AFSTUDEERVERSLAG

3D IFC-MODEL VIEWER

SANDER ADAM | 452695

V1.11

12 juni 2022

Saxion | ConstruSteel

Voorwoord

Tijdens het oriënteren op verschillende bedrijven voor een afstudeeropdracht kwam de directeur van Construsteel, Jos Bonte, op een halfjaarlijkse contactlenscontrole bij mijn vader. Ze kwamen in gesprek en Jos heef zijn visitekaartje achtergelaten. Met de vraag of ik even wilde bellen, zodat er mogelijkheden voor werk en/of afstuderen besproken kon worden.

Dat heb ik gedaan, en ter plekke aan de telefoon werd ik uitgenodigd voor een sollicitatiegesprek. Na het tweede gesprek, waar ik toonde wat ik zoal in het verleden heb gemaakt aan projecten, werd er in overleg een opdracht bepaald.

Op 7 februari ben ik gestart bij ConstruSteel als afstudeerder. Pascal Brinkers en André Romijn namen de begeleiding op zich. Later in het afstudeerproces kwam Heini Veneberg daar nog bij. Ze hebben allemaal goede begeleiding gegeven en ik heb veel van ze geleerd. Heini en Pascal deden de projectbegeleiding, en André ging over het product en wat er ontwikkeld moest worden. Doordat er een product owner bij zat, kon er in de ontwerp en designfase goed overlegd worden over design keuzes.

Vanuit Saxion ben ik begeleid door Eelco Jannink. Eelco gaf goede begeleiding en zei altijd waar het op stond. Met de feedback van Eelco Jannink kon ik veel. Het was duidelijk en altijd ‘to the point’. Daarvoor wil ik hem bedanken. Deze directe feedback heeft me goed geholpen bij het afstudeertraject.

Naast alle begeleiders die ieder mijn dank krijgen, wil ik ook de steun vanuit de persoonlijke kring bedanken. De steun vanuit mijn ouders, familie en vrienden heeft me zeer goed gedaan. Zonder deze hulp had ik de afstudeerperiode waarschijnlijk anders ervaren.

Ik wens u veel leesplezier,

Sander Adam

Versiebeheer

Versie	Datum	Omschrijving	Auteur
1.0	5 mei 2022	Initiële opzet van het verslag	Sander Adam
1.1	6 mei 2022	Inleiding aangevuld	Sander Adam
1.2	8 mei 2022	Onderzoekfase en designfase aangevuld	Sander Adam
1.3	10 mei 2022	Feedback Eelco Jannink toegepast	Sander Adam
1.4	17 mei 2022	Aanvulling verslag nadat feedback Eelco Jannink toegepast is	Sander Adam
1.5	23-05-2022	Feedback bedrijfsbegeleiders toegepast.	Sander Adam
1.6	23-05-2022	Fouten van versie v1.5 gecorrigeerd	Sander Adam
1.7	23-05-2022	Verwoording van hoofdstuk 8 aangepast	Sander Adam
1.8	26-05-2022	Feedback Heini Veneberg en Eelco Jannink toegepast	Sander Adam
1.9	6 juni 2022	Feedback Dick Heijink toegepast	Sander Adam
1.10	8 juni 2022	Feedback Eelco Jannink toegepast	Sander Adam
1.11	11 juni 2022	Feedback Heini Veneberg toegepast	Sander Adam

Inhoudsopgave

Voorwoord 1
Versiebeheer 2
1. Begrippenlijst 7
2. Inleiding 8
2.1 Het bedrijf in context 8
2.2 De huidige situatie 8
2.2.1 ConstruSteel On Premise 8
2.2.2 De Digitale Werkplaats 8
2.3 Het probleem dat opgelost moet worden 9
2.4 De doelstelling van de opdracht 9
2.5 Deliverables 9
3. Aanpak project 11
3.1 Kanban 11
3.2 Overgang SCRUM-achtige werkwijze 11
4. Contextanalyse 12
4.1 De doelgroep 12
4.2 Klassieke werkwijze staalbouwers 12
5. Probleemanalyse 12
5.1 De huidige oplossing 12
5.2 Hoofd- en deelvragen 14
6. De onderzoeksfase 17
6.1 Bestaande IFC-viewers 17
6.2 Onderzoek naar IFC-bestand formaten 17
6.3 Welke techniek kan het best gebruikt worden? 18
6.3.1 Inleiding 19
6.3.2 Wat is de beste techniek voor het visualiseren van IFC-bestanden uit Tekla? 19
6.3.3 Welke technieken blijven over in de Short-List? 19
6.3.4 De overgebleven frameworks 20
6.3.5 Resultaten IFC-viewer onderzoek 20
6.3.6 Resultaat “Welke techniek kan het best gebruikt worden voor de integratie van het 3D model?” 21
6.4 Backend onderzoek 21
6.4.1 Hoe kan het IFC-bestand via de API gedeeld worden? 21
6.4.2 Wat kan er gedaan worden om performance op de tablet optimaal te houden? 22

6.4.3 Zou de API een stuk pre-processing kunnen doen om het inladen van IFC-bestanden makkelijker te maken? ... 22
6.4.4 Hoe gaan we de webapp op Android tablets en browsers draaien? ... 23
7. De designfase ... 24
7.1 Inleiding ... 24
7.2 Bepalen van de requirements ... 24
7.3 De requirements ... 26
7.4 Het systeemontwerp ... 27
7.4.1 Het initiële systeemdiagram ... 27
7.4.2 Het usecase diagram ... 29
7.4.3 De flowchart ... 31
7.4.4 Uiteindelijke webapp structuur in een systeemdiagram ... 33
7.4.5 Dependency diagram ... 34
7.5 Het UX/UI-design onderzoek ... 35
7.5.1 De Emphasize fase ... 35
7.5.2 De Define fase ... 37
7.5.3 De Ideate fase ... 39
7.5.4 De Prototype fase ... 39
7.5.5 De Test fase ... 39
8. De implementatie fase ... 41
8.1 Het initiële systeem design ... 41
8.2 De JavaScript front-end ... 42
8.2.1 Wireframes ... 48
8.3 De Android-applicatie ... 48
8.4 De backend ... 50
9. De test fase ... 51
9.1 UX/UI tests (de test fase van design thinking) ... 51
9.1.1 De initiële test voor het beste design ... 51
9.1.2 De low fidelity prototypen ... 53
9.1.3 De high fidelity prototypen ... 54
10. Projectmanagement reflectie ... 55
10.1 Planningsmethode ... 55
10.2 Overlegstructuur ... 55
11. De conclusie ... 57
12. Aanbevelingen ... 58
12.1 Migratie API van NodeJS naar C# ... 58

12.2 Highlighten via GUID 58
12.3 Maak ConstruSteel On Premise klaar voor het aanleveren van IFC-bestanden 58
12.4 Optimalisatie IFC-bestanden 58
12.5 Verder onderzoek naar performance en performanceverbetering 58
13. Persoonlijke reflectie 59
Bibliografie 60
Bijlagen 62

Figuur 1: De huidige 2D-view in de Digitale Werkplaats 9
Figuur 2: De 2D-tekening van een merk 13
Figuur 3: De 3D-visualisatie met rode markering van het merk van de trap. 14
Figuur 4: Een gemarkeerd merk in het rood. In dit geval de trap van het ConstruSteel kantoorgebouw. 24
Figuur 5: Het systeem diagram ConstruSteel. 28
Figuur 6: Een uitgebreid sequence diagram aan de hand van het systeem diagram hierboven. 28
Figuur 7: Het usecase diagram 30
Figuur 8: Een versimpeld sequence diagram van het werkproces 31
Figuur 9: Een flowchart van het proces van ConstruSteel 32
Figuur 10: Het sequence diagram voor de webapp 33
Figuur 11: Het uiteindelijke systeem 34
Figuur 12: Het dependency diagram 35
Figuur 13: Bird's Eye View, dit toont de voordelen van een 3D-visualisatie naast een 2D-tekening. .. 36
Figuur 14: Een 2D-view van een staal-model. 37
Figuur 15: De 3D-viewer na de derde feedback sessie 40
Figuur 16: Het deployment diagram van het proof of concept 41
Figuur 17: Het class diagram van de webapplicatie. 43
Figuur 18: De eerste vertoning van een 3D-model in Three.js 43
Figuur 19: De code die de RuntimeGenerator error oploste 44
Figuur 20: Het IFC-bestand van het model van het ConstruSteel kantoor 44
Figuur 21: Een gemarkeerd IFC-element. De trap is gemarkeerd in het rood. 45
Figuur 22: Een voorbeeld van een IFCELEMENTASSEMBLY 46
Figuur 23: De huidige looks en lay-out van de IFC-viewer 47
Figuur 24: De uiteindelijke lay-out van de 3D-viewer 47
Figuur 25: Een screenshot van de Android applicatie waar de knoppen weer groter lijken. 48
Figuur 26: De 3D-viewer in Android. Er komen nog wat lay-out aanpassingen 49
Figuur 27: De uiteindelijke viewer zoals hij in Android komt (afbeelding gemaakt met de telefoon) . 49

Tabel 1: De tabel die de IFC-frameworks op vier punten beoordeelt. 20
Tabel 2: De requirements gebaseerd op de onderzoeken en de requirements vanuit de productmanager. Deze tabel bevat alle functionele requirements geordend op de MoSCoW prioriteit. 26
Tabel 3: De tabel met alle niet-functionele requirements, geordend op MoSCoW prioriteit 27
Tabel 4: De feedback op de papieren prototypes. Gegeven door André en Pascal 52

Tabel 5: De feedback op de digitale LoFi-designs van André en Pascal 53
Tabel 6: De feedback van André en Pascal op het eerste HiFi prototype 54

1. Begrippenlijst

ConstruSteel/ConstruSteel On Premise (in applicatie context) – De applicatie die werkvoorbereiders, projectbeheerders en mensen die de calculatie van het project regelen gebruiken.

Digitale Werkplaats – De app gevormd door de samenwerking tussen de API en de Androidapplicatie.

De Android app - “De module “Mijn werk“ in de ConstruSteel mobiele applicatie is gericht op de digitale werkplaats. Het uitgangspunt van de digitale werkplaats zijn merk werkbatches. Deze zullen aangemaakt moeten worden in ConstruSteel. De voortgang van de merken kun je inzien in de module Productiebeheersing. In de module op de app krijgt de gebruiker een to-do-lijst te zien met het werk waar hij mee bezig moet en kan hierop verschillende acties uitvoeren. Eén van de belangrijkste onderdelen in deze module is het digitaal inzien van tekeningen (merken/positienummers).” (De Groot, 2021)

Het planbord - “Op het planbord is een overzicht met nesting, posnr en/of merk werkbatches vanuit ConstruSteel te zien. Dit planbord heeft als doel om de werkbatches te plannen, zodat men op de werkvloer weet waar ze mee bezig moeten.” (Wind, 2021)

IFC – Information Foundation Classes: “Een IFC-bestand is een modelbestand dat gemaakt is met de IFC-(Industry Foundation Classes)-indeling, een open bestandsindeling dat wordt gebruikt door BIM-(Building Information Modeling) programma's. Het bevat een model van een gebouw of faciliteit, inclusief ruimtelijke elementen, materialen en vormen." (Sharpened Productions, 2022)

Merk – een onderdeel van een staalconstructie bestaande uit meerdere posnummers.

OBJ – een standaard 3D-bestandsindeling. “Een OBJ-bestand is een standaard 3D-beeldformaat dat kan worden geëxporteerd en geopend door verschillende 3D-bewerkingsprogramma's. Het bevat een driedimensionaal object, inclusief 3D-coördinaten, texture maps, veelhoekige oppervlakken en andere objectinformatie. OBJ-bestanden kunnen ook verwijzingen naar een of meer MTL-bestanden die oppervlakteschaduwmateriaal voor het object bevatten.” (Sharpened Productions, 2022)

Posnummer – een enkel onderdeel dat met meerdere posnummers een merk kan vormen.

Werkvoorbereider – de persoon/personen die binnen een bedrijf de werkvoorbereiding regelt voor de productiefase van het staalbouwproces. Deze persoon/personen bereiden de productie-opdrachten voor de fabrieksmedewerkers voor.

Tekla – De software waar staalbouw- en constructiebedrijven hun modellen voor calculatie en voor de technische tekeningen in uitwerken. Deze modellen zijn in 3D.

WebAssembly/WASM – “WebAssembly is een nieuw type code dat kan worden gebruikt in moderne webbrowsers. Het is een assembly-achtige taal op laag niveau met een compact, binair formaat. Het wordt uitgevoerd met near-native performance en biedt talen zoals C/C++, C# en Rust een compilatiedoel aan, zodat ze op het web kunnen worden uitgevoerd. Het is ook ontworpen om naast JavaScript te draaien, waardoor beide talen kunnen samenwerken." (MDN, 2022)

2. Inleiding

2.1 Het bedrijf in context

ConstruSteel is een bedrijf dat software produceert voor staalbouw- en constructiebedrijven. Ze zetten in op de digitalisering en automatisering van het traditionele staalbouwproces waarbij nog veel met de hand gedaan wordt. In een zin zou ConstruSteel zich zo beschrijven: “Elke dag werken wij hard aan het optimaliseren van het staalbouwproces om meer efficiency en minder faalkosten te realiseren voor onze klanten” (Jansen, 2018)

ConstruSteel bouwt en ondersteunt generieke software die door iedere staalbouwer te gebruiken is. Dit zal het staalbouwproces efficiënter maken en daarmee kan een productieproces met minder faalkosten gerealiseerd worden. ConstruSteel zet zich in op innovatie, zodat de klanten van ConstruSteel elke dag met de meest innovatieve software aan de slag kunnen. Van werkvoorbereiding tot aan productie wordt alles geautomatiseerd en gedigitaliseerd om efficiëntie te verbeteren en (faal)kosten te verlagen.

Momenteel werken er 23 mensen die zich specialiseren in ICT. Deze specialisaties variëren van de business kant van ICT tot aan software engineering. Er is veel kennis op het gebied van software en support bij ContruSteel aanwezig.

2.2 De huidige situatie

In de context van de opdracht zijn er twee producten die een grote rol spelen. De ConstruSteel On Premise client en de Digitale Werkplaats mobile app.

2.2.1 ConstruSteel On Premise

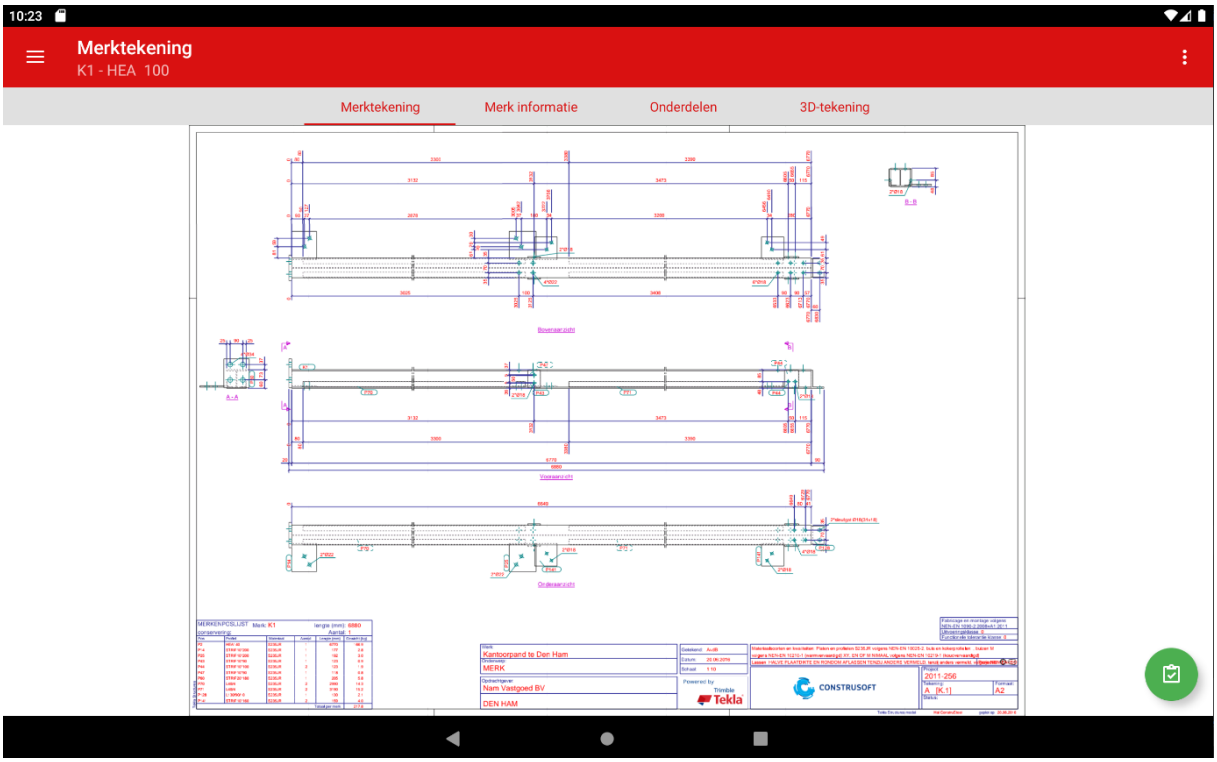
In ConstruSteel On Premise worden de projecten die aangenomen worden gecalculeerd. Deze calculatie wordt naar de klant gestuurd. Als de klant akkoord gaat wordt er een factuur opgesteld. Deze kan dan in termijnen betaald worden. Zodra dit alles rond is komt de werkvoorbereiding op gang. Dit kan allemaal een stuk makkelijker dan wanneer dit op papier en in Excel moet. Alles (prijzen van staal, parameters voor in facturen) zijn vooraf gedefinieerd. Dit helpt een werkvoorbereider en calculator veel. Want als dit allemaal met de hand en uit het hoofd bepaald moest worden duurt het proces een stuk langer. Dit manueel doen is ook een stuk foutgevoeliger.

2.2.2 De Digitale Werkplaats

Nadat het kantoor geautomatiseerd en gedigitaliseerd was, is het tijd om datzelfde te doen met de werkplaats. Op het moment werken veel staalbouwbedrijven nog veelal met papier. Deze kunnen beschadigd of kwijtraken. Dan moet er telkens opnieuw op papier zaken uitgeprint worden. Dit kost allemaal tijd en geld.

De Digitale Werkplaats vervangt het papier met tablets. Deze tablets zorgen ervoor dat de informatie altijd up to date is door de live API-connectie. Daardoor zijn tekeningen altijd up to date. Kunnen ze de tekeningen niet kwijtraken en als er wijzigingen zijn komen ze direct bij de persoon binnen. Dit heeft bij de klanten die deze vernieuwing hebben geïmplementeerd voor veel gemak gezorgd. Er hoeft nu geen papier meer geprint en verdeeld te worden, en bij wijziging van klantwensen is het stukje werkdruk om alles te opnieuw te printen en te verdelen minder, omdat het live via de API wordt geüpdatet. Zowel voor de fabrieksmedewerkers als voor de werk chef en productiemanagers.

Maar ook deze 2D-visualisatie brengt nog problemen met zich mee. Soms mist een stukje context. Hoe is bijvoorbeeld de plaatsing van een balk in een constructie? Een 3D-visualisatie zou deze missende context aan kunnen vullen.



Figuur 1: De huidige 2D-view in de Digitale Werkplaats

2.3 Het probleem dat opgelost moet worden

Naarmate tijd, en technologie, vordert wordt de wens voor 3D-visualisatie in de werkplaats veel groter. Werkvoorbereiding in Tekla wordt al in 3D gedaan. Tekla is de software die werkvoorbereiders binnen staalbouw- en constructiebedrijven gebruiken om constructies mee te modelleren. Daarnaast worden er pdf-bestanden gegenereerd die de constructie in 2D tonen. Deze technische tekeningen kunnen nog steeds onduidelijk zijn, omdat er geen context gegeven wordt aan bepaalde onderdelen die net even wat anders zijn of die uitsteken.

3D-visualisatie wordt, voor duidelijkheid voor de fabrieksmedewerker, een must. Daarnaast is het voor ConstruSteel een ambitie die ze al jaren hebben.

Er moet dus een speciale 3D-viewer in de Digitale Werkplaats komen, die de 3D-modellen vertoont als contextuele verduidelijking van bepaalde onderdelen. De PDF-tekening blijft leidend, want dit is de technische bouwtekening. De 3D-viewer wordt de visuele ondersteuning.

2.4 De doelstelling van de opdracht

Het doel van deze opdracht is om de Digitale Werkplaats te voorzien van een extra viewer. Naast de PDF-viewer moet er dus een 3D-viewer komen. Daarnaast wordt het merk gemarkeerd. Dit houdt in dat een serie aan losstaande onderdelen (posnummers) met een kleur gemarkeerd worden, zodat de medewerker die op dat moment werkt aan een merk van een project weet hoe een onderdeel in de constructie geplaatst zal worden.

2.5 Deliverables

Tijdens het afstuderen komt er een aanvulling op de Digitale Werkplaats tot stand. De deliverables die daarbij horen zijn:

- Onderzoeksrapport over IFC-bestanden en welke techniek het best te gebruiken is;
- Het UX/UI onderzoek;

- Documentatie;
  - Functionele documentatie;
  - Technische documentatie;
- De broncode;
- De test documentatie;
- Het afstudeerverslag en -presentatie.
  - Presentatie wordt ingeleverd in het daarvoor bestemde inleverpunt.

3. Aanpak project

Dit project heeft twee hoofdaanpakken gehad. De Kanban aanpak, waarmee gewerkt werd toen de afstudeerperiode begon en degene waar het later naar aangepast is om zo op een meer SCRUM-achtige wijze te werken. De aanpak van het project is alleen qua agile strategie aangepast.

3.1 Kanban

Dit project begon met de Kanban aanpak. Deze aanpak hield vast aan “oppakken is afmaken”. In Jira werden stories aangemaakt en die werden niet van punten voorzien. Het was in het begin een kwestie van een voor een oppakken, en pas een nieuwe oppakken als er eentje door review af was.

Deze aanpak is vastgehouden omdat het in het begin met het onderzoek goed werkte. Dit gaf de gelegenheid om de volle focus op een onderzoek story te leggen. De focus leggen op stories was de kern van de aanpak in het begin van het afstuderen.

Volgens de website Kanbanize is Kanban het volgende:

“Kanban is een populaire Lean workflow management methode voor het definiëren, beheren en verbeteren van diensten die kenniswerk leveren. Het helpt u werk te visualiseren, de efficiëntie te maximaliseren en continu te verbeteren.” (Kanbanize, 2022)

3.2 Overgang SCRUM-achtige werkwijze

Na een zekere periode is de overgang naar een SCRUM-achtige werkwijze gemaakt. Met deze overstap is er nog meer inzicht in de hoeveelheid werk gekomen. Deze werkwijze werkte prettiger, gaf meer rust en gaf een beter inzicht in de taken die er staan.

Deze werkwijze was een definitie van 6 weken, met per week een doel en daarin de stories die voltooid moesten worden om dat doel te halen. Deze werkwijze leek meer op SCRUM omdat er een wekelijkse oplevering en een wekelijkse retrospective in beeld kwam (Bijlage 7. Retrospective (Adam, 2022) voor een voorbeeld van zo’n Retrospective).

“Scrum is een agile werkwijze die je wendbaarheid vergroot en het mogelijk maakt om zo goed mogelijk aan te sluiten op de wensen van je doelgroep en eindgebruiker. Waar bij een traditionele aanpak veel tijd zit in de voorbereiding en het schrijven van plannen, zorg je met Scrum dat je zo snel mogelijk waarde toevoegt voor de klant.” (Scrum company, 2022)

4. Contextanalyse

4.1 De doelgroep

Het is belangrijk dat de doelgroep van deze casus bekend is. “Doelgroep is een term voor een subgroep consumenten binnen een grotere groep consumenten.” (DoubleWeb, 2021). Met doelgroep “iedereen” kan een product niet tot stand komen. Iedereen heeft andere belangen.

In dit geval gaat het ook niet om de doelgroep “staalbouwbedrijven”. Dat is te breed en binnen zo’n bedrijf zitten natuurlijk meerdere belanghebbenden. Een CEO en een kantoormedewerker hebben andere belangen dan een medewerker van productie.

Uiteindelijk moet de 3D-viewer in de Digitale Werkplaats komen, dus de doelgroep is gespecificeerd naar “De fabrieksmedewerkers van staalbouw- en constructiebedrijven”. De fabrieksmedewerkers staan aan de productielijn en hebben het meest te maken met de Digitale Werkplaats. Dit is waar ze de nodige technische tekeningen in moeten kunnen vinden.

4.2 Klassieke werkwijze staalbouwers

Voordat ConstruSteel begon met het creëren van de Digitale Werkplaats en het principe van een papierloos productieproces, werkten staalbouwers met veel papier. Vanaf engineering werden alle iteraties van ontwerpen meermaals uitgeprint ter controle en ter bevestiging van de ontwerpen.

Als een ontwerp dan klaar was, en de klant had akkoord gegeven, kan het in productie genomen worden. De werkvoorbereider print alle papieren meervoudig uit en geeft ze door aan de productiechef. De productiechef verdeelt de papieren door de fabriek. Op deze manier krijgt iedereen, die aan een project werkt de juiste informatie.

Alle werkinstructies komen aan in merken en posnummers. Deze twee termen beschrijven een onderdeel, of een constructie bestaande uit onderdelen. Een merk is een constructie bestaande uit meerdere posnummers. Een posnummer is een los onderdeel.

Als er een wijziging komt moet de productiechef al het papier en weghalen en vervangen met de nieuwe geüpdatete tekening. Dit kost tijd en geld om te doen. Dit verhoogt ook de faalkosten, omdat het een foutgevoelig proces is.

Zodra alles vervangen is met de juiste papieren kan de productie weer verder. Dan kunnen de fabrieksmedewerkers weer aan de slag.

5. Probleemanalyse

5.1 De huidige oplossing

De huidige oplossing van ConstruSteel is de Digitale Werkplaats. Deze mobiele Android app vervangt al het papier. De term “paperless production” komt daar vandaan. De tablet elimineert de behoefte voor papier in de fabriek. Dit scheelt in kosten voor de fabrikant en is beter voor het milieu.

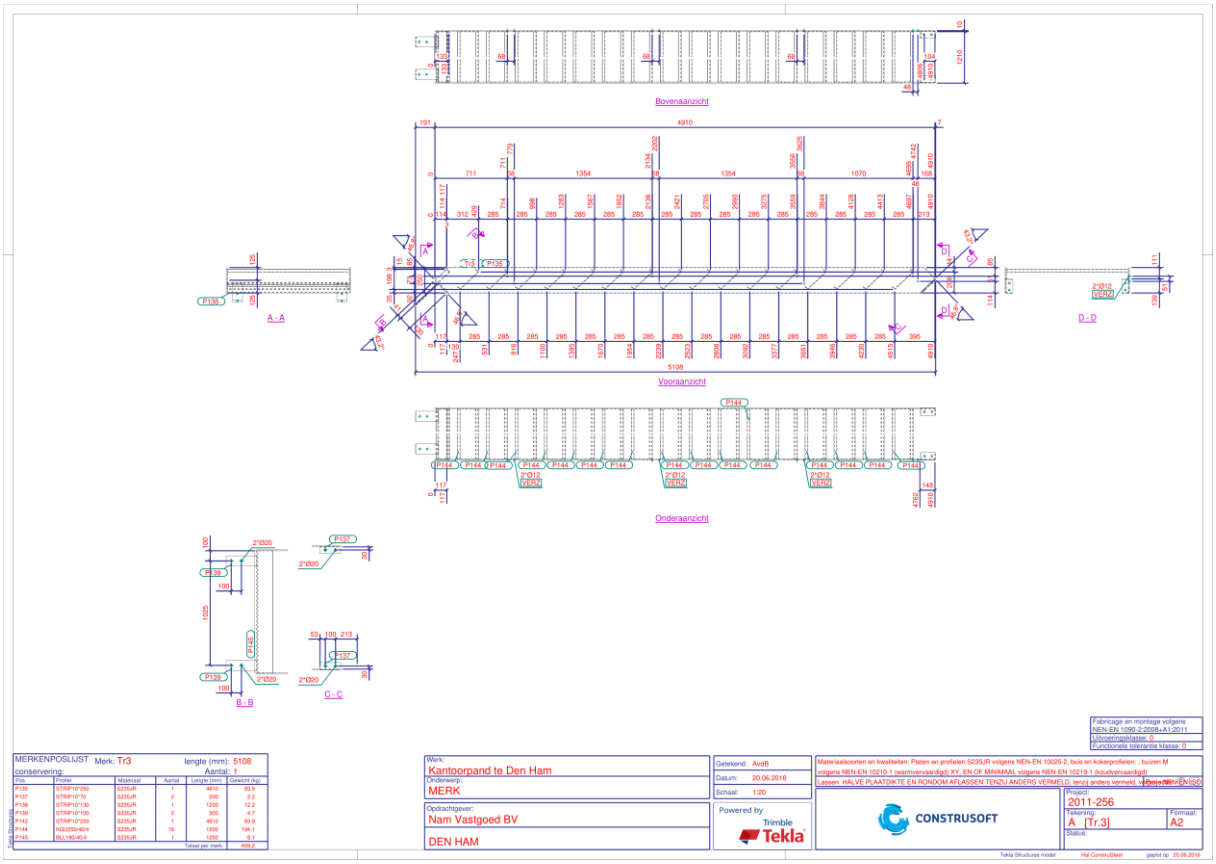
Daarnaast worden de tekeningen live geüpdatet. Er is dus niet meer een heel proces bij wijziging dat alle tekeningen vervangen moeten worden door de werk- en productiechef. Dit gaat nu allemaal via de ConstruSteel API. Dit is een stuk minder foutgevoelig, want met een klik op de knop kunnen wijzigingen bekend gemaakt worden in de Digitale Werkplaats.

Ook zal de Digitale Werkplaats de faalkosten verminderen. Het digitale display toont de tekeningen helderder, en de lijnen van de tekeningen kunnen beter gevolgd worden op het scherm en

tekeningen kunnen niet dusdanig beschadigd raken dat ze onleesbaar worden. Een citaat van de pilot klant:

“Wat bij geprinte versies nog wel eens discutabel is – waar gaat dat lijntje naartoe? – is dat bij pdf duidelijk te zien” (De Swart, 2021)

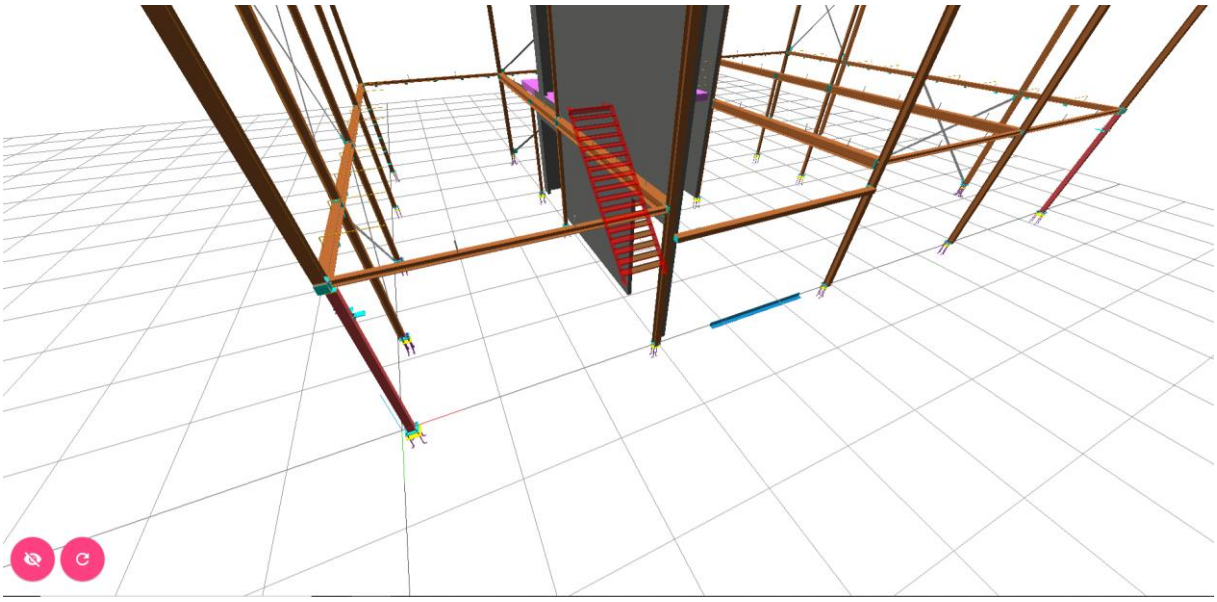
Dit maakt duidelijk dat de 2D-visualisatie er al op vooruit is gegaan. En soms blijft het nog onduidelijk. Een 2D-tekening moet een fabrieksmedewerker in het hoofd interpreteren en visualiseren naar een 3D-tekening. Dit is lastig, want niet iedereen kan op die manier in beelden denken. Een 3D tekening draagt bij aan het inzichtelijk maken van extra context rondom gebruik en toepassing van het merk. Het merk wordt op zijn plaats in de constructie gevisualiseerd.



Figuur 2: De 2D-tekening van een merk

Deze tekening geeft een technisch overzicht van wat er gemaakt moet worden. Maar het is lastig te zeggen dat dit ook daadwerkelijk een trap is. Het ziet er uit als een balk met wat lijntjes erdoor als een fabrieksmedewerker hier zo tegenaan kijkt.

Dat is waar deze afstudeeropdracht op ingaat. Wat kan nog gedaan worden om het nog duidelijker te maken? Zoals eerder aangegeven is 3D-visualisatie met de mogelijkheden van tegenwoordig de meest voor de hand liggende oplossing:



Figuur 3: De 3D-visualisatie met rode markering van het merk van de trap.

Het markeren is van belang zodat de fabrieksmedewerker een duidelijk punt heeft waar hij naar kan kijken. De trap is met een rode kleur gemarkeerd. Zo kan de fabrieksmedewerker tussen alle andere gekleurde onderdelen zien welke er uit springt en welke geassembleerd moet worden.

Fabrieksmedewerkers zullen met 3D-visualisatie beter hun werk kunnen uitvoeren, omdat ze door de 3D-visualisatie het merk ineens op het gebouw zien zoals het erop gemonteerd wordt. Daarnaast zouden mensen beter en gemakkelijker hun werk kunnen uitvoeren als ze die extra context erbij hebben.

5.2 Hoofd- en deelvragen

In hoofdstuk 3.4 staan alle onderzoeksvragen gedefinieerd. De vragen gaan in op de functionaliteit en de mogelijkheden die een IFC-viewer in een Android app met zich meebrengen. De IFC-viewer is een app die IFC-bestanden moet visualiseren. IFC-bestanden zijn specifieke bestanden uit de staalbouw. Zoals Sharpened Productions het uitlegt: “Een IFC-bestand is een modelbestand dat is gemaakt in de IFC-indeling (Industry Foundation Classes), een open bestandsindeling die wordt gebruikt door BIM-programma's (Building Information Modeling). Het bevat een model van een gebouw of faciliteit, inclusief ruimtelijke elementen, materialen en vormen.” (Sharpened Productions, 2022)

De hoofdvraag van dit onderzoek luidt: “Wat kan de IFC-viewer bijdragen aan het bestaande papierloze werkproces?”

Uit deze hoofdvraag kwamen de volgende (technische) deelvragen naar voren:

I. Welke techniek kan het best gebruikt worden voor de integratie van het 3D model?
   a. We spreken hier over het visualiseren van IFC-bestanden. Wat is hier de beste techniek voor?
   b. Wat is er qua performance mogelijk op het huidige moment qua techniek?
II. Hoe kan het IFC-bestand via de API gedeeld worden?
   a. Wat kan er gedaan worden om performance op de tablet optimaal te houden?
   b. Zou de API voorafgaand pre-processing kunnen doen om dit makkelijker te maken?
   c. Hoe gaan we dit op Android tablets en browsers draaien, zonder dat het de snelheid van productie en efficiëntie beïnvloed?

III. Hoe kan de user experience laagdrempelig gemaakt worden voor lager geschoolde vakmensen en meerdere nationaliteiten?
    a. Welke gebaren (zoomen, swipen etc.) zijn het meest ideaal voor in de app voor de fabrieksleden (vuile handen, handschoenen)?
    b. Hoe kan de maatvoering zo duidelijk en toegankelijk mogelijk weergegeven worden?
    c. Hoe te testen met users?
    d. Hoe kan de gebruikswijze van een 3D-omgeving duidelijk gemaakt worden?
    e. Hoe worden de standaard 3D-omgeving gebaren duidelijk gemaakt bij een gebruiker (gezien deze heel iets afwijken van dat van een 2D-omgeving)?
    f. Hoe wordt de taal zo universeel mogelijk gehouden
        i. Onderzoek hoe er zo min mogelijk woorden gebruikt kunnen worden.
        ii. Onderzoek welke iconen het best gebruikt kunnen worden (bijvoorbeeld Google Material Design Iconen).

IV. Hoe wordt de belangrijkste data uiteindelijk van de API (C#) naar het front-end (Android en Web) aangeleverd?

V. Welke informatie in de IFC-bestanden tast de performance aan?
    a. Welke informatie is cruciaal?
    b. Welke informatie kan weggelaten worden om het programma qua performance zo snel mogelijk te laten werken?

VI. Welke informatie moet er getoond worden aan de medewerkers in de fabriek?

Onderbouwing hoofd- en deelvragen:

De IFC-viewer zal uiteindelijk bijdragen aan het papierloze werkproces van de staalbouwbranche. Maar wat gaat er precies bijgedragen worden? Met de technische deelvragen en de applicatie die daaruit voortvloeit gaat dat ondervonden worden.

I. Deze onderzoeksvraag (deelvraag 1 “welke techniek kan het best gebruikt worden voor de integratie van het 3D-model?”) Is van belang omdat er een techniek moet komen die zo lichtgewicht mogelijk is. Het moet op Android tablets gedraaid kunnen worden.
    a. De gegenereerde modellen zijn complex en specifiek. Hier moet goed gezocht worden naar een goed te gebruiken framework. Het mag niet te zwaar zijn en vooral niet te complex ogen. Als de code te complex wordt is het niet meer onderhoudbaar, en de user interface moet ook laagdrempelig, en daarmee met een blik te begrijpen, blijven.
    b. Er kan tegenwoordig al heel wat meer bereikt worden dan een aantal jaar geleden. Er moet wel onderzocht worden hoe de performance van deze applicatie het beste bewaakt blijft. Het inladen, maar ook het gebruik van de viewer zelf mag niet te sloom zijn.

II. Er moet onderzocht worden hoe een IFC-bestand via een API gedeeld kan worden. Het gaat hier om een REST-API.
    a. De reden dat het om een REST-API gaat, is omdat ConstruSteel een REST-API heeft ontwikkeld. De opdracht moet op de producten van ConstruSteel aansluiten.
    b. Er moet onderzocht worden of de backend iets kan doen om de performance op de tablet te optimaliseren.
    c. Er moet goed gekeken worden naar hoe maatvoering op het scherm getoond moet worden, en of het wel getoond moet worden via de 3D-viewer.
    d. Er moet onderzocht worden wat de beste methode is om de applicatie te draaien. Dat ligt het meest aan de techniek die gebruikt gaat worden. Als het web is, dan zal

er een manier gevonden moeten worden om de webapp te integreren in Android. Als het Android native is, wat zou dan een aantrekkelijke oplossing zijn als dit naar een online platform omgezet moet worden?

III. De eindgebruikers zijn gewend aan een simpele UI. De Digitale Werkplaats heeft een simpele Material Design UI. Dit is makkelijk te gebruiken en praktisch iedereen met een telefoon kan die lay-outs gemakkelijk begrijpen. De 3D-viewer mag daar geen uitzondering op zijn. Daarom is het van uiterst belang om goed uit te zoeken hoe de UI daarop inspeelt.
   a. Er moet onderzocht worden wat de bekendste en populairste bewegingen zijn voor een 3D-viewer
   b. Er moet onderzocht worden hoe de maatvoering voor de fabrieksmedewerkers zo duidelijk mogelijk weergegeven moet worden.

IV. Er moet onderzocht worden hoe de API IFC-bestanden naar de front-end stuurt. Het is daarbij van belang om ook te onderzoeken welke data van het IFC-bestand benodigd zijn om de fabrieksmedewerker te ondersteunen bij het assembleren van het merk.
   a. Bij live wijzigingen wordt een nieuw IFC-bestand geüpload en die moet dan naar de Android app gestuurd worden.
   b. Het is de bedoeling dat telkens de laatste versie verkregen wordt.
   c. Er moet nagedacht worden over hoe iemand op de hoogte wordt gebracht van zo’n verandering. De meest logische wijze om dit te doen is als bij de huidige PDF-viewer. Daar gebeurt het volledig automatisch.
   d. Mochten volledig automatische updates niet mogelijk zijn, dan moet er gekeken worden naar manieren om mensen te attenderen dat ze moeten updaten.

V. Om de performance optimaal te houden, is het van belang om te kijken welke informatie weggelaten kan worden en welke erin gehouden moet worden

De informatie die benodigd is, zal worden uitgezocht en als enige meegenomen worden in het visualiseren van het IFC-model.

VI. Om de UX/UI nog wat gemakkelijker te houden, moet er gekeken worden wat getoond moet worden aan de fabrieksmedewerker. Hierbij moet er gedacht worden aan een onderdeel waar men mee bezig is.

6. De onderzoeksfase

Tijdens deze afstudeerperiode werden meerdere zaken onderzocht. In de design fase (hoofdstuk 5.4, 5.5 en 5.7) wordt er ingegaan op de UX en systeem kwesties. Daar wordt in detail uitgelegd wat er gedaan is om tot de juiste user interface en tot het juiste systeem design te komen. In de komende hoofdstukken wordt er ingegaan op het onderzoek naar IFC-bestanden en de technieken om het te visualiseren.

6.1 Bestaande IFC-viewers

In het begin van het traject was het van belang om de beste techniek te zoeken voor het visualiseren van IFC-bestanden. Dit was gedaan door een kort literatuuronderzoek. Er werd gekeken naar bestaande oplossingen, en aan de hand van die bestaande oplossingen konden deelvragen I (welke techniek kan het best gebruikt worden?) en III (Hoe de UI laagdrempelig te houden voor de doelgroep) behandeld worden.

Er is reverse engineering toegepast op bestaande IFC-viewers. Welke applicaties lossen het probleem dat opgelost gaat worden al op? OpenIFCViewer is een van die IFC-viewers. Het is een open source platform waar gemakkelijk een IFC-model getoond kan worden. Alleen dit mist de mogelijkheid om de merken te markeren.

Dan is er nog Tekla. Dit wordt gebruikt door de werkvoorbereiders. Het is daarnaast een duur product om zomaar aan te schaffen voor visualisatie van 3D-modellen. Dit is voor de meeste bedrijven niet te realiseren.

Er zijn meerdere PC-applicaties die IFC-bestanden kunnen vertonen. Deze viewers doen net allemaal niet wat de 3D-viewer voor de Digitale Werkplaats moet kunnen. Daarom moet er gekeken worden naar wat de ontwikkel mogelijkheden zijn.

Er bestaan ook Android apps die IFC-vertoning al mogelijk maken, maar het probleem daarbij is dat ze gratis lijken, maar zodra er maar een bestand ingeladen moet worden willen ze geld zien. Daarom is het grondig onderzoeken naar IFC-formaten en frameworks van belang, zodat we onze eigen viewer kunnen realiseren en bij klanten kunnen installeren.

Daarnaast zijn de IFC-viewers die al bestaan niet te integreren met ConstruSteel. Het is een standalone app. De data zou via een API verstuurd moeten worden naar die standalone viewer. Het is overigens een wens van ConstruSteel om een 3D-viewer in de Digitale Werkplaats te integreren. Gezien dat niet mogelijk is met de bestaande Android IFC-viewers, is er gekozen om onderzoek te doen naar frameworks en zelf een integreerbare viewer te bouwen.

6.2 Onderzoek naar IFC-bestand formaten

Dit komt ook weer terug op deelvraag I (welke techniek is het best te gebruiken), maar ook op deelvraag V (welke IFC-informatie tast de performance aan). Deze vragen zijn beide afhankelijk van het formaat IFC en hoe deze zijn ingedeeld. Om dit uit te zoeken zal er een proof of concept gemaakt moeten worden. Dit is ook het doel en met dat proof of concept kan de hoofdvraag ook beantwoord worden.

“IFC kan gecodeerd zijn in verschillende formaten. Elk formaat komt met zijn eigen voordelen en afwegingen voor softwaresupport. De gegenereerde data kunnen best groot worden (als in gigabytes groot), dit weegt ook mee met de keuze van een IFC-format.” (buildingSMART, 2018)

Het is van belang om de juiste keuze te maken bij het IFC-formaat. Er moet een formaat komen dat performance kan bewaken. Gigabytes aan data over een wifi lijn sturen is niet erg snel. Dat kan de

user experience negatief beïnvloeden. Daarnaast wordt een 3D-omgeving sloom als er te weinig RAM-geheugen beschikbaar is.

Er zijn IFC-bestanden in verschillende formaten, van het standaard IFC-format tot aan de zwaardere XML-IFC bestanden. De meest gebruikte is de STEP Physical file (SPF). Dit IFC-formaat is het meest gebruikt omdat het bij de meeste BIM-software werkt. De grootte van een IFC-step bestand is de standaard meetwaarde in IFC-bestanden. Dit is de 100% waarde. Als iets bijvoorbeeld 25% zwaarder is dan IFC-step, dan is die bestandsindeling 125% groot ten opzichte van IFC-step

Er zijn andere formaten. Zoals het Extensible Markup Language (XML) formaat. Dit heeft een extension van ifcXML, in plaats van de reguliere ifc. Het nadeel van dit formaat is dat het groter kan worden dan de standaard SPF-bestanden. Het size verschil is 13%, dus deze is 113% tegenover het SPF-formaat.

Een betere uitleg van wat ifcXML is volgens BuildingSMART: "De ifcXML-representatie is een implementatie van de ISO-10303 Part 28 Edition 2 ("part 28") standaard. Deze standaard biedt een XML-schemaspecificatie die een automatische conversie van de EXPRESS (ISO 10303 deel 1) representatie van het IFC-schema is. De toewijzing van het express naar XML-schema wordt geleid door een configuratiebestand dat de details van het vertaalproces regelt. Voor ifcXML wordt dit configuratiebestand gestandaardiseerd en gepubliceerd voor elke versie van het bijbehorende IFC-schema." (Nisbet & Liebich, 2007)

Als laatste is er het ZIP formaat. Het heeft de extensie ifcZIP in plaats van ifc. Als er een ZIP formaat van wordt gemaakt dan wordt de file size ook aanzienlijk minder. Met een relatieve size van 17% ten opzichte van een standaard SPF-bestand is het zeer klein.

De definitie van ifcZIP: "IFC-gegevens kunnen zijn ingesloten in een ZIP-bestand. De ingesloten gegevens kunnen worden gecodeerd als SPF of XML, waarbij de resulterende grootte doorgaans vergelijkbaar is." (buildingSMART, 2018)

JSON kan ook als IFC-format gebruikt worden. Dit geeft als voordeel dat het een stuk leesbaarder is en dat het ondersteund wordt door vele software tools. Het enige nadeel is, is dat deze bestanden gemiddeld nog groter zijn dan de ifcXML bestanden, met de grootte op 148% tegenover IFC-STEP.

"JSON biedt verbeterde leesbaarheid en profiteert van een breed scala aan softwaretools." (buildingSMART, 2018)

Toch is ifcJSON niet populair. De bestanden zijn groot en dat tast de performance aan. De standaard STEP variant van IFC is binnen de branche die het gebruikt het meest populair. Het ifcJSON formaat is vooral interessant voor ontwikkelaars die al veel met JSON in aanraking komen.

Uit dit onderzoek moest naar voren komen welk IFC-formaat het best is voor de performance en voor het gebruik in de staalbouwbranche. De meest gebruikte blijft de IFC-SPF variant en is daarmee de meest ondersteunde door applicaties als Tekla. Tekla heeft ook een optie om modellen die gemaakt zijn in Tekla direct naar IFC te exporteren.

Alle formaten hebben hun eigen voor- en nadelen. IFC-SPF blijft de voordeligste, omdat deze het meest ondersteund is, en de bestandsgrootte onder de niet-gezipte varianten qua performance het best is.

6.3 Welke techniek kan het best gebruikt worden?

Voor het volledige onderzoek naar de frameworks, zie Bijlage 1. Hier wordt de verkorte versie van het onderzoek beschreven.

6.3.1 Inleiding

In de staalbouwbranche is efficiëntie de sleutel, en voor ConstruSteel is het verhogen van werk efficiëntie het voornaamste doel bij staalbouwers. Het is daarom van uiterst belang dat er goed onderzocht wordt welk framework het meest efficiënt op tablets zal werken en welke het best te onderhouden is voor de ontwikkelaars bij ConstruSteel.

6.3.2 Wat is de beste techniek voor het visualiseren van IFC-bestanden uit Tekla?

In Tekla kunnen alle IFC-formaten gegenereerd worden. Aan de hand van de IFC-techniek die gebruikt gaat worden is het de vraag of er een advies om een bepaald soort IFC-bestand te genereren daarvoor.

De technieken die hiervoor gebruikt kunnen worden moeten aan drie eisen voldoen:

- Ze moeten succesvol IFC kunnen parsen.
- Ze moeten IFC-bestanden kunnen renderen.
- Er moet goede documentatie beschikbaar zijn.

Als aan een van deze drie eisen niet wordt voldaan, dan voldoet het desbetreffende framework niet. Het parsen is belangrijk voor het begrijpelijk maken van de data, het renderen is van belang voor het kunnen visualiseren en de documentatie is van belang voor het begrijpen van de frameworks en het voorkomen en misschien oplossen van bugs.

Er zijn na een klein stuk vooronderzoek een aantal frameworks naar voren gekomen. Er is voornamelijk gezocht op frameworks die met Java of JavaScript werken. Dit komt omdat deze talen goed te integreren zijn met Android. JavaScript kan via een Android WebView, en als het met Java mogelijk is, dan heb je een framework dat op Android Native kan.

Met literatuuronderzoek naar IFC-libraries kwam er een lijst naar voren die bleek aan alle criteria te voldoen. Er is heel globaal gekeken. Er kon nog niet te kritisch gekeken worden, gezien er nog niet veel bekend was over IFC-bestanden in het algemeen. Wat door een kort vooronderzoek (ook literatuur onderzoek) al wel bekend was, is dat de data van IFC te parsen en te visualiseren is met de juiste tools.

De lijst die naar voren kwam:

- Assimp
- IFC.java
- IFC.js
- IFCOpenShell

Nadat deze allemaal opgeschreven waren moest er verder gekeken worden. Wat er nu gedaan was, is lang niet genoeg om een oordeel te vellen. Voor het volledige onderzoek wordt er verwezen naar Bijlage 1.

6.3.3 Welke technieken blijven over in de Short-List?

Voor deze vraag zijn een aantal criteria vanuit de context van de ConstruSteel software opgesteld. Dit gaat vooral om performance, en hoe het werkt. Men moet wel de mogelijkheid hebben om goed te weten wat er op het scherm gebeurt en om het te bedienen. Het moet duidelijk blijven.

Er is gekozen voor een set aanvullende criteria om de keuze makkelijker te maken. Zo kan er ook kritischer naar de IFC-libraries gekeken worden. Als hier een library aan alle criteria voldoet, is de kans dat library gekozen wordt een stuk groter.

De aanvullende criteria zijn:

- Er moet een mogelijkheid zijn tot het bedienen van het display en het manipuleren van het model aan de hand van user input.
- Het moet in Java of JavaScript.
- Het moet te integreren zijn met Android.

De drie frameworks die overbleven zijn de volgende:

- IFC.java
- Assimp
- IFC.js

Alle drie zijn goede IFC-libraries met hun eigen kwaliteiten.

6.3.4 De overgebleven frameworks

Voor het volledige onderzoek naar de overgebleven frameworks, zie Bijlage 1, vanaf hoofdstuk 4

In essentie zijn er drie frameworks overgebleven. Per framework wordt er in het onderzoek document (Bijlage 1, beoordelingen IFC vanaf hoofdstuk 4) beoordeeld waarom of waarom ze niet geschikt zijn. Met als laatste een beoordelingsdocument met alle voor- en nadelen. Daar is gebleken dat IFC.js het beste framework bleek te zijn.

6.3.5 Resultaten IFC-viewer onderzoek

In Bijlage 1 is te lezen hoe er gekeken is naar alle IFC-frameworks. Er zijn veel overwegingen genomen, inclusief ervaring op het gebied van de techniek. Uiteindelijk zijn de vier belangrijkste criteria (viualiseren, renderen, UX/XI en integreerbaarheid met Android) het zwaarst gaan tellen voor de uiteindelijke beoordeling.

De conclusie in een tabel uitgewerkt	*Renderen*	*UX/UI-vriendelijk*	*Integreerbaar met Android*	*Documentatie beschikbaar*
IFC.js	✓	✓	✓	✓
IFC.java	✗	✗	✓	✗
Assimp	✗	✓	✗	✓

Tabel 1: De tabel die de IFC-frameworks op vier punten beoordeelt.

De enige die aan alle eisen voldoet is IFC.js. Deze web-library ondersteunt visualisatie in combinatie met Three.js en de library rendert alles op een overzichtelijke wijze in JSON. Het bevat functionaliteit om belangrijke data te converteren naar een leesbaar JSON-formaat.

Deze mogelijheden hebben IFC.java en Assimp niet direct. Dit komt omdat IFC.java meer gericht is op het leesbaar krijgen van IFC door er een Java syntax aan te geven. Assimp is meer gericht op game engines als Unreal Engine en Blender.

Toch zou IFC.java integreerbaar kunnen zijn met Android, gezien de Android app van ConstruSteel in Java geschreven is. Daarom is de kolom “Integreerbaar met Android” met een vinkje gemarkeerd.

De keuze ligt dus bij IFC.js in combinatie met Plain JavaScript en Three.js. Deze combinatie is gekozen omdat het voldoet aan alle eisen. De documentatie die erbij komt is compleet en voldoet daarmee aan de criteria. Een volledige documentatie helpt bij het voorkomen van bugs. Als de bug voorkomen kan worden, met behulp van een goede documentatie, scheelt het een hele hoop in een toekomstige ontwikkeling. Als de bug dan toch voorkomt, kan de documentatie de oplossing bieden.

Assimp is als kandidaat opgeschreven omdat het in het eerste artikel leek op een library die parsen en visualiseren mogelijk maakt. Als het in C# was, dan was de keuze misschien nog op deze library gevallen, maar C++ is ligt niet in het pakket van ConstruSteel en zal daarom gelaten worden.

6.3.6 Resultaat "Welke techniek kan het best gebruikt worden voor de integratie van het 3D model?"

- Het antwoord op deze deelvraag is "IFC.js" in een woord. Dit komt omdat IFC.js het meest volledig gedocumenteerde framework is. Deze library kan de data van de IFC-bestanden parsen. Het parsen houdt in dat het de data inleest en kan begrijpen binnen het IFC-STEP formaat. Als je IFC-XML (een foutieve IFC-formaat in deze context) zou aanleveren, dan zal de parser een error geven. De parser kent de IFC-indeling dan niet en geeft een foutmelding. Ook kan IFC.js de 3D-geometrie van het IFC-model renderen. Dit zijn twee belangrijke elementen waarom het framework is gekozen.
    - Zoals eerder benoemd is voor het visualiseren van IFC-modellen de keuze van IFC.js het meest haalbaar. JavaScript is bekend en de integratie met Android is te realiseren.
    - Er moet grondig getest worden op performance om te zien wat er gebeurt qua snelheid van een tablet. Een groot IFC-bestand kan nog wat traagheid veroorzaken. Om dit te voorkomen moeten er, of sterkere tablets komen, of er moet gebruik gemaakt worden van slimme JavaScript technieken.
        - De slimme JavaScript technieken waarover gesproken wordt is het asynchroon laten lopen van het proces van de IFC-modellen inladen. IFC.js presenteert het op zo'n manier dat het op multithreading lijkt. Er is gekozen om hier gebruik van te maken om performance te verbeteren.

6.4 Backend onderzoek

In dit hoofdstuk wordt aandacht besteed aan de werking van de backend en hoe een IFC-bestand het beste verwerkt kan worden door de backend.

6.4.1 Hoe kan het IFC-bestand via de API gedeeld worden?

Door middel van een analyse op een bestaand product (in dit geval de ConstruSteel API) is ondervonden hoe de API zijn bestanden met de Digitale Werkplaats deelt. De API van ConstruSteel levert inmiddels al pdf-bestanden aan via een gegeven pad in ConstruSteel. Dit pad definieert het pad waar de API op moet zoeken om bij de juiste bestanden te komen. Deze worden geserveerd aan de Digitale Werkplaats.

Ditzelfde moet gebeuren met IFC-bestanden. Eigenlijk hadden de IFC-bestanden vanuit ConstruSteel on premise aangeleverd worden, echter die moet hierop nog geschikt gemaakt worden. Voor deze opdracht is toen besloten een tijdelijk backend oplossing te bieden via NodeJS REST-API. In overleg met André is er gekozen om een proof of concept API op te zetten die bestanden direct aanlevert naar de IFC-viewer. Om in het proof of concept een zo gelijk mogelijke werking te creëren is er daarom gekozen voor een NodeJS REST-API. En NodeJS is daarnaast de taal die het best samenwerkt met de 3D-viewer webapp. Android stuurt bij het selecteren van de werkbatch het project en de merknaam toe, deze wordt opgevangen in de viewer, en dan zal aan de hand van die data het juiste IFC-model gegeven worden.

Omdat het nu in een andere taal wordt geschreven, komt er een document bij met de belangrijkste technische details. Bijlage 3 (API Definition) is daarvoor opgezet. Zodat het duidelijk is welke API-call wat doet.

Er wordt nu gekozen voor een NodeJS en ExpressJS backend als alternatief op de C#-API. Met deze API kunnen dan verschillende IFC-bestanden aangeleverd worden. Op deze manier kan er bewezen worden dat het programma correct functioneert bij het via de backend inladen van IFC-bestanden.

De NodeJS server zal zorgen voor een volledig werkend prototype dat dynamisch aanvoelt. Het document dat deze API definieert is inmiddels al gedeeld met de collega's van de afdeling. De API-definitie speelt een belangrijke rol in het omzetten naar C#.

6.4.2 Wat kan er gedaan worden om performance op de tablet optimaal te houden?

Door middel van het in elkaar zetten van het prototype kon dit getest worden. Toen het prototype begon met inladen werd het naar Android gehaald om te kijken hoe het daarop draaide. Dit was de enige manier om zeker te weten dat de performance goed zou blijven in de app.

De vraag naar goede performance en de twijfel of het haalbaar is, is WebView. Een WebView is een web-interface geïntegreerd in een app. Volgens Android zelf is een WebView het volgende: "Met WebView-objecten kunt u webinhoud weergeven als onderdeel van uw activity lay-out, maar sommige functies van volledig ontwikkelde browsers ontbreken. Een WebView is handig wanneer u meer controle over de gebruikersinterface en geavanceerde configuratieopties nodig hebt waarmee u webpagina's kunt insluiten in een speciaal voor uw app ontworpen omgeving." (Android Developers, 2022)

Dit kan gedaan worden door een zo'n efficiënt mogelijk framework te kiezen, maar ook door een stuk rendering al aan de backend te doen. De volgende onderzoeksvraag gaat daarop in.

Om performance van het parsen van IFC-bestanden optimaal te houden maakt IFC.js gebruik van WebAssembly: "WebAssembly is een nieuw type code dat kan worden gebruikt in moderne webbrowsers. Het is een assembly-achtige taal op laag niveau met een compact, binair formaat. Het wordt uitgevoerd met near-native performance en biedt talen zoals C/C++, C# en Rust een compilatiedoel aan, zodat ze op het web kunnen worden uitgevoerd. Het is ook ontworpen om naast JavaScript te draaien, waardoor beide talen kunnen samenwerken." (MDN, 2022)

Uiteindelijk is de performance optimaal te houden door de IFC-bestanden zo klein mogelijk te houden. Het inladen duurt het langst. Als het inladen klaar is loopt de IFC-viewer soepel. Het markeren van de merken gaat ook sneller dan het inladen. Dit komt omdat de processor de meeste kracht nodig heeft bij het inladen en berekenen van de geometrische vormen. Het markeren is hooguit een bestaande geometrie een kleurtje geven. Omdat de geometrische vormen al bestaan, is het makkelijker om het dan weer te geven met een extra markering.

6.4.3 Zou de API een stuk pre-processing kunnen doen om het inladen van IFC-bestanden makkelijker te maken?

Door literatuuronderzoek naar een aantal C#-game development sites werd het duidelijk dat rendering geen backend taak is. Er is in deze instantie gekozen voor literatuuronderzoek, omdat een prototype met backend rendering maken geen optie was in combinatie met de frameworks die gebruikt werden. Dat was ook puur front-end rendering. Vanuit daar kwam de vraag "Waarom niet aan de backend" naar voren. De reden is dus het volgende:

Het zou kunnen, maar de keuze in dit project is om dit toch bij het front-end te leggen. De reden? IFC.js is een front-end framework met een set specifieke WebAssembly bestanden die ervoor zorgen dat alle data van IFC-bestanden omgezet worden naar een geometrisch model. De API zou wat kunnen betekenen voor de rendering van 3D-modellen, maar er wordt gekozen om dat niet te doen, omdat IFC.js volledig front-end rendering is.

Uit het stuk van Anton Larin over backend en front-end renderen komt het volgende:

“Het meest voorkomende gebruik van dit type verwerking is het genereren van voorbeeldafbeeldingen voor gebruik op webpagina's in plaats van de 3D-modelviewer zelf. Een andere, heel andere use case is het genereren van renders van een 3D-model.” (Larin, 2021)

De reden dat er gekozen is voor de front-end rendering en niet voor de backend rendering, is omdat er te maken is met 3D-modellen die getoond moeten worden. Als het afbeeldingen van 3D-modellen waren, kon er gekozen worden voor backend rendering, maar dat is niet het geval. Het 3D-model wordt direct gerenderd en moet vertoond worden. Dit is het makkelijkst te bereiken met front-end rendering.

6.4.4 Hoe gaan we de webapp op Android tablets en browsers draaien?

Android heeft interessante widgets om web-interfaces die voor een mobiele omgeving ontworpen zijn te integreren.

Met een Android widget WebView is het mogelijk om (speciaal ontworpen) webpagina’s te tonen in de activity. De WebView zorgt voor een mooie, geïntegreerde webpagina. Daarom is het alleen mogelijk voor speciaal ontworpen omgevingen, zoals Android Developers in de documentatie heeft geschreven.

De 3D-viewer is zo’n speciaal ontworpen web-pagina. Het heeft één doel, en dat ondersteunt de Digitale Werkplaats. Daarom kan het in een layout fragment geplaatst worden binnen de Androidapp.

De code is wat zwaarder en 3D-modellen zijn zwaarder. De performance bij 3D-modellen is op apparatuur met een lager RAM-geheugen en minder processorkracht sowieso lastig goed te houden. Doordat de specificaties van een tablet wat minder zijn, duurt het inladen van het 3D-model ook iets langer.

De staalbouw- en constructiebedrijven gaan speciaal voor een 3D-viewer ook geen betere tablets inkopen. Als de huidige tablets al werken, is het al goed genoeg voor de fabrieksmedewerkers en hun leidinggevenden.

7. De designfase

7.1 Inleiding

Nu het bekend is welke frameworks gebruikt gaan worden, wat doet dat met de requirements en de eisen van het systeem zelf? Dat wordt allemaal nagegaan en beantwoord. De onderzoeksvragen over de technieken worden ook in behandeling genomen, voornamelijk de vragen met betrekking tot de backend.

7.2 Bepalen van de requirements

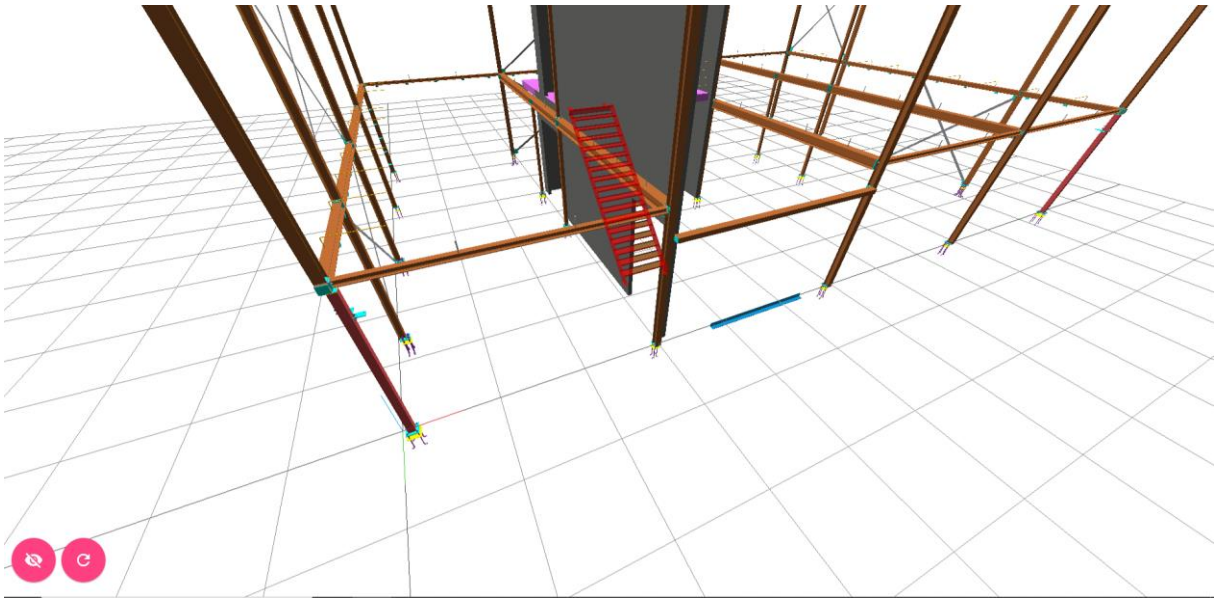
Om de requirements te bepalen is er gekeken naar de onderzoeken. Uit de conclusie kwam dat IFC.js het beste framework is voor dit project. Wat kan het bieden en wat kan het doen om de applicatie te laten zijn zoals het moet zijn?

Het kan visualiseren, dus een IFC-bestand moet getoond worden op het scherm. Daarnaast zijn 3D-modellen zware objecten. De modellen kunnen de performance aantasten. Omdat het in een werkomgeving komt waar efficiëntie van uiterst belang is, mag de performance niet te zwaar aangetast worden.

Om performance nog wat optimaler te houden is er gekozen om een IFC-bestand per project in te laden. Dus als een fabrieksmedewerker op Project A zit, dan krijgt hij alleen Project A te zien. Als de fabrieksmedewerker switcht naar Project B, dan krijgt hij alleen Project B te zien. Per merk wordt er een ander merk gemarkeerd in het model.

Meer requirements uit de onderzoeksfase waren:

- IFC moet vertoond worden;
- Een merk moet gemarkeerd worden. Een merk is een onderdeel bestaande uit meerdere losse onderdelen. Deze losse onderdelen worden posnummers genoemd. Door een merk in het 3D-model te markeren wordt duidelijk wat er geassembleerd moet worden; (Zie hoofdstuk 5.1 voor de eerste uitleg over 2D en 3D-visualisatie).



Figuur 4: Een gemarkeerd merk in het rood. In dit geval de trap van het ConstruSteel kantoorgebouw.

- Performance moet optimaal blijven;

- De 3D-viewer wordt een webapp. Deze webapp wordt geïntegreerd in de Digitale Werkplaats. Zo staat de 3D-viewer centraal in de Android app van ConstruSteel.
- Houd de app qua stijl zo simpel mogelijk.

Vervolgens samen met de productmanager via een meeting te bepalen wat de requirements en functionaliteiten zijn. Vanuit de productmanager zijn de volgende aanvullende requirements naar voren gekomen:

- Een model moet verborgen kunnen worden zodat alleen de gemarkeerde IFC-bestanden getoond worden;
- De camera moet gereset kunnen worden omdat de oriëntatie gauw kwijtgeraakt kan worden in een 3D-omgeving;
- Zorg voor een consistente lay-out. Hanteer de stijl van Android in de webapp (Material Design).

Er wordt natuurlijk ook kwaliteit verwacht. Onder andere simpele, geoptimaliseerde code, een backend connectie (een connectie met een concept API) en geteste code. Daar zijn ook stories voor aangemaakt. De kwaliteit is belangrijk voor de uiteindelijkje proof of concept. Kwaliteit wordt zowel gevraagd als verwacht vanuit ConstruSteel.

7.3 De requirements

Alle requirements zijn functioneel of niet functioneel. Ze zijn ook MoSCoW geprioriteerd. Dit houdt in dat ze een label krijgen: “must”, “should”, “could” of “”would(n’t)”.

- Must geeft aan dat het gerealiseerd moet worden voor de werking van het systeem
- Should geeft aan dat het nodig is voor de functionaliteit maar als het weggelaten wordt, dan is het noodzakelijk geweest om het weg te laten.
- Could geeft aan dat het een nodige requirement gaat worden.
- Would(n’t) geeft aan dat het puur toekomst werk is (nice to have). Dit kan bijna altijd uitgesteld worden.

Functioneel/Niet functioneel (F/NF)	Requirement	MoSCoW	Gerealiseerd/niet gerealiseerd
F	Een IFC-bestand moet getoond worden op het scherm	M	Gerealiseerd
F	Een merk moet op het IFC-bestand gemarkeerd worden	M	Gerealiseerd
F	De UI moet simpel en makkelijk te begrijpen zijn	M	Gerealiseerd
F	De camera moet gereset kunnen worden	S	Gerealiseerd
F	De niet-gemarkeerde onderdelen moeten verborgen kunnen worden	S	Gerealiseerd
F	De gebruiker moet rondom het model kunnen bewegen	M	Gerealiseerd
F	De gebruiker moet vooruit, achteruit en naar links en rechts kunnen bewegen	M	Gerealiseerd
F	De gebruiker moet kunnen inzoomen	M	Gerealiseerd
F	De gebruiker moet kunnen uitzoomen	M	Gerealiseerd
F	De gebruiker moet de camera kunnen resetten	M	Gerealiseerd
F	De gebruiker moet de niet gemarkeerde IFC-elementen kunnen verbergen	M	Gerealiseerd

Tabel 2: De requirements gebaseerd op de onderzoeken en de requirements vanuit de productmanager. Deze tabel bevat alle functionele requirements geordend op de MoSCoW prioriteit.

Functioneel/Niet	Requirement	MoSCoW	Gerealiseerd/niet

functioneel (F/NF)			gerealiseerd
NF	De performance moet op de tablet optimaal blijven	M	Niet gerealiseerd/Een advies voor verbetering
NF	De webapp moet in Android gezet worden	M	Gerealiseerd
NF	De code moet grondig getest zijn	M	Gerealiseerd
NF	Er moet een technisch en functioneel document komen.	M	Gerealiseerd
NF	De UI moet in de stijl van ConstruSteel	M	Gerealiseerd
NF	Pas Material design toe op de webapp – naar de standaarden van Google (zoals in de Digitale Werkplaats)	M	Gerealiseerd
NF	Er moet een integratie met de API komen, voor nu een Proof of Concept API	S	Gerealiseerd
NF	De code voor het markeren moet volledig geoptimaliseerd zijn.	C	Niet gerealiseerd/wordt een advies voor verbetering

Tabel 3: De tabel met alle niet-functionele requirements, geordend op MoSCoW prioriteit

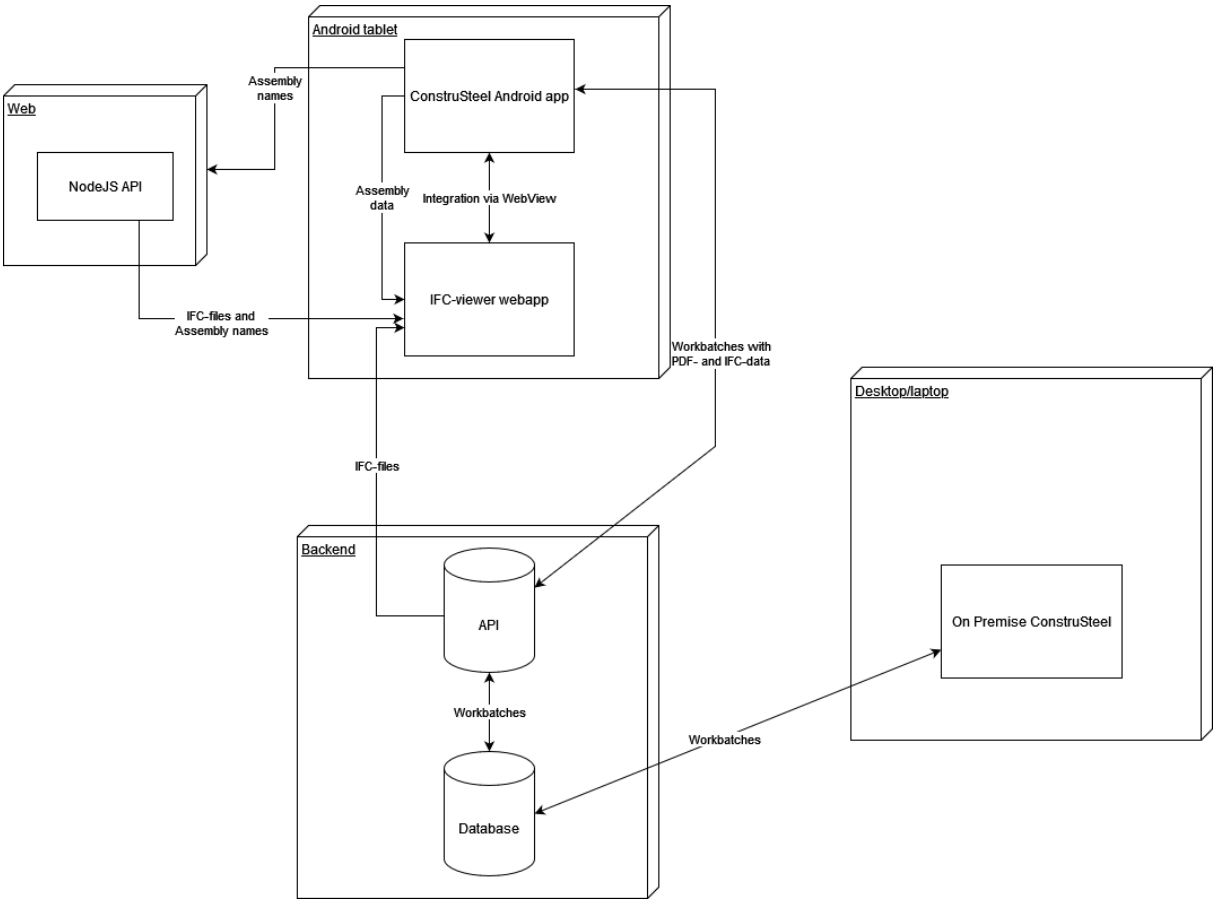
7.4 Het systeemontwerp

7.4.1 Het initiële systeemdiagram

Om het (ideale) systeemontwerp te maken moet er eerst uitgezocht worden hoe ConstruSteel haar systemen in elkaar zitten. Dit is gerealiseerd door een simpel systeemdiagram uit te werken met een duidelijke dataflow:

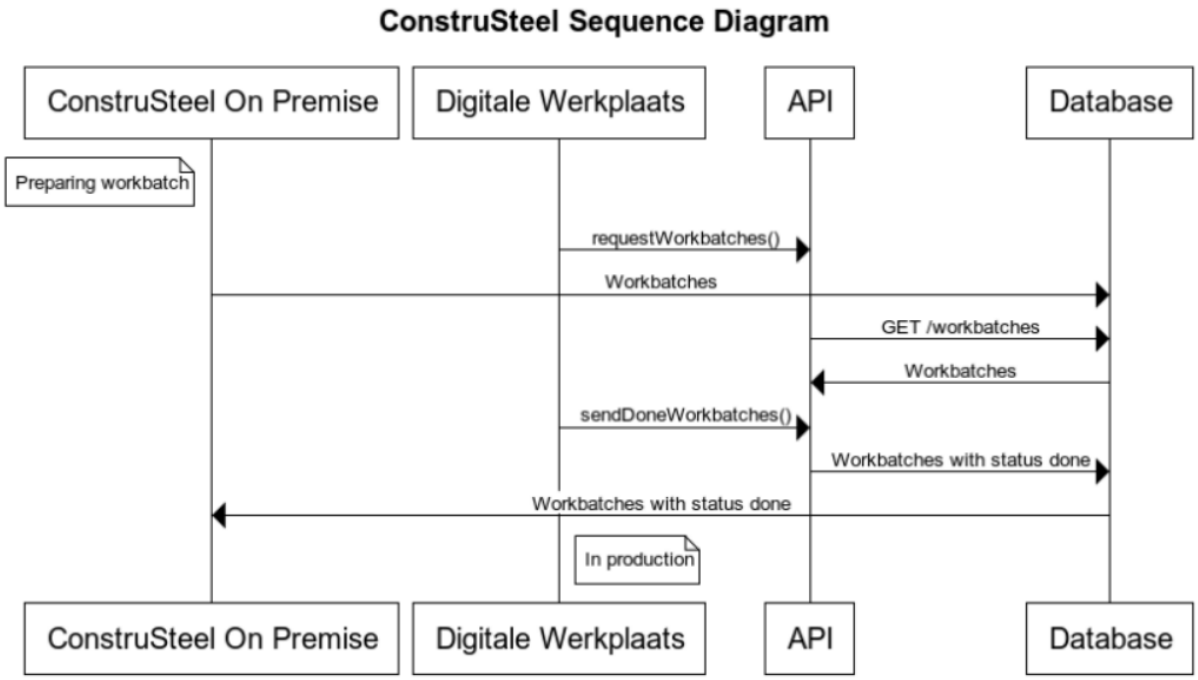
In overleg met Pascal, die veel ontwikkeld heeft aan ConstruSteel On Premise, is dit ontwerp tot stand gekomen. ConstruSteel heeft geen of erg verouderde diagrammen voor hun systemen. Daarom is het besloten om in overleg met André en Pascal een aantal nieuwe te maken ter documentatie en verduidelijking (voor de documentatie van alle diagrammen, zie Bijlage 2).

Vanuit deze modellen is verder gekeken naar de gewenste situatie. Dat is ook in kaart gebracht. Nadat dat in kaart is gebracht, moet het class diagram en het diagram van de IFC-viewer gemaakt worden. Met dit class diagram kan inzichtelijk gemaakt worden hoe de communicatie tussen front-end, backend en Android gerealiseerd kan worden.



Figuur 5: Het systeem diagram ConstruSteel.

Alle datastromen staan uitgeschreven (Bijlage 2, hoofdstuk 3). Hier werd duidelijk dat ConstruSteel On Premise direct met de database praat. En dat de API de communicatie tussen de database en de Android app mogelijk maakt.



Figuur 6: Een uitgebreid sequence diagram aan de hand van het systeem diagram hierboven.

Werkbatches gaan via ConstruSteel naar de database. Zodra de Digitale Werkplaats deze nodig heeft, wordt hij deze via de API opgevraagd. Als een aanvraag van de app binnenkomt, dan stuurt de API de gegevens naar de app toe.

Dit werkt andersom ook zo. In de Digitale Werkplaats wordt een werkbatch op gereed gezet. Dit moet in ConstruSteel On Premise ook aangepast worden. Dan stuurt de Digitale Werkplaats naar de API dat de werkbatch op gereed is gezet. De API schrijft het naar de database, en ConstruSteel haalt het uit de database op.

Nu dit allemaal inzichtelijk is kan er gekeken worden naar de use cases die daaruit voortkomen. Dit diagram is meerdere malen gecontroleerd door André en Pascal om ervoor te zorgen dat het dataproces goed in kaart is gebracht. Dit gaat dus om het huidige, bestaande ContruSteel beeld.

7.4.2 Het usecase diagram

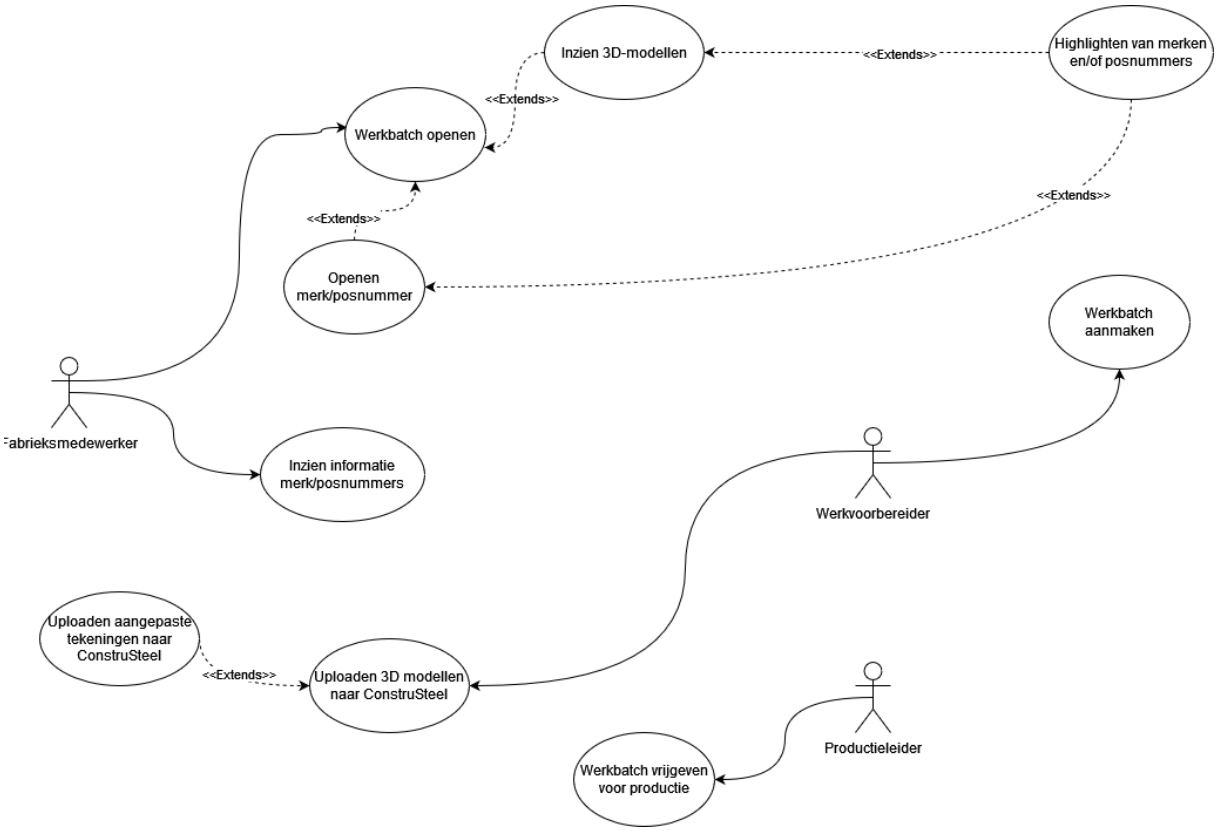
In het use case diagram wordt vastgelegd wat de use cases zijn en wat men kan doen in de app.

Wat zijn de belangrijke use cases van de app in combinatie met de 3D-viewer? Met die vraag op de achtergrond zijn er taken gedefinieerd, en daarmee kon het use case-diagram gevuld worden.

Alle belangrijkste taken zijn daarna omgezet in directe acties die een gebruiker kan uitvoeren, die zijn aan de juiste gebruiker gekoppeld, en uitbreidingen op die taken zijn toegevoegd. Daarmee zijn er op drie actoren gekomen:

- De fabrieksmedewerker
- De productieleider
- De werkvoorbereider

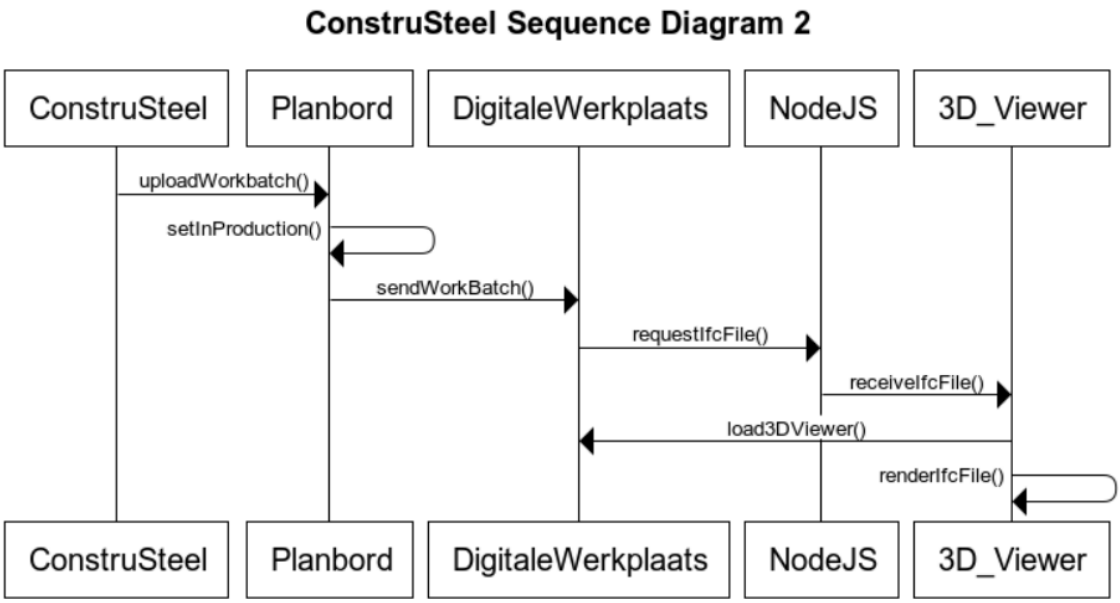
Dit zijn de belangrijkste actoren voor dit systeem. De reden daarvoor is dat de werkvoorbereider de IFC-modellen moet toevoegen aan het systeem, zodat de API ze uit de database naar de Digitale Werkplaats kan sturen. De productieleider kan de werkbatches vrijgeven, zodat ze benadert mogen worden door de fabrieksmedewerkers. Al deze use cases hebben hun ondersteunende en uitbreidende use cases. Deze hebben allemaal betrekking tot de requirements.



Figuur 7: Het usecase diagram

Dit usecase diagram (zie figuur 7, voor een volledige documentatie ervan Bijlage 2, hoofdstuk 2.1) geeft weer wat de usecases zijn en welke gebruiker deze kan uitvoeren. Deze afstudeeropdracht zoomt vooral in op de fabrieksmedewerker en de werkvoorbereider. De werkvoorbereider kan 3D-modellen toewijzen (in IFC-bestanden) en de aangepaste tekeningen opnieuw in ConstruSteel On Premise aanbieden. Daardoor zullen ze naar de NodeJS server gaan en aan de app worden aangeboden.

De werkvoorbereider kan ook een werkbatch aanmaken en doorsturen naar de productieleider. De productieleider kan deze vrijgeven voor productie. De fabrieksmedewerker kan merk/posnummer informatie inzien. Hij kan daarnaast ook de modellen inzien en de markeringen bekijken.

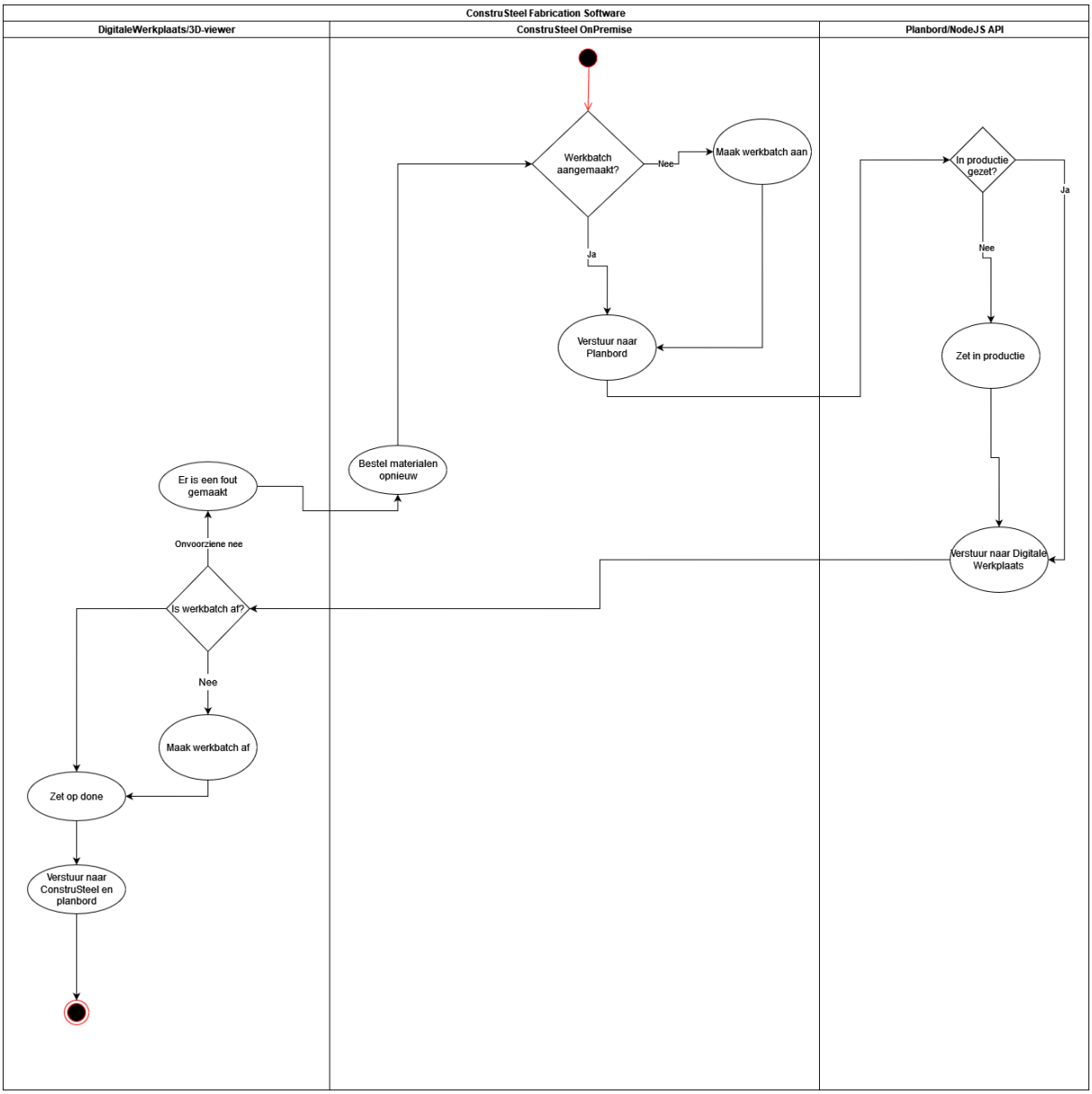


Figuur 8: Een versimpeld sequence diagram van het werkproces.

Dit sequence diagram (zie figuur 8) laat op zien hoe het proces hoe en wat er moet gebeuren om te komen tot 3D viewing. Van het uploaden van een werkbatch tot aan het finaliseren ervan. Het hele ConstruSteel proces wordt hier nogmaals in een notendop uitgelegd. Dit is in samenwerking met de productmanager opgezet, zodat er een duidelijk en functioneel overzicht is van de werkwijze.

7.4.3 De flowchart

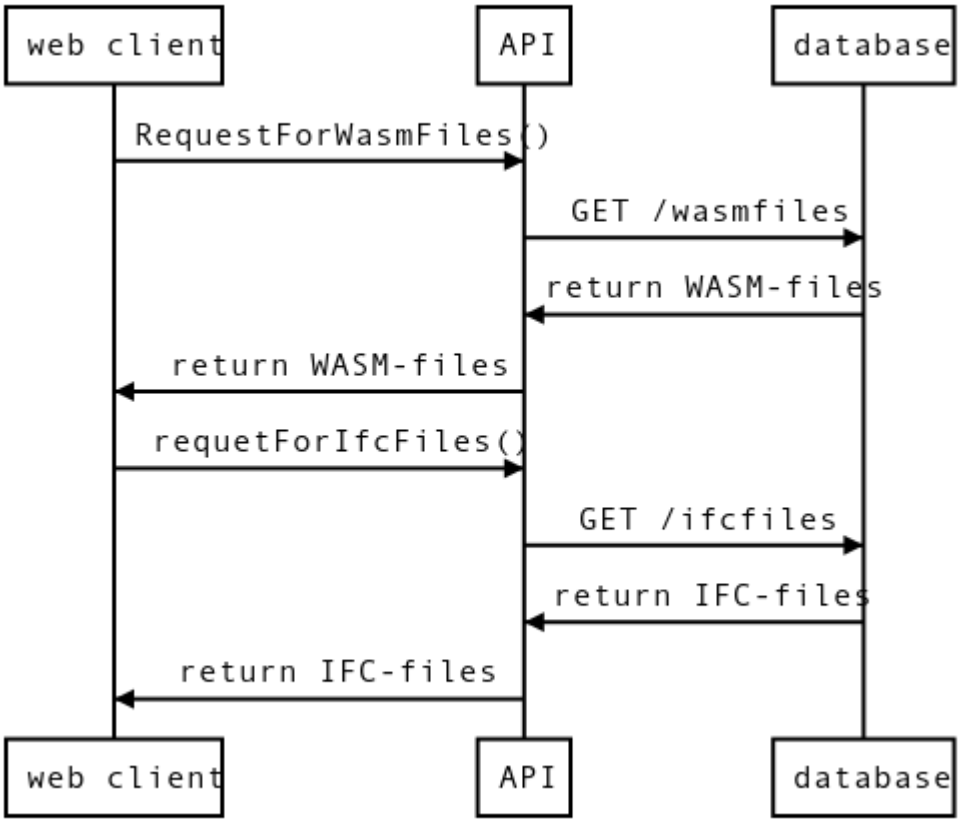
Naast deze technische diagrammen en ideale (foutloze) flows, is het ook belangrijk om het gouden pad van het werkproces te definiëren. In de app zit een ideale werkwijze. Deze is uitgewerkt in een flowchart (zie figuur 9). Deze flowchart laat elke fase en stap zien, en wat er gebeurt als iets fout gaat of anders loopt dan dat het zou moeten:



Figuur 9: Een flowchart van het proces van ConstruSteel

Aan de start van het diagram wordt een UML-node gebruikt om de start aan te geven. Dit is met een rode pijl. Vanuit daar vloeit het gehele proces door tot he bij de eind-node komt of tot het proces terugkomt bij de start-node. (Bijlage 6. Het functionele design (Adam, 2022)).

7.4.4 Uiteindelijke webapp structuur in een systeemdiagram

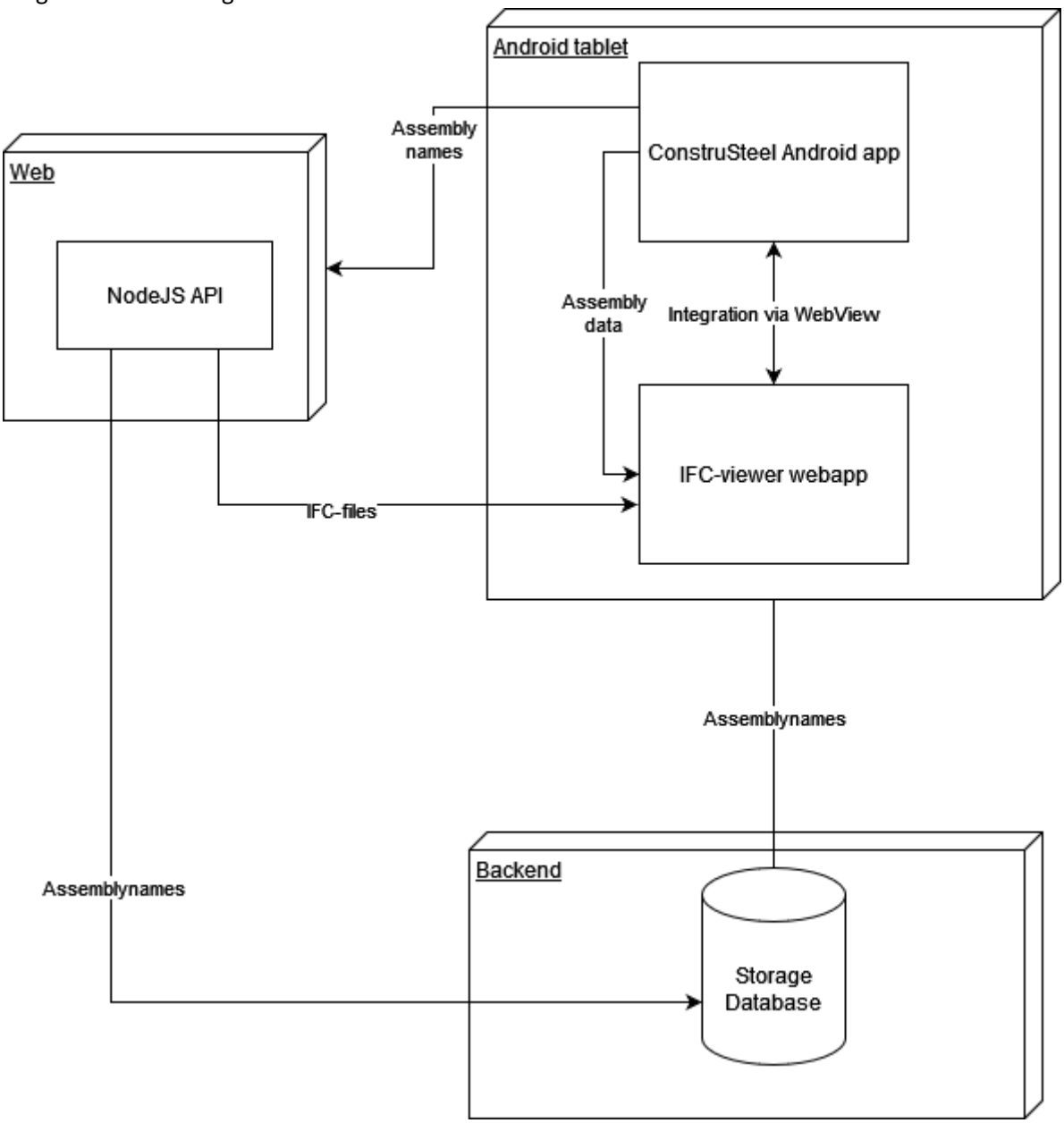


Figuur 10: Het sequence diagram voor de webapp

De uiteindelijke structuur is dat er een web front-end, geïntegreerd in Android, communiceert met de C# API. De API krijgt een aantal bestanden aangeleverd uit IIS. Om het te testen is gekozen om ook het IFC-bestand statisch aan te leveren. Dit wordt uiteindelijk dynamisch geregeld, maar omdat de performance getest moet worden wordt dat nu vanuit IIS statisch aangeleverd.

In Bijlage 6. Het Functionele Design (Adam, 2022) vanaf hoofdstuk 3.1.5 wordt dieper ingegaan op het uiteindelijke systeem design. Er is een wijziging in het plan geweest waardoor het onderste

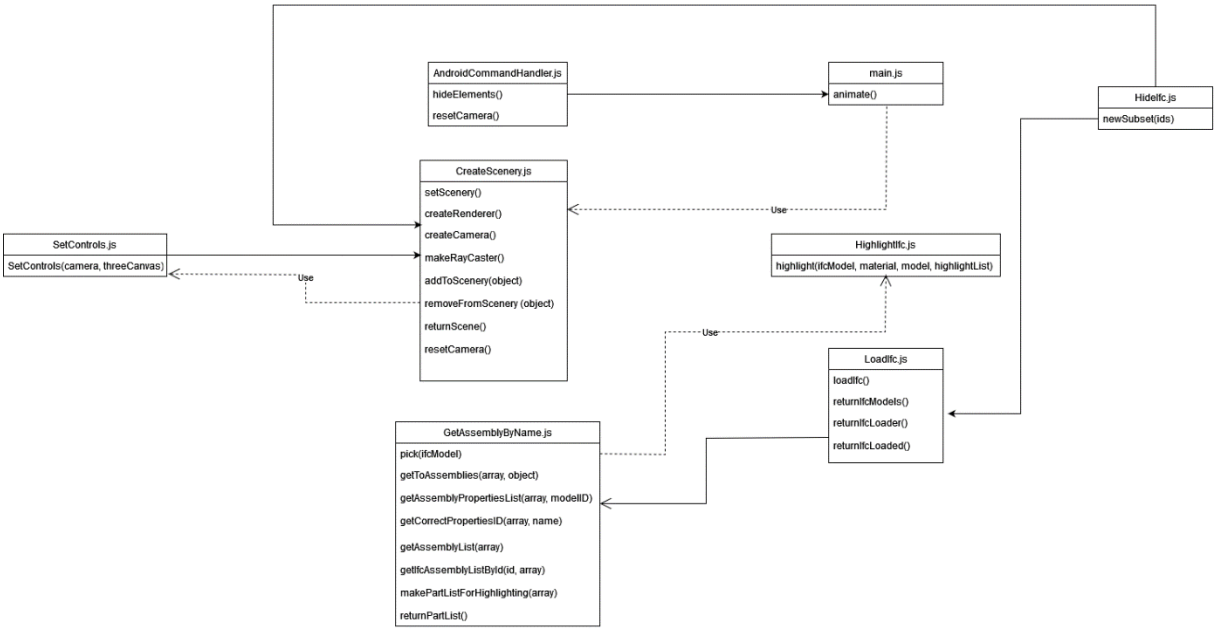
diagram tot stand is gekomen:



Figuur 11: Het uiteindelijke systeem

7.4.5 Dependency diagram

Dit diagram (zie figuur 12) toont de dependencies van alle geprogrammeerde modules aan. Deze modules praten met elkaar als op een object georiënteerde wijze. De keuze om met ES6 op deze manier te programmeren is duidelijkheid, leesbaarheid en het feit dat er niet 500 regels aan code in slechts één file staan.



Figuur 12: Het dependency diagram

Deze afbeelding geeft met pijlen aan welke classes waar afhankelijk van zijn. Er is een circulaire dependency (die benodigd was voor de werking van het product) maar voor nu zijn HighlightIfc.js, LoadIfc.js en GetAssemblyByName.js afhankelijk van elkaar. (Bijlage 6. Het functionele design (Adam, 2022))

7.5 Het UX/UI-design onderzoek

Voor het volledige UX/UI-onderzoek en alle details daarbij, wordt er verwezen naar Bijlage 2, vanaf hoofdstuk 5.

Er is gebruik gemaakt van de Design thinking fase. Deze fase heeft vijf iteratieve fasen om onderzoek mee te doen. Beginnende met de emphasize fase. Deze fasen helpen een probleem inzichtelijk te krijgen en er op een creatieve wijze oplossingen op te bedenken. Zo kunnen grote problemen inzichtelijk gemaakt worden en opgelost worden.

7.5.1 De Emphasize fase

“De eerste fase van het Design Thinking proces is het verkrijgen van een empathisch begrip van het probleem dat je probeert op te lossen. Dit omvat het raadplegen van experts om meer te weten te komen over het onderwerp. Ook door gesprekken te voeren om mensen te begrijpen, en om jezelf onder te dompelen in de fysieke (werk)omgeving, zodat je een dieper persoonlijk begrip van de betrokken problemen kunt krijgen." (Friis Dam, 2021)

Om het onderzoek te starten moet dus eerst het probleem onderzocht en begrepen worden. Hiervoor zijn meerdere media geraadpleegd, waaronder het interview met de pilot klant Jac Reijns Staalconstructie.

Bij Jac Reijns Staalconstructie is vanuit het altijd willen innoveren en vooruitlopen met machines en techniek ontstaan dat ze van de oude werkwijze wilden afstappen. Deze werkwijze was de papieren productie. Om te innoveren op de werkplaats, werd er besloten om tablets en computers de werkplaats in te doen. Met de tablets kan men gebruik maken van de Digitale Werkplaats app.

Alle werkbatches en tekeningen werden op papier uitgeprint en in grote mappen gestopt. Daarbij kwam een stempel lijstje dat altijd “ergens” te vinden was. Deze lijsten waren van belang omdat daar

alles op werd afgestempeld als iets af was. Nu is ConstruSteel er, en de werk chef geeft de fabrieksmedewerkers een tablet. Daar gebeurt alles centraal op en dat werkt een stuk makkelijker.

Op deze manier hopen ze bij Jac Reijns Staalconstructie dat de informatiestroom wat minder wordt naar de fabrieksmedewerker toe. Op het moment “verdrinken” ze in het papier en in de daarbij behorende informatie. Maar het doel is om naar de paperless production toe te gaan, en naar de situatie waar de fabrieksmedewerker alleen die informatie heeft die die nodig heeft.

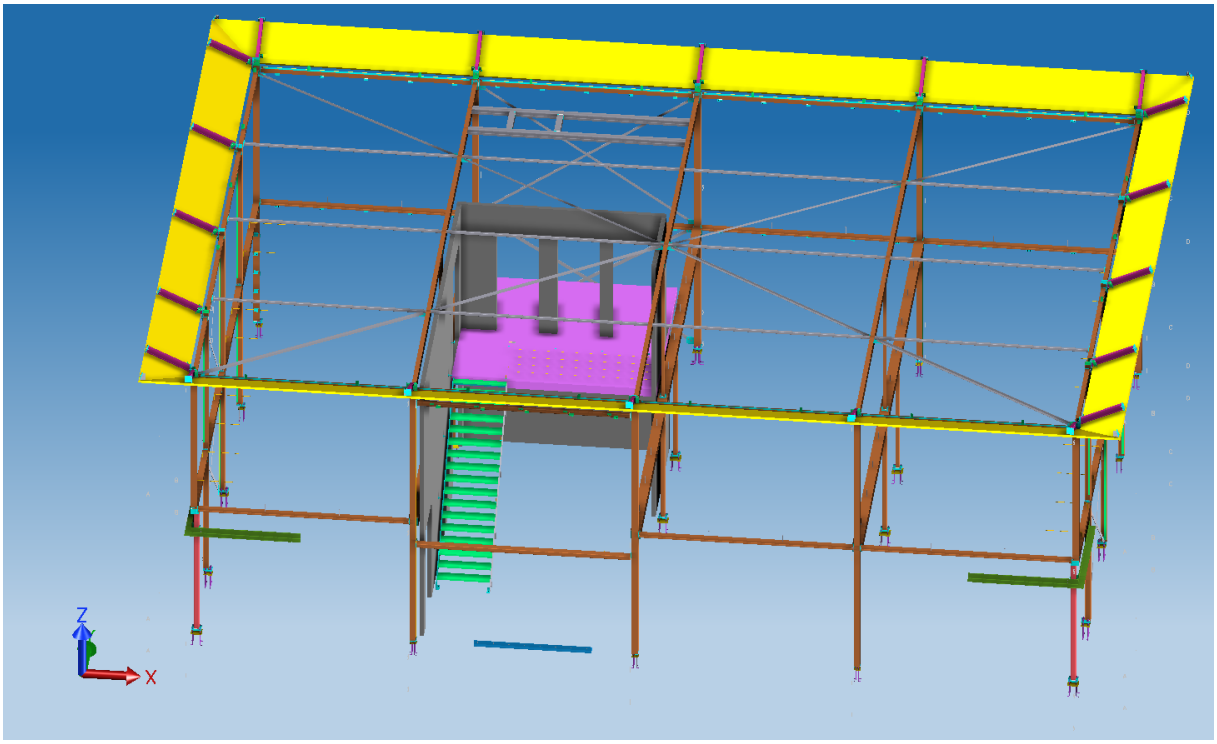
Er wordt duidelijk aangegeven dat pdf al een hele verbetering is ten opzichte van papier.

“Wat bij geprinte versies nog wel eens discutabel is – waar gaat dat lijntje naartoe? – is dat bij pdf duidelijk te zien” (De Swart, 2021)

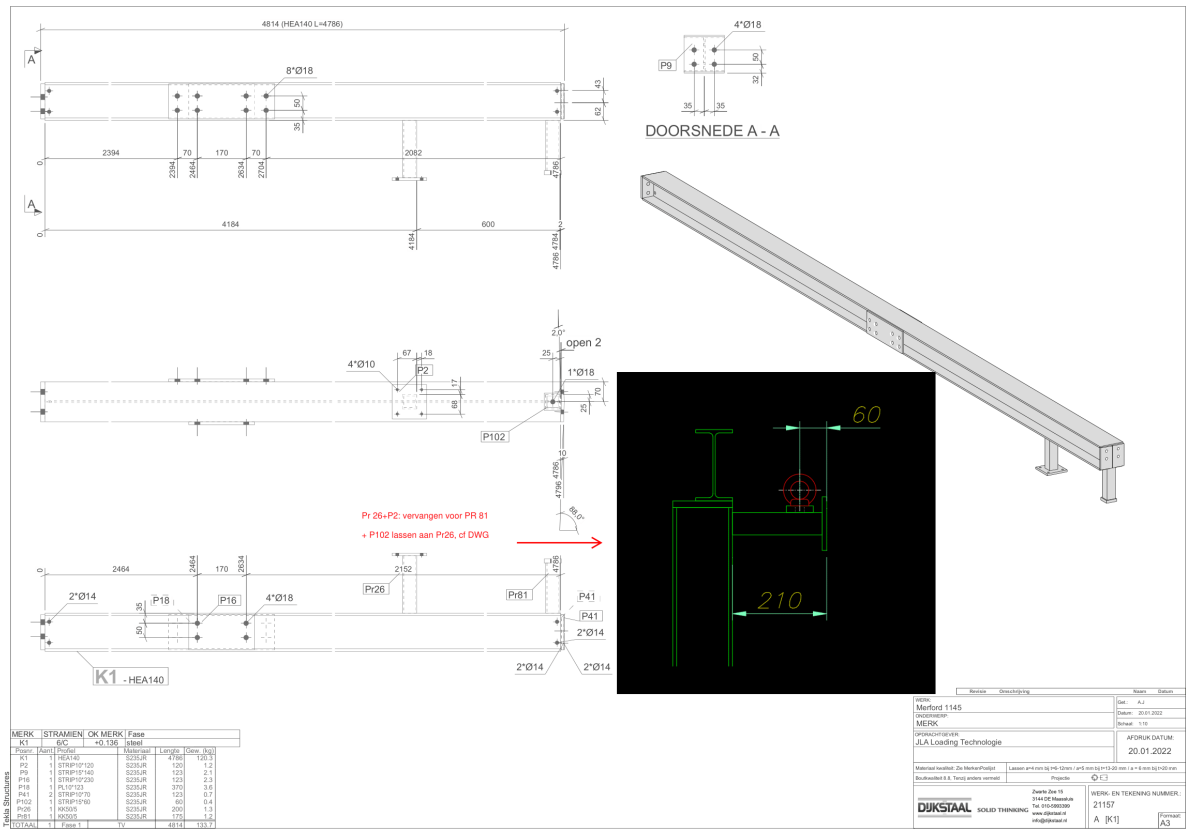
(Bron interview: (Westenenk, 2021))

Deze casus geeft aan dat het al een hele verbetering zal zijn ten opzichte van de papieren productiemethode. De 2D-visualisatie is al een stuk beter. Maar soms nog niet helemaal wat het moet zijn, en de vraag naar 3D-visualisatie wordt steeds groter.

Een 2D-model laat net wat onderdelen niet zien die een 3D-model wel zou kunnen laten zien. De volgende afbeeldingen zijn hier een goed voorbeeld van:



Figuur 13: Bird's Eye View, dit toont de voordelen van een 3D-visualisatie naast een 2D-tekening.



Figuur 14: Een 2D-view van een staal-model.

Figuur 14 is een 2D-model. Dit geeft weinig perspectief, omdat er alleen lijntjes worden weergegeven. De technische details staan er ook op, maar de 3D-view zou moeten komen als aanvulling. Op deze manier kunnen contextuele details nog missen.

Als je nou net even iets naar voren zou kunnen draaien, zodat er perspectief ontstaat, zoals in Figuur 13. Dan is ineens een stuk duidelijker hoe alle onderdelen op de plaats vallen. Klanten kunnen hier betere productie uit halen. Minder faalkosten en meer kwaliteit is het uiteindelijke doel van ieder bedrijf.

7.5.2 De Define fase

“Tijdens de Define-fase stel je de informatie samen die je hebt verzameld tijdens de Empathise-fase. Hier analyseert u uw observaties en synthetiseert u ze om de kernproblemen te definiëren die u en uw team tot nu toe hebben vastgesteld. Je moet proberen het probleem op een mensgerichte manier te definiëren als een probleemstelling.” (Friis Dam, 2021)

In deze fase van het Design Thinking proces (Bijlage 2 Software Design (Adam, 2022) is het de bedoeling om het probleem concreet en begrijpelijk te definiëren. Met behulp van de verzamelde data in de emphasize fase is dit nu mogelijk. Alles is meegenomen in het vraagstuk en daar is een duidelijk probleem naar voren gekomen dat betrekking heeft tot een deel van de staalbouwbranche.

Gebruiker	Behoefte	Inzicht
Fabrieksmedewerker	De fabrieksmedewerker moet duidelijk kunnen zien wat hij moet lassen/samenstellen.	Voor bepaalde bewerkingen is een 2D-view niet handig, deze bewerkingen moeten vanaf meerdere kanten bekeken worden. 2D-visualisatie maakt

		een complexe tekening onduidelijk.
Werk chef/productiemanager	De werk chef/productiemanager wil graag een vloeiend werkproces zien wat niet te lang duurt.	Met de digitale werkplaats is dit al verbeterd, maar als een medewerker duidelijk kan zien hoe een constructie opgebouwd is kan dit nog beter bevorderd worden en kan kwaliteit nog beter bewaakt worden.
Werkvoorbereider	De werkvoorbereider wil IFC-bestanden aan een werkbatch koppelen.	Zodra de 3D-viewer er is, kan dit gedaan worden en dan kan het IFC-bestand ook daadwerkelijk ingelezen worden. Dit bevordert de werksnelheid en de algehele vermindering van faalkosten.

Dit zijn de behoeften die tot nu toe vastgesteld zijn per actoren en de inzichten daarop. Het komt erop neer dat er veel vraag is naar een 3D-viewer omdat dat de productie goed kan bevorderen. Men ziet nu nog wel eens te veel onduidelijkheden, ook al is het al een stuk beter gevisualiseerd. Met een 3D-tekening kun je sowieso zien waar dat ene schuine lijntje heen gaat of kun je zien waar een uitsteeksel te vinden is.

De conclusie die hier getrokken kan worden is dat een 3D-viewer zeer gewenst is onder de klantenbasis van ConstruSteel om de volgende redenen:

- Kwaliteitsbevordering
    - Hoe duidelijker een tekening, hoe makkelijker een merk- of poswerkbatch in elkaar gezet kan worden
- Kostenvermindering
    - Doordat de tekeningen nog duidelijker zijn en nog steeds zonder zeeën van papier, is het mogelijk om kostenvermindering te realiseren.
    - De duidelijkere tekeningen, en de duidelijkere context in het 3D-model zorgenvoor minder faalkosten.
- Sneller werkproces
    - De 3D-viewer kan het werkproces versnellen doordat het duidelijker is wat er gedaan moet worden

Het huidige probleem is dus voornamelijk duidelijkheid. Waar zit wat en hoe zit een staalconstructie echt in elkaar? Is het ook handig om te weten welke context het onderdeel heeft wat iemand aan het bouwen is? Dat antwoord is simepweg: ja. De context is zeer belangrijk. Dit kan kwaliteit bevorderen en daarom vraagt men ook om deze 3D-viewer.

Om dit te realiseren moeten in de ideate fase en de prototype fase meegenomen worden dat de volgende design implementaties plaats gaan vinden:

- Er wordt een webapp gemaakt;
- De webapp wordt geïntegreerd in Android;
- UX/UI wordt zo strak en laagdrempelig mogelijk gehouden;
- Er wordt gebruik gemaakt van Material design;

- Het scherm moet, op de 3D-viewer na, zo leeg mogelijk;
- Betrek intern de collega's bij de ideate fase;

7.5.3 De Ideate fase

"Tijdens de derde fase van het Design Thinking-proces zijn ontwerpers klaar om ideeën te genereren. Je bent je gebruikers/klanten in de Emphasize-fase gaan begrijpen, en je hebt je observaties in de Define-fase geanalyseerd en gesynthetiseerd. Je eindigde met een mensgerichte probleemstelling. Met deze solide achtergrond kunnen jij en je teamleden beginnen met "out of the box denken" om nieuwe oplossingen te vinden voor de probleemstelling die je hebt gemaakt, en je kunt op zoek gaan naar alternatieve manieren om het probleem te bekijken." (Friis Dam, 2021)

In deze fase zijn alle belangrijke ideeën verzameld. De ideeën zijn grotendeels vanuit de bestaande Digitale Werkplaats gekomen, maar ook zijn er ideeën van collega's genoteerd (zie "Adam, Bijlage 2. Software Design, hoofdstuk 5.2.3 'De Ideate Fase', 2022"). Op deze manier konden de beste ideeën eruit gehaald worden om zo uiteindelijk de prototypes te maken en de prototypes te testen.

7.5.4 De Prototype fase

"Het ontwerpteam zal nu een aantal goedkope, verkleinde versies van het product of van specifieke functies in het product produceren, zodat ze de probleemoplossingen kunnen onderzoeken die in de vorige fase zijn gemaakt. Prototypes kunnen worden gedeeld en getest binnen het team zelf, op andere afdelingen, of op een kleine groep mensen buiten het ontwerpteam. Dit is een experimentele fase en het doel is om de best mogelijke oplossing te vinden voor elk van de problemen die tijdens de eerste drie fasen zijn geïdentificeerd." (Friis Dam, 2021)

In dit hoofdstuk zijn alle LoFi en HiFi prototypen gemaakt (zie Bijlage 2, "Adam, Bijlage 2. Software Design, hoofdstuk 5.2.4 'De Prototype Fase', 2022"). Alle ideeën uit de ideate fase zijn opgehaald en hierin gestopt. Er zijn meerdere iteraties op dit design. Dit komt omdat de test fase iedere keer wat belangrijke feedback met zich meebracht. En iedere keer als een prototype was doorgenomen, kon er terug geïtereerd worden naar de fase ervoor. De feedback werd verwerkt en de Test fase ging weer van start.

De prototypen werden nadat ze gemaakt waren getoond en gestest. André en Pascal hebben de designs van feedback voorzien, en daarmee is het design geworden zoals het in de final app zou worden.

7.5.5 De Test fase

"Ontwerpers of beoordelaars testen het volledige product grondig met behulp van de beste oplossingen die tijdens de prototypefase zijn geïdentificeerd." (Friis Dam, 2021)

De prototypen zijn drie keer gereviewed, getest en voorzien van feedback. De resultaten (Zie "Adam, Bijlage 2. Software Design, hoofdstuk 5.2.5 'De Test Fase', 2022") toonden telkens weer verbeterpunten aan. Dit is gedaan om het precies goede design door te voeren. Het moet worden zoals het in de fabriek verwacht wordt.

De belangrijkste feedback kwam na het eerste HiFi design. De 3D-viewer moest als aanvulling op de PDF-viewer komen. Deze feedback is meegenomen en verwerkt. Daarnaast moesten de web-design elementen vervangen worden met Material Design. Dit is de standaard die al gebruikt wordt in de app en dat zijn de fabrieksmedewerkers al gewend. Op UX-gebied is het dus van belang dat de webapp voorzien wordt van material design.

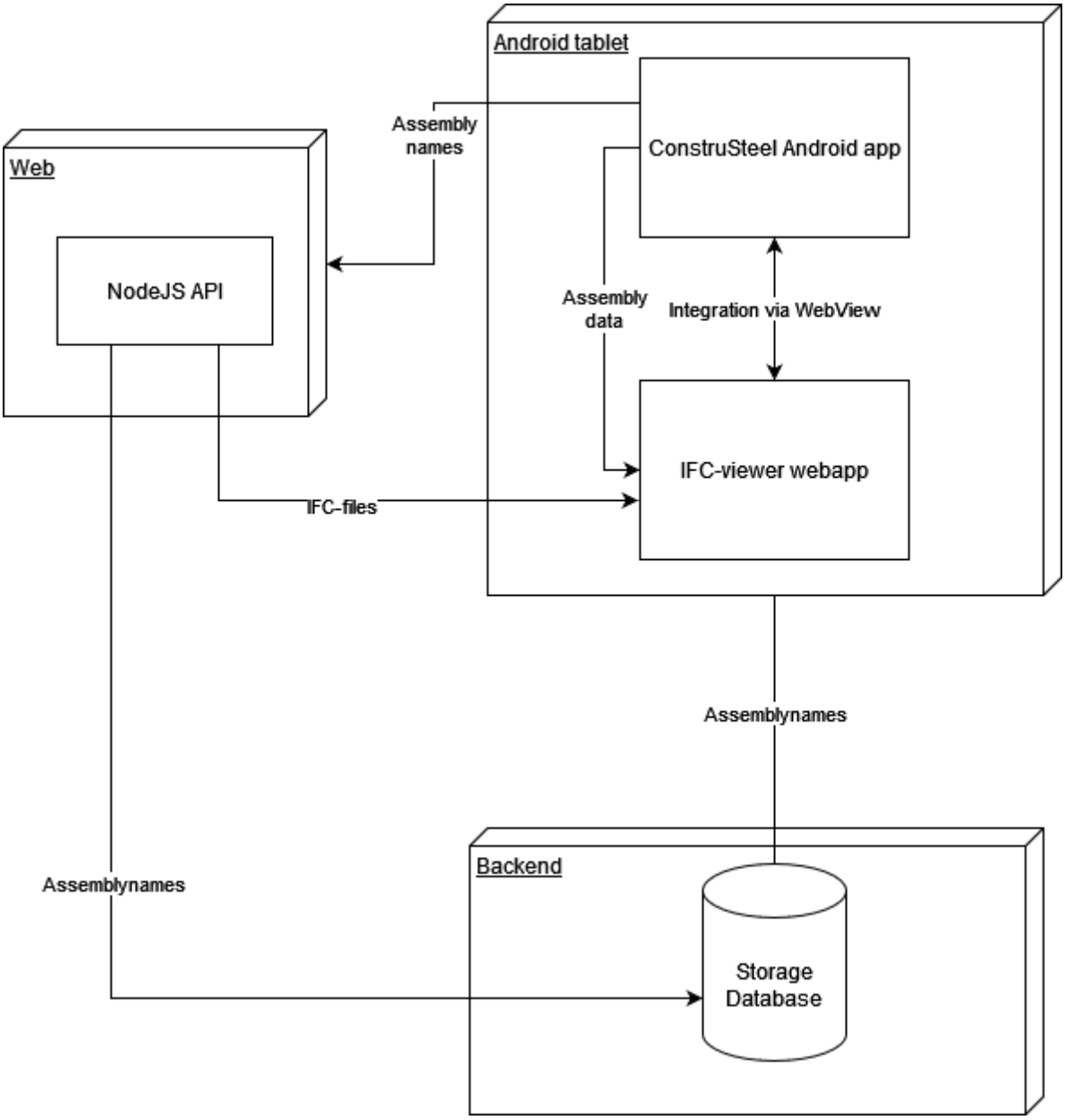
Figuur 15: De 3D-viewer na de derde feedback sessie

In deze afbeelding is goed het gebruik van Material Design te zien. De Speed Dial die de opties laat zien. De knop rechts in de hoek is voor het gereed melden van het merk. Dit is een scherm zoals die in de Android app komt te staan.

8. De implementatie fase

De implementatiefase is de fase waar echt met het product aan de slag is gegaan. In deze hoofdstukken worden de implementatie keuzes en bijzonderheden toegelicht. Daarnaast wordt er beschreven hoe tot een doel is gekomen. Dit wordt gedaan voor front-end, backend en voor de Android app.

8.1 Het initiële systeem design



Figuur 16: Het deployment diagram van het proof of concept

In Figuur 16 is het uiteindelijk gerealiseerde ontwerp te zien. Dit diagram geeft precies aan welke componenten met welke samenwerken en waar code en aanvulling op code is gemaakt. Het front-end bestaat uit de webapp en de Android applicatie. Deze praten beide tegen de NodeJS API aan en de NodeJS-API praat met een database.

De database is gekomen om merknamen en projectnamen op te slaan. Deze namen zijn belang omdat dit de hoofdvorm van communicatie tussen de Android- en webapp zijn. Deze communicatie zorgt ervoor dat de app kan wisselen tussen twee projecten waar nodig.

Als een merknaam bij een bepaald project hoort, dan zal het IFC-model van dat project ingeladen worden. En als er tussen project gewisseld wordt, dan wordt het IFC-model dat bij dat project hoort aan de hand van de projectnaam opgevraagd.

8.2 De JavaScript front-end

Voor het diagram van de JavaScript front-end: Zie figuur 16.

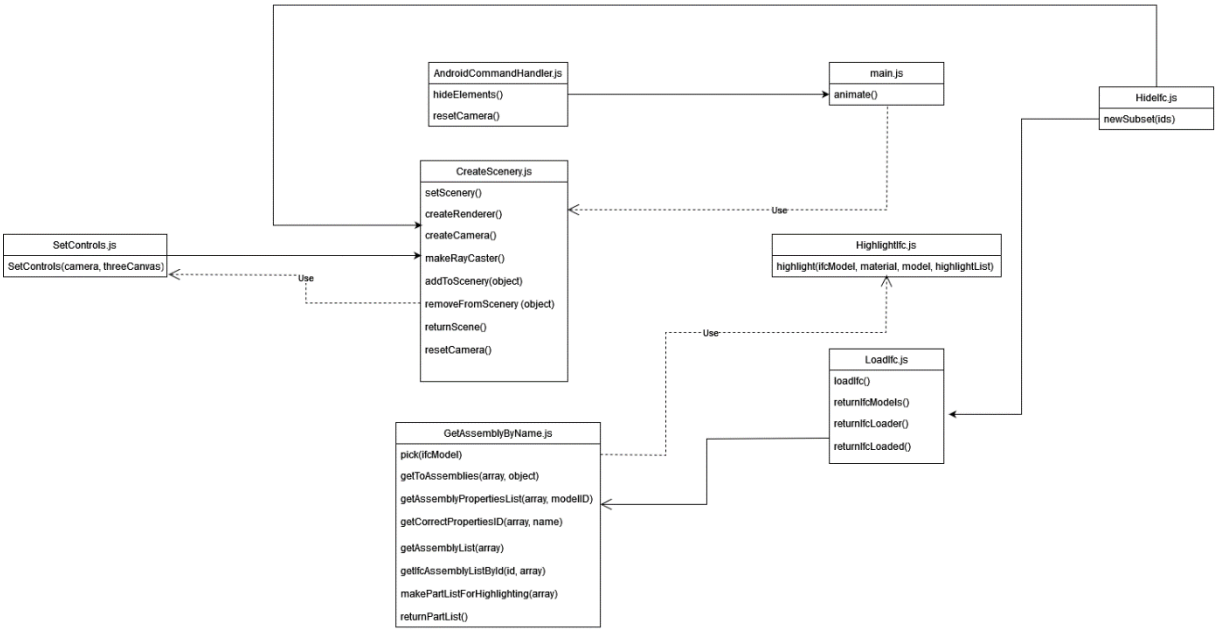
Na alle onderzoeken is het belangrijk dat de JavaScript front-end opgezet wordt. De gekozen technieken (plain JavaScript, Three.js en IFC.js). Deze technieken zijn gekozen, omdat er in hoofdstuk 5.3 naar voren kwam dat IFC.js de beste open source IFC-viewer is op het moment. Daarnaast is de documentatie compleet. De community die deze technieken beschikbaar stelt is behulpzaam en reageren adequaat. Ook voor de organisatie is het van belang voor toekomstig onderhoud dat ze kunnen terugvallen op deze community.

Ook een belangrijke factor in de keuze voor JavaScript met Three.js is dat het een snelle 3D-engine is. Op het gebied van performance is Three.js interessant en zal het de browser niet onnodig vertragen. Er zouden met Three.js browser games gemaakt kunnen worden die op goede snelheden performen. Deze games zijn over het algemeen zwaarder dan de komende IFC-viewer.

Kort samengevat: de criteria voor het kiezen van dit drietal technieken zijn:

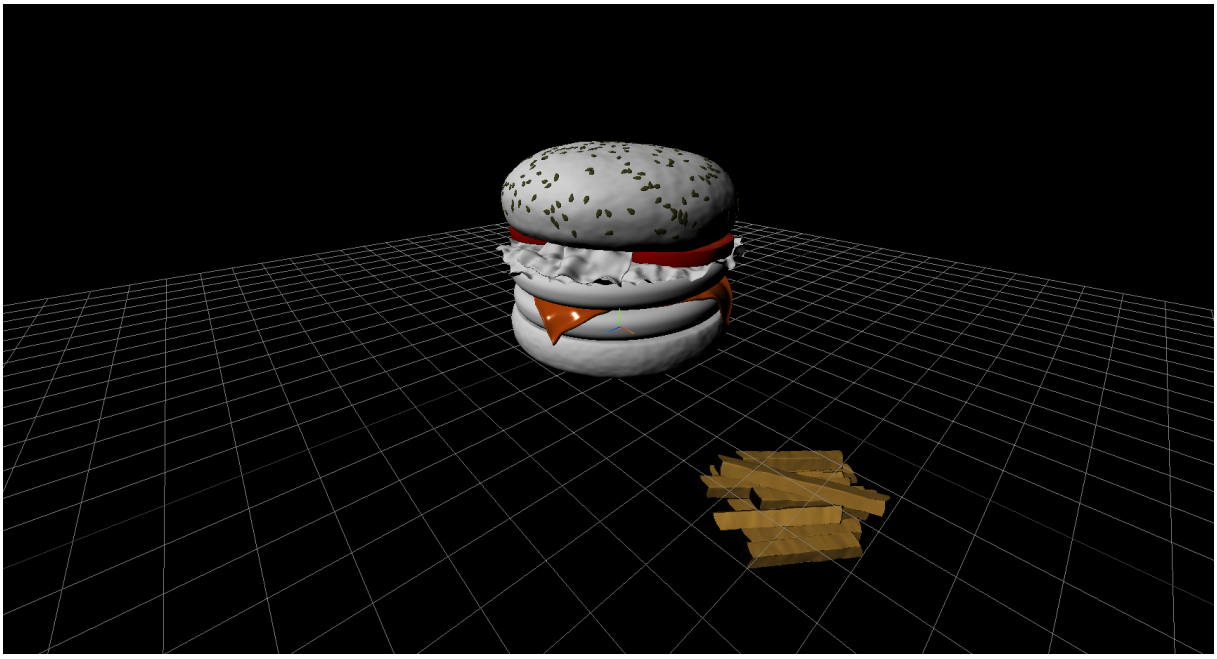
- Bekendheid, er is een sterke community;
- Compleetheid, alles is af en daarmee ook compleet gedocumenteerd;
- Het is het meest lichtgewicht framework dat er is. Samen met IFC.js;
- Ervaring met JavaScript en Three.js;
- IFC.js is een toevoeging op Three.js, en daarmee lichtgewicht en makkelijk te gebruiken;
- IFC.js is de officiële Three.js IFC-library. Support is groot;

Uit deze criteria is een class-diagram gekomen, dit class diagram was een concept diagram dat later is aangehouden. De classes die bijgekomen zijn, zijn erin gevoegd voor compleetheid, omdat sommige classes voor structuur benodigd waren.



Figuur 17: Het class diagram van de webapplicatie.

Een eerste opzet van een prototype in Three.js was het vertonen van een OBJ-bestand. In dit geval dat van een hamburger. Daardoor kon er ervaring op gedaan worden met de 3D-engine van Three.js. De licht engines, de scène die het maakt en hoe het 3D-model getoond kan worden. Het gevoel en hoe het er uit ziet is al precies zoals de standaard is.



Figuur 18: De eerste vertoning van een 3D-model in Three.js

Nadat het Three.js aspect onder de knie was, was het natuurlijk de bedoeling om over te gaan op IFC.js en de bijbehorende IFC-modellen. Dit gaf in het begin nog wat moeilijkheden, omdat de Rollup.js library niet mee wilde werken. Rollup.js is een library die de bestanden bundelt. Alleen de bestanden en de code die gebruikt worden, worden meegenomen in het bundle.js bestand. Als dit niet gebruikt wordt, is IFC.js te groot om in de browser te laden. Doordat Rollup.js gebruik maakt van Tree shaking werd de code weer uitvoerbaar en kon de applicatie geladen worden.

Rollup.js bracht wel een mild obstakel met zich mee. De library maakt gebruik van een klasse die helpt bij het builden van asynchrone functies in JavaScript. Deze functie zou moeten zorgen dat het grootste deel van IFC.js door de bundler van Rollup.js uitgevoerd moet kunnen worden, maar door een onvoorziene fout kon het niet gebundeld worden en kwam hij met de error “RuntimeGenerator has not been defined”. De RuntimeGenerator helpt dus bij het maken van de asynchrone functies in het project. Als deze functie niet gaat werken, dan kan er weinig gedaan worden met IFC.js, omdat IFC.js veel gebruik maakt van asynchrone functies.

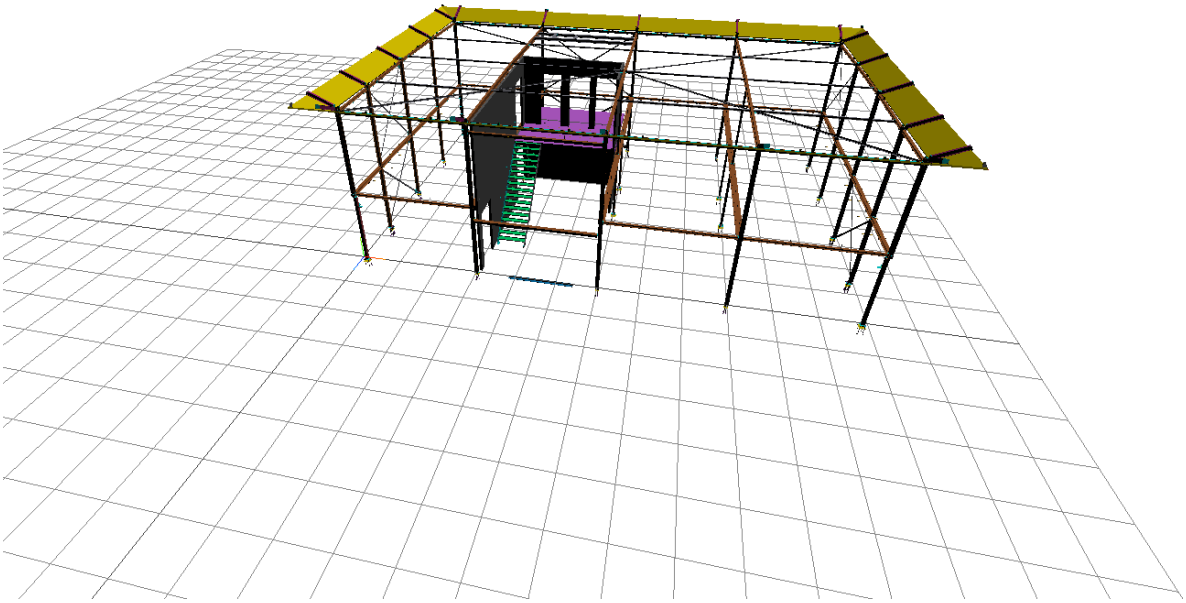
```
"browserslist": [
  "last 2 Chrome versions"
],
```

Figuur 19: De code die de RuntimeGenerator error oploste

Dit stukje code zorgde ervoor dat Rollup.js niet meer crashte op de RuntimeGenerator. De RuntimeGenerator is het stukje code wat async en await code van ES6 genereert waar dat nodig is in de code. Zonder te specificeren dat het in deze browsers moet werken kan de RuntimeGenerator niet draaien.

Er werd nadat de hobbel in Figuur 19 was overkomen een NodeJS server opgezet. Deze server zorgt ervoor dat de WASM (benodigde bestanden voor het parsen van IFC) geserveerd werden aan de webapp.

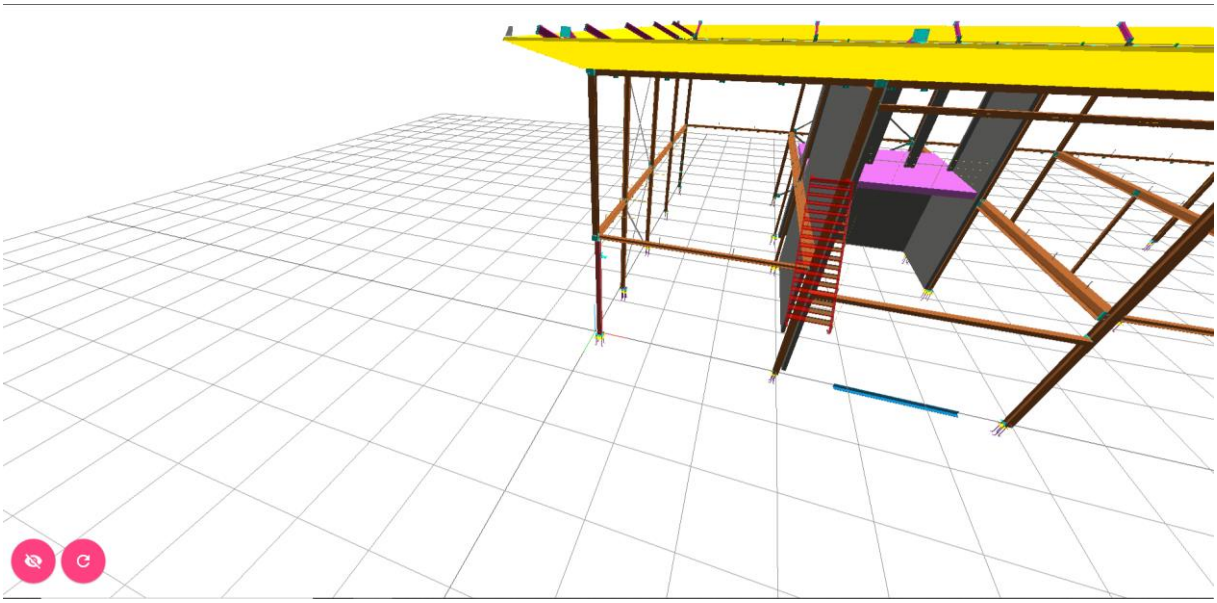
Daarna kon het eerste IFC-bestand ingeladen worden:



Figuur 20: Het IFC-bestand van het model van het ConstruSteel kantoor

Nadat het IFC-bestanden inladen was gelukt, wordt er gekeken hoe het markeren verzorgd kan worden. Om dit markeren te kunnen toepassen, hebben we onderzocht welk element tag uit het IFC

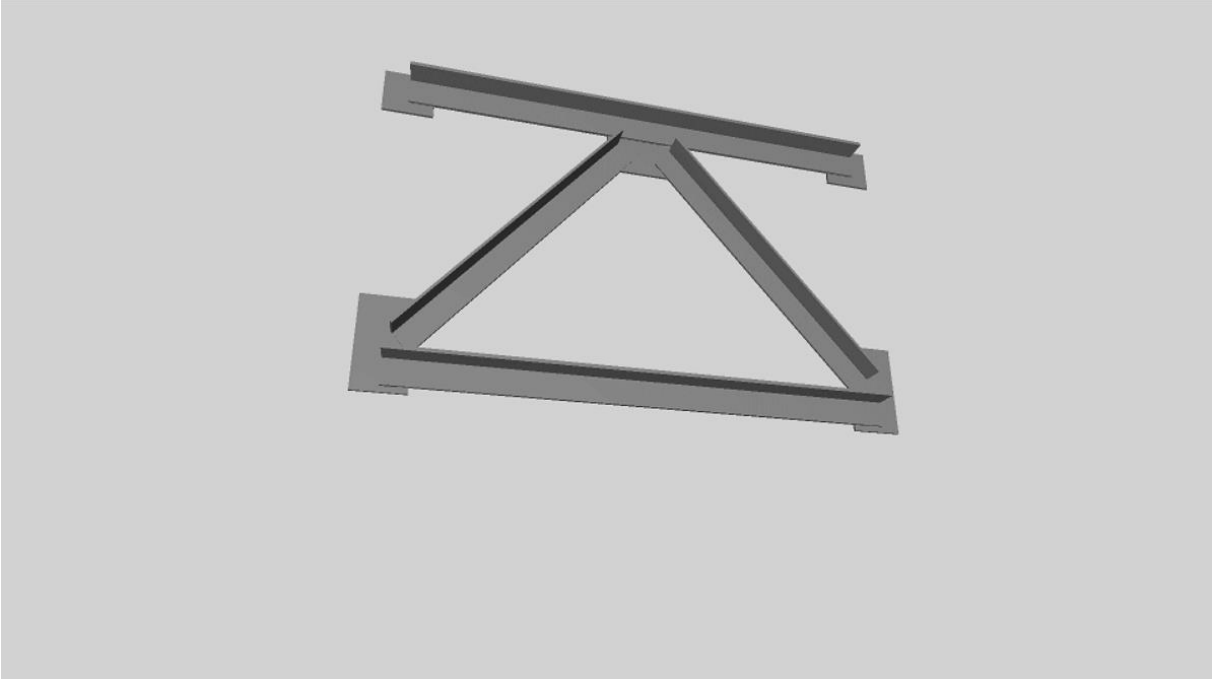
bestand hiervoor geschikt was. Vanuit documentatie en IFC bestand onderzoek zijn we uitgekomen op de “IFCELEMENTASSEMBLY”. In de documentatie van BuildingSMART staat: "De IfcElementAssembly vertegenwoordigt complexe elementenassemblages die zijn samengevoegd uit verschillende elementen, zoals discrete elementen, bouwelementen of andere elementen.



Figuur 21: Een gemarkeerd IFC-element. De trap is gemarkeerd in het rood.

Als voorbeeld: Staalconstructieassemblages, zoals spanten en verschillende soorten frames, kunnen worden weergegeven door de entiteit IfcElementAssembly. Andere voorbeelden zijn plaatvelden die zijn samengesteld uit een aantal prefab betonplaten of wapeningseenheden gemaakt van verschillende wapeningsstaven. Ook badkamerunits, trapdelen en andere voor- of prefab elementen zijn voorbeelden van de algemene IfcElementAssembly entiteit" (BuildingSMART, z.d.)

Al de bovengeoemde voorbeelden kunnen als merk worden weergegeven. Als het binnen een IfcElementAssembly tag zit, dan kan het in een werkbatch gezet worden. Dit wordt in Tekla al gedefinieerd in de daarin gegenereerde merkenlijst. Met deze merkdata wordt ook het IFC-model gegenereerd. Zo kunnen merken als in Figuur 22 gemarkeerd en gegroepeerd worden.



Figuur 22: Een voorbeeld van een IFCELEMENTASSEMBLY

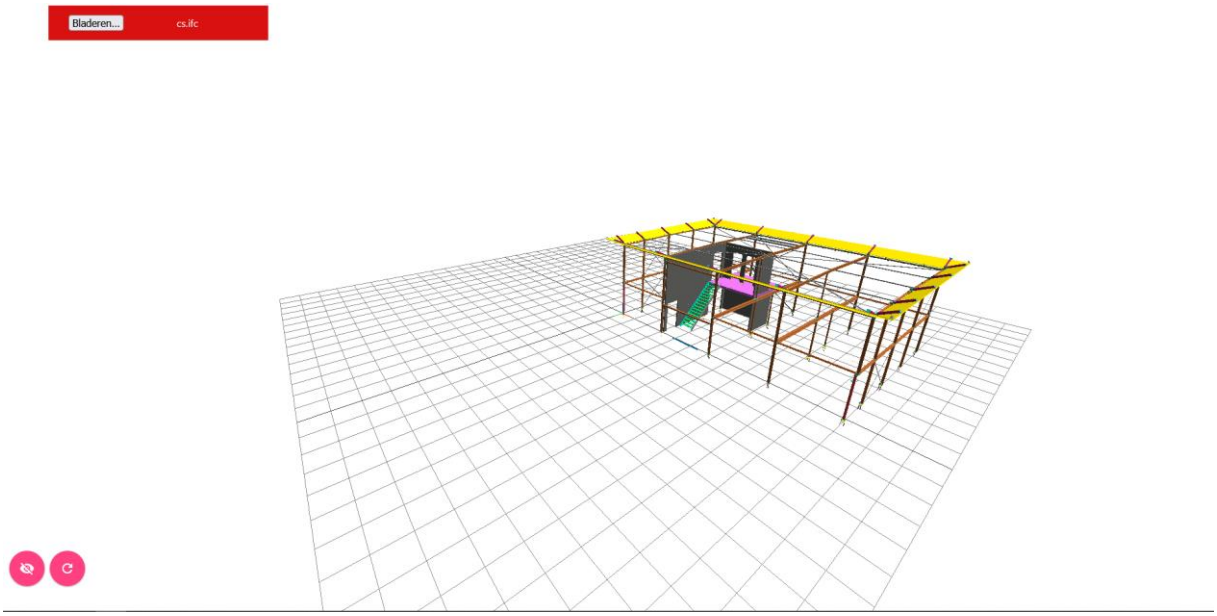
In Figuur 22 is een IFCELEMENTASSEMBLY te zien. Merken in de staalbouw hebben dezelfde constructie, het is een onderdeel bestaande uit meerdere kleinere onderdelen (posnummers). Op deze manier kan geconcludeerd worden dat deze data benodigd is om merken te kunnen markeren in het IFC-model.

Er is een loopfunctionaliteit gekomen die de data uit IFC-modellen filtert en klaarmaakt voor het markeren van IFC-modellen:

- Eerst wordt de rauwe IFC-data doorgegaan en wordt er geloopt tot er een IFC-tag tegengekomen wordt met de tag “IFCBUILDINGSTOREY”. Als deze er niet is, dan wordt er een lege lijst teruggegeven.
- Dan wordt er binnen de IFCBUILDINGSTOREY-tag alles met IFCELEMENTASSEMBLY opgehaald en opgeslagen in een lijst.
- Dan wordt alle element-data opgehaald uit de IFCELEMENTASSEMBLY-tags aan de hand van een IFC-ID genaamd de ExpressID.
- Als dit gedaan is kan er op de naam van het merk gezocht worden. Dit wordt gedaan door middel van op de IFC-tag te zoeken. Als de tag overeenkomt met de ontvangen merknaam wordt deze opgeslagen.
- Als de merknaam meerdere malen voorkomt in de lijst met IFCELEMENTASSEMBLY-tags wordt die meerdere malen opgeslagen.
- Alle child arrays van deze tags worden opgehaald. De ExpressID’s worden opgeslagen.
- Deze ExpressID’s worden meegegeven aan de markeringsmodule van de applicatie om gemarkeerd te worden.

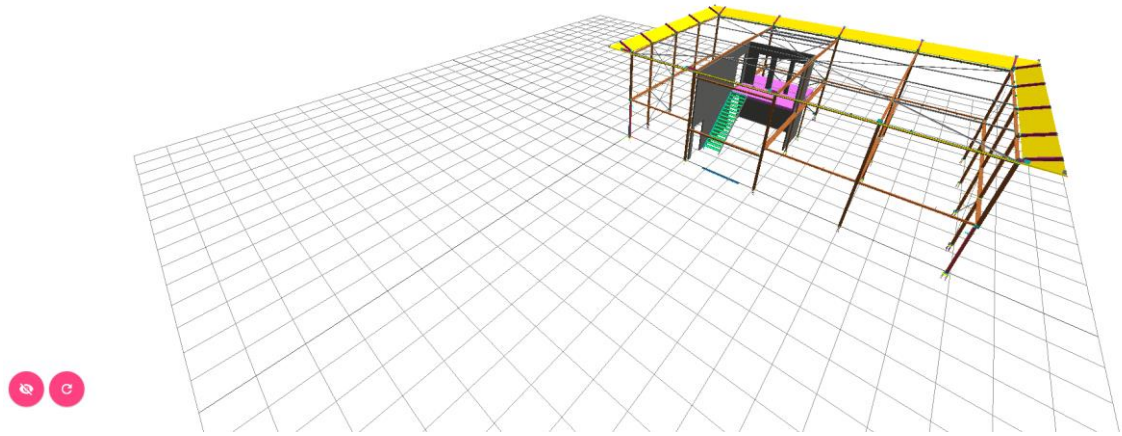
Dan is er nog de reset camera functie. De controls die Three.js aanmaakt kunnen opgeslagen en gereset worden. De functionaliteit om op te slaan wordt direct aan het begin aangeroepen en dan, als een medewerker de oriëntatie kwijt is in de omgeving kan hij de knop voor het resetten van een camera indrukken en de camera view wordt gereset naar het beginpunt.

De huidige IFC-viewer:



Figuur 23: De huidige looks en lay-out van de IFC-viewer

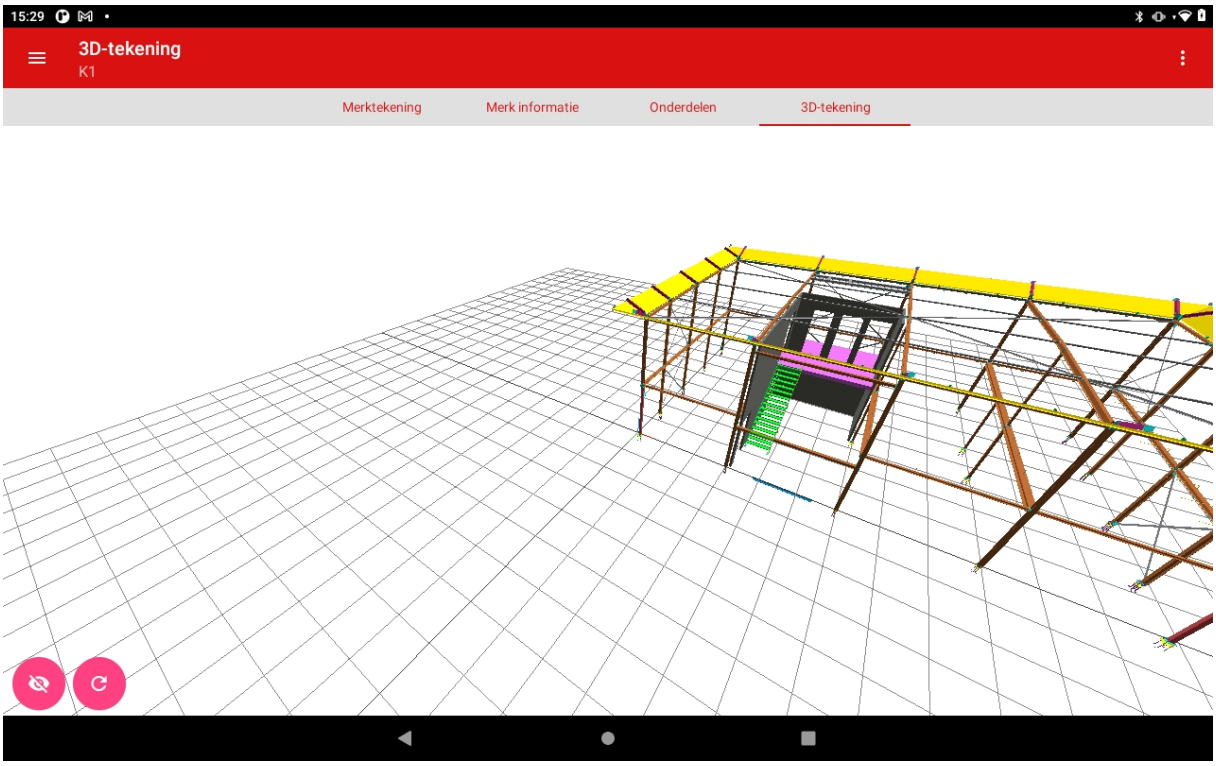
Na een iteratie op de lay-out, en de integratie van de backend ziet de IFC-viewer er nu zo uit:



Figuur 24: De uiteindelijke lay-out van de 3D-viewer

De lay-out is door het verwijderen van de bovenste rode balk weer iets schoner. De geselecteerde merken (vanuit de Android app) worden gemarkeerd en de knopjes zijn beide functioneel. Dit is de variant van de viewer die in Android komt. De backend levert dan succesvol het model aan en aan de hand van de Android view wordt de merknaam doorgestuurd.

De knoppen lijken op dit scherm klein, maar in werkelijkheid zijn ze niet zo heel klein. Deze screenshot was een test op desktop. Daarbij komt kijken dat het scherm groter is en dat de knoppen daardoor kleiner lijken. De knoppen zien er als volgt uit op Android:



Figuur 25: Een screenshot van de Android applicatie waar de knoppen weer groter lijken.

Door het kleinere scherm lijken de knoppen al een stuk groter te zijn.

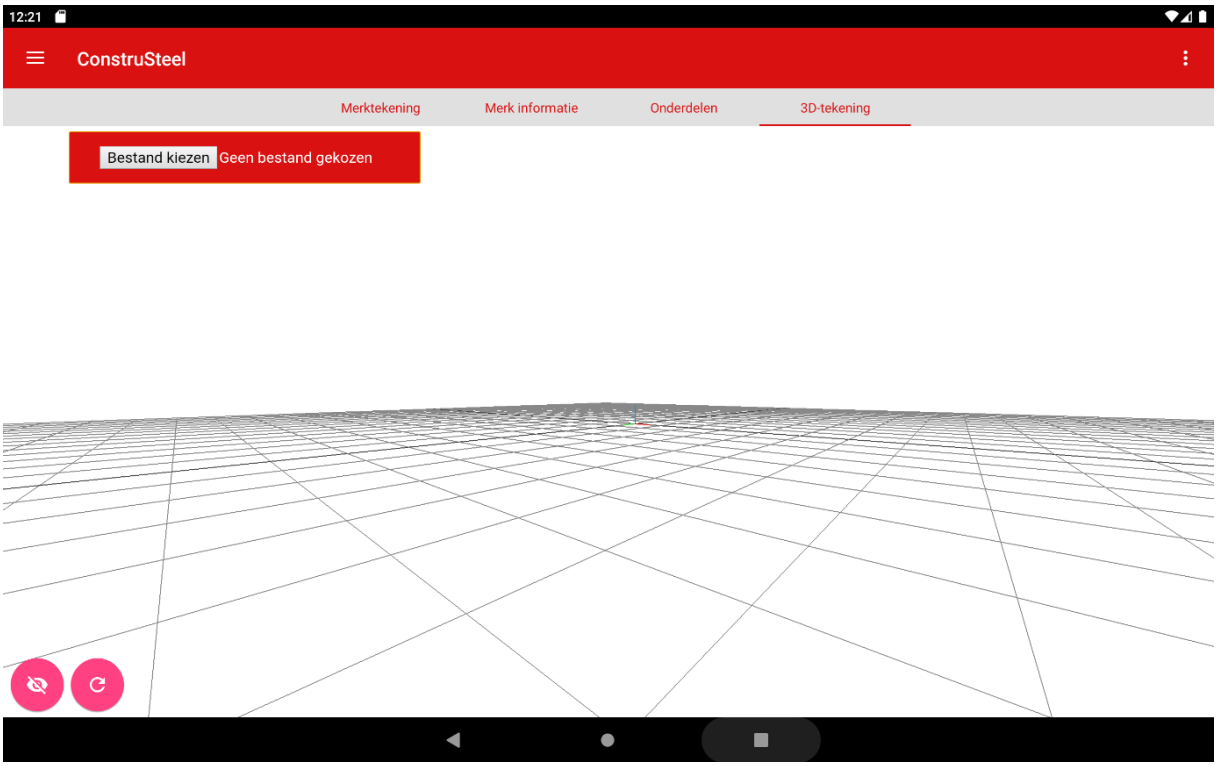
8.2.1 Wireframes

Voor de wireframes van de applicatie (die niet in Bijlage 2. Software Design staan) wordt er verwezen naar Bijlage 6. Het functionele ontwerp. Hier worden de technische wireframes getoond en toegelicht. Dit is ook het document waar alle diagrammen in beschreven en uitgelegd worden. De diagrammen zijn in dit document voor de compleetheid herhaald. Voor de latere wireframes wordt er gerefereerd naar Bijlage 2. Software Design (Adam, 2022).

8.3 De Android-applicatie

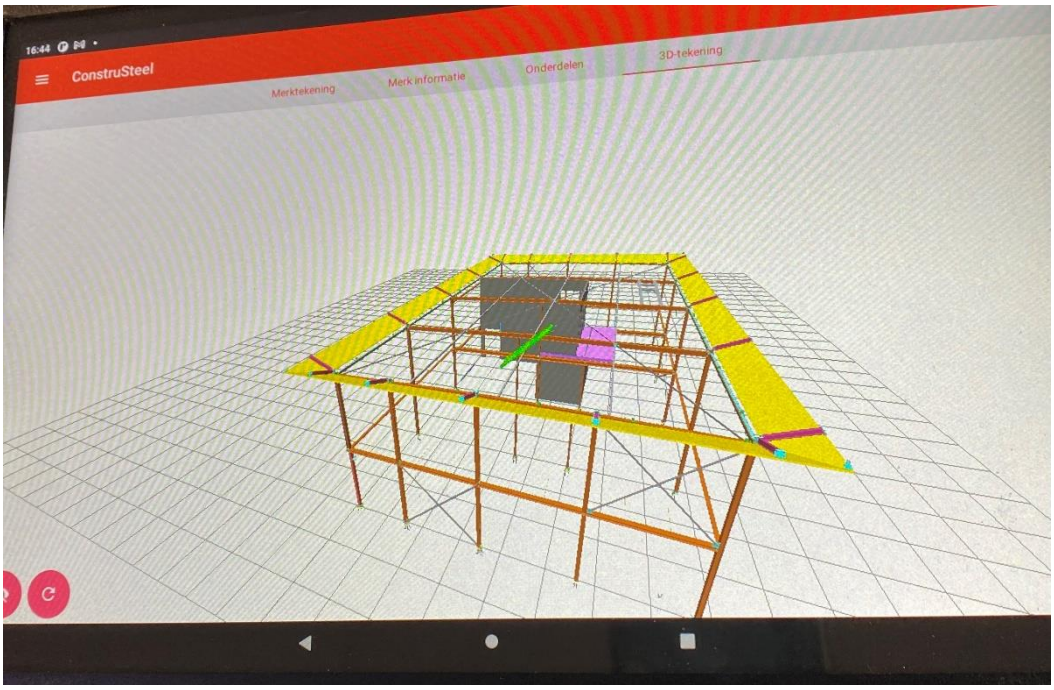
De Androidapplicatie bevat nu ook de 3D-viewer. Het enige wat het nog niet kan is het vertonen van IFC. Om dit te kunnen realiseren is er een WebView nodig geweest. Een Android WebView is een component dat een embedded browser in een activity realiseert. Dit is gemaakt voor meer grip op de lay-out. Dit is ook speciaal voor daarvoor ontwikkelde web-omgevingen.

Deze techniek is gebruikt om de 3D-viewer in IFC.js te kunnen draaien. Middels een localhost site via IIS is het tonen in Android gerealiseerd. Daardoor kan Android bij het juiste IP-adres met port komen. Met het IP-adres van de lokale pc, en port 7503 kan de applicatie de IFC-viewer vinden. Dat komt er dan zo uit te zien:



Figuur 26: De 3D-viewer in Android. Er komen nog wat lay-out aanpassingen.

Na een iteratie ziet het er als volgt uit:



Figuur 27: De uiteindelijke viewer zoals hij in Android komt (afbeelding gemaakt met de telefoon)

De bovenste balk is weg en het hele scherm wordt door de viewer opgenomen, waardoor er een duidelijke view van het model gemaakt wordt.

Zoals te zien is op het scherm wordt een volledig IFC-model geladen. Dit is de volledige constructie. De volledige constructie wordt per project ingeladen, en alleen de merken waarmee men aan het

werk is worden gemarkeerd. Er wordt geen apart model per merk in geladen, maar een model van een volledige constructie per project met merken die gemarkeerd worden.

8.4 De backend

De backend is uiteindelijk een proof of concept API geworden. Er bleek uiteindelijk dat een tekort aan C#-kennis de aanpassingen in de C#-API te veel maakten. Daardoor is er gekozen om een proof of concept API in NodeJS te maken die direct met de webapp praat. Deze backend haalt een bestand op, codeert deze met een base64 codering en stuurt het naar het front-end.

Als de gecodeerde tekst aankomt bij het front-end, decodeert de front-end het en maakt de front-end er een bestand van. Dit bestand wordt door de module “LoadIfc” ingelezen en het model wordt vertoond in de webapp.

Zoals in de API Definition staat (“Adam, Bijlage 3. API-definition, 2022”) is er maar één backend call. Dat voor het ophalen van de IFC-modellen. Dit is zo gekozen zodat het simpel en onderhoudbaar blijft. Dit is ook een logische keuze geweest omdat het simpel moet blijven voor de omzet naar de C#-omgeving.

De functionele documentatie is in ontwikkeling en zal ingaan op de functionaliteit van de applicatie zoals deze tot stand is gekomen.

9. De test fase

In de test fase wordt alles getest. De kwaliteit van de code, de UX/UI-designs, het gebruiksgemak van de webapp en Android app. Het is alleen niet mogelijk geweest om te testen met klanten. Er is dus gekozen om de tests intern te houden en daarop verder te itereren. Er is voornamelijk op de afdeling gereviewed met André en Pascal. De uitgevoerde reviews is feedback uitgekomen en die is zorgvuldig in tabellen geplaatst, zodat de feedback meegenomen kan worden in de volgende iteratie op de wireframe.

9.1 UX/UI tests (de test fase van design thinking)

"Ontwerpers of beoordelaars testen het volledige product grondig met behulp van de beste oplossingen die tijdens de prototypefase zijn geïdentificeerd." (Friis Dam, 2021)

Het is getest op toepassing en bediening. Daarmee invulling gegeven aan bijbehorende requirements. Voor het testplan wordt er naar Bijlage 8. Testplan + Testcases IFC-viewer en UX-UI tests (Adam, 2022) gerefereerd.

9.1.1 De initiële test voor het beste design

Het testen van de papieren prototypes wordt intern gedaan. In de eerste instantie worden deze prototypen bekeken, uitgelegd en daarna beoordeeld. Het idee is dat hierdoor bij collega's een favoriet naar voren komt. Een favoriet design. Dat wordt dan verwerkt in een low fidelity en later in een medium en high fidelity prototype.

De test zal in dit geval niets meer zijn dan laten zien en kijken wat er gezegd wordt, wat vinden de collega's van de designs? Zijn er vergeten elementen of kunnen er elementen weggelaten worden? Deze vragen zullen de hoofdrol spelen in deze test.

De test zal nu nog geen flow bevatten. Het is nu feedback krijgen van André (de productmanager) en Pascal (software engineer) en daarmee verder itereren op een design met flow. Dan is het de bedoeling om iemand, met minder kennis van de flow van deze uitbreiding, te laten testen en de feedback op te schrijven.

Design	**Feedback Pascal**	**Feedback André**
Het originele design	- Moet je de extra informatie wel willen? - Extra informatie komt over als verwarrend - Moeilijk leesbaar en lastig het model te zien - De medewerker ziet de context. De extra informatie zal de gebruiker niets opleveren.	- De detailview kan vragen oproepen bij medewerkers - Is het nodig? - Base information is ook niet nodig, informatie staat in ConstruSteel
Het verbeterde origineel	- De informatie in ConstruSteel moet leidend blijven	- Het wordt zo erg onduidelijk, vooral om dat de informatie in ConstruSteel nog wel eens afwijkt.
Alternatief 1	- Je moet je echt	- Met zo'n blokkerende

	afvragen of al die extra informatie wel nodig is	view op je scherm wordt je tekening en je model alsnog niet duidelijker.
Alternatief 2	- De tabs kunnen verwarrend worden	- Onduidelijk waar de informatie staat en vraag je af wat de gebruiker wil
Alternatief 3	- Deze infovertoning kan zeer goed werken - Kijken of dit ook op standaard IFC kan.	- Kijk voor een centreerknop, je kunt nog wel eens verward raken met het draaien van het model. - Kijk ook naar een knop die ervoor zorgt dat het model naar een zeker punt draait (zoals rechtsvoor en linksboven).
Alternatief 4	- Geen mooi alternatief, het ziet er technisch ook niet goed uit.	- Dit zal betekenen dat er meerdere IFCs ingeladen moeten worden.
Alternatief 5	- De split screen is erg onhandig op zo'n klein scherm als dat van een tablet. - Een knop voor het tonen van alleen een merknummer	- Alleen het geselecteerde merk tonen moet een optie zijn. - De functionaliteit om terug te gaan naar het begin punt te gaan is wenselijk. - De optie om de camera te centreren is ook wenselijk.

Tabel 4: De feedback op de papieren prototypes. Gegeven door André en Pascal

Uit deze feedback kan gehaald worden dat men het liefst geen informatie in het 3D-model wil. Het kan afwijken van de data die in ConstruSteel staat. Hierin is ConstruSteel leidend, omdat daar de werkvoorbereider alles correct heeft uitgewerkt en nagekeken.

Daarnaast zijn bepaalde knopjes ook handig. Zoals bv., een knop voor oriëntatie, om direct op een punt te kijken. Of een knop die hem automatisch van voren centreert. Allemaal functionaliteit om het bewegen om een 3D-model makkelijker te maken. Dit moet natuurlijk wel zo geïmplementeerd worden dat het geen belemmering is voor de user experience.

Er moet ook gekeken worden naar hoe 3D geïmplementeerd kan worden in programma's als Figma. Het gaat er natuurlijk om dat er een goed design uitkomt. Het zal niet zo zijn dat er op een Figma design een 3D-viewer gebouwd kan worden die interactief is. De interactiviteit in het latere HiFi design zal dus ook met de knoppen voor de functionaliteit gemaakt worden.

9.1.2 De low fidelity prototypen

Hier wordt de test data van de low fidelity prototypen uitgewerkt en er wordt op gereflecteerd, zodat het meegenomen kan worden in het high fidelity prototype. De feedback sessies zijn bedoeld voor het verbeteren van het product en het verbeteren van de user experience.

Design	**Feedback Pascal**	**Feedback André**
Generale feedback over het alle designs:	Doe vooral onderzoek naar de gebaren die nu in de Digitale Werkplaats zitten en houd alles gelijk.	Kijk ook vooral hoe je de transitie tussen de gebaren van de 2D PDF-viewer en de 3D IFC-viewer
	Kijk hoe de terug knop in de app zit verwerkt, implementeer het gelijk aan de app.	Als er twee verschillende manieren zijn zal het verwarrend zijn. Kijk voor de quick escapes naar de Android standaarden.
	Het is het beste om geen merkgegevens te tonen in de viewer. Deze staan apart in een scherm daarvoor bestemd.	Als optie zou je kunnen zeggen om hooguit de merknaam in de titelbalk van Android te vertonen.
	Zorg ervoor dat je de optie hebt om het hele model te zien, maar ook om alleen het geselecteerde merk te tonen	Maak een kleine toolbar die deze opties beschikbaar stelt.
	Zorg ook voor een optie om de camera te centeren.	Al deze opties moeten makkelijk beschikbaar zijn voor iedereen. Gebruik daarom de gestandaardiseerde iconen in plaats van tekst

Tabel 5: De feedback op de digitale LoFi-designs van André en Pascal

De eerste feedback die gegeven werd is dat er verder gekeken naar wat er nu staat. Hier kan het “everything is a remix”-principe op toegepast worden. Hetgeen wat nu staat (de Android app en de PDF-viewer) wordt genomen en het wordt dusdanig aangepast in het design dat de integratie van de 3D-viewer overblijft.

Het aanpassen van de huidige applicatie was al de bedoeling, maar deze feedback stuurt daar ook expliciet op. Ga niet veranderen wat men al qua interface gewend is, daar komen alleen maar moeilijkheden van.

Dan werd er gezegd om te kijken naar de Android standaarden. Dit wordt zoveel mogelijk gedaan. Voor quick escapes (een van de design heuristics) heeft Google al een document gedefinieerd in hun standaarden. Deze moeten aangehouden worden en er moeten geen slordige, onnodige knoppen op de interface geplaatst worden.

Om toch data en informatie te geven, mag hooguit in de titelbalk de merknaam weergeven worden. Op deze manier kan een medewerker niet vergeten met welk merk hij of zij bezig is. Dit wordt alleen getoond, omdat anders het scherm te druk en onoverzichtelijk wordt.

Dan werd de optie om niet-gemarkeerde onderdelen te verbergen. De zogeheten “hiding” optie van IFC.js maakt dit mogelijk. Dit is meer technische feedback en kan in Figma alleen naar voren komen door er een knop van te maken.

Dan werd er voor de knoppen een toolbar gesuggereerd. Wat ook een goed idee is, zo kan een medewerker ervoor kiezen een knop aan te klikken, en als het niet nodig is, de toolbar verbergen. Dit wordt ontworpen op een aantal web-standaarden, gezien de viewer een webapp die in Android geïntegreerd gaat worden is. In deze toolbar moeten twee functionaliteiten te kiezen zijn: een optie om de camera te resetten en de optie om de niet-gemarkeerde elementen te verbergen.

Met de huidige feedback zal een eerste high fidelity prototype gemaakt worden. Alles wat aan wensen en aanpassingen nu is aangegeven wordt verwerkt in een nieuw prototype. De kern van de aanpassingen zijn de suggesties die zijn gedaan bij de documentatie hierboven. Nadat de feedback in een high fidelity prototype is verwerkt wordt er een nieuwe feedback sessie ingepland.

9.1.3 De high fidelity prototypen

Als laatste feedback kwam terug dat de 3D-viewer een aanvulling wordt op de 2D-viewer, dus de PDF en de IFC-viewer zullen allebei aanwezig zijn. De PDF bevat de technische details en de 3D-viewer kan op die manier meer context bieden.

Design	**Feedback Pascal**	**Feedback André**
Generale feedback over het alle designs:	Pas in plaats van webdesign, Material design toe	Door Material Design kun je het nog meer versimpelen. Kijk naar Floating Action Buttons.
	De IFC-viewer wordt een aanvulling op de 2D-viewer.	Kijk goed naar de navigatiebalk en kijk hoe je de tab van de PDF-viewer er nog tussen krijgt.
	Hooguit komt de merknaam in de bovenste balk.	Om het project specifiek te houden, moet er een volledig IFC-model per project ingeladen worden.
	Markeer het merk via de naam, en niet via de GUID. Dat heeft te veel ConstruSteel aanpassingen nodig.	Markeer dan ALLE merken met dezelfde merknaam, en niet slechts één uit de hele selectie.

Tabel 6: De feedback van André en Pascal op het eerste HiFi prototype

Met deze feedback wordt het laatste design gemaakt. Het gefinaliseerde design dat de Android en web omgeving samenvoegt en laat overkomen zoals het in de Digitale Werkplaats hoort. Nadat de IFC-viewer is geïmplementeerd gaat deze voorzien worden van unit en systeem tests. Zo kan er zeker gesteld worden dat het systeem daadwerkelijk werkt.

10. Projectmanagement reflectie

10.1 Planningsmethode

Tijdens dit project werkte men volgens de Kanban methode. Dit kwam door een grote hoeveelheid issues, en daardoor is de organisatie tijdelijk op de Kanban-methodiek overgestapt. Logischerwijs is dit de werkmethode die gevolgd is tijdens dit project. Met behulp van een GANTT-planner (Bijlage 4, hoofdstuk 7) is de voortgang bijgehouden. Daarmee kon er goed gezien worden hoe ver bepaalde issues waren en hoe de voortgang verliep in percentages (hoe verder een issue af was, hoe hoger het percentage van "done"). Dit gaf een fijne structuur.

In de laatste 6 weken van het afstuderen werden er zes sprints en zes doelen gedefinieerd. De actiepunten die de userstories representeerden werden genoteerd per week en het weken overzicht (Bijlage 5) werd uitgedraaid als een soortement van houvast. Punten die af waren werden afgevinkt. Dit gaf weer extra inzicht in het verloop van de afstudeerperiode naast de GANTT-planner.

Al met al ging het team langzaam weer terug naar SCRUM. Daarom is ervoor gekozen om dit project op deze "6 weken en 6 sprints" methode vooruit te plannen. Zo kon overzicht voor alle partijen beter bewaakt worden, en het geeft rust. Deze planning was nodig, want door het overzicht is rust gecreëerd, en met deze rust kan kwaliteit weer beter bewaakt worden.

Als een punt af is, wordt deze afgevinkt en als de gelegenheid er is, wordt het kort gedemonstreerd aan Pascal. Dan kan er gelijk wat eventuele UX/UI feedback opgeschreven worden. Voor de grote feedback kan dan een pull-request aangemaakt worden. De pull-requests worden als werk tussendoor meegenomen in de sprint. Op deze manier kan de code opnieuw opgegeven worden voor review en als er dan nog tijd tussendoor valt kan er weer aan gewerkt worden. Uiteindelijk zijn de GANTT-planner en het wekenoverzicht (Bijlage 4, hoofdstuk 7 en Bijlage 5) leidend.

10.2 Overlegstructuur

Voor feedback op ontwerp en voorwerk werd er tijdig een meeting ingeschoten. Zo kon de feedback na review besproken worden en meegenomen worden in een nieuwe iteratie van het onderwerp. Dit was vooral nuttig bij het reviewen van UX en UI ontwerpen. Feedback werd direct tijdens het tonen van de designs gegeven en opgeschreven, op die manier konden de designs verbeterd worden. Daarna kon er een nieuwe meeting gepland worden en kon de feedback weer direct ontvangen worden. Dit was een erg prettige manier van werken.

Code reviews werden geregeld via pull-requests. Als een functie klaar was kon er een pull request aangemaakt worden. Deze werd dan bekeken en van feedback voorzien. Hiervoor hoefde geen meeting ingeschoten te worden, omdat de feedback schriftelijk bij de code gezet kon worden in Jira. Dit werkte goed en gaf gelijk de gelegenheid om de code in context met de opmerkingen van de reviewer te bekijken.

De daily standups waren momenten waar men door middel van de gebruikelijke vraag "wat heb ik gedaan en wat ga ik vandaag doen?" waar oplossingen en feedback in een compacte, maar duidelijke manier overgebracht konden worden. Als een probleem blokkerend werd, kon dat daar aangegeven worden en dan kon een collega later die dag of direct na de stand-up even meekijken.

Daarnaast werden er tussendoor ook gewoon vragen gesteld. Als het antwoord niet direct gegeven kon worden dan was het verzekerd dat het ietsje later zou komen. Soms is het ook verstandig om

een kleine demo in te plannen, dan wel te vragen of iemand mee kan kijken voor UX/UI feedback of voor code feedback. Een kort moment plannen en een kort moment vragen kan altijd.

Voor grote vragen is het logisch om even een moment te plannen als het nodig is. Als er tussendoor overlegd kan worden, of als er tussendoor een moment genomen kan worden om iemand te helpen, dan wordt dat ook gedaan. Vaak wordt de oplossing dan binnen vijftien minuten gevonden en kan er verder gewerkt worden. Als dit niet het geval is, kan er gekozen worden om even een moment te plannen om het goed uit te zoeken met elkaar.

11. De conclusie

De hoofdvraag luidde: “Wat kan de IFC-viewer bijdragen aan het bestaande papierloze werkproces?”, het huidige papierloze proces is met een tablet en PDF-tekening die de technische details laat zien. Er is gebleken dat dit belangrijke context soms laat missen. Daarom was er vraag naar een 3D-viewer.

De IFC-viewer draagt positief bij aan het bestaande papierloze werkproces, omdat het de context biedt die op een 2D-tekening niet te zien is. De kleinste uithoekjes worden duidelijk, en door de markering van het merknummer waarmee de fabrieksmedewerker bezig is wordt duidelijk hoe een merknummer in een constructie verwerkt zit.

De technologische deelvragen allemaal succesvol beantwoord. Op het gebied van performance zijn er nog wel verdere verbeteringen noodzakelijk De combinatie van gekozen tablet en de nu beschikbare oplossing laat op sommige vlakken zien, dat eindgebruikers wachttijd beleven. Dit is op te lossen door de aanschaf van snellere tablets, echter zal er meer onderzoek gedaan moeten worden op het gebied van snellere tablets. Er moet gekeken worden of dit ook echt de oplossing is. De huidige snelheid is voor dit prototype acceptabel. Op het moment dat dit als product beschikbaar gesteld gaat worden, moeten er nog aanvullende requirements qua tijd en modelgrootte komen om performance issues hopelijk te voorkomen.

Toch is dit opgelost door het markeren van een merk zo optimaal mogelijk te laten verlopen middels filtering en middels niet meer doen dan dat nodig is. Doordat die code geoptimaliseerd is op wat huidig kan, verloopt het proces na het inladen snel en kan men zonder te veel moeite gebruik maken van de IFC-viewer.

De IFC-viewer zal positief bij kunnen dragen aan het werkproces van staalbouw- en constructiebedrijven. Het geeft meer context aan de technische tekening. Dit kan faalkosten verminderen. Minder faalkosten betekent ook natuurlijk minder verspilling van staal en minder kosten om nieuw staal te bestellen.

De grootste bijdrage van de IFC-viewer zal context zijn. De context van hoe iets in elkaar is geassembleerd helpt bij het assembleren van een staalconstructie. Als die context niet aanwezig is, dan kan het zijn dat het fout wordt geassembleerd. Nu de 3D-viewer de technische tekening de vorm geeft die het gaat krijgen, wordt het voor de staalbouwers makkelijker om een merk in elkaar te zetten.

12. Aanbevelingen

12.1 Migratie API van NodeJS naar C#

Om een naadloze samenwerking tussen de Digitale Werkplaats, ConstruSteel en de IFC-viewer te realiseren is het van belang dat de NodeJS API naar C# wordt omgezet. Dit kan door te kijken naar het inladen van pdf-bestanden. Dit kan geoptimaliseerd worden voor IFC-bestanden. Deze code kan dan gebruikt worden om de bestanden aan de Android app te leveren. Deze zal de bestanden weer door moeten sturen naar de WebView.

12.2 Highlighten via GUID

Vooralsnog gaat het highlighten van merken via merknaam. Het is de bedoeling om dit via GUID te laten lopen. Het is dus van belang om te kijken of de API aangepast kan worden zodat de GUID's aangeleverd kunnen worden aan de IFC-viewer. Dan met een code aanpassing in de IFC-viewer komen de merken uniek gemarkeerd worden. Als er nu een merk gemarkeerd wordt, dan kan het zijn dat er acht gemarkeerde merken zijn, omdat die allemaal dezelfde merknaam hebben. Al deze acht merken hebben weliswaar wel allemaal een unieke GUID.

12.3 Maak ConstruSteel On Premise klaar voor het aanleveren van IFC-bestanden

ConstruSteel kan nu pdf-bestanden aan de database leveren. Deze bestanden worden dan via de API weer aan de Digitale Werkplaats geleverd. Er is een inputveld voor IFC-bestanden, maar daar wordt nu nog niks mee gedaan. Dit moet, op dezelfde manier als bij pdf-bestanden, functioneel worden. Een werkvoorbereider geeft een pad op naar een IFC-bestand, dit wordt gegeven aan de database en via de API kan het dan geleverd worden aan de Digitale Werkplaats.

12.4 Optimalisatie IFC-bestanden

Het is van belang om alsnog de IFC-bestanden geoptimaliseerd te krijgen. De performance is nu goed, maar nog niet zoals het moet. Er komt een Tekla plug-in in development die de data voor de Digitale Werkplaats gaat optimaliseren. Hier zou IFC in meegenomen kunnen worden om die dusdanig te optimaliseren dat de code lichter kan worden en dat de performance soepeler wordt in gebruik.

12.5 Verder onderzoek naar performance en performanceverbetering

Als laatste advies is het van belang om verder onderzoek te doen naar performanceverbeteringen van de 3D-viewer. De 3D-viewer is een zwaarder programma voor tablets. Bij grotere IFC-modellen kunnen de laadtijden oplopen. Het is van belang deze laadtijden zo laag mogelijk te houden. Daarom moet er onderzocht worden of snellere tablets de oplossing kunnen zijn.

13. Persoonlijke reflectie

In deze afstudeerperiode is veel geleerd. Projectmanagement, programmeren, formeler schrijven, dit zijn allemaal punten die zijn verbeterd. De afgelopen periode heeft me ook weer met nieuwe programmeer principes bekend gemaakt. De Android app van ConstruSteel is complex en maakt gebruik van technieken die nog niet eerder in de opleiding gebruikt waren. Dit was interessant om doorheen te gaan, en heeft me zeker meer laten zien van de mogelijkheden van Android.

De mogelijkheden met integratie van web en Android was ook een interessant onderdeel van het project. Dit onderdeel liet zien dat, wanneer er specifiek ontworpen webapplicaties gemaakt worden, de principes van Android en web kunnen goed met elkaar samen werken.

Naast de inzichten op technologisch gebied is er ook een goed inzicht gekomen van de werking van een bedrijf waar software engineering, consultancy en customer support nauw samenwerken. Dat heb ik nog niet op die manier eerder meegemaakt. De samenwerking zien, en de samenwerking mogen meemaken door een simpele issue tussendoor op te lossen was erg leuk en leerzaam om te doen. Deze wijze van werken gaf weer inzicht in andere varianten op het bedrijfsleven binnen de ICT.

Het samenwerkingsverband van ConstruSteel binnen het team is ook erg prettig. De kennis dat er altijd iemand kan helpen is erg fijn. Vragen stellen mag altijd, al ging dat in het begin wat moeilijk. Dat heb ik steeds beter leren doen. En als de hulp niet direct geboden kan worden, dan kan het een moment later wel. Vragen leren stellen is iets wat sinds de stageperiode in 2020 verbetert en steeds iets meer blijft door verbeteren.

Wat eerder gedaan kon worden was het toepassen van SCRUM in plaats van Kanban. Achteraf had ik in Plan van Aanpak beter moeten nadenken over de verschillende milestones in deze 20 weken. Achteraf paste de initiële keuze Kanban goed bij de 1^{e} helft. Halverwege kwam er ook een nieuwe software-afdeling manager in dienst. Die confronteerde mij met een aantal vragen, waarop ik vanuit de Kanban aanpak onvoldoende houvast bleek te hebben. Heeft me gevraagd inzage te geven in de diverse milestones voor de 2^{e} helft. Dit heeft geresulteerd in het toepassen van SCRUM, middels wekelijkse sprints. Daarnaast ook wekelijkse (persoonlijke) reflectie gesprekken met Heini Veneberg. Dit alles resulteerde in een persoonlijke verbetering in het `in control` zijn van mijn 2^{e} helft van afstuderen. Dit heeft bijgedragen aan tijdig kunnen opleveren van concept en definitieve versie. Daarnaast product doel in het oog houden, incl. testen en werken aan verslag en eind presentatie.

Bibliografie

(sd).

3D-Ace. (2022, Februari 22). *3D-Model Optimalization: How does It Work*. Opgehaald van 3D-Ace: https://3d-ace.com/blog/3d-model-optimization/

Android Developers. (2022, Mei 11). *WebView*. Opgehaald van Android Developers: https://developer.android.com/reference/android/webkit/WebView

buildingSMART. (2018, Mei 14). *IFC Formats*. Opgehaald van BuidlingSMART: https://technical.buildingsmart.org/standards/ifc/ifc-formats/

BuildingSMART. (z.d.). *IfcElementAssembly*. Opgehaald van buildingsmart.org: https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC2/HTML/schema/ifcproductextension/lexical/ifcelementassembly.htm

De Groot, H. (2021, November 22). Mijn Werk (De Digitale Werkplaats) - ConstruSteel Documentation. Den Ham, Overijssel, Nederland. Opgehaald van Confluence.

De Swart, K. (2021, November 11). Interview Pilot klant Jac Reijns Staalconstructie (uit Webinar Digitale Werkplaats). (ConstruSteel, Interviewer)

Demaine, E., Perrier, J.-Y., & MDN-Contributors. (2021, Juli 20). *export - JavaScript*. Opgehaald van MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/export?retiredLocale=nl

DoubleWeb. (2021, Juni 16). *Wat is een doelgroep?* Opgehaald van Online marketing begrippenlijst DoubleWeb: https://www.doubleweb.nl/wat-is/doelgroep/

Friis Dam, R. (2021, Februari 16). *5 stages in the Design Thinking process*. Opgehaald van Interaction Design Foundation: https://www.interaction-design.org/literature/article/5-stages-in-the-design-thinking-process

González Viegas, A. (2021, Juli 27). *The BIM that we need*. Opgehaald van IFC.js: https://ifcjs.github.io/info/blog/prometheus

González Viegas, A., Collin, H., & Van Diggelen, T. (2022, Februari 16). *Introduction*. Opgehaald van IFC.js: https://ifcjs.github.io/info/docs/introduction

Jansen, C. (2018, Oktober 10). *Missie*. Opgehaald van ConstruSteel: https://www.construsteel.com/nl/over-ons/missie/

Kanbanize. (2022, Mei 25). *What is Kanban? Explained in 10 minutes*. Opgehaald van Kanbanize: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban

Kulling, K. (2020, December 15). *Introduction*. Opgehaald van Asset-Importer-Lib December 2020 documentation: https://assimp-docs.readthedocs.io/en/v5.1.0/about/introduction.html

Larin, A. (2021, December 14). *How to build 3D webapps. Part 4: Backend of 3D webapps*. Opgehaald van CAD Exchanger: https://cadexchanger.com/blog/how-to-build-3d-web-apps-part-4-backend-of-3d-web-apps/

MDN. (2022, Mei 28). *WebAssembly*. Opgehaald van MDN: https://developer.mozilla.org/en-US/docs/WebAssembly

Mr.doob. (2010, April 24). *mrdoob/Three.js: JavaScript 3D-library*. Opgehaald van GitHub: https://github.com/mrdoob/three.js/

Mysliwiec, K. (2020, April 27). *Circular Dependency*. Opgehaald van NetJS - A progressive NodeJS framework: https://docs.nestjs.com/fundamentals/circular-dependency

Nisbet, N., & Liebich, T. (2007). *ifcXML Implementation Guide.* Londen: International Alliance for Interoperability.

Pauwels, P. (2020, Oktober 23). *pipauwel / IFC.JAVA*. Opgehaald van GitHub: https://github.com/pipauwel/IFC.JAVA

Scrum company. (2022, Mei 25). *Wat is Scrum? | Alles over de Srum methode*. Opgehaald van Scrum Company: https://www.scrumcompany.nl/wat-is-scrum/

Sharpened Productions. (2022, Januari 26). *IFC File Extension - What is an .ifc file and how do I open it?* Opgehaald van FileInfo.com: https://fileinfo.com/extension/ifc

Sharpened Productions. (2022, April 28). *OBJ File Extension - What is an .obj file and how do I open it?* Opgehaald van FileInfo.com: https://fileinfo.com/extension/obj

Westenenk, S. (2021, November 11). Digitale Werkplaats ConstruSteel. *Webinar De digitale werkplaats met ConstruSteel*. Den Ham, Overijssel, Nederland: ConstruSteel Steel Fabrication Software B.V.

Wind, R. (2021, November 3). Planbord - Technische Documentatie. Den Ham, Overijssel, Nederland.

Bijlagen

Alle bijlagen zijn te vinden in de directory “Afstudeerdossier”.

- Bijlage 1. Adam, S. (2022, Mei 25). IFC-research. Den Ham, Overijssel
- Bijlage 2. Adam, S. (2022, Mei 25). Software Design. Den Ham, Overijssel
- Bijlage 3. Adam, S. (2022, Mei 25). API Definition. Den Ham, Overijssel
- Bijlage 4. Adam, S. (2022, Februari 23). Plan van aanpak. Den Ham, Overijssel
- Bijlage 5. Adam, S. (2022, Mei 2). Wekenoverzicht. Den Ham, Overijssel
- Bijlage 6. Adam, S. (2022, Mei 27). Het functionele design. Den Ham, Overijssel
- Bijlage 7. Adam, S. (2022, Mei 25). Retrospective. Den Ham, Overijssel
- Bijlage 8. Adam, S. (2022, Mei 27). Testplan + Testcases IFC-viewer en UX-UI tests. Den Ham, Overijssel
- Bijlage 9. Adam, S. (2022, Juni 10). Het technische document. Den Ham, Overijssel