adres
Colosseum 13
7512 PV Enschede

www.ovsoftware.nl
info@ovsoftware.nl

telefoon
+31 (0)53 3033600

kvk
06049464

Afstudeerverslag

HBO-ICT Saxion

7 februari 2021 – 11 november 2021

Generiek Platform CoreXR

OVSoftware

Afstudeerder:
Michael Steen
Studentnummer: 417722

Bedrijfsbegeleider:
Eric Barten

Michael.steen@ovsoftware.com
+31 (0)6 44169224

documentversie
1.2

Afstudeerverslag van Michael Steen, HBO ICT, Saxion, Enschede

CoreXR

Publicatiedatum: 23 september 2021

OVSoftware

Adres: Colosseum 13, 7521 PV Enschede
Stagebegeleider: Eric Barten
Email: eric.barten@ovsoftware.com

Saxion

Adres: M.H. Tromplaan 28 Enschede
Begeleidend docent: Erik van der Arend
Email: h.a.vanderarend@saxion.nl

Michael Steen

Adres: Oosteinde 16 Vriezenveen
Telefoonnummer: 06-44169224
Email: michael.steen1996@gmail.com

1 Samenvatting

Deze afstudeeropdracht is uitgevoerd bij OVSoftware in Enschede. Voor de afstudeerstage is er samengewerkt samen met Total Reality. Dit bedrijf werkt namelijk aan een VR-app waarmee personeel wordt getraind in het omgaan met en onderhoud van apparatuur. Hiervoor gebruikten ze BOMs, maar deze werden door Total Reality handmatig opgehaald en in het programma gezet. Dit was niet ideaal, vooral niet als het aantal gebruikers groeit. De opdracht was het maken van een prototype van een tussenlaag tussen ERP-systemen waar de BOM te vinden is en de VR-app. Het is de bedoeling dat de oplossing generiek genoeg is om allerlei ERP-systemen met de VR-app te kunnen verbinden. Deze tussenlaag haalt de BOM op en geeft deze door aan de VR-app zodat dit niet meer handmatig hoeft. Ook moet deze tussenlaag op een generieke manier ontworpen zijn zodat de tussenlaag zonder veel moeite bij verschillende klanten bruikbaar zou zijn.

Om te beginnen is er een onderzoek gedaan met als hoofdvraag “Hoe kan ik mijn oplossing generiek genoeg maken om naast Trimergo 80% van de ERP-systemen te ondersteunen?”. Trimergo is een ERP-systeem dat gebruikt wordt door een bedrijf genaamd Hoppmann B.V. Het plan was om het prototype daar te laten werken als een Proof of Concept. Dit onderzoek is vooral gedaan met online literatuuronderzoek en een interview. Uit het onderzoek (Steen, 2021) bleek dat er minstens twee bestandstypen ondersteund zouden moeten worden om Trimergo en 80% van andere ERP-systemen te ondersteunen. Ook kwam er mogelijk waardevolle informatie naar voren over het marktaandeel van ERP-systemen.

Er is vervolgens door middel van het onderzoek een analyse gemaakt voor wat het systeem zou moeten kunnen en er is een concept bedacht. Hier zijn requirements voor gemaakt met de SMART en MoSCoW methodes. Nadat deze requirements besproken waren met de belanghebbenden, is er een ontwerp gemaakt om aan deze requirements te voldoen.

Het eindresultaat is een generiek platform genaamd CoreXR, waarmee VR-apps aangesloten kunnen worden op meerdere ERP-systemen. De opgestelde requirements voor dit systeem zijn behaald.

OVSoftware

Inhoud

2 Voorwoord

Dit is mijn afstudeerverslag van mijn afstudeeropdracht voor mijn HBO-ICT opleiding bij Saxion Hogeschool.

Aangezien ik bij het laatste half jaar van mijn opleiding ben aangekomen is het tijd voor de afstudeerstage. Centraal onderdeel van de afstudeerstage is de afstudeeropdracht.
Na goed contact heb ik er voor gekozen deze bij OVSoftware uit te voeren.
OVSoftware is in 1972 opgericht en één van de eerste softwarebedrijven van Nederland.
Vanuit vestigingen in Apeldoorn, Den Haag, Enschede en Münster helpt OVSoftware klanten door het efficiënt inzetten van de juiste expertise bij alle aspecten van de realisatie van software, beheer en onderhoud. De opdracht waar ik aan zal werken is het maken van een herbruikbare tussenlaag tussen ERP-systemen en XR-oplossingen.

Ik zal hier elders in dit document meer informatie over de opdracht geven. Ik heb voor dit bedrijf en deze opdracht gekozen omdat ik van medestudenten positieve dingen heb gehoord over OVSoftware, het bedrijf redelijk in de buurt is, en de opdracht mij interesseerde. Met mijn specialisatie Human Machine Interaction ben ik bezig geweest met XR. Hoewel ik bij deze opdracht niet zelf een XR-programma hoef te maken vind ik de stap tussen ERP en XR erg interessant omdat het me leert hoe XR goed aan kan sluiten bij de rest van de IT. Tijdens de specialisatie voelde XR vaak nogal los van de “normale” IT-zaken dus deze opdracht vult naar mijn gevoel dat goed in.

Ik wil graag ook OVSoftware bedanken voor het aanbieden van deze stage. Dat OVSoftware toch stagiairs aannam ondanks de situatie met het corona-virus kan ik zeer waarderen. Ook zou ik graag specifiek Eric Barten willen bedanken, bij wie ik altijd aan kon kloppen voor hulp en me elke dag belde hoe het ging, ook als ik die dag thuis werkte.

Ik wens u veel leesplezier.
Michael Steen
Vriezenveen, 24 oktober 2021

3 Begrippenlijst

Begrip: **XR**
Verklaring: XR staat voor Extended Reality. Dit is een verzamelterm voor alle reële en virtuele gecombineerde omgevingen en mens-machine interacties gegenereerd door computertechnologie en wearables, zoals VR-brillen.

Begrip: **VR**
Verklaring: Virtual reality is een computertechniek die je een compleet andere werkelijkheid laat ervaren. Hiervoor zijn speciale brillen ontwikkeld waarin een beeldscherm zit.

Begrip: **ERP**
Verklaring: ERP staat voor Enterprise Resource Planning. Een ERP-systeem wordt gebruikt ter ondersteuning van alle processen binnen het bedrijf. Een ERP-programma bestaat meestal uit kleine deelprogramma's die allemaal een specifieke taak ondersteunen, zoals het bijhouden van facturen of voorraden.

Begrip: **BOM**
Verklaring: BOM staat voor Bill Of Materials, ook wel een stuklijst genoemd. Dit zijn lijsten van onderdelen die verwerkt zijn in een product. Naast hoeveelheden van onderdelen staat er vaak informatie in over bijvoorbeeld de prijs en het gewicht van de onderdelen.

4 Context van de opdracht

Het doel van de afstudeeropdracht is het maken van een tussenlaag tussen een VR-app voor trainingen en ERP-software. De bedoeling is dat via deze tussenlaag de VR-app gegevens automatisch op kan halen, wat tijd en moeite bespaart bij het gebruik van de app.

Het ontwikkelen van deze VR-app was niet onderdeel van het project, deze wordt gemaakt door Total Reality, een software bedrijf gespecialiseerd in XR software waar OVSoftware mee samenwerkt. Het systeem dat tijdens de afstudeerstage ontwikkeld werd moest hier wel uiteindelijk op worden aangesloten, het is namelijk een programma ter ondersteuning van die VR-app. Total Reality is daarom ook een belangrijke belanghebbende in dit project.

De VR-app is bedoeld voor trainingen van personeel in de maakindustrie zoals bijvoorbeeld procesoperators. De procesoperator is in het productieproces verantwoordelijk voor het goede verloop van geautomatiseerde processen en is degene die controleert of het proces succesvol verloopt en indien nodig ingrijpt. Het is belangrijk dat de procesoperator weet hoe bepaalde machines werken. Normaal moeten in een bedrijf nieuwe procesoperators regelmatig ingewerkt worden. Dit zijn trajecten van intensief trainen tot passief begeleiden die vaak tot 1 jaar in beslag nemen. In alle fases van dit proces dient een trainer, veelal collega operator, mee te lopen met de nieuwe medewerker, wat weer beslag legt op de ‘productieve uren’. Deze VR-app is nog in ontwikkeling bij Total Reality.

Om dit efficiënt en effectief te doen willen de projectpartners Total Reality BV, Hoppmann en OVSoftware BV in dit project komen tot een compleet en gecertificeerd trainingsprogramma waarmee procesoperators meer autonoom én sneller op het gewenste niveau komen.
De inzichten van dit project worden tevens gebruikt voor op te zetten lesmodules op het mbo en hbo rondom flexibele productie in het MKB. Door het op te zetten trainingsprogramma moet het tevens mogelijk worden om, zonder een volledige MBO-opleiding te doorlopen, versneld de benodigde certificaten te halen om als procesoperator aan de slag te kunnen in het bedrijfsleven.

De afstudeeropdracht is niet het maken van dit trainingsprogramma, maar het maken van een herbruikbaar platform tussen bestaande ERP-systemen en dit trainingsprogramma die het opzetten van dat trainingsprogramma vereenvoudigt. Er zijn nog geen bestaande oplossingen bij Total Reality om gebruik van te maken of om uit te breiden.

Deze afstudeeropdracht vindt plaats bij OVSoftware, maar er ook veel samengewerkt met Total Reality. Deze hebben samen interesse in de stageopdracht en het plan is om deze bij Hoppmann te gebruiken. Het is het de bedoeling dat de tussenlaag generiek genoeg is om in andere situaties ook te gebruiken.

OVSoftware is het softwarebedrijf waar de afstudeerstage plaatsvond en waar ik regelmatig op kantoor ben geweest. Wegens de situatie met corona werd er vooral thuisgewerkt, maar ik was wel een dag per week op kantoor in Enschede.

Hoppmann is een bedrijf dat productiemachines maakt. Ze hebben interesse in de VR-app van Total Reality wegens de eerder genoemde redenen. Het is wel de bedoeling dat in de toekomst de VR-app en mijn tussenlaag bij andere bedrijven ook gebruikt kan worden.

Verder in dit verslag komen de volgende hoofdstukken aan bod:

Aanpak In dit hoofdstuk zal er verteld worden over de gekozen aanpak van deze stageopdracht.
Onderzoek In dit hoofdstuk zal het proces en resultaat van het onderzoek beschreven worden.
Requirements In dit hoofdstuk zullen de afgesproken eisen van het project beschreven worden.
Ontwerp In dit hoofdstuk zal er beschreven worden welk ontwerp er is gemaakt voor het project.
Implementatie In dit hoofdstuk staat het uiteindelijke resultaat beschreven, en het proces van het maken hiervan.
Aanbevelingen In dit hoofdstuk staan mijn aanbevelingen over mogelijke verdere ontwikkeling van het project. Aangezien dit project het maken van een prototype is is het aannemelijk dat er later verder ontwikkeld wordt.
Literatuurlijst In dit hoofdstuk staat een lijst met gebruikte bronnen.
Bijlagen In dit hoofdstuk staan de bijlagen van dit afstudeerverslag, zoals bijvoorbeeld mijn reflectie.

4.1 Probleemstelling

Op het moment is het zo dat Total Reality handmatig gegevens van een ERP-systeem overneemt en in Unity laadt om er vervolgens met hun VR-app, die ze aan het ontwikkelen zijn, gebruik van te maken. Dit is niet ideaal omdat dit niet efficiënt is en veel tijd gaat kosten als dit regelmatig moet gebeuren. Als er een tussenlaag zou zijn die de gegevens van de ERP-systemen op kan halen en deze aan de VR-app kan leveren zou dit tijd en moeite besparen. Mijn opdracht is dan ook om een prototype van deze tussenlaag te maken. Alhoewel deze gemaakt wordt voor een specifieke klant is het de bedoeling dat dit een generieke oplossing wordt qua verbinding met ERP-systemen en geen maatwerk oplossing.

5 Aanpak

Aan het begin van de afstudeerstage heb is er een plan van aanpak gemaakt. Hierin staat de planning voor het project die ook als leidraad is gebruikt. Het algemene plan was om na het plan van aanpak een onderzoek te doen naar ERP-systemen en de opties die ze hebben voor het exporteren van BOM's. Dit werd dan gevolgd door het maken van een ontwerp en uiteindelijk de implementatie van dit ontwerp. Deze planning is terug te vinden in de bijlage. (Zie hoofdstuk 11.2)

Er is gebruik gemaakt van de Scrum-methode tijdens de afstudeerstage. Hiervoor had was er een dagelijkse stand-up met de bedrijfsbegeleider waarin voortgang besproken werd en waar vervolgens aan gewerkt zou worden. Als er thuis werd gewerkt ging dit telefonisch. Ook werd er gebruik gemaakt van een online Scrum-bord in Jira. Er waren sprints van 1 week, Hiervoor is gekozen zodat het makkelijk is om snel bij te sturen waar dat nodig mocht zijn.

Als er tijdens het ontwikkelen iets functioneel was werd dit aan de begeleider getoond als een soort kwaliteitscontrole. Ook is er gebruik gemaakt van Jenkins voor het automatisch draaien van tests in verband met code kwaliteit. Eer voordeel hierbij is dat er zo ook werd gekeken of het programma op andere systemen ook zou werken. Daarnaast werd er ook gebruik gemaakt van GitLab voor versiebeheer van mijn project.

Ook is er voor de kwaliteit van de code via Jenkins gebruik gemaakt van SonarCube. Dit is een platform dat continu een beoordeling gaf qua code kwaliteit en eventuele beveiligingsproblemen.

6 Onderzoek

6.1 Opzet

Dit project heeft als doelstelling dat de tussenlaag die in dit project gemaakt wordt generiek genoeg is om in 80% van de gevallen bruikbaar te zijn. Een andere doelstelling is dat de oplossing werkt met het ERP-systeem Trimergo. Met een onderzoek is geprobeerd er achter te komen wat daarvoor nodig is en hoe dat verstandig aangepakt kan worden.

Voor dit onderzoek is er een hoofdvraag en een aantal deelvragen gemaakt. Om deze te beantwoorden was het plan om informatie via het internet te verkrijgen en waar mogelijk demo's van ERP-systemen uit te proberen. Andere vragen zouden beantwoord worden door in gesprek te gaan met experts en als het mogelijk is ook een interview te houden. In verband met

de corona situatie destijds waren sommige opties qua interviews beperkt, dus was er een voorkeur voor online gesprekken.

Om het onderzoek structuur te geven is er voor gekozen om het DOT-framework te bekijken voor gepaste onderzoeksmethoden. Er zijn uiteindelijk meerdere onderzoeksstrategieën gebruikt, namelijk veld, bieb, lab en showroom.

6.2 Onderzoeksvragen

De hoofdvraag van mijn onderzoek is “Hoe kan ik mijn oplossing generiek genoeg maken om naast Trimergo 80% van de ERP-systemen te ondersteunen?”.

Ter ondersteuning van deze hoofdvraag heb ik de volgende deelvragen beantwoord:

- Welke data is nodig voor de XR-app van Total Reality?
- Hoe verkrijg ik data uit Trimergo?
- Welke ERP-systemen zijn het meest in gebruik?
- Hoeveel verschillen andere ERP-systemen met Trimergo?
- Hoe kan er rekening met deze verschillen worden gehouden?

6.3 Welke data is nodig voor de XR-app van Total Reality?

6.3.1 Aanpak

Om meer te weten te komen over welke data Total Reality nodig is is er besloten hiernaar te vragen in een interview met mijn contactpersoon bij Total Reality. (Zie bijlage 11.1) Hiervoor is veldonderzoek gedaan.

6.3.2 Uitvoering en resultaten

Tijdens het interview is hiernaar gevraagd. Het antwoord was dat de BOM de kern is. Maar wat er in die BOM zit kan nogal verschillen per klant. Ook is er na het interview online onderzoek gedaan naar voorbeelden van BOM’s. Dit gaf een aantal voorbeelden waaruit bleek dat er inderdaad tussen BOM’s veel verschillen zitten. Sommige hebben maar drie kolommen, andere meer dan 20 en deze kolommen waren ook regelmatig niet hetzelfde.

6.3.3 Conclusie

Total Reality wil de BOM verkrijgen van ERP-systemen, maar de inhoud hiervan kan erg verschillen per klant.

6.4 Hoe verkrijg ik data uit Trimergo?

6.4.1 Aanpak

Om er achter te komen hoe data uit Trimergo is te verkrijgen was het plan als eerst online literatuuronderzoek te doen, zoals op hun eigen website rond te kijken. Eventueel zou er contact kunnen worden gezocht met Trimergo of mijn contactpersoon bij Total Reality. Een demoversie van Trimergo verkrijgen zou ideaal zijn. Hiervoor is biebonderzoek gedaan, veldonderzoek, en er werd geprobeerd met de demoversie showroomonderzoek te doen.

6.4.2 Uitvoering en resultaten

Voor deze deelvraag is er onderzoek op internet gedaan, maar hier is geen nuttige informatie uit gekomen. Door contact met Trimergo werd duidelijk dat de optie om een BOM te exporteren niet bestond. Om toch zeker te zijn dat dit klopte is er ook een interview geweest met mijn contactpersoon van Total Reality en is er geprobeerd een demoversie te krijgen van Trimergo.

Het is niet gelukt een demoversie te bemachtigen. Na een interview met de contactpersoon van Total Reality was er niet meteen een antwoord, maar er kon later wel bij Hoppmann worden gevraagd naar het exporteren van een BOM, dit duurde echter nogal lang omdat er via OVSoftware, via Total Reality via Hoppmann een vraag moest worden gesteld aan iemand die op dat moment bij een vierde bedrijf was. Er was na enige tijd een reactie, hoe er precies data uit Trimergo te verkrijgen is was er niet uit te halen, maar er was wel een geëxporteerd CSV bestand verkregen die uit Trimergo is gehaald. De persoon bij Hoppmann die hierover gaat was daarna een lange tijd niet beschikbaar.

6.4.3 Conclusie

Er is vanuit Trimergo een BOM handmatig als een CSV bestand te exporteren.

6.5 Welke ERP-systemen zijn het meest in gebruik?

6.5.1 Aanpak

Er is online literatuuronderzoek gedaan naar welke ERP systemen veel in gebruik zijn. Ook werd er gezocht welke ERP-systemen export van BOM's ondersteunen en is er geprobeerd demoversies te bemachtigen. Hiervoor is de Biep en Showroom strategie gebruikt van het DOT-framework (Turnhout, 2013).

6.5.2 Uitvoeren en resultaten

Er is op internet gezocht naar ERP-systemen en tijdens het zoeken is een een top 10 gevonden bij het onderzoeksinstituut Forbes (Columbus, 2020) ERP-systemen zonder BOM-ondersteuning zijn gediskwalificeerd. De volgende systemen bleven nog over:

- Oracle ERP,
- Netsuite
- Acumatica
- Epicor ERP
- Sage X3

Aangezien populariteit niet per se betekent dat een ERP-system veel in gebruik is zijn deze gegevens niet erg nuttig voor het beantwoorden van de hoofdvraag van het onderzoek, maar de stakeholders leken wel geïnteresseerd in deze data, dus zijn deze alsnog opgenomen in de resultaten.

Via de website App Run The World is er een top 10 ERP aanbieders qua marktaandeel in 2019 gevonden.
Daaruit bleek deze top 10:

- SAP
- Oracle
- Intuit Inc.
- FIS Global
- Fiserv
- Microsoft
- IQVIA
- Constellation Software Inc.
- Infor
- Cox automotive

(Pang, Markovski & Micik, 2020)

Deze informatie is relevanter, maar er moet rekening mee worden gehouden dat de oplossing bedoeld is voor de maakindustrie terwijl IQVIA zich op de zorg richt en Cox Automotive op de verkoop van auto's. Ook is Constellation Software Inc. een moederbedrijf van honderden software bedrijven en is er niet één ERP-system van te halen dat als erg veel gebruikt te beschouwen is.

Vanuit Statista zijn er data gevonden voor percentages van ERP-marktaandeel in november 2016:

- 1 SAP
- 2 Microsoft Dynamics
- 3 Oracle
- 4 Infor

Deze data is minder recent, maar het feit dat er namen in staan die ook in andere bronnen naar voren komen laat wel zien dat dit al langer grote spelers zijn. (Liu, 2018)

Een grafiek van Gartner noemde deze 5 ERP-systemen als houders van de grootste marktaandelen:

- SAP
- Oracle
- Sage
- Workday
- Infor

Interessant is dat Workday op deze grafiek staat maar niet andere grafieken. Microsoft Dynamics staat weer niet op deze, maar wel andere. Wellicht dat ze een andere definitie van ERP hebben. (Gartner, 2017)

Er is geprobeerd van de meest gebruikte ERP-systemen demoversies te krijgen, maar het lijkt dat het bij dit soort programma's niet vaak wordt aangeboden. Alleen voor Microsoft Dynamics wist ik het programma zelf in gebruik te nemen. Voor Oracle ERP lukte het me wel om een soort demo te kunnen bekijken van het programma zelf in gebruik, en hoe er met dat programma een BOM geëxporteerd kan worden. Dit gaf me een goed beeld van hoe er in de praktijk een BOM geëxporteerd word, maar ik kon het helaas niet vergelijken met hoe dat in Trimergo gaat.

6.5.3 Conclusie

In het onderzoek valt op dat SAP consistent de grootste is. Oracle, Infor, Sage en Microsoft Dynamics komen ook vaak in beeld als leveranciers van veel gebruikte ERP-systemen.

Infor en Sage geven de optie om een BOM als XLS-bestand te exporteren. Oracle en Microsoft Dynamics geven de optie om een BOM te exporteren naar Excel, of als een CSV-bestand.
Met SAP is een BOM ook naar Excel te exporteren.

6.6 Hoeveel verschillen andere ERP-systemen met Trimergo?

6.6.1 Aanpak

De ERP-systemen van vorige onderzoeksvraag zijn vergeleken met Trimergo. Aangezien het vooral om het importeren van BOM's gaat heb is daar naar gekeken. Ook werd er bedacht dat deze exportmogelijkheden exportmogelijkheden zijn voor het downloaden van een bestand, niet wat er met de API beschikbaar is, hier heb ik ook naar gekeken. Hiervoor is een combinatie van labonderzoek en biebonderzoek gedaan.

6.6.2 Uitvoering en resultaten

Er is gekeken naar hoe Trimergo een BOM kan exporteren, en geteld welke mogelijkheden er bij de top 5 ERP-systemen zijn.

Dit is het resultaat van dit stuk van het onderzoek:
Trimergo: Exporteert BOM als CSV-bestand.
Infor: Exporteert BOM als XLS-bestand.
Sage: Exporteert BOM als XLS-bestand.
Oracle: Exporteert BOM als CSV-bestand of exporteert naar Excel.
Microsoft Dynamics: Exporteert BOM als CSV-bestand of exporteert naar Excel.
SAP: Exporteert BOM naar Excel.

Van de top 5:
Export naar Excel: Ondersteund door 3/5 systemen.
CSV: Ondersteund door 2/5 systemen.
XLS: Ondersteund door 2/5 systemen.

Voor de mogelijkheden voor het exporteren via een API is er literatuur onderzoek gedaan naar welke data formats er in API's veel gebruikt worden, hier uit bleek dat om 80% van API-responses te ondersteunen JSON en XML ondersteund moeten worden. (Santos, 2020)

6.6.3 Conclusie

De BOM exportmogelijkheden van de top 5 van de vorige deelvraag zijn vergeleken met de exportmogelijkheden van Trimergo. Hieruit bleek dat veel ERP-systemen direct naar Excel exporteren, of als CSV of XLS exporteren. Aangezien Trimergo als CSV exporteert hebben sommige ERP-systemen dezelfde optie, maar de meeste niet. Veel ERP-systemen gebruiken een API voor de export en daarbij is JSON zeer populair, gevold foor XML.

6.7 Hoe kan er rekening met deze verschillen worden gehouden?

6.7.1 Aanpak

Deze onderzoeksvraag is grotendeels een aanvulling op de onderzoeksvraag “Hoeveel verschillen andere ERP-systemen met Trimergo?”. Aan de hand van de bevindingen daar is de data geanalyseerd om een beeld te krijgen waar rekening mee gehouden zou moeten houden om de gewilde 80% ondersteuning naast de ondersteuning van Trimergo te behalen. Hiervoor is lab methodiek gebruikt. Ook is er literatuuronderzoek gedaan naar al bestaande oplossingen voor deze casus.

6.7.2 Uitvoering en resultaten

Er is gekeken naar de resultaten van de onderzoeksvraag “Hoeveel verschillen andere ERP-systemen met Trimergo?” en nagedacht over waar rekening mee gehouden zou moeten houden om de doelstelling van 80% ondersteuning te behalen. Ook is er in het interview van bijlage 11.1 gevraagd naar eventuele data naast een BOM om rekening mee te houden. Uit het literatuur onderzoek naar bestaande oplossingen was er geen nuttige informatie naar voren gekomen. In de HBO kennisbank waren er met de zoektermen “vr erp” 0 resultaten gevonden. Bij EBSCOhost waren het er 20, maar deze resultaten waren allen niet relevant. Daarnaast is er nog gekeken naar de opbouw van een BOM en of er nog verschillen zijn.

6.7.3 Conclusie

Van de top 5 hebben maar twee ERP-systemen dezelfde soort export als Trimergo; CSV.
Als zowel CSV als XLS ondersteund zouden worden zouden er in 80% van de gevallen handmatig verkregen BOMs geïmporteerd kunnen worden. Als er een tussenstap met Excel gemaakt wordt is het percentage hoger. Voor een automatisch systeem is dit echter niet wenselijk.

	Zonder Excel tussenstap	Met Excel tussenstap
CSV ondersteuning	40%	60%
XLS ondersteuning	40%	80%
CSV en XLS ondersteuning	80%	100%

Figuur 1; percentages ondersteunde ERP-systemen bij ondersteuning van bestandtypen.

Dit is echter de ondersteuning voor de bestanden die een ERP-systeem handmatig als een bestand kan leveren. Aangezien het vooral de bedoeling is om met een API te werken zijn de veelvoorkomende datatypen in API responses een interessanter gegeven.

Uit het onderzoek van deelvraag 7 bleek dat 69,5% van API responses als JSON wordt gedaan, en 12,7% met XML. Als deze twee bestandstypen ondersteund worden wordt een percentage van 82% behaald en dus ook de doelstelling van 80%.

Het is belangrijk er rekening mee te houden dat naast bestandstype BOMs ook significant kunnen verschillen qua structuur. Hieronder is bijvoorbeeld een simpele BOM. Deze is heel anders dan een BOM die ik van Hoppmann heb gekregen. Er staan maar twee kolommen in en de kolommen hebben geen namen. De BOM van Hoppmann heeft wel kolomnamen en 20 kolommen. Een andere BOM zou ook zoveel kolommen kunnen hebben, maar dan compleet andere. Zoals in het interview was verteld kan het per klant erg veel verschillen. In BOM's staan soms ook afbeeldingen, maar ze worden geëxporteerd als bestanden die alleen tekst gebruiken. Als de afbeeldingen ook gewenst zijn dan zijn extra stappen misschien nodig om die ook mee te nemen. Sommige BOM's hebben verschillende lagen. In die gevallen kan een onderdeel subonderdelen hebben.

1605 Snow Shovel
13122 Top Handle Assembly (1 required)
457 Top Handle (1 required)
082 Nail (2 required)
11495 Bracket Assembly
(1 required)
129 Top Handle Bracket
(1 required)
1118 Top Handle Coupling
(1 required)
048 Scoop-Shaft Connector (1 required)
118 Shaft (1 required)
062 Nail (4 required)
—
—

Figuur 2; Voorbeeld van een simpele BOM met sub-onderdelen. (Swamidass, 2000)

Uit communicatie met Total Reality tijdens de ontwerpfase van het project is gebleken dat de voorkeur ligt bij het ophalen van gegevens van een ERP-systeem met gebruik van een API. Trimergo ondersteunt dit echter niet. Er is overlegd dat het project nog steeds waarde kan hebben, ook als het niet werkt met Trimergo zoals gehoopt. Wel zou Trimergo eventueel nog ondersteund kunnen worden door via een webinterface handmatig een BOM te uploaden, maar dit is niet ideaal omdat dit de tijdwinst nogal beperkt.

6.8 Conclusie onderzoek

Uit het onderzoek is gebleken dat een generieke oplossing zowel CSV als XLS moet kunnen importeren om 80% van de ERP-systemen te ondersteunen. Door alleen XLS te ondersteunen zou de beoogde 80% alsnog behaald kunnen worden als er een tussenstap met Excel wordt gebruikt, maar aangezien Trimergo als CSV exporteert zal deze toch ondersteund moeten worden.

Een API dient zowel JSON als XML te ondersteunen om meer dan 80% binnen ERP-systemen te halen.

Aangezien deze gegevens de exportmogelijkheden zijn voor het exporteren als een bestand en niet de API heb ik ook onderzoek gedaan naar de datatypen die gebruikt worden in API responses, waar uit bleek dat dat JSON en XML te ondersteunen meer dan 80% behaald wordt

Ook kwam uit het onderzoek het duidelijke beeld naar voren dat Total Reality geïnteresseerd is in het importeren van BOM's en kreeg ik interessante data over welke ERP-systemen het meest in gebruik zijn. Ook bleek dat BOM's zeer veel van elkaar verschillen. Tijdens het ontwerp dient hier rekening mee gehouden te worden.

7 Ontwerp

Na het uitvoeren van het onderzoek was er een goed beeld ontstaan van de eisen voor het te maken systeem en de context van het project. Met deze analyse van de situatie is er overgegaan naar het opstellen van requirements voor een systeem om aan de doelstellingen te voldoen. Uit het onderzoek bleek dat er nog geen bestaande oplossingen bestonden om gebruik van te maken. Er is gekozen voor een REST-service die kon communiceren tussen de VR-app en ERP-systemen. Hiervoor is gekozen omdat ik wist dat voor het communiceren met andere programma's een API te verwachten was. met REST was ik al bekend via de vakken op school en dit leek me een uitstekend moment om deze in de praktijk te gebruiken. Er zijn enkele gesprekken met de belanghebbenden geweest over gewilde eisen aan het systeem. Hierna heb ik er nagedacht over functionaliteiten voor het toekomstige systeem. De gewenste eisen waren nogal beperkt en te behalen met een erg simpel systeem. Ik besloot om de gewenste eisen als must-requirements rekenen. Hierna heb ik de lijst met requirements aangevuld met extra functionaliteiten om de waarde van het systeem te vergroten. Nadat de requirements besproken en goedgekeurd waren door de stakeholders is er een ontwerp gemaakt om aan deze requirements te voldoen. Dit hoofdstuk zal laten zien wat de requirements zijn en toont het ontwerp van de applicatie. Naast het ontwerp van de applicatie zal ook het ontwerp van de database besproken worden.

7.1 Requirements

Na enkele gesprekken met de belanghebbenden is er een voorstel voor een lijst requirements gemaakt. Deze is na een aantal aanpassingen goedgekeurd.
De requirements in deze lijst zijn 'SMART' opgesteld. Dit houdt in dat de requirements Specifiek, Meetbaar, Acceptabel, Realiseerbaar en Tijdgebonden zijn. Daarnaast zijn de requirements geprioriteerd door middel van de MoSCoW methode. De MoSCoW methode heeft vier verschillende prioriteitseenheden.
M –must haves: Deze requirements moeten in het product komen.
S –should haves: Dit zijn requirements die in het product zouden moeten zitten, maar niet compleet essentieel zijn.
C –could haves: Dit zijn requirements die behandeld kunnen worden als er tijd over is.
W –won't haves: Dit zijn requirements voor de toekomst die niet worden behandeld tijdens dit project.

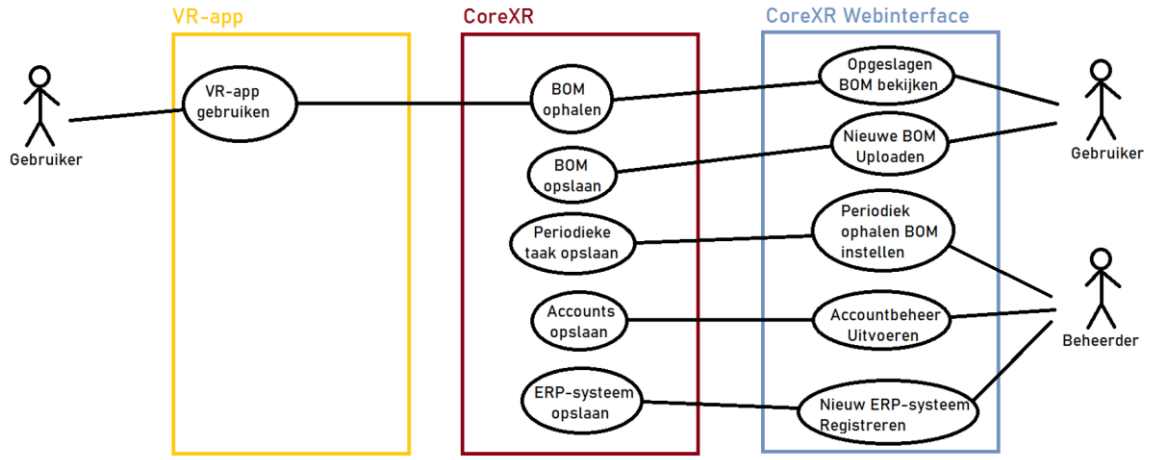
Must
REQ-M-001 Het systeem moet instaat zijn om zich via API van een ERP-systeem zich te authentiseren bij dat ERP-systeem.
REQ-M-002 Het systeem moet een BOM van een ERP-systeem kunnen importeren via een API-call naar het ERP-systeem.
REQ-M-003 Het systeem moet een via API geïmporteerde BOM kunnen converteren van XML naar JSON.
REQ-M-004 De gebruiker moet een BOM in JSON formaat kunnen verkrijgen in de response van een API-call naar het systeem.
REQ-M-005 Het systeem moet API requests kunnen ontvangen terwijl er al een request in behandeling is.
REQ-M-006 Het systeem bezit beveiliging die er voor zorgt dat de API van CoreXR is afgeschermd voor gebruikers die zich niet kunnen authentiseren.

Should
REQ-S-007 Het systeem kan BOM's opslaan voor later gebruik.
REQ-S-008 Het systeem kan een opgeslagen BOM leveren als de API van het ERP-systeem niet beschikbaar is.
REQ-S-009 Het systeem kan meerdere accounts met toegang tot hetzelfde ERP-systeem ondersteunen.
REQ-S-010 Het systeem ondersteunt meerdere ERP-verbindingen per CoreXR installatie.
REQ-S-011 Beheerders moeten nieuwe ERP-systemen toe te kunnen voegen zonder de applicatie af te hoeven sluiten.
REQ-S-012 Beheerders moeten via een webinterface nieuwe accounts kunnen aanmaken.
REQ-S-013 Beheerders moeten via een webinterface accounts kunnen verwijderen.
REQ-S-014 Beheerders moeten via een webinterface accounts kunnen aanpassen.
REQ-S-015 Gebruikers moeten via een webinterface handmatig BOMs kunnen uploaden.

Could
REQ-C-016 Het systeem moet in staat zijn de database automatisch te laten bijwerken door regelmatig een nieuwe BOM op te halen.
REQ-C-017 Het systeem ondersteunt accounts die bevoegd zijn voor meerdere ERP-systemen.

7.2 Use Case Diagram

Om de mogelijkheden qua gebruik met CoreXR te visualiseren is er een Use Case Diagram gemaakt.



Figuur 3; Use Case Diagram

In dit diagram is te zien dat er voor het gebruik van CoreXR drie systemen worden gebruikt; de VR-app, CoreXR zelf, en de webinterface van CoreXR. De VR-app wordt gemaakt door Total Reality en de twee andere systemen door mij.
Ook is te zien dat er twee soorten gebruikers zijn; gewone gebruikers en beheerders.
Gewone gebruikers gebruiken CoreXR voor één ERP-systeem, terwijl beheerders dingen kunnen instellen voor alle ERP-systemen.

Er is gekozen om het voor gebruikers zo simpel mogelijk te maken. De doelstelling was automatisering, dus waren zo min mogelijk extra handelingen van de gebruiker een streven. Bij normaal gebruik zou een gebruiker het hele systeem niet eens moeten opmerken, en vanuit gaande dat de VR-app in staat is om CoreXR inloggegevens op te slaan. De gebruiker start het VR-programma op en het programma haalt vervolgens gegevens op via CoreXR om in het programma te gebruiken.

Er is ook de optie voor gebruikers om een BOM handmatig via een webinterface naar CoreXR te uploaden. Deze optie is bedoeld om CoreXR nog steeds bruikbaar te houden als het ERP-systeem niet met een API te benaderen is. Dit zou kunnen zijn in verband met bijvoorbeeld onderhoud van het ERP-systeem. Het doel van het project is automatisering en handmatig uploaden van BOM's is dan ook bedoeld als een secundaire optie.

Voor beheerders zal het gebruik van CoreXR wel vooral via de webinterface gaan. Hiervoor is gekozen omdat beheer via een XR-app geen duidelijke meerwaarde heeft, er geen XR-hardware voor nodig is, en beheer zonder een visuele interface niet gebruiksvriendelijk zou zijn. Beheerders kunnen voor ERP-systemen instellen dat er periodiek een nieuwe BOM wordt opgehaald, accountbeheer uitvoeren of nieuwe ERP-systemen registreren.

Aangezien tijdens het ontwerpen en ontwikkelen van CoreXR de VR-app ook nog in ontwikkeling was, is er voor gekozen om de verbinding met CoreXR zo simpel mogelijk te maken. Er is alleen een API-call nodig met geldige inloggegevens. Aan de VR-app kant zou er dan ergens moeten worden opgeslagen wat de URL is van deze API en er moet een manier zijn om inloggegevens in te stellen. Aangezien dit vrij simpel is zouden er ook andere systemen met weinig moeite aan CoreXR kunnen worden aangesloten, maar voor de doelstelling van het project is alleen een verbinding met de VR-app nodig.

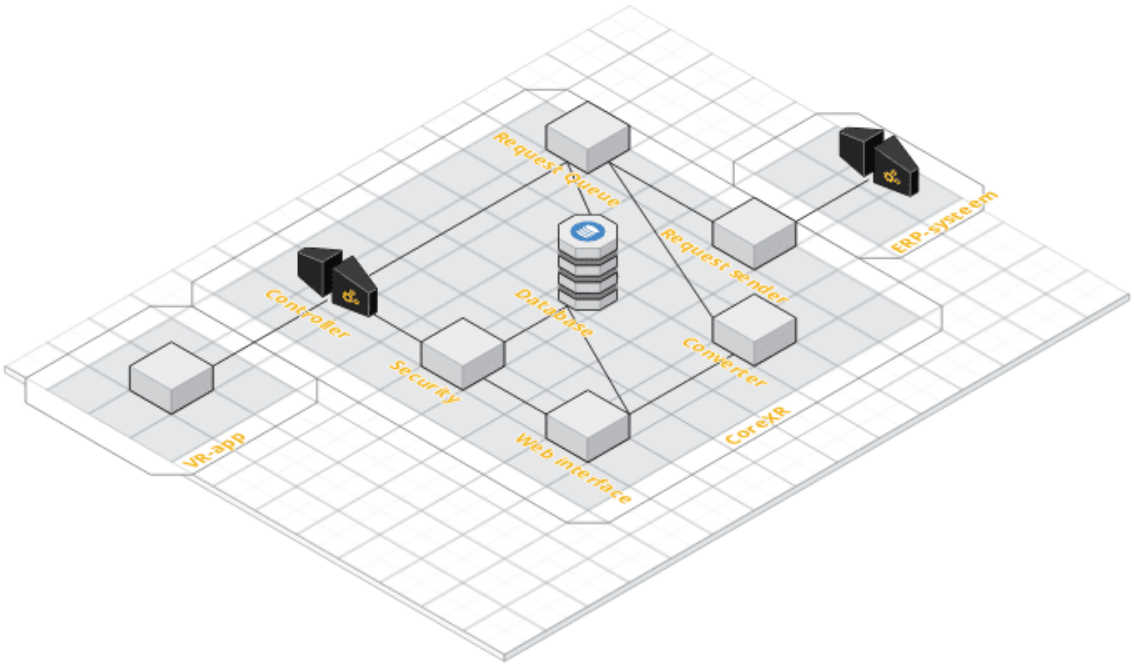
De webinterface zou eventueel in de toekomst nog verder uitgebreid kunnen worden, met bijvoorbeeld ondersteuning voor andere applicaties dan alleen de VR-app.

7.3 Architectuur diagram

Om aan deze requirements te voldoen is er een ontwerp gemaakt. Bij het maken van dit ontwerp is rekening gehouden met een aantal hoofdtaken waar de requirement onder vallen, en om deze te ondersteunen zijn er een aantal componenten ontworpen.
De hoofdtaken van het programma zijn als volgt:

1. In staat zijn API's van ERP-systemen te benaderen. (Requirements M1, M2, S4)
2. Aanspreekbaar zijn via een API. (Requirement M4)
3. Een beveiligde API hebben. (Requirement M6, S3, C11)
4. Bestandstypen om kunnen zetten naar andere bestandstypen. (Requirement M3)
5. Accountbeheer ondersteunen. (Requirements S6, S7, S8)
6. Aanspreekbaar zijn via een webinterface. (Requirements S6, S7, S8, S9)
7. Gegevens bij kunnen houden in een database. (S1, S2, S6, S7, S8, S9, C10, C11)
8. Generiek genoeg zijn om bruikbaar te zijn met veel ERP-systemen. (S4, S5, C11)

De componenten om deze hoofdtaken te ondersteunen zijn gevisualiseerd in dit diagram;



Figuur 4; CoreXR architectuur diagram

In dit diagram zijn de volgende componenten te vinden: Een controller component, een security component, een web-interface component, een database component, een converter component, een request queue component en een request sender component.

Controller
Hoofdtaak: 2
Voor de hoofdtaak om aanspreekbaar te zijn via een API wordt er een controller component gebruikt. Deze heeft een aantal API aanspreekpunten die te vinden zijn in hoofdstuk **Fout! Verwijzingsbron niet gevonden.**.
Hier is voor gekozen omdat als tussenlaag-programma het de bedoeling is dat CoreXR aan te spreken zal zijn door een ander programma; namelijk de VR app. Ook de webinterface communiceert met CoreXR via de API. Er is voor gekozen om CoreXR een API te geven omdat dit project het doel heeft om Total Reality tijd en moeite te besparen die zou gaan zitten in het opzetten en gebruik van de VR-app, en het aanspreken van een API kan geautomatiseerd worden. De VR-app vraagt via de API CoreXR om de gegevens en die worden geleverd zonder dat de gebruiker hier handmatig iets voor hoeft te doen.

Security
Hoofdtaak: 3
Voor de beveiliging van de API is er een component genaamd “Security”. Dit component zorgt ervoor dat niet iedereen gegevens via CoreXR op kan halen. In de database staat opgeslagen welke accounts er zijn, de inloggegevens van de accounts, en welke ERP-systemen deze accounts mogen benaderen. Ik heb ervoor gekozen om de API te beveiligen met *Spring Security* aangezien dit de standaard beveiliging is van Spring. Het voordeel hiervan is dat er van uit kan worden gegaan dat het werkt op CoreXR, die op Spring draait. Ook bij dit component wordt de database gebruikt. De accounts staan namelijk opgeslagen op de database. Het voordeel hiervan is dat deze aangepast kunnen worden zonder CoreXR zelf op nieuw op te hoeven starten. I.v.m. veiligheid worden wachtwoorden geëncrypt opgeslagen. Hier is voor gekozen omdat er dan in het geval van een database-lek niet een lijst met wachtwoorden op straat ligt.

Request Sender
Hoofdtaak: 1, 8
Voor hoofdtaak 1 is er het component “request sender”. Dit component heeft de functionaliteit om HTTP requests te sturen. Deze zal wel een URL nodig hebben en eventueel ook authenticatie gegevens. Deze worden opgehaald uit een database, deze wordt ook als een component gezien.

Ik heb gekozen voor het maken van een apart component om het sturen van API requests uit te voeren omdat ik verwachtte dat het project zo complex genoeg zou zijn dat het zeer onoverzichtelijk zou worden als het niet opgedeeld zou worden in componenten. Er is gekozen voor het gebruik van een database omdat om requirement S1 te behalen gegevens niet verloren moeten gaan als het programma uitgeschakeld word, ook zijn er met databases veel mogelijkheden voor het maken van back-ups. Daarom lijkt het gebruik van een database mij de logische keuze.

Om CoreXR generiek te houden kunnen er in de database meerdere ERP-systemen worden opgeslagen met API-URL en eventuele (geëncrypte) authenticatie gegevens, zo hoeft het programma niet elke keer worden aangepast voor een verbinding met de API van een ander ERP-systeem. Het request sender component is dan ook bedoeld voor gebruik met allerlei verschillende ERP-systemen.

Request queue
Hoofdtaak: 2
Het doel van dit component was om te regelen dat API-requests die binnen komen terwijl er al een API-request in behandeling is in een wachtrij komen zodat ook die uitgevoerd worden zodra dit mogelijk is. Dit component is echter komen ter vervallen, het bleek bij de implementatie al snel dat het gebruikte framework synchrone API-requests kan afhandelen, en dit component daarom niet nodig is.

Database
Hoofdtaak: 7 (ondersteunend voor andere hoofdtaken)
Het gebruik van de database component komt bij veel hoofdtaken voor, deze component is dan ook bij veel andere hoofdtaken ondersteunend. Voor accountbeheer bijvoorbeeld wordt er gebruik gemaakt van dit component om de accountgegevens bij te houden en aan te passen, en bij beveiliging worden deze gegevens ook gebruikt. In hoofdstuk 8, implementatie, staat meer informatie over hoe de database gebruikt wordt voor verschillende doelen.

Ook wordt de laatst door CoreXR opgehaalde BOM opgeslagen in de database zodat deze nog steeds beschikbaar is als de verbinding met een ERP-systeem is weggevallen. Er kan worden ingesteld dat deze automatisch wordt bijgewerkt door periodiek automatisch het ERP-systeem om een nieuwe versie te vragen. Hiervoor is gekozen om zo de opgeslagen BOM recent te houden, als het ERP-systeem dan wegvalt heeft CoreXR recente gegevens opgeslagen, ook als hier niet handmatig om gevraagd is. Meestal zal de gebruiker gewoon een nieuw opgehaalde BOM van het ERP-systeem krijgen, wat ook de bedoeling is. Er is gekozen om dit toe te voegen om CoreXR extra betrouwbaar te maken. Het algemene principe erachter is om CoreXR ook te laten werken als het ERP-systeem niet beschikbaar is.

Dit wordt gedaan door elke keer als er een nieuwe BOM wordt opgehaald van een ERP-systeem deze op de database te zetten. Als er al een opgeslagen BOM is dan word deze overschreven. Het nadeel hiervan is dat er constant met de database gecommuniceerd wordt. Echter, het is niet altijd mogelijk is van te voren te weten wanneer communicatie met een ERP-systeem niet beschikbaar is. Daarom is het ook niet mogelijk te weten welke opgehaalde BOM de laatste zal zijn. Om de meest recente BOM beschikbaar te hebben is er voor gekozen om elke keer op te slaan.

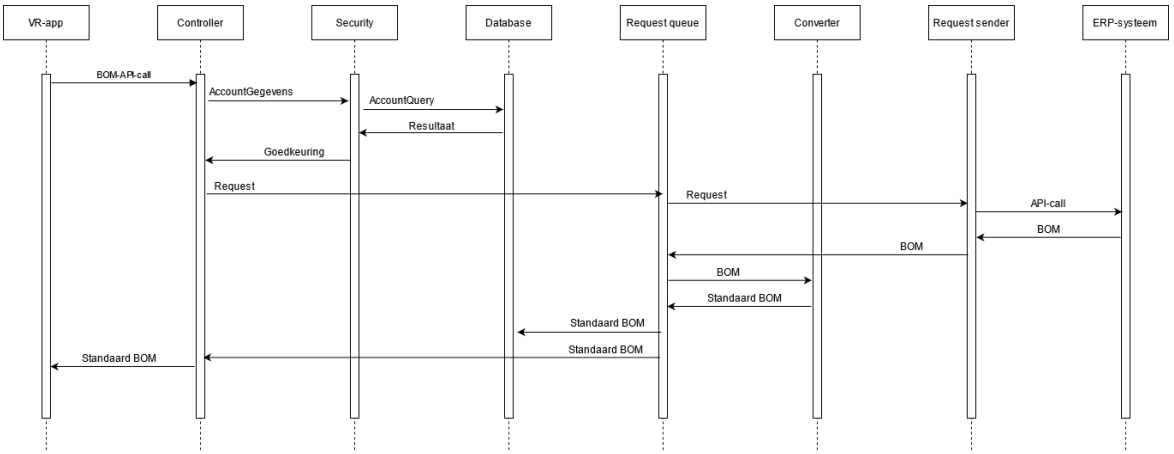
Een mogelijke verbetering hiervan zou het gebruik van een cache kunnen zijn. Als de meest recent opgeslagen BOM om in een cache staat zou er bij het ophalen van een nieuwe BOM gekeken kunnen worden of deze hetzelfde is als de opgeslagen BOM. Zo ja, dan hoeft er niet naar de database geschreven te worden. Er zal wat geëxperimenteerd moeten worden om te kijken of het vergelijken van BOM's tijd bespaart vergeleken met het direct opslaan. I.v.m. beperkte beschikbare tijd voor dit project is het gebruik van een cache iets wat niet opgepakt is.

Naast de optie om een database te gebruiken was er ook een optie om bepaalde gegevens op te slaan in een cache. Dit heeft het voordeel dat het sneller is dan een database. Het heeft echter ook een nadeel; de beschikbare ruime is ook beperkter. Aangezien het niet bekend is hoeveel gebruikers CoreXR zal krijgen is er voor gekozen om het bij het gebruik van een database te laten. Dit vermindert het risico dat de cache vol raakt. Een cache is ook meer bedoeld voor tijdelijke opslag, terwijl de data die op de database komt bedoeld is voor opslag voor langere tijd. Mocht in de toekomst CoreXR worden uitgebreid met functies waarbij gegevens moeten worden opgeslagen voor kortere tijd, zou er overwogen kunnen worden een cache te gebruiken.

Converter

Hoofdtaken: 4, 8

Voor het omzetten van bestand types naar andere bestand types is er het “Converter” component. Dit component is een verzameling van functies voor bestandstype-conversie. Aangezien Total Reality voor hun VR-app JSON bestanden gebruikt zal dit voor het ophalen van een BOM via API XML naar JSON zijn. Omdat uit onderzoek bleek dat door deze twee datatypen te ondersteunen de beoogde 80% ondersteuning behaald wordt. Voor het uploaden via de webinterface zal dit XLS en CSV zijn, aangezien het onderzoek aantoonde dat hiermee meer dan 80% van de ERP-systemen ondersteund wordt als het gaat om handmatig verkregen BOM’s. Het is de bedoeling dat dit component zo ingericht wordt dat deze gemakkelijk uit te breiden is, zo kan met een simpele aanvulling CoreXR ook worden gebruikt voor ERP-systemen die niet onder de verwachte 80% vallen. Figuur 5 hieronder is een sequence-diagram die aantoont hoe een BOM request in CoreXR afgehandeld wordt.



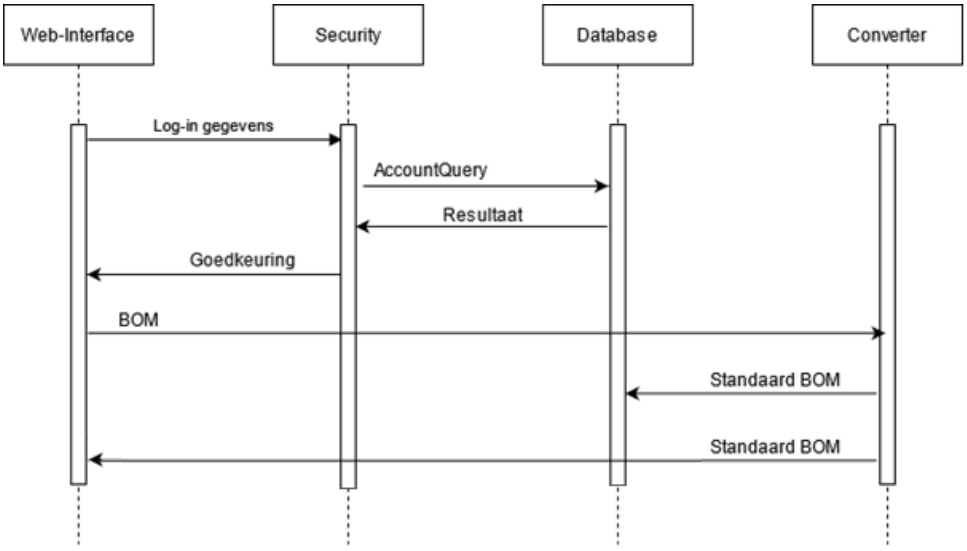
Figuur 5; Sequence diagram van een BOM-request

Web-interface

Hoofdtaken: 5 en 6

Voor accountbeheer zal er met de database gecommuniceerd moeten worden. Dit kan direct via de API, maar i.v.m. gebruiksvriendelijkheid is hier een ook webinterface voor. Deze is te openen met een browser, waar een gebruiker na het inloggen een visuele interface heeft voor accountbeheer.

Deze web interface kan ook gebruikt worden om handmatig gegevens naar CoreXR te uploaden. Deze optie is handig omdat CoreXR dan nog te gebruiken is in het geval dat een API verbinding met een ERP-systeem geen optie is. Hiermee kan het programma ERP-systemen zonder beschikbare API ondersteunen, bijvoorbeeld als een ERP-systeem technische problemen heeft. In figuur 6 is te zien hoe het uploaden van een BOM via de webinterface werkt.



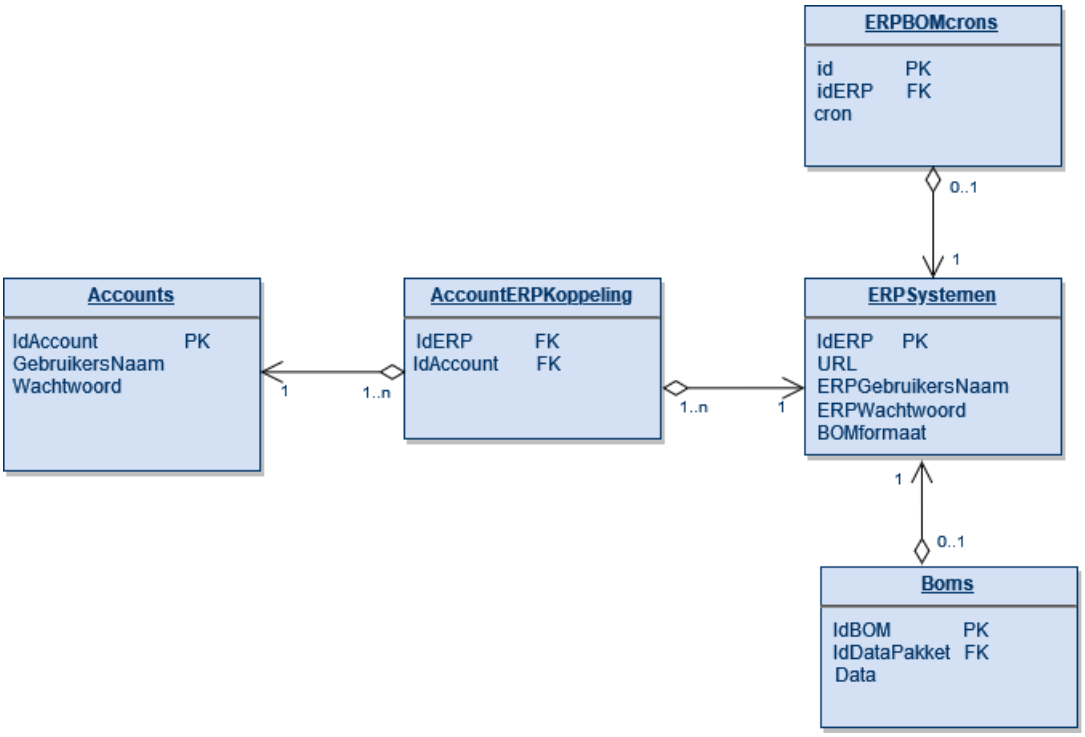
Figuur 6; BOM upload via webinterface

7.4 CoreXR API

Om te communiceren met CoreXR wordt er een API gebruikt. Voor de API is er gekozen voor de architectuurstijl REST. REST staat voor Representational State Transfer. Hier is voor gekozen omdat het aannemelijk is dat er in de toekomst andere ontwikkelaars aan CoreXR zullen werken. REST API is erg populair en het is aannemelijk dat toekomstige ontwikkelaars hier bekend mee zullen zijn. Ook is REST API erg flexibel, ideaal voor een prototype waar nog verder aan gewerkt kan worden.

In bijlage 12.2 staat een overzicht van de API van CoreXR.

7.5 Database design



Figuur 7; Database diagram

De database is ontworpen voor het bijhouden van accounts, ERP-systemen en BOMs.
Als aan alle requirements wordt voldaan kan een account meerdere ERP-systemen aanspreken en kan een ERP-systeem door meerdere accounts worden aangesproken. Voor deze “many to many” relatie is er een “bridge table” genaamd “AccountERPKoppeling” waarin wordt bijgehouden welke accounts bij welke ERP-systemen mogen komen.
Met deze informatie is op te halen welk account bij welk ERP-systeem kan, en wat de URL is voor het aanspreken van de API van het ERP-systeem is. Er staan ook wachtwoorden in de database, deze zijn gehasht in verband met veiligheid. Ook staan er in de database de laatst opgehaalde versies van ERP-systemen. Deze zijn gelinkt aan ERP-systemen. Per ERP-systeem wordt ook opgeslagen welk formaat BOM er geïmporteerd wordt, waar de converter dan rekening mee kan houden.

Ook wordt er in de tabel ERPBOMcrons “crons” bijgehouden. Deze crons of “cron jobs” zijn bedoeld om bij te houden voor welke ERP systemen er op welk moment periodiek een nieuwe BOM opgehaald moet worden. (Requirement C10). Het gebruik van cron-expressies in CoreXR is geïnspireerd door de cronjobs van Unix-systemen. Daar is het een commando dat een script op een ingesteld tijdstip uitvoert. Deze cronjobs bestaan uit een reeks van vijf velden. Het eerste veld staat voor minuten, het tweede voor uren, gevold door dagen, weken en weekdagen. Met deze velden kan worden beschreven op welk moment de taak moet worden uitgevoerd. Bij het

veld voor dagen kan bijvoorbeeld “5” worden ingevuld, wat betekend dat de taak de 5e van de maand moet worden uitgevoerd. Welke maand dit is ligt aan wat in dat veld is ingevuld. Als daar “1” staat wordt de taak uitgevoerd op 5 januari. Naast getallen kan er in een veld ook een * worden geplaats om iets te herhalen. Dus als deze bij uren ingevuld wordt dan zal de taak elk uur worden uitgevoerd, er van uitgaande dat de andere velden dat toelaten. Ook is het mogelijk om bijvoorbeeld */5 in een veld te vullen, wat om de 5 herhaald. (Om de vijf minuten, uren, dagen, etc.)

Als voorbeeld; de cron-expressie */5 * 5 1 0 Zou een taak uitvoeren om de vijf minuten, elk uur, als het de vijfde van de maand is, als het januari is en het zondag is.

Het was ook mogelijk om in de plaats van een systeem met cron-expressies simpelweg de optie te geven een aantal milliseconden in te stellen en vervolgens om de zoveel milliseconden een BOM te halen, maar er is voor gekozen dit toch niet te doen. Het zou technisch simpeler zijn, maar met cron-expressies is veel meer mogelijk. Aangezien het een doelstelling van dit project is om een generieke oplossing te ontwikkelen is er gekozen voor de ingewikkeldere optie.

8 Implementatie (Beroepsproduct)

Voor het uiteindelijke prototype is aan alle requirements voldaan.

Omdat er een REST-service in Java gemaakt zou worden is er een Spring project gemaakt. Spring is een populair framework waarvoor is gekozen omdat mijn begeleider er ervaring mee had en ik dus in staat was complicaties op te lossen, het gratis beschikbaar is, en er wegens de grote populariteit op het internet veel informatie en ondersteuning beschikbaar is.

Er is tijdens het project gebleken dat verbinden met het ERP-systeem van Hoppmann geen optie was. Dat ERP-systeem ondersteunt namelijk niet het exporteren van een BOM via een API. In overleg met de stakeholders was deze eis komen te vervallen. Aangezien CoreXR niet alleen voor dat systeem werkt had het project echter nog steeds waarde en kon het ontwikkelen van CoreXR doorgaan. Dit heeft wel het nadeel dat er niet een ERP-systeem beschikbaar was op te testen, afgezien van een test omgeving van Microsoft Dynamics waar helaas geen echte BOM op staat. Wel kan er andere data van worden opgehaald die laat zien dat het technisch wel werkt.4

Implementatie **REQ-M-001**:
"Het systeem moet instaat zijn om zich via API van een ERP-systeem zich te authentiseren bij dat ERP-systeem."
Deze requirement is geïmplementeerd door op een database inloggegevens op te slaan van ERP-systemen. De controller haalt deze op en gebruikt deze om bij een API-call naar ERP-systemen te autoriseren. De Request Sender heeft een functie getBOMBasicAuth, deze wordt door Controller aangeroepen als er een BOM moet worden opgehaald van een ERP-systeem. Aan deze functie kan een URL, gebruikersnaam en wachtwoord mee worden gegeven. Er is gekozen om dit op een database te zetten omdat het niet praktisch leek om telkens aan CoreXR de inloggegevens van ERP-systemen mee te geven als deze een BOM op moet halen, dit zou de API van CoreXR onnodig ingewikkelder maken en zou betekenen dat deze gegevens op een andere manier bijgehouden moeten worden. Deze gegevens worden meegegeven in API-calls naar ERP-systemen, en zo kan CoreXR zich authenticeren. Op het moment werkt dit alleen met Basic Authentication. Er is geprobeerd het met OAuth2 te laten werken maar daar is zoveel tijd in gaan zitten dat er uiteindelijk is besloten dat voor dit prototype nog niet te ondersteunen.

Implementatie database:
Voor de database is er besloten om MySQL te gebruiken. De reden voor deze keuze is dat MySQL zowel gratis is, als het een populaire keuze, wat het makkelijker maakt om ondersteuning te vinden bij eventuele moeilijkheden. Verder had MySQL een voorkeur wegens eerdere ervaring
Om Spring met de database te verbinden is er gebruik gemaakt van Spring JPA, dit is een populaire keuze en is specifiek gemaakt voor Spring, twee redenen waarom er gekozen is om er gebruik van te maken. Classes zijn gemaakt die overeenkwamen met wat er op de database staat, zoals gebruikers en ERP-systemen, en door annotation wist Spring naar welke tabel en welke kolom de variabelen in die class verwezen. Ook zjin er CrudRepository classes gemaakt, deze repositories bevatten de objecten die overeenkomen met wat er op de database staat. Als er bijvoorbeeld op de database 10 gebruikers staan, dan staan er in de gebruikers repository 10 objecten met dezelfde gegevens. Als de repository aanpast wordt dan worden er door Spring JPA ook queries gemaakt om de database aan te passen, en op die manier kan er met Java de gegevens op de database gebruikt worden.

Implementatie **REQ-M-002**:
"Het systeem moet een BOM van een ERP-systeem kunnen importeren via een API-call naar het ERP-systeem."
Deze requirement is behaald met een kleine uitbreiding op de code die gebruikt is om requirement M1 te behalen. De response van API-calls naar een ERP-systeem wordt namelijk terug gegeven aan de Controller, zo worden BOM's geïmporteerd. Op de database staat per ERP-systeem naar welke URL een API-call moet worden gestuurd om een BOM te verkrijgen.

Implementatie **REQ-M-003**:
"Het systeem moet een via API geïmporteerde BOM kunnen converteren van XML naar JSON."
CoreXR beschikt in de converter over een functie om XML naar JSON om te zetten. Total Reality vereist een BOM als JSON via de API van CoreXR verkrijgen, en alhoewel veel ERP-systemen dit ook als JSON leveren bleek uit het onderzoek dat XML ook redelijk veel voorkomt. Er wordt per ERP-systeem naast de URL en eventuele inloggegevens ook op de database bijgehouden welk bestandstype BOM ze leveren. Als dit JSON is dan geeft CoreXR gewoon de BOM door zonder aanpassingen. Is dit XML, dan converteert CoreXR deze BOM naar JSON voordat die geleverd wordt. Voor het converteren van XML naar JSON is de simpele functie XML.*toJSONObject* gebruikt, het was een optie om zelf een functie te maken om te converteren, maar het was qua tijd veel efficiënter een al bestaand middel te gebruiken. Deze class is opzettelijk zo simpel mogelijk gehouden omdat deze bedoeld is om gemakkelijk uit te breiden. Een extra datatype ondersteunen kan door in de Controller na de check of het bestandstype XML is nog een check te maken die kijkt naar het nieuw te ondersteunen bestandstype. Deze string moet overeenkomen met het bestandstype dat op de database staat in de tabel van het ERP-systeem staat. In deze check is dan door te verwijzen naar een nieuwe functie in de converter die het bestandstype omzet naar JSON. De check voor XML kan als voorbeeld worden gebruikt.

Implementatie **REQ-M-004**:
"De gebruiker moet een BOM in JSON formaat kunnen verkrijgen in de response van een API-call naar het systeem."
Aan deze requirement wordt voldaan door de eventueel naar JSON geconverteerde BOM via Spring als response te geven voor een API-call voor een BOM. Iets mee te geven in een response bleek standaard functionaliteit te zijn van Spring en was daarom een erg simpel onderdeel om te maken; de JSON is simpelweg de tekst in de response van de API-call. Dit wordt in Controller gedaan nadat deze eventueel geconverteerd is. Er is besloten dit niet ingewikkelder te maken dan nodig is zonder dat dit een duidelijk voordeel oplevert, en dus de standaard functionaliteit van Spring te gebruiken.

Implementatie **REQ-M-005**:
"Het systeem moet API requests kunnen ontvangen terwijl er al een request in behandeling is."
Deze requirement bleek erg simpel te behalen te zijn. Eerst was er overwogen een soort wachtrij te maken voor te behandelen taken die met de API zijn binnen gekomen, maar het bleek dat Spring al van nature deze requirement ondersteund. Dit staat standaard ingesteld op 200 requests tegelijk, maar deze waarde is aan te passen door *server.tomcat.threads.max* in de *application.properties* in te stellen. Na een vraag aan Total Reality over hoeveel gebruikers ze tegelijk verwachten bleek dat dit nog niet te zeggen valt, dus was er besloten het op de standaard waarde te laten staan. Mocht CoreXR in de toekomst veel gebruikt worden dan zou deze waarde omhoog gehaald kunnen worden, maar dan zal het ook een zwaardere applicatie worden om te draaien.

Implementatie **REQ-M-006**:
"Het systeem bezit beveiliging die er voor zorgt dat de API van CoreXR is afgeschermd voor gebruikers die zich niet kunnen authenticeren."
CoreXR maakt gebruik van Spring Security. Er is gekozen voor dit *framework* omdat het gemaakt is voor Spring en er veel aan te passen valt. Aangezien het prototype veel veranderingen kreeg tijdens de ontwikkeling was deze aanpasbaarheid erg gewild. Omdat er met een database wordt gewerkt is het een en ander overschreven, Spring Security werkt namelijk standaard op een in-memory manier die niet dynamisch genoeg is voor dit project. Wat interessant is voor het project is dat er gebruikers en ERP-systemen worden bijgehouden en dat niet elke gebruiker voor elk ERP-systeem bevoegd is. Ook kan een gebruiker voor meerdere ERP-systemen bevoegd zijn. In de database wordt dit ondersteund met een *many-to-many* relatie tussen ERP-systemen en gebruikers, hiervoor is een bridge-tabel gemaakt. Het authenticeren gebruikt maakt gebruik van basic authentication. OAuth2 werd overwogen, maar wegens tijdsoverwegingen is dat niet gebruikt. Als de gebruiker wil authenticeren wordt dit gedaan in een class die de Authentication van Spring Security overschrijft. Dit is gedaan omdat er de standaard functie geen ondersteuning biedt om na te kijken of een gebruiker ook voor een bepaald ERP-systeem bevoegd is en niet de encryptie ondersteunt die gebruikt is voor gevoelige gegevens in de database.
CoreXR encrypteerd namelijk de opgeslagen wachtwoorden van gebruikers, dit is gedaan voor extra veiligheid in het geval van een database lek. Op de database staat naast de geëncrypteerde waarde ook de gebruikte salt opgeslagen, deze wordt bij het authenticeren gebruikt om het meegegeven wachtwoord te kunnen vergelijken met het opgeslagen wachtwoord. Bij het maken van een account wordt een willekeurige salt gegenereerd, dit is gedaan om geëncrypteerde wachtwoorden unieker te maken (twee dezelfde wachtwoorden met een andere salt hebben een andere geëncrypteerde waarde, dus zijn dubbele wachtwoorden niet te zien in de database) en minder risico te lopen op aanvallen met hash-tables. De wachtwoorden zijn geëncrypteert met PBKDF2, Er is voor deze methode gekozen omdat deze aangeraden wordt voor nieuwe applicaties in RFC 8018 (Moriarty, Kaliski, & Rusch, 2017). Ook is de sterkte van het algoritme in te stellen door een aantal iteraties aan te passen, wat kan helpen tegen brute-force aanvallen. Als computers sneller worden kan deze sterkte verhoogd worden. Dit vertraagt echter ook het encrypteren van nieuwe wachtwoorden. Dit kan helpen bij de toekomstbestendigheid van deze keuze.

Implementatie **REQ-S-007**:
"Het systeem kan BOM's opslaan voor later gebruik."
CoreXR kan BOM's opslaan in een database, dit is gedaan ter ondersteuning van requirement S2. Wanneer CoreXR een verzoek voor het leveren van een BOM van ERP-systeem ontvangt dan zal het programma met een API-call naar het ERP-systeem deze ophalen en doorgeven, zoals dat in requirements M2 en M4 beschreven staat. Het opslaan van deze BOM gebeurt elke keer als CoreXR een nieuwe BOM ontvangt. Op de database staat een tabel met BOM's, van elk ERP-systeem wordt de meest recente BOM daarin opgeslagen. In deze tabel staat per BOM een automatisch gegenereerd id van de BOM, de id van het ERP-systeem waarvan deze BOM is, de datum waarop CoreXR deze BOM opgehaald heeft en de data van de BOM zelf. De data wordt opgeslagen als een *blob*, dit had ook gekund als een *varchar* waarde, maar er is gekozen voor een *blob* omdat de mogelijkheid bestaat dat CoreXR in de toekomst ook eventueel andere dingen dan BOM's zou kunnen leveren, en deze zouden mogelijk niet met een *varchar* op de database kunnen worden opgeslagen. Om al vast in het prototype te laten zien hoe er met een *blob* gewerkt zou kunnen worden wordt deze nu al gebruikt. De data van de BOM wordt opgeslagen als een byte-array van de US_ASCII charset. Dit maakt het makkelijker om de applicatie in de toekomst uit te breiden.

Implementatie **REQ-S-008**:
"Het systeem kan een opgeslagen BOM leveren als de API van het ERP-systeem niet beschikbaar is."
Als het programma bij het ophalen van een BOM van een ERP-systeem er niet in slaagt deze te krijgen zal de opgeslagen BOM van de database worden opgehaald en deze worden geleverd. Deze is opgeslagen als een *US_ASCII charset bytearray* en wordt eerst weer omgezet naar een *string* voordat deze geleverd wordt. Ook wordt de datum van het ophalen van de BOM meegegeven, hierin is te zien hoeveel de BOM verouderd is.

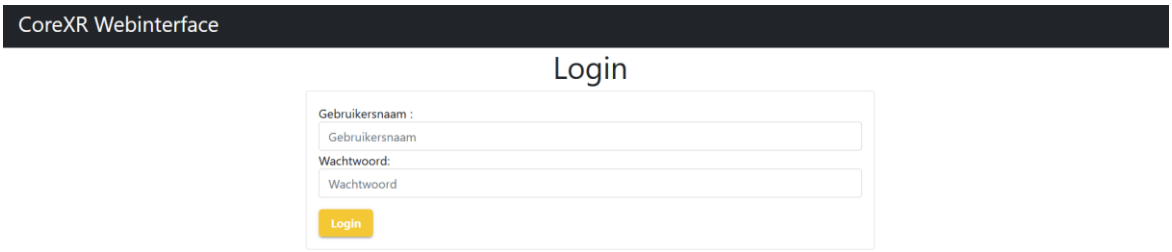
Implementatie **REQ-S-009**:
"Het systeem kan meerdere accounts met toegang tot hetzelfde ERP-systeem ondersteunen."
In CoreXR kan een ERP-systeem meerdere accounts hebben die allemaal bevoegd zijn. In requirement M6 staat al beschreven hoe dit geregeld wordt.

Implementatie **REQ-S-010**:
"Het systeem ondersteunt meerdere ERP-verbindingen per CoreXR installatie."
Om deze requirement te ondersteunen wordt er weer gebruik gemaakt van een database. Op de database staat een tabel met ERP-systemen en de benodigde gegevens per ERP-systeem. Als er via een API-call naar CoreXR bijvoorbeeld een BOM opgevraagd wordt dan zal er ook een id van het ERP-systeem meegegeven moeten worden. De gegevens bij dat id worden dan van de database opgehaald, zoals bijvoorbeeld de URL waar CoreXR de BOM van op moet halen. Er is gekozen om de database weer te gebruiken omdat deze dynamisch is, er back-ups gemaakt van kunnen worden en omdat de gegevens blijven staan ook als CoreXR opnieuw opgestart zou worden.

Implementatie **REQ-S-011**:
"Beheerders moeten nieuwe ERP-systemen toe te kunnen voegen zonder de applicatie af te hoeven sluiten."
Deze requirement is behaald tijdens de implementatie van S4; aangezien er gebruik gemaakt wordt van een database is het geen probleem om CoreXR aan te laten staan als hier aanpassingen worden gemaakt.

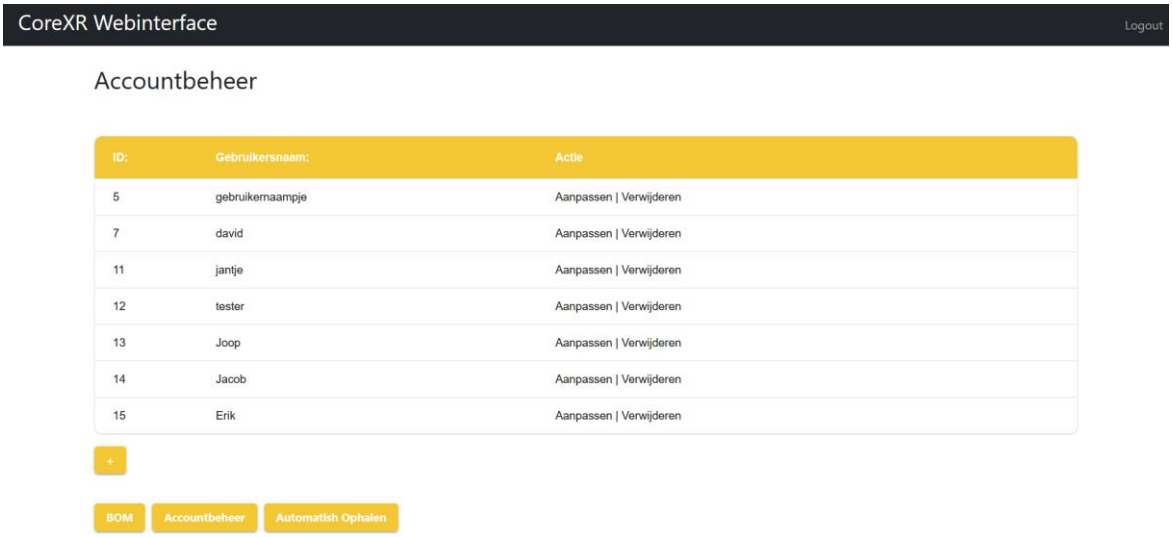
Implementatie **REQ-S-012**:
"Beheerders moeten via een webinterface nieuwe accounts kunnen aanmaken."
Voor de webinterface is er gebruik gemaakt van Angular. Er is voor Angular gekozen omdat er al ervaring met dit platform was, het een populaire keuze is en er bij OVSoftware veel kennis is met Angular, wat hielp bij het oplossen van complicaties.
Als eerst is er met Angular een inlogpagina gemaakt, aangezien CoreXR geldige inloggegevens vereist. Voor het ondersteunen van inloggen worden er drie TypeScript classes gebruikt, login.component, auth.service en httpInterceptor.service. De login.component is het script van de login html pagina, waar de gebruiker kan inloggen. Als de gebruiker dat doet dan geeft de class het ingevulde gebruikersnaam en wachtwoord door naar auth.service. Deze class doet vervolgens een API-call naar CoreXR met een token gemaakt van de inloggegevens, en in een response wordt er gelezen of de inloggegevens zijn goedgekeurd. Is dat het geval, dan worden ze door auth.service opgeslagen als session atributes en wordt de gebruiker ingelogd en mag de gebruiker door naar de hoofdpagina. De opgeslagen inloggegevens worden gebruikt door httpInterceptor.service, deze zal automatisch de inloggegevens meegeven als er tijdens de sessie API-calls worden gedaan. Dit is zo gedaan omdat de gebruiker dan niet elke keer opnieuw inloggegevens hoeft in te vullen. Als de gebruiker uitlogt of de website sluit worden de inloggegevens weer vergeten. Er is geprobeerd zoveel mogelijk een veel gebruikte manier van inloggen te vinden, en op het internet werd het vaak het op deze manier gedaan, dus werd voor CoreXR hetzelfde gedaan. Er was wel wat moeite met de verbinding tussen CoreXR en de webinterface, CSRF bescherming blokkeerde API-requests, maar i.v.m. beveiligingsredenen was het niet verstandig geacht deze simpelweg uitschakelen. Dit is uiteindelijk opgelost door de

standaard CSRF bescherming uit te schakelen en een eigen CSRF-filter te maken. Deze wordt in de plaats van de standaard CSRF bescherming gebruikt, en er wordt in dat filter een uitzondering gemaakt voor de webinterface, die wel API-requests naar CoreXR mag doen. Er wordt in Angular een CSRF token aan gemaakt om aan te tonen dat de request echt van de webinterface komt.



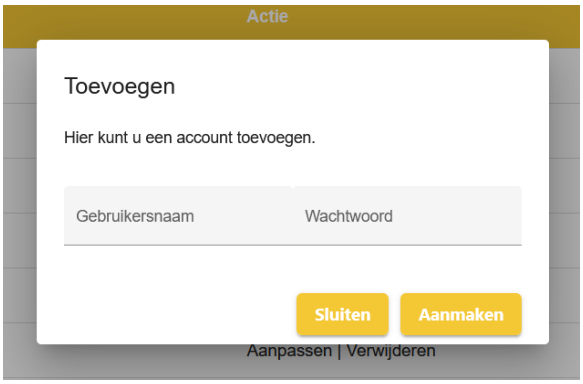
Figuur 8; CoreXR Webinterface Login-scherm

De eerste functionaliteit voor de webinterface was het aanmaken van nieuwe accounts. Hiervoor is er een tabel gemaakt op de hoofdpagina waar alle accounts op worden getoond, deze worden automatisch opgehaald na het inloggen. Voor API-calls zoals het ophalen van gebruikers is er de class corexr.service gemaakt, deze bevat aan te roepen functies die API-calls naar CoreXR doen. Deze class is gemaakt omdat het overzichtelijk werd gevonden om de API-calls op dezelfde plek te hebben. De standaard HttpClient van Angular wordt gebruikt om HTTP requests te sturen.



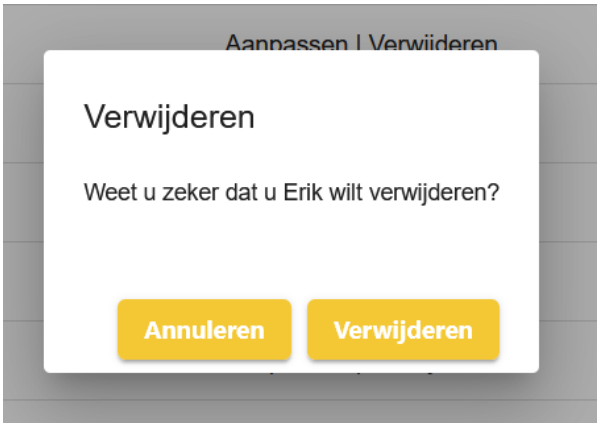
Figuur 9; CoreXR Webinterface Accountbeheer scherm

Links onder de tabel is er een knop te vinden met een plus icoon, deze knop opent een dialog venster waarmee een nieuw account aan te maken is. Deze roept een functie corexr.service aan om die API-call te sturen.



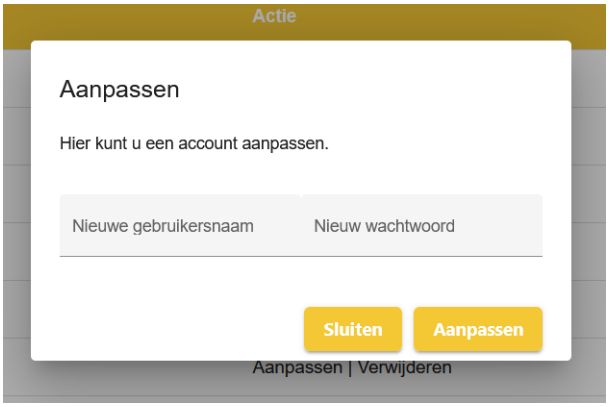
Figuur 10; Account toevoegen dialog venster

Implementatie **REQ-S-013**:
"Beheerders moeten via een webinterface accounts kunnen verwijderen."
Voor deze requirement is de tabel met gebruikers uitgebreid met een kolom voor acties, zoals het verwijderen van gebruikers. Door er op te klikken opent er een dialog venster om te vragen of de gebruiker het zeker weet, om zo de kans op het per ongeluk verwijderen van gebruikers te verminderen. Deze optie werkt technisch gezien het bijna hetzelfde als het toevoegen van accounts, alleen wordt er ook nog bijgehouden welk account in de tabel bedoeld wordt.



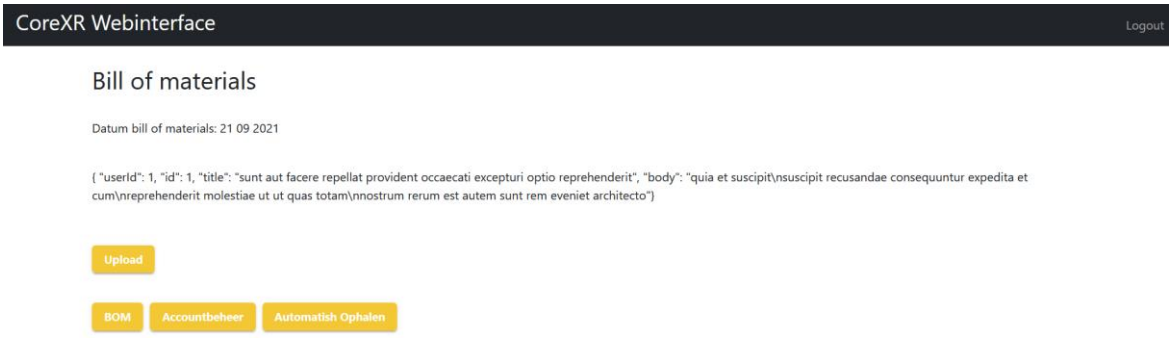
Figuur 11; Account verwijderen dialog venster

Implementatie **REQ-S-014**:
"Beheerders moeten via een webinterface accounts kunnen aanpassen."
Naast de optie om een account te verwijderen is er ook een optie om een account aan te passen. Deze werkt vrijwel op dezelfde manier als het verwijderen van een account.



Figuur 12; Account aanpassen dialog venster

Implementatie **REQ-S-015**:
“Gebruikers moeten via een webinterface handmatig BOMs kunnen uploaden.”
Het is mogelijk voor gebruikers om via de webinterface handmatig een BOM te uploaden. Deze optie zal waarschijnlijk zelden gebruikt worden, maar er zou een situatie kunnen zijn waarop iemand een BOM van een ERP-systeem wil gebruiken die niet bij via API door CoreXR te bereiken valt. Dit zou kunnen zijn wegens bijvoorbeeld een tijdelijk technisch probleem. Voor deze optie is er een knop op de webinterface die de gebruiker naar het BOM-scherm leidt;

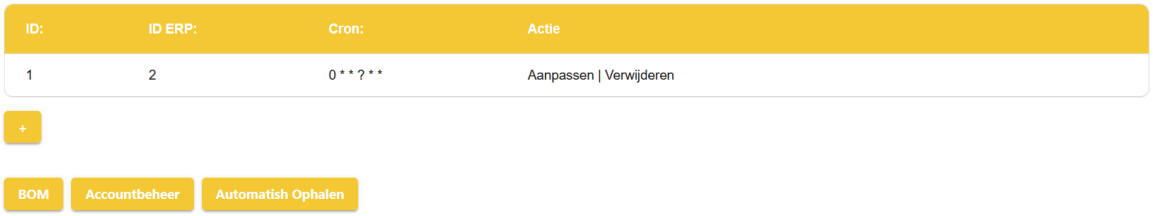


Figuur 13; CoreXR Webinterface BOM scherm

Op dit scherm is ook te zien welke BOM er op het moment opgeslagen is, dit is gedaan zodat gebruikers kunnen zien of de BOM recent is. Als de verbinding tussen CoreXR en het ERP-systeem goed werkt zou deze datum recent moeten zijn. De BOM wordt automatisch opgevraagd als het scherm wordt geopend. Er is een knop om een BOM handmatig te uploaden links onderin het scherm, deze opent een scherm waarmee de gebruiker een bestand kan uploaden.
Als deze een geldig bestandstype heeft dan wordt deze naar CoreXR gestuurd en eventueel geconverteerd voordat die wordt opgeslagen.

Implementatie **REQ-C-016**:
"Het systeem met in staat zijn de database automatisch te laten bijwerken door regelmatig een nieuwe BOM op te halen."
Aan deze requirement is voldaan door op de database automatische taken op te slaan in de tabel ERPBOMCron. Per taak wordt opgeslagen voor welk ERP-systeem er een BOM moet worden opgehaald, en een cron-expressie waaruit te lezen is wanneer deze uitgevoerd moet worden. Deze cron-expressie is opgeslagen als een varchar op de interface. Deze wordt in de java-applicatie opgehaald als een string. Als de applicatie wordt opgestart worden alle crons van de database opgehaald en in een TaskScheduler gezet. Dit is een interface van String dat taken uit kan voeren op tijdstippen aangegeven door cron-expressies. Er moet ook worden meegegeven welke taak dat is. Bij CoreXR is dat het ophalen van een BOM en deze in op te slaan op de database. Er staat op de database in de tabel ERPBOMCron naast de cron-expressie ook het id van het ERP-systeem waarvoor op het ingestelde tijdstip een BOM opgehaald moet worden. Deze wordt gebruikt als parameter voor aanroepen van de functie waarmee een nieuwe BOM wordt opgehaald. De TaskScheduler krijgt dus een cron-expressie en wordt verteld "Op de tijd aangegeven door de cron-expressie, voer de functie uit van het ophalen van een BOM met deze id als parameter.".

Het instellen van deze taken wordt gedaan met een webinterface.
Net als met gebruikers is er een tabel met taken te vinden, deze werkt technisch op dezelfde manier. Net als gebruikers kunnen de taken verwijderd of aangepast worden, en is er de optie om nieuwe toe te voegen. Er is gekozen dit op dezelfde manier als accountbeheer te implementeren om het beheer centraal te houden op de webinterface.
Als er met de webinterface een taak wordt aangemaakt of aangepast wordt deze ook automatisch op de database gezet en in de TaskScheduler.



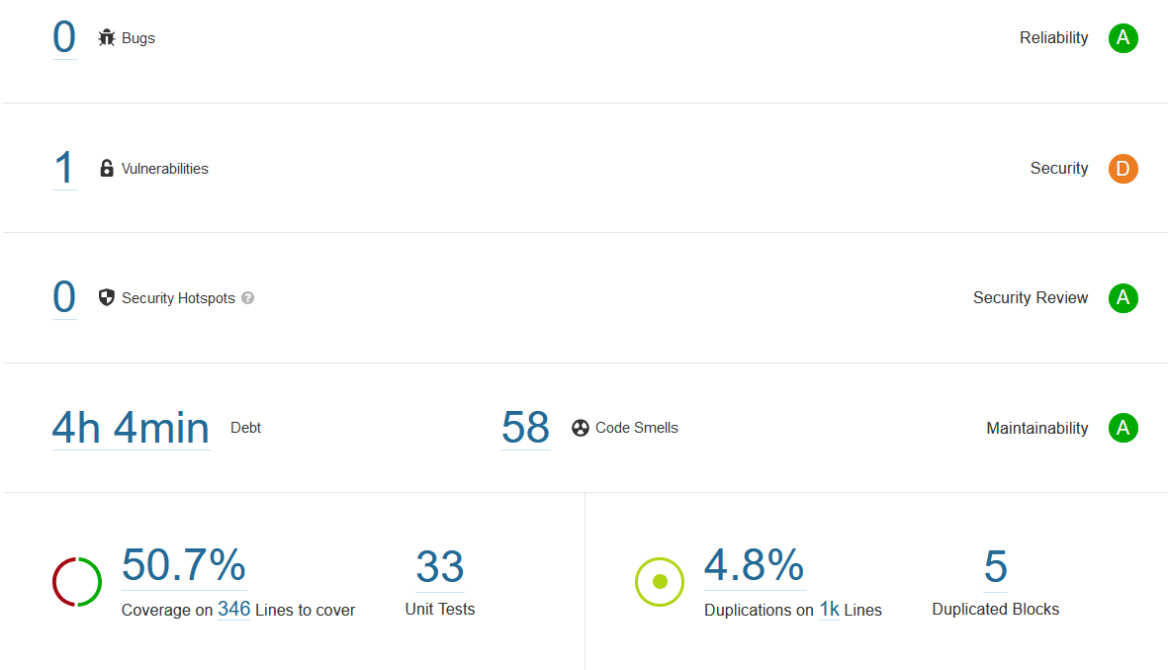
Figuur 14; Tabel ingeplande taken

Implementatie **REQ-C-017**:
"Het systeem ondersteunt accounts die bevoegd zijn voor meerdere ERP-systemen."
Deze requirement is net als requirement S3 behaald door de many-to-many relatie tussen ERP-systemen en gebruikers zoals beschreven in requirement M6.

8.1 Code kwaliteit

Voor de waarborging van code kwaliteit is er gebruik gemaakt van JUnit tests, Jenkins en SonarQube. Jenkins was zo ingesteld dat de tests automatisch werden uitgevoerd als het project naar GitLab werd gepusht. Als niet alle tests goed verliepen werd er ook automatisch een email bericht verstuurd om mij daarop te wijzen.

Naast de tests werd er ook automatisch de code kwaliteit beoordeeld door SonarQube, deze maakte hier een rapport van.



Figuur 15; SonarQube rapport code kwaliteit.

SonarQube gaf in dit rapport een Reliability score van "A", een Maintainablity score van "A", een Security score van "D", en een Security Review score van "A".

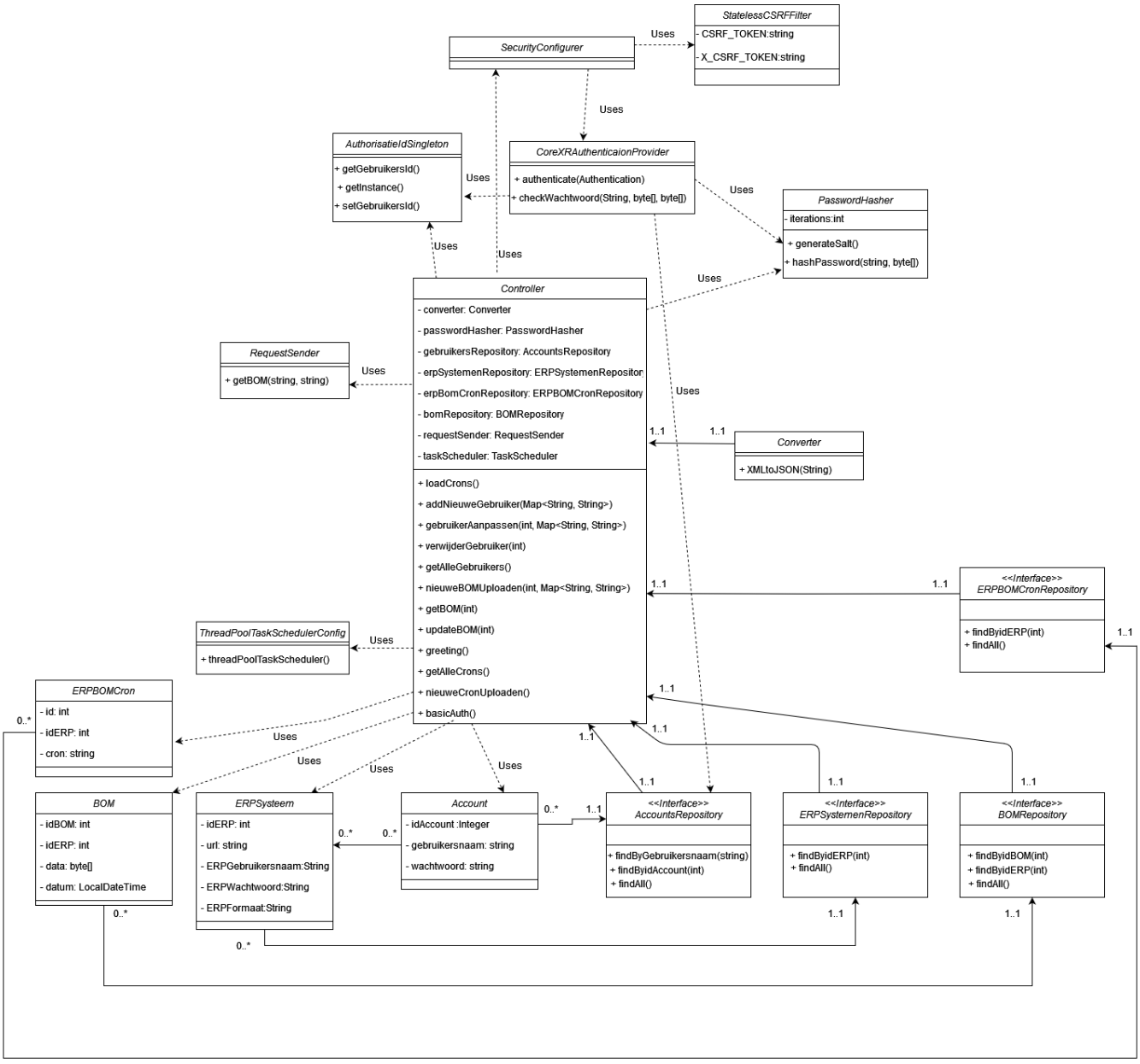
De Security score is laag omdat de RequestSender class van CoreXR met basic authentication werkt. SonarCube raadt aan om een veiligere methode te gebruiken. Er was gepland om OAuth2

te gebruiken, maar dat was uiteindelijk niet mogelijk omdat de rechten daarvoor niet beschikbaar waren. Of het gewenst is om de optie voor basic authentication weg te halen is echter discutabel. Het doel van het project is om met een generiek platform met zo veel mogelijk ERP-systemen te kunnen verbinden. Als er aan deze verbinding strenge security eisen worden gesteld dan kan er niet met ERP-systemen worden verbonden die hier niet aan voldoen.

CoreXR heeft op het moment van schrijven 50,7% code coverage en 4,8% gedupliceerde code. Het code coverage percentage word nog verwacht te stijgen omdat er na het schrijven van dit verslag nog aan gewerkt wordt.

8.2 Class Diagrammen

Deze diagrammen zijn ook terug te vinden als losse afbeeldingen in het afstudeerdossier.



Figuur 16; Class Diagram CoreXR

8.2.1 Classes CoreXR Service

8.2.1.1 Controller

In figuur 15 is te zien dat de class “Controller” het hart is van de applicatie. In deze class komen de API-requests binnen en wordt er verwezen naar andere classes om deze af te handelen.

8.2.1.2 Repositories

De classes AccountsRepository, ERPSystemenRepository, BOMRepository en ERPBOMCronRepository zijn repositories. Deze classes worden door Spring gevuld met de data die op de database staat. Door deze classes aan te roepen kan de applicatie de database gebruiken. Ook wordt nieuwe data in de resposities aangepast op de database. Deze repositories worden vooral gebruikt door de Controller, maar de AccountsRepository ook voor security door CoreXRAuthenticationProvider. Deze repositores leveren respectievelijk de classes Account, ERPSysteem, BOM en ERPBomCron.

8.2.1.3 Converter

De class Converter wordt aangeroepen door Controller als er een BOM naar JSON omgezet moet worden. Op het moment bevat deze alleen een functie om XML naar JSON om te zetten, maar de class is zeer makkelijk uit te breiden.

8.2.1.4 ThreadPoolTaskScedulerConfig

In deze class wordt ingesteld hoeveel ingeplande taken er tegelijk uitgevoerd kunnen worden. Het staat op het moment van schrijven ingesteld op 50. Mocht er in de toekomst blijken dat bij druk gebruik er teveel ingeplande taken tegelijk uitgevoerd moeten worden dan kan deze waarde hoger worden ingesteld.

8.2.1.5 RequestSender

De class RequestSender handelt het sturen van API-calls af. Deze class wordt aangeroepen door de class Controller als er een BOM opgehaald moet worden van een ERP-systeem. Er wordt door de Controller de functie getBOMBasicAuth aangeroepen. Hiervoor wordt een URL en inloggegevens mee gegeven. RequestSender maakt hier een API-request van, voert deze uit en geeft het resultaat weer terug aan Controller.

8.2.1.6 CoreXRAuthenticationProvider

Deze class overschrijft de standaard Authentication class van Spring en maakt gebruik van de AccountsRepository class en de PasswordHasherClass. Deze class is overschreven omdat CoreXR niet standaard Spring accounts gebruikt. CoreXR accounts moeten namelijk naast geldige inloggegevens ook bevoegd zijn voor een ERP-systeem om hiervan gegevens op te vragen. De standaard authenticatie ondersteunt deze complexiteit niet. Aangezien wachtwoorden geëncrypteert op worden geslagen haalt deze class ook de salt op van een account dat in wil loggen. Per account wordt namelijk bewaard welke salt er is gebruikt om het wachtwoord te encrypteren. Als er moet worden ingelogd dan wordt het ingevulde wachtwoord hier geencrypt met dezelfde waarde en vergeleken met het opgeslagen geencrypte wachtwoord. Dit wordt op deze manier gedaan omdat er nergens wachtwoorden zijn opgeslagen die niet geëncrypteert zijn. Een wachtwoord kan daarom niet worden vergeleken zonder deze eerst op dezelfde manier te encrypteren.

8.2.1.7 PasswordHasher

Deze class wordt gebruikt om wachtwoorden te encrypten en om salts te genereren. Een salt wordt simpelweg gemaakt door een array van 16 bytes te vullen met willekeurige waarden. Deze salt wordt gebruikt bij het encrypteren van een wachtwoord. Deze salt is elke keer anders zodat er niet in de database te zien valt of een wachtwoord meerdere keren gebruikt word. Als een wachtwoord met dezelfde tekst en dezelfde salt geëncrypteert wordt resulteert dit namelijk in dezelfde geencrypte waarde. Een wachtwoord met dezelfde tekst en een andere salt niet. Wachtwoorden met dezelfde tekst zijn dus niet terug te zien in de database. (Tenzij een wachtwoord met dezelfde tekst toevallig precies dezelfde salt krijgt, maar die kans is 1 op 65.536) Ook beschermt de salt tegen het gebruik van rainbow tables. De wachtwoorden zijn geëncrypteert met PBKDF2, zie Implementatie REQ-M-006 voor meer informatie hierover.

8.2.1.8 AuthorisatieSingleton

Er wordt op het moment een singleton gebruikt om te zien welke gebruiker zich geautoriseerd heeft om zo ook te kijken of die bevoegd is voor een specifiek ERP-systeem. Als een gebruiker via de CoreXRAuthenticationProvider wordt geauthenticeerd wordt er in de singleton bijgehouden welke gebruiker dit was. In de Controller wordt dan gekeken welke gebruiker het laatst zich geauthenticeerd heeft, en kijk dan of die bevoegd is voor de API-request die deze probeert te doen. Dit is echter niet ideaal, als er twee gebruikers tegelijk een API-request proberen te doen zou het kunnen dat gegevens in de singleton worden overschreven voordat er in de Controller gekeken wordt in de singleton, en er op die manier naar de verkeerde gegevens wordt gekeken.

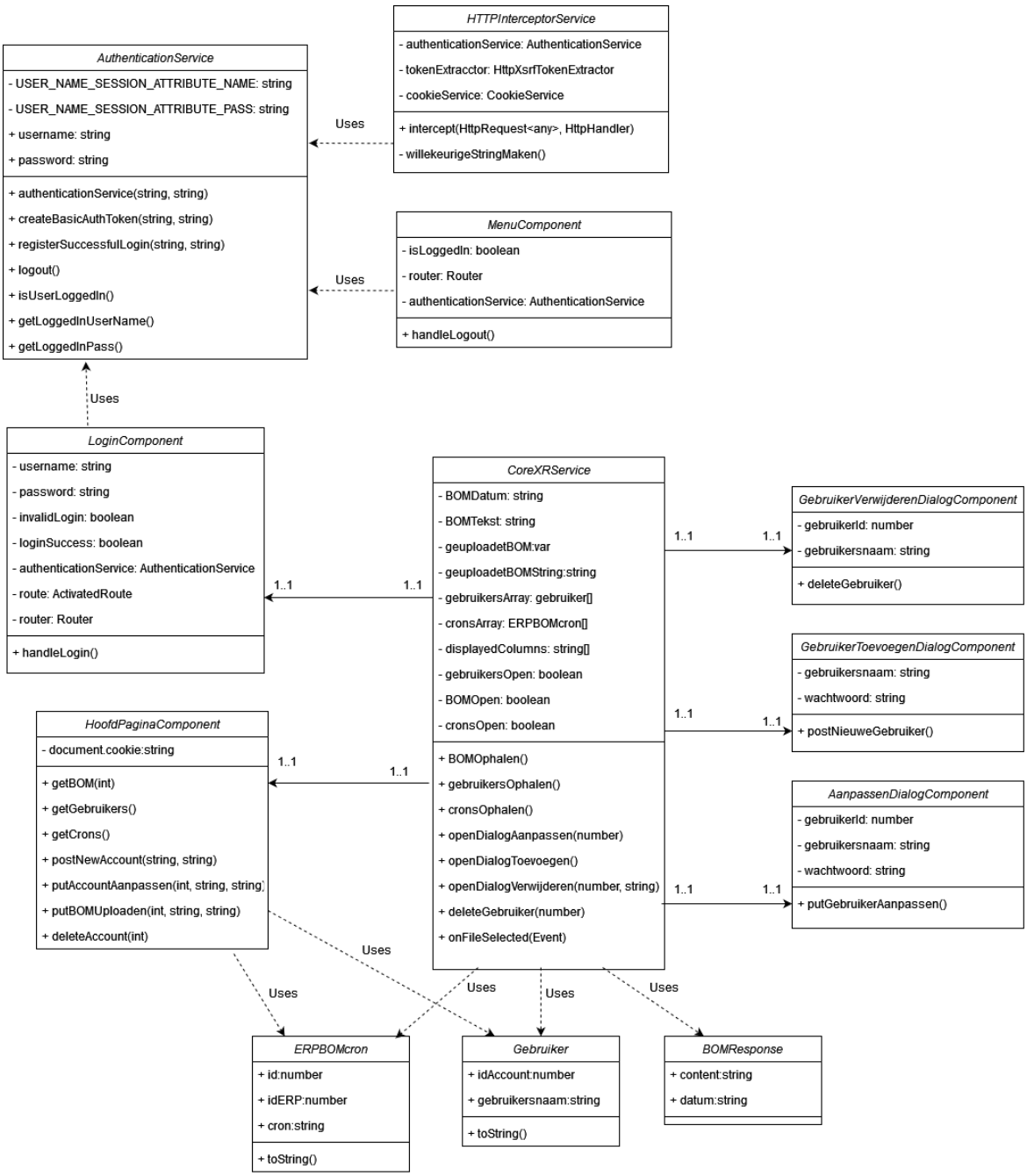
8.2.1.9 SecurityConfigurer

Dit is een configuratie class van Spring, hierin worden allerlei dingen ingesteld qua beveiliging. Zo wordt hier ingesteld dat de standaard Spring authenticatie wordt overschreven door de class CoreXRAuthenticationProvider. Ook wordt hier CORS en CSRF ingesteld. Tijdens het ontwikkelen van de webinterface bleek dat communicatie met CoreXR vaak niet goed werkte wegens CORS instellingen. De CORS instellingen in deze class verhelpen dit. Ook wordt er in deze class een CSRF filter ingesteld. Dit wordt gedaan om CSRF-aanvallen te voorkomen. Er moet nu een geldige CSRF-token door de webinterface worden meegegeven in de header van een API-request om met CoreXR te communiceren.

8.2.1.10StatelessCSRFFilter

In deze class wordt het CSRF-filter ingesteld. Hier wordt ook nagekeken of een CSRF-token geldig is. Zo niet, dan wordt er een exception gegooid en zal de API-request falen.

8.2.2 Classes CoreXR Webinterface



Figuur 17; Class diagram Webinterface

8.2.3 CoreXRService

Deze class is een verzameling van functies die API-requests naar de CoreXR service sturen en de opgehaalde data terug geven. Dit word in andere classes gebruikt om bijvoorbeeld een lijst met gebruikers op te halen.

8.2.4 ERPBOMCron

Deze class wordt gebruikt om de gegevens van een cron-taak in op te slaan. Een id, een id van een ERP-systeem en de cron-expressie worden opgeslagen.

8.2.5 Gebruiker

Deze class wordt gebruikt om de gegevens van een gebruiker in op te slaan. Een id en een gebruikersnaam worden opgeslagen.

8.2.6 BOMResponse

Deze class wordt gebruikt om de gegevens van een BOMResponse in op te slaan. Deze wordt gevuld door de JSON in te lezen van de API-response. Hieruit wordt de datum en de content gehaald.

8.2.7 LoginComponent

Deze class verzorgt de scripting voor de login-pagina. De ingevulde login-gegevens worden doorgegeven aan de AuthenticationService, en afhankelijk van het antwoord wordt de gebruiker naar de hoofdpagina doorgestuurd of niet.

8.2.8 HoofdPaginaComponent

In deze class wordt de scripting van de hoofdpagina geregeld. Naast het tonen van de juiste data in een tabel wordt ook het openen van dialog vensters hier gedaan.

8.2.9 Dialog Components

Deze classes kunnen vanuit de HoofdPaginaComponent worden geopend. Deze vensters worden gebruikt om onder andere gegevens van een nieuwe gebruiker in te vullen. Vanuit deze vensters worden er ook de CoreXRService functies aangeroepen om bepaalde API-requests uit te voeren. Dit kan bijvoorbeeld het aanmaken van een nieuwe gebruiker zijn.

8.2.10 AuthenticationService

In deze functie worden de ingevulde login-gegevens bijgehouden zolang als de gebruiker is ingelogd. Als de gebruiker wordt ingelogd worden de gegevens hier naartoe gestuurd. Met de HTTPInterceptor class worden ze automatisch altijd meegegeven als de webinterface een API-call stuurt. Ook kan deze class een API-request naar CoreXR sturen om te kijken of inloggegevens geldig zijn. Dit wordt gebruikt bij het inloggen.

8.2.11 MenuComponent

Deze class regelt de functionaliteiten van de menubalk bovenin het hoofdscherm. Er zit een knop op deze balk om uit te loggen, en deze class roept de AuthenticationService aan om dat te doen.

8.2.12 HTTPInterceptorService

In deze class wordt elke API-request die de webinterface stuurt onderschept. Van de AuthenticationService worden de inloggegevens opgehaald. Hier wordt een token van gemaakt en in de header van het onderschepte API-request gestopt. Ook wordt hier een CSRF-token toegevoegd.

9 Conclusies

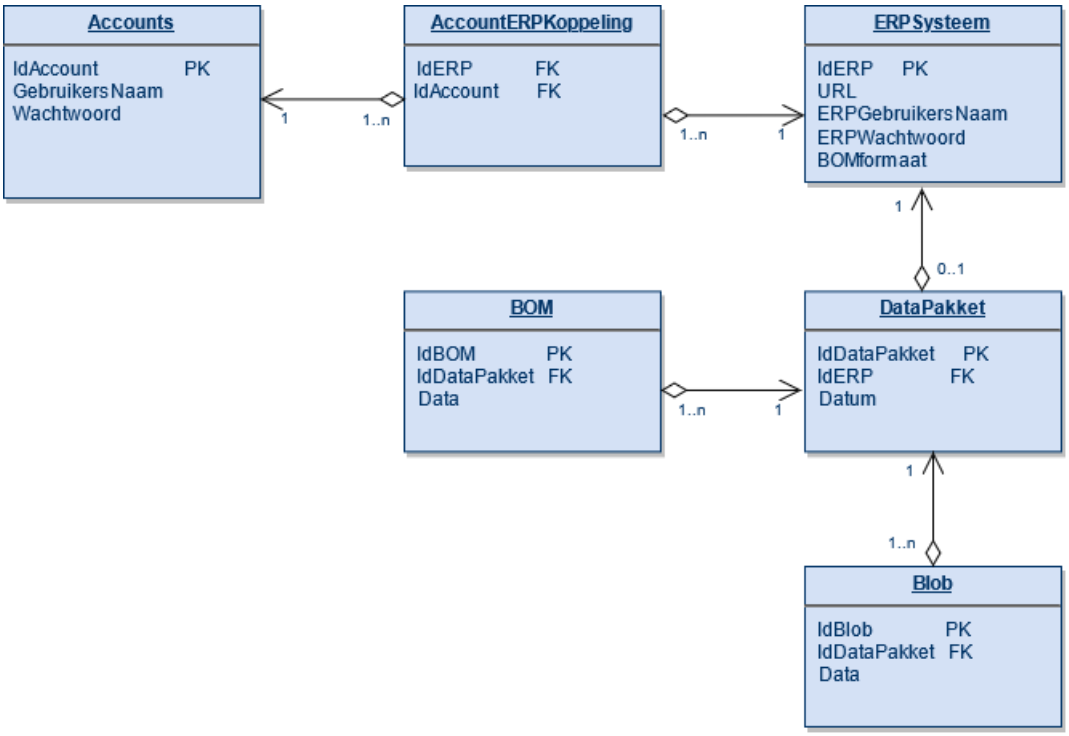
Het is gelukt om alle requirements te behalen voor het project. Wel is requirement REQ-M-006 wat discutabel en zou de webinterface nog verbeterd kunnen worden. De beveiliging van REQ-M-006 is namelijk wegens de gebruikte oplossing met de singleton niet helemaal betrouwbaar en zou nog verbeterd kunnen worden.

Naast de requirements is het ook gelukt om de optie te maken waarbij de gebruiker een BOM op kan vragen met een extra selectie over welke data er geleverd wordt. Deze functie is beschikbaar in de API en zou eventueel in de toekomst nog gebruikt kunnen worden.

Voor het waarborgen van de codekwaliteit is gebruik gemaakt van SonarQube.
In een rapport van SonarQube is te zien dat in de meeste aspecten de code van goede kwaliteit is. De test coverage is echter op het moment van schrijven aan de lage kant met 50.7%.
Ook geeft SonarQube een lage Security score aangezien de RequestSender class met basic authentication werkt, en niet met een veiligere methode.

10 Aanbevelingen

Er is te overwegen de database uit te breiden om ook andere dingen dan een BOM te ondersteunen. In een eerdere versie van het ontwerp stond een andere database diagram waar ook eventueel andere dingen dan alleen BOMs in opgeslagen kunnen worden, maar er werd me na het laten zien van het ontwerp aan Total Reality verteld dat ze alleen geïnteresseerd waren in BOMs omdat ze voor andere data andere software gebruiken. Daarom is de database diagram aangepast om alleen een BOM op te kunnen slaan. Mocht er later toch aan gedacht worden om CoreXR voor andere dingen dan alleen BOMs te gebruiken dan zou de database nog aangepast kunnen worden en is de oude database diagram mogelijk interessant om er bij te houden.



Figuur 18; De oude database diagram. Bij deze is er ook een "blob" tabel en een "DataPakket" tabel. Het idee hierachter is dat een ERP niet alleen een BOM heeft, maar een pakket waar een BOM en eventueel ander soorten data inzitten.

Vanuit Total Reality was het nog niet bekend hoeveel API calls er naar een CoreXR installatie gedaan worden, maar het is mogelijk om een maximum aantal gelijktijdige API-requests in te stellen om te voorkomen dat CoreXR overloaded wordt, mocht dit een probleem worden. Dit kan ingesteld worden door *server.tomcat.max-threads* aan *application.properties* toe te voegen.

CoreXR zoals het nu is kan alleen met API-calls een ERP-systeem benaderen om een BOM op te halen. Het zou interessant kunnen zijn om de mogelijkheid te geven aan CoreXR om ook een BOM direct van een database op te kunnen halen, dit zou CoreXR in nog meer gevallen inzetbaar maken. Er zou hiervoor een extra component aangemaakt kunnen worden die met deze databases verbindt, deze zou dan vergelijkbaar zijn met het request-sender component. Per ERP-systeem zou er kunnen worden opgeslagen of er van een database of een API gegevens moeten worden opgehaald, net zoals er nu wordt bijgehouden of er geconverteerd moet worden.

Er zou ook gekeken kunnen worden of het inloggen op CoreXR eventueel te verbinden zou zijn met bestaande login-systemen in een bedrijf. Er werd al gebruik gemaakt van Azure Active Directories voor een verbinding tussen CoreXR en Dynamics. Misschien dat de authenticatie van de VR-app naar CoreXR ook gebruik hiervan zou kunnen maken zodat een gebruiker kan inloggen met het account dat ze ook gebruiken voor Microsoft applicaties zoals Office.

De *converter* zou uitgebreid kunnen worden met meer conversie opties. Er kan voor elke optie een extra public functie aangemaakt worden die vervolgens kan worden aangeroepen wanneer dat nodig is.

Mochten er in de toekomst eisen komen aan de JSON die door CoreXR wordt geleverd dan zou er overwogen kunnen worden om een extra component toe te voegen die de JSON kan aanpassen aan bepaalde wensen. Dit zou vergelijkbaar werken als het converter component. Er zou bijvoorbeeld een bepaalde eis kunnen zijn om alleen een bepaald element van een JSON bestand op te halen. Dit zou net zo kunnen werken als bij de converter dat er per ERP-systeem wordt bijgehouden wat hier de bepaalde eisen voor zijn, of alternatief, de API zou kunnen worden uitgebreid dat deze wensen in de request kunnen worden gezet. Het nadeel met het gebruik van de database hiervoor is dat er dan van uit wordt gegaan dat deze eisen constant hetzelfde zijn. Een nadeel van dit extra component is dat dit voor meer maatwerk en meer complexiteit zou kunnen zorgen.

Er wordt op het moment een singleton gebruikt om te zien welke gebruiker zich geautoriseerd heeft om zo ook te kijken of die bevoegd is voor een specifiek ERP-systeem. Dit werkt in het prototype, maar als het veel gebruikers tegelijkertijd heeft zou deze in de war kunnen raken, het is een goed idee om hier een betere oplossing voor te vinden, deze singleton is bedoeld als een tijdelijke tussenstop en is niet betrouwbaar.

Er kan ook overwogen worden om niet alleen als JSON te kunnen exporteren maar ook als andere veel voorkomende bestandstypen zoals CSV of XML. Dit kan handig zijn in het geval dat er in de toekomst wensen bijkomen voor meer export mogelijkheden dan alleen JSON. Om dit te realiseren zou er voor gekozen kunnen worden om het gewilde bestandstype als parameter mee te geven in de API-call naar CoreXR toe, die daar vervolgens rekening mee houd door de response te converteren.

Een mogelijke verbetering bij het opslaan van BOM's zou het gebruik van een cache kunnen zijn. Als de meest recent opgeslagen BOM om in een cache staat zou er bij het ophalen van een nieuwe BOM gekeken kunnen worden of deze hetzelfde is als de opgeslagen BOM. Zo ja, dan hoeft er niet naar de database geschreven te worden. Er zal wat geëxperimenteerd moeten worden om te kijken of het vergelijken van BOM's tijd bespaard vergeleken met het direct opslaan.

11 Bronnen

1
Steen, M (2021) Onderzoeksverslag CoreXR Generiek Platform

2
Moriarty, Kaliksi, Rusch, K. B. A. (2017, januari). *PKCS #5: Password-Based Cryptography Specification Version 2.1*. ietf.org. https://datatracker.ietf.org/doc/html/rfc8018

3
Columbus, L. (2020, 27 oktober). *The Most Popular ERP Systems Of 2020 Based On Customer Feedback*. Forbes. https://www.forbes.com/sites/louiscolumbus/2020/10/27/which-erp-systems-are-most-popular-with-their-users-in-2020/?sh=18eb169148ff

4
Pang, Markovski, Andrej, A. M. A. (2020, 5 november). *Top 10 ERP Software Vendors, Market Size and Market Forecast 2019–2024*. appsruntheworld. https://www.appsruntheworld.com/top-10-erp-software-vendors-and-market-forecast/

5
Liu, S. (2018, 10 oktober). *Breakdown of the ERP software market worldwide 2016, by vendor*. Statista. https://www.statista.com/statistics/558784/worldwide-erp-market-share-distribution-by-vendor/

6
Gartner. (2017). *ERP Systems (world market)*. TADviser. https://tadviser.com/index.php/Article:ERP_systems_(world_market)

7
Santos, W. (2020, 3 april). JSON is Clearly the King of API Formats in 2020. *ProgrammableWeb.* https://www.programmableweb.com/news/json-clearly-king-api-data-formats-2020/research/2020/04/03

8
(2000) BILL OF MATERIALS, INDENTED. In: Swamidass P.M. (eds) Encyclopedia of Production and Manufacturing Management. Springer, Boston, MA . https://doi.org/10.1007/1-4020-0612-8_92

9
Turnhout, K. (2013). *DOT-framework*. wikiwijs. https://maken.wikiwijs.nl/127721/DOT_framework

12 Bijlagen

12.1 Interview Robin Kuiper, Total Reality contactpersoon

Is er al iets bekend van Hoppmann en Trimergo?

Over 10 uur, dus over een half uurtje, heb ik een meeting met Hoppmann via teams. Het ging een beetje traag, maar nu hebben we een model van Hoppmann. Dat heb ik in ons platform gezet en nu gaan we dat bespreken in een half uurtje en mijn plan is ook om dan direct een beetje meer druk te zetten op het Trimergo gedeelte of ze daar wat data van konden sturen.

Ik wou ook graag ook vragen of er naast de Bill of Materials nog andere data is die Total Reality graag zou krijgen.

Nou ja, dat verschilt nogal per klant. Met de Bill of Materials bedoel je gewoon de onderdeelnummers?

Ja, onderdeelnummers, namen van onderdelen, lijst wat er in zo een apparaat zit. Ik ging er van uit dat deze regelmatig terug zou komen

Ja, dat is wel de kern denk ik ook inderdaad, daar heb je wel gelijk aan. Het verschilt een beetje per klant, bij de ene klant zit daar denk ik ook prijsdata aan/ ook hoeveel een onderdeel kost aan en bij de andere klant zit daar meer technische data aan. Ik denk dat dat een beetje verschilt. Bij een Bill of Materials, valt door voor jou ook al prijsdata onder of niet?

Dat weet ik eigenlijk niet, ik denk dat nogal verschilt per situatie.

Ja, dat verschilt heel erg, ik denk per software en per klant. Maar goed, wat je zegt, die Bill of materials is waarschijnlijk de kern, zal bij elke klant verschillend zijn. Is waarschijnlijk een lijst met filiaal data. Per klant zal het verschillen wat dat precies inhoud. Bij de ene klant zul je ordernummers kijken, pakketnummers misschien zelfs, maar bij een andere klant zal dat misschien helemaal niet van belang zijn. Dus dat loopt wel een beetje uit een denk ik.

Zijn er op het moment bepaalde oplossingen die al bestaan? Misschien dat er al bepaalde dingen zijn die herbruikbaar zijn.

Zoveel kennis hebben we er niet van specifiek, maar wat ik wel weet is dat er vaak Microsoft Dynamics wordt gebruikt om verschillende koppelingen met elkaar te maken. Maar iets bestaands is ook eigenlijk wat we van jou moeten horen, maar van wat wij zo ver weten niet. Ik heb wel een bepaalde meeting gehad met een bedrijf die voor Microsoft Dynamics dingen ontwikkeld en die heeft wel een soort tussenlaag, maar die is bedoeld voor alles wat met Microsoft Dynamics te maken heeft, dat is ook heel veel. Maar, dan sluit je wel in die zin programma's zoals Trimergo uit. Maar zij waren dan ook Microsoft partner en volledig op Microsoft gefocust, dus voor hun was dat niet zo erg. Zij hadden wel zelf een soort custom laag ontwikkeld daar in. Maar daar weet ik het fijne niet van.

12.2 API Overzicht

De volgende API requests zijn op het moment mogelijk:

GET /bom (http://localhost:8080/bom?idERP=1)
Params: idERP
Header:

```
GET /bom HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: Geen
Authorization: Basic Auth
Response: JSON met BOM

GET /selectbom (http://localhost:8080/selectbom?idERP=1)
Params: idERP
Header:

```
GET /bom HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: {“jsonpath”:”* 0 0 1 0 0”}
Authorization: Basic Auth
Response: JSON met BOM, met selectie

PUT /bom?idERP=1 (http://localhost:8080/bom?idERP=1)
Params: idERP
Header:

```
GET /bom HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: : {bestandsType : “(JSON?)”, BOM: (de gegevens)}
Authorization: Basic Auth
Response: “Sucess”

GET /users (http://localhost:8080/users)
Params: Geen
Body: Geen
Header:

```
GET /users HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Response status: 200 OK
Response body:

```
[
    {
        "idAccount": 1,
        "gebruikersnaam": "david"
    }
]
```

POST /user (http://localhost:8080/user)
Params: Geen

```
POST /user HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: {gebruikersnaam : ”jantje”, wachtwoord: “wachtwoordje”}
Authorization: Basic Auth
Response status: 200 OK
Reponse body success: “Saved”

PUT /user (http://localhost:8080/user?id=1)
Params: id

```
POST /user HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: {gebruikersnaam : ”jantje”, wachtwoord: “wachtwoordje”}
Authorization: Basic Auth
Response status: 200 OK
Reponse body success: “Aangepast”
Reponse body foutmelding?

DELETE /user (http://localhost:8080/user?id=1)
Params: id

```
POST /user HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: Geen
Authorization: Basic Auth
Response status: 200 OK
Reponse body: “Verwijderd”

GET /erps (http://localhost:8080/erps)
Params: Geen
Body: Geen
Header:

```
GET /erps HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Response status: 200 OK
Response body:

```
[
    {
        "idERP": 1,
        "url": "https://api.businesscentral.dynamics.com/v2.0/51e2536e-ef57-
4fa7-a2e1-8c229612e8a4/Sandbox/WS/CRONUS%20NL/Codeunit/CompanySetupService",
        "erpgebruikersNaam": "MICHAEL.STEEN",
        "bomformaat": "XML"
    },
    …
]
```

POST /erp (http://localhost:8080/erp)
Params: Geen

```
POST /user HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: {“url”:”https://api.erpsys.com/bom”, bomformaat : ”JSON”, gebruikersnaam: “bob”, wachtwoord: “wachtwoordje”}
Authorization: Basic Auth
Response status: 200 OK
Reponse body success: “Saved”

PUT /erp (http://localhost:8080/erp?id=1)
Params: id

```
POST /user HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: {“url”:”https://api.erpsys.com/bom”, bomformaat : ”JSON”, gebruikersnaam: “bob”, wachtwoord: “wachtwoordje”}
Authorization: Basic Auth
Response status: 200 OK
Reponse body success: “Aangepast”

Reponse body foutmelding?

DELETE /erp (http://localhost:8080/erp?id=1)
Params: id

```
POST /user HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: Geen
Authorization: Basic Auth
Response status: 200 OK
Reponse body: "Verwijderd"

GET /crons (http://localhost:8080/crons)
Params: Geen
Body: Geen
Header:

```
GET /users HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Response status: 200 OK
Response body:

```
[
    {
        "id": 1,
        "idERP": 2,
        "cron": "0 * * ? * *"
    }
]
```

POST /cron (http://localhost:8080/cron)

```
POST /user HTTP/1.1
Host: localhost:8080
Authorization: Basic dXNlcjE6dXNlcjFwYXNz
```

Body: {idERP : 2, cron: "2 * * ? * *"}
Authorization: Basic Auth
Response status: 200 OK
Reponse body: "Opgeslagen"

12.3 Planning

Week 1
- 7 februari - 13 februari

Week 2
- 14 februari - 20 februari
- Inleveren afstudeerovereenkomst

Week 3
- 21 februari - 27 februari
- Inleveren concept plan van aanpak

Week 4
- 28 februari – 6 maart
- Bespreken plan van aanpak met afstudeerbegeleider en bedrijfsbegeleider
- Inleveren maandelijkse tussentijdse versie van afstudeerverslag

Week 5
- 7 maart - 13 maart
- Inleveren definitief plan van aanpak
- Begin afstudeer opdracht door start onderzoek

Week 6
- 14 maart – 20 maart

Week 7
- 21 maart - 27 maart

OVSoftware

Week 8
- 28 maart - 3 april
- Inleveren maandelijkse tussentijdse versie van afstudeerverslag

Week 9
- 4 april - 10 april
- Afronden onderzoek
- Beginnen met maken van ontwerp

Week 10
- 11 april - 17 april

Week 11
- 18 april – 24 april
- Beginnen met implementatie

Week 12
- 25 april - 1 mei
- Inleveren maandelijkse tussentijdse versie van afstudeerverslag

Week 13
- 2 mei - 8 mei

Week 14
- 9 mei - 15 mei

OVSoftware

Week 15
- 16 mei - 22 mei

Week 16
- 23 mei - 29 mei

Week 17
- 30 mei - 5 juni
- Inleveren maandelijkse tussentijdse versie van afstudeerverslag
- Concept formulier Eindbeoordeling Afstuderen HBO-ICT (31 mei)
- Concept afstudeer dossier (31 mei)

Week 18
- 6 juni - 12 juni
- Feedback documentatie gebruiken voor verbeteringen

Week 19
- 13 juni - 19 juni
- Formulier Eindbeoordeling Afstuderen HBO-ICT (14 juni)
- Afstudeer dossier (14 juni)
- Reflectie
- Evaluatieformulier afstuderen invullen

Week 20
- 20 juni - 26 juni
- Eindpresentatie inlevern

Week 21
- 27 juni - juli
- Inleveren abstract van afstudeerproject voor afstudeerboek