg stuurde ik nog een e-mail met de vraag of hij wou reageren.

Heeft het gewerkt?

Ik heb vaker contact gehad via de mail en vaak kreeg in binnen een korte tijd antwoord. Dit was in deze afstudeerperiode niet. Ik kreeg vaak pas antwoord op de dag van de besprekingen zelf. Ook de extra mails sturen werkte niet.

Wat heb je ervan geleerd?

Dat ik niet teveel moet vertrouwen op het e-mailen. Het kan voorkomen dat een persoon het niet leest, zoals in dit geval. Dit zorgde bij mij voor veel frustratie. Deze frustraties heb ik echter niet uitgesproken naar de leraar omdat ik dacht dat het uitspreken van de frustraties zou zorgen voor conflicten tussen ons. Ik denk wel dat ik de volgende keer deze frustratie moet uiten zodat de andere partij het ook weet dat ik gefrustreerd ben door de manier van communicatie. Ook kan ik vragen naar een andere vorm van communicatie. Door bijvoorbeeld met de leraar te chatten kan ik sneller antwoord krijgen.

4. Vastzitten met een probleem

Wat was de situatie?

Tijdens het programmeren moest ik een bestand uploaden naar de server om deze vervolgens op te slaan in de database. Dit heb ik niet werkend gekregen.

Wat was je taak?

Het was de bedoeling om een foto en een videobestand te uploaden naar de server om deze vervolgens op te slaan op de server.

Hoe heb je het aangepakt?

Ik heb op internet gezocht naar manieren om bestanden te uploaden. Deze manieren heb ik gevolgd en getest op mijn platform. Er waren een aantal manieren om het te doen met javascript en een aantal manier om dit te doen met de Node.js server.

Heeft het gewerkt?

Helaas is het mij niet gelukt om bestanden te uploaden naar de server. Ik kreeg vaak een lege body aan de server kant waardoor ik geen bestanden kan uploaden naar de server. Wel kreeg ik de bestanden te zien aan de javascript kant, maar zodra ik deze stuurde naar de server kreeg ik niks meer te zien. Dit heeft mij veel tijd gekost omdat ik dit onderdeel te graag wou laten werken. Ik heb er veel tijd aanbesteed die ik beter aan andere onderdelen kon besteden.

Wat heb je hiervan geleerd?

Ik heb geleerd om sneller door te gaan als een onderdeel niet werkt. Door te accepteren dat het niet werkt kan ik eerder zoeken naar een andere manier. Dit heeft als resultaat dat ik sneller verder kan gaan met het project. Ook heb ik geleerd om eerder hulp te vragen. Vaak wil ik graag van mijn eigen kracht uitgaan maar dit kan niet altijd. Ik heb geleerd dat het niet erg is om hulp te vragen als het niet lukt.