

ALLOC8 – STAGE FAROS

Realisatie- document

Siebe Michiels

Voorwoord

Het schrijven van deze thesis markeert de voltooiing van een uitdagende en leerzame periode in mijn academische carrière. Deze thesis is het resultaat van mijn stageproject bij Lykios, waarin ik de kans kreeg om een webapplicatie genaamd Alloc8 te ontwikkelen. Alloc8 is ontworpen om een efficiënte en flexibele werkplek oplossing te bieden in het huidige tijdperk van hybride werken. Dit document dient ter afronding van mijn bacheloropleiding Application Development aan Thomas More Kempen.

Doorheen dit academiejaar heb ik gemerkt hoeveel ik ben gegroeid als developer en mijn stage heeft hier enorm veel aan bijgedragen. Gedurende 13 weken heb ik me helemaal mogen verdiepen in de wondere wereld van React en ik ben nu ook zeer trots op het werk dat ik hiermee heb geleverd. Deze stage heeft mij goed voorbereid waardoor ik nu ook op een zelfzekere manier kan starten aan mijn professionele carrière.

Tot slot zou ik nog graag mijn stagementors en de andere collega's bedanken omdat zij steeds bereid waren te helpen waar nodig, zonder hen had deze stage niet mogelijk geweest. Ik ben enorm blij om in zo een gezellige en toch professionele sfeer terecht te zijn gekomen.

Ik wens u veel leesplezier toe.

Siebe Michiels

Lier, 20/06/2024

Inhoud

VOORWOORD	2
INHOUD	3
INLEIDING	4
1. OPDRACHT	5
1.1. Opdrachtschrijving	5
2. BETROKKEN PERSONEN	6
2.1. Lykios	6
2.2. Team	6
3. TECHNOLOGIEËN	7
3.1. Java (Spring Framework)	7
3.2. React.js	7
3.3. Bitbucket	7
4. AANPAK	8
4.1. MoSCoW-methode	8
4.2. Class Diagram	9
5. FUNCTIONALITEITEN	11
5.1. Frontend design	11
5.2. Authenticatie in frontend	12
5.3. Frontend validatie	13
5.4. Kalender weergave	13
5.5. Frontend herwerken	15
5.6. Role-based access control	15
5.7. Teams beheren	16
5.7.1. Gebruiker uitnodigen	16
5.7.2. Team kleur instellen	17
5.8. Plattegrond visualisatie	18
5.8.1. Collision detection en prevention	18
5.8.2. Beheren van bureaus en segmenten	19
5.8.3. Reservatie maken voor bureaus	20
5.9. Soft delete	22
SLOT	23

Inleiding

Tijdens deze stage bij Lykios (Faros) lag de focus op de ontwikkeling van de applicatie genaamd Alloc8, die gebruikers in staat stelt om bureaus op kantoor te reserveren. Deze applicatie is van essentieel belang geworden in het huidige tijdperk van hybride werken, waarbij medewerkers zowel op afstand als op kantoor werken. Alloc8 is ontworpen om situaties te voorkomen waarin bureaus leeg staan of waar medewerkers op kantoor aankomen om vervolgens te ontdekken dat er geen bureaus beschikbaar zijn.

Om dit doel te bereiken, werd een gestructureerde aanpak gehanteerd. Allereerst werd een scope vastgesteld die paste binnen de stageperiode van 13 weken. Dit gebeurde met behulp van een project plan. Deze omvatte een overzicht van de verschillende taken die moesten worden voltooid, een tijdlijn en een aantal deliverables die waren gekoppeld aan vaste mijlpalen in de tijdlijn.

Dit realisatiedocument biedt meer inzicht in het ontwikkelingsproces en de implementatie van Alloc8. Het document onderzoekt de uitdagingen waarmee we geconfronteerd werden, de strategieën die zijn toegepast om deze uitdagingen te overwinnen, en de resultaten die zijn behaald.

1. Opdracht

Tijdens mijn stage heb ik gewerkt aan Alloc8 met als doel een webapplicatie te ontwikkelen die het reserveren van bureaus in kantoorgebouwen mogelijk maakt. In deze tijd van hybride werken, waarin werknemers zowel op kantoor als thuis werken, is er een toenemende behoefte aan flexibele werkplekoplossingen. Alloc8 biedt een oplossing voor deze behoefte.

1.1. Opdrachtomschrijving

Met Alloc8 kunnen gebruikers eenvoudig bureaus in hun kantoorgebouwen reserveren. Dit helpt om werkplekken efficiënt te benutten en te voorkomen dat er te veel of te weinig mensen op kantoor aanwezig zijn. Door een bureau te reserveren, weten werknemers zeker dat er een werkplek voor hen beschikbaar is wanneer ze naar kantoor komen, wat bijdraagt aan een soepelere werkervaring.

Daarnaast biedt Alloc8 beheerders van kantoorgebouwen de mogelijkheid om hun gebouwen en de beschikbare bureaus te beheren via de applicatie. Beheerders kunnen bureaus toevoegen, verwijderen en aanpassen, zodat de beschikbare werkplekken altijd up-to-date zijn. Dit maakt het beheer van de kantoorruimte eenvoudiger en efficiënter, en zorgt ervoor dat de werkplekken optimaal benut worden.

Een andere belangrijke functie van Alloc8 is het beheer van toegang. Beheerders kunnen specifieke gebruikers of teams toegang geven tot bepaalde lokalen of werkplekken. Dit zorgt ervoor dat alleen de juiste personen toegang hebben tot bepaalde gebieden binnen het kantoorgebouw, wat bijdraagt aan de veiligheid en efficiëntie. Door te bepalen wie waar mag werken, kunnen bedrijven hun werkplekken beter organiseren en optimaal gebruiken.

Het hoofddoel van Alloc8 is om de efficiëntie en flexibiliteit van werken te verbeteren. Door een eenvoudige manier te bieden om werkplekken te reserveren en te beheren, kunnen bedrijven hun kantoorruimte optimaal benutten. Dit is vooral belangrijk in een tijd waarin hybride werken de norm is geworden en er een groeiende behoefte is aan flexibele werkplekoplossingen. Alloc8 helpt bedrijven om beter om te gaan met hun werkplekken, waardoor werknemers productiever kunnen zijn en bedrijven efficiënter kunnen opereren. Daarnaast biedt het gemak voor werknemers, die zonder moeite een werkplek kunnen reserveren wanneer ze naar kantoor komen.

2. Betrokken personen

In dit onderdeel beschrijf ik wie er allemaal betrokken was bij deze stage: het stagebedrijf waar ik de kans kreeg om stage te lopen en de mede-stagiair waar ik nauw mee heb moeten samenwerken.

2.1. Lykios

Bij Lykios, een business unit binnen Faros, heb ik de kans gekregen om stage te lopen. Faros is een gerenommeerd bedrijf dat IT-consultancydiensten aanbiedt, gespecialiseerd in diverse server-side Java-technologieën. Faros is een bedrijf binnen de Cronos en Xplore group.

Met meer dan twee decennia ervaring staat Faros bekend om het ontwikkelen van hoogwaardige webgebaseerde en cloud-native applicaties. De consultants bij Faros zijn experts op het gebied van enterprise Java-technologieën.

Kortom, mijn stage bij Lykios heeft me de kans geboden om te werken in een professionele omgeving waar kwaliteit, samenwerking en innovatie centraal staan.

2.2. Team

Doorheen mijn stage heb ik nauw moeten samenwerken met mijn mede-stagiair Thomas de Ceuster, hij heeft zich voornamelijk gefocust op het backend gedeelte in Java terwijl de focus bij mij lag op de frontend in React. Doordat de taken op deze manier waren ingedeeld was een efficiënte communicatie noodzakelijk zodat we steeds op dezelfde golflengte zaten.

3. Technologieën

Om deze applicatie te bouwen moesten we uiteraard weten met welke technologieën we aan de slag moesten. Deze keuzes worden hieronder uitgelegd.

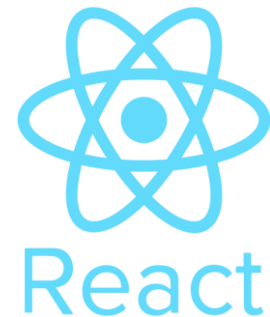
3.1. Java (Spring Framework)

De keuze voor de backend van onze applicatie was beperkt, Faros is een Java consultancy bedrijf dus zou het ook raar zijn om een ander framework te kiezen. Voor de rest kregen we wel volledige vrijheid, zo mochten we een ander framework kiezen om de user authenticatie te maken. Wij kozen hier voor Spring Security omdat dit heel veelzijdig is en dus ook aan onze eisen zou voldoen.



3.2. React.js

Als frontend framework hadden we twee opties, op school zagen we enkel Angular en React. Omdat we beiden vonden dat we al voldoende (school)projecten hadden gemaakt met behulp van Angular besloten we om deze keer voor React te gaan. Zo konden we eens een nieuw framework gebruiken in een realistisch project. Bij Lykios was er ook een collega die veel kennis had over React dus dit was de perfecte manier om sterker te worden in een nieuw framework.



3.3. Bitbucket

Van ons bedrijf kregen we toegang tot een door hun beheerde Bitbucket omgeving, dit is een op Git gebaseerde tool waarmee teams makkelijker kunnen samenwerken aan één software project. Deze tool is gebouwd om pijnloos Jira te kunnen integreren in deze workflow. We maakten dus ook gebruik van Jira om onze sprints te plannen en beheren.



4. Aanpak

In dit onderdeel ga ik dieper in op de werkwijze die we hebben gehanteerd. Op de eerste dag van onze stage kregen we een, door ChatGPT verzonden, beschrijving van de opdracht. Met deze beschrijving zijn we meteen aan de slag gegaan om deze zelf te interpreteren en om te zetten tot een plan van aanpak (ook dit document kan je terugvinden op mijn portfolio).

4.1. MoSCoW-methode

Aan het einde van de eerste dag hadden we de MoSCoW-methode toegepast om een onderscheid te maken tussen noodzakelijke en optionele functionaliteiten binnen de applicatie. Op Figuur 1 vindt u het eindresultaat van deze analyse, overgenomen van het plan van aanpak.

4.1. Must have

- Gebruikers kunnen inloggen
- Gebruikers kunnen een bureau reserveren voor een bepaalde periode of voor een lange tijd
- Gebruikers kunnen een reservatie annuleren
- Gebruikers kunnen een team aanmaken en gebruikers toevoegen aan een team waarvan zij admin zijn
- Team admins kunnen een reservatie maken voor hun team
- Gebouw beheerders kunnen hun gebouw aanmaken, beheren en hier bureaus specificeren

4.2. Should have

- Gebruikers kunnen inloggen met Multi Factor Authenticatie
- Gebouw beheerders kunnen teams toegang geven tot bepaalde bureaus
- Gebouw beheerders kunnen hun gebouw opsplitsen in segmenten (verdieping, kantoren, ...)

4.3. Could have

- Gebouw beheerders kunnen rapporten/analyses opvragen over de bezetting van bureaus
- Gebouw beheerders kunnen prijzen toekennen aan bureaus zodat gebruikers deze kunnen huren
- Gebouw beheerders kunnen segmenten visueel instellen (Konva.js)
- Gebruikers kunnen een visueel plaatje van een gebouw krijgen en hier een bepaald bureau aanduiden om een reservatie te maken
- Kalenderintegratie met third-party apps

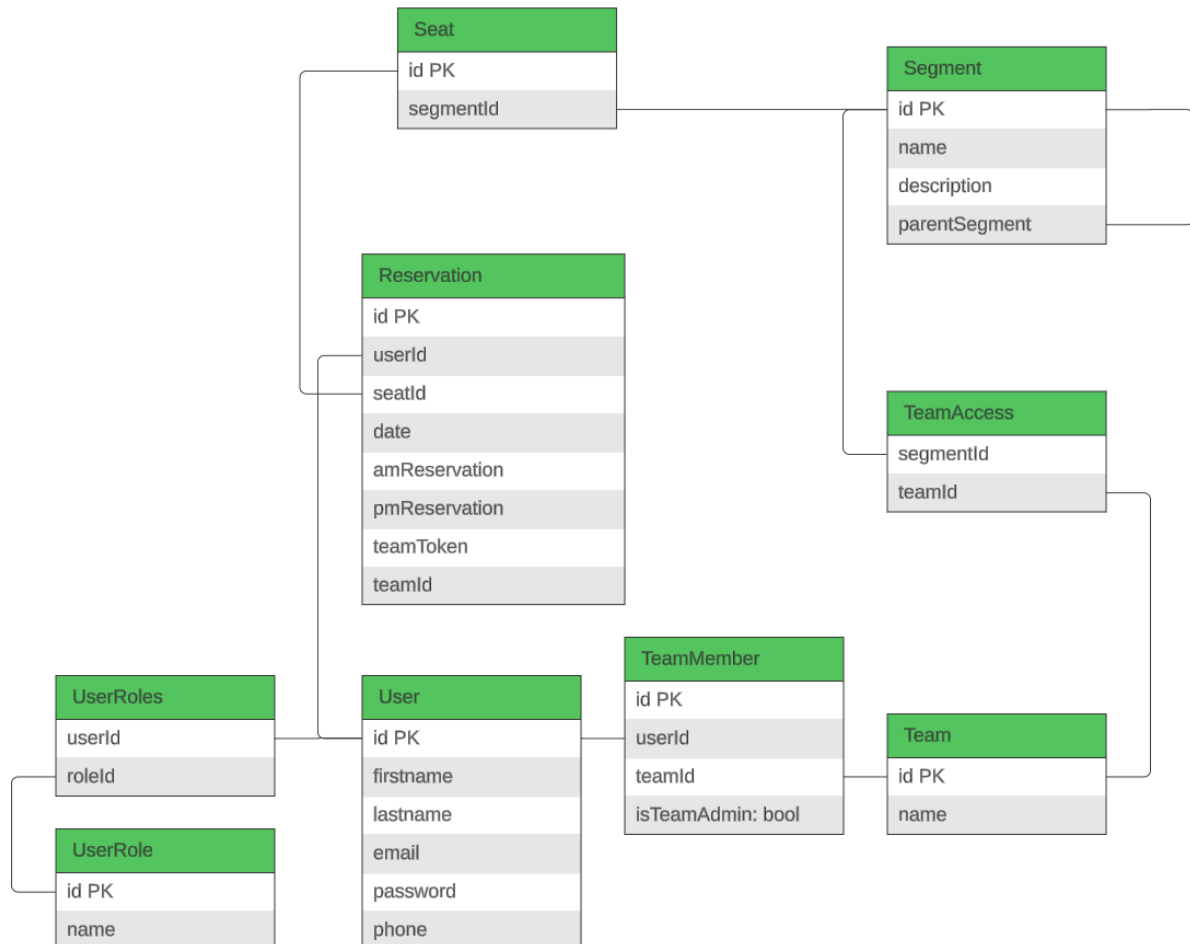
4.4. Won't have

- Lokalisatie

Figuur 1: MoSCoW-analyse

4.2. Class Diagram

Nadat de projectafbakening voltooid was, werd een class diagram opgesteld. Dit is een diagram waarin de structuur van de klassen (en indirect ook die van de database) duidelijk wordt. Op deze manier werd er al een schets gevormd van hoe de applicatie tot stand zou komen. Zo hebben we ook vroegtijdig een aantal problemen ontdekt en opgelost. Op Figuur 2 **Fout! Verwijzingsbron niet gevonden.Fout!** **Verwijzingsbron niet gevonden.** wordt het resultaat weergegeven.



Figuur 2: Class diagram

Omdat dit diagram nogal technisch is, wordt hier onder wat meer uitleg gegeven over iedere klasse. Op die manier is alle terminologie ook duidelijk voor de rest van dit document.

- **User**: dit is een gebruiker die is ingelogd in de applicatie, hiermee kunnen we gebruikers van mekaar onderscheiden.
- **UserRoles en UserRole**: deze klassen houden informatie bij over welke rollen elke gebruiker heeft, een gebruiker kan meerdere rollen hebben. Deze rollen bepalen tot welke data gebruikers toegang hebben en wat zij allemaal kunnen doen binnen de applicatie. De mogelijke rollen zijn:
 - o **Building manager**: deze gebruiker kan zijn kantoorgebouwen beheren en ingeven in de applicatie. Dit betekent dat deze gebruiker segmenten (bv. eerste verdieping of een specifiek lokaal) en de bureaus binnen deze segmenten kan aanmaken en beheren.
 - o **Normal user**: deze gebruiker zal reservaties kunnen aanmaken voor bepaalde bureaus in een kantoorgebouw. Verder kan deze ook nog een team beheren maar hier wordt later nog meer uitleg over gegeven.

- **Team:** in de opdracht stond ook beschreven dat teams moesten worden geïmplementeerd zodat meerdere gebruikers (van bijvoorbeeld hetzelfde bedrijf) een aantal bureaus samen kunnen reserveren.
- **TeamMember:** deze klasse is noodzakelijk om een verbinding te maken tussen **User** en **Team**, daarnaast wordt hier ook bijgehouden of een gebruiker al dan niet admin (beheerder) is van een team. Een team admin kan gebruikers uitnodigen, verwijderen en admin maken van een team.
- **Segment:** zoals eerder al vermeld, segmenten zijn subdivisies binnen een kantoorgebouw. Een segment kan eender wat zijn, van een bepaalde verdieping tot een specifiek lokaaltje tot misschien een aantal bureaus die specifiek voorbehouden zijn voor een bepaald team (zie **TeamAccess**). Met deze segmenten wordt het beheren van een kantoorgebouw veel overzichtelijker.
- **Seat:** deze klasse verwijst naar een bepaald bureau waarvoor gebruikers reservaties kunnen maken.
- **TeamAccess:** gebouw beheerders moeten teams toegang geven tot bepaalde segmenten, op deze manier is er de mogelijkheid om een bepaald lokaal aan een bepaald team toe te wijzen zodat enkel leden van dit team reservaties kunnen maken binnen dit lokaal.
- **Reservation:** bij een reservatie tool zijn de reservaties uiteraard ook van belang, dit wordt in deze klasse bijgehouden. Hierin wordt de datum van de reservatie bijgehouden en of het een reservatie voor de voormiddag, namiddag of de hele dag is (amReservation en pmReservation).

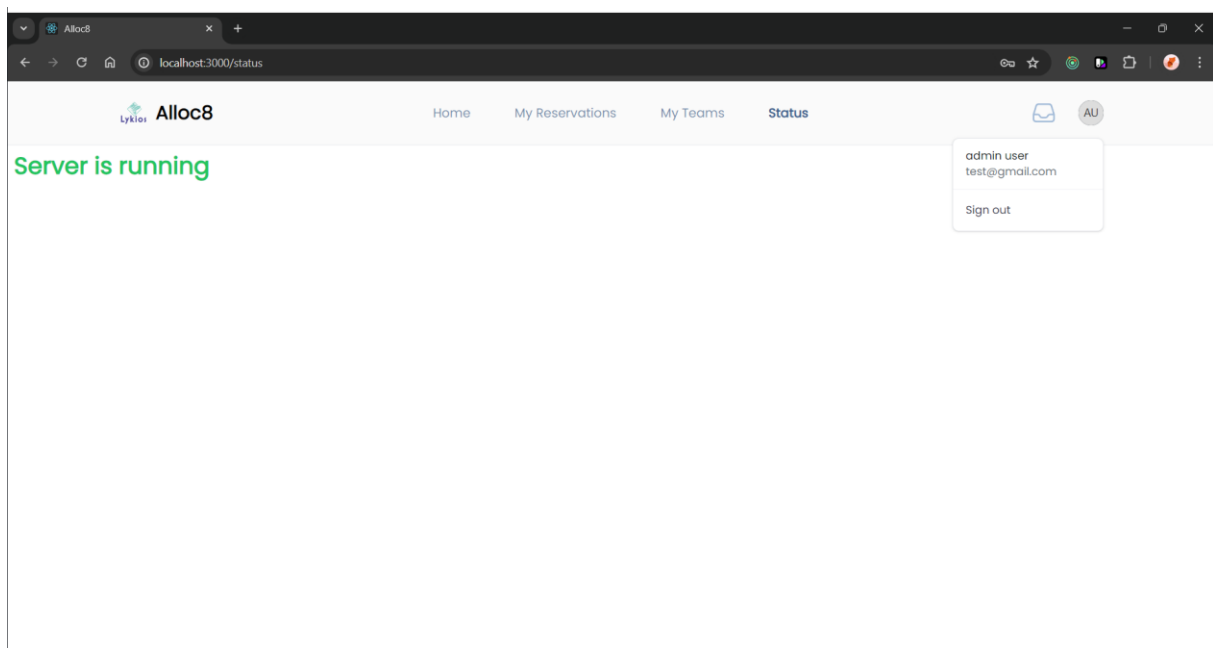
Nu zijn alle klassen van onze applicatie kort uitgelegd, deze termen zullen doorheen het document opnieuw gebruikt worden dus als iets niet duidelijk is kunt u steeds terugverwijzen naar deze uitleg.

5. Functionaliteiten

In dit onderdeel zullen de functionaliteiten worden beschreven waar ik doorheen mijn stage aan heb gewerkt, deze staan in chronologische volgorde.

5.1. Frontend design

Aan het begin van de stage hebben mijn mede stagiair en ik besloten dat ik het frontend project zou opzetten terwijl hij het backend project opzette. Hier heb ik een simpel design gemaakt met een navigatiebalk, in deze navigatiebalk is er ook een dropdown menu rechtsboven waarmee de gebruiker zich kan uitloggen (en in de toekomst mogelijks zijn/haar account kan beheren). Deze navigatiebalk en dropdown menu heb ik in React componenten gezet zodat deze makkelijk opnieuw kunnen worden gebruikt op andere pagina's. Daarnaast hebben we ook een simpele statuspagina gemaakt die controleert of de verbindingen tussen frontend, backend en de database succesvol tot stand zijn gekomen. Dit alles staat afgebeeld op **Fout! Verwijzingsbron niet gevonden..**

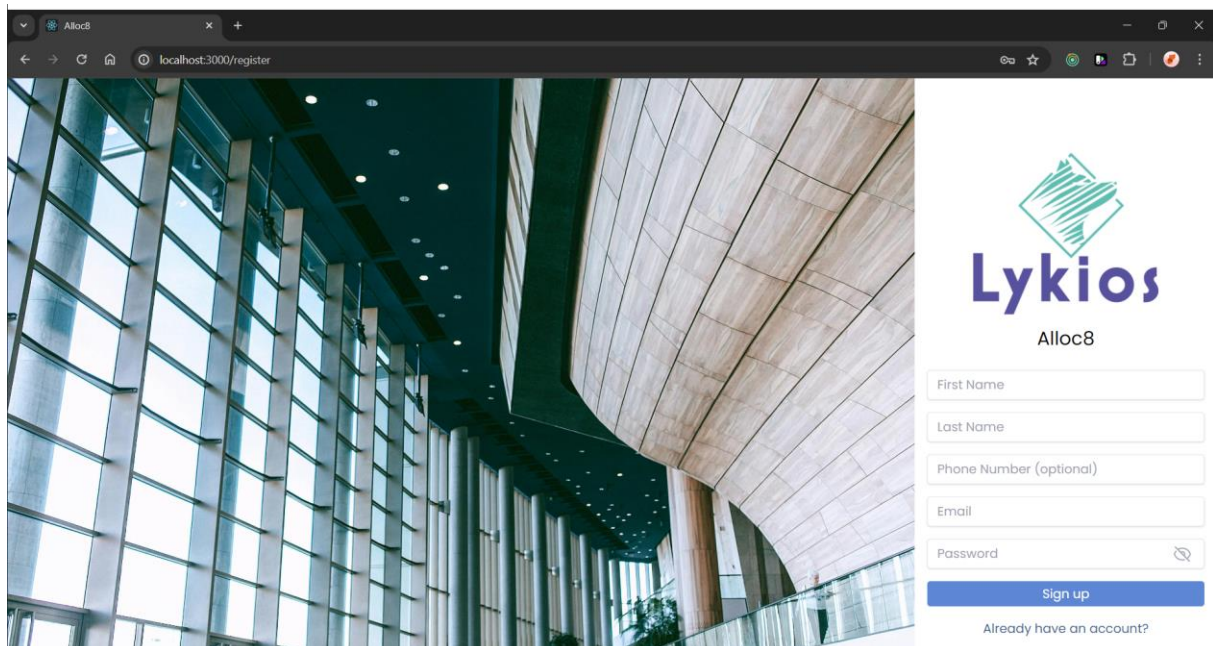


Figuur 3: Statuspagina met navigatiebalk

5.2. Authenticatie in frontend

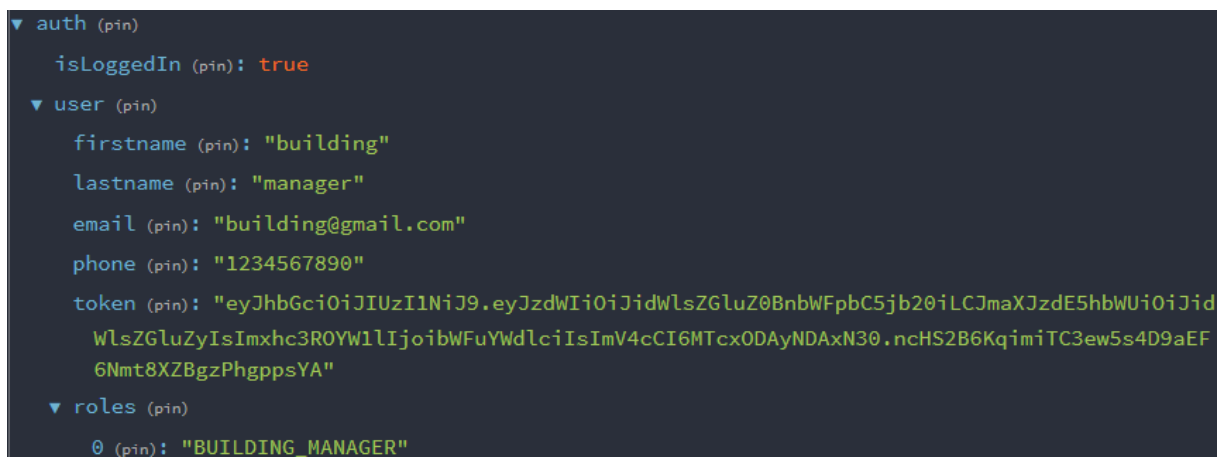
Terwijl mijn mede stagiair zich focuste op de het backend gedeelte voor authenticatie met Spring Security, focuste ik op het frontend gedeelte. Omdat informatie met betrekking tot authenticatie en autorisatie op veel verschillende plaatsen moet worden gebruikt, heb ik besloten om dit in een Redux state te maken. Op deze manier kan er makkelijk worden gecontroleerd of een gebruiker ingelogd is en of deze de rollen heeft die nodig zijn om een bepaalde pagina of informatie te kunnen bekijken.

Op Figuur 4 **Fout! Verwijzingsbron niet gevonden.** ziet u de registreer pagina die ik gemaakt heb, de login pagina ziet er hetzelfde uit maar dan enkel met een *email* en *password* invulveld.



Figuur 4: Registreer pagina

Op Figuur 5 **Fout! Verwijzingsbron niet gevonden.** ziet u dat er een *token* waarde wordt opgeslagen, dit is de JWT-token die bij iedere request naar de backend moet worden meegegeven. Op die manier kan de backend achterhalen welke gebruiker een request heeft gemaakt en dus ook controleren of deze gebruiker wel permissies heeft om bepaalde gegevens op te vragen. Om dit makkelijker te maken heb ik een methode geschreven die automatisch deze JWT-token toevoegt aan iedere request header.



Figuur 5: Redux state voor een ingelogde gebruiker

5.3. Frontend validatie

Omdat bijna ieder invulveld op een webapplicatie één of andere vorm van validatie nodig heeft, heb ik hier een React hook (een soort van component) voor gemaakt zodat we makkelijk validatie kunnen implementeren op ieder invulveld. Op Figuur 6 wordt de validatie op de login pagina getoond.

In deze hook zitten een aantal checks standaard ingebouwd:

- Required: als een veld verplicht is
- Min- en maxLength: om een bepaalde lengte van een waarde in te stellen, zo moet een password bijvoorbeeld minstens 8 karakters bevatten.
- Min: om een minimale waarde in te stellen, dit wordt voornamelijk gebruikt om het doorsturen van negatieve getallen te voorkomen.

Naast deze standaard checks is er ook de mogelijkheid om extra checks toe te voegen die specifiek zijn voor een bepaalde situatie. Een voorbeeld hiervan gebeurt bij het opnieuw instellen van je wachtwoord, hier moet je tweemaal je nieuwe wachtwoord invullen als extra beveiliging, hier is dus een extra validatie check nodig die controleert of beide velden hetzelfde wachtwoord bevatten.



Alloc8

Email is invalid

Password is required

Login

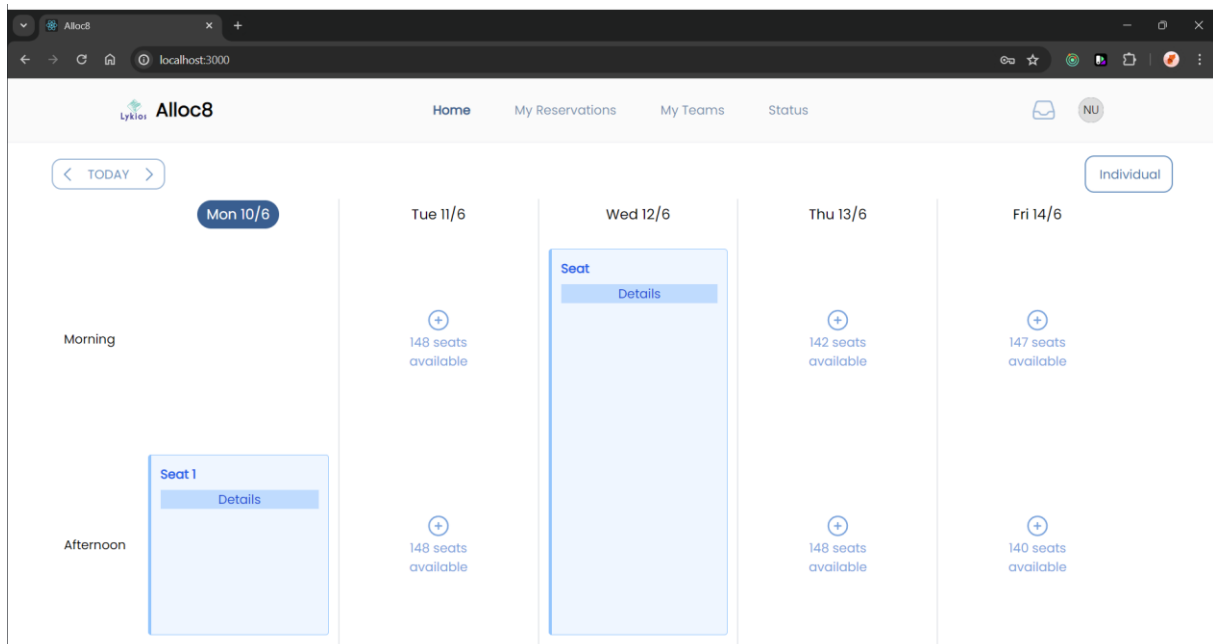
Sign up

[Forgot password?](#)

Figuur 6: Validatie op de login pagina

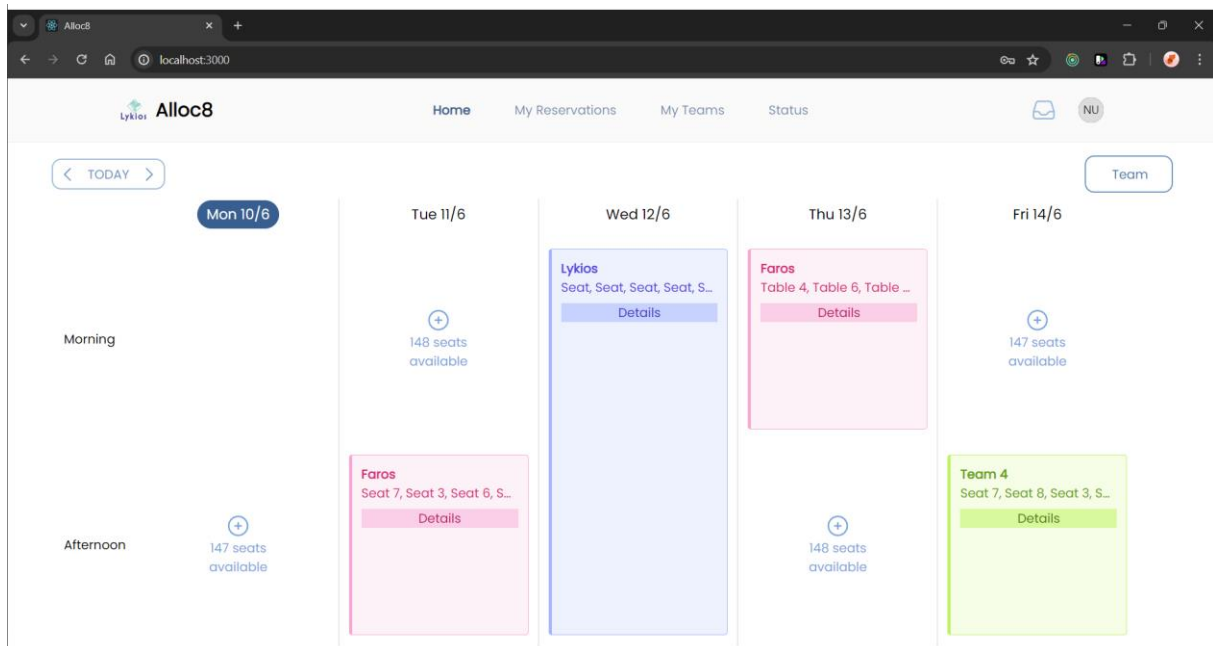
5.4. Kalender weergave

Op de startpagina wordt er een kalender getoond waarop de gebruiker voor een bepaalde dag kan kiezen om een reservatie te maken in de voor- of namiddag, alle reservaties van deze gebruiker worden ook op deze kalender weergegeven. Dit is te zien op Figuur 7. Daarnaast kan een gebruiker linksboven op het scherm ook navigeren doorheen de weken om zijn reservaties te raadplegen.



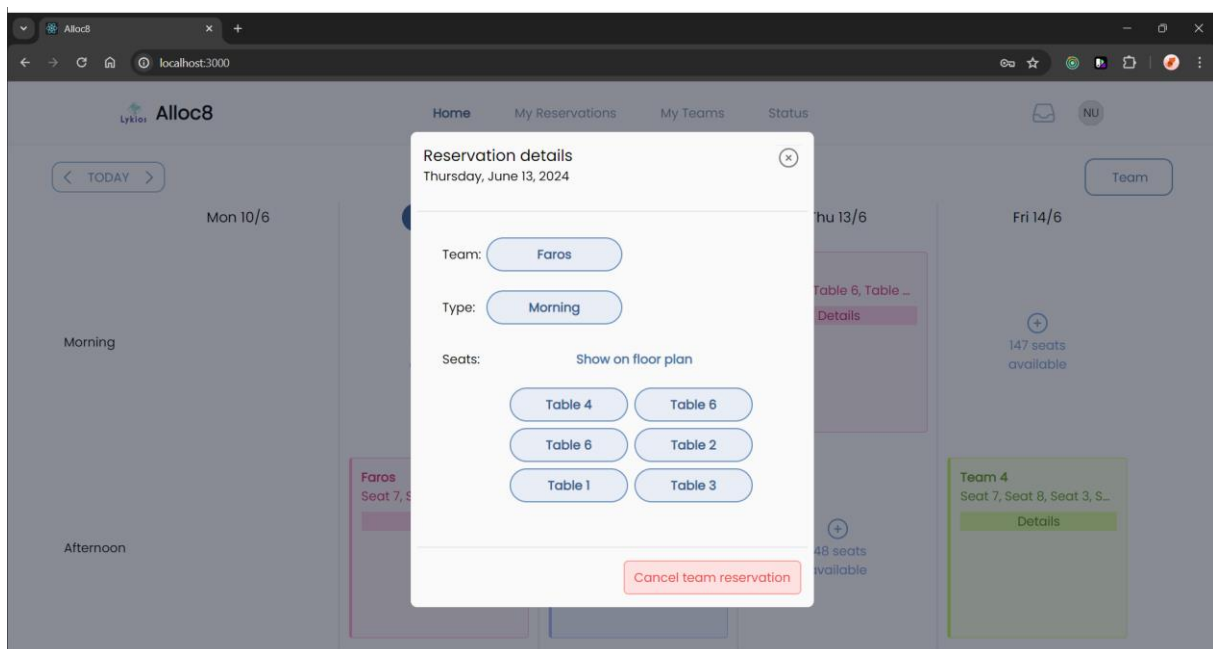
Figuur 7: Kalender weergave voor individuele reservaties

Later hebben we ook de team reservaties hier op weergegeven. Rechts bovenaan op Figuur 8 staat er een switch waarmee te gebruiker kan kiezen of zijn individuele of team reservaties worden getoond.



Figuur 8: Kalender weergave voor team reservaties

Tot slot kan een gebruiker ook details bekijken van een reservatie, dit opent een modal (te zien op Figuur 9) waarin alle info betreffend de geselecteerde reservatie.



Figuur 9: Details modal van een reservatie

5.5. Frontend herwerken

Zoals bij het onderdeel Class Diagram eerder beschreven stond, wordt het moment van een reservatie (voormiddag, namiddag of hele dag) bijgehouden als twee boolean (true-or-false) waarden, namelijk *amReservation* en *pmReservation*:

- Als beide waarden true zijn, is het een reservatie voor een hele dag
- Als enkel *amReservation* true is, is het een reservatie voor de voormiddag
- Als enkel *pmReservation* true is, is het een reservatie voor de namiddag

In het begin hadden we dit soort checks doorheen de code op veel verschillende plaatsen gebruikt, deze code is uiteraard niet heel goed en herbruikbaar. Daarom heb ik dit soort checks herwerkt tot een aantal standaard functies die we vanuit verschillende plaatsen kunnen oproepen. Zo kregen we een *isMorning*, *isAfternoon* en *isFullDay* functie die checkte wat voor reservatie het is.

Hetzelfde heb ik gedaan voor nog een aantal andere checks zoals bijvoorbeeld het controleren of een datum in het verleden ligt of dat dit de datum van vandaag is.

5.6. Role-based access control

Role-based access control staat in voor het beheren van toegang tot bepaalde pagina's of informatie, op deze manier wordt ervoor gezorgd dat enkel gebruikers met de building manager rol (zie Class Diagram) pagina's te zien krijgen waar zij hun kantoorgebouw kunnen beheren.

Net zoals bij de Frontend validatie zijn hier een aantal standaard functies ingebouwd zoals bijvoorbeeld het checken voor een specifieke rol, zo ziet u op Figuur 10 hoe dit in de praktijk werkt. Het element dat voor specifieke rollen zichtbaar moet zijn wordt tussen een *AllowedAccess*-tag gezet. Deze *AllowedAccess* heeft een attribuut *allowedRoles* waarin de rollen worden gedefinieerd die dit element mogen zien. In dit geval kunnen enkel gewone gebruikers dit element zien.

```
<AllowedAccess allowedRoles={ [UserRoles.ROLE_USER] }>
  <div className="sm:flex sm:divide-x-2 text-left divide-primary-200 text-primary"...>
    <div...>
  </AllowedAccess>
```

Figuur 10: Element dat enkel zichtbaar is voor normale gebruikers

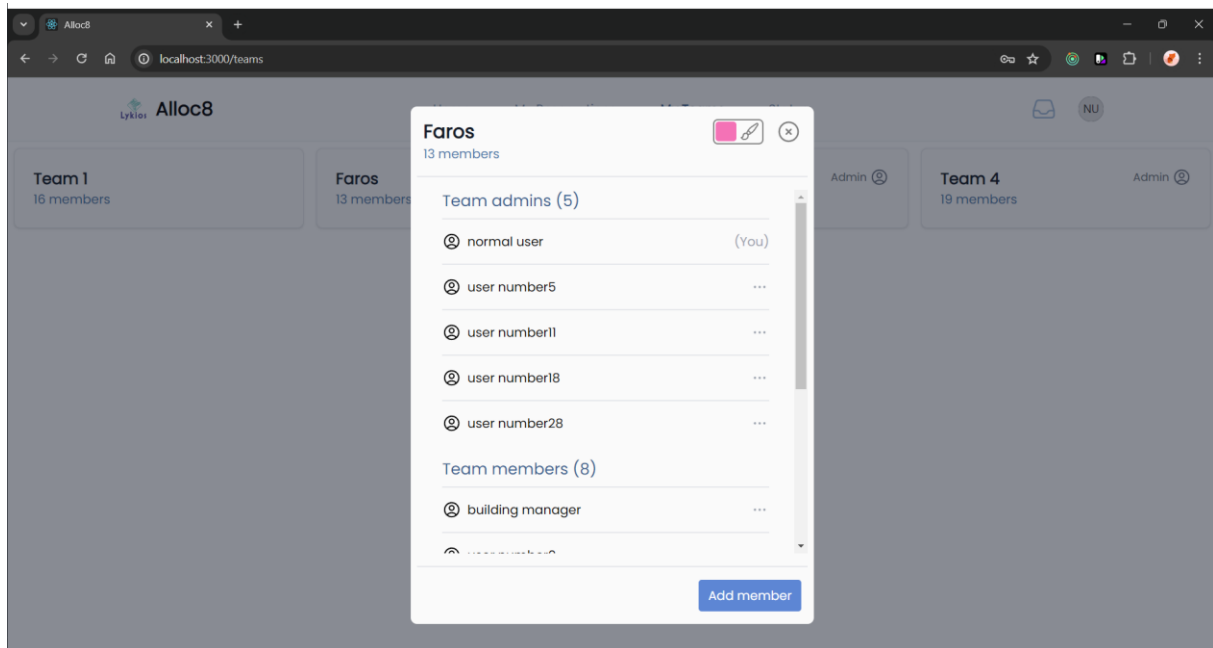
Daarnaast is er ook een makkelijke manier om specifieke checks toe te voegen, zo ziet u op Figuur 11 **Fout! Verwijzingsbron niet gevonden.** een element dat enkel zichtbaar zal zijn voor team admins. Dit gebeurt door een extra functie toe te voegen aan het *customCheck* attribuut.

```
<AllowedAccess customCheck={ () => team.isTeamAdmin }>
  <button onClick={ () : void => setIsAddMemberOpen( value: true) }
    className="py-2 px-3 bg-accent-500 text-white hover:bg-accent-600 duration-200 rou
    Add member
  </button>
</AllowedAccess>
```

Figuur 11: Element dat enkel zichtbaar is voor team admins

5.7. Teams beheren

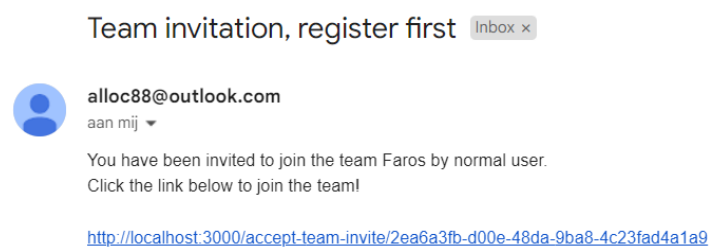
Als team administrator kan je ook je team beheren, dit houdt in dat je gebruikers kan uitnodigen, verwijderen, admin kan maken of hun admin rechten kan ontnemen. Eerst en vooral krijgt een gebruiker een lijst van alle teams, met een indicator die aantoont van welke teams hij/zij admin is. Als de gebruiker dan een team selecteert, verschijnt er een modal. Ik heb gewerkt aan deze modal waarin alle gebruikers van dit team worden weergegeven, deze gebruikers zijn opgesplitst in twee groepen: team admins en gewone team members. Op Figuur 12 ziet u deze modal.



Figuur 12: Modal om een team te bekijken en beheren

5.7.1. Gebruiker uitnodigen

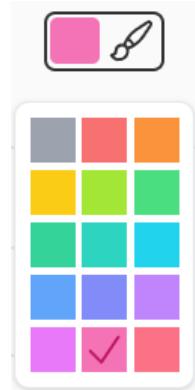
Als een team admin links onderaan de modal op “Add member” klikt opent er een scherm waarin een email adres wordt gevraagd. Na het invoeren van een email adres wordt er een email gestuurd met daarin een uitnodiging tot het team (dit is te zien op Figuur 13). Als de gebruiker deze unieke link opent wordt hij/zij doorverwezen naar de login of registreer pagina (afhankelijk van of er al dan niet een account met het ingevoerde email adres bestaat). Nadat de gebruiker zich heeft ingelogd of geregistreerd wordt de uitnodiging automatisch geaccepteerd en zal deze dus lid zijn van het team.



Figuur 13: Email met uitnodigingslink

5.7.2. Team kleur instellen

Rechts bovenaan de modal ziet u ook een knop met een penseel-icoon, hiermee kan een team admin de kleur van een team aanpassen door een andere kleur te selecteren zoals op Figuur 14. Reservaties voor dit team zullen dan in de geselecteerde kleur worden weergegeven op de kalender weergave, merk op dat op *Figuur 8: Kalender weergave voor team reservaties* de reservaties worden weergegeven in de kleur die correspondeert met het team.



Figuur 14: Team kleur instellen

5.8. Plattegrond visualisatie

De functionaliteit waar ik me het langst mee bezig heb gehouden is die van de plattegrond, ik heb hier dan ook zes van dertien weken aan gewerkt. Met deze plattegrond kunnen gebouwbeheerders hun kantoor visualiseren en normale gebruikers kunnen op deze visuele plattegrond bureaus aanduiden die zij willen reserveren.

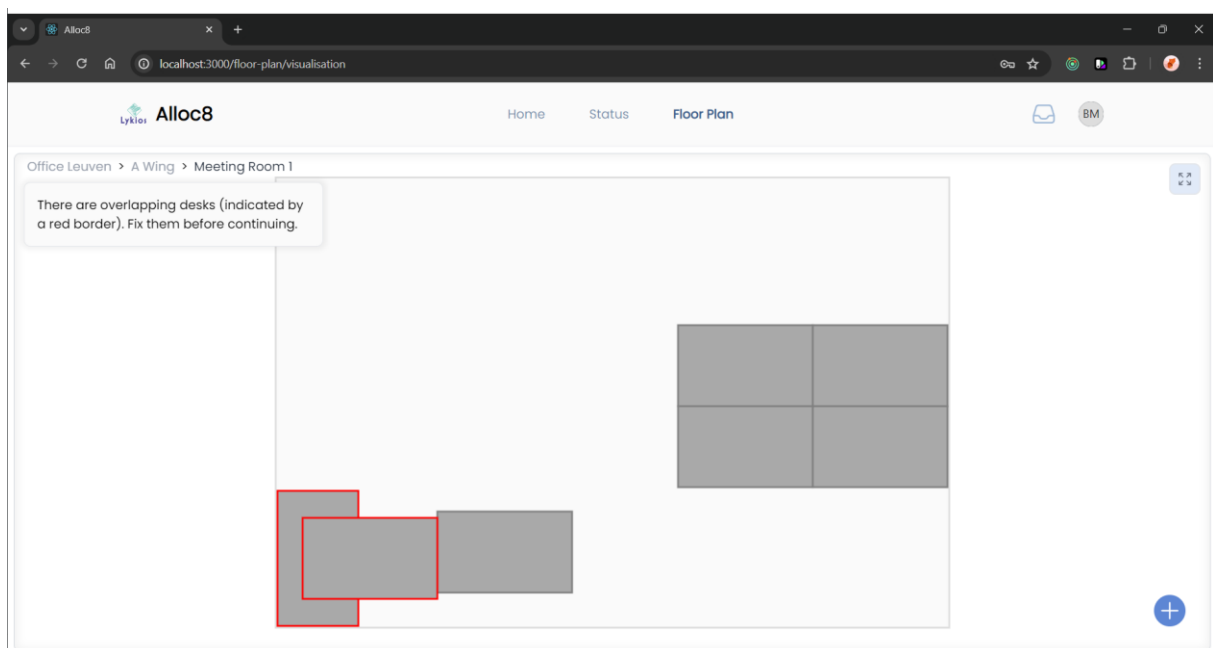
Deze visualisatie heb ik gemaakt met behulp van Konva.js, dit is een 2D canvas library. Deze library heeft ingebouwde functionaliteiten die het makkelijker maken om vormen op een 2D canvas weer te geven in webapplicaties.

Aangezien ik nog nooit met Konva had gewerkt, stelde de stagementor voor om eerst een proof of concept te maken zodat ik Konva kon leren kennen en tegelijk zou kunnen onderzoeken of Konva wel degelijk aan onze vereisten voldeed.

5.8.1. Collision detection en prevention

Bij de plattegrond visualisatie waren er veel struikelblokken maar de collisions waren veruit het grootste struikelblok. Collision detection is het systeem dat instaat voor het detecteren van overlappingen tussen bureaus terwijl collision prevention er juist voor gaat zorgen dat bureaus niet kunnen overlappen. Op deze manier voorkomen we dat gebouwbeheerders meerdere bureaus op mekaar kan stapelen op de plattegrond.

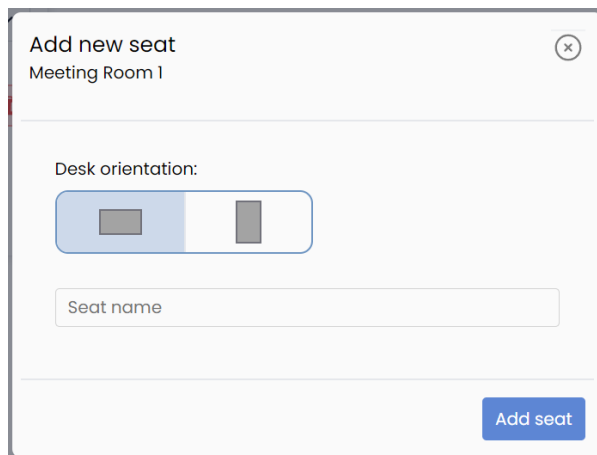
Ik heb me lang bezig gehouden met het voorkomen dat bureaus zouden overlappen, als de bureaus één-op-één interageren werkt dit ook foutloos. Als een bureau tegen een ander bureau wordt gesleept, blijft deze als het ware kleven tegen het andere bureau in plaats van te overlappen. Het probleem begint als er meerdere bureaus met elkaar interageren. Het bureau dat gesleept wordt kan altijd maar aan één ander bureau tegelijk blijven kleven. Omdat ik hier geen oplossing voor vond heb ik ervoor gezorgd dat de gebouwbeheerder zijn aanpassingen niet kan opslaan zolang er overlappende bureaus zijn, zulke situatie ziet u op Figuur 15.



Figuur 15: Plattegrond met overlappende bureaus

5.8.2. Beheren van bureaus en segmenten

Een gebouwbeheerder kan segmenten en bureaus aanmaken, bewerken en verwijderen. Op Figuur 15 ziet u rechtsonder een blauwe knop, als de gebruiker hierop klikt kan deze kiezen tussen “Add seat” of “Add segment”. Beide opties openen een modal die te zien zijn op Figuur 16 en Figuur 17 respectievelijk.

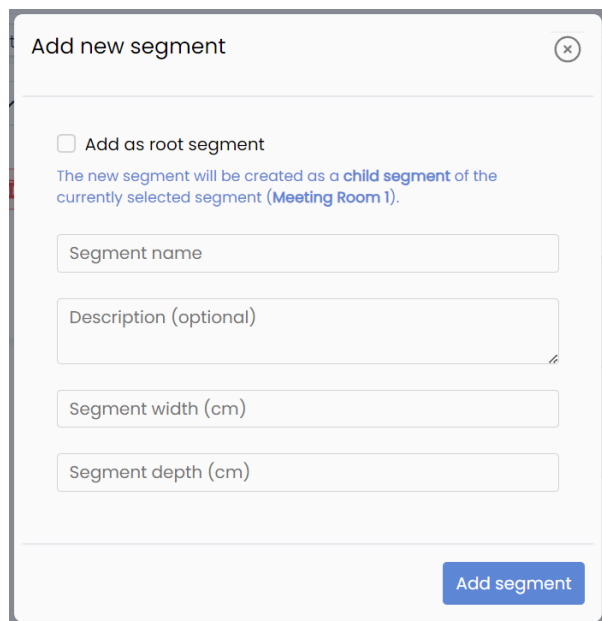


The modal is titled "Add new seat" with a subtitle "Meeting Room 1". It features a "Desk orientation:" section with two radio buttons and corresponding icons: "Horizontal" (selected) and "Vertical". Below this is a "Seat name" input field. At the bottom right is a blue "Add seat" button.

Figuur 16: Aanmaken van een nieuw bureau

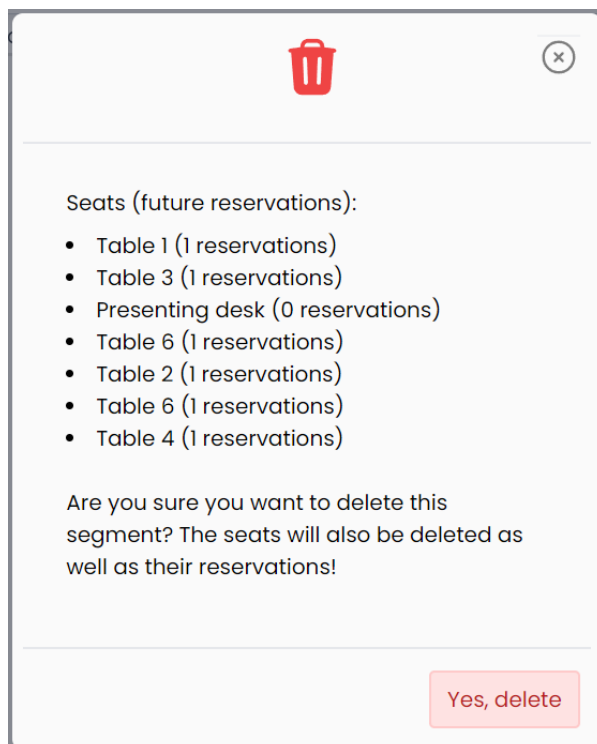
Als de gebruiker kiest om een nieuw segment aan te maken wordt de volgende informatie gevraagd: de naam, (beschrijving,) breedte en diepte van het nieuwe segment. Helemaal bovenaan staat er ook een check box die bepaalt of het segment een sub-segment is van het geselecteerde segment of een *root segment* (dit is een segment dat geen sub-segment is van een ander segment, zie Figuur 18). Na het aanmaken van een segment wordt dit automatisch geselecteerd en weergegeven.

Als de gebruiker kiest om een nieuwe seat (bureau) toe te voegen, krijgt deze een scherm te zien waarin de oriëntatie en de naam van het bureau gekozen moet worden. Nadat het nieuwe bureau is aangemaakt verschijnt deze in het midden van het geselecteerde segment.



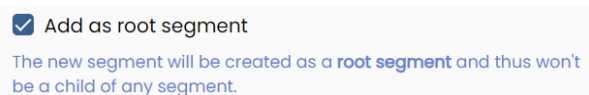
The modal is titled "Add new segment". It starts with a checkbox "Add as root segment". Below it, a note states: "The new segment will be created as a child segment of the currently selected segment (Meeting Room 1)." There are four input fields: "Segment name", "Description (optional)", "Segment width (cm)", and "Segment depth (cm)". A blue "Add segment" button is at the bottom right.

Figuur 17: Aanmaken van een nieuw segment



The modal has a red trash icon at the top left. It lists "Seats (future reservations):" with a bulleted list: Table 1 (1 reservations), Table 3 (1 reservations), Presenting desk (0 reservations), Table 6 (1 reservations), Table 2 (1 reservations), Table 6 (1 reservations), and Table 4 (1 reservations). Below the list is a warning: "Are you sure you want to delete this segment? The seats will also be deleted as well as their reservations!". A red "Yes, delete" button is at the bottom right.

Figuur 19: Verwijderen van een segment met reservaties



The modal shows the checkbox "Add as root segment" checked. A note states: "The new segment will be created as a root segment and thus won't be a child of any segment."

Figuur 18: Aanmaken van een root segment

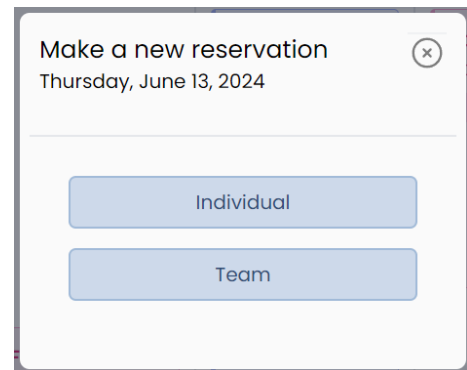
Daarnaast kan de gebruiker ook een segment of bureau selecteren en kiezen om te bewerken of te verwijderen. De bewerk functie opent een soortgelijke modal waarin de meeste waarden kunnen worden aangepast.

Voor het verwijderen van een segment of bureau is er wel iets extra, tijdens het verwijderen wordt de gebruiker gewaarschuwd als er nog reservaties gepland staan voor een bureau binnen dit segment, deze waarschuwing ziet u op Figuur 19.

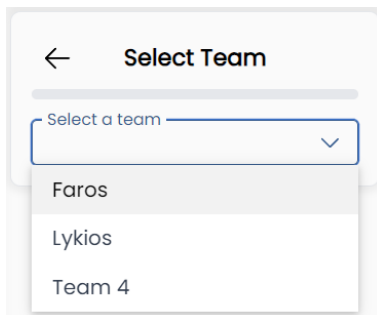
5.8.3. Reservatie maken voor bureaus

Een normale gebruiker kan ook reservaties maken voor een bureau, ook hier wordt gebruik gemaakt van de plattegrond. Het reserveren wordt beschreven in een aantal stappen die hieronder worden beschreven.

De gebruiker kiest eerst een datum waarop deze een reservatie wilt maken, de gebruiker navigeert op de kalender weergave naar deze datum en kiest dan tussen voor- of namiddag. Dit opent een modal. Als de gebruiker administrator is van een team krijgt deze eerst de keuze tussen het aanmaken van een individuele of team reservatie (dit ziet u op Figuur 20), als de gebruiker voor geen enkel team administrator is wordt deze stap niet getoond.



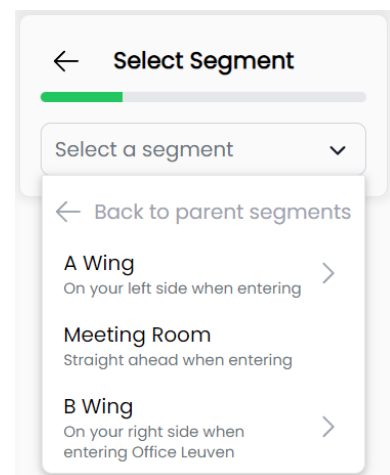
Figuur 20: Keuze tussen individuele of team reservatie



Figuur 21: Team opties

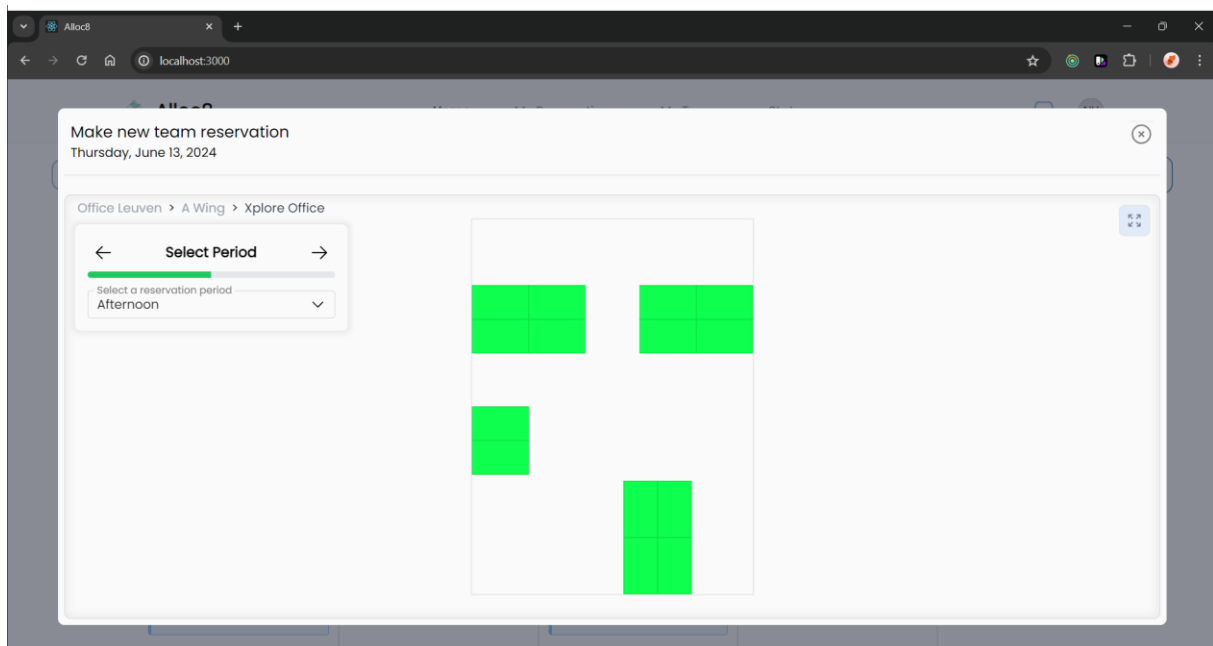
De workflow voor het aanmaken van individuele of team reservaties is licht verschillend. Voor een team reservatie wordt er eerst gevraagd voor welk team de gebruiker een reservatie wilt maken. Enkel teams waarvoor de gebruiker administrator is worden getoond.

In de volgende stap wordt een dropdown menu weergegeven waarin de gebruiker kan navigeren tussen de (child en parent) segmenten. Hier moet de gebruiker dan uiteindelijk een segment selecteren waarvoor hij/zij een reservatie wil maken.



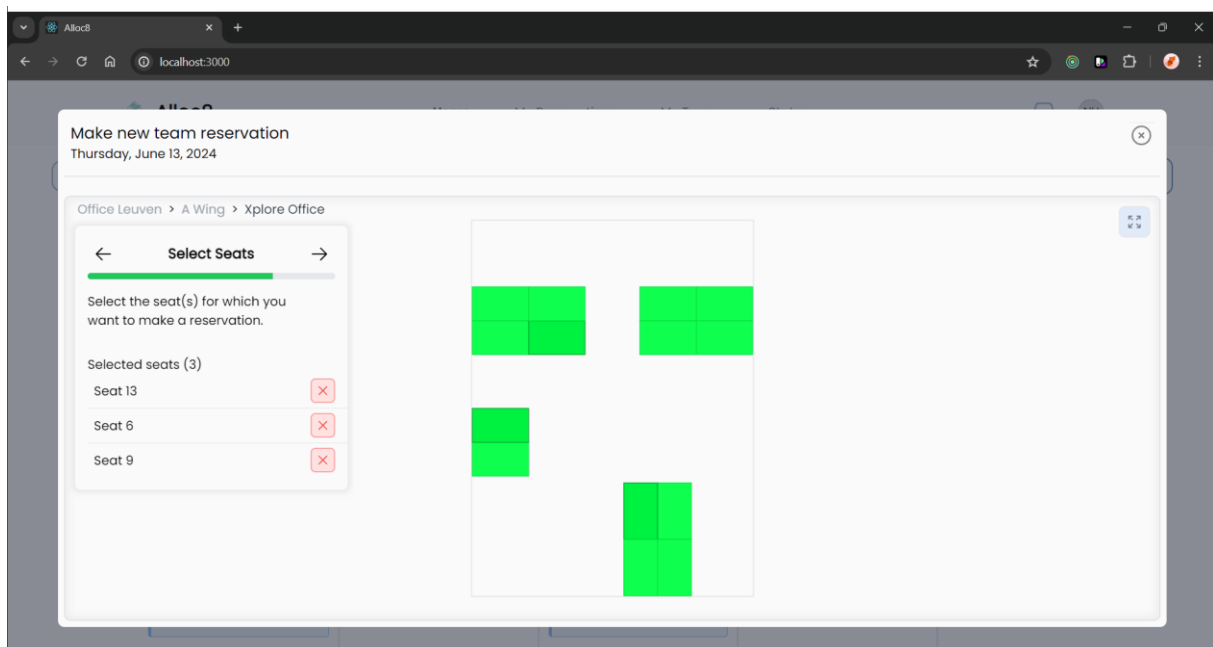
Figuur 22: Segment dropdown met sub-segmenten

Als de gebruiker een segment heeft geselecteerd wordt de plattegrond van dit segment getoond, dit is te zien op Figuur 23. Nu wordt de gebruiker gevraagd om een periode te selecteren, hier kan deze kiezen voor morning, afternoon of full day (sommige opties zijn niet mogelijk als de gebruiker voor deze periode al een reservatie heeft).

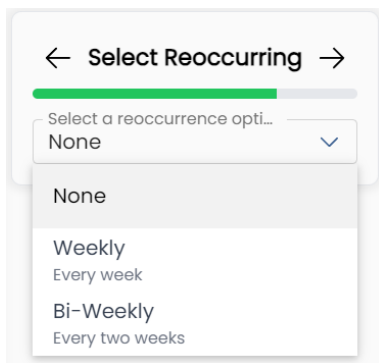


Figuur 23: Segment met optie om reservatie periode te selecteren

Als volgende stap kan de gebruiker bureau(s) selecteren, voor een team reservatie kunnen er meerdere bureaus geselecteerd worden terwijl een individuele reservatie voor maximaal 1 bureau mag zijn. Linksboven (op Figuur 24) krijgt de gebruiker ook alle geselecteerde bureaus te zien.



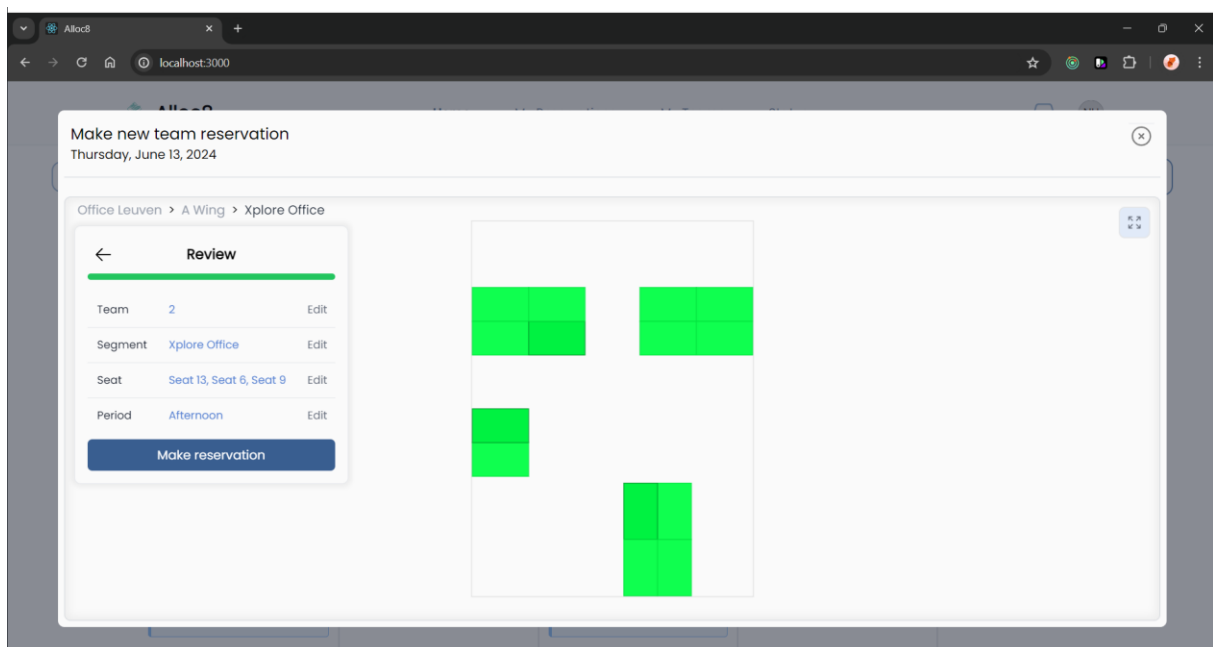
Figuur 24: Bureaus aanduiden om te selecteren



De volgende stap geldt enkel voor individuele reservaties, de gebruiker kan hier kiezen om een herhalende reservatie te maken. Dit betekent dat een reservatie bijvoorbeeld elke week op donderdag namiddag zal plaatsvinden.

Figuur 25: Opties voor herhalende reservaties

Tot slot krijgt de gebruiker als laatste stap nog een overzicht van alle geselecteerde waardes. Zo kan deze nog een laatste bevestiging geven voordat de reservatie wordt aangemaakt.



Figuur 26: Overzichtsscherm als laatste stap

5.9. Soft delete

Het laatste waar ik aan heb gewerkt is het implementeren van soft-deletes in de backend, dit houdt in dat data niet effectief wordt verwijderd uit de database maar eerder op "inactief" wordt gezet. In de originele opdracht stond dat bureaus ook verhuurd zouden kunnen worden, als dit zou worden geïmplementeerd willen we niet dat gegevens van een reservatie zomaar verwijderd worden aangezien hier mogelijks facturen voor moeten worden opgemaakt. Dit was een vrij makkelijke functionaliteit die ook heel vlot kan worden toegepast op nieuwe klassen.

Slot

Mijn stage bij Lykios heeft me een waardevolle ervaring geboden in de wereld van IT-consultancy en softwareontwikkeling. Door te werken aan het project Alloc8 heb ik niet alleen mijn technische vaardigheden kunnen aanscherpen, maar ook inzicht gekregen in de dynamiek van een professionele werkomgeving.

Eén van de belangrijkste leerpunten van deze stage was het belang van samenwerking en communicatie. Het agile werken met frequente feedbackmomenten zorgde ervoor dat we flexibel konden inspelen op veranderingen en snel aanpassingen konden maken afhankelijk van de verwachtingen.

Mijn tijd bij Lykios heeft me niet alleen technische kennis bijgebracht, maar ook waardevolle soft skills, zoals teamwerk, probleemoplossend vermogen en klantgericht denken. Deze vaardigheden zijn essentieel voor een succesvolle carrière in de IT en zullen me zeker van pas komen in mijn toekomstige professionele leven.

In het kort, mijn stage bij Lykios was een uiterst leerzame en verrijkende ervaring. Ik ben dankbaar voor de kansen die ik heb gekregen en voor de begeleiding en ondersteuning van mijn collega's. Deze stage heeft mijn passie voor technologie en softwareontwikkeling verder aangewakkerd en me voorbereid op de volgende stappen in mijn carrière.

