THOMAS MORE

Gheys – aanmeld en administratiesoftware

Realisatiedocument

Wout Van Ael
Student Bachelor in de Toegepaste Informatica

2024 - 2025

Inhoudsopgave

1. Inleiding

Hallo lezer,

Tijdens mijn stage heb ik gewerkt aan een aanmeld- en administratieprogramma voor de BCC-afdeling van Gheys Transport. In overleg met de stakeholders is gebleken dat de huidige manier van werken – waarbij chauffeurs hun gegevens op papier invullen en deze vervolgens handmatig worden overgetypt – inefficiënt en foutgevoelig is. Om dit proces te verbeteren, was er behoefte aan een digitale aanmeldoplossing die automatisch gekoppeld kan worden aan het bestaande administratieprogramma.

De digitale aanmelding maakt het mogelijk om gegevens direct te verwerken en uit te breiden met extra functies, zoals het toevoegen van foto's en het kiezen van een voertuigtype uit een dynamische lijst met afbeeldingen. Bovendien kan de aanmeldmodule beheerd worden vanuit het administratieprogramma, waarbij voertuigtypes en vertalingen eenvoudig kunnen worden toegevoegd, aangepast of verwijderd.

In dit document beschrijf ik de realisatie van dit project. Je leest hoe het idee is uitgewerkt, welke keuzes zijn gemaakt tijdens de ontwikkeling, en hoe het eindresultaat functioneert. Elk onderdeel van het project komt hierbij aan bod.

2. Analyse

In dit hoofdstuk geef ik een overzicht van de analyse die ik heb uitgevoerd ter voorbereiding van de ontwikkeling van het digitale aanmeld- en administratieprogramma. Hierbij zijn verschillende mogelijke ontwikkelomgevingen, tools en technologieën onderzocht en vergeleken. Vervolgens licht ik toe welke keuzes ik heb gemaakt en waarom, met als doel een zo efficiënt, toekomstbestendig en gebruiksvriendelijk mogelijke oplossing te realiseren.

2.1. Onderzoek van Blazor Component Libraries

Voor dit project was het gebruik van een component library niet strikt noodzakelijk, maar het gebruik ervan draagt sterk bij aan de snelheid van ontwikkeling en een consistente gebruikerservaring (UX). Component libraries bieden vooraf gebouwde UI-componenten zoals formulieren, knoppen en tabellen, die eenvoudig geïntegreerd kunnen worden in een Blazor-applicatie. Daarom heb ik een aantal populaire Blazor component libraries onderzocht en vergeleken op basis van beschikbaarheid, populariteit, documentatie en toekomstbestendigheid.

2.1.1. Fluent UI Blazor

- Ontwikkeld door Microsoft.
- Volledig open source en gratis.
- Gebaseerd op het Fluent Design System van Microsoft, wat consistentie biedt met andere Microsoft-applicaties.
- Documentatie is oke, maar het aantal beschikbare componenten is beperkter in vergelijking met andere libraries.

2.1.2. MudBlazor

- Open source en gratis.
- Op dit moment de populairste Blazor component library, met een grote community en actieve ontwikkeling.
- Uitstekende documentatie en veel tutorials beschikbaar.
- Biedt de meest uitgebreide componenten aan van de vergeleken libraries.

2.1.3. Radzen Blazor Components

- Open source en gratis voor een basisset aan componenten; bevat ook premium componenten en een betalende visual app builder.
- Documentatie is aanwezig, maar vaak beperkt tot voorbeeldcode. Een deel van de componenteigenschappen worden niet toegelicht.

- Werd eerder al gebruikt binnen Gheys, wat een belangrijk voordeel bood op vlak van ondersteuning en interne kennisdeling.

2.1.4. Conclusie

Hoewel MudBlazor technisch gezien de beste keuze zou zijn op vlak van populariteit, community-ondersteuning en documentatie, is er voor dit project gekozen om Radzen Blazor Components te gebruiken. Deze keuze werd gemaakt omdat Radzen al in een eerder project bij Gheys gebruikt werd. Hierdoor was er intern meer ondersteuning beschikbaar.

2.2. Onderzoek van Handtekening component

Een van de functionele vereisten van dit project was dat chauffeurs hun digitale aanmelding konden bevestigen met een handtekening. Hoewel het technisch mogelijk is om zo'n component volledig zelf te ontwikkelen met canvas-elementen en JavaScript, werd besloten om eerst bestaande oplossingen te onderzoeken. Dit om ontwikkeltijd te besparen en fouten te vermijden in functionaliteit die al goed uitgewerkt bestaat.

Hiervoor heb ik twee mogelijke libraries vergeleken die handtekeningfunctionaliteit bieden binnen een Blazor-omgeving.

2.2.1. Signature Pad (JavaScript)

- GitHub: szimek/signature_pad
- Een van de populairste JavaScript-libraries voor digitale handtekeningen.
- Biedt uitgebreide functionaliteit en een stabiele API.
- Kan in een Blazor-project geïntegreerd worden via JavaScript interop.
- Vereist echter extra werk om Blazor met JavaScript te laten communiceren, wat de complexiteit verhoogt.

2.2.2. Blazor SignaturePad

- GitHub: MarvinKlein1508/SignaturePad
- Specifiek ontwikkeld voor gebruik binnen Blazor-applicaties.
- Eenvoudig te integreren.
- Minder bekend en minder uitgebreid dan de JavaScript-versie, maar wel toereikend voor dit project.

2.2.3. Conclusie

Uiteindelijk is gekozen voor Blazor SignaturePad. De belangrijkste reden hiervoor is het gebruiksgemak binnen een Blazor-project. De component was onmiddellijk inzetbaar, zonder dat er extra JavaScript-interop nodig was. De functionaliteit voldeed bovendien volledig aan de eisen van het project, waardoor dit de meest efficiënte keuze was. Als extra functionaliteit nodig is kan dit component later vervangen worden sinds ze vergelijkbaar werken.

2.3. Onderzoek naar Pdf generatie

Voor de applicatie was het noodzakelijk om PDF-bestanden te kunnen genereren. Dit was vereist om drie redenen: ten eerste om de digitale aanmeldingen automatisch via e-mail te kunnen versturen in PDF-formaat; ten tweede om werkbonnen met spoelinformatie correct en overzichtelijk af te drukken; en ten derde om het EFTCO-cleaningdocument te kunnen printen. Om dit te realiseren werd onderzocht welke technologie het best geschikt was voor PDF-generatie binnen de context van deze applicatie.

2.3.1. QuestPDF

- Open source met een uitgebreide set features.
- 12.9K sterren op GitHub wat duidt op populariteit en actieve ontwikkeling.
- Makkelijk te gebruiken via een C# Fluent API.
- Nadelig is de prijs van €700 voor zakelijk gebruik.

2.3.2. System.Drawing.Printing.PrintDocument

Vereist echter een printerdriver met “Save as PDF”-functionaliteit, wat de betrouwbaarheid en portabiliteit beperkt. Dit maakt het minder geschikt voor een webapplicatie die platformonafhankelijk moet zijn.

2.3.3. iText Core/Community

- Minder populair met 1.8K GitHub-sterren.
- Betaalde licentie voor commercieel gebruik.

2.3.4. PDFsharp

- Open source en gratis te gebruiken.
- Complexe API, wat het gebruik en onderhoud bemoeilijkt.

2.3.5. PuppeteerSharp

- Headless Chrome .NET API die HTML omzet naar PDF via Google Chrome.

- Populair en vrij eenvoudig in gebruik met goede documentatie.
- Ondersteunt alle benodigde functies voor het project.
- Wordt direct in de applicatie gebruikt, zonder extra infrastructuur.

2.3.6. Gotenberg

- Populair met 9.3K GitHub-sterren.
- Gebruikt ook Google Chrome om HTML naar PDF te converteren.
- Draait in een Docker-container, wat momenteel niet beschikbaar is bij Gheys.
- Er is een API-client beschikbaar voor C# integratie.
- Zou een goede tweede keuze zijn als containerisatie bij Gheys geïntroduceerd wordt.

2.3.7. Carbone

- Vergelijkbaar met Gotenberg, deels betaald maar ook open source te hosten.
- Biedt extra functionaliteit zoals templatebeheer via web GUI.
- Niet gekozen vanwege overbodige functionaliteit voor dit project en licentiekosten.

2.3.8. Scriban

- Open source template-engine voor C#.
- Ondersteund custom functies die in C# geschreven kunnen worden voor in de templates te gebruiken.
- Wordt gebruikt om de HTML-sjablonen te genereren die vervolgens door PuppeteerSharp worden omgezet naar PDF.

2.3.9. Conclusie

PuppeteerSharp in combinatie met Scriban scoort het hoogste dankzij de combinatie van gratis gebruik, hoge functionaliteit, eenvoud en directe compatibiliteit met de infrastructuur bij Gheys. Gotenberg is een goede tweede keuze maar wordt minder aantrekkelijk vanwege de noodzaak van Docker en dus een extra service.

2.4. Gebruikte tools

Hierbij al de tools die ik tijdens mijn stage heb gebruikt tijdens de ontwikkeling van het project.

2.4.1. Draw.io

Draw.io heb ik gebruikt voor het maken van architectuurschema's, databasemodellen en use case-diagrammen.

2.4.2. Word

Ik heb Word gebruikt voor dit document te maken en het projectplan.

2.4.3. Figma

Om een duidelijk beeld te geven van het uiterlijk van de applicatie, heb ik wireframes gemaakt met Figma. Figma was hiervoor een goede keuze, omdat het online gedeeld kan worden en vlot werkt. Ik heb ook de MudBlazor UI Kit: Blazor Component Library gebruikt voor de wireframes te maken. Dit was nog tijdens de projectplanningsfase waarna pas beslist was om de Radzen Component Library te gebruiken.

2.4.4. Cwebp

Cwebp is een tool ontwikkeld door Google om afbeeldingen te converteren naar het .webp-formaat. Ik heb deze tool gebruikt omdat de oorspronkelijke foto's vrij groot waren, wat ervoor zorgde dat het enkele seconden duurde voordat alle afbeeldingen van de voertuigtypes geladen waren. Door de afbeeldingen met Cwebp te comprimeren, werden ze aanzienlijk kleiner, waardoor het laden nog maar enkele tientallen milliseconden per afbeelding duurde.

2.4.5. Blazor

De front-end van de applicatie is ontwikkeld met Blazor. Gheys heeft ervoor gekozen om dit framework in de toekomst te gebruiken voor al hun webapplicaties, en wilde dit ook toepassen binnen deze stageopdracht. Blazor maakt het mogelijk om interactieve webapplicaties te bouwen met C#, wat goed aansluit bij de bestaande technologieën binnen het bedrijf. Bovendien zorgt het ervoor dat zowel front-end als back-end in dezelfde programmeertaal ontwikkeld kunnen worden, wat het onderhoud en de samenwerking vereenvoudigt.

2.4.6. Entity Framework Core

Voor de communicatie met de database is gebruikgemaakt van Entity Framework Core. Ik heb voor dit framework gekozen omdat het robuust en gebruiksvriendelijk is, en de mogelijkheid biedt om met de Code First-aanpak te werken. Hierdoor kan de database direct vanuit de C#-klassen worden opgebouwd, wat het ontwikkelproces aanzienlijk versnelt en overzichtelijker maakt. Dit zorgt ervoor dat het ontwikkelen van een applicatie sneller gaat dan zonder een ORM.

Daarnaast heb ik het gebruikersbeheer via Identity van Entity Framework Core gebruikt, zodat ik data aan specifieke gebruikers kan koppelen.

2.4.7. MS SQL-Server

Voor de staging-omgeving is gebruikgemaakt van een MS SQL Server-database, omdat dit de standaarddatabase is binnen Gheys. De keuze hiervoor sluit aan bij de bestaande infrastructuur van het bedrijf.

Hoewel MS SQL Server in dit project is gebruikt, is de applicatie dankzij Entity Framework Core niet afhankelijk van één specifieke database. Andere databases zoals SQLite, PostgreSQL, MySQL of MariaDB kunnen ook zonder veel aanpassingen worden gebruikt.

2.4.8. Visual Studio

Als IDE heb ik Visual Studio gebruikt sinds deze door Gheys voorzien werd. En een goede IDE maakt de ontwikkeling van de applicatie sneller.
Voor de ontwikkeling van de applicatie heb ik Visual Studio gebruikt als IDE. Omdat deze werd door Gheys ter beschikking gesteld. Dankzij functies zoals IntelliSense, debuggingtools en integratie met Git, verloopt het ontwikkelproces efficiënter en overzichtelijker. Een goede IDE zoals Visual Studio verhoogd de snelheid van ontwikkeling.

2.4.9. Scriban

Voor het genereren van HTML op basis van sjablonen heb ik gebruikgemaakt van de Scriban template engine. Deze tool maakt het mogelijk om op een flexibele manier HTML-pagina's te genereren met dynamische inhoud, zoals de data uit de digitale aanmelding.

Een van de voordelen van Scriban is dat het tijdens het genereren ook C#-functies kan aanroepen. Dit heb ik benut om onder andere afbeeldingen op te halen en te converteren naar Base64, zodat ze direct in de HTML kunnen worden weergegeven. Daarnaast heb ik logica ingebouwd, bijvoorbeeld om automatisch de juiste taalversie van een vertaling te tonen.

2.4.10. PuppeteerSharp

Ik heb PuppeteerSharp gebruikt om de HTML, die eerder met Scriban werd gegenereerd, om te zetten naar een PDF-bestand. PuppeteerSharp is een .NET-wrapper rond de populaire Puppeteer-tool van Google, en maakt gebruik van de Chromium-browser om webpagina's nauwkeurig te renderen.

Deze tool leek mij de beste keuze omdat het een robuuste en betrouwbare oplossing biedt met ondersteuning voor verschillende platformen (cross-platform). Bovendien is PuppeteerSharp gratis en open source, wat het toegankelijk en flexibel maakt. Dankzij deze tool kon ik een stabiele en efficiënte workflow opzetten voor het automatisch genereren van goed opgemaakte PDF-documenten.

2.4.11. ChatGPT

Ik heb ChatGPT ingezet om repetitieve taken te versnellen en ondersteuning te bieden bij het debuggen van code. Daarnaast gebruikte ik ChatGPT om teksten te vertalen wanneer ik de betreffende taal niet beheers. Ook heeft het geholpen bij het herschrijven van teksten om taalfouten te voorkomen en de formuleringen te verbeteren. Dankzij deze assistentie kon ik efficiënter werken en de kwaliteit van mijn documentatie en code verhogen.

2.4.12. Bcp utility

De BCP Utility is een officiële commandline-tool van Microsoft om data van een Microsoft SQL Server te kopiëren. Ik heb deze tool gebruikt om data te importeren in de database, bijvoorbeeld voor het overzetten van vlootnummers en kentekenplaten van trekkers en opleggers.

2.4.13. Microsoft SQL Server Management Studio

Deze tool heb ik gebruikt om via een GUI SQL-scripts uit te voeren op de database. Dit werkt wat beter dan de ingebouwde SQL Server Object Explorer van Visual Studio.

2.4.14. Radzen

Om mooie UI-componenten te gebruiken in de Blazor front-end heb ik de Radzen Blazor component library ingezet.

2.4.15. Microsoft Localization

Ik heb de lokalisatieservices gebruikt om de app te vertalen aan de hand van resourcebestanden. Deze package voorziet standaard in de native taalnaam van elke taal, waardoor gebruikers deze niet zelf hoeven in te voeren wanneer ze een nieuwe taal toevoegen. De daadwerkelijke vertalingen van de appteksten moeten gebruikers echter wel zelf toevoegen.

2.4.16. Blazor SignaturePad

Dit is een open source component waarmee gebruikers via Blazor een handtekening kunnen zetten. Ik heb dit component gebruikt zodat de chauffeur zijn aanmelding kan bevestigen met een handtekening.

2.4.17. Bootstrap

Ik heb Bootstrap gebruikt voor de lay-out van de front-end, omdat het duidelijk maakt welke styling er is toegepast. Aangezien Bootstrap door veel mensen gekend is, hoeven ontwikkelaars minder tijd te besteden aan het uitzoeken van de betekenis van bepaalde klassen, wat het onderhoud en de samenwerking vergemakkelijkt. Wanneer Bootstrap geen geschikte klasse bood voor specifieke styling, heb ik gebruikgemaakt van eigen custom classes.

3. Lokalisatie

Een van de kernfunctionaliteiten van de applicatie is de ondersteuning van meertaligheid. Dit was ook de voornaamste reden waarom Zoho Creator niet geschikt bleek als ontwikkelplatform.

3.1. Beperking van Zoho Creator

Initieel was het de bedoeling om de applicatie te ontwikkelen met Zoho Creator. Bij nader onderzoek bleek echter dat Zoho Creator geen flexibele ondersteuning biedt voor taalkeuze door de eindgebruiker. De platformfunctionaliteit beperkt zich tot het automatisch aanpassen van de taal op basis van de browserinstellingen of de ingestelde taal van een ingelogde gebruiker. In ons geval werken chauffeurs echter zonder account, en zelfs indien ze wel zouden inloggen, is het wijzigen van de taalinstellingen te omslachtig en niet intuïtief genoeg voor snel gebruik. Hierdoor viel Zoho Creator af als optie.

3.2. Lokalisatie via Database

Voor de lokalisatie van de applicatie heb ik gekozen voor een combinatie van een eigen databaseopzet en de NuGet-package Microsoft.Extensions.Localization.

In de database zijn er tabellen voorzien voor zowel de beschikbare talen als de vertalingen van dynamische inhoud, zoals keuzeknoppen (checkboxes) en radioknoppen. Hieronder volgt een voorbeeld van het model voor de Language-tabel:

```
public class Language
{
    public Guid Id { get; set; } = Guid.NewGuid();
    public string NeutralCultureName { get; set; } = string.Empty;
    public bool IsActive { get; set; } = true;
    public string? ImageUrl { get; set; }
}
```

Deze tabel wordt gekoppeld aan een vertalingstabel, zoals onderstaand voorbeeld voor de vertaling van wasbeurttypes (CleaningTypeTranslation):

```
public class CleaningTypeTranslation : ITranslatable
{
    public Guid CleaningTypeId { get; set; }
    public Guid LanguageId { get; set; }
    public string Name { get; set; } = string.Empty;
}
```

Op basis van de gekozen LanguageId kan de applicatie automatisch de juiste vertalingen laden voor de geselecteerde taal.

3.3. Gebruik van Microsoft.Extensions.Localization

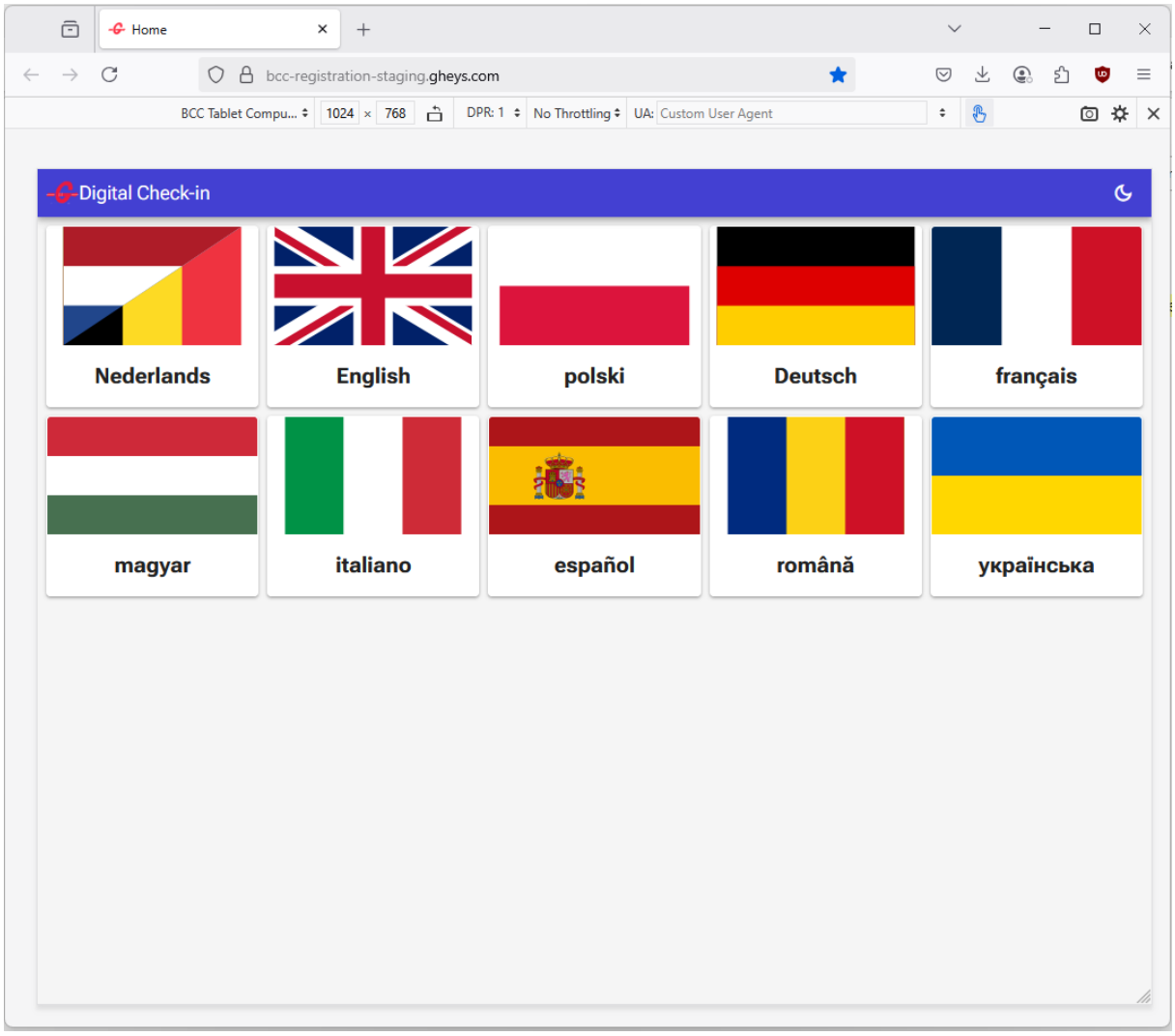
Voor de statische onderdelen van de gebruikersinterface maak ik gebruik van de NuGet-package Microsoft.Extensions.Localization. Via .resx-bestanden kunnen vertalingen worden beheerd voor elementen zoals knoppen, titels en foutmeldingen.

Deze vertalingen zijn niet aanpasbaar door de eindgebruiker en moeten bij het toevoegen van een nieuwe taal dus door een ontwikkelaar worden aangevuld.

Een bijkomend voordeel van deze package is de ondersteuning van het automatisch ophalen van de native naam van een taal op basis van de cultuurcode. Dit maakt het mogelijk om in het taalselectiemenu de naam van een taal in zijn oorspronkelijke vorm weer te geven, zonder dat dit manueel vertaald hoeft te worden:

```
new CultureInfo(language.NeutralCultureName).NativeName
```

Dankzij deze aanpak is de applicatie flexibel uitbreidbaar naar nieuwe talen en blijft de gebruikservaring voor internationale chauffeurs optimaal.



Figuur 1 Knipsel van taalkeuze scherm

4. Pdf genereren via C#

Een belangrijk onderdeel van de applicatie is het automatisch genereren van een PDF-document als bevestiging van een digitale aanmelding, een spoelbon, of iets anders. Dit document is bedoeld om af te drukken of per e-mail te versturen. In dit hoofdstuk wordt uitgelegd hoe de PDF-generatie technisch is uitgewerkt met C#, welke tools en bibliotheken daarbij zijn ingezet, en welke informatie in het document wordt opgenomen.

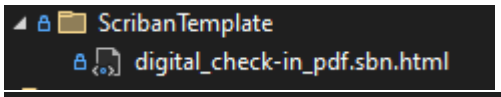
4.1. Scriban

Voor het genereren van PDF-documenten op basis van dynamische gegevens heb ik gebruikgemaakt van Scriban, een krachtige template-engine. Scriban wordt gebruikt om HTML te genereren aan de hand van vooraf gedefinieerde templates die worden gevuld met data op runtime.

4.1.1. Digital_checkin.sbn.html

Deze template bevat HTML gecombineerd met Scriban-syntax, waarmee velden dynamisch worden ingevuld op basis van de aangeleverde gegevens. In dit geval werden de Checkin-entity en een lijst van Article-entities aan de template meegegeven. Hierdoor kunnen in het gegenereerde document ook artikelen worden getoond die nog niet expliciet aan de check-in zijn toegevoegd, maar wel relevant zijn voor de context.

Ik heb ervoor gekozen om de template in bestandvorm op te slaan, omdat dit eenvoudiger aan te passen is met IntelliSense. Dit zou echter evengoed een string kunnen zijn die in de database is opgeslagen.




```
<!-- Display available but unselected options (unchecked) -->
{{ if has_article_options }}
{{ for article_option in extra_article_options }}
<!-- Only show if not already in checkin.extra_articles -->
{{ if !(checkin.extra_articles | array.map 'article_id' | array.contains article_option.id) }}
<div class="col border d-flex align-content-center px-2 text-muted">
    {{ if article_option.is_multiple_allowed == true }}
    <input type="checkbox" class="me-2" />
    <div class="d-inline-flex w-100 justify-content-between">
        <span>{{ article_option.translations | custom_functions.get_dutch_or_english_or_first_translation }}</span>
        <span class="w-50">aantal: ...</span>
    </div>
    {{ else }}
    <input type="checkbox" class="me-2" />
    <span>{{ article_option.translations | custom_functions.get_dutch_or_english_or_first_translation }}</span>
    {{ end }}
</div>
{{ end }}
```

4.1.2. Custom functies

Om de HTML-template correct te laten renderen met alle gewenste data en functionaliteit, heb ik enkele custom functies toegevoegd die vanuit de Scriban-template konden worden aangeroepen. Hieronder een overzicht van de belangrijkste functies

4.1.2.1. ByteArrayToBase64String

Deze functie zet een byte-array om naar een Base64-string. Dit was vooral handig om handtekeningen, die als byte-array in de database zijn opgeslagen, correct weer te geven als inline afbeelding in de gegenereerde PDF.

```
public static string ByteArrayToBase64String(byte[] bytes)
{
    return Convert.ToBase64String(bytes);
}
```

```
<tr>
    <td colspan="12">
        <p class="m-0">Handtekening chauffeur:</p>
        {{ if checkin.driver_signature }}
        <div class="container-fluid">
            <div class="row">
                <div class="col border d-flex align-content-center px-2" style="min-height: 100px;">
                    <div class="w-100" style="min-height: 100px">
                        <img class="w-100 d-inline-flex signature" style="object-fit: contain"
                             src="data:{{ checkin.signature_mime_type }};base64,{{ checkin.driver_signature | custom_functions.byte_array_to_base64_string }}" />
                    </div>

                </div>
            </div>
        </div>
        {{ end }}
    </td>
</tr>
```

4.1.2.2. GetDutchOrEnglishOrFirstTranslation

Deze functie zoekt eerst een Nederlandse vertaling in een lijst van vertalingen. Als die er niet is, probeert ze een Engelse te vinden. Als ook die ontbreekt, wordt de eerste beschikbare vertaling gebruikt. Zo wordt altijd een geschikte tekst getoond in de PDF.

```
public static string GetDutchOrEnglishOrFirstTranslation(IEnumerable<ITranslatable> translations)
{
    var translation = translations.FirstOrDefault(t =>
        t.CultureName.Equals("nl", StringComparison.OrdinalIgnoreCase));

    translation ??= translations.FirstOrDefault(t =>
        t.CultureName.Equals("en", StringComparison.OrdinalIgnoreCase));

    return translation?.Text ?? translations.FirstOrDefault()?.Text ?? "No translations found";
}
```

4.1.2.3. Sort

Omdat Scriban geen ondersteuning biedt voor het sorteren van lijsten op geneste eigenschappen, heb ik een eigen sorteermethode gemaakt. Deze functie maakt het mogelijk om lijsten in de template op een specifiek veld te sorteren.

4.1.2.4. LoadAssetFunction

Deze custom functie laadt bestanden (zoals afbeeldingen of logo's) van het bestandssysteem en zet ze om naar een Base64-string, zodat ze gebruikt kunnen worden in de HTML-template.

```
public class LoadAssetFunction(IAssetLoader AssetLoader) : IScriptCustomFunction
{
```

```
    public object Invoke(
        TemplateContext context,
        ScriptNode callerContext,
        ScriptArray arguments,
        ScriptBlockStatement blockStatement)
    {
        ...
    }
}
```

4.2. PuppeteerSharp

Voor het genereren van PDF-bestanden heb ik gebruikgemaakt van de NuGet-package PuppeteerSharp. Deze library is een .NET-wrapper rond headless Chromium, waarmee je webpagina's betrouwbaar kunt renderen en omzetten naar PDF.

In dit project zorgde ik ervoor dat alle benodigde data en afbeeldingen direct in de HTML-template met Scriban verwerkt werden. Daardoor zijn er tijdens het renderen geen netwerkverzoeken nodig, wat het proces sneller en stabieler maakt.

In dit project maak ik gebruik van een reeds geïnstalleerde versie van Chromium op de computer om PuppeteerSharp aan te sturen.

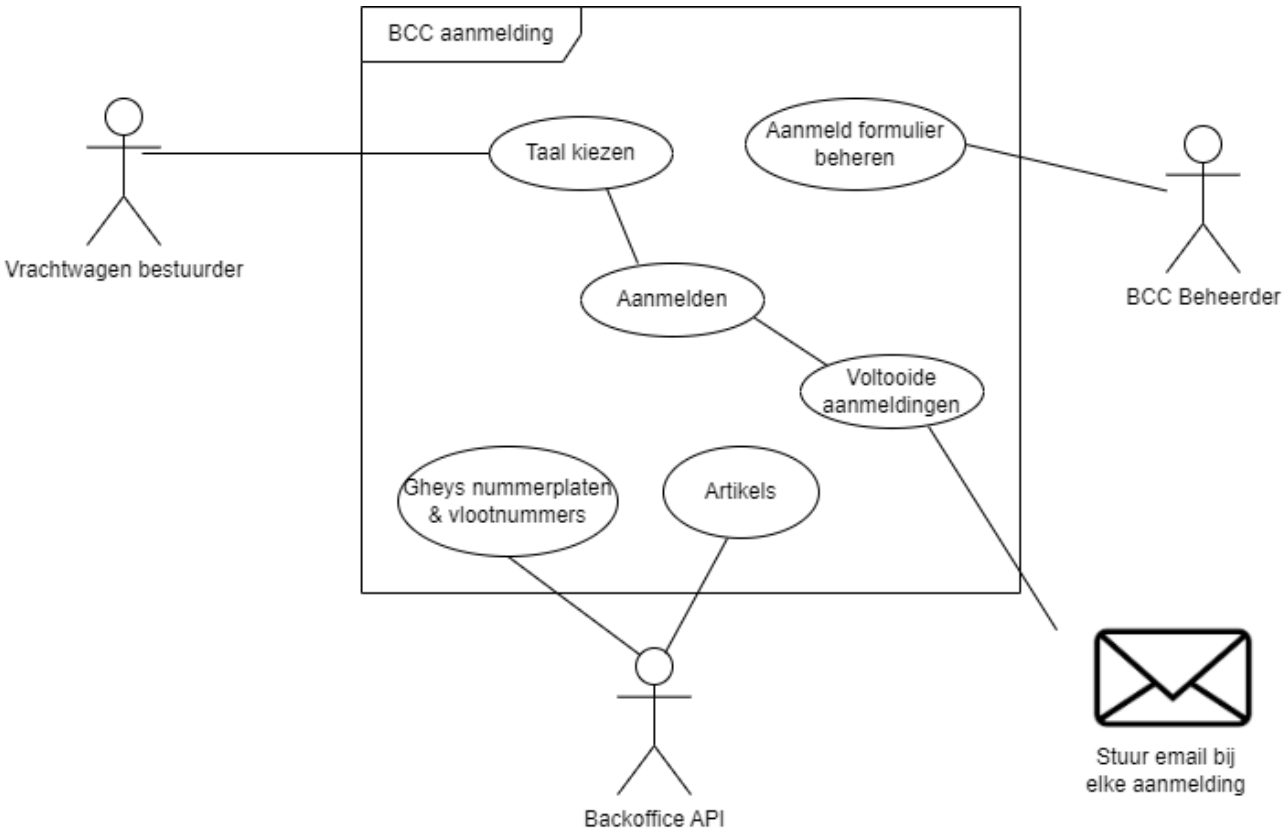
Met PuppeteerSharp wordt de HTML-inhoud via C# geladen en als PDF geëxporteerd

Code voorbeeld:

```
using var browser = await Puppeteer.LaunchAsync(new LaunchOptions
{
    Headless = true,
    ExecutablePath = @"C:\Path\To\Chromium\chrome.exe"
});
using var page = await browser.NewPageAsync();
await page.SetContentAsync(htmlContent);
await page.PdfAsync("output.pdf");
```

5. Eisenanalyse

USE CASE DIAGRAM



USE CASE BESCHRIJVINGEN

5.1.1. Taal kiezen

Hier kan de gebruiker de taal wijzigen naar een van de volgende talen:

- Nederlands
- Engels
- Pools
- Duits
- Frans
- Hongaars
- Italiaans
- Spaans
- Roemeens
- Oekraïens

5.1.2. Aanmelden

In dit onderdeel kan de gebruiker aangeven welk type reiniging voor zijn vrachtwagen nodig is.

5.1.3. Gheys nummerplaten & vlootnummers

De API stuurt alle nummerplaten van Gheys-vrachtwagens en de bijbehorende vlootnummers door naar de database, zodat deze gegevens beschikbaar zijn in het aanmeldformulier.

5.1.4. Aanmeld formulier beheren

In dit onderdeel kunnen aanpassingen worden gemaakt, zoals het wijzigen van voertuigtypes en andere keuzemogelijkheden in het formulier.

5.1.5. Bekijk aanmeldingen

Hier kunnen alle aanmeldingen die zijn ingediend worden geraadpleegd. Deze aanmeldingen worden ook per e-mail verzonden naar de BCC-beheerder.

6. Wassen

Wanneer een chauffeur zijn vrachtwagen wil laten wassen, moet hij de benodigde informatie doorgeven. Dit kan zowel via een papieren formulier als digitaal via Zoho Creator.

Volgende velden moeten dan ingevult worden:

- Datum en tijd
  - Onzichtbaar maar moet wel bij aanmelding
- Naam chauffeur
- Klant
- Referentienummer
  - Optioneel
- Vlootnummer trekker
- Nummerplaat trekker
  - Auto invullen bij Gheys d.m.v. vlootnummer
- Vlootnummer oplegger
  - Als klant Gheys is dan is dit veld verplicht
- Nummerplaat oplegger
  - Als klant Gheys is wordt dit veld verplicht & auto ingevult a.h.v. de vlootnummer
- Containernummer
- Disinfecteren
  - Ja of nee
- Type voortuig
  - Is een meerkeuze lijst in Zoho Creator
- Extra opmerkingen
  - OPTIONEEL
- Handtekening

Figuur 2 Knipsel Figma design wassen Zoho Creator

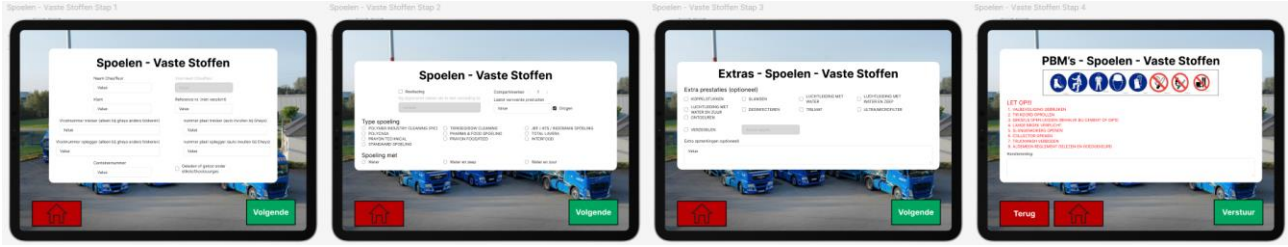
7. Spoelen – Vaste stoffen

Als een chauffeur zijn oplegger wilt spoelen moet hij ook een formulier invullen zodat het BCC weet hoe het gespoelt moet worden.

Volgende velden moeten dan ingevult worden.

- Datum en tijd
  - Onzichtbaar maar moet wel bij aanmelding
- Naam chauffeur
- Klant
- Referentienummer
  - Optioneel
- Vlootnummer trekker
- Nummerplaat trekker
  - Automatisch invullen bij Gheys d.m.v. vlootnummer
- Vlootnummer oplegger
- Nummerplaat oplegger
  - Automatisch invullen bij Gheys d.m.v. vlootnummer
- Containernummer
- Laatst vervoerde product
- Drogen
  - Voor het compartiment
- Geladen of gelost onder stifstof
- Restlading
- Kg registreren bij een restlading
- Type spoeling
  - POLYMER INDUSTRY CLEANING (PIC)
  - TRINSEO/DOW CLEANING
  - JBF / ATS / INDORAMA SPOELING
  - POLYCASA
  - PHARMA & FOOD SPOELING
  - TOTAL LAVERA
  - PRAYON TECHNICAL
  - PRAYON FOOD/FEED
  - INTERFOOD
  - STANDAARD SPOELING
- Spoeling met
  - Water
  - Water en zeep
  - Water en zuur
- Extra prestaties (optioneel)
  - KOPPELSTUKKEN
  - SLANGEN
  - LUCHTLEIDING MET WATER
  - LUCHTLEIDING MET WATER EN ZEEP
  - LUCHTLEIDING MET WATER EN ZUUR
  - DESINFECTEREN
  - TRILMAT
  - ULTRA/MICROFILTER
  - VERZEGELEN

    - ONTGEUREN
- Extra opmerkingen
    - OPTIONEEL
- Handtekening



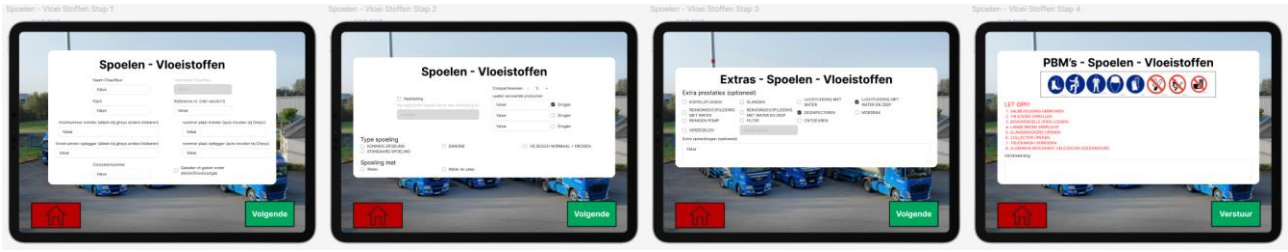
Figuur 3 Knipsel Figma design Spoelen vaste stoffen Zoho Creator

8. Spoelen – Vloeistoffen

Wanneer een chauffeur een tankwagen wil laten spoelen.

Volgende velden moeten dan ingevuld worden.

- Datum en tijd
  - Onzichtbaar maar moet wel bij aanmelding
- Naam chauffeur
- Klant
- Referentienummer
  - Optioneel
- Vlootnummer trekker
- Nummerplaat trekker
  - Automatisch invullen bij Gheys d.m.v. vlootnummer
- Vlootnummer oplegger
- Nummerplaat oplegger
  - Automatisch invullen bij Gheys d.m.v. vlootnummer
- Containernummer
- Aantal compartimenten
- Laatst vervoerde product(en)
  - Elk compartiment is een mogelijk product
- Drogen
  - Ja of nee vraag voor elk compartiment
- Geladen of gelost onder stikstof of koolzuurgas
- Restlading
- Kg registreren bij een restlading
- Type spoeling
  - KONINGS SPOELING
  - DANONE
  - VD BOSCH NORMAAL + DROGEN
  - STANDAARD
- Spoeling met
  - Water
  - Water en zeep
- Extra prestaties (optioneel)
  - KOPPELSTUKKEN
  - SLANGEN
  - LUCHTLEIDING MET WATER
  - LUCHTLEIDING MET WATER EN ZEEP
  - REINIGINGS(CIP)LEIDING MET WATER
  - REINIGINGS(CIP)LEIDING MET WATER EN ZEEP
  - DESINFECTEREN
  - MORSBAK
  - REINIGEN POMP
  - FILTER
  - VERZEGELEN
  - ONTGEUREN
- Extra opmerkingen
  - OPTIONEEL
- Handtekening



Figuur 4 Knipsel Figma design Spoelen vloeistoffen Zoho Creator

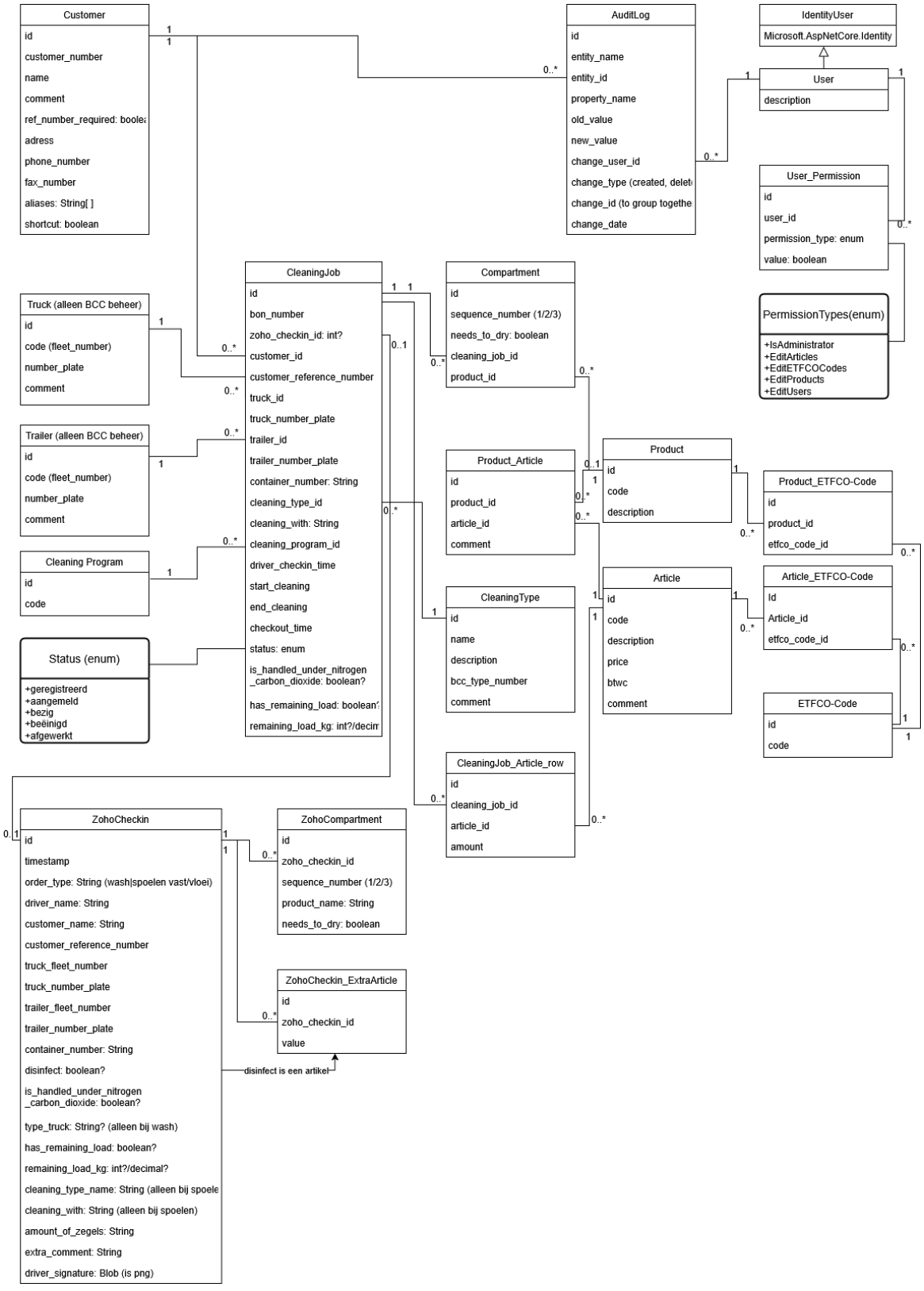
ZOHO CREATOR

De Zoho Creator-app wordt ontwikkeld met ondersteuning door Aziri. Oorspronkelijk was het plan om de aanmeld functionaliteit in Zoho Creator te realiseren, op basis van een door Aziri opgesteld schema. Tijdens de ontwikkeling bleek echter dat het instellen van de taal via knoppen technisch niet haalbaar was binnen Zoho Creator.

Als mogelijke oplossing stelde Aziri voor om het low-code-platform WeWeb in te zetten om deze functionaliteit alsnog te realiseren. Na evaluatie is besloten dat deze aanpak onnodige complexiteit met zich mee zou brengen. Daarom is de keuze gemaakt om de digitale aanmelding, inclusief de taalinstelling, eveneens in Blazor te ontwikkelen.

Figuur 5 Zoho Creator logo

9. Database diagram (Versie met zoho)

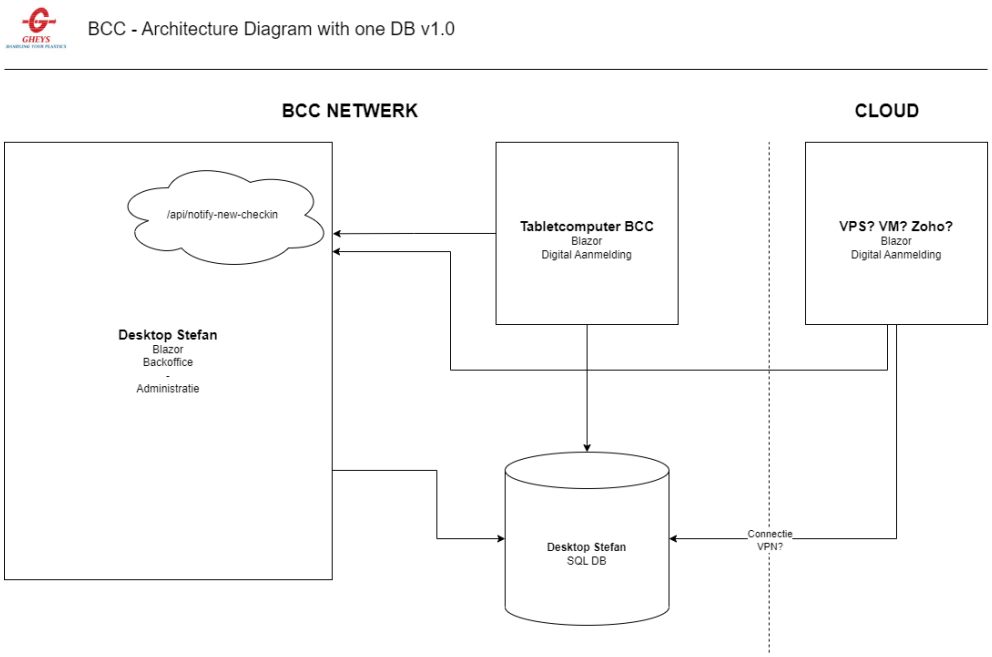


Figuur 6 Database entities diagram

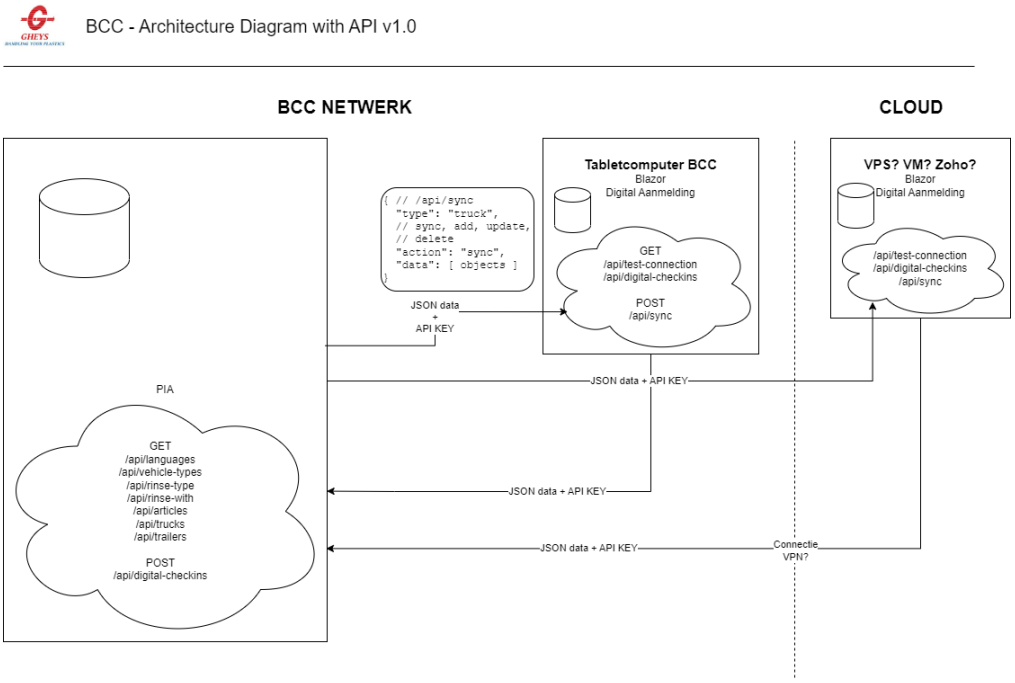
10. Architectuur schema's

Hieronder zijn alle schema's die ik gemaakt heb. Het schema 'One DB' wordt gebruikt voor dit project maar de andere schema's zijn ook mogelijkheden.

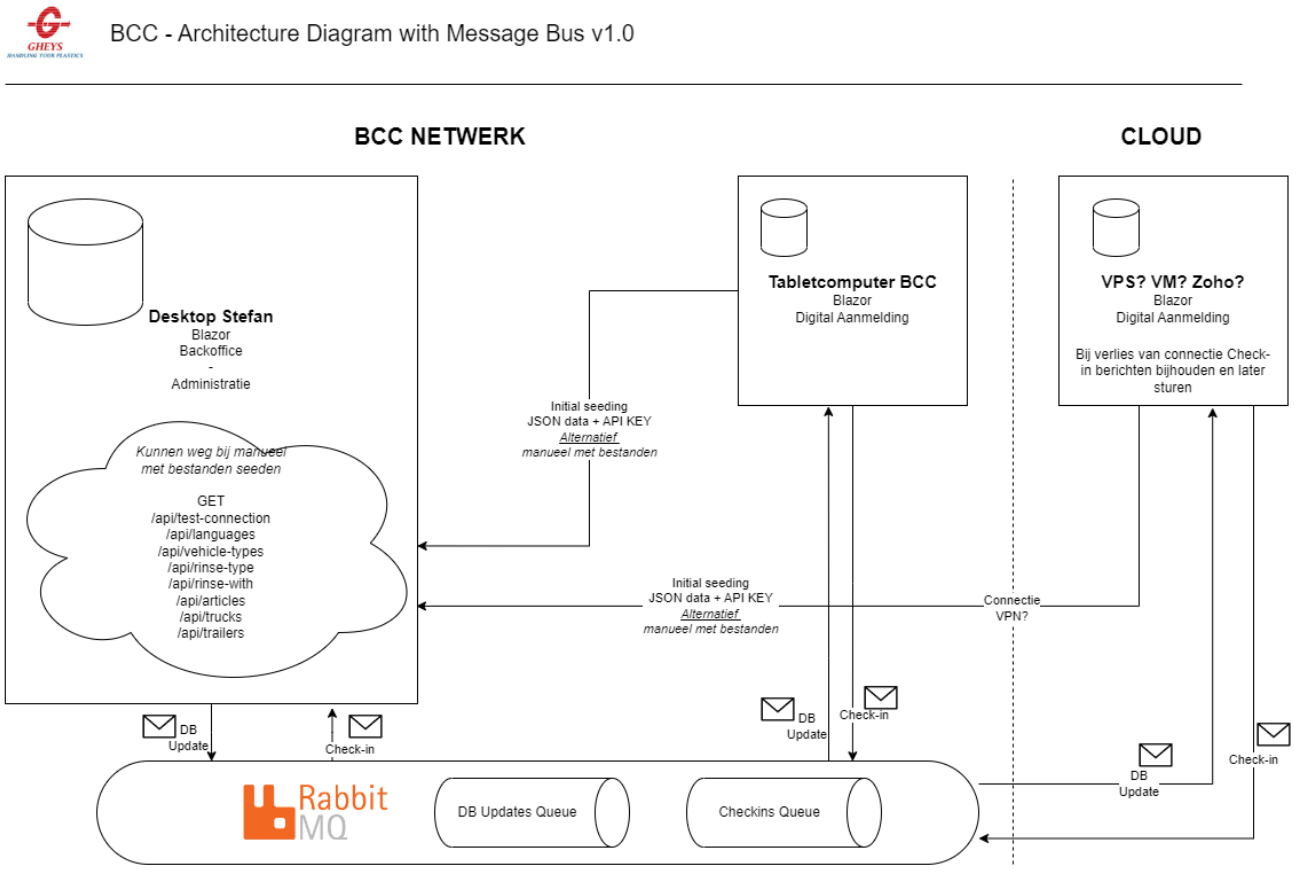
Achteraf had ik ook gezien dat er een design pattern genaamd Outbox Pattern bestaat. Wat gebruikt kan worden voor de notify-new-checkin endpoint opnieuw te callen na dat een verbinden terug hersteld is.



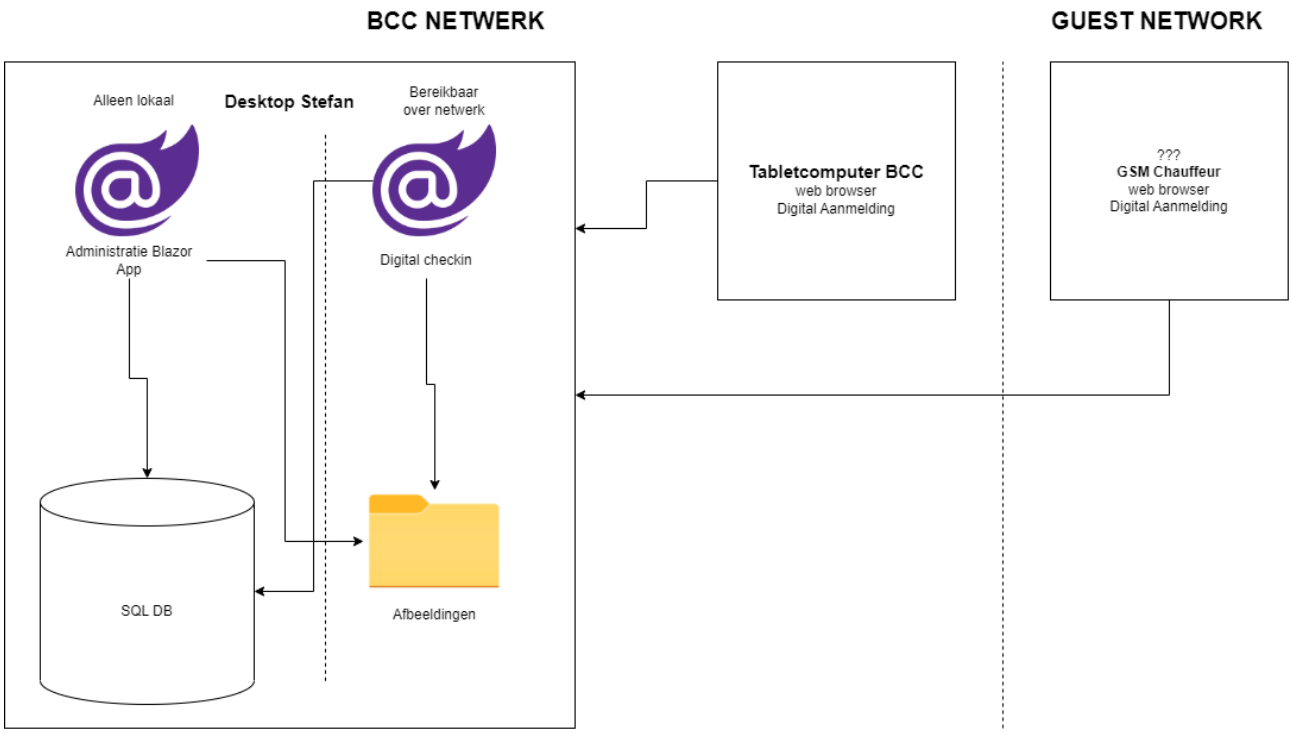
Figuur 7 Architectuur diagram - One DB v1.0



Figuur 8 Architectuur diagram - Separate DB met API v1.0



Figuur 9 Architecture schema - Separate DB with RabbitMQ v1.0



Figuur 10 Architecture schema - All on one device v1.0

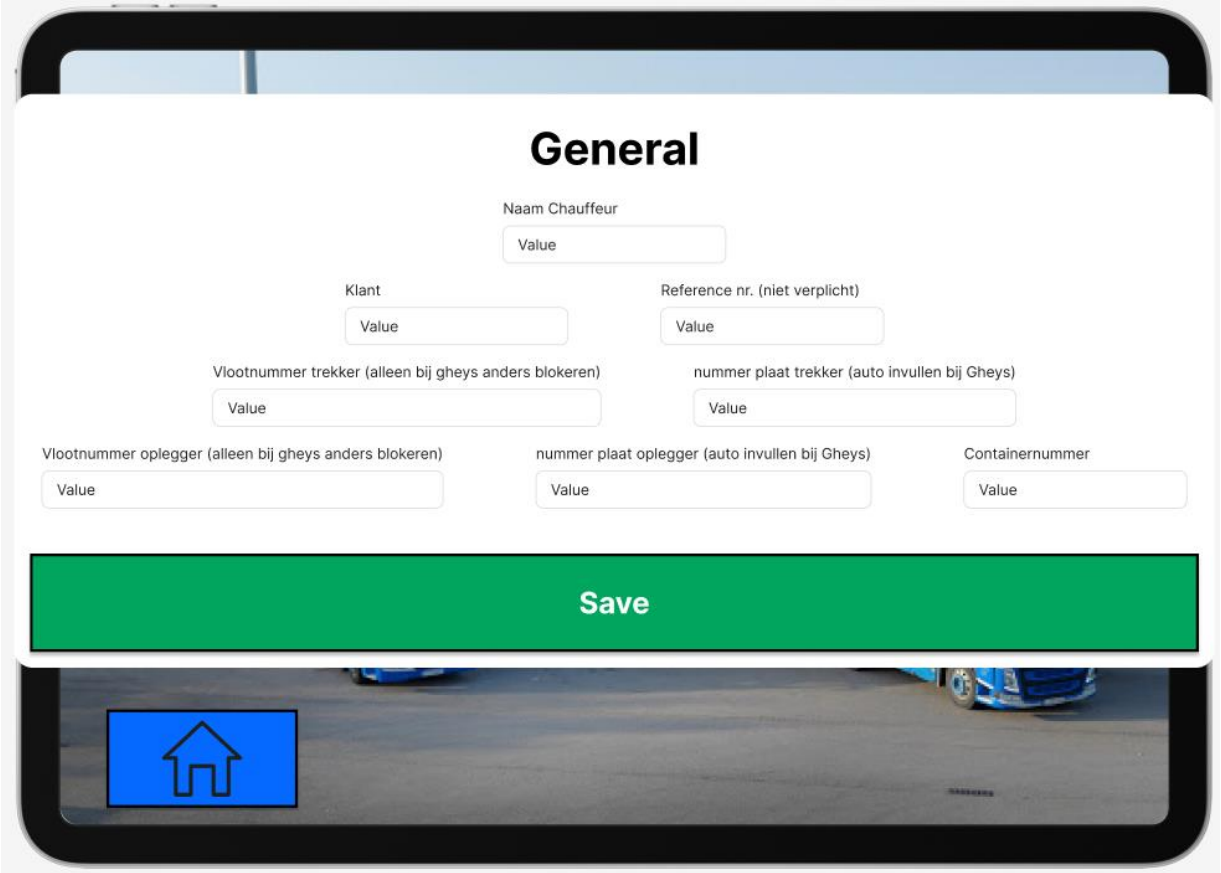
11. Digitale Aanmelding

WIREFRAMES

Online versie: https://www.figma.com/design/Lpm2Mw1kNLvJLVwaz2wrAe/BCC-Zoho-Creator-Design?node-id=0-1&t=j8kbkX79htNHsFXX-1

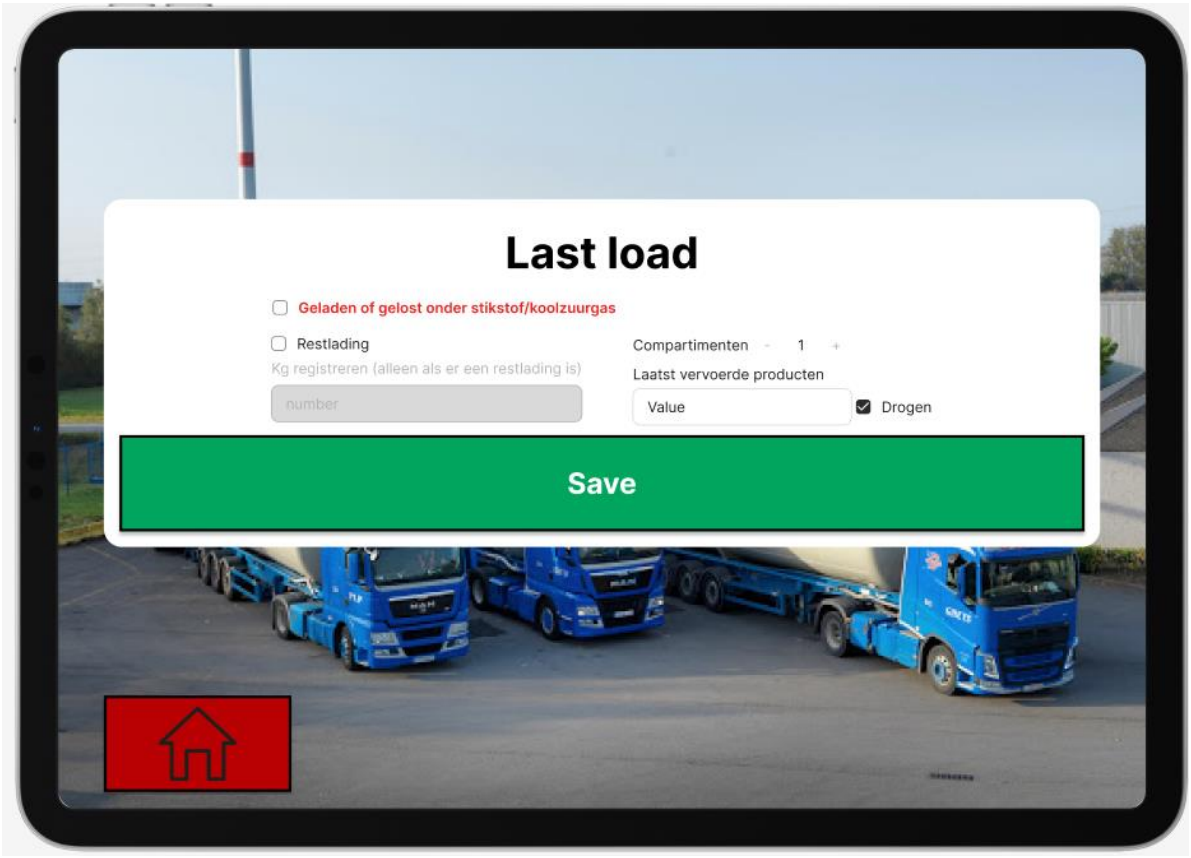
Figuur 11 Digitale aanmelding - Taal kiezen

Figuur 12 Digitale aanmelding - Order Type Keuze

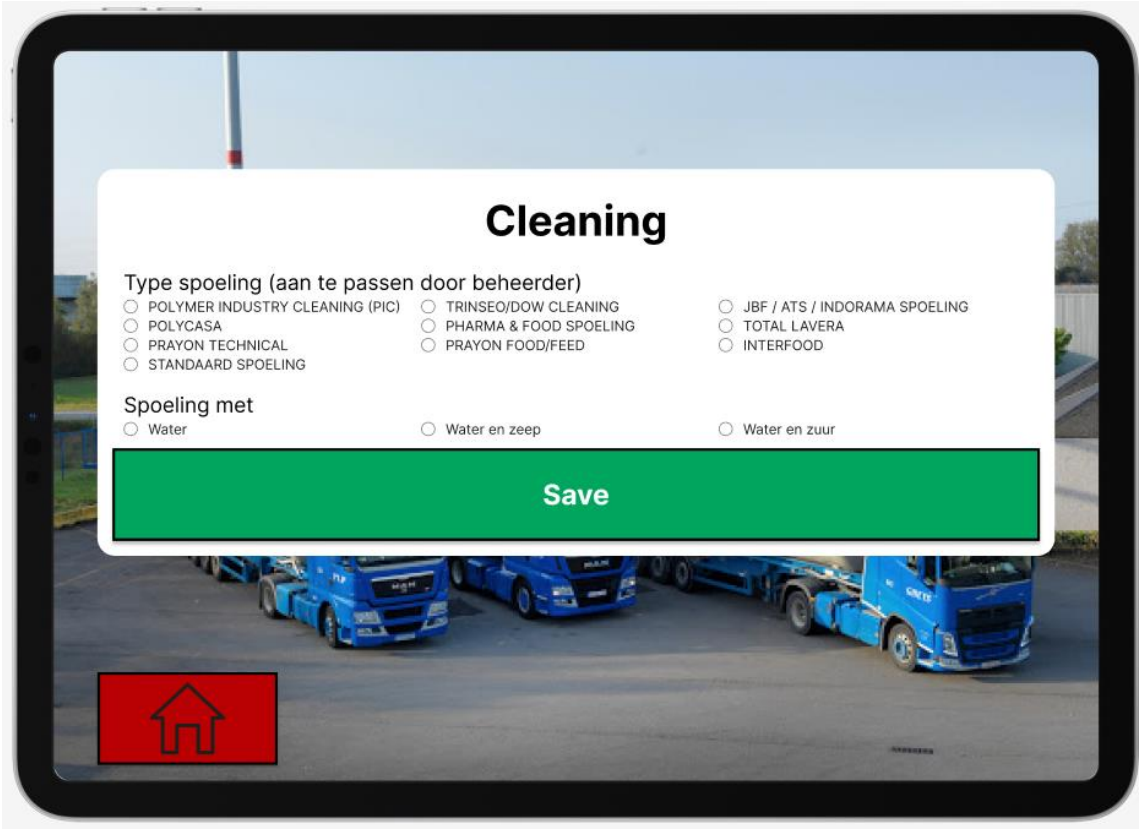


Figuur 13 Digitale aanmelding - Algemene informatie invullen

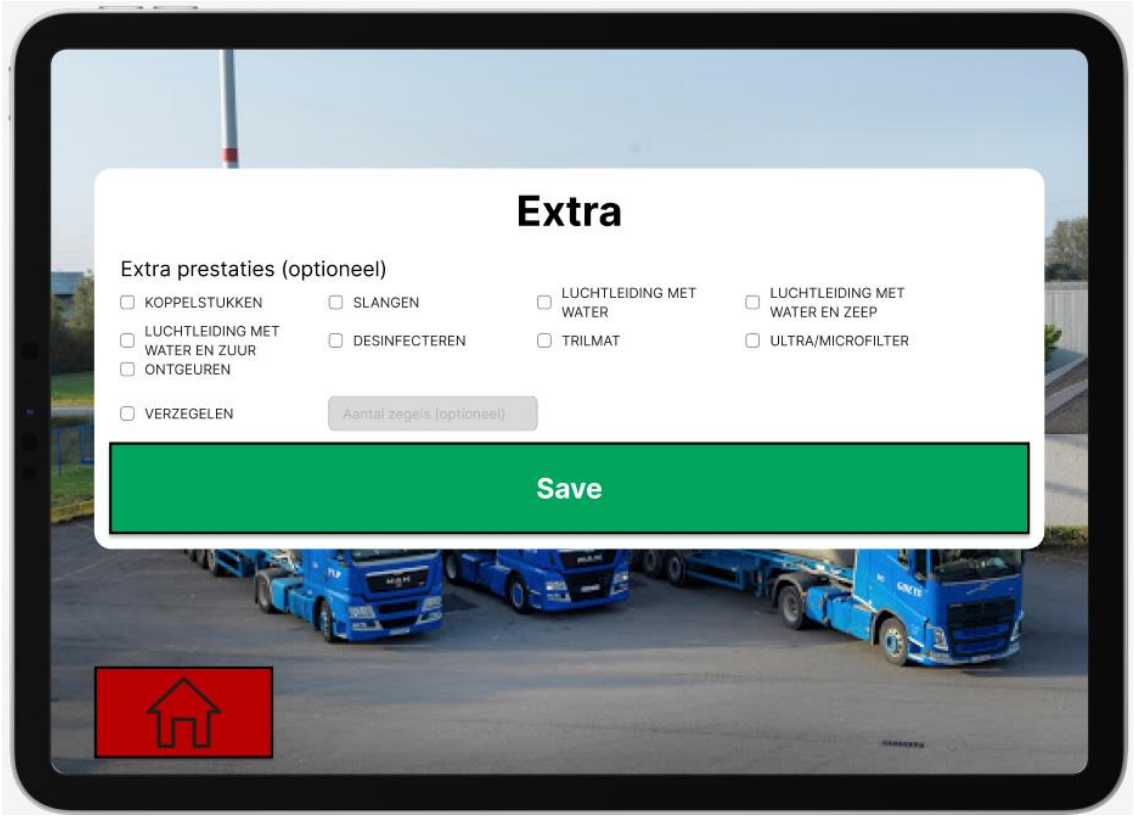
Figuur 14 Digitale aanmelding - Voertuig keuze



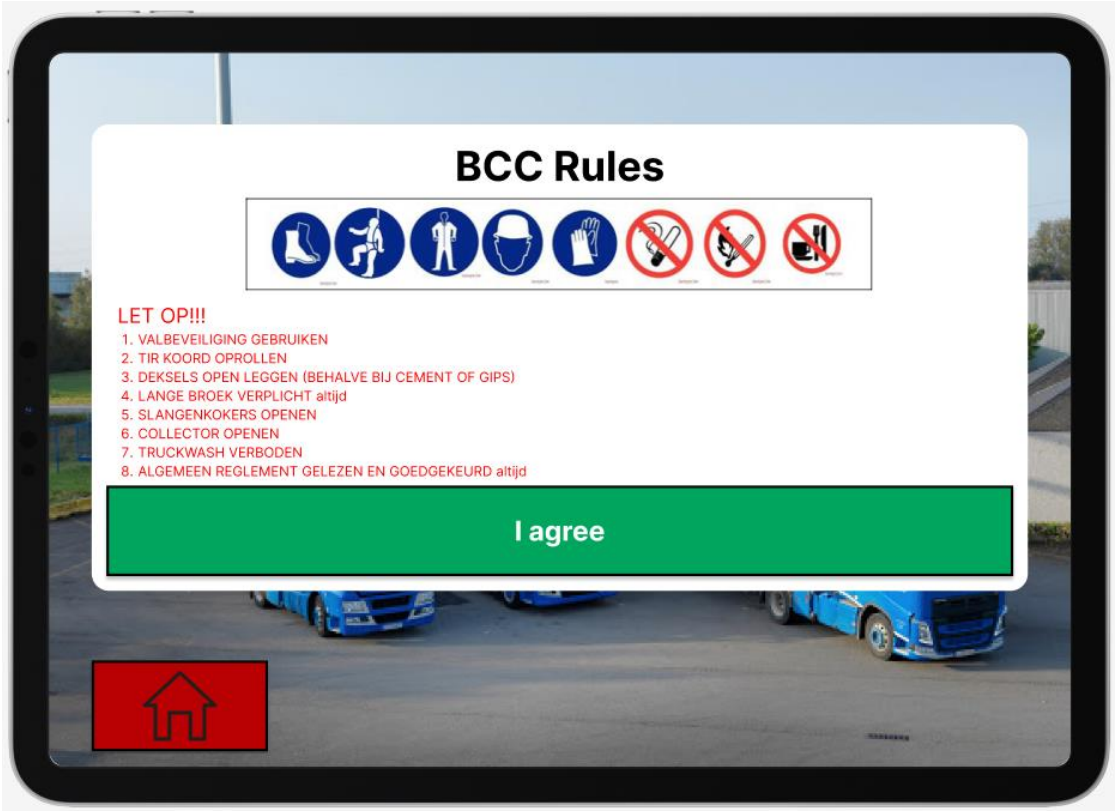
Figuur 15 Digitale aanmelding - Laatste lading formulier



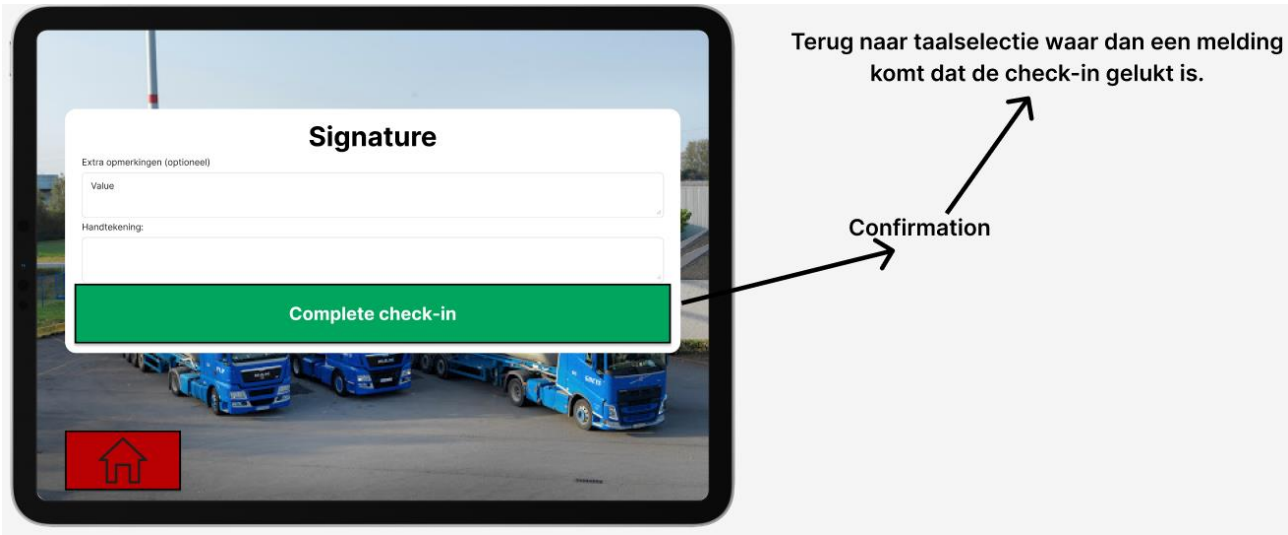
Figuur 16 Digitale aanmelding - Type reiniging formulier



Figuur 17 Digitale aanmelding - Extra prestaties formulier



Figuur 18 Digitale aanmelding - Regels tonen aan chauffeur



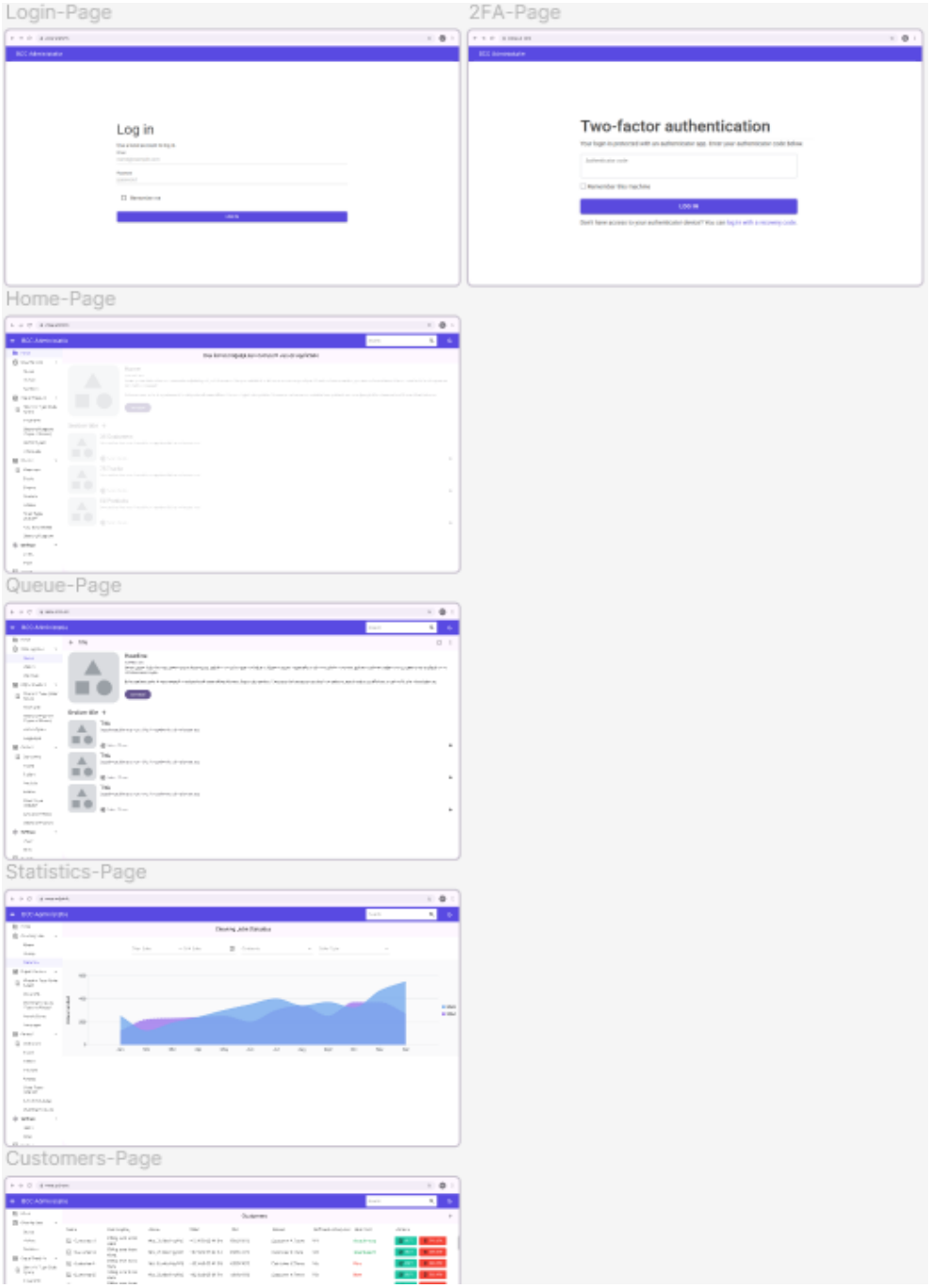
Figuur 19 Digitale aanmelding - Handtekening formulier

BACKOFFICE

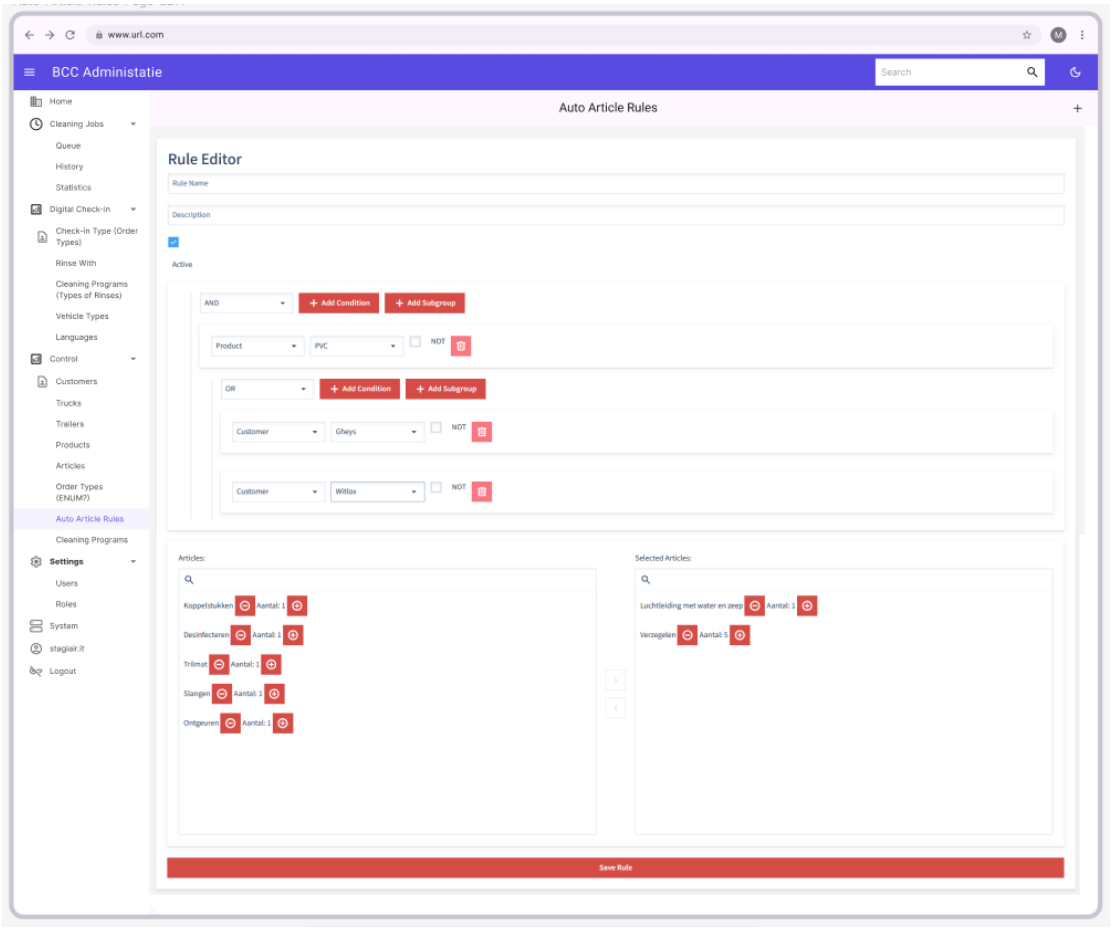
WIREFRAMES

De volgende wireframes zijn minder uitgewerkt dan die van de digitale aanmelding en kunnen dus nog aanzienlijk worden aangepast. Ze omvatten onder andere de basisstructuur voor de CRUD-functionaliteit van formulieropties, zoals: ordertypes, interne richtlijnen, talen, vrachtwagens, opleggers, klanten, producten, artikelen, reinigingsprogramma's en gebruikers.

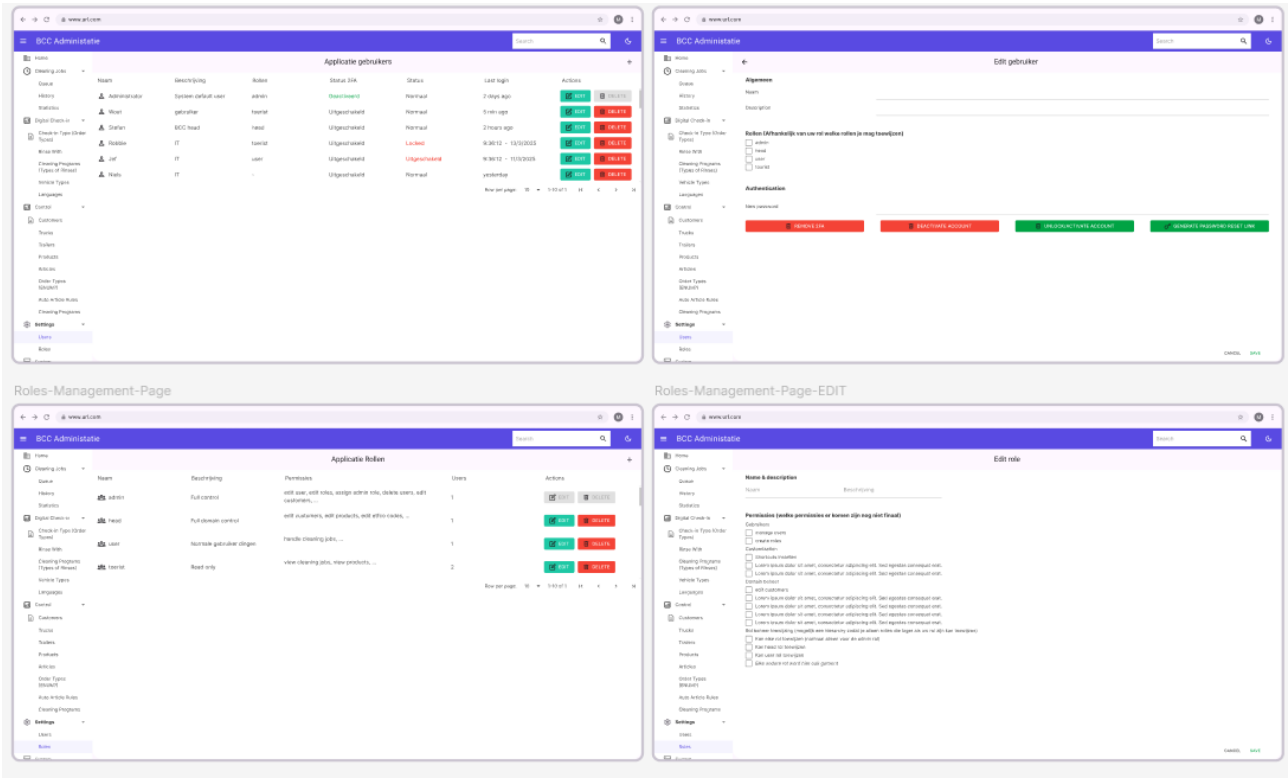
Online versie: https://www.figma.com/design/IbhAtgHTGknX0Ug4BJeinU/BCC-Blazor-Design?node-id=748-18787&t=uRwMlurd3YmQ361U-1



Figuur 20 Wireframe - BCC beheerprogramma - deel 1



Figuur 21 Wireframe - BCC beheerprogramma - automatisch artikels toevoegen



Figuur 22 Wireframe - BCC beheerprogramma - gebruikers en rollen

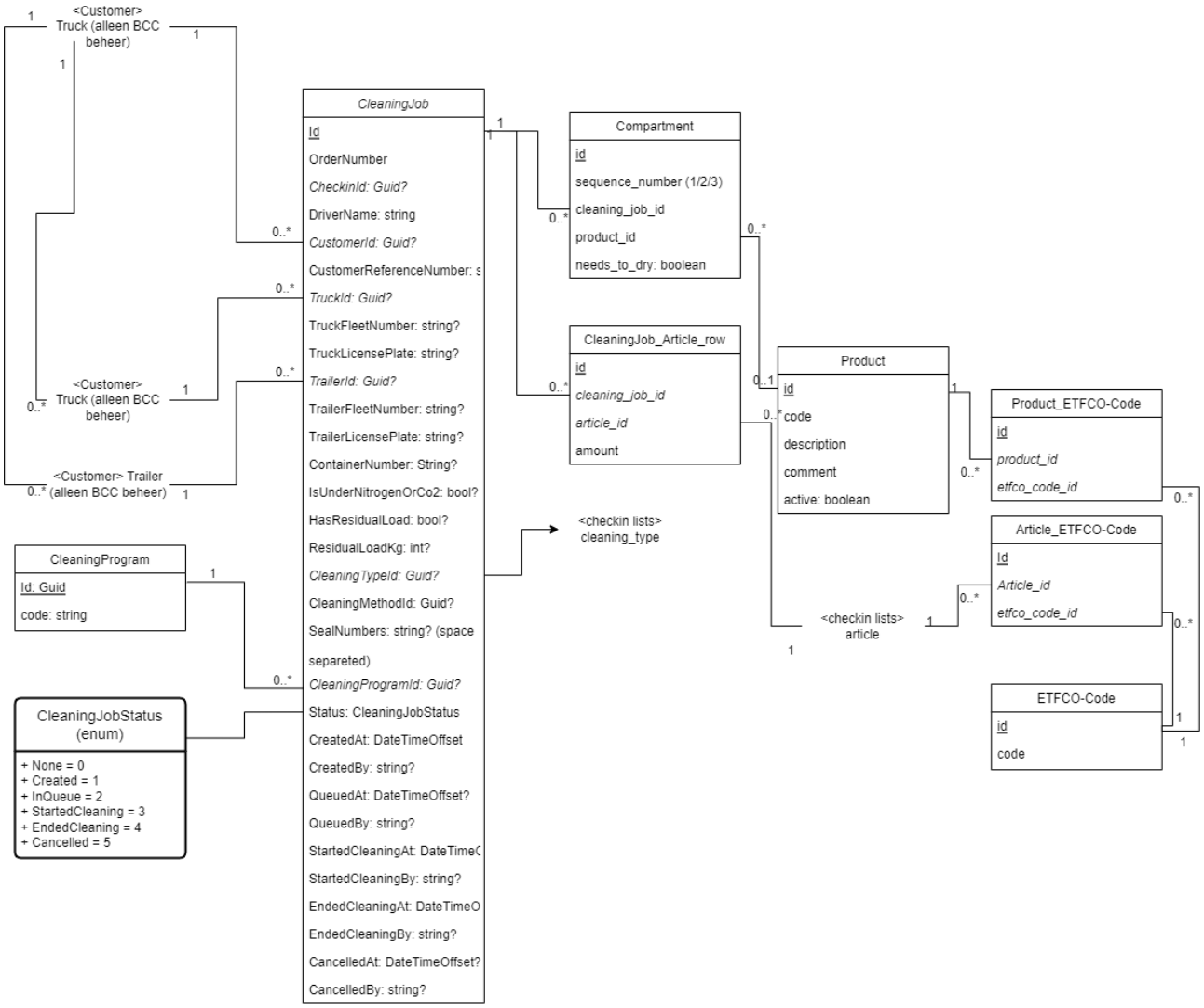
Na het inloggen krijgt de gebruiker toegang tot de digitale aanmeldingspagina, waar alle binnengekomen aanmeldingen kunnen worden bekeken en verwerkt. Wanneer bepaalde gegevens, zoals klant- of productinformatie, niet automatisch worden herkend, moeten deze handmatig worden ingevoerd.

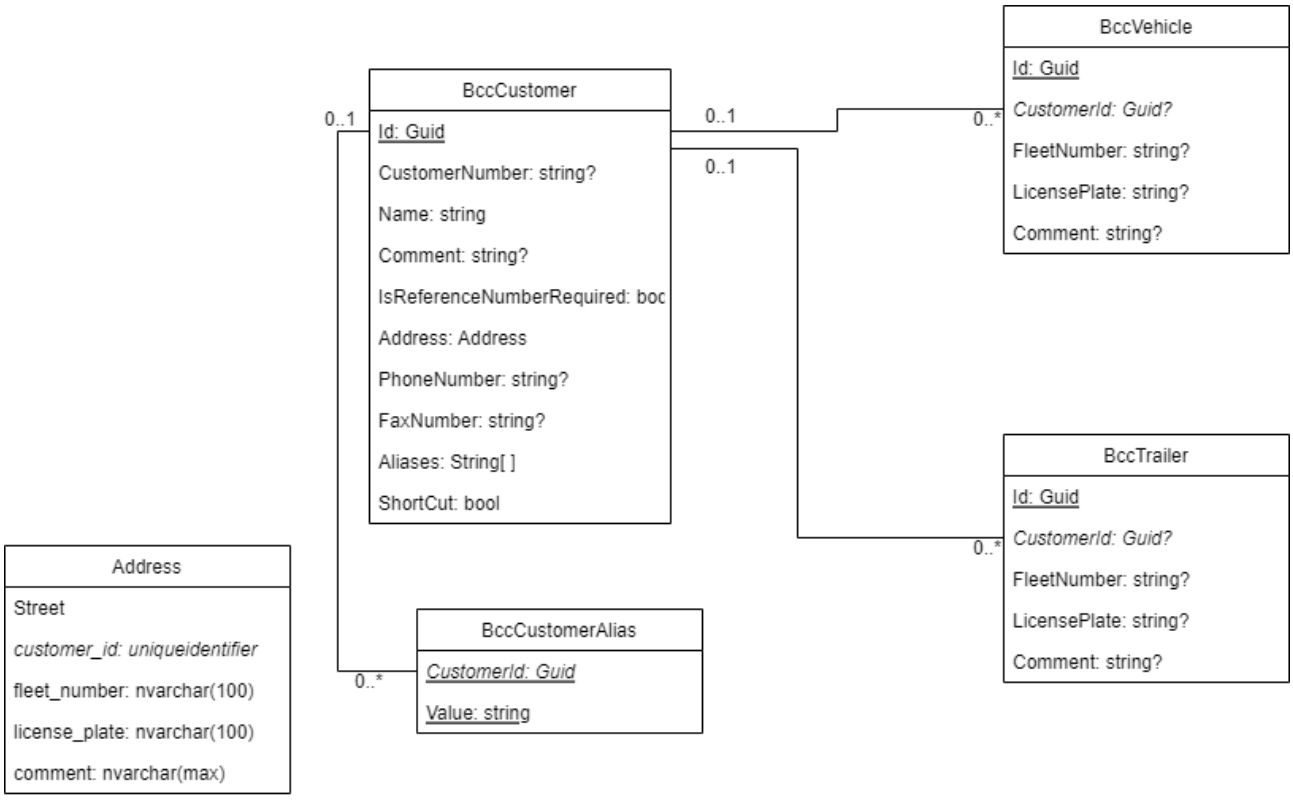
Tijdens de verwerking wordt ook het juiste reinigingsprogramma geselecteerd. Vervolgens wordt de aanmelding gekoppeld aan een order en toegevoegd aan de wachtrij. In deze wachtrij krijgt het order een status toegewezen.

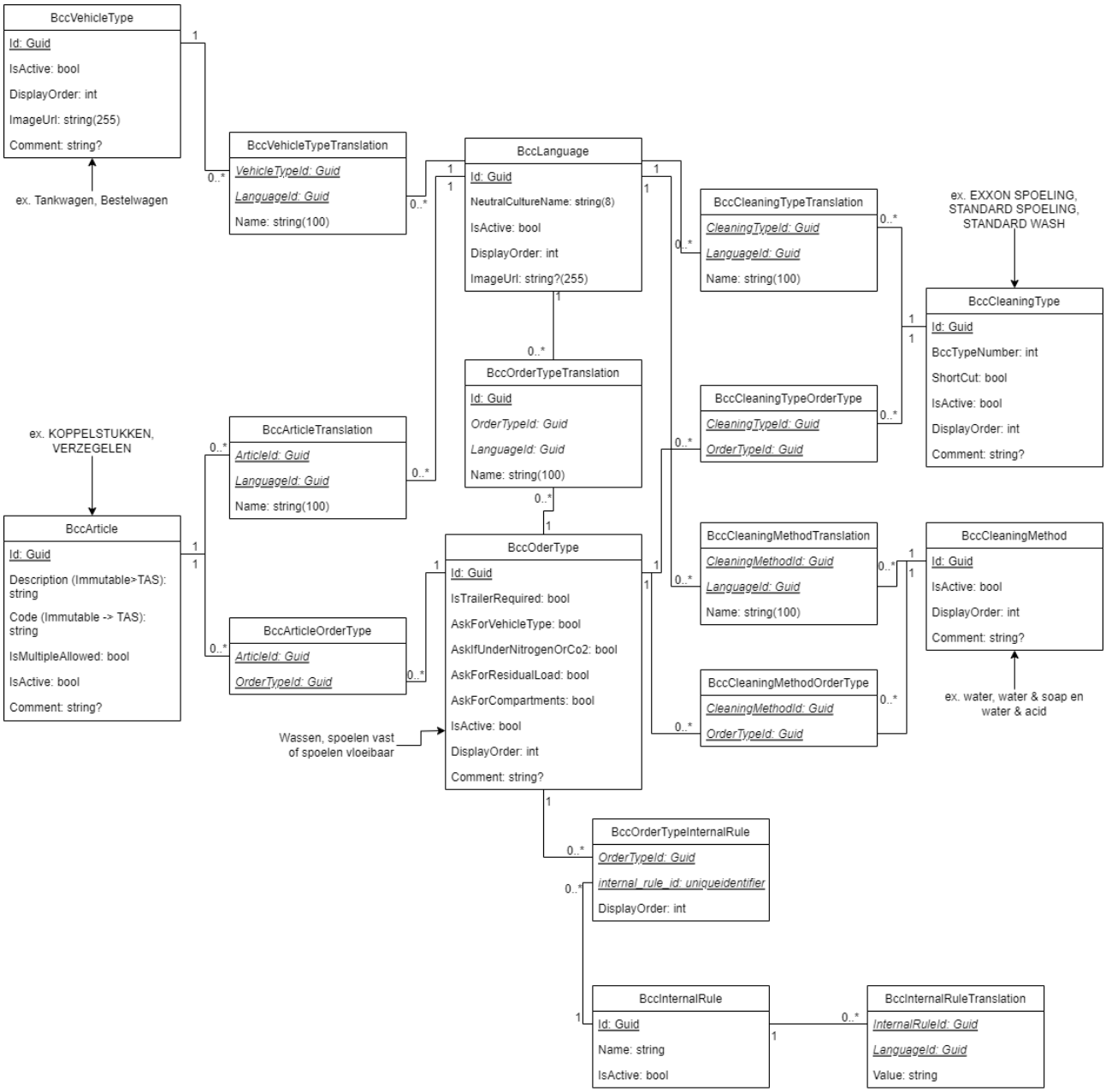
Zodra de spoelmedewerkers met het order starten, kunnen zij dit aangeven door de status op "Bezig" te zetten. Wanneer de reiniging is voltooid, kan de status worden aangepast naar "Klaar".

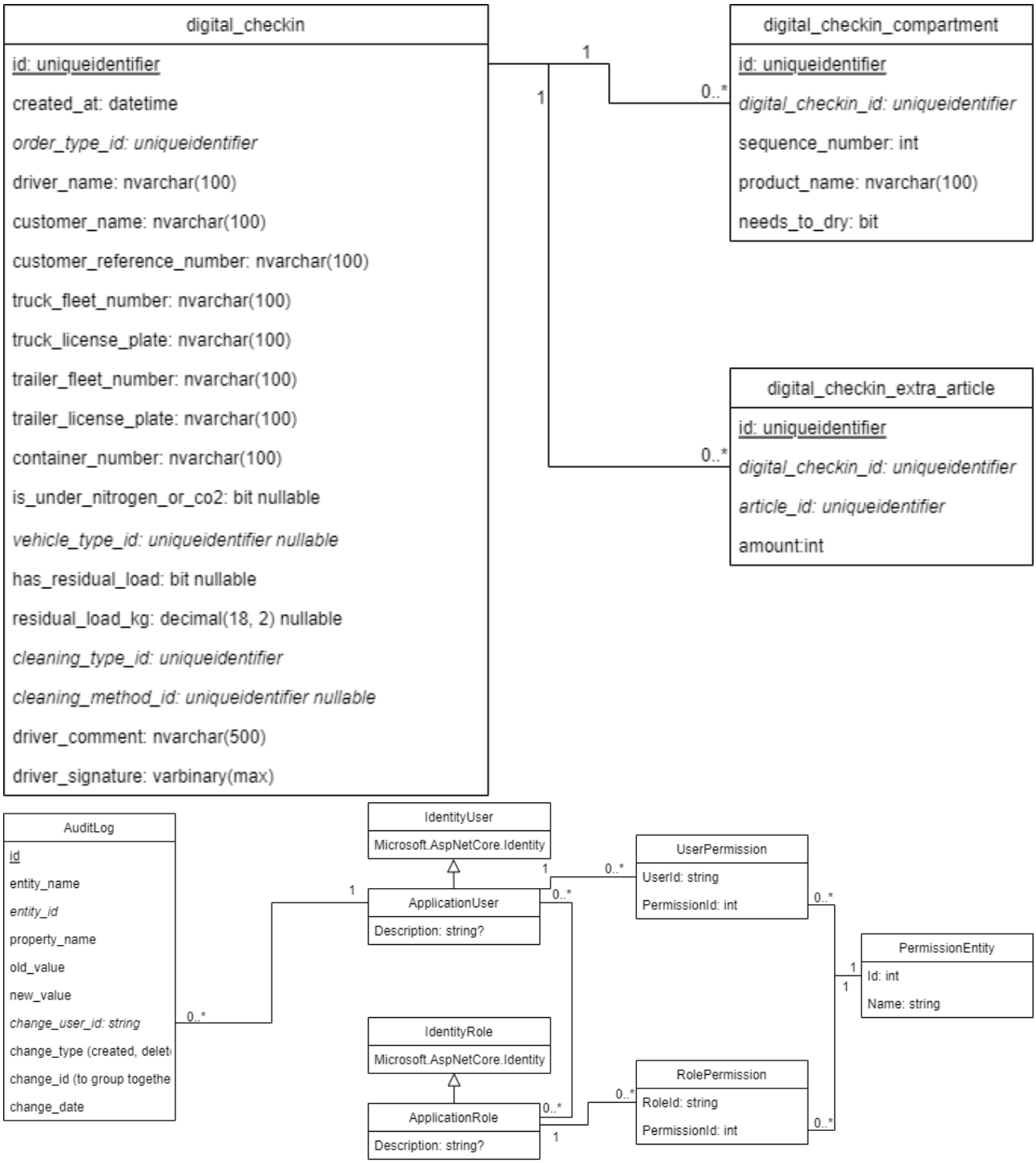
12. Database Schema

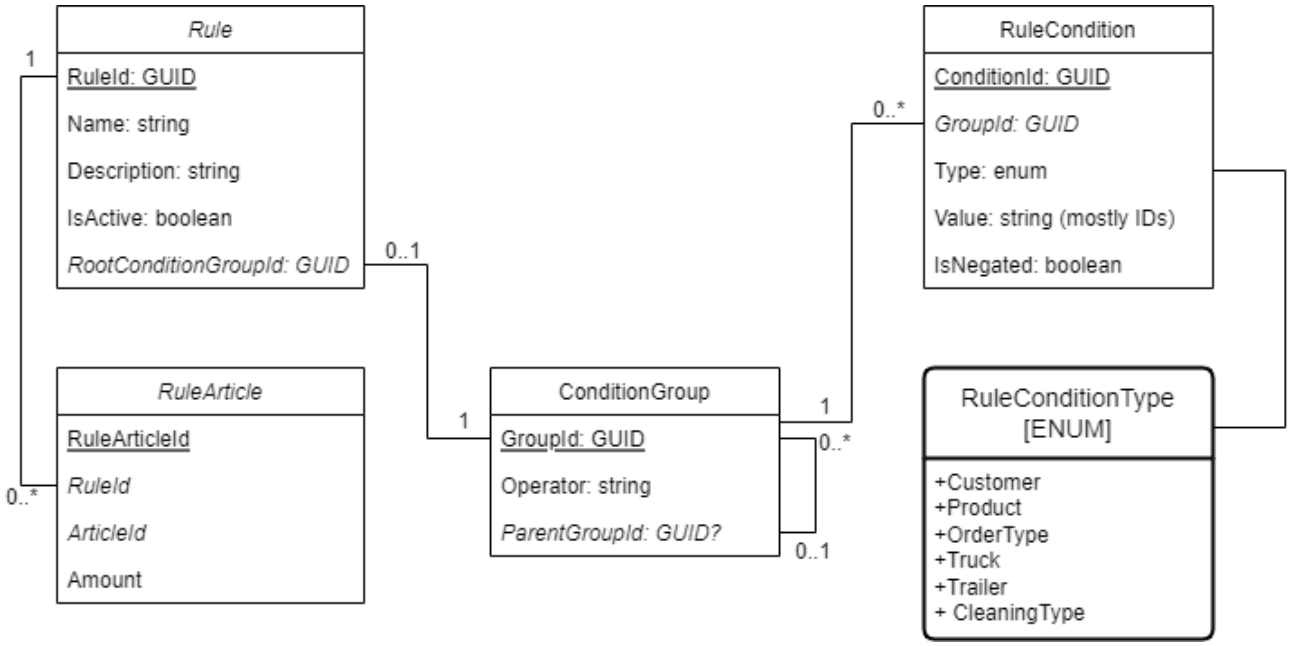
Zie bestand bcc_database_diagram_v1.2.drawio




BccVehicle
Id: Guid
CustomerId: Guid?
FleetNumber: string?
LicensePlate: string?
Comment: string?
BccCustomer
Id: Guid
CustomerNumber: string?
Name: string
Comment: string?
IsReferenceNumberRequired: boo
Address: Address
PhoneNumber: string?
FaxNumber: string?
Aliases: String[ ]
ShortCut: bool
0..1
0..*
0..1
0..1
BccTrailer
Id: Guid
CustomerId: Guid?
FleetNumber: string?
LicensePlate: string?
Comment: string?
0..*
Address
Street
customer_id: uniqueidentifier
fleet_number: nvarchar(100)
license_plate: nvarchar(100)
comment: nvarchar(max)
BccCustomerAlias
CustomerId: Guid
Value: string
0..*


BccVehicleType
Id: Guid
IsActive: bool
DisplayOrder: int
ImageUrl: string(255)
Comment: string?
ex. Tankwagen, Bestelwagen
BccVehicleTypeTranslation
VehicleTypeId: Guid
LanguageId: Guid
Name: string(100)
BccLanguage
Id: Guid
NeutralCultureName: string(8)
IsActive: bool
DisplayOrder: int
ImageUrl: string?(255)
BccCleaningTypeTranslation
CleaningTypeId: Guid
LanguageId: Guid
Name: string(100)
ex. EXXON SPOELING, STANDARD SPOELING, STANDARD WASH
BccCleaningType
Id: Guid
BccTypeNumber: int
ShortCut: bool
IsActive: bool
DisplayOrder: int
Comment: string?
BccOrderTypeTranslation
Id: Guid
OrderTypeId: Guid
LanguageId: Guid
Name: string(100)
BccCleaningTypeOrderType
CleaningTypeId: Guid
OrderTypeId: Guid
ex. KOPPELSTUKKEN, VERZEGELEN
BccArticleTranslation
ArticleId: Guid
LanguageId: Guid
Name: string(100)
BccArticle
Id: Guid
Description (Immutable>TAS): string
Code (Immutable -> TAS): string
IsMultipleAllowed: bool
IsActive: bool
Comment: string?
BccArticleOrderType
ArticleId: Guid
OrderTypeId: Guid
BccOderType
Id: Guid
IsTrailerRequired: bool
AskForVehicleType: bool
AskIfUnderNitrogenOrCo2: bool
AskForResidualLoad: bool
AskForCompartments: bool
IsActive: bool
DisplayOrder: int
Comment: string?
Wassen, spoelen vast of spoelen vloeibaar
BccCleaningMethodTranslation
CleaningMethodId: Guid
LanguageId: Guid
Name: string(100)
BccCleaningMethod
Id: Guid
IsActive: bool
DisplayOrder: int
Comment: string?
ex. water, water & soap en water & acid
BccCleaningMethodOrderType
CleaningMethodId: Guid
OrderTypeId: Guid
BccOrderTypeInternalRule
OrderTypeId: Guid
internal_rule_id: uniqueidentifier
DisplayOrder: int
BccInternalRule
Id: Guid
Name: string
IsActive: bool
BccInternalRuleTranslation
InternalRuleId: Guid
LanguageId: Guid
Value: string


digital_checkin
id: uniqueidentifier
created_at: datetime
order_type_id: uniqueidentifier
driver_name: nvarchar(100)
customer_name: nvarchar(100)
customer_reference_number: nvarchar(100)
truck_fleet_number: nvarchar(100)
truck_license_plate: nvarchar(100)
trailer_fleet_number: nvarchar(100)
trailer_license_plate: nvarchar(100)
container_number: nvarchar(100)
is_under_nitrogen_or_co2: bit nullable
vehicle_type_id: uniqueidentifier nullable
has_residual_load: bit nullable
residual_load_kg: decimal(18, 2) nullable
cleaning_type_id: uniqueidentifier
cleaning_method_id: uniqueidentifier nullable
driver_comment: nvarchar(500)
driver_signature: varbinary(max)
digital_checkin_compartment
id: uniqueidentifier
digital_checkin_id: uniqueidentifier
sequence_number: int
product_name: nvarchar(100)
needs_to_dry: bit
digital_checkin_extra_article
id: uniqueidentifier
digital_checkin_id: uniqueidentifier
article_id: uniqueidentifier
amount:int
1
0..*
1
0..*
AuditLog
id
entity_name
entity_id
property_name
old_value
new_value
change_user_id: string
change_type (created, delet
change_id (to group togethe
change_date
0..*
1
IdentityUser
Microsoft.AspNetCore.Identity
ApplicationUser
Description: string?
1
0..*
0..*
UserPermission
UserId: string
PermissionId: int
0..*
1
1
PermissionEntity
Id: int
Name: string
IdentityRole
Microsoft.AspNetCore.Identity
ApplicationRole
Description: string?
0..*
1
0..*
RolePermission
RoleId: string
PermissionId: int
0..*



Het onderstaande diagram geeft de kernstructuur weer van de **regel-entity**, die wordt gebruikt om bedrijfsregels op een flexibele manier te definiëren en toe te passen:

- **Rule**: Vertegenwoordigt een specifieke regel met attributen zoals naam, beschrijving en activatiestatus.
- **ConditionGroup** & **RuleCondition**: Samen vormen zij de logische opbouw van een regel. Een *ConditionGroup* groepeert meerdere voorwaarden op basis van een logische operator (bijvoorbeeld AND of OR), terwijl een *RuleCondition* een individuele voorwaarde beschrijft, gebaseerd op een specifiek type (zoals klant, product of vrachtwagen).
- **RuleArticle**: Verbindt een regel aan een artikel, inclusief de bijbehorende hoeveelheid.
- **RuleConditionType**: Een enumeratie die de mogelijke types voorwaarden definieert.

13. Besluit

In dit realisatiedocument werd het ontwikkelingsproces van een aanmeld- en administratieprogramma voor de Cleaning-afdeling van Gheys Transport technisch onderbouwd en gedocumenteerd. Doorheen de analyse zijn verschillende component libraries, handtekeningoplossingen en PDF-generatietools onderzocht en vergeleken om tot de meest geschikte technologieën te komen.

Op basis van gebruiksvriendelijkheid, compatibiliteit en functionaliteit is gekozen voor Radzen als component library, Blazor SignaturePad voor digitale handtekeningen en een combinatie van Scriban en PuppeteerSharp voor het genereren van PDF-bestanden. Elk van deze keuzes is onderbouwd met technische criteria en praktische overwegingen, zoals integratiegemak binnen Blazor, flexibiliteit en uitbreidbaarheid.

Tijdens de implementatiefase werd extra aandacht besteed aan lokalisatie, waarbij Microsoft's IStringLocalizer werd ingezet in combinatie met een databasegestuurde aanpak. Deze oplossing bood voldoende controle en schaalbaarheid om meertalige ondersteuning correct te implementeren, in tegenstelling tot de beperkingen die eerder werden ervaren in Zoho Creator.

Verder zijn er tal van tools en frameworks gebruikt om het project te ondersteunen, waaronder Visual Studio, MS SQL Server, Entity Framework Core en Draw.io voor architectuurschema's. Ook werd gebruik gemaakt van AI (ChatGPT) voor ondersteuning bij documentatie en probleemoplossing.

LITERATUURLIJST

Pluralsight. (z.d.). *Pluralsight online learning platform*. https://www.pluralsight.com/
Microsoft. (z.d.). *Microsoft Learn – Documentatie en handleidingen voor ontwikkelaars*. https://learn.microsoft.com/
Scriban. (z.d.). *Scriban GitHub-documentatie*. https://github.com/scriban/scriban/tree/master/doc
PuppeteerSharp. (z.d.). *PuppeteerSharp Documentatie*. https://www.puppeteersharp.com/docs/index.html

Bijlagen