cegeka
shaping
digital together

Angular, .NET en AI-driven development in de praktijk

Yenthe Jannis – Toegepaste Informatica

Opgemaakt door **Yenthe Jannis**

Academisch jaar 2025-2026

Cegeka nv - Thomas More

1 Contact

Deze stage was (in)direct onder begeleiding van volgende perso(o)n(en):

Geoffrey, Vanhearen
Hoofd-Stagementor
E: Geoffrey.Vanhaeren@cegeka.com

Bart Eijckmans
Stagebegeleider/uitgangspunt klant
E: bart.eijckmans@cegeka.com

Frazs Iacob
Stagementor in samenwerking met Geoffrey
E: Franz.Iacob@cegeka.com

cegeka

Inhoudstafel

2 Inleiding

Dit realisatiedocument beschrijft het volledige verloop van de stage bij Cegeka. Cegeka is een Belgisch IT-bedrijf dat oplossingen ontwikkelt voor klanten in uiteenlopende sectoren, gaande van landbouw en zorg tot financiële dienstverlening en industrie. De stage vond plaats binnen een team dat zich voornamelijk richt op de ontwikkeling van moderne webapplicaties, zowel voor klantprojecten als voor interne tools.

De hoofdopdracht betrof de herontwikkeling van FarmFit, een applicatie van DGZ, een organisatie die digitale toepassingen ontwikkelt voor veehouders en specialisten binnen de landbouwsector. De bestaande FarmFit-applicatie, ongeveer acht jaar geleden gebouwd in React met een .NET-backend, was inmiddels verouderd: de codebase werd moeilijk onderhoudbaar en de prestaties resulteerden in een trage gebruikerservaring. Hieruit volgde de centrale onderzoeksvraag van dit document:

Op welke manier kan een verouderde, React-gebaseerde webapplicatie herontwikkeld worden tot een moderne, onderhoudbare en performante Angular-applicatie met offlinefunctionaliteit, en welke inzichten levert dit op voor het bredere technologische en methodologische beleid binnen Cegeka?

Naast deze hoofdvraag stelt het document drie deelvragen centraal, die voortvloeien uit de bijkomende projecten die tijdens de stage werden opgenomen na de vroegtijdige oplevering van FarmFit in week acht van de dertien:

- *Hoe kan een bestaande, organisch gegroeide codebase (SafetySquad) gerefactord en geredesigned worden zodat ze opnieuw onderhoudbaar, schaalbaar en marktklaar wordt?*
- *In welke mate is het haalbaar om een volledige applicatie (RaamContracten) op te leveren via AI-driven development, en welke randvoorwaarden zijn nodig om de kwaliteit hiervan te garanderen?*
- *Hoe kunnen gerichte UI- en UX-aanpassingen (OWASP-tool) bijdragen aan een consistentere en gebruiksvriendelijkere ervaring binnen een bestaande enterprise-applicatie?*

De technologische rode draad doorheen de vier projecten is Angular aan de frontendzijde, waar nodig aangevuld met .NET aan de backendzijde. Bijkomend werd ervaring opgedaan met PrimeNG, Tailwind CSS, signal-gebaseerde state management, Azure Entra ID, Progressive Web Apps met offlinefunctionaliteit en AI-driven development. Deze technologieën weerspiegelen de moderne werkwijzen die binnen Cegeka in de praktijk worden toegepast en vormen samen het kader waarbinnen de bovenstaande onderzoeksvragen worden beantwoord.

Het document is opgebouwd uit één hoofdstuk per project. Per project komen achtereenvolgens de analyse, de opzet, de bewuste technische en methodologische keuzes, de uitdagingen en de bijhorende oplossingen aan bod. Het document sluit af met een overkoepelende conclusie waarin de stage als geheel wordt geëvalueerd en waarin wordt teruggekoppeld naar de hierboven geformuleerde onderzoeksvragen.

3 FarmFit

Figuur 1. *Logo van FarmFit, de applicatie van DGZ voor veehouders en specialisten binnen de landbouwsector.*

3.1 Inleiding

FarmFit vormde de hoofdopdracht van de stage bij Cegeka en werd uitgevoerd in samenwerking met DGZ. De applicatie werd ongeveer acht jaar geleden ontwikkeld door Cegeka en DGZ, op basis van React (een JavaScript-framework voor het bouwen van webapplicaties) met een .NET-backend (de serverzijde die instaat voor de gegevensverwerking).

De bestaande applicatie was inmiddels verouderd. De code werd moeilijk onderhoudbaar en de prestaties resulteerden in een trage gebruikerservaring. Vanuit deze problematiek werd de opdracht uitgewerkt om, samen met een medestagiair van UCLL, elk afzonderlijk een nieuwe versie van de applicatie te ontwikkelen. Deze nieuwe applicaties dienden functioneel volledig overeen te komen met de bestaande versie, terwijl er op vlak van gebruiksvriendelijkheid en visueel ontwerp verbeteringen mochten worden doorgevoerd waar mogelijk.

Voor de uitwerking koos Cegeka bewust voor Angular (een modern framework voor het ontwikkelen van webapplicaties), terwijl de medestagiair met Vue (een ander populair framework) aan de slag ging. Naast het opleveren van een vernieuwde en performantere applicatie voor DGZ, had Cegeka als doel om inzichten te verwerven in verschillende technologieën. Hierbij werd onder andere gekeken naar ontwikkelsnelheid, onderhoudbaarheid, leercurve, prestaties en het gebruik van externe pakketten. Op basis van deze vergelijking wil Cegeka bepalen welk framework het best aansluit bij toekomstige projecten.

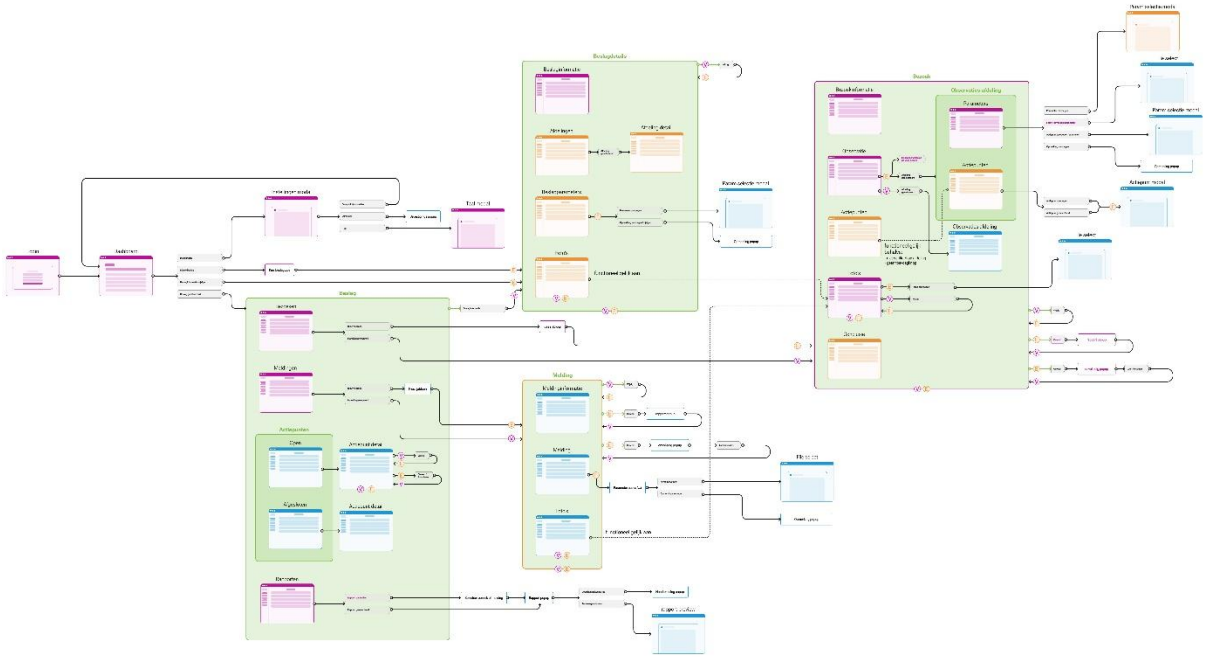
In dit hoofdstuk komen achtereenvolgens de gehanteerde analysemethode, de opzet van het project en de bewuste keuzes tijdens de ontwikkeling aan bod. Vervolgens worden de functionele aspecten van de applicatie besproken, inclusief de uitdagingen die zich voordeden en de bijhorende oplossingen. Tot slot wordt toegelicht hoe de applicatie werd opgebouwd als een PWA (Progressive Web App, een webapplicatie die zich gedraagt als een mobiele app), waardoor veehouders en specialisten deze ook offline kunnen gebruiken op afgelegen locaties.

3.2 Analyse

Voorafgaand aan de effectieve uitwerking van de applicatie werd een analysefase doorlopen. In deze fase werd onderzocht welke functionaliteiten nodig zijn, hoe de applicatie gestructureerd is en op welke manier de ontwikkeling het best aangepakt kon worden.

3.2.1 Visuele analyse: Figma

De visuele analyse werd in eerste instantie uitgewerkt door de medestagiair, die een flowdiagram opstelde waarin alle schermen en mogelijke acties binnen de applicatie worden weergegeven. Hoewel er aan dit diagram zelf niet actief werd bijgedragen, vormde het een zeer waardevol hulpmiddel tijdens de stage. Dankzij dit overzicht ontstond snel inzicht in de volledige applicatieflow en de samenhang tussen de verschillende schermen, waardoor er minder tijd nodig was om de globale structuur van de applicatie te analyseren. Daarnaast werd het diagram ook ingezet voor voortgangsrapportage, waarbij per scherm werd genoteerd hoeveel tijd eraan besteed werd.



Figuur 2. Figma-flowdiagram opgesteld door de medestagiair, met alle schermen en mogelijke acties binnen de FarmFit-applicatie.

3.2.2 Analyse van structuur en componenten: React app

Naast de functionele analyse werd ook aandacht besteed aan de structuur van de applicatie en de opbouw in componenten. Deze analyse verliep grotendeels gaandeweg tijdens de ontwikkeling. Door stap voor stap functionaliteiten uit te werken, ontstond gaandeweg meer inzicht in hoe componenten logisch konden worden opgesplitst en hoe deze onderling samenwerken. Op die manier bleef de applicatie overzichtelijk en onderhoudbaar.

3.2.3 Analyse van frameworks

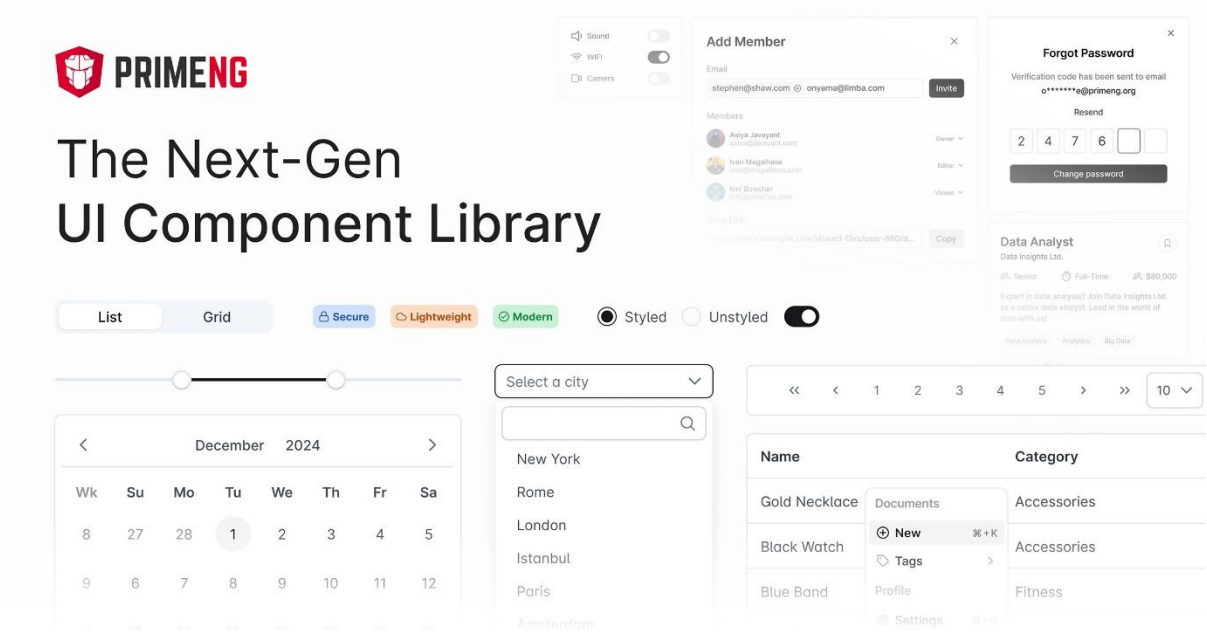
Zoals eerder vermeld koos Cegeka ervoor om de vergelijking te maken tussen Angular en Vue. De reden hiervoor is dat deze twee frameworks momenteel het meest gebruikt worden binnen Cegeka voor webapplicaties. Door de focus op deze frameworks te leggen, kon Cegeka inzichten verwerven in ontwikkelsnelheid, onderhoudbaarheid, prestaties en leercurve van technologieën die daadwerkelijk in interne projecten worden toegepast.

Andere populaire hedendaagse frameworks, zoals React, Next.js of Laravel, kwamen in aanmerking, maar werden voor deze opdracht niet weerhouden. React werd bijvoorbeeld gebruikt in de oorspronkelijke versie van FarmFit, en hoewel het een gangbare technologie is, wilde Cegeka onderzoeken hoe Angular en Vue zich verhouden op het vlak van ontwikkelsnelheid, structuur en onderhoudbaarheid. Door Angular en Vue als basis te nemen, sluit de analyse direct aan bij de interne praktijk van Cegeka, wat de relevantie van de resultaten vergroot.

Figuur 3. Logo van Angular, het frontend-framework dat door Cegeka werd gekozen voor de herontwikkeling van FarmFit (Angular Team, z.d.-a).

3.2.4 Analyse van Component Library

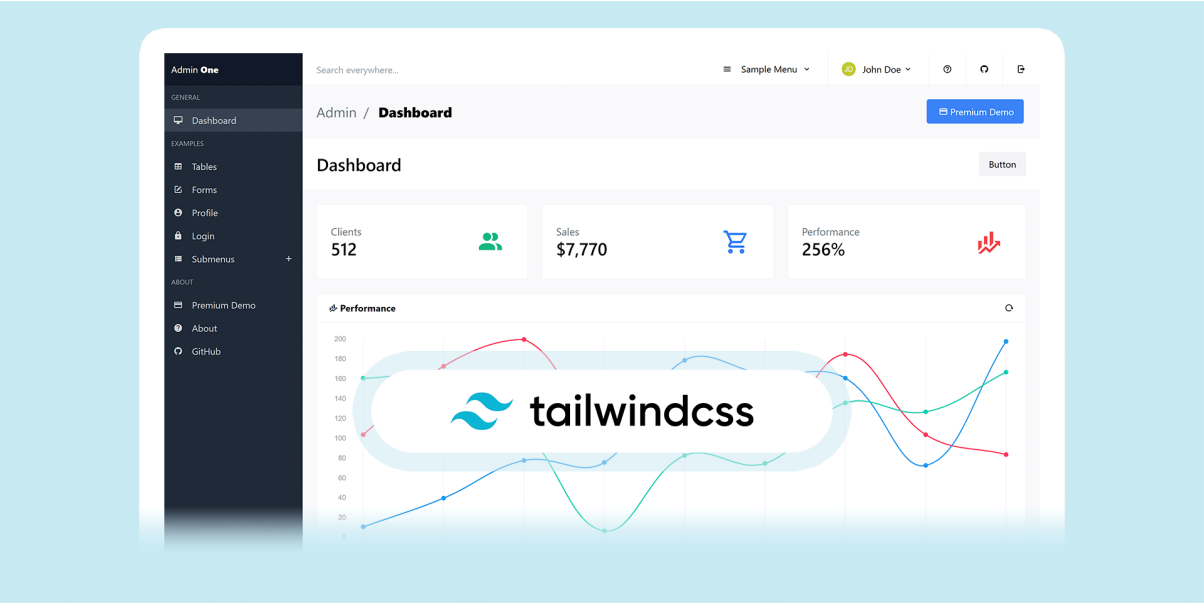
Voor deze applicatie koos Cegeka om gebruik te maken van PrimeNG als component library. Andere mogelijke opties waren bijvoorbeeld Angular Material of NG Bootstrap, die eveneens uitgebreide UI-componenten bieden voor Angular. PrimeNG had echter als voordeel dat er naast de Angular-variant ook een PrimeVue-variant bestaat, waardoor consistentie tussen de twee applicaties eenvoudiger te bewaken viel. Daarnaast biedt PrimeNG voldoende kant-en-klare componenten, wat de ontwikkeling versnelt en de onderhoudbaarheid bevordert. Deze keuze sloot bovendien goed aan bij de technologieën die tijdens de voorbereiding van de stage reeds waren verkend.



Figuur 4. PrimeNG, de component library die voor de FarmFit-applicatie werd ingezet (PrimeTek Informatics, z.d.-a).

3.2.5 Analyse van Styling Library:

Voor de styling library waren er verschillende mogelijkheden, zoals Bootstrap, Materialize of Bulma. De medestagiair koos voor TailwindCSS, waarna voor de Angular-applicatie dezelfde keuze werd gemaakt. Deze beslissing was enerzijds gebaseerd op voorkeur, maar vooral omdat TailwindCSS een utility-first aanpak hanteert waardoor stijlen snel en flexibel toepasbaar zijn. Bovendien werd het hierdoor eenvoudiger om pagina's en componenten tussen de twee applicaties over te nemen en consistent te houden, wat zowel de samenwerking als de onderhoudbaarheid ten goede kwam.



Figuur 5. Voorbeeldweergave van Tailwind CSS, het utility-first CSS-framework dat voor de styling van de applicatie werd gebruikt (Tailwind Labs, z.d.).

3.3 Opzet van FarmFit project

Figuur 6. Overzicht van de gekozen frontend-stack voor het FarmFit-project: Angular, PrimeNG en Tailwind CSS.

3.3.1 Framework en naming conventions:

Voor de opzet van het project werd gebruikgemaakt van Angular 21, de meest recente stabiele versie die door het framework zelf wordt aanbevolen (Angular Team, z.d.-a). Deze keuze werd ook goedgekeurd door Cegeka. Angular 21 introduceert echter nieuwe naming conventions waar Cegeka niet volledig achter stond. Daarom werd ervoor gekozen om de oude benamingen te behouden.

Concreet betekent dit bijvoorbeeld een verschil in naamgeving van componentbestanden:

- Angular 21 noemt een component zoals Login Component gewoon login.ts
- Voorheen werd dit login.component.ts genoemd

Deze keuze maakt het zoeken naar bestanden eenvoudiger en overzichtelijker. Bij het zoeken op "Login Component" zou het bestand met de nieuwe Angular 21-benaming immers moeilijker terug te vinden zijn.

Daarnaast hanteert Angular 21 een andere manier om componenten op te splitsen. In plaats van algemene folders zoals components of pages, wordt de structuur per module en feature georganiseerd. Cegeka gaf de voorkeur aan de oude indeling met algemene folders, waardoor ook met deze richtlijn rekening werd gehouden om consistentie en overzichtelijkheid te behouden.

3.3.2 Styling en component libraries:

Voor de styling werd gekozen voor Tailwind CSS, een utility-first CSS-framework dat het mogelijk maakt om stijlen rechtstreeks in HTML-templates toe te passen zonder afzonderlijke CSS-bestanden (Tailwind Labs, z.d.). Voor de UI-componenten werd PrimeNG ingezet (PrimeTek Informatics, z.d.-a), met in het Vue-project de overeenkomstige PrimeVue-variant (PrimeTek Informatics, z.d.-b).

Deze componentenlibrary sluit goed aan bij de oude MaterialUI-library die Cegeka eerder gebruikte voor FarmFit. Door deze libraries te combineren, wordt consistentie tussen de Angular- en Vue-versies van de applicatie behouden en kan er sneller en uniformer worden ontwikkeld.

3.4 Ondersteuning van meerdere talen

De vertalingen werden overgenomen uit de bestaande applicatie om consistentie te behouden. De keuze voor ngx-translate was snel gemaakt, aangezien deze library door Angular in de officiële documentatie wordt aanbevolen. Bovendien gaat het om een lichte library die goed aansluit bij de structuur en omvang van de applicatie.

Ondersteunde talen:

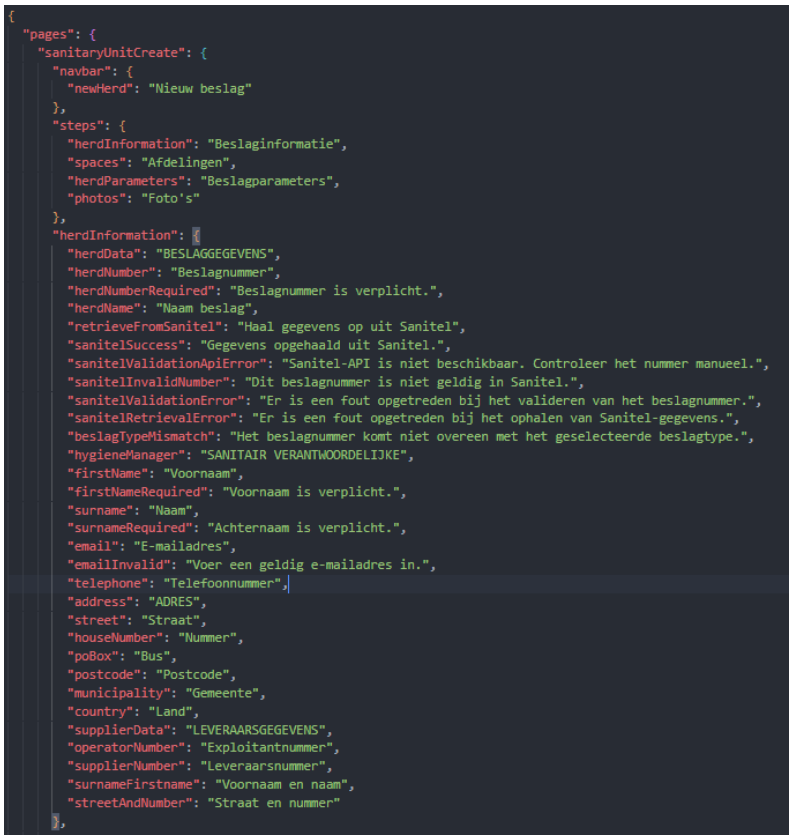
- Nederlands
- Engels
- Duits
- Frans

3.4.1 Opzet en gebruik van vertalingen:

1. **Importeren.** Het TranslateModule wordt geïmporteerd in de root-NgModule van de applicatie, zodat het overal beschikbaar is.
2. **Standaardtaal instellen.** De standaardtaal voor de applicatie wordt ingesteld via de TranslateService.
3. **Vertalingen laden.** Voor elke taal wordt een JSON-bestand aangemaakt in de map assets/i18n. Een voorbeeld is en.json voor Engelse vertalingen, waarin items als volgt worden gedefinieerd:
    - { "newHerd": "New herd" }
4. **Gebruik in de applicatie:** Gebruik in de applicatie. In de HTML-bestanden kunnen de vertalingen aangeroepen worden via de translate-pipe, bijvoorbeeld:
    - {{ "newHerd" | translate }}

De vertaling wordt dan automatisch weergegeven op basis van de gekozen taal.

Figuur 7. ngx-translate, de internationalisatielibrary die voor de meertaligheid van FarmFit werd gebruikt (Ngx-translate Contributors, z.d.).



```nl
{
  "pages": {
    "sanitaryUnitCreate": {
      "navbar": {
        "newHerd": "Nieuw beslag"
      },
      "steps": {
        "herdInformation": "Beslaginformatie",
        "spaces": "Afdelingen",
        "herdParameters": "Beslagparameters",
        "photos": "Foto's"
      },
      "herdInformation": {
        "herdData": "BESLAGGEGEVENS",
        "herdNumber": "Beslagnummer",
        "herdNumberRequired": "Beslagnummer is verplicht.",
        "herdName": "Naam beslag",
        "retrieveFromSanitel": "Haal gegevens op uit Sanitel",
        "sanitelSuccess": "Gegevens opgehaald uit Sanitel.",
        "sanitelValidationApiError": "Sanitel-API is niet beschikbaar. Controleer het nummer manueel.",
        "sanitelInvalidNumber": "Dit beslagnummer is niet geldig in Sanitel.",
        "sanitelValidationError": "Er is een fout opgetreden bij het valideren van het beslagnummer.",
        "sanitelRetrievalError": "Er is een fout opgetreden bij het ophalen van Sanitel-gegevens.",
        "beslagTypeMismatch": "Het beslagnummer komt niet overeen met het geselecteerde beslagtype.",
        "hygieneManager": "SANITAIR VERANTWOORDELIJKE",
        "firstName": "Voornaam",
        "firstNameRequired": "Voornaam is verplicht.",
        "surname": "Naam",
        "surnameRequired": "Achternaam is verplicht.",
        "email": "E-mailadres",
        "emailInvalid": "Voer een geldig e-mailadres in.",
        "telephone": "Telefoonnummer",
        "address": "ADRES",
        "street": "Straat",
        "houseNumber": "Nummer",
        "poBox": "Bus",
        "postcode": "Postcode",
        "municipality": "Gemeente",
        "country": "Land",
        "supplierData": "LEVERAARSGEGEVENS",
        "operatorNumber": "Exploitantnummer",
        "supplierNumber": "Leveraarsnummer",
        "surnameFirstname": "Voornaam en naam",
        "streetAndNumber": "Straat en nummer"
      },
```

```en
{
  "pages": {
    "sanitaryUnitCreate": {
      "navbar": {
        "newHerd": "New herd"
      },
      "steps": {
        "herdInformation": "Herd information",
        "spaces": "Spaces",
        "herdParameters": "Herd parameters",
        "photos": "Photos"
      },
      "herdInformation": {
        "herdData": "HERD DATA",
        "herdNumber": "Herd number",
        "herdNumberRequired": "Herd number is required.",
        "herdName": "Herd name",
        "retrieveFromSanitel": "Retrieve data from Sanitel",
        "sanitelSuccess": "Data retrieved from Sanitel.",
        "sanitelValidationApiError": "The Sanitel API is unavailable. Please verify the number manually.",
        "sanitelInvalidNumber": "This herd number is not valid in Sanitel.",
        "sanitelValidationError": "An error occurred while validating the herd number.",
        "sanitelRetrievalError": "An error occurred while retrieving data from Sanitel.",
        "beslagTypeMismatch": "The herd number does not match the selected herd type.",
        "hygieneManager": "HYGIENE MANAGER",
        "firstName": "First name",
        "firstNameRequired": "First name is required.",
        "surname": "Surname",
        "surnameRequired": "Surname is required.",
        "email": "E-mail address",
        "emailInvalid": "Please enter a valid email address.",
        "telephone": "Telephone number",
        "address": "ADDRESS",
        "street": "Street",
        "houseNumber": "Number",
        "poBox": "PO Box",
        "postcode": "Postcode",
        "municipality": "Municipality",
        "country": "Country",
        "supplierData": "SUPPLIER DATA",
        "operatorNumber": "Operator number",
        "supplierNumber": "Supplier number",
        "surnameFirstname": "First name and surname",
        "streetAndNumber": "Street and number"
      },
```

Figuur 8. Voorbeeld van een vertaalbestand uit de map assets/i18n, met de Nederlandse vertaling (boven) en de Engelse vertaling (onder) voor de aanmaakflow van een beslag.

3.5 Benaming

Dit korte hoofdstuk licht de belangrijkste termen toe die binnen de FarmFit-applicatie worden gebruikt. Een eenduidig begrip van deze termen is noodzakelijk om de verdere hoofdstukken van dit document correct te kunnen interpreteren. Een aantal termen worden in het Engels gehanteerd, omdat de code en de applicatie zelf in het Engels zijn geschreven. Hieronder volgen de vertalingen en bijhorende uitleg.

Belangrijke benamingen

- **Sanitary Units (Beslagen):** de objecten die worden gecontroleerd of onderzocht. Er bestaan verschillende types, zoals Rundvee, Kleine Herkauwers, Varkens en Pluimvee.
- **Visits (Bezoeken):** bezoeken die een arts kan uitvoeren voor een beslag.
- **Messages (Meldingen):** meldingen die een arts kan registreren voor een beslag.
- **Action Points (Actiepunten):** actiepunten die tijdens een bezoek door een arts worden aangemaakt en die verdere acties aangeven.
- **Parameters:** vragen die tijdens een observatie moeten worden beantwoord.
- **Observations (Observaties):** gegevens die tijdens een bezoek worden ingevuld. Het betreft een lijst van vragen, waaronder verplichte vragen, die door de arts beantwoord moeten worden.
- **Spaces (Afdelingen):** categorieën waarin parameters georganiseerd zijn, bijvoorbeeld de verschillende afdelingen binnen een bedrijf of het type activiteit dat wordt waargenomen.

Hoe dit in de praktijk werkt:

Om een beter beeld te geven van hoe deze termen binnen de applicatie samenhangen, kan het proces van een arts als volgt worden beschreven:

1. De arts kiest een type beslag (Rundvee, Pluimvee, Kleine Herkauwers of Varkens).
2. Het beslag wordt aangemaakt en ingevuld.
3. Na het aanmaken kan de arts een bezoek plannen en invullen.
4. Tijdens het invullen van een bezoek kunnen actiepunten worden toegevoegd en observaties worden gemaakt.
5. Observaties worden aangemaakt door eerst de relevante afdelingen (Spaces) te selecteren en vervolgens de parameters binnen deze afdelingen in te vullen.

De betekenis van deze termen en processen wordt verder verduidelijkt aan de hand van de concrete voorbeelden en functionaliteiten die later in dit document aan bod komen.

3.6 Overzicht functionaliteiten (high-level)

Dit hoofdstuk biedt een overzicht van de belangrijkste functionaliteiten van de vernieuwde FarmFit-applicatie. Per onderdeel wordt aangegeven wat gebruikers kunnen doen binnen de applicatie, welke onderdelen wel of niet door de applicatie worden afgehandeld, en hoe bepaalde keuzes — zoals offline gebruik en PWA-functionaliteit — bijdragen aan een efficiënte en gebruiksvriendelijke ervaring. Het overzicht schetst de mogelijkheden en structuur van de applicatie voorafgaand aan de gedetailleerde voorbeelden en implementatie in de volgende hoofdstukken.

Authenticatie

- Inloggen
- Registratie (via DGZ, out-of-scope)

Gebruiker

- Instellingen: praktijk kiezen, FarmFit installeren, taal beheren
- Cookies beheren (out-of-scope)

Beslagen

- Aanmaken: beslagnummer invoeren, afdelingen en parameters kiezen, foto's uploaden
- Bewerken en bekijken
- Binnen een beslag: bezoeken, meldingen, actiepunten en rapporten aanmaken, bekijken en bewerken
- Verwijderen is niet voorzien

Bezoeken

- Sjabloon kiezen
- Reden en observaties invoeren
- Actiepunten beheren
- Foto's toevoegen
- Conclusie vastleggen

Meldingen

- Sjabloon kiezen
- Reden en parameters invoeren
- Foto's toevoegen

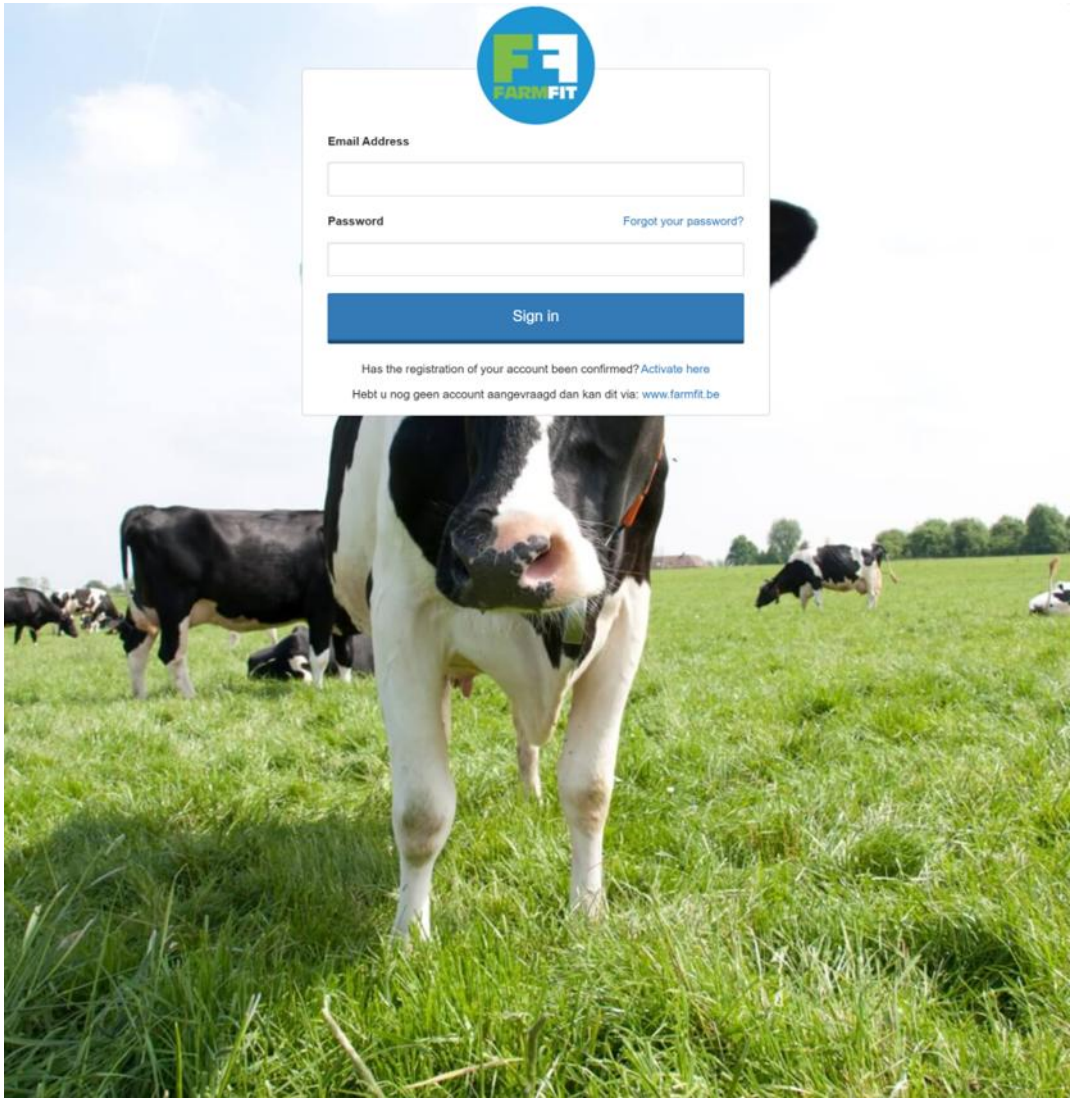
Rapporten

- Genereren
- Ondertekenen

Offline & PWA

- Automatische backups tijdens invoer
- Offline opslaan en later synchroniseren

3.7 Authenticatie



Figuur 9. Inlogscherm van de bestaande FarmFit-applicatie, dat via Microsoft Entra ID wordt afgehandeld.

Figuur 10. Microsoft Entra ID en de MSAL Angular-package, gebruikt voor de authenticatie binnen FarmFit (Microsoft, z.d.-b; Microsoft, z.d.-c).

Dit hoofdstuk licht toe hoe de authenticatie in de vernieuwde FarmFit-applicatie werkt, zonder gevoelige informatie te delen.

Voor de authenticatie werd gebruikgemaakt van Microsoft Entra ID, het identiteitsplatform van Microsoft (Microsoft, z.d.-b). Dit systeem wordt door DGZ al gebruikt voor meerdere applicaties, inclusief de oude versie van FarmFit, en regelt wie toegang krijgt tot de applicatie. De integratie binnen Angular verliep via de Microsoft Authentication Library voor Angular (Microsoft, z.d.-c). Deze kant-en-klare oplossing maakt veilig inloggen via Entra ID mogelijk en zorgt zowel voor identificatie van de gebruiker (authenticatie) als voor controle op welke delen van de applicatie toegankelijk zijn (autorisatie).

Hoewel deze technologie aanvankelijk onbekend was, werd de werking snel duidelijk en kon de implementatie vlot worden uitgevoerd. Deze authenticatielaag vormt de basis van de applicatie, aangezien bepaalde gegevens veilig opgehaald en doorgestuurd moeten worden.

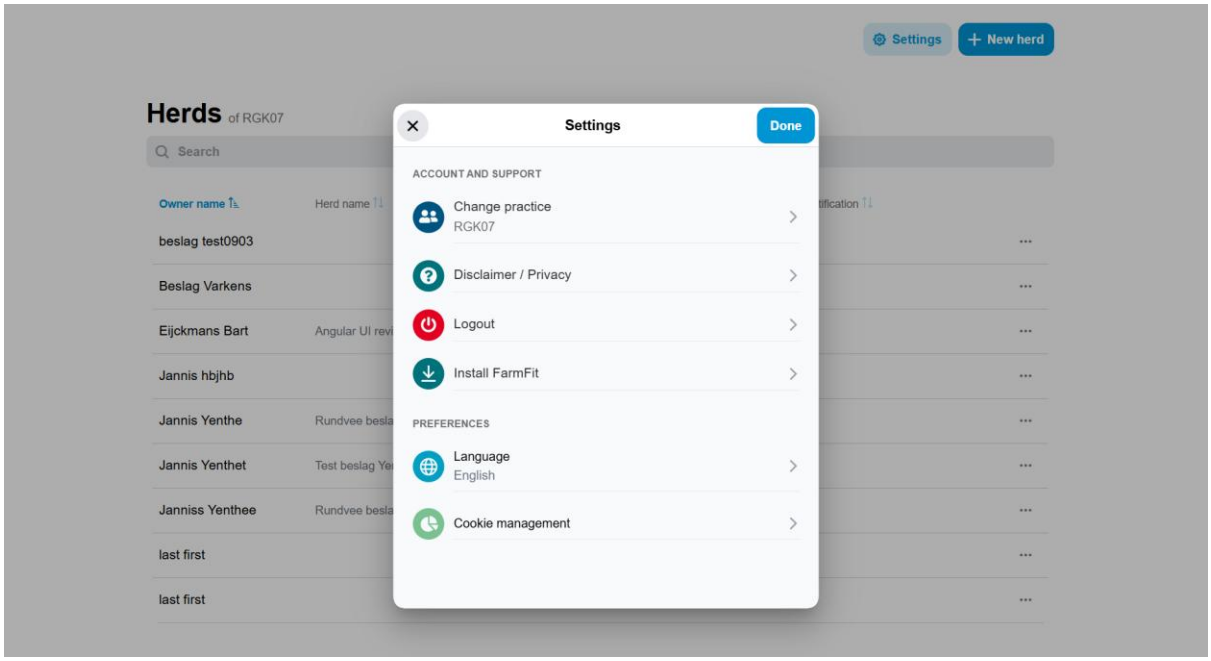
De keuze voor Entra ID en MSAL Angular kwam vanuit Cegeka. Voor de toegang tot de applicatie werden guards ingesteld. Een guard fungeert als een "beveiligingspoort" binnen de applicatie: deze controleert of een gebruiker toegang mag krijgen tot bepaalde routes of pagina's.

Daarnaast wordt gebruikgemaakt van Single Sign-On (SSO). Dit betekent dat gebruikers slechts één keer moeten inloggen en dat ze geen enkel deel van de applicatie kunnen bekijken voordat ze aangemeld zijn. Het inlogscherm is daarom het eerste en enige scherm dat zichtbaar is wanneer een gebruiker de applicatie opent zonder ingelogd te zijn.

Soorten guards die we gebruiken:

- **Authenticated User Guard:** controleert of een gebruiker daadwerkelijk is ingelogd en wordt toegepast op de hele applicatie.
- **User Group Guards:** controleren of de gebruiker tot de juiste user group behoort, aangezien gebruikers soms proberen om routes rechtstreeks te benaderen. Een user group is gekoppeld aan de praktijk van de gebruiker, zodat enkel geautoriseerde personen toegang hebben tot specifieke functies of pagina's.

3.8 Gebruiker instellingen



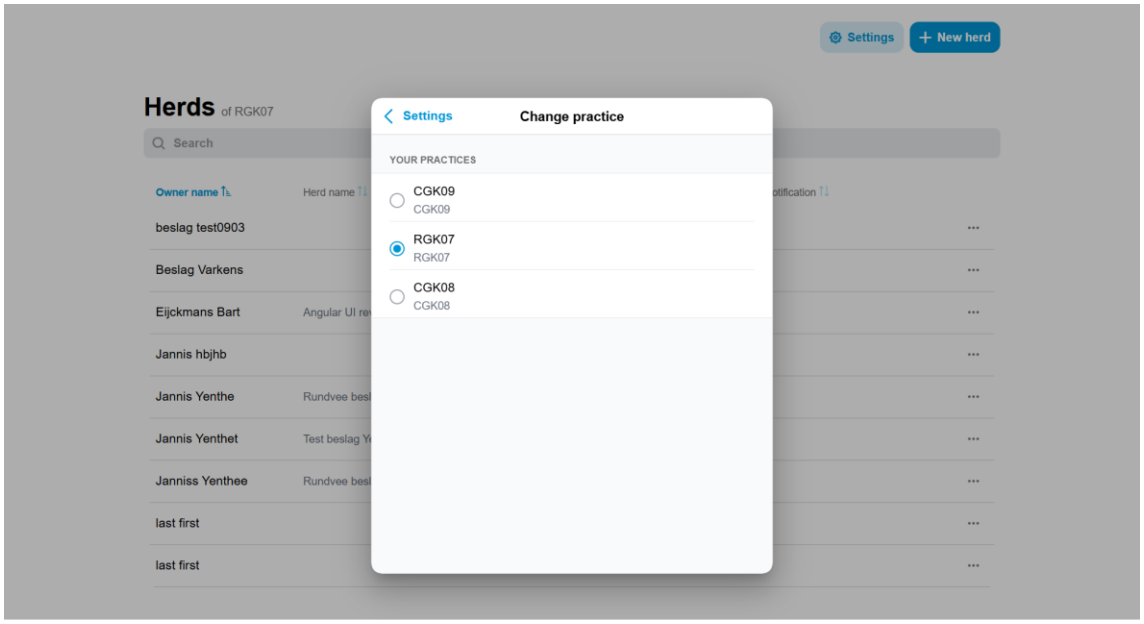
Figuur 11. Settings-modal binnen FarmFit, met opties voor praktijkkeuze, disclaimer- en privacybeleid, uitloggen, PWA-installatie, taalbeheer en cookiebeheer.

Dit hoofdstuk beschrijft de functionaliteiten die een gebruiker in de instellingen kan aanpassen om de applicatie optimaal te kunnen gebruiken. De belangrijkste instellingen zijn:

- Van praktijk wisselen
- Disclaimer of privacy policy bekijken
- Uitloggen
- FarmFit PWA installeren
- Taal wijzigen
- Cookiebeheer (out-of-scope)

3.8.1 Praktijk wijzigen:

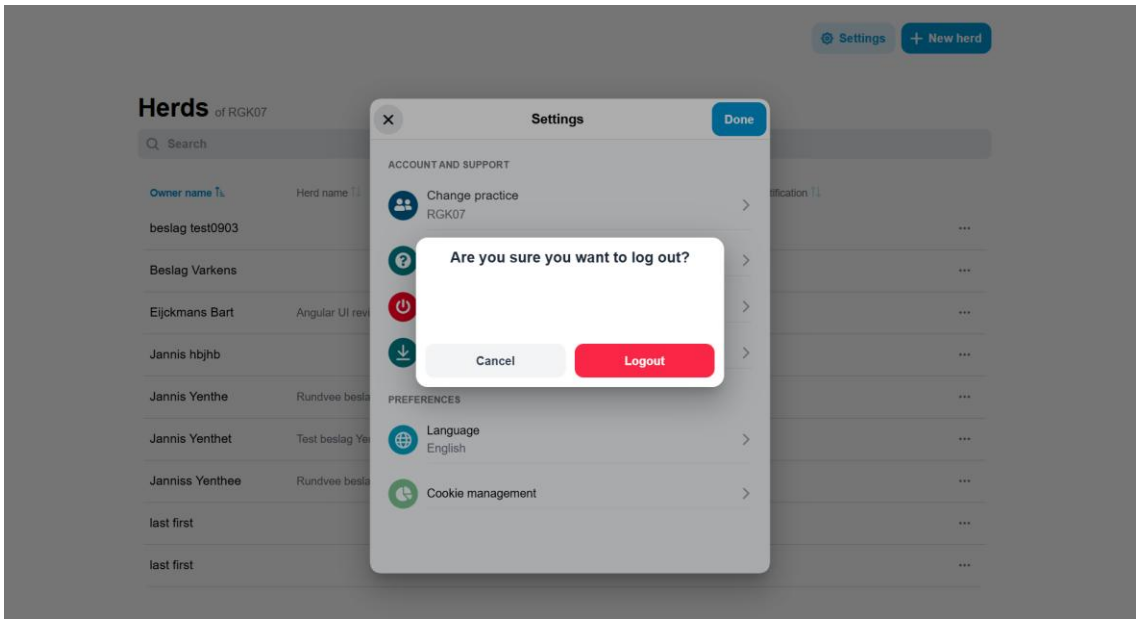
Door de praktijk te wijzigen, krijgt de gebruiker andere data te zien in de applicatie, zoals beslagen, bezoeken, meldingen en andere relevante informatie. Dit is een cruciale functionaliteit, omdat de applicatie op die manier altijd de gegevens toont die bij de gekozen praktijk horen. Wanneer de gebruiker een praktijk aanklikt, veranderen de data en het praktijknummer achter de modal mee.



Figuur 12. Functionaliteit "Praktijk wijzigen": de gebruiker selecteert een praktijk uit de beschikbare lijst, waarna de getoonde gegevens automatisch worden aangepast.

3.8.2 Afmelden:

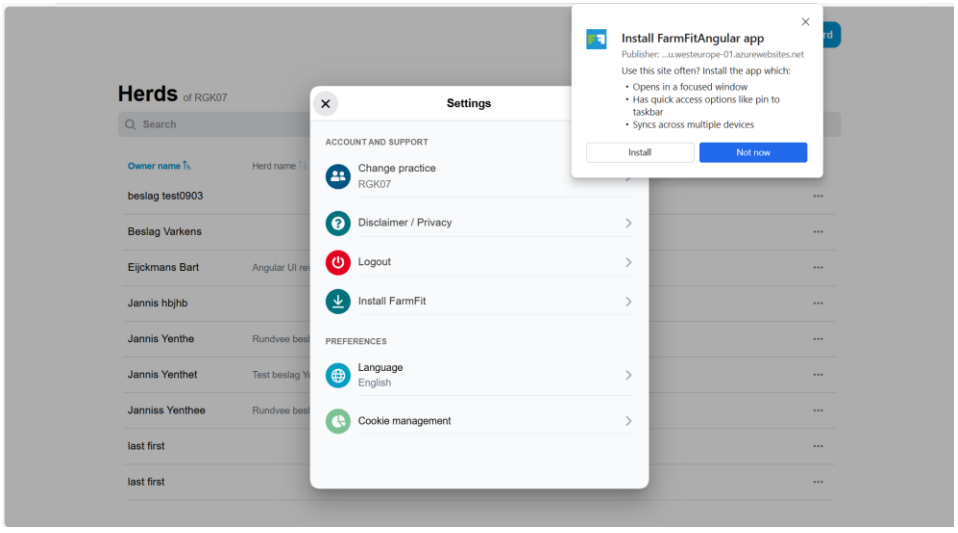
Een gebruiker moet uiteraard de mogelijkheid hebben om uit te loggen. Via deze functionaliteit kan veilig worden afgemeld van de applicatie. Na het afmelden heeft de gebruiker geen toegang meer en wordt deze automatisch doorgestuurd naar de inlogpagina.



Figuur 13. Bevestigingsprompt bij het uitloggen, die voorkomt dat de gebruiker ongewild wordt afgemeld.

3.8.3 Installeren van de PWA (Progressive Web App):

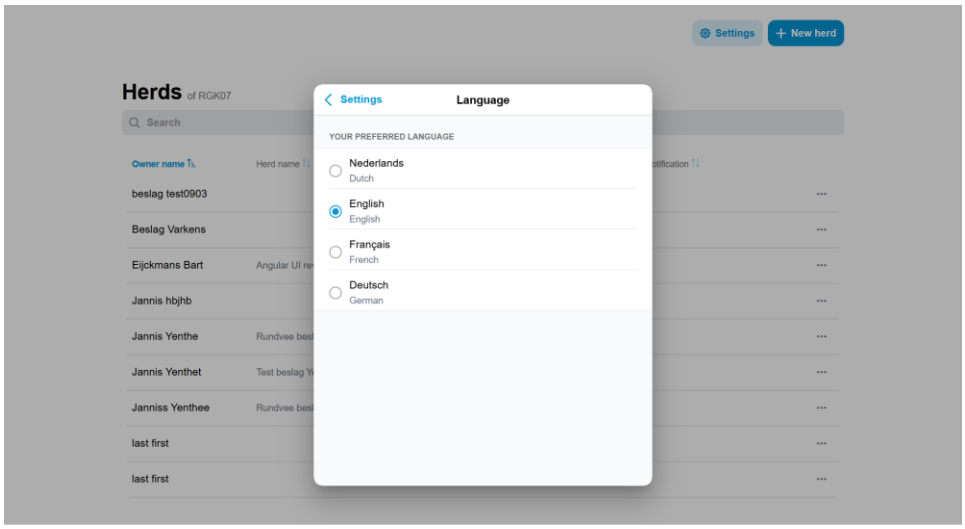
Via deze optie opent de gebruiker een browser-popup om de applicatie als PWA te installeren. Een PWA werkt als een app op het apparaat van de gebruiker en kan zelfs offline gebruikt worden.



Figuur 14. Browser-popup waarmee de gebruiker FarmFit als Progressive Web App op het apparaat kan installeren.

3.8.4 Taal wijzigen:

Met deze functionaliteit kan de gebruiker de taal van de applicatie aanpassen. De vertalingen worden onmiddellijk toegepast, zodat de applicatie meteen in de gewenste taal beschikbaar is. De implementatie verloopt via ngx-translate.



Figuur 15. *Functionaliteit "Taal wijzigen", waarbij de gebruiker kiest tussen Nederlands, Engels, Frans en Duits. De vertalingen worden direct toegepast via ngx-translate.*

3.8.5 Cookiebeheer:

De gebruiker kan cookies beheren via een externe dienst genaamd Cookiebot.js. Omdat dit onderdeel afhankelijk is van een licentie die niet beschikbaar was tijdens de stage, valt deze functionaliteit buiten de scope en werd er geen implementatie voor uitgewerkt.

3.9 Beslagen aanmaken/bekijken/bewerken

Onderliggende structuur en aanpak

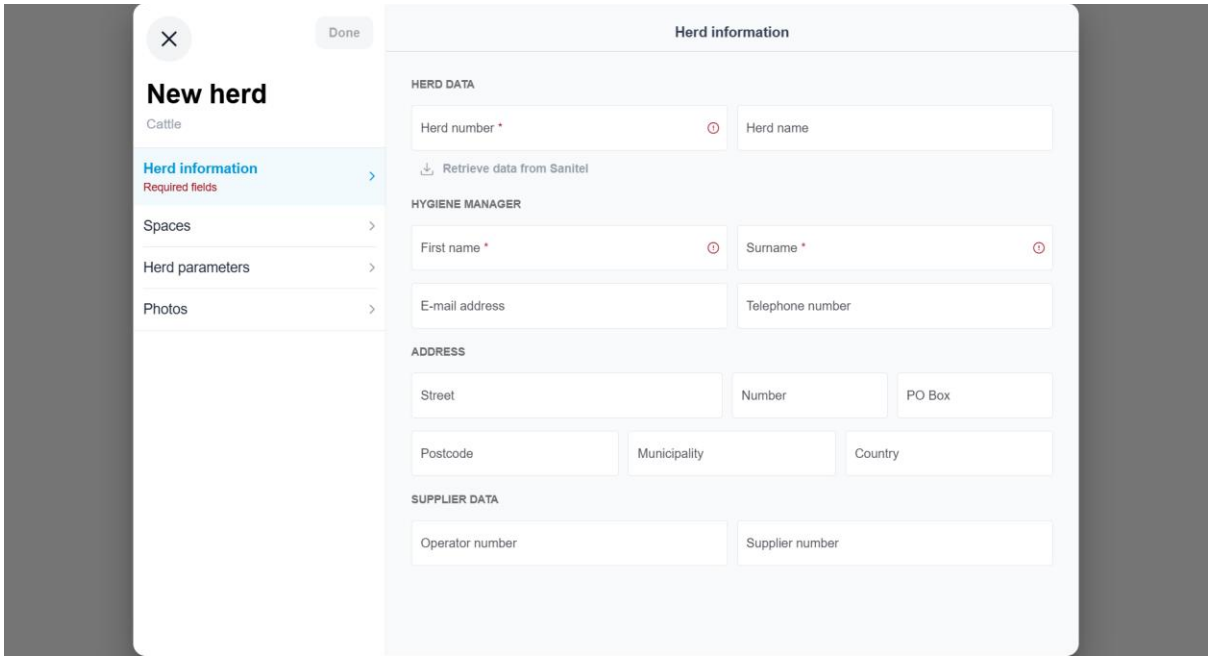
Bij de ontwikkeling van de pagina's voor beslagen werd duidelijk dat een maintainable (onderhoudbare) en reusable (herbruikbare) aanpak essentieel is. Veel pagina's binnen de applicatie hebben immers een vergelijkbare structuur, zoals bezoeken, meldingen en detailpagina's van een beslag.

Layout

Voor de layout werden twee hoofdcomponenten ontworpen die op meerdere plaatsen worden hergebruikt:

- **Sidebar:** bevindt zich aan de linkerkant van de pagina en toont verschillende acties afhankelijk van de context, zoals *Edit*, *View* of *Create*. Daarnaast geeft de sidebar een overzicht van de stappen die nodig zijn voor de specifieke functionaliteit, zodat de gebruiker altijd weet waar deze zich in het proces bevindt.
- **Dialog:** vormt de hoofdcontainer van de pagina, waarin de sidebar links staat en rechts de content via andere componenten wordt geladen. Deze structuur wordt consistent toegepast bij beslagen, bezoeken en meldingen.

Volledige layout:



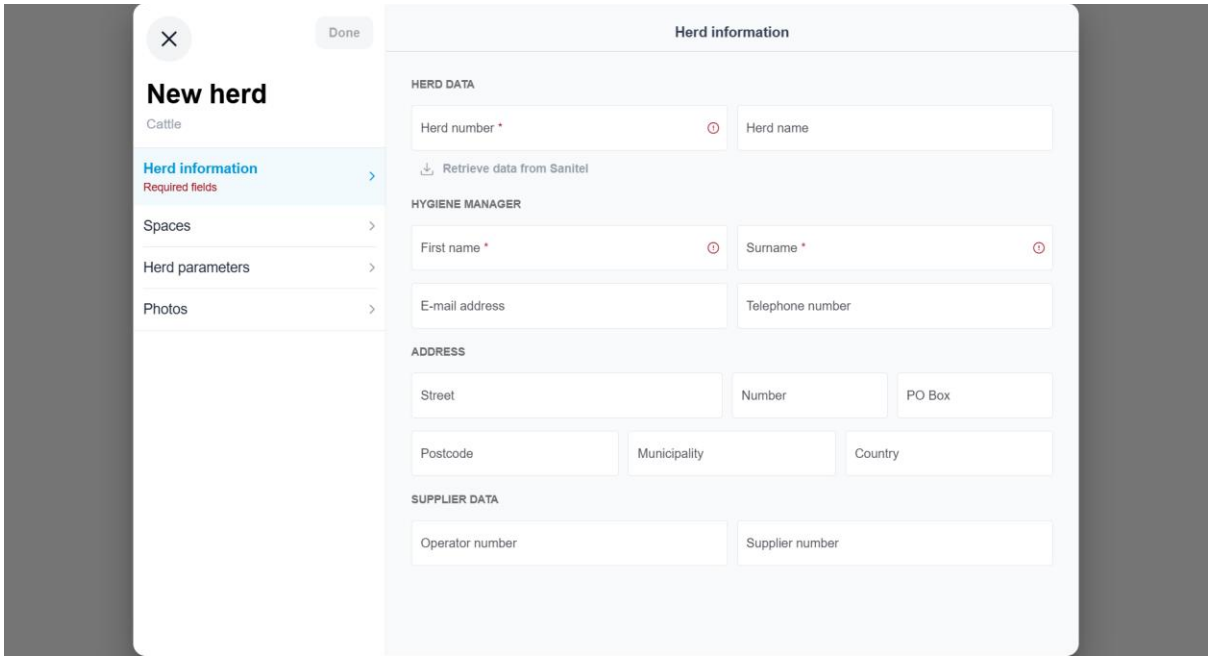
Figuur 16. Volledige layout van de aanmaakflow voor beslagen, met links de sidebar voor stapnavigatie en rechts het content-paneel.

3.9.1 Aanmaken van een beslag

Het aanmaken van een beslag begint vanaf het dashboard, waar de gebruiker eerst het type beslag kiest. Vervolgens wordt de layout geladen zoals hierboven beschreven, met de verschillende stappen:

3.9.1.1 Beslaginformatie:

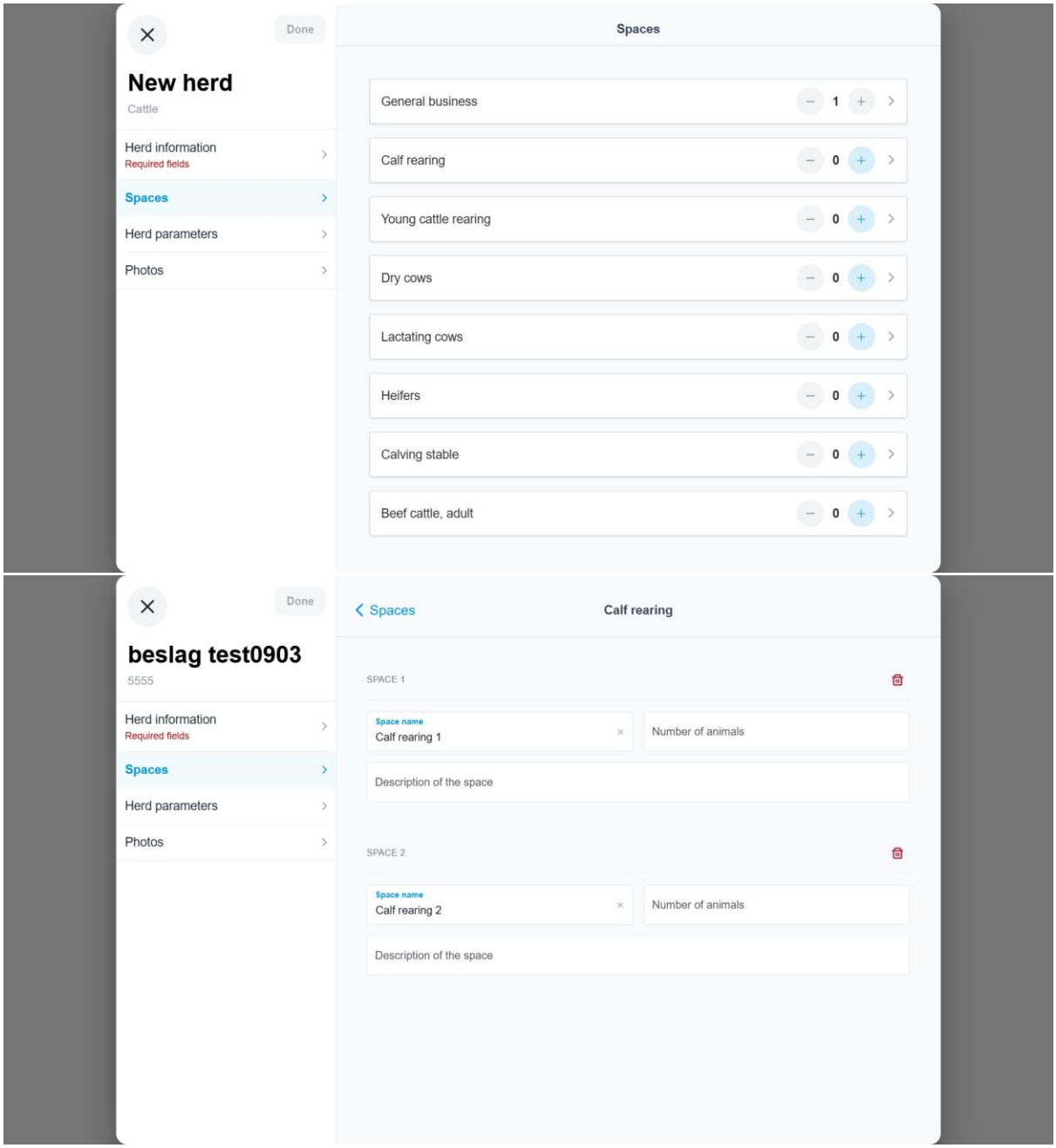
- Verplichte velden sturen hun status of waarde naar de sidebar, zodat zichtbaar is of deze stap volledig is.
- Op het beslagnummer wordt een regex-check uitgevoerd om te controleren of het correct is ingevuld.
- Wanneer het beslagnummer correct is, wordt een backend-endpoint aangeroepen bij Sanitel. Door technische beperkingen kon dit niet volledig getest worden, maar de functionaliteit werd geïmplementeerd zodat ze later wel getest kan worden.



Figuur 17. Stap "Beslaginformatie" tijdens het aanmaken van een beslag, met verplichte velden, regex-validatie op het beslagnummer en een knop voor het ophalen van data uit Sanitel.

3.9.1.2 Afdelingen stap (Spaces):

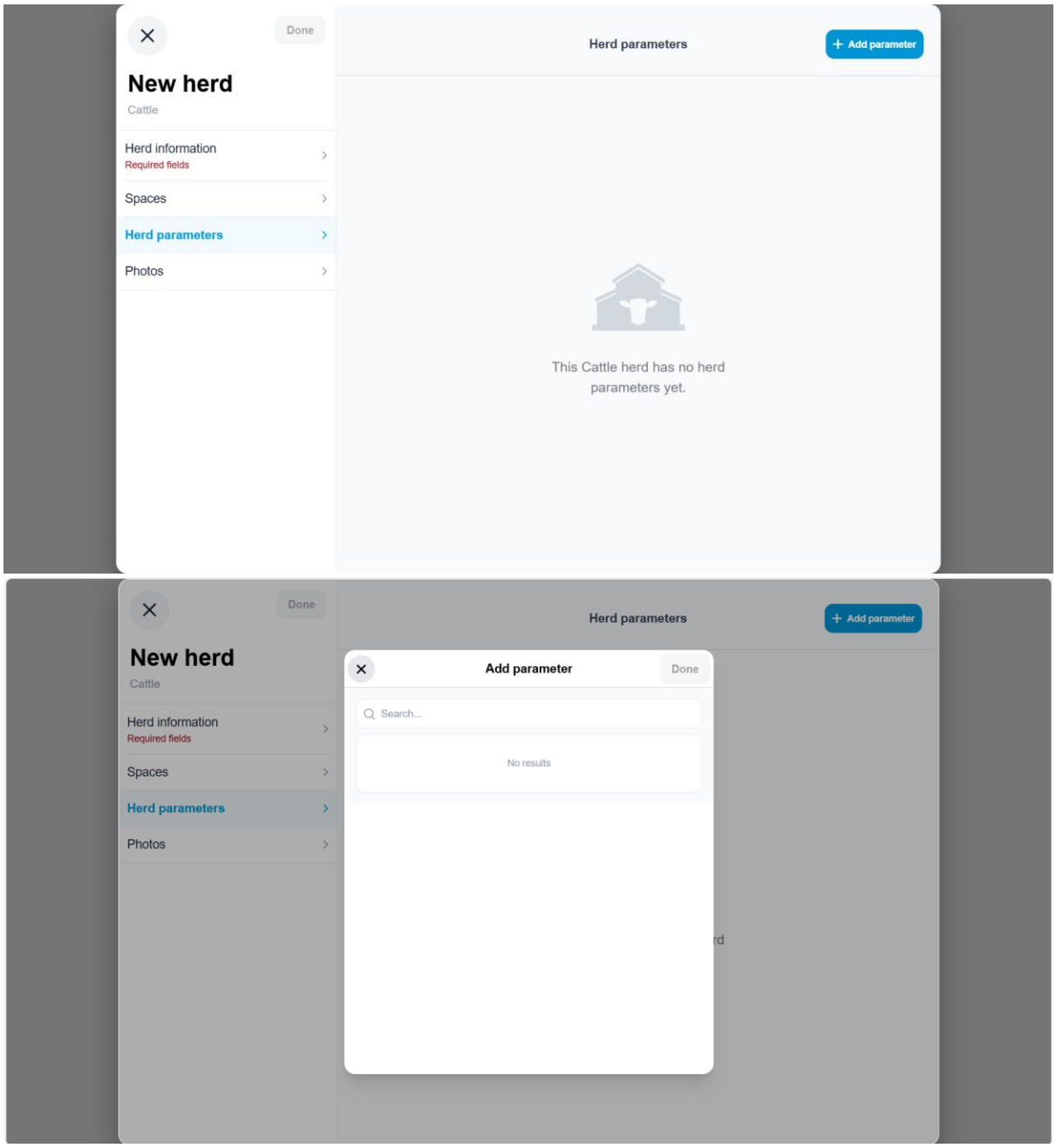
- Hier worden de afdelingen weergegeven waarin verschillende waarden kunnen worden ingevuld.
- Sommige afdelingen hebben minimum- en maximumwaarden, afkomstig uit de API-responses. Velden worden dynamisch weergegeven en uitgeschakeld waar nodig.



Figuur 18. Stap "Afdelingen (Spaces)": boven het overzicht van beschikbare afdelingen met minimum- en maximumwaarden, onder het detailscherm voor het invullen van afzonderlijke ruimtes per afdeling.

3.9.1.3 Beslagparameters stap:

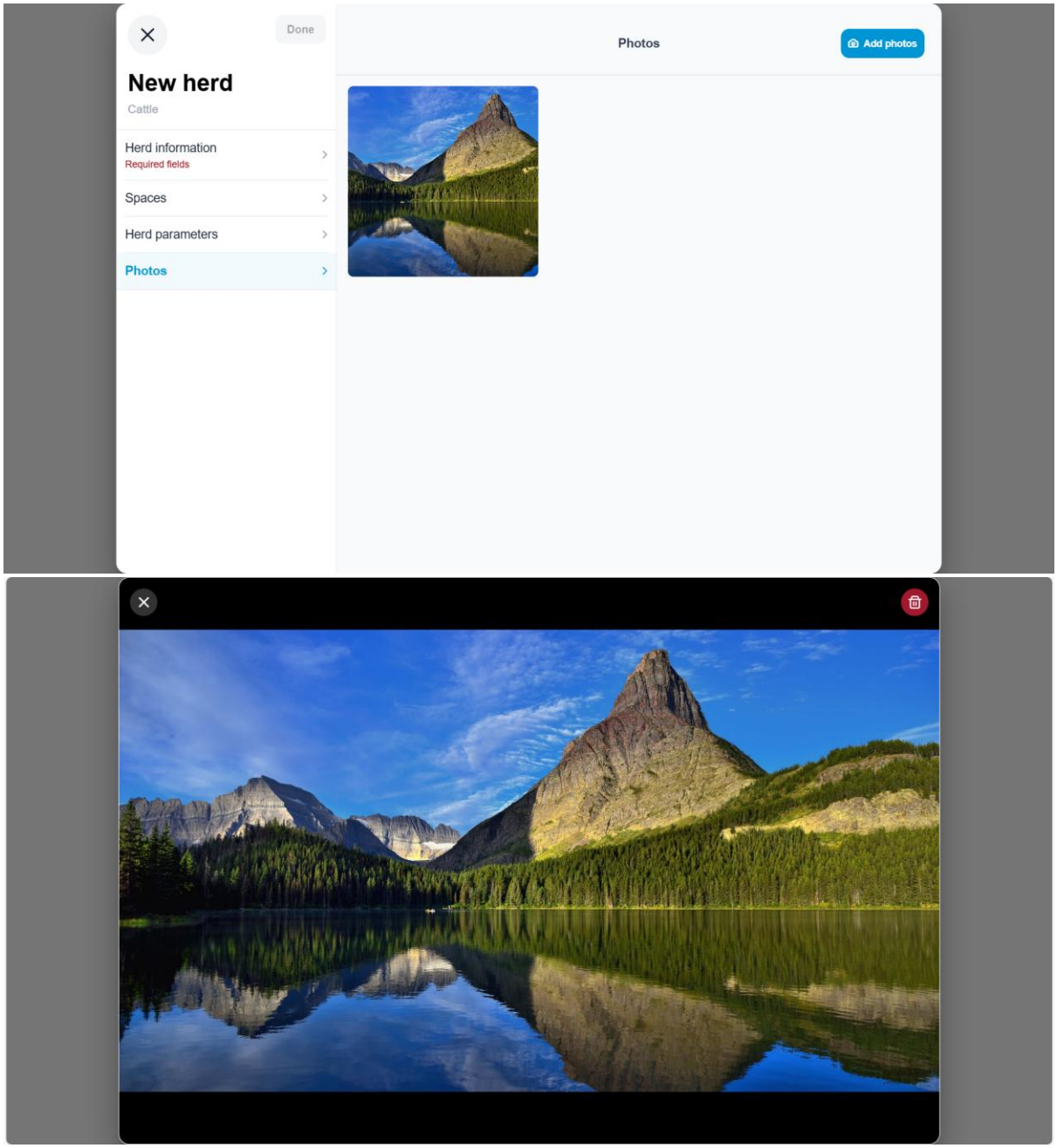
Vanwege beperkte toegang tot data en de beschikbare mockdata kon enkel met lege waarden worden gewerkt. De functionaliteit werd zo ver mogelijk uitgewerkt, maar niet volledig afgerond.



Figuur 19. Stap "Beslagparameters": boven de lege state wanneer nog geen parameters zijn toegevoegd, onder de modal waarmee een parameter kan worden gezocht en toegevoegd.

3.9.1.4 Foto’s stap:

- Deze stap maakt gebruik van een herbruikbare component die ook bij bezoeken en meldingen wordt ingezet.
- Foto's kunnen worden geüpload, automatisch gecomprimeerd, getoond, vergroot bekeken of verwijderd.

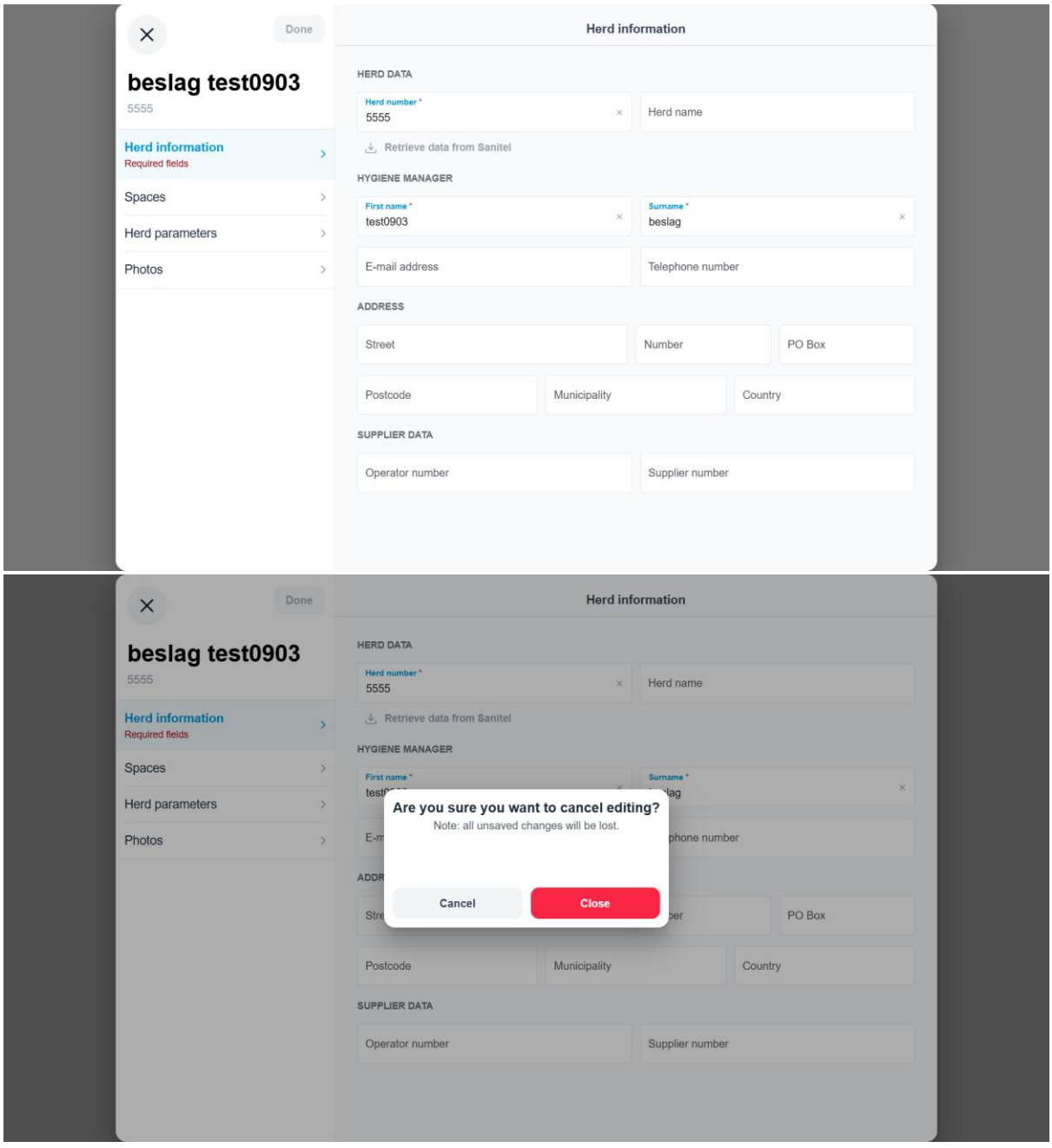


Figuur 20. Stap "Foto's" tijdens het aanmaken van een beslag: foto's kunnen worden geüpload, automatisch gecomprimeerd, in vergrote weergave bekeken en verwijderd.

3.9.2 Bewerken van een beslag

De bewerkingsmodus lijkt sterk op de aanmaakmodus, met enkele verschillen:

- De waarden zijn vooraf ingevuld om de gebruikerservaring te verbeteren.
- De actieknoppen in de sidebar zijn aangepast aan de bewerkingscontext.
- Bij het sluiten van een pagina met onopgeslagen wijzigingen verschijnt een bevestigingsprompt om gegevensverlies te voorkomen

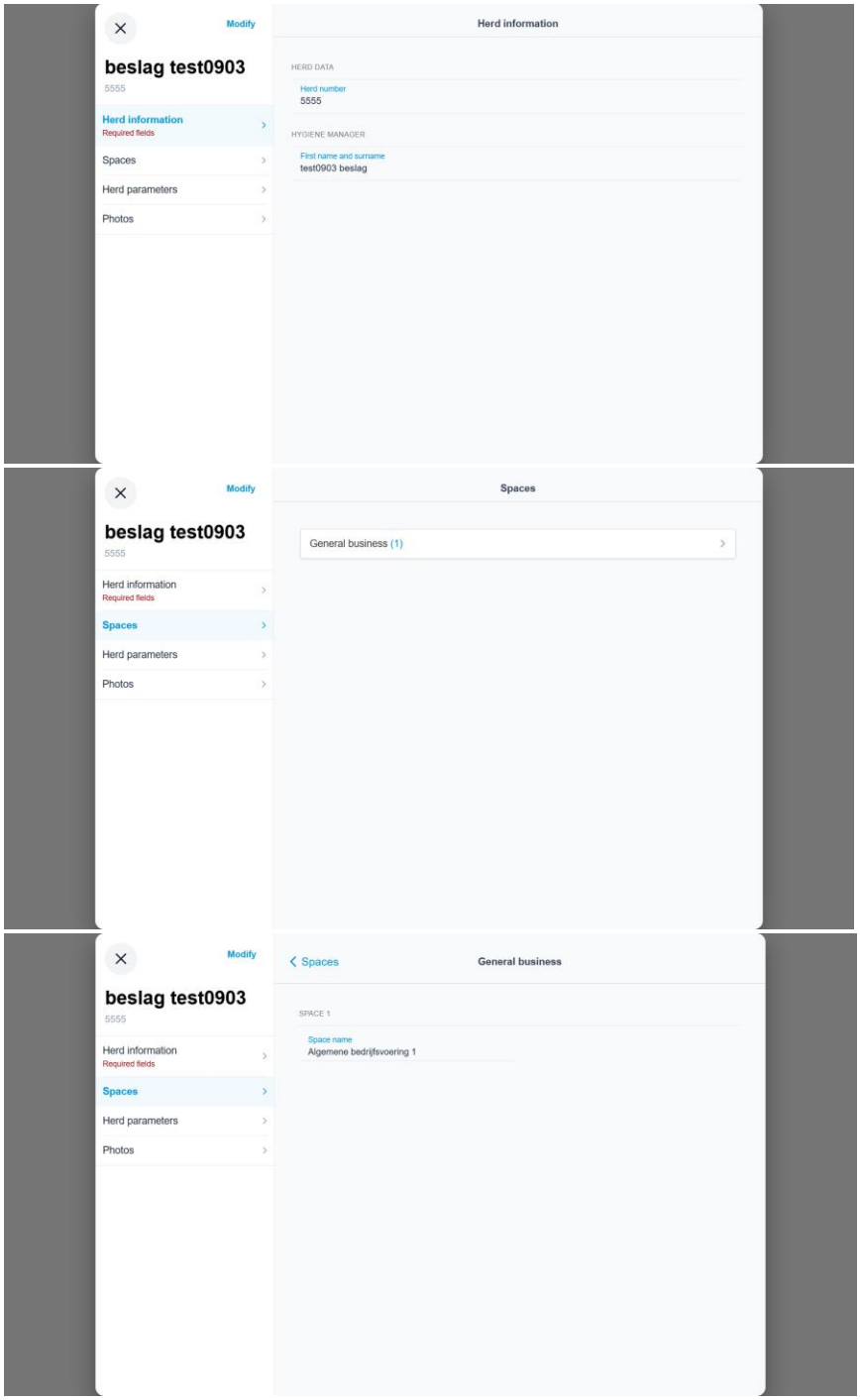


Figuur 21. Bewerkingsmodus van een beslag: boven het scherm met vooraf ingevulde waarden, onder de bevestigingsprompt die verschijnt bij onopgeslagen wijzigingen.

3.9.3 Bekijken van een beslag

Bij het bekijken van een beslag gelden de volgende regels:

- Enkel de velden en waarden die daadwerkelijk zijn ingevuld, worden getoond.
- In de sidebar wordt de optie weergegeven om het beslag te wijzigen.



Figuur 22. Bekijkmodus van een beslag op drie niveaus: het overzicht van de beslaginformatie (boven), de geselecteerde afdelingen (midden) en het detail van een afdeling (onder).

3.10 Bezoeken

Dit hoofdstuk licht toe hoe een bezoek wordt aangemaakt, hoe de functionaliteit tot stand kwam, welke problemen zich voordeden en welke inzichten daaruit voortvloeiden. Het bezoek vormde de meest uitdagende functionaliteit van de applicatie.

De uitdaging: dynamisch tonen van parameters en bijbehorende vragen

Bij de observatiestap (verder uitgelegd in 3.10.2) wordt gewerkt met sjablonen uit de database die specifiek kunnen zijn voor één partij of voor meerdere partijen. Hierdoor moeten in de frontend meerdere endpoints (drie tot vier) worden opgeroepen om alle benodigde gegevens op te halen.

Vervolgens worden deze gegevens geflattend en gefilterd op het sjabloon en de partij waartoe de gebruiker op dat moment behoort.

De uitdaging hierbij is dat er veel data beschikbaar is, weinig overzicht ontstaat en veel manipulatie in de frontend nodig is. De reden hiervoor is dat al deze gegevens lokaal beschikbaar moeten zijn opdat de applicatie offline kan blijven werken. Indien deze combinatie volledig op de backend zou gebeuren, zou de load te zwaar worden en zou het systeem te traag reageren.

Dynamisch tonen van vragen

Na het ophalen en verwerken van alle data ontstaat een lijst van parameters per afdeling. Deze parameters kunnen verschillende typen hebben: tekst, numeriek, datum, multi-select of multi-choice.

Elke parameter bevat een display logic in de response, vaak in de vorm van een regex-formule die bepaalt welke velden wel of niet getoond worden afhankelijk van eerdere keuzes.

```
const regex = /question\[(\d+)\]\['([^']+)'\](==|!=|<=|>=|<|>)"([^"]*)"/;
const match = displayLogic.match(regex);
```

Figuur 23. Regex-formule uit de display logic van een parameter. Op basis van eerdere antwoorden bepaalt deze formule welke volgende vragen wel of niet worden getoond.

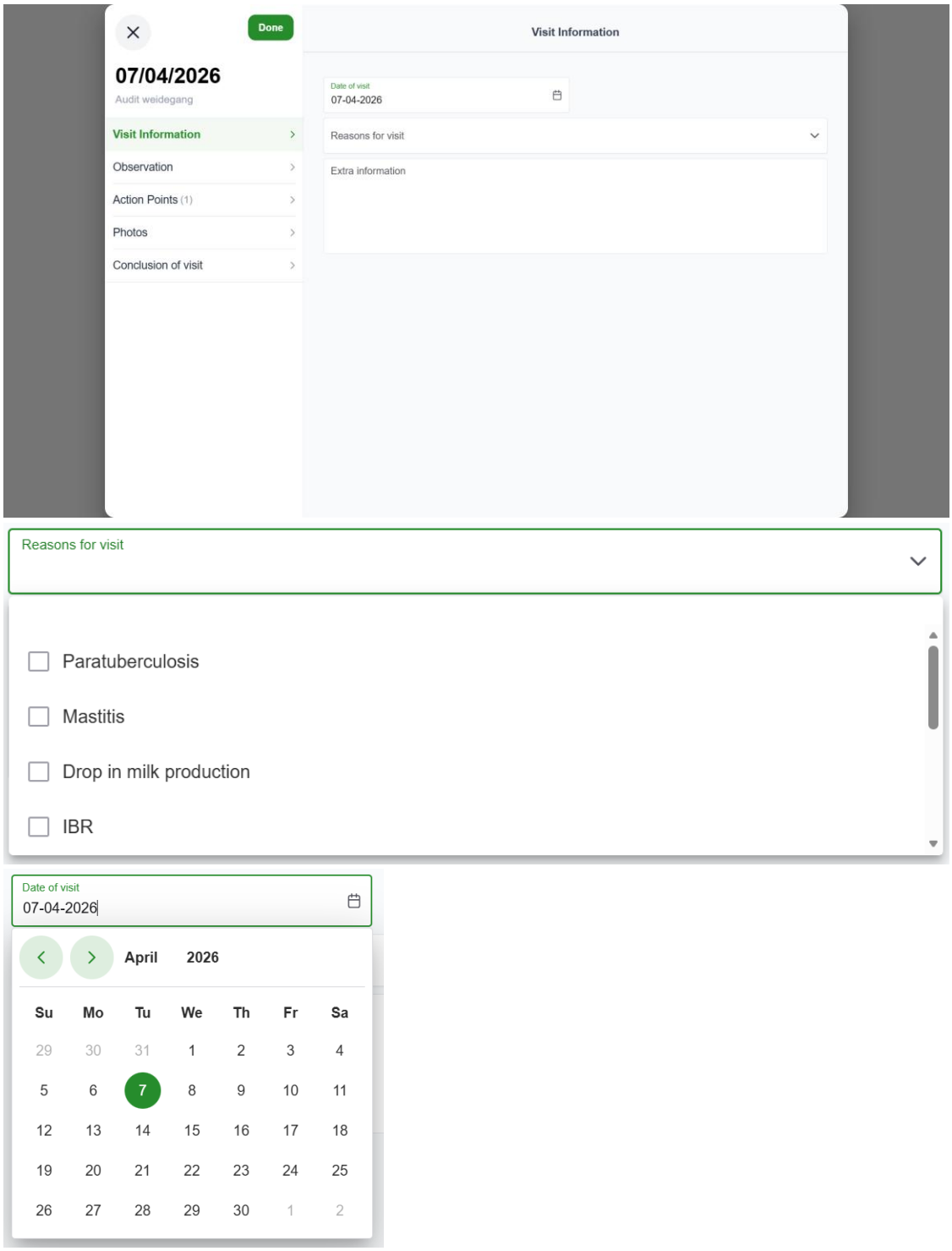
Wanneer bijvoorbeeld waarde 1 van vraag 1 wordt gekozen, verschijnt vraag 2 automatisch. Dit maakt de functionaliteit dynamisch en flexibel, maar ook complex om te realiseren en te analyseren.

Aanpak

Om deze complexiteit beheersbaar te maken, werd gebruikgemaakt van een methode die hier agentic development wordt genoemd. Hierbij werden specifieke agents en plannen opgesteld om de oude code te analyseren en nieuwe stappen te definiëren die de agents in Angular konden uitvoeren. Op die manier kon de functionaliteit stap voor stap worden opgebouwd, met tussentijdse tests om fouten vroegtijdig op te sporen.

3.10.1 Bezoekinformatie

Dit is een algemene stap waarbij de gebruiker de gegevens van een bezoek ingeeft. Hierbij wordt gebruikgemaakt van een aantal PrimeNG-componenten die voorheen nog niet aan bod kwamen, zoals een textarea, datepicker en multiselect.

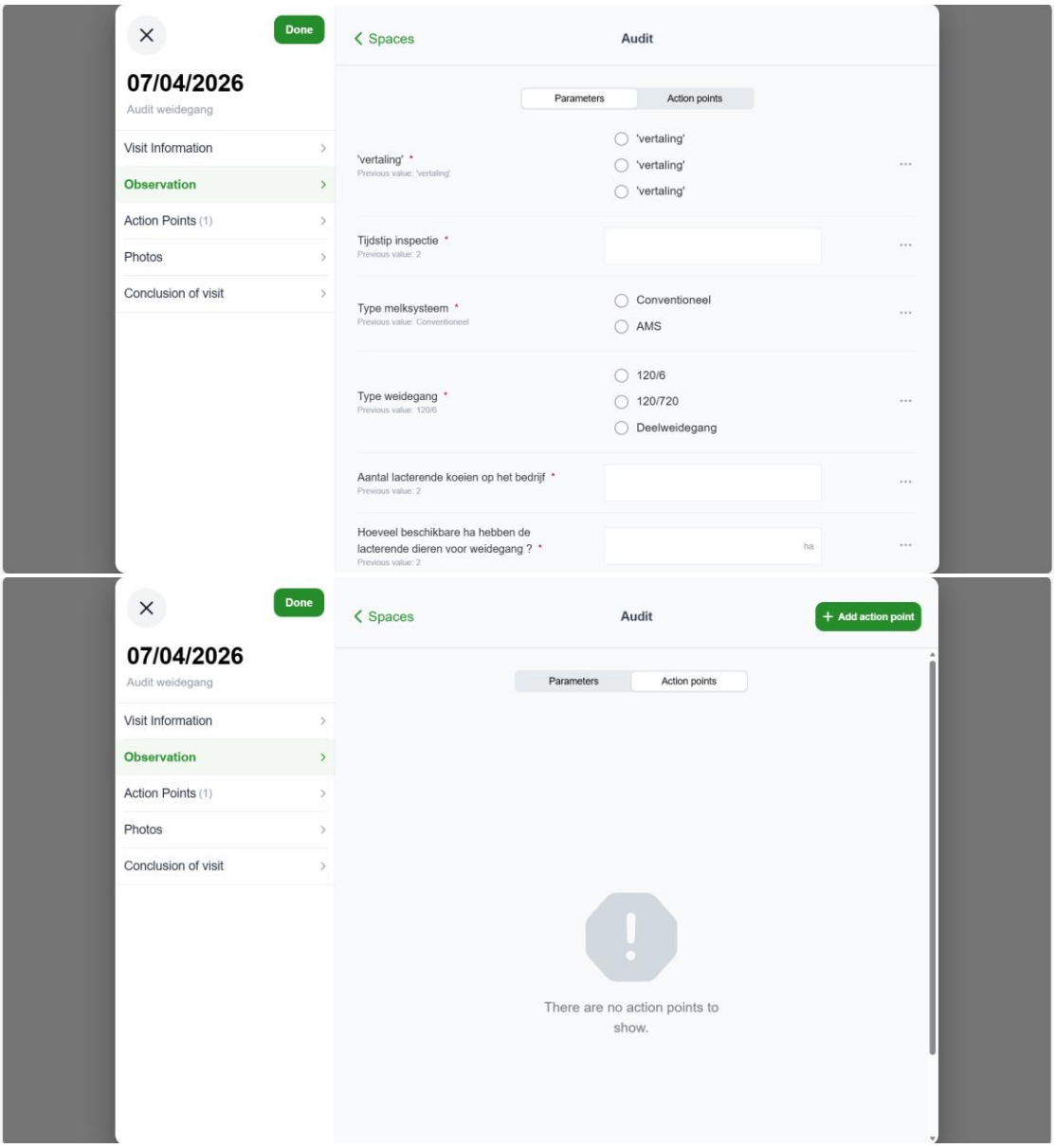


Figuur 24. Stap "Bezoekinformatie" met de gebruikte PrimeNG-componenten: een multiselect voor de reden van het bezoek (midden) en een datepicker voor de bezoekdatum (onder).

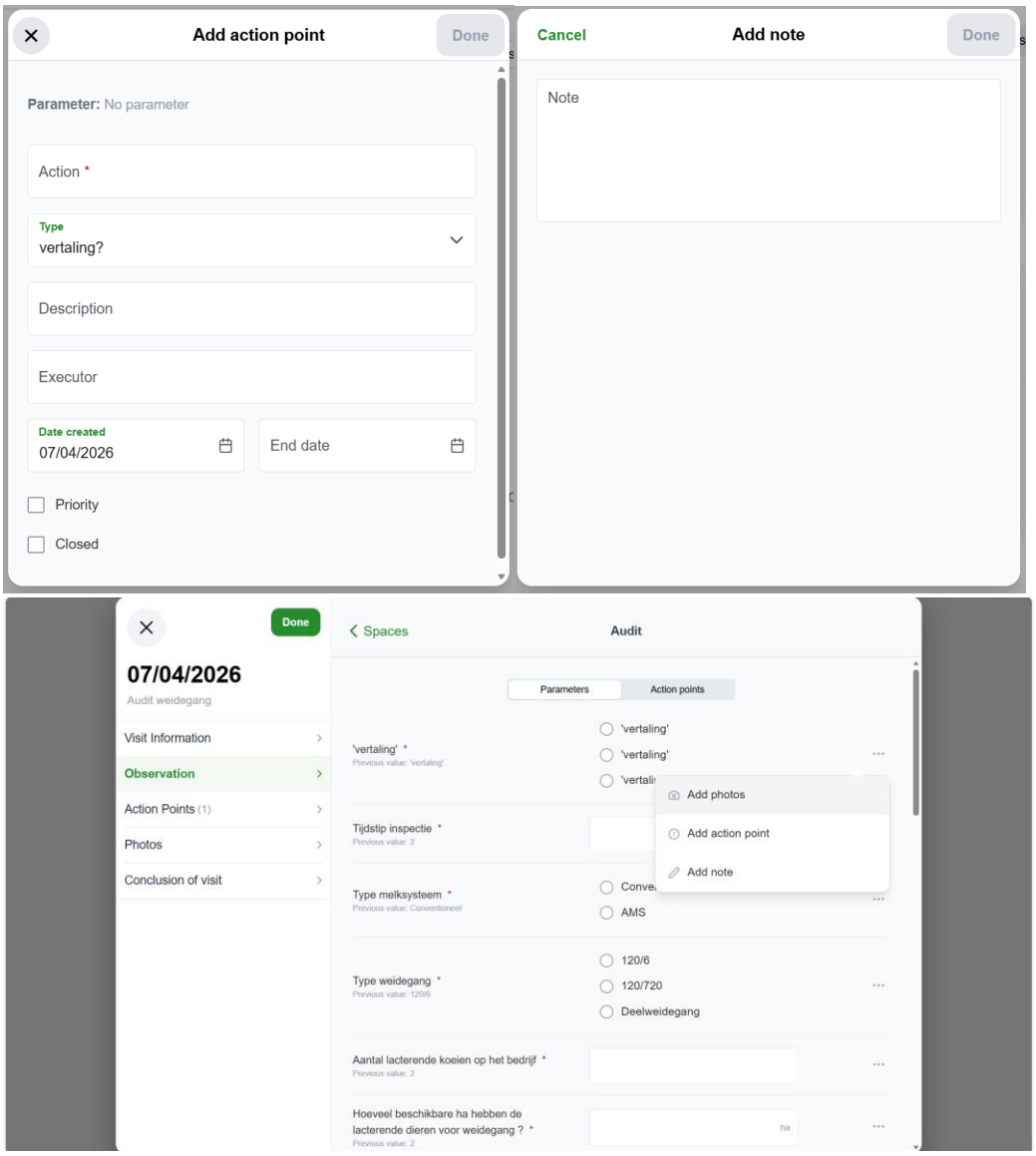
3.10.2 Observaties

Dit was de meest uitdagende stap binnen het project. Hierbij kwamen meerdere functionaliteiten samen:

- Afdelingen bekijken.
- Parameters invullen, waarbij het mogelijk is dat een bepaalde waarde nieuwe vragen activeert.
- Notities toevoegen aan een parameter.
- Actiepunten toevoegen aan een parameter.
- Foto's toevoegen aan een parameter.
- Actiepunten toevoegen in de actiepunten-tab.



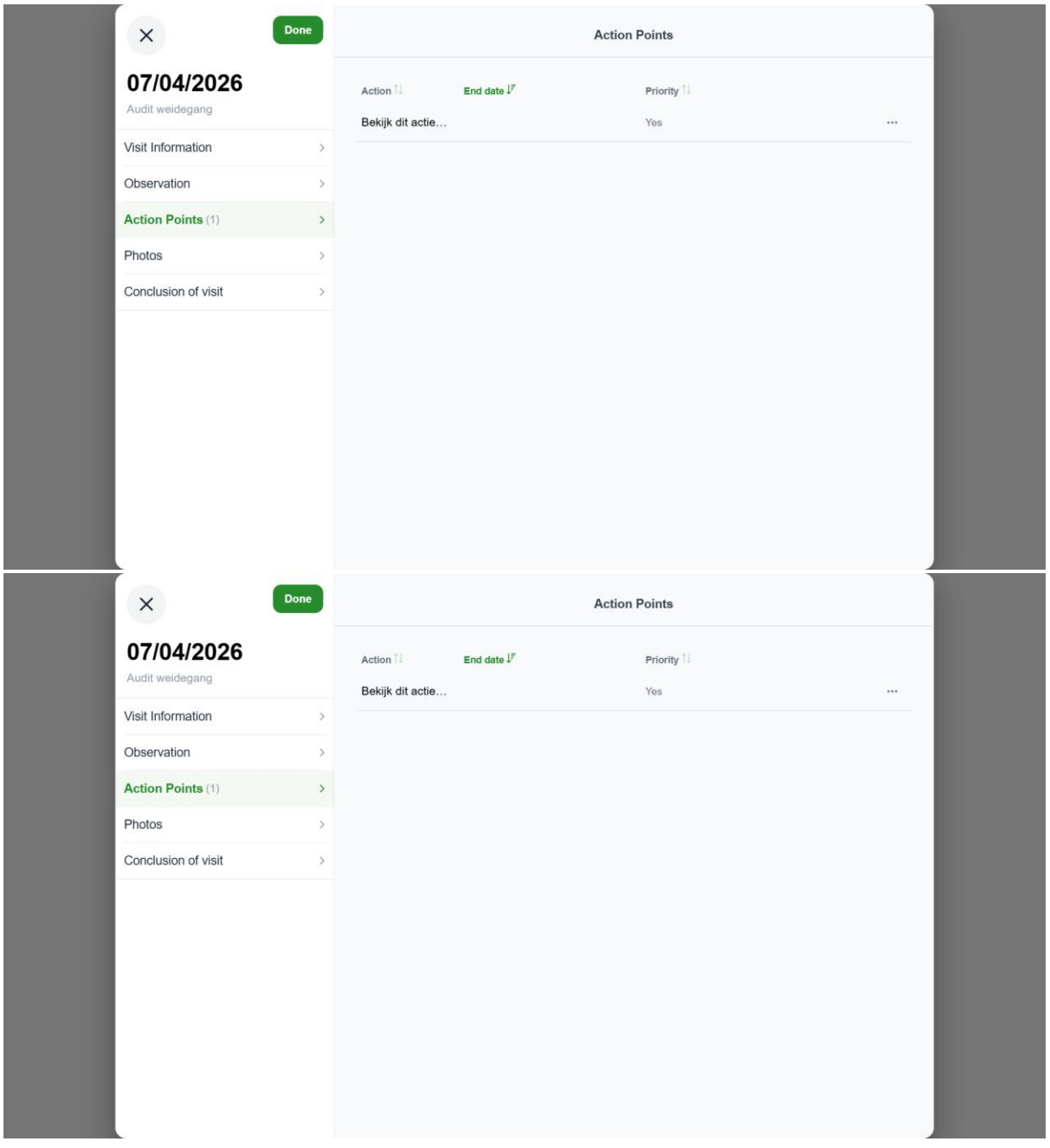
Figuur 25. Stap "Observaties": boven het invullen van parameters per afdeling met dynamisch tonen van vervolgvragen, onder het overzicht van actiepunten binnen deze observatiestap.



Figuur 26. Aanvullende functionaliteiten binnen een observatie: het toevoegen van een actiepunt (links), het toevoegen van een notitie (midden) en het contextmenu per parameter (rechts).

3.10.3 Actiepunten:

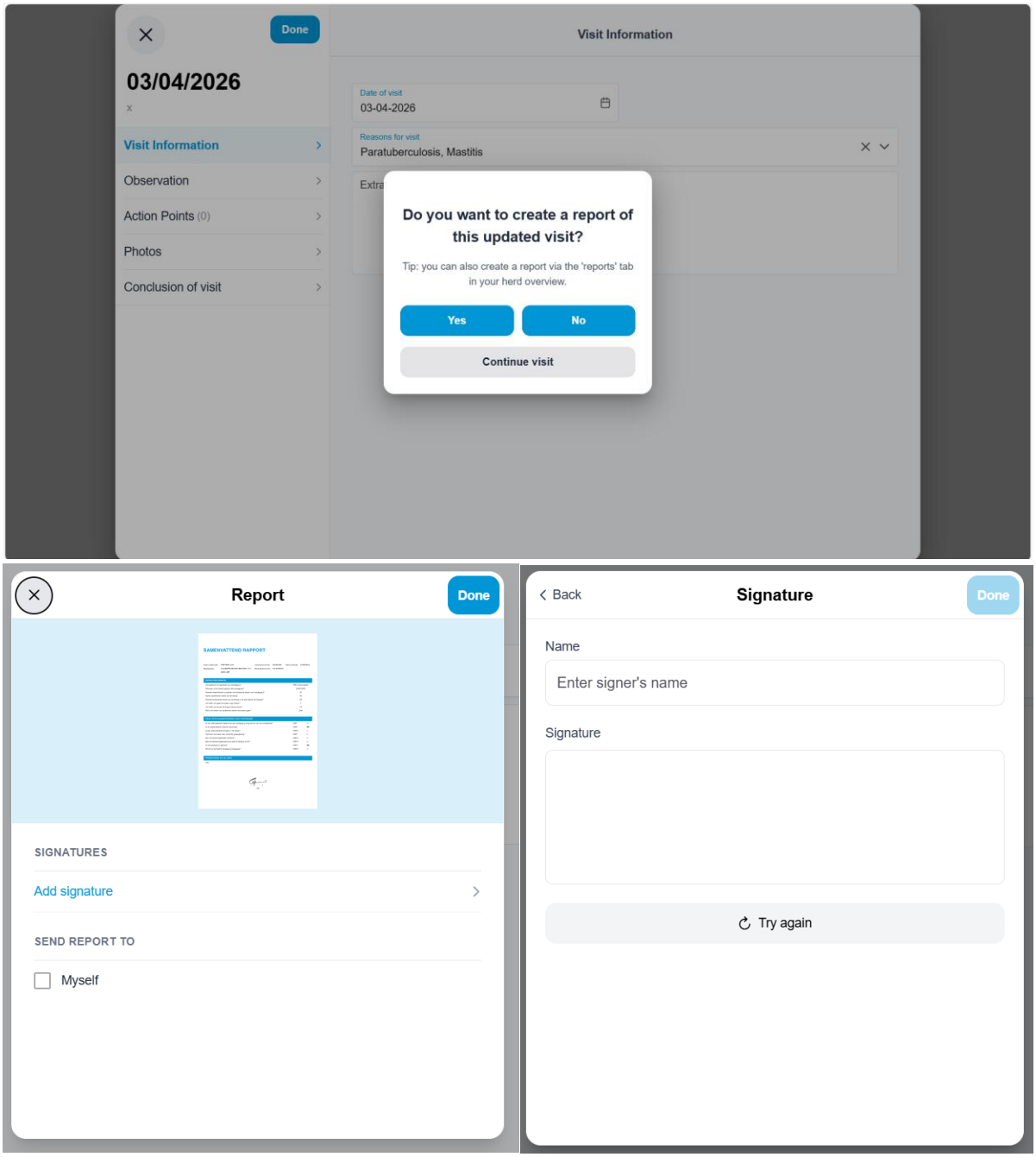
In deze stap wordt een overzicht weergegeven van alle actiepunten die aan andere bezoeken werden toegevoegd. De actiepunten kunnen bekeken en bewerkt worden zolang ze openstaan, en ze kunnen vanuit deze stap ook worden gesloten of verwijderd.



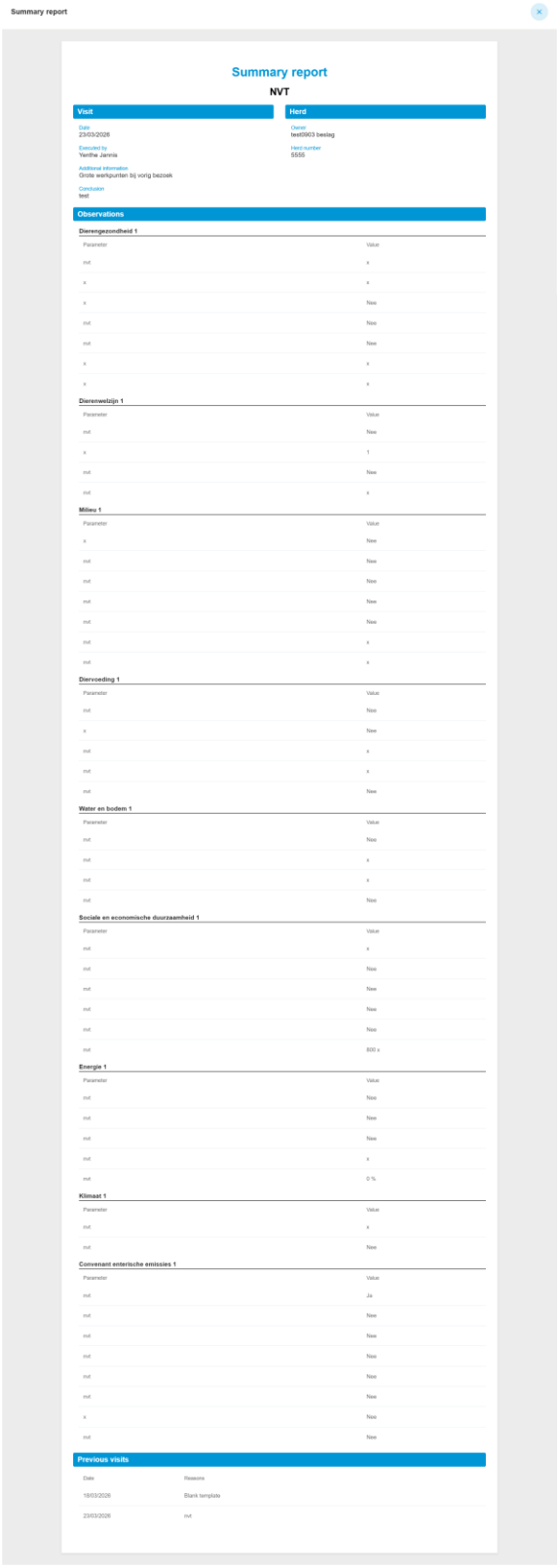
Figuur 27. Stap "Actiepunten": overzicht van openstaande actiepunten uit eerdere bezoeken, met de mogelijkheid om ze te bekijken, bewerken, sluiten of verwijderen.

3.10.4 Rapporten:

Wanneer een bezoek is afgerond, kan er optioneel een rapport worden gegenereerd. Indien hier niet voor wordt gekozen, wordt het bezoek gewoon opgeslagen. Wanneer er wel een rapport wordt aangemaakt, verschijnt een dialoog met een preview, die open geklikt kan worden voor een gedetailleerdere weergave. Indien gewenst kan een handtekening worden toegevoegd. Ten slotte kunnen de ontvangers van het rapport worden geselecteerd, waarna het rapport wordt aangemaakt.



Figuur 28. Genereren van een rapport na een bezoek: links de bevestigingsprompt, midden de preview met optie tot ondertekenen en versturen, rechts het scherm voor het toevoegen van een handtekening.



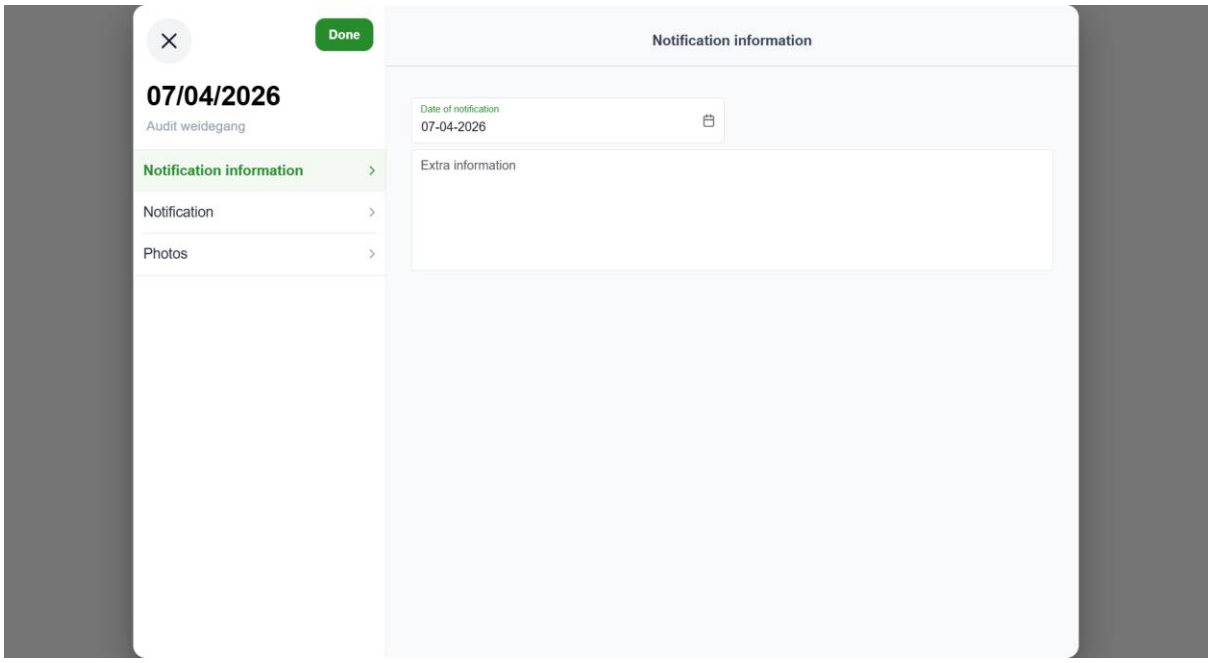
Figuur 29. Voorbeeld van een gegenereerd Summary Report, met de ingevulde observaties per afdeling en de bijhorende waarden.

3.11 Meldingen

De meldingfunctionaliteit is in grote lijnen vergelijkbaar met de bezoekfunctionaliteit. Een melding kan worden aangemaakt, bewerkt en bekeken, en ook het offline werken met backups en synchronisatie verloopt op dezelfde manier. Net zoals bij bezoeken is ook hier de logica nodig voor het tonen van de afdelingen en parameters.

3.11.1 Melding info:

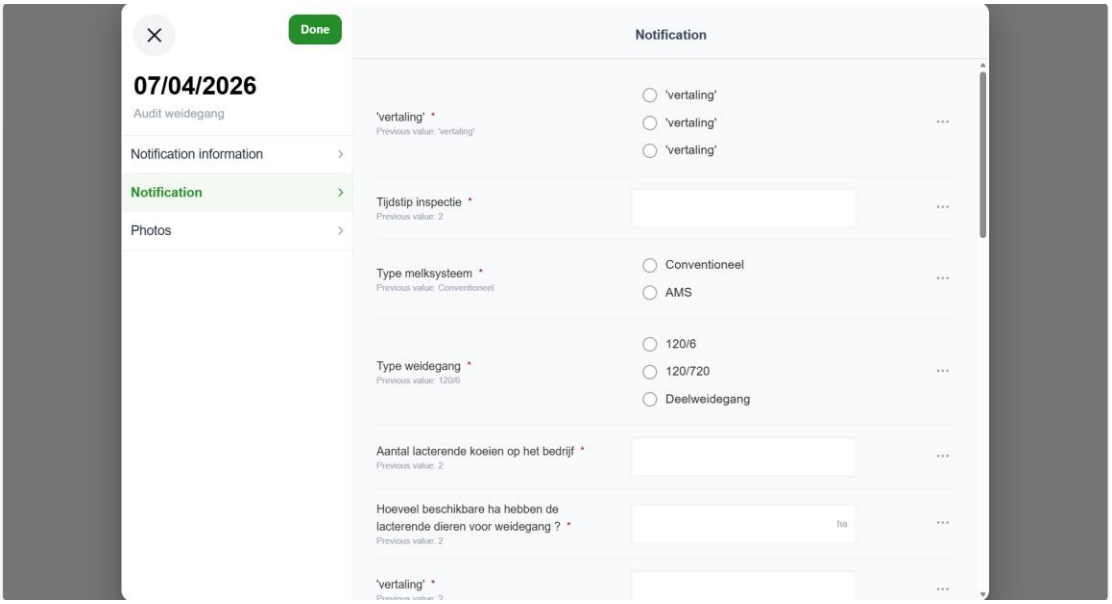
Deze eerste stap bevat twee velden: de datum waarop de melding mag worden aangemaakt en een extra-informatieveld dat de gebruiker optioneel kan invullen.



Figuur 30. Stap "Melding info" bij het aanmaken van een melding, met een datumveld en een veld voor optionele extra informatie.

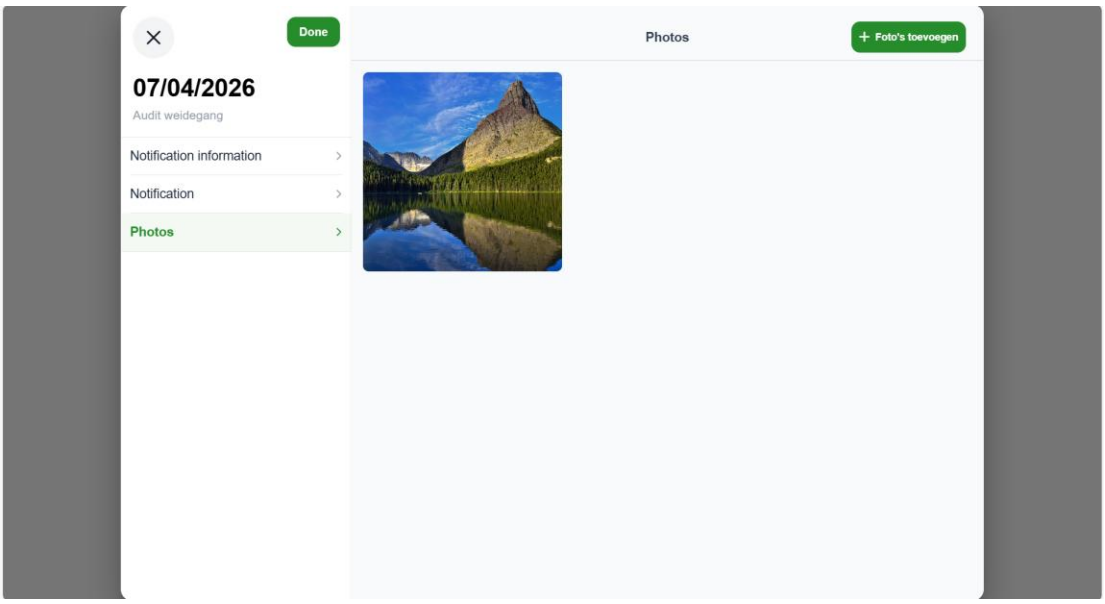
3.11.2 Melding parameters:

Deze stap is vergelijkbaar met de observatiestap van een bezoek. Het verschil is dat er bij meldingen geen actiepunten worden gebruikt, terwijl dezelfde logica voor het tonen van dynamische vragen wel van toepassing is. Afdelingen kunnen worden aangeklikt en de parameters kunnen worden aangevuld. Daarnaast is het mogelijk om een opmerking toe te voegen of een foto te koppelen.



3.11.3 Foto’s:

Deze stap is identiek aan de fotostap bij bezoeken. Foto's kunnen worden toegevoegd, bekeken en verwijderd.

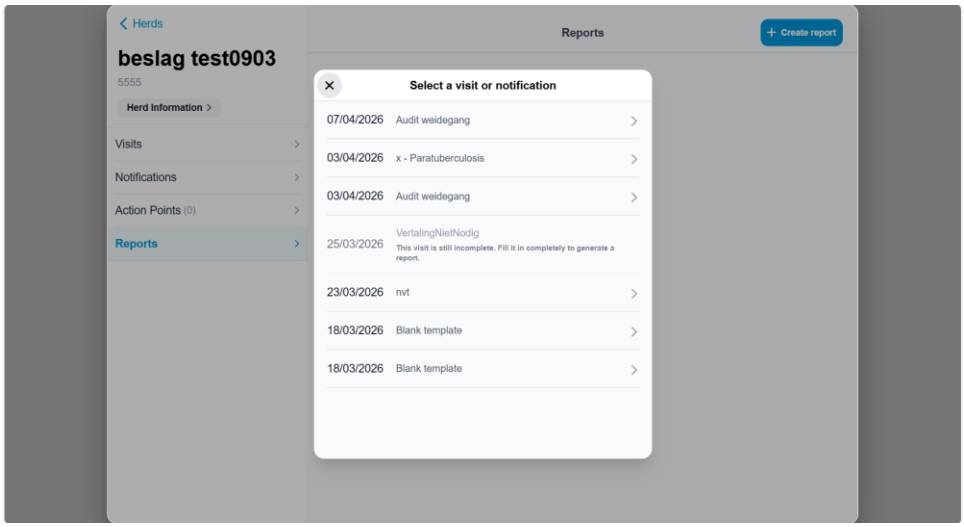


Figuur 31. Stappen binnen een melding: boven de parameters-stap met dezelfde dynamische logica als bij bezoeken, onder de foto's-stap voor het uploaden van bijhorende afbeeldingen.

3.12 Rapporten

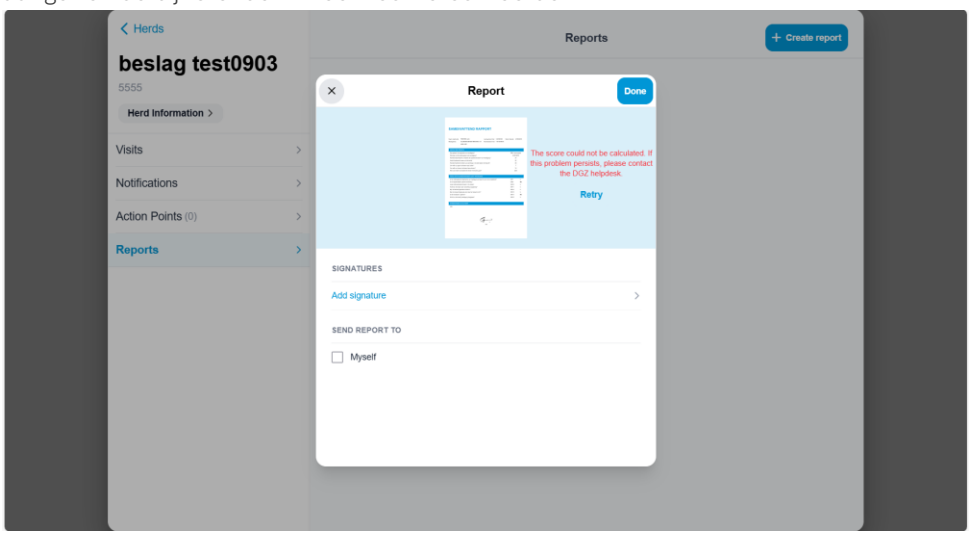
3.12.1 Genereren:

Zoals eerder vermeld bij bezoeken, biedt de applicatie de mogelijkheid om rapporten aan te maken. De gebruiker kiest eerst tussen de verschillende bezoeken en meldingen die eerder werden geregistreerd. Indien een bezoek of melding niet alle benodigde data bevat, wordt dit visueel aangeduid en wordt de keuze gedeactiveerd, zodat er geen onvolledig rapport kan worden gegenereerd.



3.12.2 Score berekening:

Bepaalde bezoeken of meldingen bevatten voldoende gegevens om een rapport met score te berekenen. Deze score wordt berekend na het selecteren van een bezoek of melding en wordt zowel naast als binnen de preview weergegeven. In de gebruikte ontwikkelomgeving kon deze functionaliteit niet worden gevalideerd, aangezien de bijhorende API een fout retourneerde.



Figuur 32. Genereren van een rapport: links de selectie van het te rapporteren bezoek of melding, rechts de preview met de melding dat de score niet kon worden berekend door een fout in de API van de ontwikkelomgeving.

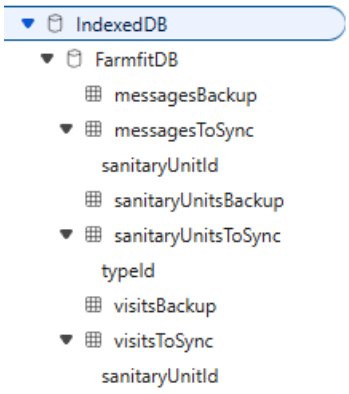
3.13 Offline + PWA

Doorheen dit document is meermaals aangehaald dat de applicatie offline kan werken. De reden hiervoor is functioneel: FarmFit wordt voornamelijk gebruikt op boerderijen, die vaak afgelegen liggen en waar geen stabiele toegang tot wifi of mobiele data beschikbaar is. Daarom is het belangrijk dat een gebruiker de applicatie nog op een verbonden locatie kan opstarten, de nodige gegevens kan ophalen, en vervolgens het volledige bezoek ter plaatse kan afwerken zonder internetverbinding.

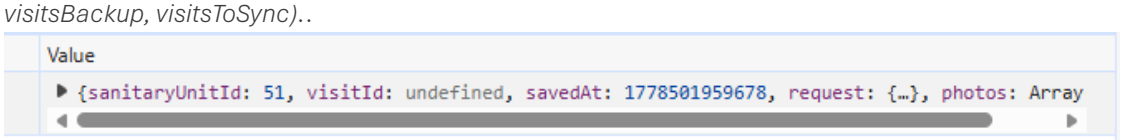
Om dit mogelijk te maken, werd de applicatie opgezet als Progressive Web App met behulp van service workers (Angular Team, z.d.-c; Mozilla Developer Network, z.d.-a). De lokale opslag van backups werd gerealiseerd via de IndexedDB-API (Mozilla Developer Network, z.d.-b). De twee functionaliteiten die hierbij centraal staan, zijn het aanmaken van backups en het synchroniseren ervan.

Backups. Wanneer een gebruiker een bezoek, melding of beslag invult, wordt de ingevoerde data automatisch lokaal opgeslagen op het apparaat van de gebruiker. Op die manier gaan gegevens niet verloren wanneer de gebruiker de applicatie verlaat en later terugkeert. Op het moment dat de applicatie onverwacht wordt afgesloten of dat er ongewenst naar een andere pagina wordt genavigeerd, maakt de applicatie een backup aan. Wanneer de gebruiker vervolgens opnieuw start met hetzelfde type bezoek, melding of beslag, wordt eerst gevraagd of er met een eerder opgeslagen backup verder gewerkt moet worden.

Synchronisatie. De offlinemodus maakt het mogelijk om een bezoek, melding of beslag aan te maken zonder internetverbinding. Een concreet voorbeeld: een gebruiker registreert een beslag op het moment dat er geen internet beschikbaar is. Het beslag wordt opgeslagen en in een opslagruimte met de naam *Sync* geplaatst. Zodra de gebruiker opnieuw online komt, detecteert de applicatie dit automatisch en wordt het opgeslagen item gesynchroniseerd met de backend. Op die manier kan de volledige flow van de applicatie ook offline worden doorlopen.



Figuur 33. IndexedDB-structuur in de browser, met afzonderlijke object stores voor backups en synchronisatiewachtrijen (messagesBackup, messagesToSync, sanitaryUnitsBackup, sanitaryUnitsToSync, visitsBackup, visitsToSync)..



Figuur 34. Voorbeeld van een offline opgeslagen visit-object, klaar om te worden gesynchroniseerd zodra de internetverbinding hersteld is.

3.14 Besluit

Dit hoofdstuk blikt terug op de ontwikkeling van de vernieuwde FarmFit-applicatie, met bijzondere aandacht voor maintainability, performance en gebruiksvriendelijkheid. Achtereenvolgens komen de belangrijkste resultaten, de opgedane leerervaringen en de aansluiting bij de doelstellingen uit het projectplan aan bod.

3.14.1 Terugblik en belangrijkste resultaten

Het FarmFit-project leverde een diepgaand inzicht op in moderne front-end frameworks, met name Angular, en in de mogelijkheden die deze bieden op het vlak van componentstructuur, herbruikbaarheid en performance. De vergelijking tussen Angular en Vue droeg bij tot een beter begrip van welke frameworks binnen Cegeka het meest geschikt zijn voor specifieke toepassingen, op basis van criteria zoals ontwikkelsnelheid, onderhoudbaarheid en leercurve.

Daarnaast werd kennis opgebouwd rond het opzetten en configureren van een Progressive Web App (PWA), zodat gebruikers de applicatie ook offline kunnen gebruiken. Het integreren van PWA-functionaliteit in een bestaande applicatieomgeving, inclusief automatische synchronisatie van data, vormde een uitdagende maar waardevolle realisatie.

Het project werd professioneel opgezet, met een duidelijke structuur in de repository en het gebruik van Azure DevOps (Microsoft, z.d.-d) voor versiebeheer en releases. Hierdoor werd een solide basis gelegd voor toekomstige ontwikkeling en onderhoud. De ontwikkelde applicatie is volledig werkend en kan binnen de repository worden getest en uitgebreid. Bovendien werd een duidelijke en toegankelijke README toegevoegd, zodat andere ontwikkelaars snel inzicht krijgen in de opzet en werking van het project.

3.14.2 Reflectie op leerervaringen

Het grootste leermoment binnen dit project lag in het analyseren van complexe functionaliteiten, zoals dynamische observaties en parameterlogica bij bezoeken. Door gebruik te maken van agentic development en het stapsgewijs opbouwen van functionaliteiten, werd inzicht verworven in het structureren van complexe processen en het implementeren van efficiënte oplossingen.

Daarnaast bood het werken met verschillende technologieën — zoals Angular, PrimeNG, Tailwind CSS, MSAL Angular en Microsoft Entra ID — een waardevol overzicht van moderne tooling binnen front-end ontwikkeling en enterprise-applicaties. Deze inzichten vormen een belangrijke basis voor toekomstige projecten waarin maintainability, performance en gebruikerservaring centraal staan.

3.14.3 Conclusie

Het FarmFit-project leidde niet alleen tot een werkende, moderne en onderhoudbare applicatie, maar leverde ook een belangrijke bijdrage aan de professionele ontwikkeling die tijdens deze stage werd doorlopen. Alle doelstellingen uit het projectplan — van het verwerven van inzicht in frameworks tot het opzetten van een werkend project en een duidelijke documentatie — werden behaald. De opgedane kennis en ervaring vormen een solide basis voor toekomstige ontwikkeling binnen Cegeka of andere projecten in front-end engineering.

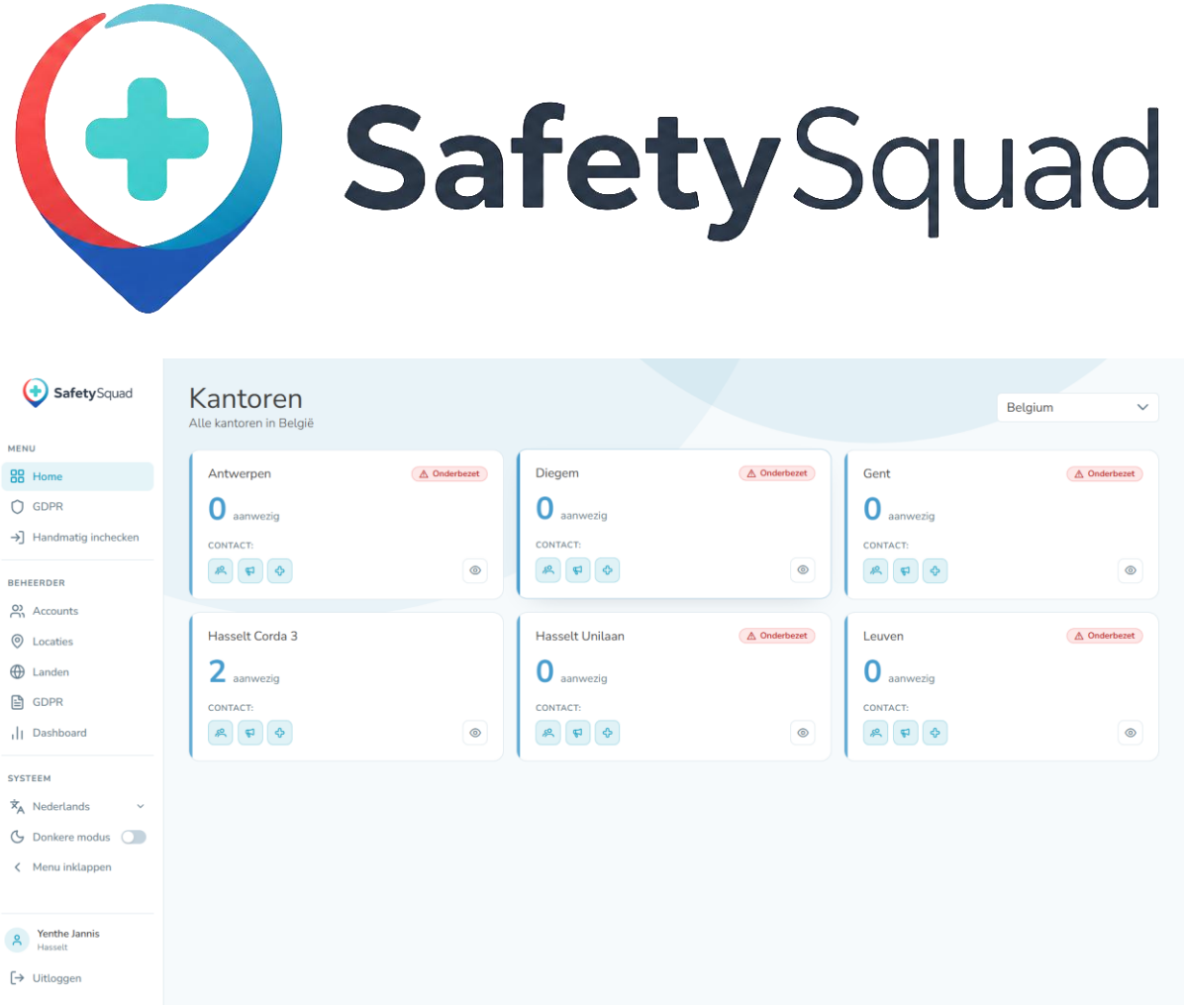
4 Sneller dan verwacht — drie bijkomende projecten

Wanneer een stage vlotter verloopt dan verwacht, ontstaat er ruimte voor iets wat zelden voorkomt: de kans om meer te doen dan afgesproken. FarmFit was de centrale opdracht van deze stage, een project waarvan zowel Cegeka als ikzelf verwachtten dat het de volledige dertien weken zou vullen. Dat het al in week acht werd opgeleverd, was een verrassing en tegelijk het startpunt van een tweede, onverwachte fase in de stage.

In de resterende weken werd bijgedragen aan drie heel verschillende projecten binnen Cegeka. SafetySquad was een bestaande applicatie die door jaren slordig was geworden en nood had aan een grondige refactor en een nieuw visueel jasje. RaamContracten was een volledig nieuw project, maar met een bijzondere twist: de opdracht was uitdrukkelijk om de applicatie volledig via AI-driven development op te bouwen, zonder zelf een regel code te schrijven. OWASP ten slotte was het kleinste project, maar vroeg een andere manier van werken: niet bouwen, maar verfijnen. Een onbekende codebase leren kennen en gericht aanpassingen doorvoeren zonder de bestaande werking te verstoren.

Elk van deze drie projecten had zijn eigen technische context, zijn eigen uitdagingen en zijn eigen leermomenten. Samen zorgden ze voor een stage die uiteindelijk veel meer omvatte dan oorspronkelijk gepland en die des te waardevoller was omdat ze dat deed.

4.1 SafetySquad



Figuur 35. Logo en dashboard van SafetySquad, een interne Cegeka-applicatie voor het oproepen van EHBO'ers in noodgevallen.

4.1.1 Inleiding

SafetySquad is een applicatie waarmee werknemers in geval van nood een EHBO'er kunnen oproepen via een desktop-app, mobiele app of website, waarna nabije EHBO'ers een melding ontvangen en kunnen reageren. De applicatie wordt momenteel intern gebruikt binnen Cegeka, met als doel om deze in de toekomst ook publiek aan te bieden aan andere bedrijven.

SafetySquad is een project dat reeds meerdere jaren in ontwikkeling is door verschillende stagiairs. Vlak voor de start van deze stage was de applicatie door twee andere stagiairs gemigreerd van Angular v17 naar v21, maar de code zelf was niet mee geüpdatet naar de nieuwe manier van werken. Doordat het project doorheen de jaren steeds aan andere stagiairs werd doorgegeven, was de codebase slordig geworden: een onoverzichtelijke mapstructuur, verouderde syntax zoals *ngIf in plaats van de nieuwe control flow, NgModules in plaats van standalone components, geen gebruik van signals en een algemene inconsistentie in de code.

De opdracht bestond uit drie onderdelen. Eerst werd de volledige frontend gerefactord en herbouwd volgens de werkwijze van Angular v21, waarbij Bootstrap werd vervangen door Tailwind CSS in combinatie met PrimeNG en PrimeIcons. Vervolgens werd een redesign van de frontend uitgewerkt om de gebruiksvriendelijkheid en het visuele ontwerp te verbeteren. Tot slot werd in samenwerking met een medestagiair een admin dashboard ontwikkeld, waarbij de frontend door deze stage werd verzorgd en de backend door de medestagiair. Dit dashboard stelt admins in staat om de EHBO-bezetting per kantoor en per land op te volgen, onderbezetting tijdig te detecteren en via een kalenderoverzicht te zien of er voldoende EHBO-dekking is per dag en locatie.

In dit hoofdstuk komen achtereenvolgens de opzet van de nieuwe frontend, de refactor, het redesign en het admin dashboard aan bod. Waar relevant worden per onderdeel kort de uitdagingen toegelicht die zich voordeden.

4.1.2 Refactor

Het eerste onderdeel van het werk aan SafetySquad bestond uit een volledige refactor van de bestaande frontend. De oorspronkelijke applicatie was opgebouwd in Angular 17, maakte gebruik van Bootstrap voor de styling en Material Icons voor de iconografie. Daarnaast werd er gewerkt met de oudere control flow-syntax zoals *nglf en *ngFor, en was de algemene mapstructuur onoverzichtelijk geworden door de vele stagiairs die doorheen de jaren aan het project hadden gewerkt.

Het doel van de refactor was om de applicatie volledig om te zetten naar Angular 21 en op te bouwen volgens de moderne werkwijze van het framework. Dit betekende een signal-gebaseerde aanpak, een duidelijke en schaalbare mapstructuur, en productieklare code. Tegelijk werd de technologiestack vernieuwd: Bootstrap werd vervangen door Tailwind CSS in combinatie met PrimeNG, en Material Icons werden vervangen door PrimeIcons en Lucide Icons. Verder werden ngx-translate en MSAL Angular toegevoegd voor respectievelijk meertaligheid en Microsoft-authenticatie.

De refactor werd in twee fasen aangepakt. In de eerste fase werd het project opgezet en werden de nodige libraries geïnstalleerd en geconfigureerd, zodat de basis klaarstond om verder op te bouwen. Hierbij werd de applicatie voorbereid om vlot gebruik te kunnen maken van alle gekozen pakketten, en werd de mapstructuur uitgewerkt volgens een feature-gebaseerde opbouw.

In de tweede fase werd pagina per pagina en component per component te werk gegaan. Elk onderdeel uit de oude applicatie werd overgebracht naar de nieuwe frontend, waarbij de volledige code werd herschreven volgens de moderne Angular-werkwijze. Concreet betekende dit het omzetten van *nglf en *ngFor naar de nieuwe @if- en @for-controlflow, het migreren van NgModules naar standalone components, en het invoeren van signals voor state management. Op die manier kreeg de applicatie niet alleen een technische upgrade, maar werd de codebase ook opnieuw onderhoudbaar en consistent.

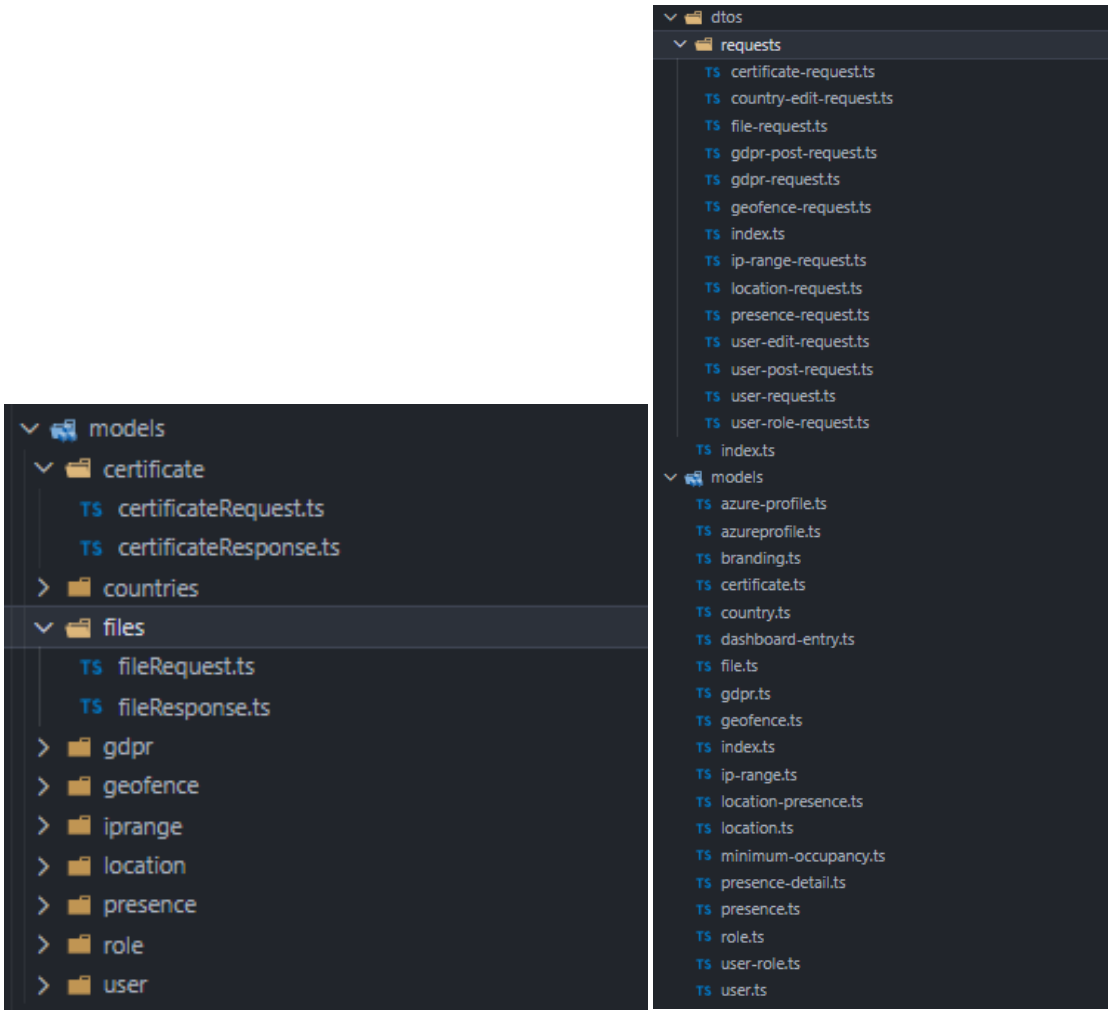
Figuur 36. Mapstructuur van SafetySquad vóór de refactor — inconsistente folder-indeling met onder andere location-card (rood gemarkeerd) als component op pages-niveau.

Figuur 37. Mapstructuur van SafetySquad na de refactor — consistente folder-indeling per feature, duidelijke benamingen en de scheiding van models en DTO's.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { NgxTranslateModule } from './translate/translate.module';
import { BsDropdownModule } from 'ngx-bootstrap/dropdown';
import { NgbModule, NgbPaginationModule } from '@ng-bootstrap/ng-bootstrap';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomepageComponent } from '@pages/homepage/homepage.component';
import { DetailpageComponent } from '@pages/detailpage/detailpage.component';
import { HTTP_INTERCEPTORS, provideHttpClient, withInterceptorsFromDi } from '@angular/common/http';
import { RegisterComponent } from '@pages/register/register.component';
import { NavbarComponent } from './navbar/navbar.component';
import { LocationCardComponent } from '@pages/location-card/location-card.component';
import { LocationManagementComponent } from '@pages/location-management/location-management.component';
import { LocationDetailComponent } from '@pages/location-detail/location-detail.component';
import { MsalGuard, MsalInterceptor, MsalBroadcastService, MsalInterceptorConfiguration, MsalModule, MsalService, MSAL
import { MatTabsModule } from '@angular/material/tabs';
import { MatChipsModule } from '@angular/material/chips';
import { IPublicClientApplication, PublicClientApplication, InteractionType, BrowserCacheLocation, LogLevel } from '@a
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { NoPermissionComponent } from '@pages/no-permission/no-permission.component';
import { GdprEditComponent } from '@pages/gdpr-edit/gdprEdit.component';
import { GdprpageComponent } from '@pages/gdpr/gdprPage.component';
import { CountriesManagementComponent } from '@pages/country-management/countries-management.component';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { MatDialogModule } from '@angular/material/dialog';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatOptionModule } from '@angular/material/core';
import {  MatSelectModule } from '@angular/material/select';
import { MobilePresenceDetailComponent } from '@pages/mobile-presence-detail/mobile-presence-detail.component';
import { SelectComponent } from "@components/select/select.component";
import { CountryDetailComponent } from '@pages/country-detail/country-detail.component';
import { environment } from '@config/environment';
import { FormsModule } from '@angular/forms';
import { AccountsComponent } from '@pages/account-management/accounts-management.component';
import { AccountDetailComponent } from '@pages/account-detail/account-detail.component';


const isIE = window.navigator.userAgent.indexOf("MSIE ") > -1 || window.navigator.userAgent.indexOf("Trident/") > -1;

export function loggerCallback(logLevel: LogLevel, message: string) {

}

export function MSALInstanceFactory(): IPublicClientApplication {
  return new PublicClientApplication({
    auth: {
      clientId: 'fb8debf5-a37a-4919-ab42-1360fcee9fc3',
      authority: 'https://login.microsoftonline.com/organizations',
      redirectUri: '/',
      postLogoutRedirectUri: '/'
    },
    cache: {
      cacheLocation: BrowserCacheLocation.LocalStorage,
      storeAuthStateInCookie: isIE,
    },
    system: {
      allowNativeBroker: false,
      loggerOptions: {
        loggerCallback,
        logLevel: LogLevel.Info,
```

```
@Component({
  selector: 'app-location-detail',
  standalone: true,
  imports: [FormsModule, TranslatePipe, Skeleton, SelectModule, BreadcrumbComponent, ToastModule],
  providers: [MessageService],
  templateUrl: './location-detail.component.html',
  styleUrls: ['./location-detail.component.css'],
})
export class LocationDetailComponent {
```

Figuur 38. Vergelijking van de Angular-codestijl voor en na de refactor: boven de verouderde aanpak met een centrale app.module.ts zonder standalone components, onder de moderne aanpak met een standalone LocationDetailComponent.



Figuur 39. Scheiding van DTO's en domeinmodellen na de refactor, conform de Angular-aanbevelingen.

4.1.3 Redesign

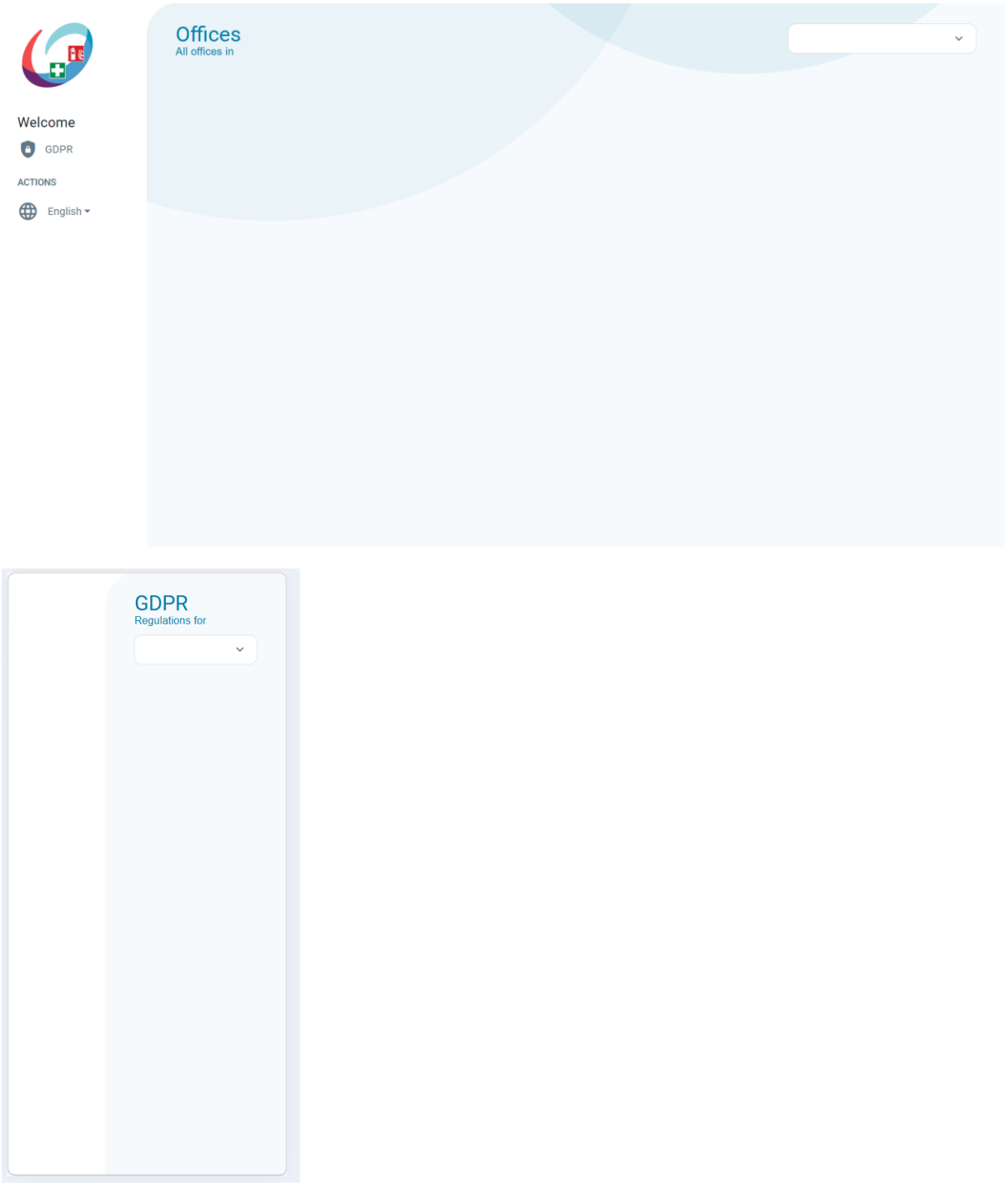
Naast de technische refactor werd ook een redesign van de applicatie uitgewerkt om de gebruikerservaring en het visuele ontwerp naar een hoger niveau te tillen. De focus lag hierbij op consistentie, duidelijkheid en het klaarmaken van de applicatie voor een breder publiek, aangezien Cegeka Safety Squad in de toekomst ook aan andere bedrijven wil aanbieden.

Een eerste belangrijke aanpassing was de navigatiebalk. Deze werd herwerkt zodat ze overzichtelijker oogt en beter aansluit bij de moderne uitstraling van de rest van de applicatie. Daarnaast werd een nieuw logo ontwikkeld dat de identiteit van SafetySquad sterker neerzet.

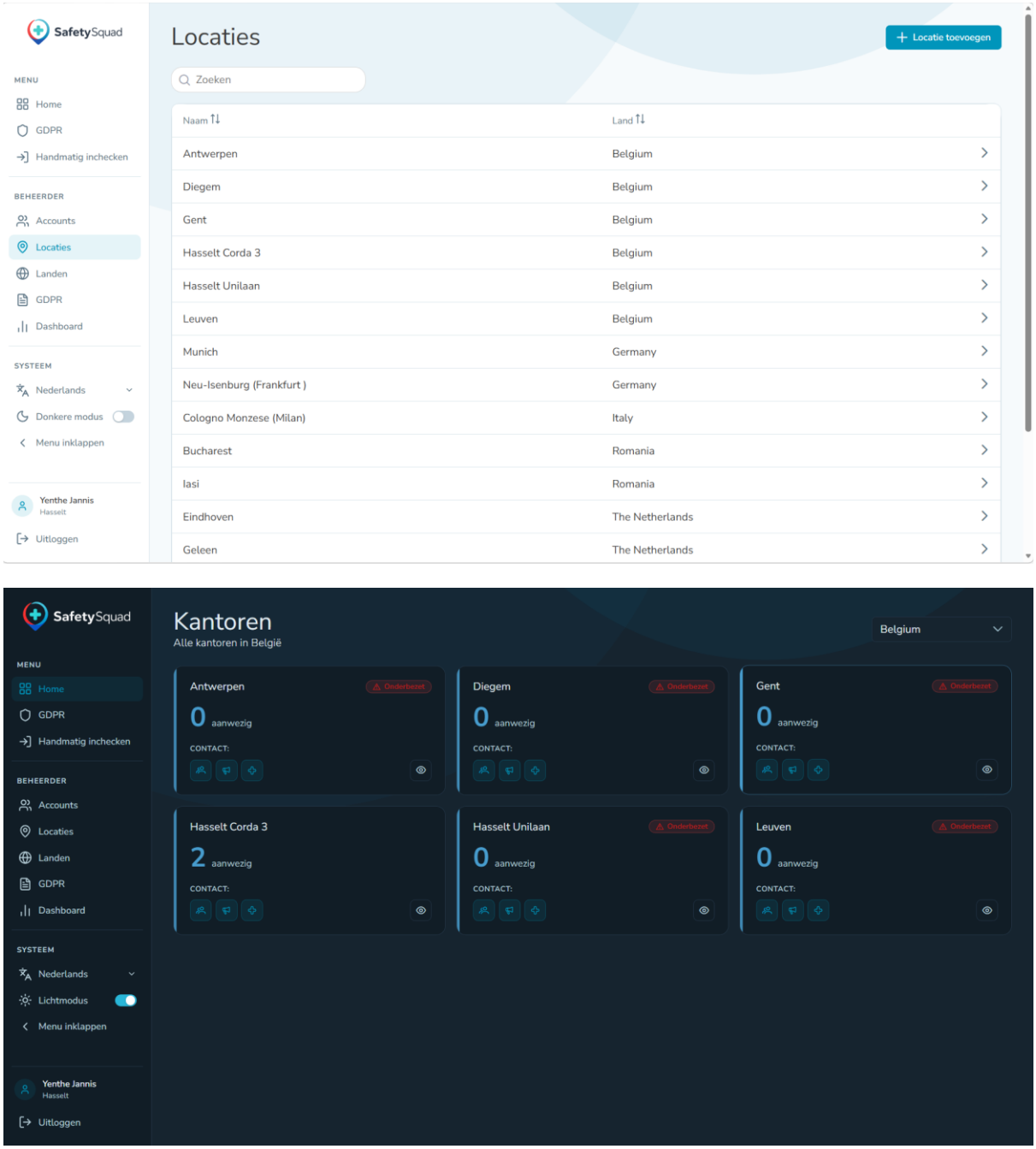
Op het vlak van UX werden verschillende verbeteringen doorgevoerd. Knoppen kregen consistente kleuren op basis van hun functie (primary voor hoofdacties, secondary voor ondersteunende acties), zodat gebruikers in één oogopslag zien welke actie centraal staat. Verder werden toast messages toegevoegd voor duidelijke feedback bij acties, en loading states om de gebruiker te informeren wanneer data wordt opgehaald of verwerkt. Tot slot werden er ook een dark en light mode geïmplementeerd, zodat gebruikers de applicatie naar eigen voorkeur kunnen instellen.

Als laatste onderdeel van het redesign werd een brandingpagina toegevoegd. Deze pagina kan in de toekomst gebruikt worden om de branding van de applicatie aan te passen per organisatie, zoals het wijzigen van het logo en de kleuren. Op die manier kan SafetySquad eenvoudig worden uitgerold bij andere bedrijven met behoud van hun eigen huisstijl.

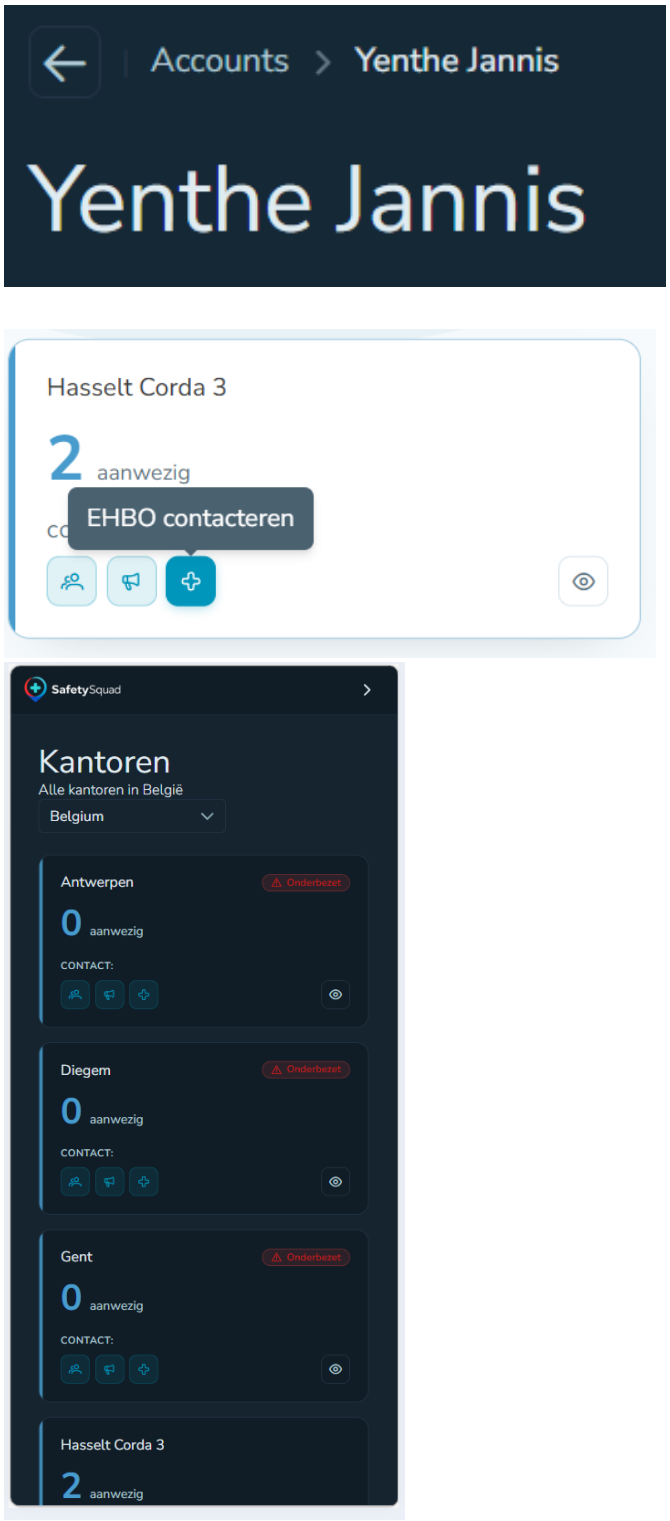
Voorheen — een eenvoudige interface zonder professionele enterprise-uitstraling en zonder mobile-friendly opmaak.



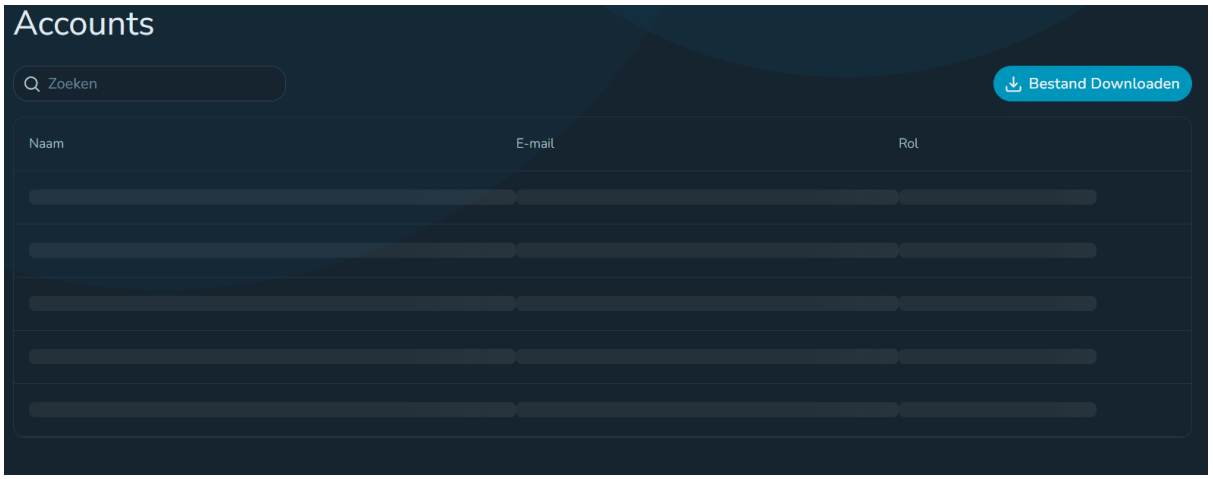
Figuur 40. *Oude versie van SafetySquad — eenvoudige interface zonder professionele uitstraling en zonder mobile-friendly opmaak.*



Figuur 41. Vernieuwde SafetySquad-interface: light mode met tabelweergave en filteropties (boven), dark mode met kaartweergave per locatie (onder).



Figuur 42. Verbeteringen op het vlak van UX: breadcrumb-navigatie op detailpagina's (boven), tooltips bij actieknoppen zoals "EHBO contacteren" (midden) en een mobile-responsive weergave (onder).



Figuur 43. *Loading states binnen SafetySquad — skeleton-componenten die de gebruiker informeren wanneer data wordt opgehaald.*

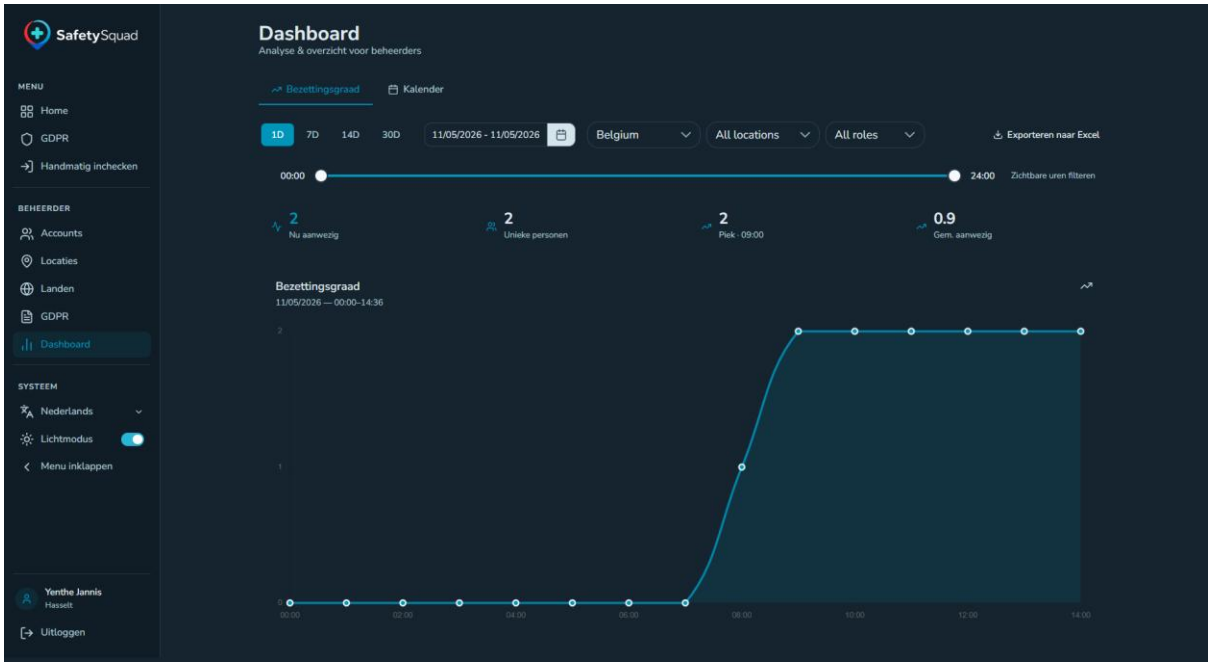
4.1.4 Admin dashboard | bezetting

Het admin dashboard was een volledig nieuwe functionaliteit binnen SafetySquad, die in samenwerking met een medestagiair werd ontwikkeld. De medestagiair stond in voor de backend, terwijl de frontend door deze stage werd uitgewerkt. Het doel van dit dashboard is om admins een helder overzicht te geven van de EHBO-bezetting, zodat zij tijdig kunnen ingrijpen wanneer er onderbezetting dreigt.

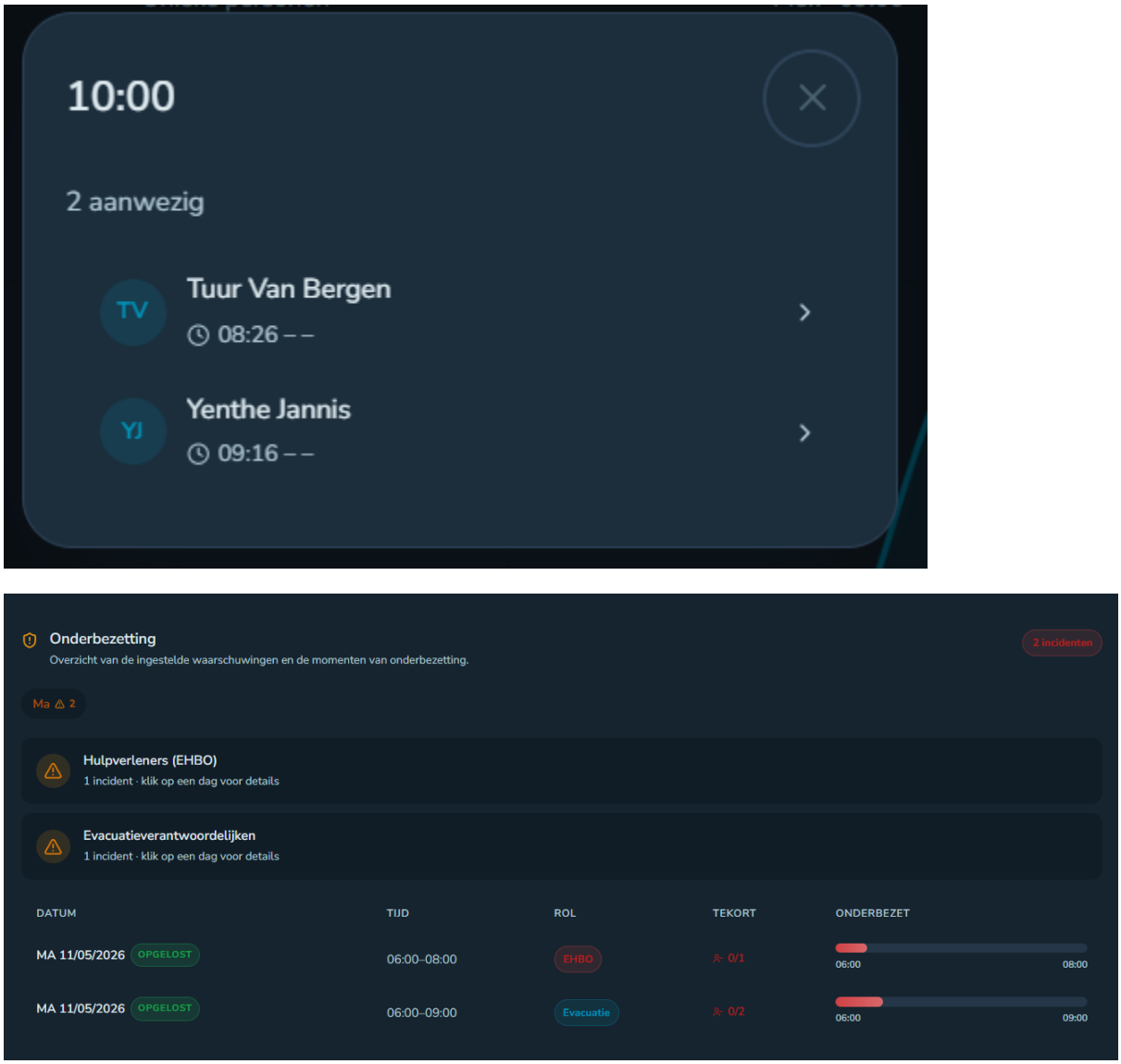
De kern van het dashboard is het in kaart brengen van de bezettingsgraad. Admins kunnen filteren op land en locatie, en krijgen zowel een realtime beeld als een evolutie over de dagen heen van het aantal aanwezige EHBO'ers. Wanneer een specifieke locatie wordt geselecteerd, verschijnt onderaan een overzicht van de onderbezetting. Vanuit dit overzicht kunnen admins meteen acties ondernemen om de tekorten op te vangen.

Daarnaast werd een kalenderweergave toegevoegd, waarmee admins per dag kunnen terugkijken naar wat er die dag heeft plaatsgevonden en hoe de bezetting eruitzag. Op die manier kunnen ze patronen herkennen en beter inschatten op welke dagen extra aandacht nodig is.

Bij de uitwerking van dit dashboard stond duidelijkheid steeds centraal. Admins moeten in één oogopslag kunnen zien waar de aandachtspunten liggen, zonder te verdrinken in data. Door middel van een doordachte indeling, filters en visuele accenten werd geprobeerd om deze duidelijkheid maximaal te waarborgen.



Figuur 44. Admin dashboard van SafetySquad — grafiek met de evolutie van de EHBO-bezettingsgraad doorheen de dag, met filters op land, locatie en rol.



Figuur 45. Detailweergave van de bezetting op een specifiek moment (boven) en het overzicht van geregistreerde momenten van onderbezetting per locatie en rol (onder).

Figuur 46. Kalenderweergave van het admin dashboard — basisweergave per maand (boven) en uitgebreide weergave met dagdetails, bezettingsgraad-grafiek en aanwezigheidslijst (onder).

4.1.5 Besluit

SafetySquad was een veelzijdig project dat de mogelijkheid bood om op meerdere vlakken bij te dragen aan de applicatie. Met de refactor werd de codebase opnieuw onderhoudbaar en toekomstbestendig gemaakt door de overstap naar Angular 21 en de moderne werkwijze van het framework. Met het redesign werden de gebruikerservaring en de visuele uitstraling versterkt, en werd de applicatie klaargemaakt om in de toekomst aan andere organisaties aangeboden te worden. Tot slot werd met het admin dashboard een nieuwe functionaliteit toegevoegd die admins de nodige tools biedt om de EHBO-bezetting actief op te volgen en bij te sturen.

Het werken aan SafetySquad leverde waardevolle inzichten op, zowel op technisch vlak als op het vlak van samenwerking. Het opnieuw opbouwen van een bestaande codebase toonde aan hoe belangrijk een consistente structuur en duidelijke conventies zijn, zeker in een project waaraan meerdere ontwikkelaars opeenvolgend werken. Daarnaast maakte de samenwerking met de medestagiair voor het admin dashboard duidelijk hoe waardevol een goede afstemming tussen frontend en backend is om tot een vlot werkend geheel te komen.

4.2 Raam Contracten

4.2.1 Inleiding

RaamContracten is een volledig nieuwe applicatie die binnen Cegeka vanaf nul werd opgebouwd. Een raamcontract is een overkoepelende overeenkomst tussen Cegeka en een klant of leverancier, waarbinnen meerdere opdrachten of leveringen kunnen worden uitgevoerd zonder dat er telkens een nieuw contract moet worden opgesteld. Het beheren van deze contracten is binnen Cegeka een terugkerende taak die op dit moment niet efficiënt verloopt.

Vandaag worden raamcontracten beheerd via een veelheid aan Excel-bestanden. Dit brengt verschillende problemen met zich mee: medewerkers kunnen niet tegelijk aan hetzelfde raamcontract werken, er is geen centrale plaats waar alle raamcontracten zijn terug te vinden, en het overzicht raakt snel zoek. Hierdoor ontstond de vraag naar een centrale applicatie die het beheer van raamcontracten eenvoudiger, sneller en betrouwbaarder maakt. De applicatie is in eerste instantie bedoeld voor intern gebruik binnen Cegeka, maar ook hier wordt gekeken naar de mogelijkheid om deze in de toekomst publiek aan te bieden aan andere bedrijven.

Functioneel moet de applicatie toelaten om raamcontracten aan te maken, te bewerken, te archiveren en te exporteren naar Excel, waarbij zowel een light- als een full-export mogelijk is. Voor de technische uitwerking werd gekozen voor Angular aan de frontendzijde en .NET aan de backendzijde. Het project werd volledig solo uitgewerkt en bevond zich op het einde van de stage in de MVP-fase, waarbij nog volop aan iteraties werd gewerkt om de werking verder te verfijnen.

Wat dit project extra bijzonder maakt, is de manier waarop het tot stand is gekomen. De opdracht had namelijk een dubbel doel: naast het opleveren van een werkende applicatie wilde Cegeka ook in kaart brengen hoe snel iemand met ervaring in AI-driven development een volledig project kan opzetten zonder zelf een regel code te schrijven. De opdracht was dus uitdrukkelijk om de applicatie volledig met behulp van AI te ontwikkelen. Het vertrekpunt hierbij was een analysedocument dat eerder werd opgesteld, maar dat tijdens de uitwerking inaccuraat en onvolledig bleek te zijn, wat de aanpak extra interessant maakte.

In dit hoofdstuk komen achtereenvolgens de opzet van het project en de verschillende uitgewerkte onderdelen aan bod. Waar relevant worden per onderdeel kort de uitdagingen toegelicht die zich voordeden.

4.2.2 Opzet

Aangezien dit project volledig met behulp van AI moest worden ontwikkeld, was de opzet van cruciaal belang. De kwaliteit van wat AI oplevert, hangt namelijk sterk af van de context en richtlijnen die worden meegegeven. Voor dit project werd gewerkt met GitHub Copilot (GitHub, z.d.), dat binnen Cegeka beschikbaar is als interne tooling.

Als vertrekpunt werd een eerder opgesteld analysedocument aangereikt. Tijdens een eerste doorlezing bleek echter al snel dat dit document op verschillende vlakken inaccuraat en onvolledig was. Daarom werd eerst tijd geïnvesteerd in het verfijnen van de analyse, zodat er een duidelijk beeld ontstond van wat de applicatie precies moest doen.

Vervolgens werden in samenwerking met de AI alle functionaliteiten uitgeschreven in readme-bestanden, die als context dienden tijdens het verdere ontwikkelproces. Concreet werd gewerkt met een hoofd-README, MVP genaamd, waarin alle functionaliteiten apart werden uitgewerkt. Daarnaast werd een tweede README opgesteld waarin het volledige analysedocument was opgenomen, zodat de AI te allen tijde kon teruggrijpen naar de oorspronkelijke context van het project. Op die manier kon functionaliteit per functionaliteit worden overlopen en kon de AI gericht worden aangestuurd.

Voor de backend werd gebruikgemaakt van een bestaand referentieproject binnen Cegeka, een gelijkaardige tool waarvan de architectuur als kwalitatief werd beschouwd. De AI kreeg de opdracht om dit project volledig te analyseren, zodat de nieuwe backend volgens dezelfde structuur, patronen en conventies kon worden opgebouwd. Op die manier sluit RaamContracten aan bij de standaarden die binnen Cegeka gehanteerd worden.

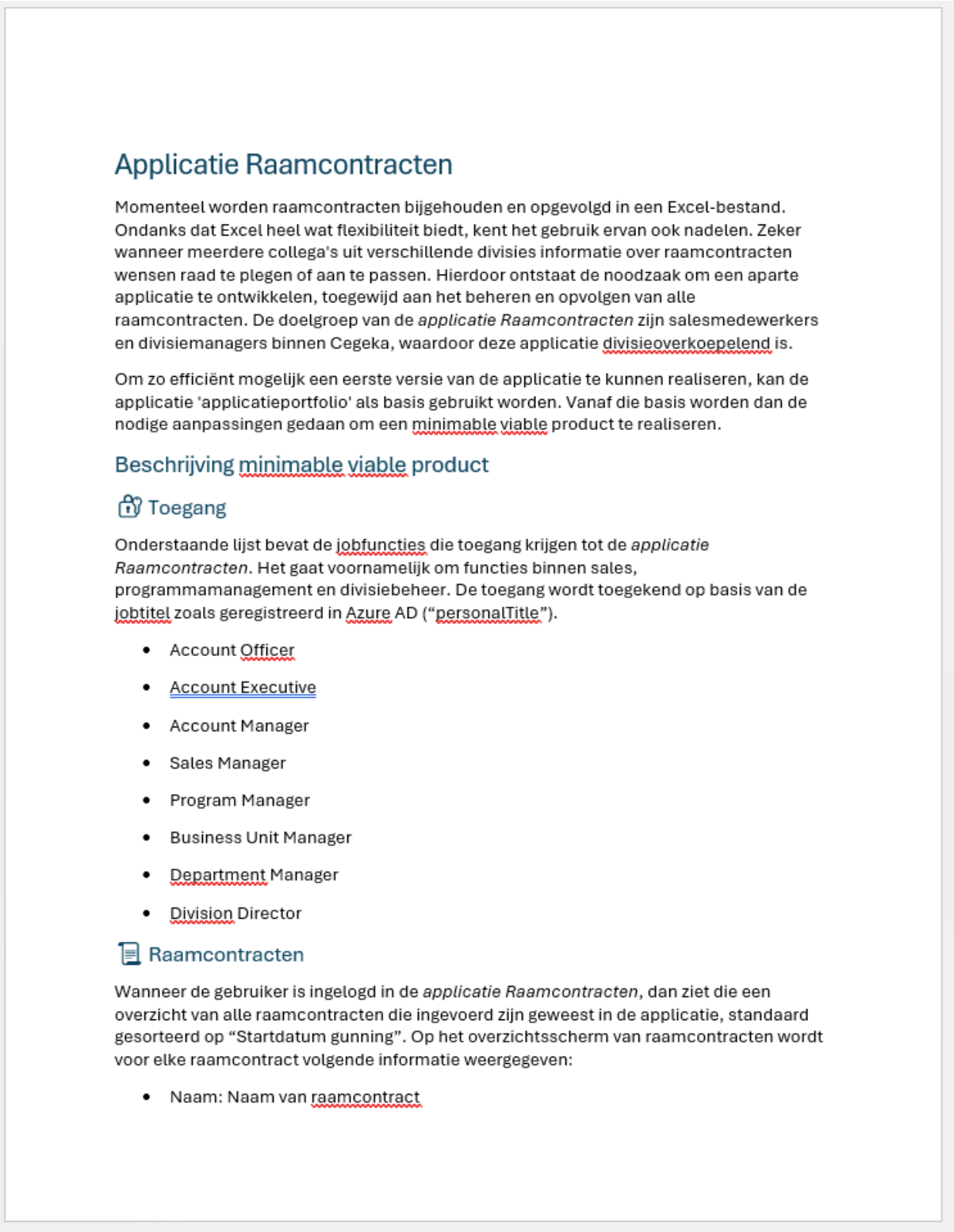
Voor de frontend werden de officiële Angular coding guidelines gevolgd (Angular Team, z.d.-b), die beschikbaar worden gesteld bij de opzet van een nieuw project met ng new. Door deze richtlijnen aan de AI mee te geven, kon worden gegarandeerd dat de Angular-code volgens de aanbevolen werkwijze werd opgebouwd.

4.2.3 MVP

Na de opzet werd gestart met het uitwerken van de MVP. Deze omvat de kernfunctionaliteiten die nodig zijn om raamcontracten op een vlotte manier te beheren binnen de applicatie.

Gebruikers kunnen via de applicatie raamcontracten aanmaken, bekijken, bewerken en archiveren. Daarnaast is er een exportfunctionaliteit voorzien, met twee varianten. De light-export bevat enkel een beperkt aantal velden en is handig voor een snel overzicht, terwijl de full-export het volledige raamcontract met alle details exporteert naar Excel. Tot slot werden audit logs geïmplementeerd, waarin alle aanpassingen door gebruikers worden bijgehouden, zodat steeds traceerbaar is wie wat heeft aangepast en wanneer.

De ontwikkeling van de MVP verliep vlot dankzij de doordachte opzet en de duidelijke readmes. De voorbereidende stappen wierpen hier duidelijk hun vruchten af, waardoor de AI gericht en consistent aan de verschillende functionaliteiten kon werken.



Applicatie Raamcontracten

Momenteel worden raamcontracten bijgehouden en opgevolgd in een Excel-bestand. Ondanks dat Excel heel wat flexibiliteit biedt, kent het gebruik ervan ook nadelen. Zeker wanneer meerdere collega's uit verschillende divisies informatie over raamcontracten wensen raad te plegen of aan te passen. Hierdoor ontstaat de noodzaak om een aparte applicatie te ontwikkelen, toegewijd aan het beheren en opvolgen van alle raamcontracten. De doelgroep van de *applicatie Raamcontracten* zijn salesmedewerkers en divisiemanagers binnen Cegeka, waardoor deze applicatie divisieoverkoepelend is.

Om zo efficiënt mogelijk een eerste versie van de applicatie te kunnen realiseren, kan de applicatie 'applicatieportfolio' als basis gebruikt worden. Vanaf die basis worden dan de nodige aanpassingen gedaan om een minimable viable product te realiseren.

Beschrijving minimable viable product

Toegang

Onderstaande lijst bevat de jobfuncties die toegang krijgen tot de *applicatie Raamcontracten*. Het gaat voornamelijk om functies binnen sales, programmamanagement en divisiebeheer. De toegang wordt toegekend op basis van de jobtitel zoals geregistreerd in Azure AD (“personalTitle”).

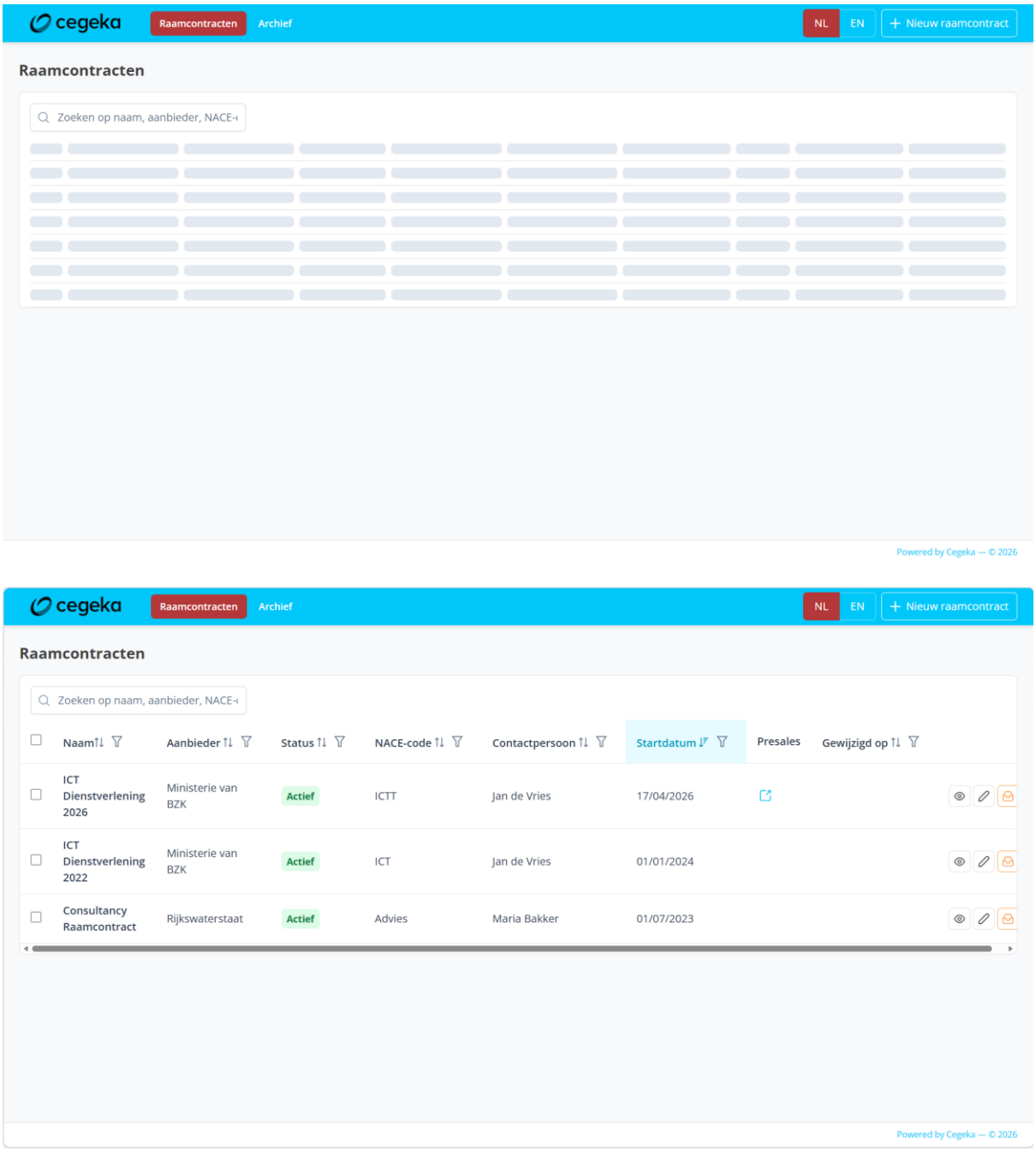
- Account Officer
- Account Executive
- Account Manager
- Sales Manager
- Program Manager
- Business Unit Manager
- Department Manager
- Division Director

Raamcontracten

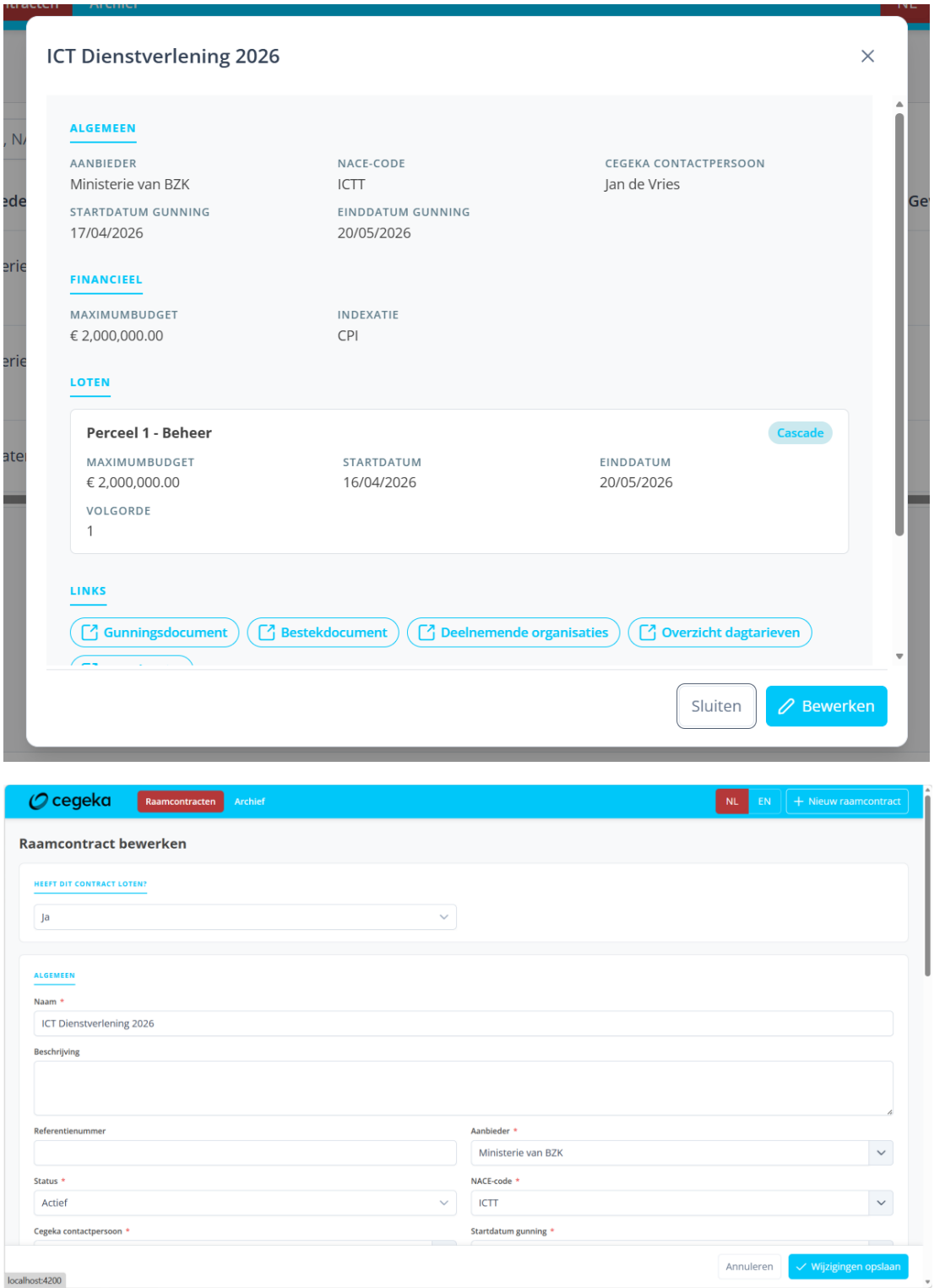
Wanneer de gebruiker is ingelogd in de *applicatie Raamcontracten*, dan ziet die een overzicht van alle raamcontracten die ingevoerd zijn geweest in de applicatie, standaard gesorteerd op “Startdatum gunning”. Op het overzichtsscherm van raamcontracten wordt voor elke raamcontract volgende informatie weergegeven:

- Naam: Naam van raamcontract

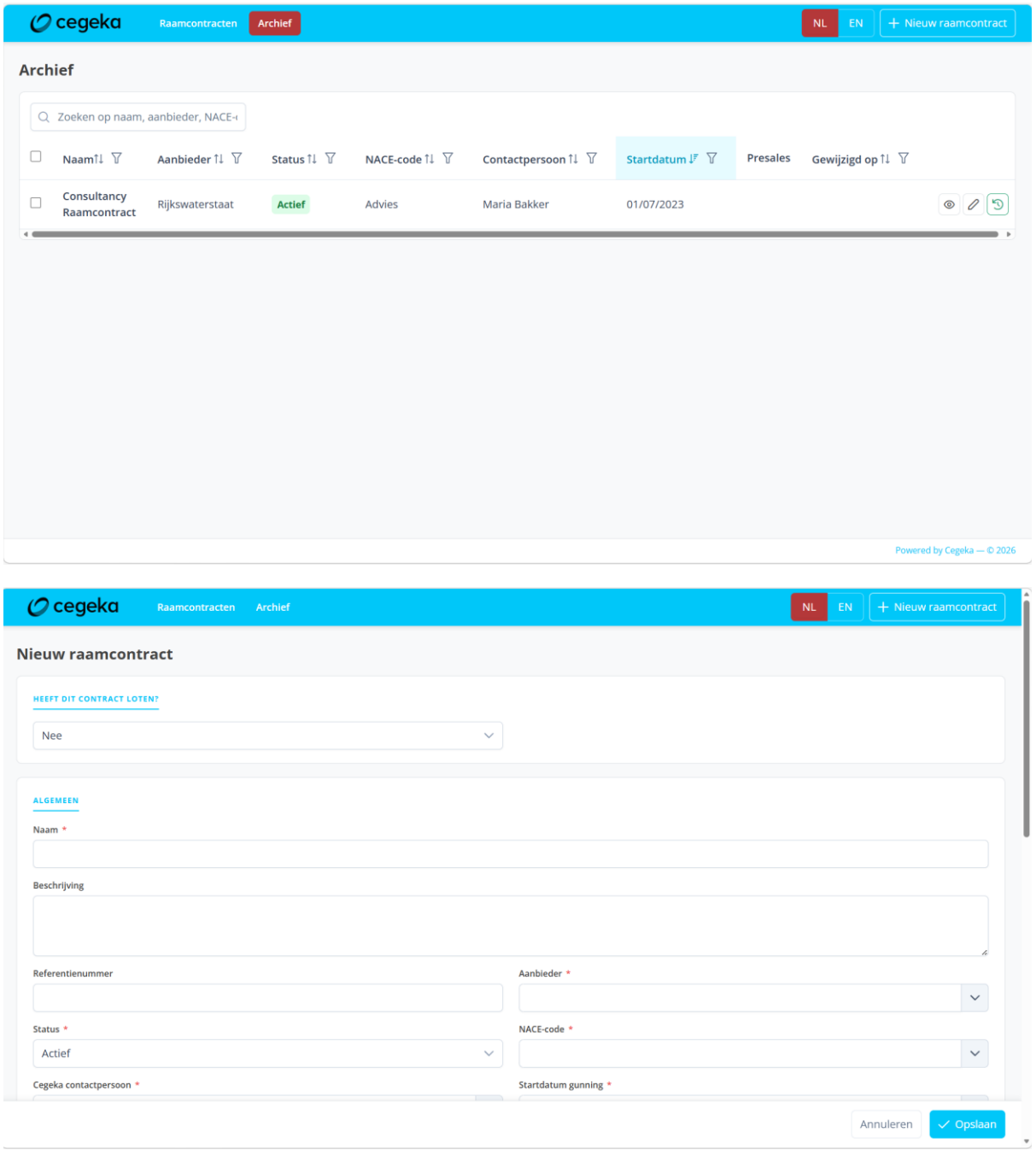
Figuur 47. Fragment uit het oorspronkelijke analysedocument van Cegeka, dat als startpunt diende voor de ontwikkeling van RaamContracten (Cegeka, 2026).



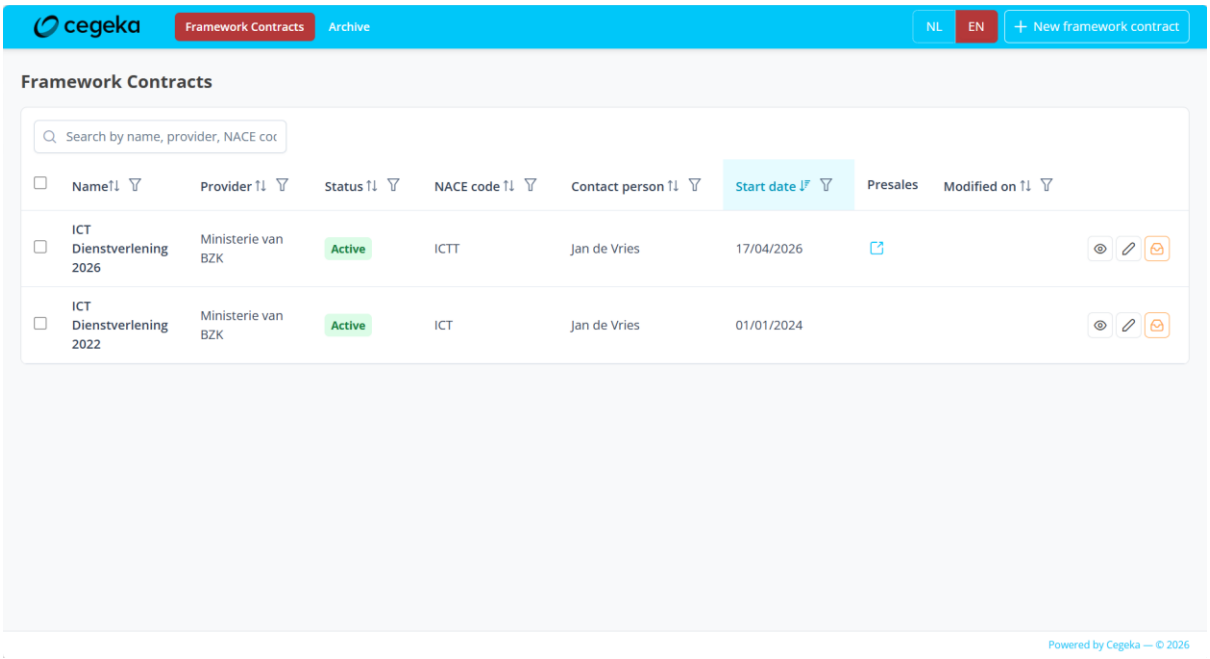
Figuur 48. Overzichtspagina van RaamContracten: skeleton loading state tijdens het ophalen van data (boven) en de uiteindelijke tabelweergave met sortering, filtering en zoekfunctionaliteit (onder).



Figuur 49. Detailweergave van een raamcontract met alle relevante secties zoals algemene gegevens, financiële gegevens en loten (boven), en het bijhorende bewerk-formulier (onder).



Figuur 50. Archieffunctionaliteit met enkel gearchiveerde raamcontracten (boven) en het formulier voor het aanmaken van een nieuw raamcontract (onder).



Figuur 51. Engelstalige versie van RaamContracten, dankzij de meertaligheid-ondersteuning binnen de applicatie.

4.2.4 Iteraties

Na het opleveren van de MVP werd gestart met het iteratief verbeteren van de applicatie. De aanpassingen werden doorgevoerd op basis van feedback uit wekelijkse meetings met de product owners en domeinexperten binnen Cegeka (Cegeka, 2026). Door regelmatig te toetsen aan hun expertise kon de applicatie stap voor stap dichter naar de werkelijke behoeften van de gebruikers groeien.

Een van de belangrijkste inzichten uit deze meetings was dat het oorspronkelijke analysedocument, dat als startpunt diende, op een aantal vlakken niet aansloot bij de werkelijke werking van raamcontracten binnen Cegeka. Hierdoor moest de flow van de applicatie op meerdere plaatsen worden bijgestuurd, onder andere door het toevoegen van bijkomende ja/nee-vragen die de gebruiker doorheen het aanmaakproces leiden. Daarnaast werd gevraagd om bepaalde gegevens te vervangen door gestandaardiseerde alternatieven, zoals het gebruik van NACE-codes voor de sector in plaats van een vrije tekstinvoer. Op die manier wordt de data binnen de applicatie eenduidiger en bruikbaarder voor verdere analyses.

Door de korte iteratiecycli kon snel op deze feedback worden ingespeeld en kon de applicatie stap voor stap verder verfijnd worden.

4.2.5 Besluit

RaamContracten was een bijzonder project, niet alleen omwille van het eindresultaat, maar vooral omwille van de manier waarop het tot stand is gekomen. Het volledig ontwikkelen van een applicatie via AI-driven development zonder zelf een regel code te schrijven, is een aanpak die vraagt om een andere mindset: in plaats van zelf te coderen, ligt de focus op het duidelijk formuleren van context, het kritisch beoordelen van wat de AI oplevert en het bijsturen waar nodig.

Een belangrijke beperking was dat het startdocument op meerdere plaatsen onnauwkeurig bleek. Dit zorgde voor aanpassingen middenin het bouwproces, wat trager werkt dan wanneer de analyse van bij het begin correct is. Dat leert dat AI-driven development de kwaliteit van de voorbereiding nog meer onder druk zet dan traditionele ontwikkeling: garbage in, garbage out geldt hier des te sterker.

Desalniettemin toonde dit project aan dat een ervaren ontwikkelaar met de juiste opzet en begeleiding van AI in relatief korte tijd een werkende MVP kan neerzetten. Het eindresultaat is een functionele applicatie die intern bij Cegeka in gebruik kan worden genomen en verder kan worden uitgebreid.

4.3 OWASP

4.3.1 Inleiding

OWASP is een interne tool binnen Cegeka die teams ondersteunt bij het opvolgen van beveiligingsrisico's op basis van de OWASP-standaarden. De applicatie was al grotendeels uitgewerkt, maar vertoonde op verschillende plaatsen visuele inconsistenties en kleine functionaliteitsfouten die de gebruikservaring negatief beïnvloedden. De opdracht bestond er dan ook niet uit om iets nieuws te bouwen, maar om de bestaande tool te verfijnen: een branding-aanpassing doorvoeren zodat de tool visueel aansluit bij SharePoint, en een reeks UI- en UX-bugs oplossen. In dit hoofdstuk komen achtereenvolgens het redesign en de bugfixes aan bod.

4.3.2 Redesign

Het redesign binnen OWASP bleef beperkt in omvang, aangezien de applicatie reeds een werkende en grotendeels uitgewerkte interface had. De voornaamste aanpassing situeerde zich op het vlak van branding. De navigatiebalk werd herwerkt zodat deze visueel aansluit bij de stijl van SharePoint, het interne platform dat binnen Cegeka veelvuldig wordt gebruikt. Door deze aanpassing krijgt de OWASP-tool een uitstraling die consistent is met de andere interne tools waarmee medewerkers dagelijks werken, wat bijdraagt aan een herkenbare en vertrouwde gebruikerservaring.

Figuur 52. Navigatiebalk van SharePoint, het interne Cegeka-platform dat als referentie diende voor het redesign van de OWASP-tool.

Figuur 53. Vernieuwde navigatiebalk van de OWASP-tool, visueel afgestemd op de stijl van SharePoint.

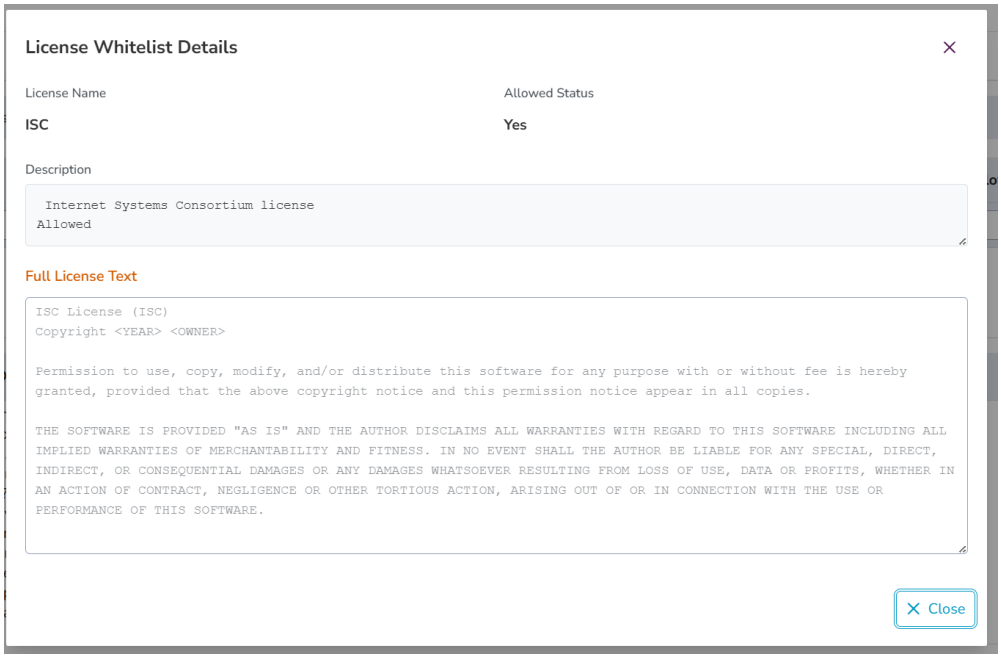
4.3.3 UI- en UX-bugs

Het grootste deel van het werk binnen OWASP bestond uit het oplossen van een reeks kleinere visuele en UX-bugs die zich doorheen de applicatie voordeden. Hoewel het op het eerste gezicht om kleine zaken ging, hadden ze samen een duidelijke impact op de gebruikservaring.

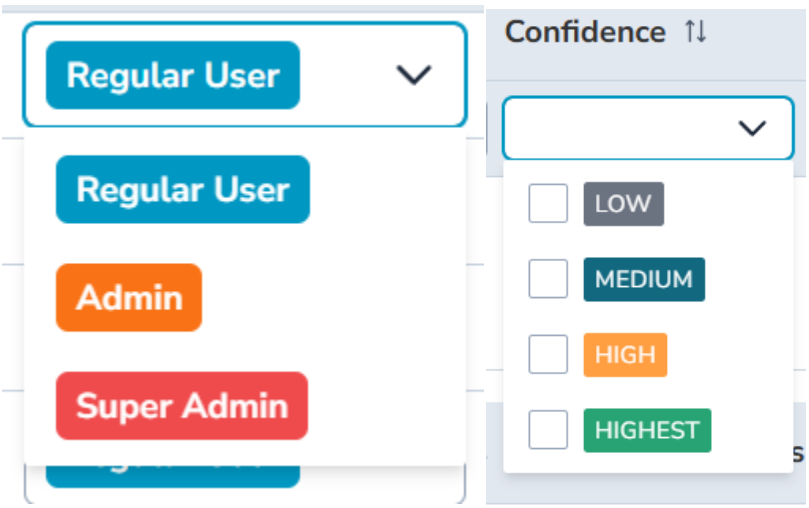
Op verschillende plaatsen in de applicatie was er sprake van text overlap, waarbij tekst over andere elementen heen viel of buiten de daarvoor voorziene ruimte uitstak. Dit werd opgelost door de styling van de betrokken componenten bij te werken. Daarnaast waren er tabellen die de beschikbare ruimte niet optimaal benutten, waardoor het overzicht verloren ging. Deze werden herwerkt zodat ze de ruimte op een efficiëntere manier gebruiken en de inhoud beter leesbaar wordt.

Verder waren er meerdere knoppen die niet correct werkten, waaronder save-knoppen die geen actie uitvoerden. Deze werden opnieuw aangesloten op de juiste functies, zodat gebruikers er opnieuw op kunnen vertrouwen. Ook enkele componenten die niet meer functioneerden, werden hersteld. Tot slot werden de toast messages aangepakt, aangezien deze in de oorspronkelijke versie niet altijd duidelijk communiceerden wat er precies gebeurde. Door de boodschappen te herformuleren en visueel beter te kaderen, krijgen gebruikers nu duidelijke en eenduidige feedback bij hun acties.

Figuur 54. Dependency-overzicht binnen de OWASP-tool, met de verbeterde "Show suppressed"-toggler (rood aangeduid) die voorheen niet correct functioneerde.



Figuur 55. License Whitelist Details met het uitgebreide tekstvak voor de licentie en de verbeterde comment-iconografie die duidelijker aangeeft of een comment al dan niet aanwezig is.



Figuur 56. Verbeterde selectie-componenten voor user role (links) en confidence (rechts), met visueel onderscheidende kleurcoderingen per niveau.

4.3.4 Besluit

OWASP vormde binnen deze stage een kleiner project, maar daarom niet minder waardevol. Waar de andere projecten draaiden rond het opzetten of herbouwen van applicaties, lag de focus hier op het verfijnen van een bestaande tool. Dit vroeg een andere manier van werken, waarbij eerst een grondige inwerking in een onbekende codebase noodzakelijk was, om vervolgens gericht aanpassingen door te voeren zonder de bestaande werking te verstoren.

Hoewel het redesign beperkt bleef tot de branding van de navigatiebalk, hadden de talrijke bugfixes een duidelijke impact op de gebruikservaring. Kleine zaken zoals tekst die overlapt, tabellen die de ruimte niet benutten of knoppen die niet reageren, lijken op zich misschien onschuldig, maar samen bepalen ze in grote mate hoe professioneel en betrouwbaar een applicatie aanvoelt. Door deze punten één voor één aan te pakken, draagt de OWASP-tool nu opnieuw bij aan een vlotte en aangename ervaring voor de Cegeka-medewerkers die er dagelijks gebruik van maken.

5 Conclusie van realisatie

6 LITERATUURLIJST

6.1 Frameworks en technologieën

Angular Team. (z.d.-a). Angular documentation. https://angular.dev

Angular Team. (z.d.-b). Angular coding style guide. https://angular.dev/style-guide

Microsoft. (z.d.-a). .NET documentation. https://learn.microsoft.com/en-us/dotnet/

Vue.js Team. (z.d.). Vue.js documentation. https://vuejs.org

6.2 UI-libraries en styling

Lucide Contributors. (z.d.). Lucide — Beautiful & consistent icon toolkit. https://lucide.dev

PrimeTek Informatics. (z.d.-a). PrimeNG — Angular UI component library. https://primeng.org

PrimeTek Informatics. (z.d.-b). PrimeVue — Vue UI component library. https://primevue.org

PrimeTek Informatics. (z.d.-c). PrimeIcons. https://primeng.org/icons

Tailwind Labs. (z.d.). Tailwind CSS documentation. https://tailwindcss.com/docs

6.3 Authenticatie en identity

Microsoft. (z.d.-b). Microsoft Entra ID documentation. https://learn.microsoft.com/en-us/entra/identity/

Microsoft. (z.d.-c). Microsoft Authentication Library for Angular (MSAL Angular). https://github.com/AzureAD/microsoft-authentication-library-for-js/tree/dev/lib/msal-angular

6.4 Internationalisatie

Ngx-translate Contributors. (z.d.). ngx-translate — The internationalization (i18n) library for Angular. https://github.com/ngx-translate/core

6.5 Progressive Web Apps

Angular Team. (z.d.-c). Service workers & PWA. https://angular.dev/ecosystem/service-workers

Mozilla Developer Network. (z.d.-a). Progressive web apps (PWAs). https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

Mozilla Developer Network. (z.d.-b). IndexedDB API. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

6.6 Security

OWASP Foundation. (z.d.). OWASP — Open Worldwide Application Security Project. https://owasp.org

6.7 Tooling en DevOps

GitHub. (z.d.). GitHub Copilot documentation. https://docs.github.com/en/copilot

Microsoft. (z.d.-d). Azure DevOps documentation. https://learn.microsoft.com/en-us/azure/devops/

6.8 Mondelinge bronnen

Cegeka. (2026). Wekelijkse meetings met product owners RaamContracten [Interne werksessies]. Cegeka nv, Hasselt.

Cegeka & DGZ. (2026). Functionele besprekingen FarmFit en analyse bestaande React-applicatie [Interne werksessies]. Cegeka nv, Hasselt.

Eijckmans, B. (2026). Stagebegeleiding vanuit klantperspectief [Mondelinge communicatie]. Cegeka nv, Hasselt.

Iacob, F. (2026). Technische begeleidingsgesprekken [Mondelinge communicatie]. Cegeka nv, Hasselt.

Vanhaeren, G. (2026). Wekelijkse opvolggesprekken stagementor [Mondelinge communicatie]. Cegeka nv, Hasselt.

6.9 Interne documenten Cegeka

Cegeka. (2026). Analysedocument applicatie Raamcontracten [Intern document]. Cegeka nv, Hasselt.

Cegeka & DGZ. (z.d.). Bestaande FarmFit React-applicatie en bijhorende vertaalbestanden [Interne codebase en documentatie]. Cegeka nv, Hasselt.