

# Furnify Home

Eindverslag

19/05/2024

Team 02



Alexandra Ganseman

Nathan Salabiaku

Thomas Lonneville

Wiebe Vandendriessche

Xander Vanparys

<http://157.193.171.41/>

Furnify Home, een Gentse start-up, verkoopt modulaire, plaatsbesparende meubels. Momenteel moeten medewerkers naar klanten toe om kamers op te meten en geschikte modules te bepalen, wat veel tijd kost en de groei beperkt. Het doel is om dit euvel op te lossen met een applicatie waarmee klanten zelf een eigen kamer na kunnen bootsen en de geschikte module kunnen selecteren.

Dit project loste dit op aan de hand van een webapplicatie bestaande uit een 3D-ruimte en een zijscherm dat dienst doet als vragenlijst. In de 3D-ruimte kunnen gebruikers obstakels toevoegen. Aan de hand van de vragenlijst zal er worden bepaald welke module een gebruiker wenst en of dit mogelijk is in de kamer. Daarna zal deze in de ruimte worden ingeladen. In het verslag wordt besproken wat er hiervoor nodig is en hoe dit is gerealiseerd.

## Inhoudsopgave

Lijst met figuren .....	5
Inleiding.....	6
Probleemstelling .....	6
Doelstelling.....	7
Hoofdstuk 1: Gebruikersaspecten.....	8
Hoofdstuk 2: Systeemarchitectuur .....	10
Activiteitendiagram .....	10
Toestandsdiagram .....	12
Opbouw applicatie .....	17
App.jsx en Main.jsx.....	17
Zijscherf .....	18
Databank .....	18
Contexten .....	19
I18N .....	19
2D .....	19
3D .....	20
Bepalingsalgoritme.....	21
Hoofdstuk 3: Testplan .....	22
End-to-end.....	22
Cypress .....	22
Doel tests.....	22
Stop .....	23
Locatie .....	23
Hoofdstuk 4: Evaluatie en discussies .....	25
Performantie .....	25
<i>Computergraphics</i> .....	25
<i>Vite</i> .....	25
Aandachtspunten bij implementeren .....	25
Laadscherf .....	26
Beveiliging databank .....	26
Schaalbaarheid .....	27
Extra modellen .....	27
Extra kleuren en houttexturen.....	28

Afmetingen van het model.....	28
GDPR-analyse .....	29
MongoDB Databank .....	29
Mailchimp.....	29
Furnify .....	29
Duurzaamheid .....	30
Algemene problemen.....	31
Databank .....	31
Communicatie opdrachtgever.....	31
Niet rechthoekige kamers .....	31
Specifieke moeilijkheden bij de implementaties .....	32
Verslepen.....	32
Roteren.....	33
Kamer achteraf verkleinen .....	33
Ramen en deuren .....	33
Overlappende objecten.....	34
Probleem met één <i>tick</i> vertraging.....	34
Coördinaten muis in 3D.....	34
Overlap muren en vloer .....	35
Hoofdstuk 5: Handleidingen.....	36
Applicatie vereisten.....	36
Applicatie bereiken .....	40
Applicatie uittesten in ontwikkelingsomgeving .....	40
Applicatie zelf ontplooiën op een linux server.....	41
Toevoegen/aanpassen eisen van een model .....	43
SSL-certificaat, HTTPS en Domeinnaam .....	46
SSL-certificaat: .....	46
Domeinnaam:.....	46
Gebruikershandleiding .....	47
Besluit.....	49
Referentielijst .....	50

## Lijst met figuren

Figuur 1: usecase-dagram .....	9
Figuur 2: activiteitendiagram voor gebruiker .....	10
Figuur 3: activiteitendiagram Login .....	12
Figuur 4: toestandsdiagram vragenlijst .....	13
Figuur 5: deeltoestand vragenlijst ruimte .....	14
Figuur 6: deeltoestand vragenlijst functies .....	15
Figuur 7: deeltoestanden voor het tonen van de modules .....	16
Figuur 8: architectuur van de applicatie .....	17
Figuur 9: zijscherm .....	18
Figuur 10: vloerplan .....	20
Figuur 11: 3D-omgeving .....	20
Figuur 12: Cypress .....	24
Figuur 13: Cypress overview .....	24
Figuur 14: weergave CSV-bestand in Excel .....	27
Figuur 15: MongoDB connectiestring .....	36
Figuur 16: voorbeeld MongoDB .....	36
Figuur 17: type tekstvelden (links) en formulervelden (rechts) .....	37
Figuur 18: afgewerkt mailchimp formulier .....	38
Figuur 19: formulier afwerken .....	38
Figuur 20: formulier url .....	39
Figuur 21: meubels.csv bestand .....	43
Figuur 22: voorbeeld blender .....	44
Figuur 23: exporteren in blender .....	45

## Inleiding

Furnify Home werd opgericht door Joachim Schouten en Isaï Cornelis. Volgens de website is het een Gentse start-up die ontstond dankzij het 'Expedition DO!' programma van de Universiteit Gent. In 2022 overtuigde het concept de jury en begonnen de ondernemers met de start-up Furnify Home. Het jaar 2023 bracht bekroningen als een van de snelst groeiende start-ups binnen 'Start it @KBC', en een derde plaats als beste studentenonderneming van België (Furnify, z.j.).

Furnify Home richt zich op de innovatieve sector van modulaire, plaatsbesparende meubels. Door de voortdurende evolutie van woningontwerpen, waar ruimtes steeds compacter worden, komt de noodzaak voor efficiënte inrichtingsoplossingen. Furnify Home legt zich toe op het optimaliseren van leefruimtes door multifunctionele modules te ontwerpen, waaronder slaapkamers, bureauruimtes en inloopkasten. Zo wordt elke vierkante meter optimaal benut.

## Probleemstelling

Momenteel rijdt Furnify Home zelf met de auto ter plaatste naar de klant. Dit om de afmetingen van de ruimte op te meten, maar ook om de kamer te bestuderen en zo te bepalen wat de optimale plaatsing van de module in de ruimte zou zijn. Het probleem dat hierbij ondervonden wordt, is dat een groot deel van het cliënteel zich in de groten steden bevindt. Dit zorgt ervoor dat Furnify Home veel in de file staat, wat veel tijdsverlies en CO<sub>2</sub>-uitstoot oplevert. Hiernaast neemt het aantal klanten ook steeds meer toe en is er ook een uitbreiding naar de Nederlandse markt. Het is niet langer schaalbaar om op deze manier te blijven werken.

Er is dus nood aan een slimme digitale oplossing, waarbij geen verlies optreedt aan de huidige diensten die Furnify Home levert aan de klant. De focus ligt op het ontwikkelen van een geavanceerd instrument om dit proces te optimaliseren.

## Doelstelling

Zoals eerder vermeld, ligt de focus op het ontwikkelen van een geavanceerd instrument. Het doel is een gebruiksvriendelijke webapplicatie te creëren die zowel op gsm als op laptop functioneert. De applicatie dient uit twee grote componenten te bestaan, namelijk een vragenlijst en een 3D-visualisatie van de kamer.

De webapplicatie zal volledig *front-end* werken. Hiervoor wordt er gebruik gemaakt van React, een JavaScript-bibliotheek die wordt gebruikt om dynamische gebruikersinterfaces te bouwen (Describing UI, z.j.). Het heeft als voordeel dat codecomponenten kunnen worden hergebruikt. Dit maakt het mogelijk om de vragenlijst op een eenvoudige manier in verschillende secties te verdelen, waardoor de applicatie overzichtelijk blijft. De opdrachtgever verwacht dat de vragenlijst een vijfde van het scherm inneemt. Om het gebruiksgemak op een gsm te verbeteren zal dit worden opgelost door het gebruik van een uitklapbare balk.

Voor de 3D-visualisatie zal er gebruikgemaakt worden van Three.js. Dit is een JavaScript-bibliotheek die wordt gebruikt om 3D-figuren te creëren en af te beelden op een website (Coleman, 2017). Hiermee kan ook aan de wens van de opdrachtgever worden voldaan om de ruimte van de gebruiker in meerdere perspectieven te tonen.

In het eerste hoofdstuk van dit verslag wordt er toegelicht hoe de vragenlijst eruitziet en hoe deze vragen het 3D-model beïnvloeden vanuit het standpunt van de gebruiker. In het daaropvolgende hoofdstuk wordt er aan de hand van een activiteitendiagram besproken welke processen er door het systeem zullen worden uitgevoerd. Tot slot komen de verschillende toestanden waarin de webapplicatie zich kan bevinden aan bod. Dit zal worden geïllustreerd met toestandsdiagrammen.

## Hoofdstuk 1: Gebruikersaspecten

Om het heen en weer rijden van klant tot klant te vermijden, biedt Furnify Home een digitale configuratietool aan. Mensen die geïnteresseerd zijn in de meubels van Furnify Home kunnen zelf de afmetingen van de kamer en mogelijke obstakels invoeren. Op die manier kan de online tool bepalen of de meubels in de kamer passen en hoeveel ruimte bespaard zal worden.

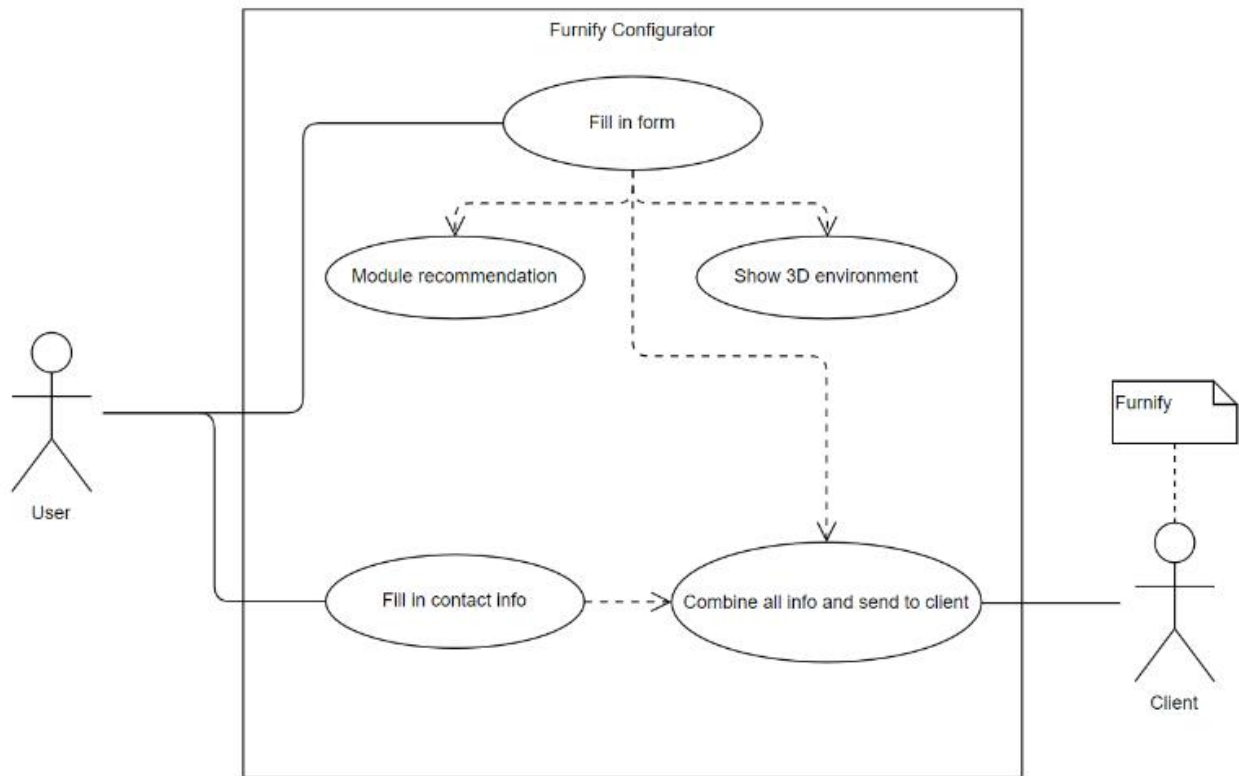
Wanneer een gebruiker de website bezoekt, wordt deze een vragenlijst gepresenteerd. Het doel van deze lijst is om de voorkeuren en eisen van de gebruiker vast te stellen. De verzamelde informatie omvat:

- de belangrijkste functies die de woonruimte moet bieden, zoals een bed, opbergruimte, bureau, enzovoort;
- de gewenste indeling van de ruimte, bijvoorbeeld of modules gebruikt moeten worden als scheidingswand of centraal in de ruimte geplaatst moeten worden;
- de voorkeur voor materiaalkeuze van de modules;
- de afmetingen van de kamer en eventuele obstakels die aanwezig zijn;
- de matrasvoorkeur van de gebruiker, indien een bed wordt geselecteerd;
- eventuele andere specifieke wensen of voorkeuren die voor Furnify Home van belang zijn om te weten.

Op basis van deze voorkeuren en rekening houdend met de beschikbare ruimte in de kamer worden er combinaties van modules voorgesteld.

Zoals afgebeeld in figuur 1, komt er een 3D-omgeving tevoorschijn terwijl de gebruiker de vragenlijst invult. Met elk gegeven antwoord past de omgeving zich aan om de ingevoerde voorkeuren steeds nauwkeuriger weer te geven. Na het invullen van de afmetingen verschijnt er een lege kamer. Uiteindelijk zal de gebruiker een voorstel krijgen voor de plaatsing van de modules. Dankzij de 3D-tool kan de gebruiker een goed beeld schetsen van hoe zijn kamer eruit zal zien met de modules erin. De gebruiker kan interageren met deze omgeving om de ruimte volledig naar wens in te richten.





*Figuur 1: usecase-diagram*

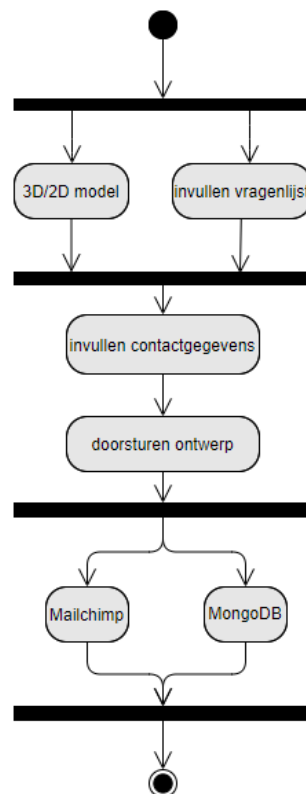
Als de gebruiker tevreden is met de configuratie van de modules in de ruimte, kan deze zijn contactgegevens invullen. De gegevens van de vragenlijst gaan naar Furnify Home voor het uitwerken van een passende oplossing. Op deze manier is er een efficiëntere samenwerking en krijgen de gebruiker en Furnify Home een versneld inzicht in het potentiële eindresultaat.

## Hoofdstuk 2: Systeemarchitectuur

### Activiteitendiagram

Het activiteitendiagram van de webapplicatie begint met het invullen van de vragenlijst die geprogrammeerd is in React.js. React.js werd gekozen omdat het eenvoudig te implementeren is voor *single-page* en mobiele applicaties. Dankzij herbruikbare componenten en de virtuele *Document Object Model ()* worden alleen de wijzigingen opnieuw ingeladen, wat de efficiëntie verhoogt. Bovendien biedt React.js ook veel open-source hulpmiddelen van derden om een responsieve webapplicatie te ontwikkelen.

Gebruikers kunnen hiermee de benodigde informatie voor de woonruimte invoeren. De vragenlijst is ontworpen met responsieve lay-outs om een optimale gebruikerservaring te bieden op alle apparaten. Tijdens het invullen van de vragenlijst worden de gegevens bijgehouden in contextklassen.



Figuur 2: activiteitendiagram voor gebruiker

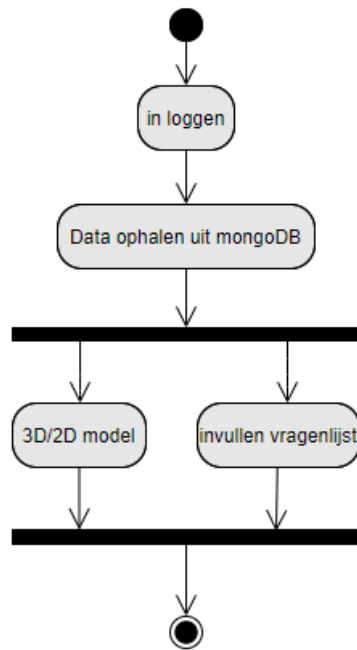
Parallel aan het invullen van de vragenlijst wordt ook een 3D-model van de woonruimte gegenereerd met behulp van Three.js. Een opensource bibliotheek voor het maken van 3D-

modellen in Javascript. Door gebruik te maken van *Web Graphics Library* (WebGL), een Javascript-API om 3D en 2D modellen te maken zonder *plug-ins*. Het model wordt gemaakt op basis van de ingevulde gegevens in de vragenlijst. Hierbij is het mogelijk om verschillende modules in de woonruimte te plaatsen. De aangeboden modules variëren afhankelijk van de informatie die de gebruiker heeft ingegeven in de vragenlijst. Dit geeft de gebruiker een volledig beeld van hoe de modules eruit zullen zien in de eigen ruimte. Het 3D-model wordt ook ontworpen met een responsieve lay-out.

Wanneer de gebruiker tevreden is met het resultaat, kan deze zijn contactgegevens invullen. Dit stelt Furnify Home in staat om contact op te nemen met de klant. De ingevoerde gegevens worden samen met de rest van de vragenlijst bijgehouden in de contextklassen.

Nadat de vragenlijst is ingevuld en het 3D-model is voltooid, worden de ingevoerde gegevens via het formulier verzonden naar zowel Mailchimp als de MongoDB databank. Mailchimp is een e-mailmarketingplatform dat contactgegevens kan beheren, wat een van de vereisten van de opdrachtgever was voor het verzenden van gegevens. De gegevens worden in een querystring gestoken en verzonden aan de hand van JSON with padding(JSONP) om Cross origin resource (CORS) sharing te vermijden. CORS is een beveiligingsmechanisme in webbrowsers dat bepaalt of en hoe inhoud van één domein kan worden opgevraagd door een webpagina van een ander domein. JSONP is een techniek waarmee scripts van een ander domein kunnen worden geladen en uitgevoerd in de browser door een script-tag te gebruiken, waarmee cross-origin verzoeken mogelijk worden zonder de beperkingen van CORS. MongoDB is een NoSQL-database die gegevens opslaat in Binary JSON (BJSON), een binair gecodeerde serialisatie van JSON-achtige documenten die efficiënter door MongoDB kunnen worden verwerkt. MongoDB is ook *schemaless*, wat betekent dat de structuur van de database niet op voorhand gedefinieerd moet worden, waardoor meer flexibiliteit ontstaat. Op deze manier kan het bedrijf de gegevens verwerken volgens de behoeften van de gebruiker.

Figuur 3 toont wat er gebeurt wanneer de gegevens worden opgehaald via de e-mail van de gebruiker. De persoon die de gegevens probeert op te halen, moet eerst inloggen via een inlogpagina en een gebruikersnaam en wachtwoord invoeren. Vervolgens wordt de vragenlijst opnieuw ingevuld en het 3D-model opnieuw opgebouwd zoals de gebruiker het eerder heeft gedaan.



*Figuur 3: activiteitendiagram Login*

## Toestandsdiagram

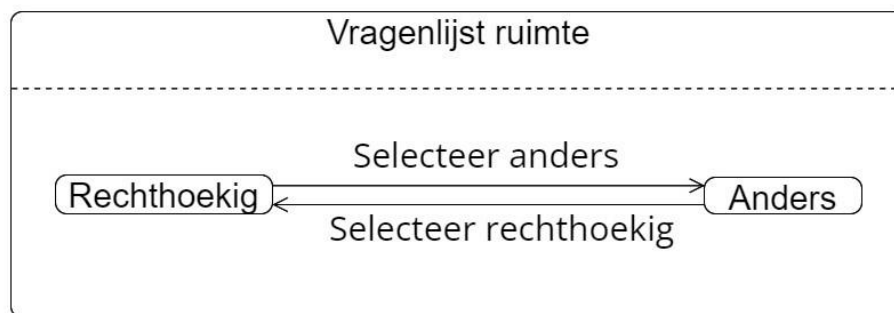
Op figuur 4 zijn de zes verschillende toestanden, die de vragenlijst kan aannemen, zichtbaar. De gebruiker kan de toestand van de vragenlijst veranderen door op de knoppen 'volgende' en 'terug' te klikken. De verschillende toestanden en hun eventuele deeltoestanden worden hieronder verder uitgelegd.



*Figuur 4: toestandsdiagram vragenlijst*

Bij het open van de website, zal de gebruiker de vragenlijst over de ruimte te zien krijgen. Deze heeft zelf nog twee deeltoestanden, zichtbaar op figuur 5. De gebruiker kan selecteren wat de vorm van de ruimte is. Afhankelijke van de optie zal er iets anders getoond worden. De eerste optie is dat de ruimte rechthoekig is, in dit geval zal de gebruiker de optie krijgen om de

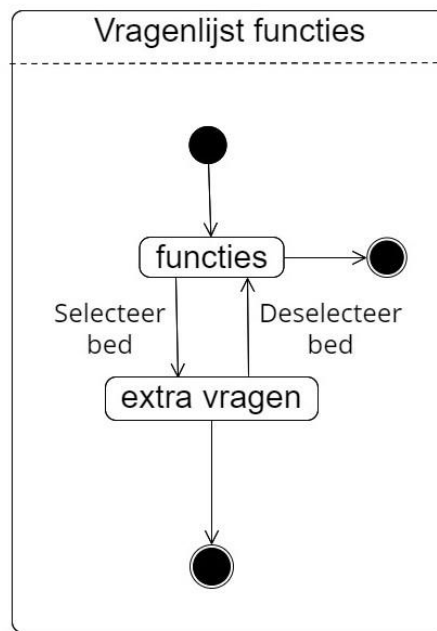
afmetingen in te geven. De tweede optie is dat de ruimte een andere vorm heeft, hier dient de gebruiker enkel nog de hoogte van de kamer op te geven.



*Figuur 5: deeltoestand vragenlijst ruimte*

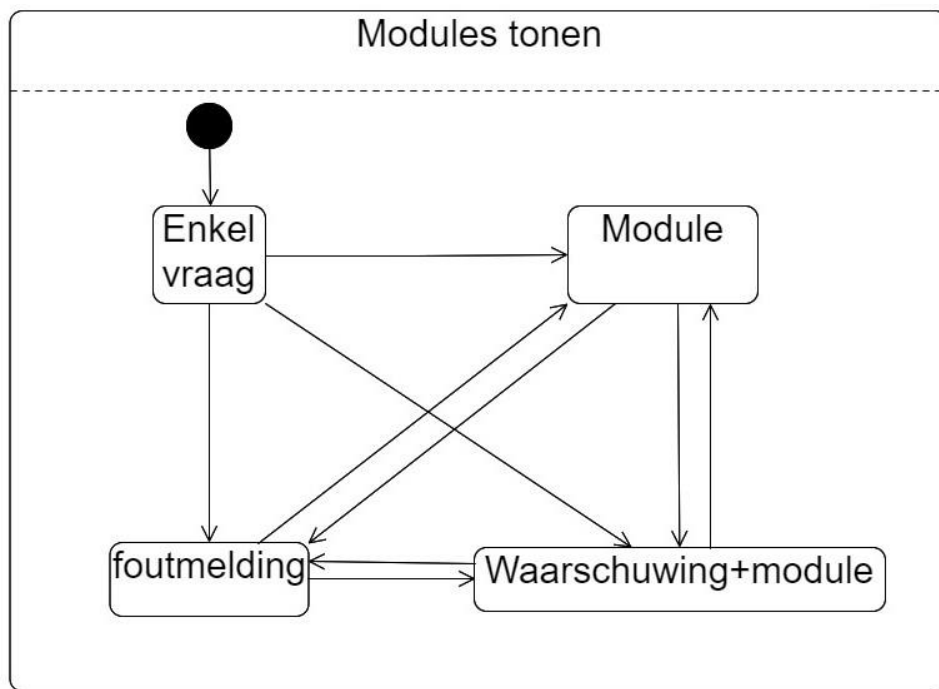
Wanneer de gebruiker op de knop 'volgende' drukt zal deze van de vragenlijst over de ruimte naar de obstructie pagina gaan. Op deze pagina krijgt de gebruiker de optie om alle obstakels toe te voeg. Elk obstakel dat aangedrukt wordt zal onderaan de pagina verschijnen, deze kan ook open geklikt, dicht geklikt en verwijderd worden.

De volgende pagina die de gebruiker te zien krijgt is de vragenlijst over de functionaliteit, deze is zichtbaar op figuur 6. Enkel wanneer de gebruiker opgeeft dat deze nood heeft aan een bed zullen er twee extra vragen verschijnen. De eerste vraag zal gaan over de keuze van matras die de gebruiker wenst. De tweede over de breedte van het bed.



*Figuur 6: deeltoestand vragenlijst functies*

Hierna krijgt de gebruiker de module pagina te zien. Op figuur 7 zijn de vier verschillende deeltoestanden te zien. Als de gebruiker voor de eerste keer de pagina opent zal deze enkel de vraag en een knop te zien krijgen. Wanneer deze op de knop drukt, zal deze één van de drie andere toestanden te zien krijgen. Welke toestand is afhankelijk van de ingegeven gegevens op de vorige pagina. Als er geen problemen zijn, zal de gebruiker gewoon de mogelijk modules te zien krijgen en de informatie over de geselecteerde module. Het kan zijn dat bij de functies de klant een combinatie ingegeven had die Furnify Home niet aanbiedt. Er zal dan gekeken worden of ze geen producten aanbieden met gelijkaardige vereisten. Als dit zo is, zullen er opnieuw opties weergegeven worden, en zal er ook een waarschuwing aan de klant gegeven worden dat er gelijkaardige vereisten ingegeven werden. De laatste toestand kan in drie gevallen voorkomen. Als eerste wanneer er geen gelijkaardige optie gevonden werd, als tweede wanneer de ruimte die de gebruiker opgaf te klein is om een van de opties in te plaatsten en als laatste, als de gebruiker geen punten getekend heeft wanneer deze anders aangeduid had. In elk van de drie gevallen wordt een specifieke foutmelding getoond.



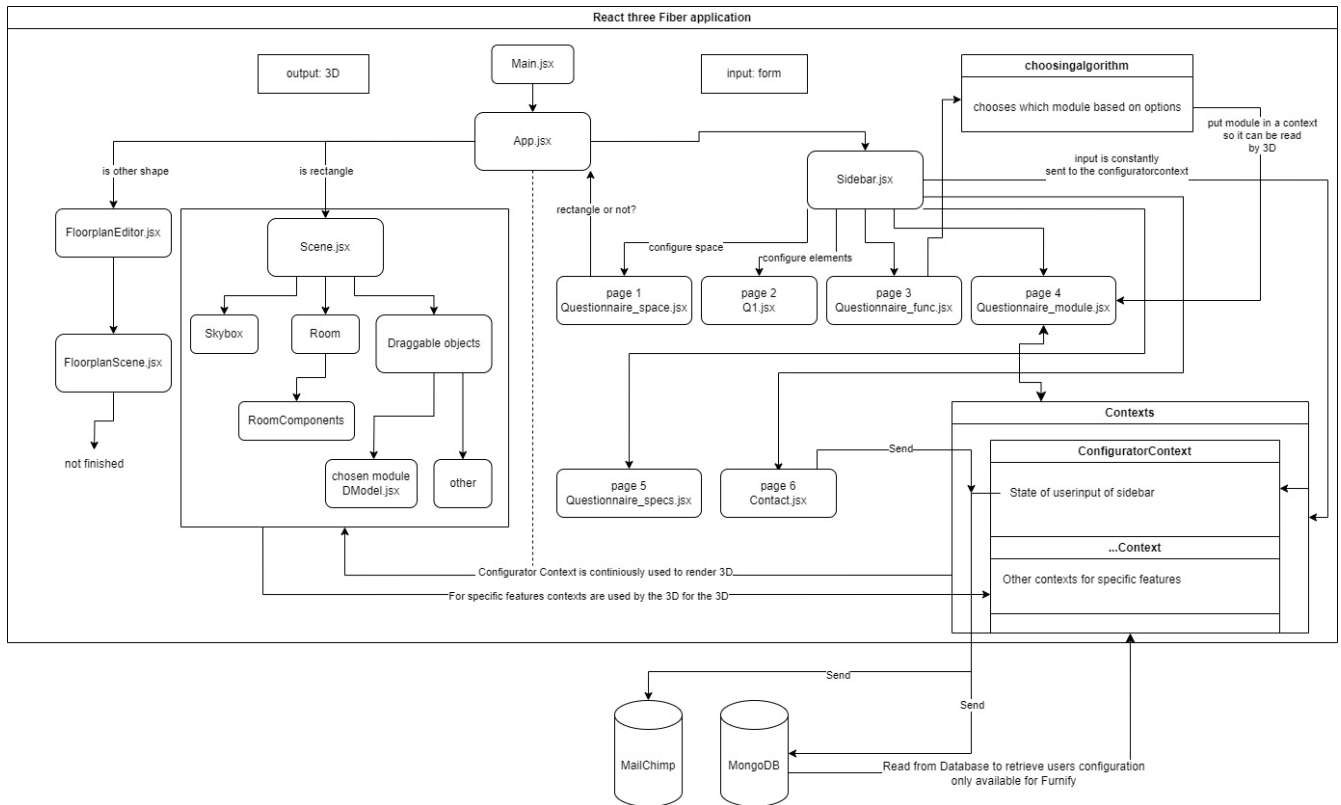
*Figuur 7: deeltoestanden voor het tonen van de modules*

Hierna volgt de vragenlijst over de specificaties en de contact pagina. Op beide lijsten kan de gebruiker nog dingen ingeven maar de toestand van de pagina zelf zal niet veranderen.



## Opbouw applicatie

React volgt een component gebaseerde architectuur. Ieder element is voorgesteld door een aparte component, die doorheen de applicatie kan hergebruikt worden. De onderlinge opbouw van belangrijke componenten in de webapp is vrij complex en wordt hier samengevat op figuur 8. De belangrijkste componenten worden kort toegelicht.



Figuur 8: architectuur van de applicatie

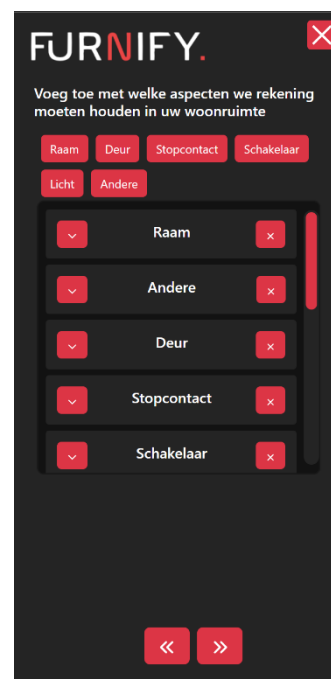
### App.jsx en Main.jsx

Het bestand `Main.jsx` is het startpunt van de applicatie. Het maakt gebruik van de methode `ReactDOM.createRoot` om de applicatie te renderen in het DOM-element met het id: 'root'. Binnen deze methode worden verschillende *contextproviders* gewikkeld rond de *Routing*-component. Deze providers stellen verschillende stukken toestand en bijhorende functies beschikbaar aan de componenten in de applicatie. Wat deze contexten precies zijn wordt later uitgelegd. `App.jsx` is dan de compositie van wat initieel te zien is op het startscherm. Enerzijds de 3D-omgeving links en anderzijds het zijscherm rechts.

## Zijscherm

Het zijscherm is gedefinieerd in de map *sidebar*. Deze bevat verschillende componenten die samen het zijscherm van de app vormen zoals op figuur 9. `Sidebar.jsx` is de hoofdcomponent van het zijscherm. Het beheert de toestand en bevat logica voor het navigeren tussen de verschillende delen.

De verschillende delen bevatten verschillende formulieren die de input van de gebruiker kunnen ontvangen. Deze input wordt later opgehaald door de configuratorcontext. `Contact.jsx` bevat het laatste deel van het zijscherm en zal de contactgegevens van de klant verzamelen en opsturen naar een mailinglijst beheerd door Mailchimp. Indien dit is gelukt, wordt deze ook opgeslagen in de databank.



Figuur 9: zijscherm

## Databank

Wegens een laattijdige vereiste van de opdrachtgever werd in de laatste weken nog een databank geïmplementeerd. De gebruiker kan de ingegeven data in de applicatie op het einde verzenden naar Furnify Home door een connectie met Mailchimp. Deze data wordt echter ook bewaard in een MongoDB database. Zo kan Furnify Home zelf de kamers van de verschillende gebruikers achteraf ophalen en nog eens bekijken. Er wordt later in het verslag dieper ingegaan op de databank.

De keuze om de gegevens pas naar de databank door te sturen nadat het versturen naar Mailchimp is geslaagd, is gemaakt omdat Mailchimp zelf kan aangeven dat sommige velden niet zijn ingevuld. Op deze manier is de data op de in de databank in sync met die van mailchimp

## Contexten

Een context in React is een manier om de data en toestand van een parameter bij te houden en deze beschikbaar te stellen doorheen de componentenboom zonder dat ze als eigenschappen moeten doorgegeven worden. Dit is zeer handig aangezien data kan gedeeld worden tussen componenten die niet direct verwant zijn aan elkaar.

In de applicatie werd gebruik gemaakt van verschillende contexten. De `configuratorContext` is een belangrijke context in de applicatie. Deze haalt de data op uit het zijscherm, zal deze opslaan in zijn toestand en vervolgens wordt, bij het renderen van de 3D-omgeving, deze data van de context gelezen en gebruikt om de visuele 3D-componenten op te maken. Dit is duidelijk in figuur 8.

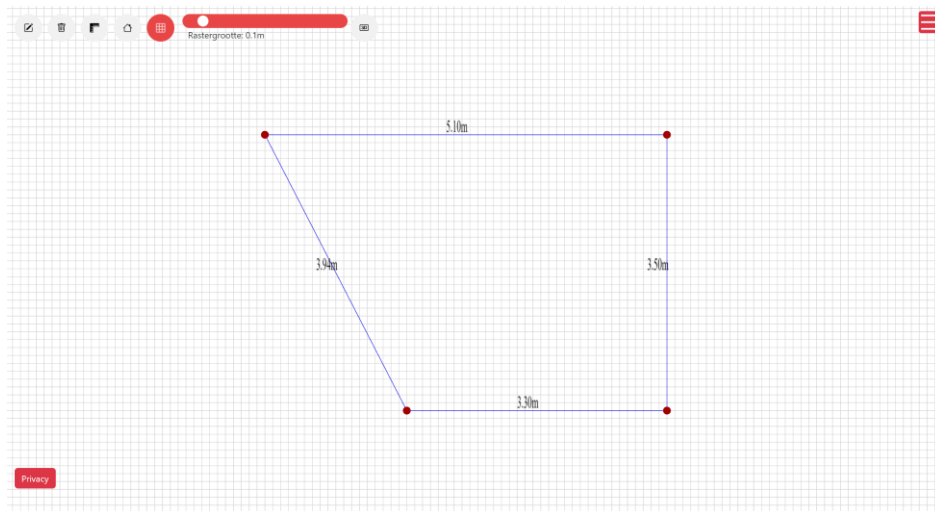
Nog een toepassing van context was de `intersectionContext`. Deze werd gebruikt bij de detectie van overlapping tussen verschillende 3D-*meshes* in de scène. Dit zal een referentie naar alle versleepbare obstructies bijhouden in de 3D. Ook bevat deze een functie, die checkt op overlapping, en een lijst met kubussen die intersectie hebben. De context zorgt er dus voor dat al deze elementen beschikbaar zijn over de volledige app. De werking van deze overlappingsdetectie wordt later gedetailleerder beschreven in het verslag.

## i18N

Ook internationalisatie is toegevoegd. In het `i18n.ts` bestand bevindt zich een geneste structuur van sleutel-waarde paren, die alle vertalingen bevat in verschillende talen, gebruikt in onder andere het zijscherm. Delen van het zijscherm zullen deze dus opvragen. De juiste vertaling wordt dan gegeven dankzij een detector die de taal van de browser zoekt en gebruikt. Dit werd geïmplementeerd met de *i18next*-bibliotheek van React.

## 2D

De componenten in de 2D-map voorzien het 2D-grondplan. `FloorplanEditor.tsx` bevat al de logica voor het gebruik van het vloerplan, zichtbaar op figuur 10. `FloorplanScene.jsx` is dan de component voor de compositie van de 3D-ruimte gebaseerd op dat plan. De gegevens tussen deze twee worden uitgewisseld met behulp van de 2D-context. Deze context bevat ook de logica voor de omzetting naar 3D. Dit maakt het makkelijk om via de scene gebruiker acties op te vangen en de gegevens hiervan in de context te plaatsen. De 'editor' logica wordt dan uitgevoerd als deze gegevens veranderen en kan dan bijvoorbeeld een 3D punt genereren op de plaats waar een gebruiker geklikt heeft en dit toevoegen aan de puntenlijst in de context. De scene reageert op de aanpassing in de puntenlijst door ze te renderen. Herbruikbare componenten zoals punten en lijnen zitten in de *submap*, 'components'.



*Figuur 10: vloerplan*

### 3D

De 3D-map bevat alle componenten voor de creatie van de 3D-omgeving, zichtbaar op figuur 11, weliswaar met de rechthoekige kamer. Deze maakt dus geen gebruik van de 2D-grondplanontwerper. De belangrijkste component hier is `Scene.jsx`. Dit bevat de compositie van de omgeving die de gebruiker te zien krijgt. Deze leest de data uit de `configuratorContext` en stelt zo een omgeving samen op basis van die waarden.



*Figuur 11: 3D-omgeving*

Binnen deze 3D-folder zijn nog tal van componenten gedefinieerd die bijdragen aan de visualisatie. Zo bevat `roomComponents` de componenten voor de kamer. `Draggables` bevat alle versleepbare modellen en obstructies. Al deze objecten worden dan gebruikt in de scène met de waarden uit de context.

## Bepalingsalgoritme

Het bepalingsalgoritme is terug te vinden in de folder `algorithm`. Het bestaat uit drie bestanden waar een bewuste opdeling tussen werd gemaakt. Het eerste bestand `read_file_csv.ts` zorgt ervoor dat de producten ingelezen kunnen worden. Het tweede bestand `module.ts` is een klasse waarin alle informatie van een product gestoken kan worden. Ook de logica voor het nagaan of de kamer groot genoeg is, is toegevoegd in deze klasse. Het derde bestand `module_choice.ts` bevat twee functies, de eerste zal de beslissing van de modules maken en de tweede zal de functie uit `read_file_csv.ts` oproepen. Bij het maken van de beslissingen zal er gekeken worden naar de informatie die aangeduid werd op de website. Voor het vergelijken van de types en afmetingen van de modules zullen de functies uit `module.ts` opgeroepen worden.

Deze opsplitsing zorgt ervoor dat het beslissingsalgoritme in `module_choice.ts` niet op de hoogte moet zijn van de specifieke logica in de andere bestanden. Er kunnen aanpassingen gedaan worden aan de manier dat informatie over het product opgeslagen en verwerkt wordt, zonder dat dit gevolgen heeft voor `module_choice.ts`. Analoog kan het inlezen van alle informatie van de producten, ook aangepast worden zonder gevolgen voor de algemene logica.

## Hoofdstuk 3: Testplan

### End-to-end

De *front-end* applicatie wordt getest aan de hand van *end-to-end-testing* (E2E-testing). Hiermee is het mogelijk het gedrag van een eindgebruiker na te bootsen en te bekijken of de applicatie op de interactie reageert zoals ervan wordt verwacht. Deze keuze is gemaakt omdat *end-to-end testing* de volledige samenwerking tussen alle componenten test, waardoor ook afzonderlijke componenten indirect worden getest door de resultaten van de samenwerking.

De tests zijn geautomatiseerd wat het voordeel biedt dat bij toevoegingen van nieuwe functies de tests steeds opnieuw kunnen worden uitgevoerd. Volgens Test Guild (2023) zijn de top 3 automatisatie testtools Selenium, Cypress en PlayWright. Voor dit project is Cypress gebruikt.

### Cypress

Cypress is een opensource *end-to-end testing framework* dat ook gebruikt kan worden voor *component*-, *integration*- en *unit testing*, hoewel dat laatste minder eenvoudig is. Dit *framework* ondersteunt het testen van JavaScript en TypeScript toestanden, wat het ideaal maakt voor dit project.

In tegenstelling tot Selenium is het namelijk eenvoudiger en sneller in gebruik, doordat het directe toegang heeft tot de DOM van de browser zonder een externe *web driver* nodig te hebben. Dit zorgt voor snellere en efficiëntere communicatie met de browser tijdens het uitvoeren van tests. Bovendien biedt Cypress de mogelijkheid om mee te kijken met de testuitvoering terwijl deze plaatsvindt, waardoor ontwikkelaars eenvoudiger fouten kunnen opsporen dan bij PlayWright.

Een nadeel van Cypress is dat het enkel kan testen op Chromium-gebaseerde browsers. Het is dus niet mogelijk om de tests te laten uitvoeren via Safari. Dit wordt opgelost door Apple-gebruikers de webapplicatie te laten uitproberen.

### Doel tests

Met de geschreven tests wordt getracht de handelingen van de eindgebruiker zo nauwkeurig mogelijk na te bootsen. Dit omvat onder andere het toevoegen van obstakels. Ook wordt nagegaan of, afhankelijk van de browsertaal, de correcte tekst steeds wordt weergegeven. De testsuite bevat ook controles voor foutieve invoer zoals een telefoonnummer met te veel cijfers, letters of negatieve waarden bij een afmeting. Belangrijk is ook het behoud van data. Zo mogen aangebrachte wijzigingen niet verdwijnen bij het navigeren naar een volgende sectie van de vragenlijst.

Hoewel Cypress zich focust op DOM-elementen kan de 3D-visualisatie ook worden getest. Hiervoor is er nood aan een extra Cypress-plugin. De controle gebeurt dan aan de hand van regressie waarbij het verkregen beeld wordt vergeleken met een snapshot. Dit is echter niet meer toegepast op het project, omdat er geen tijd meer beschikbaar was.

De tests worden tijdens het ontwikkelproces voornamelijk gebruikt om na te gaan of nieuwe toevoegingen er niet voor hebben gezorgd dat oude functionaliteiten niet werken of dat het samenvoegen van een *repository* geen problemen oplevert. Deze worden uitgevoerd na iedere *commit*, maar kunnen ook lokaal worden uitgevoerd. Nieuwe tests werden uitgeschreven na het toevoegen van een nieuwe *feature*. Tijdens het uitwerken van een feature worden eventuele fouten opgespoord door de webapplicatie lokaal te runnen en door het medeteamgenoten ook te laten testen.

Het nabootsen van de handelingen van de eindgebruiker is uiteraard niet voldoende. Daarom worden personen die niets afweten van het ontwikkelproces, gevraagd om de applicatie uit te testen en het van feedback te voorzien. Dit gaat niet enkel over mogelijke fouten, maar ook over de gebruiksvriendelijkheid van de applicatie. Zo werd er opgemerkt dat het openklappen van een obstakel niet vanzelfsprekend was en dat de vragen van de vragenlijst onduidelijk geformuleerd waren.

## Stop

Na sprint 2 zijn er echter geen geautomatiseerde testen meer toegevoegd. Dit komt omdat nog een groot deel van de implementaties moest worden toegevoegd en de tijdsdruk te groot was. Verder moesten bestaande testen ook vaak worden aangepast doordat er wijzigingen werden doorgevoerd wat betreft het deel waar het aan bod komt op de zijdscherm bijvoorbeeld. De applicatie werd wel nog steeds getest, maar dit werd door teamgenoten gedaan.

## Locatie

De testbestanden bevinden zich binnen de map “project/cypress/e2e/” en kunnen worden uitgevoerd met behulp van volgende commando's:


```
npm run dev
```

```
npm run test
```

Na het uitvoeren van deze commando's opent er zich een scherm waarop gevraagd wordt welk soort testing er moet worden uitgevoerd. Dit wordt ook weergegeven op figuur 12. Na het selecteren van “E2E Testing” en de browser waarop de testen dienen te worden uitgevoerd. Kan er in specs gekozen worden welk testbestand er moet worden uitgevoerd figuur 13 en wordt deze uitgevoerd.

# Welcome to Cypress!


[Review the differences between each testing type →](#)



### E2E Testing

Build and test the entire experience of your application from end-to-end to ensure each flow matches your expectations.

● Configured

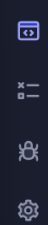


### Component Testing

Build and test your components from your design system in isolation in order to ensure each state matches your expectations.

● Not Configured

Figuur 12: Cypress



7 matches [+ New spec](#)

E2E

Component ?

Last updated ?

Runs ?

▼ cypress\e2e

▼ langCheck

Dutch.cyjs

2 days ago

--

English.cyjs

2 days ago

--

French.cyjs

2 days ago

--

German.cyjs

2 days ago

--

Italian.cyjs

2 days ago

--

Spanish.cyjs

2 days ago

--

Functionalities.cyjs

2 weeks ago

--

Figuur 13: Cypress overview



## Hoofdstuk 4: Evaluatie en discussies

### Performantie

Bij het ontwikkelen van de 3D applicatie is veel aandacht besteed aan de optimalisatie van de performantie. Hieronder worden enkele van de belangrijkste maatregelen en technieken beschreven die zijn toegepast om een soepele gebruikerservaring te garanderen.

#### Computergraphics

De applicatie is gebouwd in React Three Fiber (R3F) als *wrapper* voor WebGL en three.js. 3D-computergraphics genereren vergt veel rekenkracht van de computer. De keuze voor R3F biedt veel voordelen van reactieve declaratieve componenten zonder in te boeten aan prestaties. R3F is niet inherent trager dan standaard three.js en stelt de krachtige features van React in staat om benut te worden, zoals de reactieve renderingscyclus en de uitgebreide ecosysteemondersteuning. Dit helpt om de *codebase* proper en onderhoudbaar te houden terwijl hoogwaardige 3d-visualisaties worden geleverd. (R3F, z.j.)

Achterliggend wordt, ingebouwd in three.js, WebGL gebruikt. Dit zorgt ervoor dat *computergraphics* kunnen weergegeven worden zonder een speciale plug-in voor de webbrowser. Dit maakt hardware-acceleratie mogelijk indien de computer een grafische kaart bevat. (WebGL, 2022)

#### Vite

Als *build-tool* en ontwikkelserver werd gebruik gemaakt van Vite. Vite staat bekend om zijn snelle opstarttijden en efficiënte *hot module replacement* (HMR). Dit alles biedt een heel vlotte ontwikkelervaring. De geoptimaliseerde *build-output* van Vite zorgt ervoor dat de applicatie snel laadt en soepel presteert in productieomgevingen. (Vitejs, z.j.)

#### Aandachtspunten bij implementeren

Ook werd er rekening gehouden met het efficiënt beheer van de *rendercyclus* in React. In plaats van overmatig gebruik te maken van de `useFrame-hook`, die bij elke frame opnieuw wordt aangeroepen, wordt voornamelijk gebruik gemaakt van de `useEffect-hook`. `useEffect` wordt alleen geactiveerd bij specifieke wijzingen, waardoor de *rendercyclus* efficiënter wordt beheerd en onnodige herbewerkingen, wanneer de applicatie in rust is, worden vermeden. Dit resulteert in een lagere CPU-belasting.

Het raster dat getoond kan worden bij het tekenen in 2D vergt veel rekenkracht, aangezien er honderden lijntjes moeten worden gegenereerd. Om deze complexe berekening wat simpeler te maken, worden eerst alle lijntjes in één vorm samengevoegd. Op die manier moet maar één

vorm gegenereerd worden in plaats van honderden kleine. Ook wordt de `useMemo-hook` gebruikt, die ervoor zorgt dat deze complexe berekening in het geheugen wordt bijgehouden waardoor hij slechts eenmaal uitgevoerd moet worden en daarna terug opgehaald kan worden.

Hoewel 4k texturen indrukwekkende visuele resultaten zouden opleveren, is ervoor gekozen om lichtere texturen te gebruiken. Dit helpt om laadtijden te verminderen en onnodige belasting van de GPU, wat vooral belangrijk is voor gebruikers met minder krachtige hardware. De afweging tussen beeldkwaliteit en prestatie is iets waar continue rekening mee werd gehouden.

### Laadscherm

Tot slot werd een laadscherm toegevoegd om de gebruikerservaring te verbeteren tijdens het laden van grote 3D-modellen en texturen. Dit laadscherm wordt weergegeven in het begin van de applicatie en telkens wanneer zware *assets* of de *scene* opnieuw geladen moeten worden. Als de zwaardere *assets* al in de *cache* zitten, worden deze direct opgeroepen.

Door deze optimalisaties en technieken toe te passen, is een webapplicatie ontwikkeld die niet alleen visueel indrukwekkend is, maar ook efficiënt presteert. Dit stelt in staat om een hoogwaardige 3D configuratie-ervaring te bieden aan gebruikers, ongeacht hun hardwareconfiguratie. Iedere recente gsm of laptop is in staat om de applicatie vlot te gebruiken.

### Beveiliging databank

Het is voor Furnify Home mogelijk om het resultaat van de eindgebruiker achteraf te bekijken. Dit kan gedaan worden door achter het webadres het mailadres van de eindgebruiker toe te voeren. Uiteraard is het niet de bedoeling dat anderen hieraan kunnen en daarom is er ook een login-pagina voorzien. De klant dient eerst in te loggen voordat er een *GET-request* wordt uitgevoerd om de gegevens in te laden. Dit zorgt ervoor dat de gegevens niet opgehaald kunnen worden tenzij de login succesvol is. Indien een e-mailadres niet wordt teruggevonden tijdens het inloggen, wordt de klant omgeleid naar de startpagina van de webapplicatie.

## Schaalbaarheid

### Extra modellen

Aangezien Furnify Home een groeiend bedrijf is, is het te verwachten dat in de toekomst de aangeboden producten uitgebreid zullen worden. Het is daarom belangrijk om hier al mee rekening te houden in de code. Alle producten worden bijgehouden in een komma gescheiden (CSV) bestand. Ondanks de naam is ervoor gekozen om het bestand te scheiden door punt komma's, dit om problemen met kommagetallen te vermeiden. Het bestand is zichtbaar op figuur 14. De rijen stellen de producten voor, terwijl de kolommen de eigenschappen van de producten voorstellen.

naam	hoogte	diepte	breedte140	breedte160	breedte180	dicht	open	besparing	opklapbed	zetel	bureau	kast_met_zijtschappen	kast	bed_bewegend	kast_bewegend	bureau_bewegend	tweede_kast_bewegend
opklapbed	2211	400	1685	1741	1941	400	2190	4	true	false	false	false	false	false	false	false	false
opklapbed_zetel	2211	400	1685	1741	1941	1225	2190	10	true	true	false	false	false	false	false	false	false
bureau	2211	400	1685	1885	2085	400	1000	2	false	false	true	false	false	false	false	false	false
kast_zij	2211	400	1685	1885	2085	400	400	2	false	false	false	true	false	false	false	false	false
kast	2211	400	1685	1885	2085	400	400	2	false	false	false	false	true	false	false	false	false
bureau_bedm	2211	800	1685	1885	2085	800	2750	6	false	false	true	false	false	true	false	false	false
kast_bedm	2211	800	1685	1885	2085	800	2750	5.5	false	false	false	false	true	true	false	false	false
kast_kastm	2211	800	1685	1885	2085	800	1600	3	false	false	false	false	true	false	true	false	false
bureau_kastm	2211	800	1685	1885	2085	800	1600	5	false	false	true	false	false	false	true	false	false
bureauum_bedm_kast	2211	800	1685	1885	2085	1200	3150	8	false	false	false	false	true	true	false	true	false
bureauum_kastm_kast	2211	800	1685	1885	2085	1200	3150	5	false	false	false	false	true	false	true	true	false
bureauum_bedm_kastm_kast	2211	800	1685	1885	2085	1600	3550	12	false	false	false	false	true	true	true	true	false
kastm_bedm_kastm_kast	2211	800	1685	1885	2085	1600	3550	10	false	false	false	false	true	true	true	false	true

*Figuur 14: weergave CSV-bestand in Excel*

Dit bestand wordt telkens opnieuw ingelezen bij het opstarten van de site, op deze manier zal bij wijzigingen de meest recente versie gebruikt worden. Een extra product, opgebouwd uit bestaande componenten, toevoegen kan eenvoudig worden gedaan door in het CSV-bestand een nieuwe rij toe te voegen. Er moet dan wel een 3D-module voorzien worden volgens de manier die in handleiding beschreven staat en de nodige vertalingen moeten worden voorzien in het i18n-bestand. De namen zelf worden nergens in de code hard gecodeerd.

Het is ook mogelijk voor Furnify Home om extra eigenschappen toe te voegen. In het geval dat dit gebeurd zijn er twee gevallen. De eerste en meest eenvoudigste is als er een extra soort meubel toegevoegd wordt, maar in een al bestaande categorie. In dit geval volstaat het om dit in de Module klasse toe te voegen. Het tweede geval is dat er een nieuwe categorie zou bijkomen, dan moet dit wel op meerder plaatsen aangepast worden. Een categorie wordt niet in het CSV gezet maar wordt wel gebruikt om de gebruiker de gewenste functionaliteit te laten kiezen. In beide gevallen moet ook de juiste vertaling voorzien worden.

### Extra kleuren en houttexturen

De kleuren en houttexturen worden dynamisch toegevoegd aan het model. Dit zorgt ervoor dat er op dit moment slechts één model nodig is voor zes combinaties van eenzelfde model te tonen (drie houttexturen en twee kleuren). Door het dynamisch toevoegen van kleur en houttexturen aan het model is het voor de opdrachtgever later ook zeker mogelijk om nieuwe kleuren en/of houttexturen voor modellen te introduceren.

### Afmetingen van het model

De modellen die gemaakt zijn voldoen steeds aan de kleinste afmetingen van het model. Indien een gebruiker een bredere matras wenst voor een bed, dienen de modellen ook breder getoond te worden in de 3D-omgeving. Door het bijhouden van de verschillende breedtes in de CSV is het dan ook mogelijk om de module in de webapplicatie te schalen. Indien de opdrachtgever later een nog bredere module zou wensen hoeft deze dan ook geen extra modellen te voorzien. Dit zorgt ervoor dat de webapplicatie relatief snel kan blijven werken en niet te groot waardoor het ook nog op minder rekenkrachtige toestellen gebruikt kan worden.

## GDPR-analyse

De website maakt gebruik van een MongoDB-databank en stuurt gegevens naar Mailchimp. Bovendien wordt ook de privacy policy van Furnify Home zelf in rekening gehouden.

### MongoDB Databank

De MongoDB databank verzamelt de gegevens die de gebruiker invult in het formulier op de website. Die gegevens bevatten persoonlijke informatie en de gegevens om het 3D-model op te bouwen. De gegevens worden opgestuurd naar de databank nadat de gebruiker zijn gegevens succesvol heeft laten verzenden naar Mailchimp en kunnen enkel opgehaald worden door een *admin* nadat die zich heeft kunnen inloggen op de loginpagina. Dit zorgt ervoor dat de gegevens beveiligd zijn.

### Mailchimp

Wanneer gebruikers een formulier op de website invullen en verzenden, worden hun gegevens automatisch naar een Mailchimp contactenlijst gestuurd. De gegevens worden gebruikt voor het versturen van nieuwsbrieven en marketingcommunicatie in overeenstemming met de *GDPR*. Toestemming wordt gegeven via een *double opt-in*. Dit betekent dat de gebruiker eerst moet bevestigen door op de *submit* knop te drukken in het formulier. Vervolgens wordt er een bevestiging mail doorgestuurd naar de gebruiker om te bevestigen dat de gebruiker zich wil inschrijven. Wat garandeert dat de gebruiker expliciet toestemming geeft om toegevoegd te worden aan de contactenlijst.

### Furnify

De website bevat ook de privacyverklaring van Furnify. Hierin wordt toegelicht hoe Furnify Home de gegevens van de gebruikers verwerkt en welke verschillende diensten Furnify Home gebruiken voor de gegevensverwerking.

## Duurzaamheid

In de inleiding is al vermeld dat Furnify Home een website wilde ontwikkelen om de verplaatsing naar de klant voor opmetingen onnodig te maken en zo hun CO<sub>2</sub>-uitstoot te verminderen. Helaas zal dit niet alle uitstoot oplossen, aangezien het gebruik van internet ook CO<sub>2</sub> uitstoot. Daarom is tijdens dit project ook rekening gehouden met duurzame ontwikkelingsdoelstellingen (SDG's), met name de richtlijnen voor duurzaamheid op het web (WSG's).

De eerste richtlijn voor duurzaamheid op het web (WSG) waarmee rekening gehouden is, is het vermijden van geduplicateerde code. Het hergebruiken van code is duurzamer en maakt het opsporen van fouten eenvoudiger, wat leidt tot minder uitstoot (3.6 Avoid Code Duplication, z.j.). In het project werd gebruik gemaakt van types in CSS voor verschillende elementen die zichtbaar zijn op de website. Daarnaast werden er functies gecreëerd die kunnen worden opgeroepen, zodat dezelfde code niet herhaaldelijk hoeft te worden geschreven. Hoewel er in de code veel componenten zijn die sterk op elkaar lijken, konden deze door kleine verschillen niet altijd worden samengevoegd en hergebruikt.

Als tweede WSG is er ook rekening gehouden met het vermijden van onnodige of een overvloed aan middelen. Het is belangrijk om te zorgen dat de website niet te vaak opnieuw moet worden ingeladen en dat er een visueel simpele interface aangeboden wordt (2.7 Avoid Unnecessary or an Overabundance of Assets z.j.). De site is zeer afhankelijk van de 3D voorstelling, om te zorgen dat de impact hiervan beperkt wordt, is ervoor gekozen om objecten in de ruimte zo simpel mogelijk te houden. Het doel van de objecten die toegevoegd kunnen worden, is immers om aan te geven waar de module niet geplaatst kan worden. Er is daarom gekozen om deze weer te geven aan de hand van een balk en geen specifiek model voor het object zelf te voorzien. Er is wel gekozen om de lichten effectief licht te laten geven zodat de gebruiker wel weet of de belichting in de kamer beïnvloed zou kunnen worden door de module.

Hiernaast is zoals eerder al vermeld ook de keuze gemaakt om te werken met React. Op deze manier zal bij het wisselen tussen pagina's van de vragenlijst enkel de pagina herladen worden en niet de volledige 3D-visualisatie.

De laatste WSG waar er rekening mee is gehouden is het respecteren van de gebruiker zijn aandacht. De criteria die hier bij horen zijn dat de gebruiker gemakkelijk controle moet hebben tot hoe en wanneer hij informatie krijgt. Ook is het best om oneindig scrollen te vermijden en de tijd dat ze op de site spenderen te beperken (2.9 Respect the Visitor's Attention z.j.). Dit wordt gerealiseerd door de vragen die de klant moet invullen over zijn ruimte op te splitsen in meerdere pagina's en het zo overzichtelijk te houden. De gebruiker zal op deze manier niet moeten scrollen en slechts een paar vragen tegelijkertijd moeten beantwoorden.

## Algemene problemen

### Databank

Na de presentatie van de tweede sprint aan de opdrachtgever werd er meegedeeld dat de opdrachtgever graag de mogelijkheid kreeg om de resultaten van de gebruiker ook achteraf visueel te kunnen zien. Dit werd niet aangegeven door de opdrachtgever zelf waardoor dit via mail nog eens werd nagevraagd.

De opdrachtgever beantwoordde deze mail met de boodschap dit fantastisch te vinden. Hierdoor is er op het laatste moment nog een databank geïmplementeerd. Als deze behoefte eerder bekend was geweest, kon de projectstructuur er beter op worden afgestemd.

### Communicatie opdrachtgever

De communicatie met de opdrachtgever verliep stroef. Er moest lang gewacht worden op antwoorden via mail (soms meer dan een week) en de antwoorden waren vaak niet verhelderend. Ook werd er aan de opdrachtgever gevraagd voor modellen van de modules te voorzien en werden deze vaak laattijdig en verkeerd geleverd. Zo werd bijvoorbeeld donderdag 16/05/2024 het verkeerde model nog opgestuurd en werd een dag later, vlak voor het einde van het project, het juiste model gestuurd, maar met de verkeerde afmetingen.

Ook gaf de opdrachtgever geen feedback terwijl dit juist zeer waardevol zou zijn geweest om te weten wat er niet goed is en/of zou moeten worden aangepast.

### Niet rechthoekige kamers

Er werd pas later gestart met het implementeren van de optie om ook andere vormen van kamers te voorzien dan enkel rechthoekig. De prioriteit werd gegeven aan de rechthoekige kamer. Het verschil in implementatie tussen de rechthoekige en de anders vormige variant is groot, waardoor deze functie niet volledig is uitgewerkt. De applicatie is voorlopig dus enkel volledig bruikbaar met rechthoekige kamers.

## Specifieke moeilijkheden bij de implementaties

### Verslepen

Eén van de complexe toevoegingen binnenin de 3D is het verslepen. Verslepen zorgt voor een zeer goede gebruikservaring voor het bepalen van de plaats van de module. Daarom is dit ook een belangrijke toevoeging. Bepaalde componenten zijn versleepbaar zoals de gekozen module, maar ook bepaalde types obstructies. De lamp langs het plafond en bijvoorbeeld een radiator over de vloer zijn versleepbaar.

Het implementeren van de versleepfunctie is volledig zelf ontworpen, aangezien gelijkaardige functies uit bepaalde bibliotheken niet voldoen aan de verwachtingen. Het verslepen is dan ook ontworpen in drie delen.

Het eerste deel is de `Surface`. Dit is een component die kinderen kan hebben. Die kinderen zijn dan bijvoorbeeld de module en de verslaapbare obstructies. De `Surface` zorgt ervoor dat al deze kinderen toegang krijgen tot de code voor de versleepfunctionaliteit.

Ook moet iedere versleepbare component een `onDrag` functie bevatten die het versleepgedrag van die component definieert. Dit zal de functionaliteit bevatten om de positie van de *mesh* van toepassing, telkens bij te werken naargelang de locatie van de cursor.

Ten slotte is ook een algemene `useDrag` functie gedefinieerd. Deze luistert naar de muis-*events* en zal de `onDrag` van toepassing aanroepen om te bepalen wat er met de specifieke *mesh*-component gebeurt.

Een vereiste van de opdrachtgever was dat de gekozen furnify-module steeds aan een muur moest hangen. Daarom werd bij het verslepen van de module een *snapping*-functionaliteit toegevoegd. Deze werd geïmplementeerd door eerst de afstanden tot iedere muur te berekenen. Dan werd de muur met de kortste afstand bepaald en vervolgens de positie geüpdatet tegen die muur. Extra moeilijk hierbij was rekening houden met de rotatie van de module.



## Roteren

Roteren werd dus tot nu toe enkel voor rechthoekige kamers geïmplementeerd. Dat wil zeggen dat er enkel over een hoek van 90 graden moet geroteerd worden. De rotatiehoek wordt ook bijgehouden in de configuratorcontext. Afhankelijk van die hoek moet dus ook met veel dingen rekening gehouden worden. Bijvoorbeeld: het verslepen, plakken aan muren, de kast mag niet buiten de ruimte worden geroteerd, ...

Dit alles werd opgelost aan de hand van `useEffect-hooks`. Dit is een functie die toegang geeft tot toestand en *lifecycle* van React. Meer specifiek kan de `useEffect` ervoor zorgen dat er een 'side effect' gebeurt op basis van een veranderende waarde. Als bijvoorbeeld de rotatiewaarde ergens veranderd in de configuratorcontext, dan zal de code binnen de `useEffect` opnieuw uitgevoerd worden. Zo wordt telkens de positie van de kast herberekend binnen de ruimte als er zou geroteerd worden.

## Kamer achteraf verkleinen

In de vragenlijst kan steeds worden teruggekeerd om antwoorden aan te passen. Daar moet de 3D telkens rekening mee houden. Een kamer kan bijvoorbeeld al ontworpen zijn en bepaalde objecten bevatten. Als dan terug wordt gekeerd in het invulformulier kan achteraf nog de grootte van de kamer aangepast worden. Indien de posities van de objecten dezelfde zouden blijven en de kamer verkleind, dan kan het zijn dat objecten buiten de kamer belanden.

`UseEffect` is hier opnieuw de oplossing. Zo worden opnieuw posities berekend als de dimensies van de kamer veranderen in de `configuratorContext`.

## Ramen en deuren

Bij het ontwerpen van de applicatie werd veel belang gehecht aan de visuele voorstelling. Schaduw en licht speelden hierbij een belangrijke rol. Het maakt de ervaring realistischer en zorgt voor een beter beeld bij de gebruiker. Daarom werd er gekozen om de ramen en deuren effectief als een holte in de muur te beschouwen, zodat deze licht konden doorlaten. Dit zorgt voor een mooi effect. Dit was echter niet zo simpel aangezien dit type *mesh* niet zomaar bestaat in *three.js*. Er zijn veel manieren, maar de ideale oplossing voor deze situatie was het gebruik van Constructive Solid Geometry (CSG). Dit laat het optellen of aftrekken van *meshes* toe. Hierdoor kan de vorm van een raam afgetrokken worden van de muur, zodat er een muur met een holte, met de grootte van het raam, ontstaat.

## Overlappende objecten

Als één van de laatste belangrijke implementaties werd detectie van overlapping geïmplementeerd. Tot voordien was het nog steeds mogelijk om verschillende objecten in elkaar te plaatsen. Om dit te voorkomen bij het eindresultaat werd een knop toegevoegd, waarmee de gebruiker kan controleren of niets in de compositie overlapt met elkaar. Indien dit wel zo is, wordt een melding opgeroepen en zal de overlapping visueel zichtbaar worden gemaakt. De gebruiker kan dan de opstelling verplaatsen en nogmaals checken.

De implementatie gaat als volgt. Er is een nieuwe context gedefinieerd die alle aanwezige *meshes* van de scène ophaalt. Deze worden in een lijst gestoken. Dan zal de functie, `checkIntersections`, alle *boundingboxes* berekenen van de objecten en checken op overlapping. De overlappende boxen worden toegevoegd aan een lijst met objecten, die visueel worden weergegeven na het checken.

## Probleem met één tick vertraging

Na de tussentijdse feedback werd de opmerking gegeven dat bij het verkleinen of vergroten van de kamer de objecten, met één tick vertraging mee bewogen. Dat is niet de bedoeling. Eerst werd gedacht dat dit door een fout in de *gameloop* kwam.

Na enige tijd zoeken bleek dat dit niet het geval was. De oorzaak was een *easing*-functie, die ervoor zorgde dat tijdens het verslepen objecten langzaam starten, versnellen, en dan weer langzaam vertragen. Dit droeg bij aan de gebruikerservaring, aangezien het lineair verslepen nogal stroef overkomt.

Het bleek dus dat dit niet alleen effect had op het verslepen, maar op alle beweging van objecten. Daardoor leek het dus dat de objecten vertraagd meebewogen bij aanpassingen aan de kamer of roteren. De oplossing was dus de *easing*-functie enkel te activeren bij het verslepen. Om het resultaat nog vloeiender te maken, werd de functie ook nog een halve seconde na het verslepen actief gehouden, zodat ook het einde van de beweging vloeiend vertraagt.

## Coördinaten muis in 3D

Omdat er niet direct in de HTML DOM wordt gewerkt, maar in de Three.js-omgeving, is het noodzakelijk om de 3D-coördinaten van de muis te kunnen bepalen binnen deze omgeving. Om dit te realiseren wordt gebruikgemaakt van *raycasting*. *Raycasting* houdt in dat vanuit een bekend punt, zoals de camera, een virtuele straal (*ray*) wordt uitgezonden door de muispositie op het scherm. De 3D-muiscoördinaat is het punt waar deze straal het vlak  $Z = 0$  (het XY-vlak), dat gedefinieerd is in de 3D-ruimte, snijdt.

## Overlap muren en vloer

Bij de conversie van 2D naar 3D, worden de muren gegenereerd op basis van opeenvolgende punten die de grenzen van de kamer vormen. Elk paar opeenvolgende punten wordt gebruikt om de lengte en oriëntatie van een muur te bepalen. De muren worden als balken weergegeven.

Om een muur correct te positioneren, wordt eerst het middelpunt tussen de twee punten berekend. Dit middelpunt bepaalt de positie van de balk. Vervolgens wordt de juiste richting van de balk vastgesteld door middel van goniometrie. Hierbij wordt de hoek tussen de punten en de X-as berekend, zodat de muur correct georiënteerd wordt.

Het grootste probleem dat hierbij optrad, was dat de muren voor de helft overlaptten met de vloer. Dit kwam doordat het middelpunt van de balk zich precies op het lijnstuk tussen de twee punten bevond, waardoor de muur gedeeltelijk in de vloer terechtkwam.

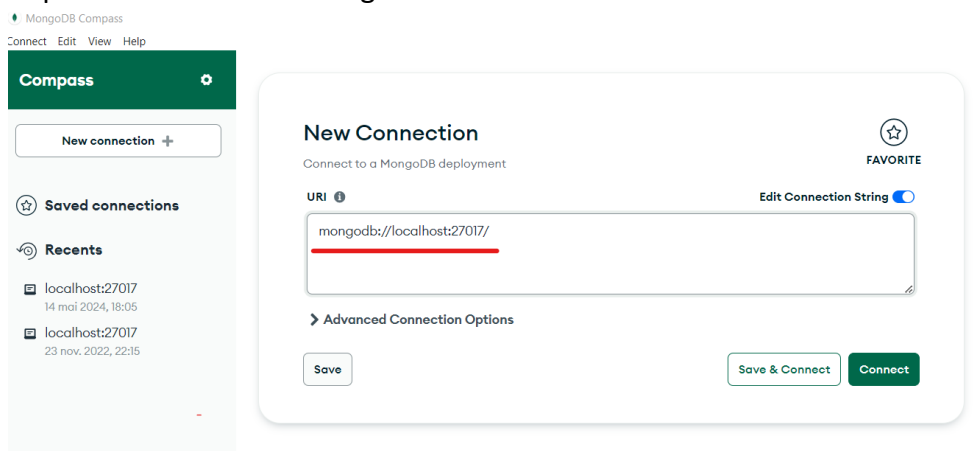
De oplossing voor dit probleem bestond uit het bepalen van de normaalvector van elk muursegment. Voordat dit kon worden gedaan, moest echter eerst worden vastgesteld of de punten van de kamer met de klok mee of tegen de klok in waren getekend. Dit was nodig om de zin van de normaalvector correct te bepalen, zodat deze naar buiten wijst. De normaalvector is een vector die loodrecht staat op het muursegment. Nadat deze normaalvector is berekend, wordt de muur voor de helft van de muurdikte verplaatst in de richting van deze normaalvector. Hierdoor wordt de muur volledig buiten de vloer geplaatst, zonder overlap.

## Hoofdstuk 5: Handleidingen

### Applicatie vereisten

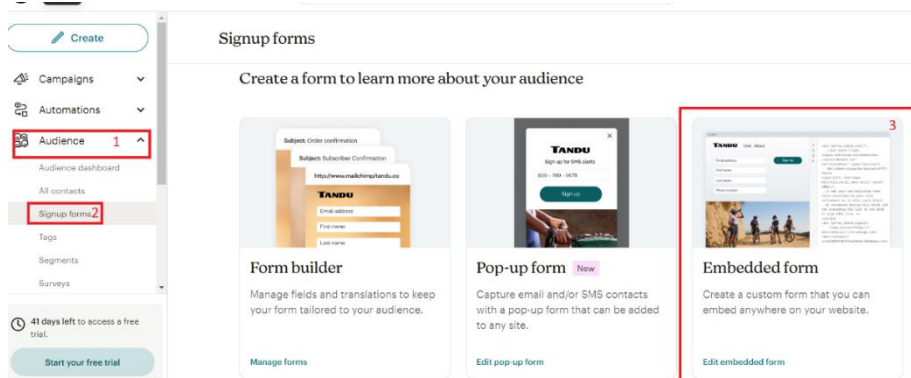
De applicatie vereist een mongoDB databank en een mailchimp account.

- MongoDB
  1. Download mongoDB versie 7.0.9
  2. Volg de installatie-instructies voor u besturingssysteem
  3. Kopieert de connectie string die u ziet staan



*Figuur 15: MongoDB connectiestring*

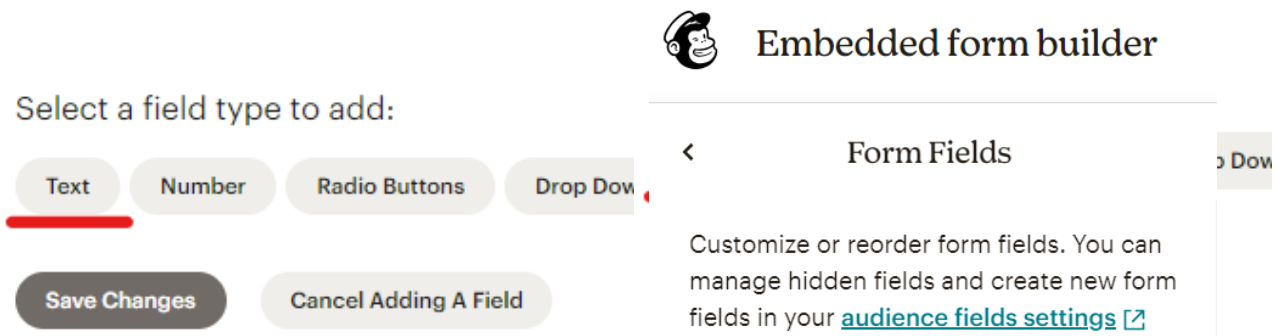
- Mailchimp
  1. Maak een Mailchimp-account
  2. Ga naar 'audience' > 'signup form' > 'embedded form'



*Figuur 16: voorbeeld MongoDB*

3.

Ga naar 'Form Fields' > 'audience fields settings' en voeg een *textfield* voor van deze tags: FIRSTNAME, LASTNAME, ADDRESS, DIMENSIONS, ROOM, LAYOUT,



*Figuur 17: type tekstvelden (links) en formulievelden (rechts)*

MATERIAL, COLOR, FUNCTIONAL REQUIREMENTS, MODULE, POSTCODE, CITY, en COUNTRY.

4. Uiteindelijk moet uw lijst er zo uitzien (als de tags niet exact dezelfde zijn zal het niet mogelijk zijn om de data te verzenden naar mailchimp)

Field label and type	Required?	Visible?	Put this tag in your content:	Default merge tag value
Adresse e-mail email	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* EMAIL * or * MERGE0 *	
Prénom text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* FIRSTNAME * or * MERGE1 *	Default merge tag val.
Nom de famille text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* LASTNAME * or * MERGE2 *	Default merge tag val.
Address text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* ADDRESS * or * MERGE4 *	Default merge tag val.
Dimensions text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* DIMENSIONS * or * MERGE5 *	Default merge tag val.
Room text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* ROOM * or * MERGE6 *	Default merge tag val.
Layout text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* LAYOUT * or * MERGE7 *	Default merge tag val.
Material text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* MATERIAL * or * MERGE8 *	Default merge tag val.
Color text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* COLOR * or * MERGE9 *	Default merge tag val.
Functionalities text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* FUNCTIONAL * or * MERGE10 *	Default merge tag val.
Requirements text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* REQ * or * MERGE11 *	Default merge tag val.
Module text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* MODULE * or * MERGE13 *	Default merge tag val.
postcode text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* POSTCODE * or * MERGE14 *	Default merge tag val.
country text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* COUNTRY * or * MERGE15 *	Default merge tag val.
city text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	* CITY * or * MERGE16 *	Default merge tag val.

*Figuur 18: afgewerkt mailchimp formulier*

5. Keer terug op embedded form builder en druk op continue

The screenshot shows the Mailchimp Embedded Form Builder interface. On the left, there is a sidebar with 'Form Fields', 'Settings', and 'Tags'. On the right, there is a preview of a 'Subscribe' form. The form has three input fields: 'Adresse e-mail', 'Prénom', and 'Nom de famille'. A red asterisk indicates required fields. The 'Continue' button is highlighted with a red circle.


*Figuur 19: formulier afwerken*

6. Kopieer de link van form action en verander post van de link in post-json

# Your form is ready!

Copy & paste this code into your website's HTML where you want the form to appear.

Embedded Form Code

 [Copy Code](#)

```
</style>
<div id="mc_embed_signup">
  <form action="https://hotmail.us18.list-manage.com/subscribe/post?
u=dbf86de75caa0bdaee7da1262&id=18a2dee28f&v_id=13&f_id=
00ed11e1f0" method="post" id="mc-embedded-subscribe-form" name="mc-
embedded-subscribe-form" class="validate" target="_blank">
```

*Figuur 20: formulier url*

https://hotmail.us18.list-  
manage.com/subscribe/**post**?u=dbf86de75caa0bdaee7da1262&id=18a2dee2  
8f&v\_id=13&f\_id=00ed11e1f0

wordt

https://hotmail.us18.list-manage.com/subscribe/**post-**  
**json**?u=dbf86de75caa0bdaee7da1262&id=18a2dee28f&v\_id=13&f  
\_id=00ed11e1f0

## Applicatie bereiken

Een standaard webbrowser, zoals Chrome, Firefox of Safari volstaat om de applicatie te bereiken. Deze is momenteel ontplooid op een Linux-server van de UGent met IP-adres: 157.193.171.41. Surfen naar de website kan dus via gewoonweg <http://157.193.171.41>.

## Applicatie uittesten in ontwikkelingsomgeving

Tijdens de ontwikkeling van de applicatie werd gebruik gemaakt van Vite. Vite is een lokale ontwikkelingsserver en wordt vaak gebruikt bij React-projecten. Volg de volgende stappen om de applicatie lokaal uit te testen:

1. Indien u het project als een .zip bestand hebt, unzippt u dit en opent u de nieuwe map "project".
2. Open nu een terminal in deze map (Hulp nodig hierbij? Klik op [deze link](#)) en sla stap 3, 4, 5 en 6 over en doe verder bij stap 7.
3. Installeer git (indien u dit nog niet heeft): <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
4. Indien u toegang hebt tot gitlab: Zorg dat de *gitlab-repository* lokaal staat.
  - a. Open een terminal venster.
  - b. Voer het volgende commando uit: `git clone [repository-name]`
5. Ga in een terminal venster naar de map van de *repository*.
6. Voor de meest recente versie van de code te verkrijgen: `git pull`
7. Verifieer of u effectief in de project map zit door het commando `dir` uit te voeren in de terminal. Als u de mappen 'src', 'public' en nog wat andere configuratiebestanden ziet staan zit u in de correcte map.
8. Installeer Node en npm (indien u dit nog niet heeft):  
<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>
9. Voeg uw MongoDB connection string en Mailchimp link in .env (link tussen de twee dubbel aanhalingsteken)
  - a. *ATLAS\_URI = "MongoDB connection string"*



b. `MC_URI = "Mailchimp link"`

10. Pas de username en password in `.env` aan (link tussen de twee dubbel aanhalingsteken)

a. `VITE_ADMIN = "uw username"`

b. `VITE_PASSWORD = "uw password"`

11. Installeer de juiste *packages* met het volgende commando: `npm install`

12. Run de applicatie lokaal met het volgende commando: `npm run dev`

13. Start de MongoDB server: `npm run server`

## Applicatie zelf ontplooiën op een linux server

Zelf de site *builden* en ontplooiën op een linux server met een *apache2-webserver* is ook iets wat mogelijk wordt gemaakt door Vite. Dit kan via volgende commando's:

1. Zorg dat MongoDB versie 7.0.9 op de server geïnstalleerd is.
2. Indien u het project als een .zip bestand hebt, unzippt u deze en opent u de nieuwe map "project". Sla stap 2, 3 en 4 over en doe verder bij stap 7.
3. Installeer git (indien u dit nog niet heeft): <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
4. Indien u toegang heeft tot gitlab: Zorg dat de *gitlab-repository* lokaal staat.
  - a. Open een terminal venster.
  - b. Voer het volgende commando uit: `git clone [repository-name]`
5. Ga in een terminal venster naar de map van de *repository*.
6. Voor de meest recente versie van de code te verkrijgen: `git pull`
7. Verifieer of u effectief in de project map zit door het commando `ls` uit te voeren in de terminal. Als u de mappen 'src', 'public' en nog wat andere configuratiebestanden ziet staan zit u in de correcte map.
8. Installeer Node en npm (indien u dit nog niet hebt):  
<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

9. Installeer de juiste *packages* met het volgende commando: `npm install`
10. Voer `npm run build` uit.
11. In de root van de server voer deze commando's uit om de site op de apache2 server te zetten:
  - a. `sudo rm -rf /var/www/html/*`
  - b. `sudo cp -r furnishify2/project/dist/* /var/www/html`
12. Start de MongoDB server: `npm run server`
13. Om de *productionbuild* lokaal te *previewen*, gebruik: `npm run preview`

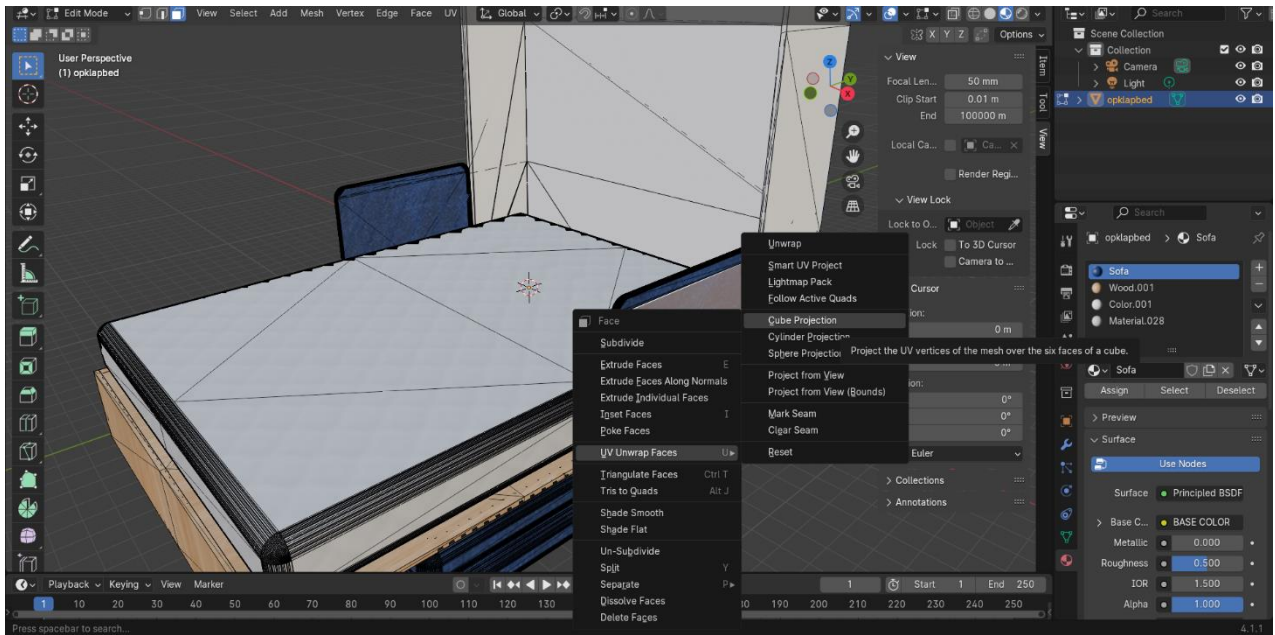
## Toevoegen/aanpassen eisen van een model

1. Open het bestand 'meubels.csv'. Deze kan teruggevonden via het pad "project/public/meubels.csv". Dit ziet er als volgt uit.

naam	hoogte	diepte	breedte140	breedte160	breedte180	dicht	open	besparing	opklapbed	zetel	bureau	kast_met_zijenschappen	kast	bed_bewegend	kast_bewe	bureau_be	tweede_kast_bewegend
opklapbed	2211	400	1541	1741	1941	400	2190	4	true	false	false	false	false	false	false	false	false
opklapbed_zetel	2211	400	1541	1741	1941	1225	2190	10	true	true	false	false	false	false	false	false	false
bureau	2211	400	1685	1885	2085	400	1000	2	false	false	true	false	false	false	false	false	false
kast_zij	2211	400	1685	1885	2085	400	400	2	false	false	false	true	false	false	false	false	false
kast	2211	400	1685	1885	2085	400	400	2	false	false	false	false	true	false	false	false	false
bureau_bedm	2211	800	1685	1885	2085	800	2750	6	false	false	true	false	false	true	false	false	false
kast_bedm	2211	800	1685	1885	2085	800	2750	5,5	false	false	false	false	true	true	false	false	false
kast_kastb	2211	800	1685	1885	2085	800	1600	3	false	false	false	false	true	false	true	false	false
bureau_kastm	2211	800	1685	1885	2085	800	1600	5	false	false	true	false	false	false	true	false	false
bureauum_bedm_kast	2211	800	1685	1885	2085	1200	3150	8	false	false	false	false	true	true	false	true	false
bureauum_kastm_kast	2211	800	1685	1885	2085	1200	3150	5	false	false	false	false	true	false	true	true	false
bureauum_bedm_kastm_kast	2211	800	1685	1885	2085	1600	3550	12	false	false	false	false	true	true	true	true	false
kastm_bedm_kastm_kast	2211	800	1685	1885	2085	1600	3550	10	false	false	false	false	true	true	true	false	true

Figuur 21: meubels.csv bestand

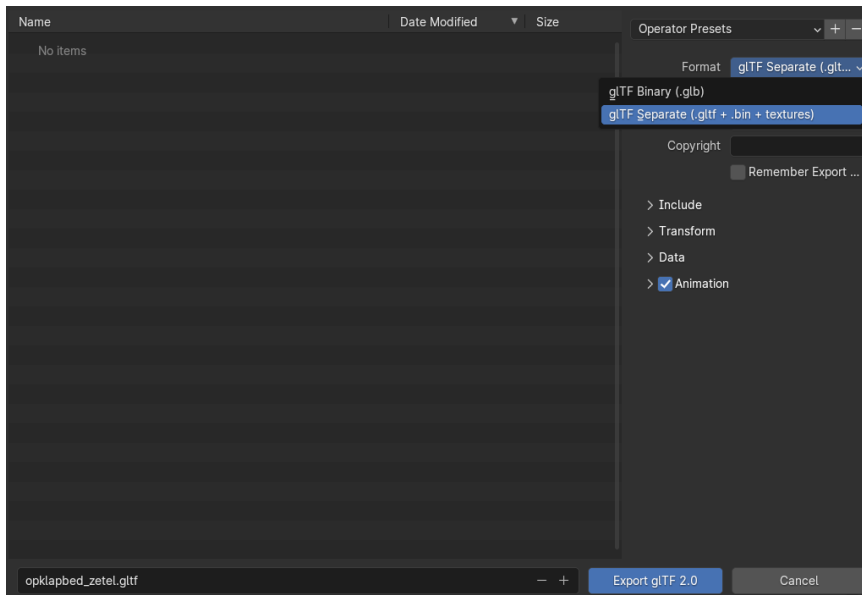
2. Voeg een nieuwe rij toe/pas een rij aan in het bestand en voer voor iedere kolom de bijbehorende waarde toe. Zorg ervoor dat kommagetallen steeds een punt bevatten in plaats van een komma.  
bv. 3.9
3. Open Blender en importeer uw model als .stl-bestand. Het .stl-bestand dient qua afmetingen overeen te komen met deze die zijn opgegeven in het CSV-bestand voor breedte140. Open moet dus gelijk zijn aan Y, X moet gelijk zijn aan breedte140 en Z moet gelijk zijn aan hoogte.
4. Ga naar Edit Mode, selecteer de vlakken en voeg voor iedere vlak de juiste materiaal toe.
  - a. Indien het materiaal een Image Texture gebruikt, zorg dan dat deze ".jpg" als extensie heeft
  - i. Zorg er ook voor dat het vlak een *Cube Projection* krijgt. Dit zorgt voor zichtbaar textuur op het vlak.



*Figuur 22: voorbeeld blender*

- b. Indien het materiaal een houtsoort is dat mag wisselen tussen berk, eik en notelaar steekt u in de naam van het materiaal het woord “Wood”.
  - c. Indien het materiaal een kleur is dat mag wisselen tussen wit en zwart steekt u in de naam van het materiaal het woord “Color”.
5. Zorg ervoor dat het model exact dezelfde naam heeft als de kolom naam in bijhorende rij in de csv.
6. Exporteer het model naar glTF 2.0 en kies voor het formaat glTF Separate. Zorg er hier ook voor dat de naam overeenkomt met deze die is gegeven aan het model en die is

opgeslagen in het CSV-bestand.



*Figuur 23: exporteren in blender*

7. Tot slot voegt u de bestanden (.gltf, .bin en bijhorende afbeeldingen) toe aan de map "project/public/models/" binnenin het project.

## SSL-certificaat, HTTPS en Domeinnaam

Dit deel wordt best uitgevoerd door een IT-verantwoordelijke.

Om de applicatie veilig en professioneel te hosten, zijn de volgende zaken essentieel:

### SSL-certificaat:

Een SSL-certificaat is nodig om de communicatie tussen de gebruikers en de server te beveiligen. Dit zorgt ervoor dat alle gegevens versleuteld worden verstuurd, wat essentieel is voor de privacy en veiligheid van gebruikersinformatie.

Zorg ervoor dat een SSL-certificaat wordt aangevraagd en geïnstalleerd. Dit kan meestal worden gedaan via de hostingprovider of een erkende certificeringsinstantie.

Controleer of de server correct is ingesteld om HTTPS-verbindingen te accepteren en te verwerken.

### Domeinnaam:

Voor een professionele uitstraling en betere toegankelijkheid is een eigen domeinnaam noodzakelijk. Dit maakt het voor gebruikers gemakkelijker om de applicatie te vinden en verhoogt de geloofwaardigheid.

Registreer een nieuwe domeinnaam of gebruik een sub domein binnen het huidige domein `furnifyhome.eu`.

## Gebruikershandleiding

Wanneer de gebruiker de site opent krijgt hij een standaard rechthoekige kamer te zien. Dit is het vertrekpunt maar kan natuurlijk volledig aangepast worden.

1. Selecteer de woonruimte die u wenst te optimaliseren op de sidebar.
2. Selecteer de vorm van de ruimte.
  - a. Rechthoekig: Ga verder naar stap 3.
  - b. Anders:
    - i. Kies of u met het raster wilt werken of niet.
      1. Zo ja: selecteer de gewenste rastergrootte met de slider.
      2. Nee: zet het raster uit (meest rechtse knop).
    - ii. Zet tekenmodus aan (meest linkse knop).
    - iii. Teken de vorm van uw woonruimte en zorg dat u de vorm sluit.  
Het is mogelijk om:
      1. Loodrecht te tekenen met de middelste knop.
      2. Alles te verwijderen met de tweede knop van links.
    - iv. Klik op de net verschenen 3D knop (rechts).
    - v. Bekijk uw kamer in 3D (verdere functionaliteit is hier nog niet toegevoegd).
3. Voer de afmetingen van uw woonruimte in.
4. Kies de indeling van uw woonruimte.
5. Klik op het pijltje vanonder om naar het volgende tabblad te gaan.
6. Voeg aspecten toe in de ruimte, waar rekening mee moet worden gehouden bij het plaatsen van de meubel.
  - a. Klap een aspect open en vul de gevraagde gegevens in.
  - b. Een licht en “andere” aspecten kunnen versleept worden in de 3D omgeving.

- c. Bij een raam, deur, stopcontact en schakelaar kiest u op welke muur deze moet worden geplaatst.

7. Ga naar het volgende tabblad.
8. Selecteer de functies die u graag zou hebben.
9. Ga naar het volgende tabblad.
10. Klik op modules zoeken.
11. U ziet de mogelijke modules die passen bij uw opgegeven specificaties of indien hier geen combinatie van bestaat een zachtere combinatie.
12. Draai en versleep de module zodat deze naar wens gepositioneerd is.
13. Klik op de controleer overlap knop en los eventuele problemen op.
14. Ga naar het volgende tabblad.
15. Geef de gewenste afwerkingen en eventuele andere vereisten aan.
16. Ga naar het volgende tabblad.
17. Vul uw contactgegevens in.
18. Druk op verzenden.

Furnify zelf kan de kamers van zijn klanten opvragen door de URL + /'email' in te vullen in de adresbalk of op de link te klikken die via mailchimp wordt ontvangen per klant. Er dient nog ingelogd te worden op de administratiepagina en dan is de kamer zichtbaar.



## Besluit

Furnify Home, een Gentse startup, heeft als doel modulaire, ruimtebesparende meubels te verkopen en het proces van kameropmetingen en modulebepaling te digitaliseren. Door het ontwikkelen van een webapplicatie wordt het mogelijk voor klanten om zelfstandig de kamers te modelleren en geschikte meubelmodules te selecteren.

De doelstellingen van het project zijn gerealiseerd door het creëren van een gebruiksvriendelijke webapplicatie bestaande uit een 3D-ruimte en een interactieve vragenlijst. De belangrijkste realisaties van het project zijn als volgt:

1. **Ontwikkeling van een responsieve webapplicatie:** Met behulp van React.js is een *frontend* ontwikkeld die goed werkt op verschillende apparaten.
2. **3D-visualisatie met Three.js:** Door Three.js te integreren, is een realistische 3D-omgeving gecreëerd waarin gebruikers de kamers kunnen nabouwen en meubelmodules kunnen plaatsen. Deze visualisatie biedt gebruikers meerdere perspectieven van de ruimte.
3. **Slimme vragenlijst:** De vragenlijst is ontworpen om gebruikersvoorkeuren en kamerafmetingen te verzamelen en deze informatie te gebruiken om geschikte modules te selecteren en in de 3D-ruimte te plaatsen.
4. **Data-opslag en -verwerking:** Een MongoDB-*database* is geïmplementeerd voor het opslaan van kamerconfiguraties. Bovendien worden deze gegevens veilig verzonden naar Mailchimp, waarbij voldaan wordt aan GDPR-regelgeving.
5. **Performantie-optimalisatie:** Door het gebruik van Vite als *build-tool*, React Three Fiber en het implementeren van efficiënte rendercycli in React, draait de applicatie soepel op de meeste hardware.
6. **Gebruikersinteractie en -feedback:** De applicatie biedt gebruikers de mogelijkheid om objecten in de 3D-ruimte te verslepen en te roteren, wat bijdraagt aan een intuïtieve en interactieve gebruikerservaring.

Ondanks enkele uitdagen heeft het project geleid tot een efficiënte en schaalbare oplossing die Furnify Home helpt met het proces van kameropmetingen en modulebepaling.

## Referentielijst

3.6 Avoid Code Duplication (z.j.). Geraadpleegd op 18 april 2024 via <https://sustainablewebdesign.org/guidelines/3-6-avoid-code-duplication/>

2.7 Avoid Unnecessary or an Overabundance of Assets (z.j.). Geraadpleegd op 18 april 2024 via <https://sustainablewebdesign.org/guidelines/2-7-avoid-unnecessary-or-an-overabundance-of-assets/>

Coleman, B. (2017). The Beginner's Guide to Beginning Three.js. Geraadpleegd op 19 februari 2024 via <https://medium.com/@benjamin.c.coleman/the-beginners-guide-to-beginning-three-js-c36b8947c2aa>

Describing the UI. (z.j.). Geraadpleegd op 21 februari 2024 via <https://react.dev/learn/describing-the-ui>

Furnify. (z.j.). Geraadpleegd op 20 februari 2024 via <https://www.furnifyhome.eu>

MongoDB Community Server Download (z.j) Geraadpleegd op 06 mei 2024 via <https://www.mongodb.com/try/download/community-edition>

2.9 Respect the Visitor's Attention (z.j.). Geraadpleegd op 18 april 2024 via <https://sustainablewebdesign.org/guidelines/2-9-respect-the-visitors-attention/>

R3F (z.j.). Geraadpleegd op 18 april 2024 via <https://docs.pmnd.rs/react-three-fiber/getting-started/introduction>

The Top 3 Automation Testing Tools (2023 Edition). (2023). Geraadpleegd op 17 maart 2024 via <https://testguild.com/top-3-automation-testing-tools/>

Vitejs (z.j.). Geraadpleegd op 18 april 2024 via <https://vitejs.dev/>

WebGL (2022). Wikipedia. Geraadpleegd op 18 april 2024 via <https://nl.wikipedia.org/wiki/WebGL>