

**Het Blazor Framework**

Steven Kazmierczak

Promotoren:

|  |  |
| --- | --- |
| Vincent Vanbellingen | Benvitec |
| Wouter Meerten | Hogeschool PXL Hasselt |



**Het Blazor Framework**

Steven Kazmierczak

Promotoren:

|  |  |
| --- | --- |
| Vincent Vanbellingen | Benvitec |
| Wouter Meerten | Hogeschool PXL Hasselt |

Dankwoord

Met dit dankwoord wil ik graag mijn oprechte waardering uitspreken aan alle personen die hebben bijgedragen aan het tot stand komen van dit academische paper. Hun onschatbare hulp, begeleiding en steun hebben een cruciale rol gespeeld in mijn onderzoek en het succesvol voltooien van mijn stage.

Allereerst wil ik mijn dankbaarheid uitdrukken aan het personeel van Benvitec, in het bijzonder Vincent Vanbellingen. Ik ben ontzettend dankbaar voor de warme ontvangst die ik heb ontvangen en voor de unieke kans die ik heb gekregen om mijn stage bij Benvitec te voltooien. De samenwerking met het team was buitengewoon waardevol en ik ben dankbaar voor de mogelijkheid om intensief met hen samen te werken tijdens de analyse en tests. Hun expertise en toewijding hebben mijn onderzoek aanzienlijk versterkt en hebben bijgedragen aan de kwaliteit van dit paper.

Ten tweede wil ik mijn dankbaarheid uitspreken aan PXL, mijn hogeschool, voor het bieden van de mogelijkheid om mijn stage uit te voeren en dit academische traject te doorlopen. In het bijzonder wil ik mijn oprechte waardering uitspreken naar mijn hogeschoolpromotor, Wouter Meerten. Zijn continue ondersteuning, begeleiding en inzichten waren van onschatbare waarde gedurende het hele proces. Zijn toewijding aan mijn academische groei en zijn constructieve feedback hebben mijn werk naar een hoger niveau getild.

Tot slot wil ik ook mijn oprechte dank uitspreken naar de taallectoren van mijn hogeschool. Hun zeer gewaardeerde hulp bij het nalezen en corrigeren van mijn paper heeft ervoor gezorgd dat de kwaliteit van mijn schrijven verbeterd is en dat de boodschap helder en beknopt overkomt. Ik ben dankbaar voor hun toewijding en professionele ondersteuning.

Dit academische paper zou niet tot stand zijn gekomen zonder de onschatbare bijdragen en betrokkenheid van deze individuen en organisaties. Ik ben vereerd om met hen te hebben mogen samenwerken en ben hen diep dankbaar voor hun waardevolle inbreng in mijn academische ontwikkeling.

Abstract

Benvitec is een bedrijf uit Beringen dat onder andere in kunststofverwerking actief is. Het bedrijf heeft nood aan een webapplicatie die kan berekenen hoeveel items er uit een metaal- of kunststofplaat versneden kunnen worden. Dit zal voornamelijk gebruikt worden bij het opstellen van offertes, maar kan bijvoorbeeld ook gebruikt worden als blauwdruk.

Er was een duidelijke voorkeur voor C# als programmeertaal omwille van de reeds aanwezige kennis bij het personeel. Dit beperkte de keuze van framework tot Blazor of ASP.NET Core MVC. De applicatie die ontwikkeld moet worden, wisselt voornamelijk informatie uit tussen front- en backend waardoor ASP.NET Core MVC minder geschikt is. Dit framework herlaadt namelijk de volledige pagina bij het uitvoeren van een actie, tenzij veelvuldig gebruik wordt gemaakt van JavaScript. Blazor daarentegen kan segmenten herladen zonder gebruik van JavaScript. Daarom was dit een logische keuze.

Benvitec heeft zijn behoefte uitgedrukt voor een webapplicatie. Toch biedt Blazor de mogelijkheid aan om over te schakelen naar een mobiele of native applicatie. Dit garandeert een toename in flexibiliteit indien er in de toekomst wijzigingen zijn ten opzichte van de oorspronkelijke behoefte.

De doelstelling van deze onderzoeksopdracht is een grondige en intensieve verkenning van de werking van Blazor. De voor- en nadelen van dit framework worden behandeld. Er wordt afgetoetst in welke mate het toepassen van JavaScript vermeden kan worden. Ook de verschillende architecturen binnen Blazor komen aan bod. Zo wordt er een vergelijking gemaakt tussen "Blazor Server" en "Blazor WebAssembly". Bij “Blazor Server” wordt alle logica op de server verwerkt, in tegenstelling tot “Blazor WebAssembly”, waar de code client-side wordt uitgevoerd. Uit het resultaat van het onderzoek kan er beoordeeld worden hoe volwassen het Blazor-framework is

Inhoudsopgave

[Dankwoord ii](#_Toc137576154)

[Abstract iii](#_Toc137576155)

[Inhoudsopgave iv](#_Toc137576156)

[Lijst van gebruikte figuren vi](#_Toc137576157)

[Lijst van gebruikte afkortingen vii](#_Toc137576158)

[Inleiding 1](#_Toc137576159)

[I. Stageverslag 2](#_Toc137576160)

[1 Bedrijfsvoorstelling 2](#_Toc137576161)

[2 Voorstelling stageopdracht 2](#_Toc137576162)

[2.1 Probleemstelling 2](#_Toc137576163)

[2.2 Doelstellingen 2](#_Toc137576164)

[2.3 Omgeving 2](#_Toc137576165)

[3 Uitwerking stageopdracht 3](#_Toc137576166)

[3.1 Analyse 3](#_Toc137576167)

[3.1.1 Vereisten 3](#_Toc137576168)

[3.1.2 Data ingaven 3](#_Toc137576169)

[3.1.3 Instellingen en offerte 3](#_Toc137576170)

[3.1.4 Plan berekenen 3](#_Toc137576171)

[3.1.1 Opslaan van een optimalisatieplan 7](#_Toc137576172)

[3.1.2 UI 8](#_Toc137576173)

[3.2 Werkwijze 8](#_Toc137576174)

[3.3 Resultaat 8](#_Toc137576175)

[3.3.1 Instellingen 8](#_Toc137576176)

[3.3.2 Offerte 9](#_Toc137576177)

[3.3.3 Materialen en items 9](#_Toc137576178)

[3.3.4 Optimalisatieplan 10](#_Toc137576179)

[3.3.5 API 12](#_Toc137576180)

[3.4 Reflectie 14](#_Toc137576181)

[II. Onderzoekstopic 15](#_Toc137576182)

[1 Onderzoeksvraag 15](#_Toc137576183)

[1.1 Is het Blazor framework volwassen genoeg? 15](#_Toc137576184)

[2 Onderzoeksmethode 15](#_Toc137576185)

[3 Uitwerking 15](#_Toc137576186)

[3.1 Literatuurstudie 15](#_Toc137576187)

[3.1.1 Wat is Blazor? [4] 15](#_Toc137576188)

[3.1.2 Blazor Server [6] 16](#_Toc137576189)

[3.1.3 Blazor WebAssembly [6] 16](#_Toc137576190)

[3.1.4 Blazor Hybrid [6] 17](#_Toc137576191)

[3.1.5 JavaScript Interoperability [7] 17](#_Toc137576192)

[3.1.6 Razor [8] 17](#_Toc137576193)

[3.1.7 Projectstructuur Blazor 18](#_Toc137576194)

[3.2 Implementatie 19](#_Toc137576195)

[3.2.1 Razor Component 19](#_Toc137576196)

[3.2.2 Variabele en functies gebruiken 19](#_Toc137576197)

[3.2.3 Razor Component verdiepen 20](#_Toc137576198)

[3.2.3.1 Hergebruiken 20](#_Toc137576199)

[3.2.3.2 Variabele doorgeven 21](#_Toc137576200)

[3.2.3.3 Functies doorgeven 22](#_Toc137576201)

[3.2.3.4 EventCallback 23](#_Toc137576202)

[3.2.4 Form & Data binding [10] 24](#_Toc137576203)

[3.2.5 Validatie [10] 26](#_Toc137576204)

[3.2.6 Dependency Injection (DI) [11] 28](#_Toc137576205)

[3.2.7 JavaScript Interoperability 29](#_Toc137576206)

[3.2.7.1 Hoe wordt JavaScript gebruikt vanuit C# 30](#_Toc137576207)

[3.2.7.2 Hoe wordt C# gebruikt in JavaScript 30](#_Toc137576208)

[Conclusie 31](#_Toc137576209)

[Reflectie 32](#_Toc137576210)

Lijst van gebruikte figuren

[Figuur 1: Voorbeeld RestResultType Horizontal [2] 4](#_Toc137576211)

[Figuur 2: Voorbeeld RestResultType HorizontalRotated [2] 4](#_Toc137576212)

[Figuur 3: Voorbeeld RestResultType Vertical [2] 4](#_Toc137576213)

[Figuur 4: Voorbeeld RestResultType VerticalRotate [2] 5](#_Toc137576214)

[Figuur 5: RestResult Model [2] 5](#_Toc137576215)

[Figuur 6: Verticale snijlijn [2] 6](#_Toc137576216)

[Figuur 7: Horizontale snijlijn [2] 6](#_Toc137576217)

[Figuur 8: Sorteren en berekenen [2] 7](#_Toc137576218)

[Figuur 9: Optimalisatieplan data [2] 7](#_Toc137576219)

[Figuur 10: Cutting Optimizer – Instellingen [2] 8](#_Toc137576220)

[Figuur 11: Cutting Optimizer - Local Storage [2] 9](#_Toc137576221)

[Figuur 12: Cutting Optimizer – Offerte [2] 9](#_Toc137576222)

[Figuur 13: Cutting Optimizer – Materialen & Items [2] 10](#_Toc137576223)

[Figuur 14: Cutting Optimizer – Overzicht [2] 10](#_Toc137576224)

[Figuur 15: Cutting Optimizer –Plannen overzicht [2] 11](#_Toc137576225)

[Figuur 16: Cutting Optimizer - Plan overzicht deel één [2] 11](#_Toc137576226)

[Figuur 17: Cutting Optimizer - Plan overzicht deel twee [2] 12](#_Toc137576227)

[Figuur 18: Voorbeeld API [2] 12](#_Toc137576228)

[Figuur 19: Voorbeeld post plate [2] 13](#_Toc137576229)

[Figuur 20: Blazor Future [5] 16](#_Toc137576230)

[Figuur 21: Projectstructuur [9] 18](file:///C:\Users\Dtc-Xtreme\Downloads\Eindwerk_Kazmierczak_Steven.docx#_Toc137576231)

[Figuur 22: Blazor-component [9] 19](#_Toc137576232)

[Figuur 23: Voorbeeld variabele en functies [9] 19](#_Toc137576233)

[Figuur 24: Variable en functies [9] 20](#_Toc137576234)

[Figuur 25: Razor-component gebruiken [9] 20](#_Toc137576235)

[Figuur 26: Voorbeeld Razor-component gebruiken 21](#_Toc137576236)

[Figuur 27: Variabele doorgeven aan component [9] 21](#_Toc137576237)

[Figuur 28: Variabele ontvangen in component [9] 21](#_Toc137576238)

[Figuur 29: functies doorgeven aan component [9] 22](#_Toc137576239)

[Figuur 30: Functie ontvangen in component [9] 22](#_Toc137576240)

[Figuur 31: EventCallback doorgeven aan component [9] 23](#_Toc137576241)

[Figuur 32: EventCallback ontvangen in component [9] 23](#_Toc137576242)

[Figuur 33: InputFormComponent [9] 25](#_Toc137576243)

[Figuur 34: Person model [9] 26](#_Toc137576244)

[Figuur 35: Aangepaste validatie [9] 27](#_Toc137576245)

[Figuur 36: Voorbeeld form validatie 27](#_Toc137576246)

[Figuur 37: Service registreren [9] 28](#_Toc137576247)

[Figuur 38: Di injecteren en gebruiken [9] 29](#_Toc137576248)

[Figuur 39: JavaScript bestand toevoegen [9] 29](#_Toc137576249)

[Figuur 40: JavaScript-code [9] 30](#_Toc137576250)

[Figuur 41: JavaScript uitvoeren adv. C# [9] 30](#_Toc137576251)

Lijst van gebruikte afkortingen

.NET : Microsoft .NET-framework  
API : Application programming interface  
ASP : Active Server Pages  
C# : C-Sharp  
CSS : Cascading Style Sheets  
DI : Dependency Injection  
DOM : Document Object Model  
HTML : HyperText Markup Language  
HTTPS : HyperText Transfer Protocol Secure  
IIS : Internet Information Services  
JS : JavaScript  
JSON : JavaScript Object Notation  
MVC : Model-view-controller  
REST : Representational State Transfer  
UI : User Interface

Inleiding

Blazor is een opkomend framework dat steeds meer aandacht krijgt in de wereld van webontwikkeling. Het is een moderne technologie die het mogelijk maakt om interactieve en dynamische webapplicaties te bouwen met behulp van .NET en C#. Dit framework maakt gebruik van WebAssembly, een binair executieformaat voor webapplicaties, waardoor de code direct in de browser kan worden uitgevoerd.

Het doel van dit onderzoek is om een grondige analyse uit te voeren van Blazor en de relevante aspecten ervan te onderzoeken. Hierbij zullen zowel de technische als de praktische implicaties van het gebruik van Blazor worden belicht. Door de resultaten van dit onderzoek kunnen ontwikkelaars en organisaties een beter inzicht krijgen in de mogelijkheden van Blazor en weloverwogen beslissingen nemen bij het implementeren van dit framework in hun projecten.

1. Stageverslag

# Bedrijfsvoorstelling

Benvitec is een bedrijf dat opgericht is 1992. Het bedrijf is momenteel actief in verschillende sectoren waaronder fire protection, kunststofbewerking, luchtzuivering, linnings, service & maintenance, piping en solutions in plastics. Het bedrijf is gevestigd in de Koolmijnlaan 185 te Beringen.

Innovatie staat centraal bij Benvitec. Ze stellen zich voor met de volgende verwoording:

“We zouden kunnen zeggen dat we elke dag in de wondere wereld van industriële piping en plastics werken. Dat we al drie decennia lang milieu-oplossingen bedenken. Grensverleggend onder de grond en gedurfd boven de grond.  
De waarheid is echter dat wij niet gewoon komen werken. Wij doen dit omdat we elke dag dieper willen graven, schouder aan schouder, op zoek naar de juiste oplossingen. En als één kollectiev van bescheiden experts verklaren we hier en nu dat we elke uitdaging trots zullen aangaan op onze eigen manier.  
De Benvitec manier.” [1]

# Voorstelling stageopdracht

## Probleemstelling

De huidige applicaties die momenteel door Benvitec worden gebruikt, zullen binnenkort niet langer beschikbaar zijn. De ondersteuning voor deze applicaties nadert het einde van hun levenscyclus. Deze applicaties voorzien het bedrijf van essentiële informatie om te bepalen hoeveel items uit een plaat versneden kunnen worden. Deze gegevens worden vervolgens gebruikt bij het opstellen van offertes. Vanwege het naderende einde van de ondersteuning voor de huidige software, is het bedrijf op dit moment op zoek naar alternatieve oplossingen.

## Doelstellingen

Er is een noodzaak voor de ontwikkeling van een webapplicatie in C#, die lokaal gehost kan worden op een IIS-webserver. Het is essentieel dat de code van de webapplicatie gemakkelijk te onderhouden is. De hoofdfunctie van deze applicatie is het genereren van optimalisatieplannen op basis van de door de eindgebruiker ingevoerde gegevens. Deze gegevens omvatten onder andere de zaagbladdikte, de afmetingen van de platen en de gewenste items. Bovendien is er specifiek gevraagd om een functionaliteit waarbij rekening wordt gehouden met fineer, zodat platen en items niet worden gedraaid en de nerven altijd in de juiste richting blijven.

Belangrijke extra informatie die niet mag ontbreken, betreft de hoeveelheid materiaalverlies door het snijproces, het aantal snijlijnen, de identificatie van items op elke specifieke plaat en de hoeveelheid restmateriaal. Daarnaast is er behoefte aan een overzicht waarin het aantal benodigde platen wordt weergegeven. Het gegenereerde optimalisatieplan moet ook kunnen worden opgeslagen, zodat de informatie op een later tijdstip kan worden opgevraagd.

## Omgeving

Voor de ontwikkeling van de applicatie is gekozen voor het gebruik van het Blazor Server framework in combinatie met Bootstrap voor styling, evenals HTML en CSS. Het gebruik van JavaScript zal tot een minimum worden beperkt, aangezien het vereiste was om het gebruik van andere programmeertalen zoveel mogelijk te vermijden. Benvitec heeft een voorkeur uitgesproken voor C#, vanwege de interne kennis binnen het bedrijf. Daarnaast is besloten om naast de Blazor-webapplicatie ook een REST API te implementeren. Als webserver wordt de IIS-webserver gebruikt, die toegankelijk is via HTTPS voor zowel de applicatie als de API. Uiteraard wordt er ook gebruikgemaakt van een versiecontrolesysteem, namelijk Git, dat beschikbaar is via Microsoft DevOps. Op dit platform wordt ook gebruikgemaakt van het Kanban-bord om de voortgang te bewaken en eventuele aanpassingen door te voeren wanneer er beperkte ruimte is voor alle gevraagde opties.

# Uitwerking stageopdracht

## Analyse

Gedurende de eerste week van het project is er in samenspraak met de stagepromotor gewerkt aan het vaststellen van de projectvereisten en het bepalen van de haalbare scope binnen de gegeven tijd. Vervolgens is er gelegenheid geboden om kennis te maken met de huidige applicatie, verantwoordelijk voor het snijden van platen. Dit werd gezamenlijk gedaan met het personeel dat dagelijks met de applicatie werkt, zowel de medewerkers die offertes opstellen als degenen die werkzaam zijn in de zagenrij.

### Vereisten

Er dienen vier vereisten te worden vervuld. Het systeem moet ten eerste in staat zijn om zowel de afmetingen van platen als items in te voeren, inclusief de dikte van het zaagblad. Daarnaast moet het systeem de mogelijkheid bieden om fineer toe te passen, met name bij materialen die een nervenstructuur hebben. Verder moet er een functionaliteit aanwezig zijn waarmee prioriteiten kunnen worden toegekend aan elke plaat op basis van deze gegevens, zodat een optimalisatieplan gegenereerd kan worden dat de oppervlakte zo efficiënt mogelijk benut. Tevens moet het systeem de optie bieden om een bepaalde plaat als standaard te markeren, zodat deze gegevens hergebruikt kunnen worden indien er geen ruimte meer is op andere platen.

### Data ingaven

Een substantieel deel van de gegevens wordt opgeslagen en opgehaald via een Representational State Transfer (REST) API. In de komende paragrafen zal nader worden ingegaan op dit onderwerp.

* Zaagblad (naam, zaagsnede)
* Platen (naam, aantal, lengte, breedte, dikte, trim, fineer, prioriteit en standaardplaat)
* Items (naam, aantal, lengte, breedte, dikte)

### Instellingen en offerte

Tijdens de ontwikkeling van de applicatie werd het idee naar voren gebracht om bepaalde instellingen te onthouden, zodat deze niet telkens opnieuw ingevoerd hoefden te worden. Hierbij werd gedacht aan het onthouden van instellingen zoals het zaagblad, fineer en initialen. Bovendien moest het mogelijk zijn om een offertenummer te koppelen aan het gegenereerde optimalisatieplan.

### Plan berekenen

De berekening van de optimale oppervlakte is een complex vraagstuk dat gepaard gaat met aanzienlijke uitdagingen. Er bestaat een verschil tussen het maximaliseren van de efficiënte benutting van een plaat en het optimaliseren van het snijproces. Bij de optimalisatie van het snijproces is het vooral belangrijk om het aantal snijlijnen te verminderen, aangezien dit resulteert in tijdsbesparing en een efficiënter gebruik van middelen. Daarentegen zorgt maximalisatie ervoor dat er meer oppervlakte overblijft die later gebruikt kan worden, waarbij beide opties worden toegepast in de applicatie.

Allereerst moeten alle mogelijke opties waar een product geplaatst kan worden, worden geëvalueerd. Dit wordt gedaan aan de hand van een lijst met lege groepen die de nodige informatie bevatten, zoals de afmetingen. Met behulp van zowel de productgegevens als de lege groepen worden berekeningen uitgevoerd. Dit proces resulteert in een RestResult-object per product en lege groep, voor elk RestResultType. Er zijn vier verschillende RestResultTypes:

A picture containing text, screenshot, diagram, display

Description automatically generated

Figuur : Voorbeeld RestResultType Horizontal [2]

A picture containing text, screenshot, display, software

Description automatically generated

Figuur : Voorbeeld RestResultType HorizontalRotated [2]

A screenshot of a graph

Description automatically generated with low confidence

Figuur : Voorbeeld RestResultType Vertical [2]

A picture containing text, screenshot, diagram

Description automatically generated

Figuur : Voorbeeld RestResultType VerticalRotate [2]

Het teruggegeven RestResult omvat verschillende parameters (zie figuur 5), die kunnen worden gebruikt om beslissingen te nemen met betrekking tot de selectie en positionering ervan. Het is opmerkelijk dat er acht verschillende RestResultTypes beschikbaar zijn, terwijl er slechts vier zijn getoond. Dit vormt een uitbreiding op de eerdere bespreking van de vier RestResultTypes, waarin ook gekeken is naar de maximale hoeveelheid blokken die naast elkaar en onder elkaar geplaatst kunnen worden. In de huidige versie van de applicatie worden de block-varianten gebruikt. Echter, indien nodig, kunnen de niet-block-varianten worden toegepast als optimalisatiemaatregel.

A screen shot of a computer program

Description automatically generated with low confidence

Figuur : RestResult Model [2]

Als tweede stap wordt de overgebleven lengte voor horizontale plaatsing en hoogte voor verticale plaatsing, evenals de overgebleven oppervlakte berekend. Deze gegevens worden gebruikt om te bepalen of de snijlijn horizontaal of verticaal geplaatst moet worden. Raadpleeg figuur 6 en 7 voor een visuele weergave.

A screenshot of a graph

Description automatically generated with low confidence

Figuur : Verticale snijlijn [2]

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

Figuur : Horizontale snijlijn [2]

Nadat alle berekeningen zijn uitgevoerd, worden deze resultaten gesorteerd en wordt de beste pasvorm geselecteerd. Deze geselecteerde pasvorm wordt vervolgens doorgegeven aan een functie die verantwoordelijk is voor het plaatsen van het item op de plaat.

**A screen shot of a computer code

Description automatically generated with low confidence**

Figuur : Sorteren en berekenen [2]

### Opslaan van een optimalisatieplan

De gegevens worden opgeslagen met een identificatienummer dat bestaat uit de laatste twee cijfers van het jaartal, gevolgd door een volgnummer van vijf cijfers. Als illustratie kan het nummer 2300001 worden gebruikt. Nadat een nieuwe berekening is opgeslagen wordt dit nummer verhoogd. Indien een dergelijk nummer al bestaat en de berekening wordt opgeslagen, zal er geen nieuwe identificatie worden aangemaakt, maar zal de bestaande identificatie worden bijgewerkt. De ingevoerde gegevens worden opgeslagen in het JSON-formaat. Dit heeft zowel voordelen als nadelen. Aan de ene kant hoeft er minder data bijgehouden te worden dan het volledige plan. Aan de andere kant moet er rekening worden gehouden met de mogelijkheid dat de wijziging van de berekening, bijvoorbeeld door items op een plaat te plaatsen, kan leiden tot een ander resultaat. Dit maakt het echter ook gemakkelijk om later nieuwe berekeningen toe te passen op de oorspronkelijke gegevens ter vergelijking en het wordt ook makkelijker om gegevens toe te voegen in plaats van de datastructuur van de database bij elke wijziging te moeten bijwerken.



Figuur : Optimalisatieplan data [2]

### UI

In de ontwikkeling van de applicatie is een expliciete eis gesteld met betrekking tot het consistent toepassen van de huisstijlkleuren, met als doel de consistentie en herkenbaarheid binnen de merkidentiteit van Benvitec te waarborgen. Daarnaast wordt de nadruk gelegd op de noodzaak van een hoog niveau van gebruikersvriendelijkheid en een intuïtieve gebruikerservaring in de applicatie. Deze vereisten zijn van essentieel belang om ervoor te zorgen dat gebruikers moeiteloos kunnen navigeren en efficiënt kunnen interageren met de functionaliteiten en inhoud van de applicatie.

## Werkwijze

Er wordt wekelijks een vergadering georganiseerd om de prioriteiten te bespreken die gedurende die week moeten worden aangepakt. Ter ondersteuning van dit proces wordt gebruikgemaakt van een Kanban-bord van Microsoft DevOps. Tijdens deze bijeenkomst worden ook de nieuwe functionaliteiten geïdentificeerd die moeten worden toegevoegd en aan de wekelijkse sprint moeten worden toegewezen. Daarnaast worden er ook demonstraties gegeven van de functionaliteiten die reeds zijn voltooid. Bovendien worden er tussen deze vergaderingen door ook bijeenkomsten gepland om eventuele verduidelijkingen samen met de interne klant te bespreken. Deze momenten worden eveneens benut om verbeteringsvoorstellen te presenteren.

Zodra de prioriteiten zijn vastgesteld, worden de issues aangepakt die in de wekelijkse sprint zijn opgenomen. Allereerst wordt de gewenste issue geanalyseerd om te bepalen hoe deze geïmplementeerd kan worden. Vervolgens wordt er gestart met programmeren om dit te realiseren. Zodra de issue is afgerond, wordt er een uitgebreide testprocedure uitgevoerd om de functionaliteit te valideren.

Er wordt wekelijks één release naar de productieomgeving gepusht met daarin de afgeronde issues, zodat deze door de medewerkers kunnen worden getest. De reden waarom de release slechts één keer per week wordt uitgevoerd, is om tijd te creëren voor feedback en om eventuele ongemakken te minimaliseren die kunnen optreden tijdens de releasefase van de website.

Zodra er feedback binnenkomt, worden de bugs geprioriteerd op basis van de issues die de grootste impact hebben. Het vastleggen van prioriteiten vindt plaats tijdens de wekelijkse bijeenkomst en de bugs worden vervolgens opgenomen in de wekelijkse sprints om opgelost te worden.

## Resultaat

### Instellingen

Deze webpagina biedt de mogelijkheid om zaagblad, fineer en initialen in te vullen, zodat deze gegevens kunnen worden opgeslagen. Raadpleeg figuur 10 voor een visuele weergave. Het opslaan van de gegevens wordt ondersteund door de *Local Storage*-functie van webbrowsers. Figuur 11 biedt een visuele weergave van hoe de opgeslagen data eruit ziet.

A screenshot of a computer

Description automatically generated

Figuur : Cutting Optimizer – Instellingen [2]

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Figuur 11: Cutting Optimizer - Local Storage [2]

### Offerte

Er is ook een optie om een offertenummer in te voeren, wat kan dienen als een uniek identificatienummer voor het koppelen van bijbehorende bestellingen. Dit offertenummer wordt eveneens weergegeven in het overzicht, waar later uitvoeriger op zal worden ingegaan.

Afbeelding met tekst, software, Computerpictogram, Webpagina

Automatisch gegenereerde beschrijving

Figuur : Cutting Optimizer – Offerte [2]

### Materialen en items

Op deze pagina is het mogelijk om één of meerdere materialen toe te voegen met de gevraagde opties, zoals prioriteit en standaardinstellingen. De prioriteitsoptie kan op de volgende manier worden toegepast. Een prioriteit van 0 wordt niet gebruikt voor de berekening. Hoe lager het nummer, hoe hoger de prioriteit en wordt het als eerste verbruikt. Ten minste één materiaal dient te worden geselecteerd met een prioriteit groter dan 0, en bovendien dient er één materiaal te worden geselecteerd waarbij de standaardinstelling is aangevinkt. Daarnaast is het mogelijk om items toe te voegen.

De knoppen met het "+"-symbool bieden de mogelijkheid om een extra regel toe te voegen voor materialen of items. Daarentegen kunnen de regels ook verwijderd worden door gebruik te maken van de rode knoppen met een vuilnisbak-symbool

Nadat alle gegevens zijn ingevuld, kan het optimalisatieplan worden berekend door gebruik te maken van de knop "Uitrekenen". Als de berekening succesvol is, wordt het optimalisatieplan weergegeven met de vereiste informatie. In geval van fouten of problemen tijdens het berekenen, worden de foutmeldingen weergegeven. Bovendien is het mogelijk om het gehele formulier te resetten door gebruik te maken van de knop "Resetten".

Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving

Figuur : Cutting Optimizer – Materialen & Items [2]

Nadat een plan is berekend, worden de opties "Opslaan" en "Afdrukken" beschikbaar gesteld. Als het opslaan succesvol is, wordt er een melding weergegeven met het gegenereerde volgnummer, bijvoorbeeld: 2300001. Dit volgnummer kan later opgevraagd worden op deze webpagina.

### Optimalisatieplan

Als eerste wordt een overzicht van de optimalisatie weergegeven. Dit overzicht omvat het plannummer, het offertenummer en de initialen van de persoon die de berekening heeft uitgevoerd, samen met informatie over de zaagbladen en de aanwezigheid van fineermateriaal, op voorwaarde dat de vereiste gegevens zijn ingevuld in de instellingen- en offerte-tabbladen. Daarnaast wordt een inventarisatie gepresenteerd van de vereiste materialen en afmetingen. Raadpleeg figuur 14 voor een visuele weergave.

A screenshot of a computer

Description automatically generated with medium confidence

Figuur : Cutting Optimizer – Overzicht [2]

Vervolgens wordt er een overzicht van de plannen weergegeven, waarbij elk plan wordt vermeld dat moet worden versneden. In dit overzicht wordt ook aangegeven hoeveel items op een specifieke manier moeten worden versneden, evenals het materiaalverlies. Dit overzicht biedt een snelle visuele samenvatting van het geheel. Raadpleeg figuur 15 voor een visuele weergave.

A screenshot of a computer

Description automatically generated with medium confidence

Figuur : Cutting Optimizer –Plannen overzicht [2]

Als laatste in het optimalisatieplan wordt een overzicht gegeven van elk gebruikt materiaal, waarin de te snijden items worden vermeld, samen met de berekening van het materiaalverlies, de snijlijnen en de snijlengte. Tevens wordt een voorbeeld getoond van de wijze waarop de items moeten worden versneden. Raadpleeg figuur 16 en 17 voor een visuele weergave.

A screenshot of a computer

Description automatically generated with medium confidence

Figuur : Cutting Optimizer - Plan overzicht deel één [2]

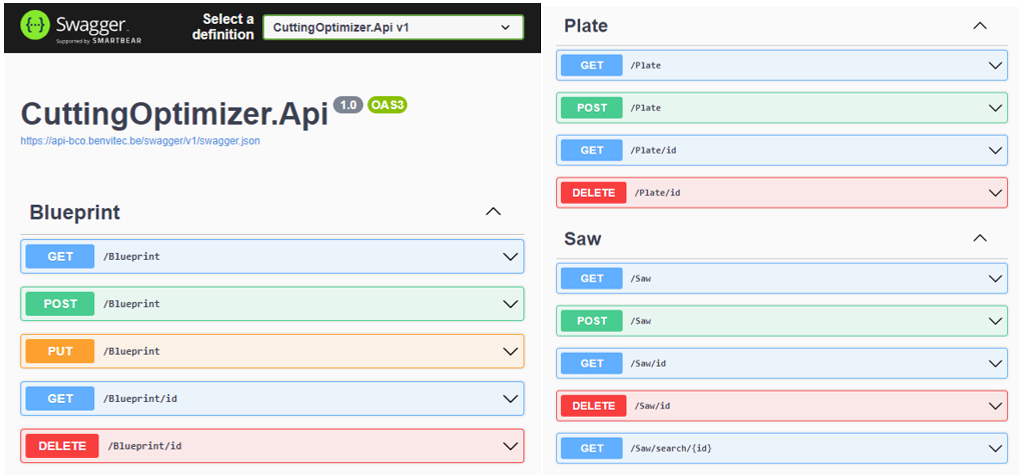
A screenshot of a computer

Description automatically generated

Figuur : Cutting Optimizer - Plan overzicht deel twee [2]

### API

De Cutting Optimizer-tool maakt gebruik van de *API*, die zorgt voor onderlinge communicatie om gegevens op te halen en op te slaan. Swagger wordt ingezet om het proces van het invoeren van nieuwe gegevens te vereenvoudigen. Bovendien biedt deze tool de mogelijkheid om gegevens op te halen, aan te passen en indien nodig te verwijderen. Daarnaast is het ook mogelijk om met behulp van een externe tool, zoals bijvoorbeeld Postman, verzoeken te verzenden. Een overzicht van alle beschikbare acties is te vinden in Figuur 18 en 19. Deze web AP*I* is ontwikkeld met behulp van het ASP.NET Core-framework [3] en is geprogrammeerd in C#.



Figuur : Voorbeeld API [2]

Gegevens kunnen op de volgende manier worden toegevoegd. Door de *post*-actie van het gewenste item te selecteren, opent zich een scherm waarin de gegevens van het verzoek worden ingevuld. Na het invullen van de gegevens in het verzoeklichaam wordt op "uitvoeren" geklikt. Als het verzoek succesvol is, wordt het nieuwe object teruggestuurd. Als er zich een fout voordoet, wordt de bijbehorende foutmelding weergegeven die aangeeft wat er is misgegaan.

Afbeelding met tekst, schermopname, software, scherm

Automatisch gegenereerde beschrijving

Figuur : Voorbeeld post plate [2]

## Reflectie

Dankzij PXL en Benvitec heb ik de mogelijkheid gekregen om een zeer interessant stageproject te voltooien. Dit avontuur begon op 20 maart 2023 en had een duur van 12 weken.

Het project omvatte de ontwikkeling van een webapplicatie voor het optimaal snijden van items uit een plaat, ter vervanging van de huidige applicatie die binnenkort buiten gebruik wordt gesteld. Het was mijn taak om een functionele applicatie op te leveren die bruikbaar is in een productieomgeving en voldoet aan alle gestelde eisen.

Als eerste stap heb ik onderzocht welke frameworks mogelijk geschikt waren en aan de eisen van de klant voldeden. In overleg zijn we tot de conclusie gekomen dat Blazor een zeer goede keuze is voor de client-side. De *API* wordt ontwikkeld met behulp van ASP.NET Core en de eisen van de klant werden vastgelegd op het Kanban-bord van Microsoft DevOps waar dit project zich bevindt. Deze eisen werden wekelijks besproken en aangevuld, zodat ze gebruikt kunnen worden voor het opstellen van de sprintplanning. Dit helpt bij het monitoren van de voortgang van het werk en het bepalen van de verdere werklast.

De communicatie tussen mij en de klant verliep zeer goed, waardoor de vereisten snel duidelijk werden en ik snel kon beginnen met programmeren. Natuurlijk zijn er bepaalde vereisten over het hoofd gezien tijdens de initiële analyse. Deze werden echter aangepakt zodra ze zich voordeden en ook gecommuniceerd met de klant. Het is van groot belang om de klant zo snel en concreet mogelijk op de hoogte te stellen van de situatie.

De complexiteit van de berekening voor deze optimalisatie leidde tot voortdurende wijzigingen in de code. Dit werd voornamelijk veroorzaakt door het gebrek aan beschikbare informatie en mijn beperkte kennis om dit specifieke probleem op te lossen. Deze factoren beïnvloedden ook het schrijven van tests, waarbij aanvankelijk mogelijkheden waren, maar al snel naar de achtergrond verdwenen en te veel tijd in beslag namen. Daarom hebben we besloten om dit aspect voorlopig opzij te zetten en indien mogelijk op een later tijdstip weer op te pakken. Ik zou graag willen leren hoe ik dit in de toekomst anders kan aanpakken, maar momenteel zie ik geen andere manier dan meer ervaring opdoen.

Ik ben uitermate tevreden met zowel het behaalde resultaat als de samenwerking met alle betrokken partijen gedurende dit stageproject. Bovendien ben ik dankbaar voor de kennis die ik tijdens deze ervaring heb kunnen vergaren, die ik zeker zal meenemen naar toekomstige projecten. Ik koester deze waardevolle opportuniteit en ben erkentelijk voor de geboden mogelijkheid.

1. Onderzoekstopic

# Onderzoeksvraag

## Is het Blazor framework volwassen genoeg?

In dit onderzoek vindt er geen vergelijking plaats tussen verschillende frameworks. Het doel is om aan de hand van praktische voorbeelden te onderzoeken of Blazor de benodigde tools biedt voor gebruik in een productieomgeving. De volgende specifieke onderwerpen zullen worden onderzocht:

* Wat zijn de basisonderdelen van Blazor?
* In welke mate kan het gebruik van JavaScript vermeden worden?
* Wat zijn de voor- en nadelen van “Blazor Server” ten opzichte van “Blazor WebAssembly”?

# Onderzoeksmethode

Het onderzoek bestaat uit drie onderdelen:

* Allereerst wordt er gestart met een grondige literatuurstudie om de nodige kennis op te doen.
* Het tweede deel bestaat uit praktische voorbeelden hoe het framework te werk gaat met specifieke elementen.
* Ten slotte wordt er een evaluatie uitgevoerd aan de hand van deze praktische voorbeelden om vast te stellen of Blazor voldoende volwassen is om in productie te worden gebruikt.

# Uitwerking

## Literatuurstudie

### Wat is Blazor? [4]

Met Blazor kunnen ontwikkelaars complexe webapplicaties maken met een rijke gebruikersinterface, vergelijkbaar met traditionele desktopapplicaties. Het framework volgt een component gebaseerde architectuur, waarbij verschillende componenten herbruikbaar zijn en samenwerken om de functionaliteit van de applicatie te realiseren.

Dankzij de naadloze integratie met C# kunnen ontwikkelaars optimaal gebruikmaken van geavanceerde taalfunctionaliteiten, waaronder een sterk getypeerde taal, asynchrone programmering en geavanceerde objectgeoriënteerde concepten. Bovendien maakt Blazor gebruik van Razor-syntax, een combinatie van HTML en C#, waardoor het eenvoudig wordt om de gebruikersinterface op een duidelijke en flexibele manier te definiëren en te manipuleren. Deze combinatie van krachtige taalfuncties en een intuïtieve syntax draagt bij aan een efficiënte ontwikkeling van rijke en interactieve webtoepassingen.

Blazor biedt verschillende uitvoeringsmodellen:

A picture containing text, screenshot

Description automatically generated

Figuur : Blazor Future [5]

### Blazor Server [6]

Blazor Server is een uitvoeringsmodel binnen het Blazor-framework waarbij de logica van de applicatie op de server wordt uitgevoerd en de gebruikersinterface via een real-time verbinding naar de client wordt gestreamd. Het maakt gebruik van *SignalR*-technologie om de communicatie tussen de server en de client te faciliteren.

Bij Blazor Server wordt de gebruikersinterface (UI) op de server gegenereerd en alleen de wijzigingen worden via een *SignalR*-verbinding naar de client verzonden. Hierdoor hoeft de client geen volledige pagina-herlaadacties uit te voeren, maar ontvangt het alleen de relevante updates om de UI bij te werken. De interactie van de gebruiker wordt teruggestuurd naar de server, waar de bijbehorende logica wordt uitgevoerd.

Dit model biedt verschillende voordelen, zoals een snellere initiële laadtijd omdat alleen de UI en niet de volledige applicatie naar de client worden verzonden. Het minimaliseert ook de hoeveelheid gegevens die tussen de server en de client worden uitgewisseld, waardoor de netwerkbelasting wordt verminderd. Bovendien kan Blazor Server schaalbaarheid bereiken doordat meerdere clients dezelfde serverapplicatie kunnen delen.

### Blazor WebAssembly [6]

Blazor WebAssembly is een uitvoeringsmodel binnen het Blazor-framework waarbij de Blazor-applicatie direct in de webbrowser van de client wordt uitgevoerd met behulp van *WebAssembly*-technologie. Hierbij worden zowel de applicatielogica als de gebruikersinterface (UI) volledig naar de client gedownload en lokaal uitgevoerd.

Bij Blazor WebAssembly wordt de Blazor-applicatie gecompileerd naar *WebAssembly*-*binaries*, die vervolgens in de webbrowser worden geladen en uitgevoerd. Dit betekent dat de volledige applicatie, inclusief de C#-code en de Razor-component, naar de client wordt gedownload en daar wordt uitgevoerd. De interactie van de gebruiker vindt lokaal plaats en er is geen constante communicatie met de server nodig, tenzij er specifieke serverfuncties moeten worden aangeroepen.

Dit model biedt verschillende voordelen, zoals een verbeterde responsiviteit en een verrijkte gebruikerservaring, omdat de applicatie lokaal op de client wordt uitgevoerd. Het maakt ook offline functionaliteit mogelijk, omdat de client de applicatie kan blijven gebruiken zonder een constante internetverbinding. Blazor WebAssembly is platformonafhankelijk en kan worden uitgevoerd op verschillende besturingssystemen en webbrowsers.

Naast de voordelen van Blazor WebAssembly zijn er ook enkele nadelen, waaronder de tragere initiële laadtijd vanwege het feit dat de volledige applicatie, inclusief zowel de C#-code als de Razor-componenten, moet worden gecompileerd naar WebAssembly en naar de browser moet worden gedownload voordat deze kan worden uitgevoerd, kan dit resulteren in aanzienlijke vertragingen tijdens het laadproces van de applicatie. Dit kan vooral merkbaar zijn bij grote applicaties met complexe functionaliteiten.

### Blazor Hybrid [6]

Blazor Hybrid is een variant van het Blazor-framework die de mogelijkheid biedt om Blazor-webapplicaties te laten draaien als native mobiele apps op verschillende platformen, zoals iOS en Android. Deze aanpak combineert webtechnologieën, zoals HTML, CSS en JavaScript, voor het renderen van de gebruikersinterface, terwijl de applicatielogica wordt uitgevoerd met behulp van de .NET*-runtime*.

Bij Blazor Hybrid wordt de webgebaseerde UI-hosting samengevoegd met een *native* "webview"-component, die fungeert als een container voor de Blazor-applicatie. Deze *webview* maakt het mogelijk om de Blazor-webapplicatie te integreren in een *native* mobiele app en deze vervolgens te verspreiden via app-winkels.

Het doel van Blazor Hybrid is om ontwikkelaars in staat te stellen om met één codebase zowel web- als mobiele apps te ontwikkelen, waardoor de ontwikkeltijd wordt verkort en de herbruikbaarheid van code wordt vergroot. Dit stelt ontwikkelaars in staat om te profiteren van zowel de voordelen van web- als native mobiele apps, zoals een geavanceerde gebruikersinterface en toegang tot apparaat functies.

### JavaScript Interoperability [7]

Dankzij JavaScript Interoperability is het mogelijk om code in twee richtingen uit te voeren. Zowel vanuit JavaScript kan een C#-functie worden aangeroepen als vanuit C# kan een JavaScript-functie worden aangeroepen. In een later stadium van het onderzoek zal hier nader op worden ingegaan.

### Razor [8]

Blazor maakt gebruik van Razor als templating-engine voor het opbouwen van webpagina's. Razor is een combinatie van C# en HTML en wordt opgeslagen als het bestandstype ".cshtml". Dit is vergelijkbaar met andere templating-engines, zoals bijvoorbeeld Angular, maar in plaats van JavaScript wordt er gebruikgemaakt van C#. Razor wordt voornamelijk gebruikt in combinatie met ASP.NET, dat ook door Microsoft is ontwikkeld. Daarom heeft Microsoft ervoor gekozen om Razor opnieuw te gebruiken voor Blazor.

### Projectstructuur Blazor

A screenshot of a computer program

Description automatically generated with medium confidenceIn een Blazor-project zijn doorgaans de volgende onderdelen en mappen te vinden:

Figuur : Projectstructuur [9]

* De "wwwroot"-map: Deze map bevat de statische bestanden van de applicatie, zoals HTML-, CSS- en JavaScript-bestanden, afbeeldingen en andere bronnen die rechtstreeks toegankelijk zijn via de webserver.
* De "Pages"-map: Deze map bevat de Blazor-pagina's (Razor-bestanden) die deel uitmaken van de gebruikersinterface. Elke pagina vertegenwoordigt een specifieke weergave of functionaliteit van de applicatie.
* De "Shared"-map: De "Shared"-map bevat herbruikbare Blazor-componenten, zoals lay-outs, navigatiebalken, voetteksten en andere elementen die in meerdere pagina's kunnen worden gebruikt.
* De "Models"-map: Dit is de map waarin de modelklassen worden geplaatst. Deze klassen definiëren de gegevensstructuren die worden gebruikt in de applicatie en fungeren als een communicatiemiddel tussen de verschillende onderdelen van de applicatie.
* Het "App.razor"-bestand: Dit bestand vertegenwoordigt de hoofdcomponent van de applicatie. Hier worden de routering en andere configuraties gedefinieerd die van toepassing zijn op de gehele applicatie.
* Het "Program.cs"-bestand: Het "Program.cs"-bestand bevat de configuratie en opstartcode voor de Blazor-applicatie, inclusief de instellingen voor het hosten van de applicatie.

## Implementatie

### Razor Component

Om een nieuwe pagina toe te voegen, wordt gebruik gemaakt van een Razor-component. In de onderstaande afbeelding zijn verschillende elementen te zien. Zo wordt het element "@page" gebruikt om de route van de pagina in te stellen. Het tweede element is de "PageTitle"-tag, die de titel zichtbaar maakt in het tabblad van de browser. Als laatste is er een "code"-blok waarin de logica wordt geschreven. Opmerkelijk is dat zowel de front-end als de back-end samen op één pagina aanwezig zijn. Een Razor-component kan ook worden hergebruikt en genest. Dit zal later nader worden toegelicht.

A screenshot of a computer

Description automatically generated with medium confidence

Figuur : Blazor-component [9]

### Variabele en functies gebruiken

Als eerste moet er een variabele gemaakt worden in het "code"-blok wat in dit geval een *Integer* is met de naam “Counter”. Om een variabele weer te geven op een pagina moet er buiten de "code"-blok waar HTML mag geschreven worden gebruik maken van de “@” waarmee verwezen kan worden naar een bestaande variabele. In onderstaande voorbeeld wordt er gebruikt gemaakt van de variabele “@Counter”.

Naast het gebruik van variabelen kunnen ook gebruikmaken van functies. Deze functionaliteit wordt ook gedefinieerd in het “@code”-blok. Dit kan gekoppeld worden aan DOM-events zoals onder andere *onclick*. Er zijn meerdere manieren om dit te bewerk stellen. Als de functie geen parameters vereist kan er gebruik worden gemaakt van een “@” gevolgd door de functienaam.

Bij Standaard events worden eventparameters automatische meegeven zoals aangegeven in onderstaande afbeelding op lijn twaalf. Als er manueel parameters dienen meegegeven worden kan dit bewerkstelligd worden met behulp van een *Lamda*, zoals te zien op lijnen 11 en 14 van figuur 24.

A screenshot of a computer

Description automatically generated

Figuur : Voorbeeld variabele en functies [9]

A screen shot of a computer program

Description automatically generated with medium confidence

Figuur : Variable en functies [9]

### Razor Component verdiepen

### Hergebruiken

Zoals eerder aangegeven, is het mogelijk om een Razor-component te hergebruiken. Zodra een Razor-component is gemaakt, voorziet het framework automatisch een tag die in de HTML-code kan worden geplaatst. Dit is te zien in Figuur 25, regel 11. Op deze manier wordt de "counter"-component zichtbaar op de homepagina.

A screenshot of a computer program

Description automatically generated with medium confidence

Figuur : Razor-component gebruiken [9]

A screenshot of a computer

Description automatically generated

Figuur : Voorbeeld Razor-component gebruiken

### Variabele doorgeven

Er bestaat ook de mogelijkheid om variabelen door te geven aan componenten. Het is van belang om te weten om welk type variabele het gaat: of het een *reference* of een *primitive* type is. Dit bepaalt of de variabele al dan niet aanpasbaar is.

In het gegeven voorbeeld wordt als eerste een variabele aangemaakt genaamd "Errors". Deze variabele is van het type "List<string>", wat betekent dat het een lijst van strings is en zal doorgegeven worden via de “ErrorList”-parameter van de component.

A screen shot of a computer program

Description automatically generated with medium confidence

Figuur : Variabele doorgeven aan component [9]

In het "code"-blok van de component kan een variabele worden gedefinieerd met de “Parameter”-annotatie, waarbij de toegankelijkheid wordt ingesteld op *public* zodat deze van buitenaf kan worden benaderd. Dit is te zien in Figuur 28, regel 15. Uiteraard kan deze variabele worden weergegeven zoals eerder gezien. Aangezien het hier een lijst van *strings* betreft, wordt in het onderstaande voorbeeld gebruikgemaakt van “@foreach” om de volledige lijst weer te geven.

A screen shot of a computer program

Description automatically generated with medium confidence

Figuur : Variabele ontvangen in component [9]

### Functies doorgeven

In Blazor is het mogelijk om een functie door te geven aan een andere component. Dit kan worden bereikt door gebruik te maken van parameters en *delegates*.

Voordat ze aan een component worden doorgegeven, wordt er eerst een parameter gedefinieerd van het *delegate-*type in de ontvangende component. Een *delegate* is een verwijzing naar een methode met een specifieke signatuur. In deze context wordt een *delegate* gebruikt om een functie te representeren die aan de component wordt doorgegeven.

A screen shot of a computer program

Description automatically generated with medium confidence

Figuur : functies doorgeven aan component [9]

De ontvanger kan de doorgegeven functie vervolgens aanroepen wanneer dat nodig is. Dit biedt een flexibele manier om functionaliteit van de ene component naar de andere over te dragen en maakt het mogelijk om gebeurtenissen in de ene component af te handelen in een andere component.

Door *delegates* te gebruiken en functies door te geven aan componenten, kan de functionaliteit van Blazor-component uitgebreid worden en wordt de bevordering van herbruikbare code mogelijk gemaakt. Dit stelt ontwikkelaars in staat om krachtige en flexibele webapplicaties te bouwen met behulp van Blazor.

A screen shot of a computer program

Description automatically generated with low confidence

Figuur : Functie ontvangen in component [9]

### EventCallback

*Event callbacks* stellen Blazor-componenten in staat om gebeurtenissen te genereren en erop te reageren, zowel voor standaard HTML-elementen als voor aangepaste componenten. Ze bieden een flexibele manier om de interactie en communicatie tussen componenten in een Blazor-toepassing te beheren.

In het onderstaande voorbeeld wordt de functie "PrintText" gedefinieerd in de *parent-*component met de benodigde logica en wordt vervolgens doorgegeven aan de "PassEventCallBackComponent" via de parameter "ConsoleEventsCallback". Deze overdracht vindt plaats op regel 13 van figuur 31.

A screen shot of a computer program

Description automatically generated with medium confidence

Figuur : EventCallback doorgeven aan component [9]

Hierdoor kan het worden gebruikt in een component van de *EventCallback*-variabele "ConsoleEventCallBack". Dit fungeert als een doorgeefkanaal en zorgt ervoor dat de functie wordt uitgevoerd wanneer het gerelateerde event plaatsvindt.

A screen shot of a computer program

Description automatically generated with low confidence

Figuur : EventCallback ontvangen in component [9]

### Form & Data binding [10]

Het tot stand brengen van een formulierkoppeling in Blazor vereist de inzet van een EditForm-component, waarbij het formulier wordt geassocieerd met een gegevensmodel om automatische synchronisatie van waarden te faciliteren. Binnen deze EditForm-component kunnen de formuliervelden worden verbonden met eigenschappen in het gegevensmodel middels het gebruik van de “@bind-directive” (Zie figuur 33, regel 4, waarin het EditForm-component wordt gekoppeld aan de waarde “@Person”).

Eveneens omvat de functionaliteit het valideren van de gegevens in het formulier en het tonen van foutmeldingen met het gebruik van de ValidationMessage-component. Dit kan worden bewerkstelligd via de EditContext of via een apart model. Een gedetailleerde behandeling van het validatieaspect volgt in het komende gedeelte.

Blazor biedt diverse ingebouwde invoercomponenten die kunnen worden gebruikt om gebruikersinvoer te verwerken. Enkele voorbeelden van deze componenten zijn:

* InputText: Hiermee kan de gebruiker een tekstwaarde invoeren.
* InputNumber: Hiermee kan de gebruiker een numerieke waarde invoeren.
* InputCheckbox: Hiermee kan de gebruiker een waarde selecteren door een selectievakje aan of uit te vinken.
* InputSelect: Hiermee kan de gebruiker een waarde selecteren uit een vooraf gedefinieerde lijst met opties.
* InputDate: Hiermee kan de gebruiker een datum selecteren.
* InputTextArea: Hiermee kan de gebruiker een tekst met meerdere regels invoeren.

Deze ingebouwde invoercomponenten vereenvoudigen de verwerking van diverse vormen van gebruikersinvoer en voorzien in ingebouwde functionaliteiten zoals gegevensbinding en validatie. De binding tussen de invoercomponenten en de gegevens wordt tot stand gebracht door middel van de “@bind-value” attribuutnotatie. Hierdoor zijn de componenten in staat om automatisch de ingevoerde waarden bij te houden en te synchroniseren met de corresponderende gegevensbron, wat resulteert in een naadloze interactie tussen de gebruikersinvoer en de gegevens.

In essentie genereren de ingebouwde Blazor-invoercomponenten daadwerkelijk een HTML <input> element met de vereiste parameters om een vergelijkbare functionaliteit te bewerkstelligen. Ze stellen gebruikers in staat om dezelfde soorten invoervelden te gebruiken als die van de HTML <input> elementen, zoals tekstvelden, selectievakjes, keuzelijsten, enzovoorts. Bovendien bieden ze de mogelijkheid om attributen zoals type="text", type="email", type="number", enzovoorts te gebruiken om het gewenste gegevenstype van de invoervelden te definiëren, terwijl de functionaliteit voor het valideren van de gegevens wordt bereikt via andere middelen.

Het is ook mogelijk om events te activeren op formuliervelden, zoals het gebruik van "@oninput" zoals te zien is in figuur 33 op regel 16 en 51.

A screenshot of a computer program

Description automatically generated with medium confidence

Figuur : InputFormComponent [9]

In de EditForm-component van Blazor bieden de gebeurtenissen OnSubmit, OnValidSubmit en OnInvalidSubmit verschillende manieren om de verwerking van formulierinzendingen te beheren op basis van de validatie-uitkomst.

OnSubmit: Deze gebeurtenis wordt geactiveerd wanneer het formulier wordt ingediend, ongeacht of de formuliervelden al dan niet aan de validatiecriteria voldoen. Het is een algemene gebeurtenis waarmee u de actie van het formulier verzenden kunt afhandelen zonder rekening te houden met de validatiestatus.

OnValidSubmit: Deze gebeurtenis wordt alleen geactiveerd wanneer het formulier wordt ingediend en de validatie succesvol is. Het scenario waarin het formulier met ongeldige gegevens wordt ingediend, wordt uitgesloten. Gebruik deze gebeurtenis wanneer u specifieke acties wilt uitvoeren die alleen gelden voor succesvolle formulierinzendingen.

OnInvalidSubmit: Deze gebeurtenis wordt alleen geactiveerd wanneer het formulier wordt ingediend en de validatie mislukt, wat aangeeft dat er fouten in het formulier aanwezig zijn. Gebruik deze gebeurtenis wanneer u het scenario wilt afhandelen waarin de gebruiker probeert het formulier met ongeldige gegevens in te dienen.

Samengevat bieden deze gebeurtenissen verschillende mogelijkheden om formulierinzendingen te verwerken op basis van de validatiestatus.

### Validatie [10]

Input validatie in het Blazor-framework gebeurt door middel van *Data Annotations*. Het gebruik van *Data Annotations* in de context van softwareontwikkeling biedt verschillende voordelen. Allereerst voorzien *Data Annotations* in een gestandaardiseerde en consistente benadering van gegevensvalidatie. Door het toepassen van annotaties op de eigenschappen van gegevensmodellen kunnen ontwikkelaars specifieke validatieregels definiëren. Deze aanpak waarborgt een uniforme validatie van gegevens en draagt bij aan de integriteit en betrouwbaarheid van de gegevens binnen de softwaretoepassing.

Daarnaast bevorderen *Data Annotations* de herbruikbaarheid van code. Ontwikkelaars kunnen de annotaties die zijn gedefinieerd voor een bepaald gegevensmodel opnieuw gebruiken in andere delen van de applicatie. Dit vermindert de noodzaak om herhaaldelijk dezelfde validatielogica te implementeren en bevordert een efficiëntere ontwikkeling.

Een ander voordeel van *Data Annotations* is dat ze bijdragen aan de documentatie en begrijpelijkheid van de code. Door annotaties toe te passen, worden de validatieregels en metadata van gegevensmodellen duidelijk gedocumenteerd. Dit vergemakkelijkt het begrip van de code voor ontwikkelaars en draagt bij aan een betere samenwerking binnen het ontwikkelingsteam.

Bovendien dragen *Data Annotations* bij aan het verbeteren van de gegevensintegriteit. Door het toepassen van annotaties zoals verplichte velden, lengtebeperkingen en andere validatieregels, kunnen ontwikkelaars voorkomen dat ongeldige gegevens in de applicatie worden ingevoerd. Dit helpt bij het waarborgen van de betrouwbaarheid en consistentie van de gegevens, wat cruciaal is voor het correct functioneren van de softwaretoepassing.

In het kader van gegevensvalidatie kan gebruik worden gemaakt van *Data Annotations* (System.ComponentModel.DataAnnotations). Deze klasse biedt een verzameling van validaties, zoals *Required*, *MinLength*, *MaxLength*, *Range*, en andere. Door gebruik te maken van de parameters in de annotaties kunnen voorwaarden worden gespecificeerd en kan ook een specifiek foutbericht worden weergegeven. Deze annotaties worden in een *Model* (klasse) geplaatst boven de eigenschap. In de onderstaande figuur wordt een voorbeeld gegeven van hoe deze annotaties worden toegepast. Het is vermeldenswaardig dat deze annotaties wijdverspreid zijn onder .NET-ontwikkelaars vanwege hun integratie in het ASP.NET MVC-framework.

A screenshot of a computer program

Description automatically generated with medium confidence

Figuur : Person model [9]

In de onderstaande figuur wordt een concreet voorbeeld gegeven ter illustratie van een specifieke op maat gemaakte validatie die wordt toegepast door middel van de annotatie "FirstLetterCapital". Deze aangepaste annotatie is ontwikkeld om te controleren of de eerste letter van een te valideren tekenreeks een hoofdletter is. Wanneer de eerste letter niet aan deze voorwaarde voldoet, zal de validatie mislukken en wordt er een specifiek foutbericht gegenereerd om de gebruiker op de hoogte te stellen van de ongeldige invoer. Door de flexibele en gestandaardiseerde benadering van deze annotaties bij het valideren van gegevens en het behandelen van uitzonderingen, bieden ze een toegevoegde waarde. Dit wordt duidelijk geïllustreerd aan de hand van het specifieke voorbeeld van de "FirstLetterCapital"-annotatie.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figuur : Aangepaste validatie [9]

Er zijn meerdere manieren om validatiefouten weer te geven. Dit kan per eigenschap worden gedaan met behulp van de "<ValidationMessage/>"-tag, waarbij specifieke foutmeldingen voor elke eigenschap worden getoond getoond (zie figuur 33 als implementatie en figuur 36 als voorbeeld). Daarnaast is er de mogelijkheid om alle fouten tegelijkertijd weer te geven met behulp van de "<ValidationSummary/>"-tag.

A screenshot of a computer

Description automatically generated

Figuur : Voorbeeld form validatie

### Dependency Injection (DI) [11]

*Dependency Injection* (DI) is een ontwerppatroon in de softwareontwikkeling dat tot doel heeft de koppeling tussen componenten te verminderen en de modulariteit, herbruikbaarheid en testbaarheid van software te vergroten.

Het toepassen van DI biedt verschillende voordelen. Ten eerste maakt het de componenten losgekoppeld van hun afhankelijkheden, waardoor wijzigingen in de afhankelijkheden kunnen worden doorgevoerd zonder dat de componenten zelf aangepast hoeven te worden. Dit bevordert de onderhoudbaarheid van de software. Ten tweede vergemakkelijkt DI het testen van componenten, omdat afhankelijkheden kunnen worden vervangen door *mocks* tijdens het testen. Dit draagt bij aan de verbetering van de testbaarheid van de software. Daarnaast bevordert DI de modulariteit en herbruikbaarheid van componenten, omdat deze eenvoudig kunnen worden geïntegreerd in verschillende contexten en projecten door de externe configuratie van de afhankelijkheden.

Services worden toegevoegd in het bestand “Program.cs”. Om dit te bereiken, wordt er gebruikgemaakt van de methoden builder.Services.AddScoped, builder.Services.AddSingleton of builder.Services.AddTransient. Deze methoden worden gebruikt om de gewenste service toe te voegen. Dit is te zien in figuur 37, regel 11.

A screenshot of a computer program

Description automatically generated with medium confidence

Figuur : Service registreren [9]

Om een service te injecteren in een Blazor*-*component, moet men gebruikmaken van de “@Inject-directive”, zoals te zien is in Figuur 38, regel 5 en 6. Daarbij wordt ook een variabele aan deze injectie gekoppeld, zodat deze later toegankelijk is in de code. Dit wordt geïllustreerd in Figuur 38, regel 27 en 33.

A picture containing text, screenshot, software, display

Description automatically generated

Figuur : Di injecteren en gebruiken [9]

### JavaScript Interoperability

Voor het gebruik van JavaScript *Interoperability* in Blazor dient er een JavaScript-bestand aan het project te worden toegevoegd in de “\_layout.cshtml” onder de “<script>”-tag (zie figuur 39, lijn 14). Daarnaast vereist het gebruik van de JavaScript *Interoperability*-functionaliteit ook de injectie van *IJSRuntime*.

A picture containing text, electronics, screenshot, software

Description automatically generated

Figuur : JavaScript bestand toevoegen [9]

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figuur : JavaScript-code [9]

A screen shot of a computer program

Description automatically generated with low confidence

Figuur : JavaScript uitvoeren adv. C# [9]

### Hoe wordt JavaScript gebruikt vanuit C#

Binnen de Blazor-component wordt een C#*-*methode gecreëerd die de JavaScript-functie zal aanroepen. Om dit te bereiken, wordt de *JSRuntime* gebruikt binnen de Blazor-component om de JavaScript-functie aan te roepen met behulp van de *InvokeAsync*-methode.

### Hoe wordt C# gebruikt in JavaScript

Er wordt een functie gedefinieerd met de “JSInvokable”-annotatie, zodat deze van buitenaf toegankelijk wordt. Hierdoor kan *JavaScript* deze functie oproepen door middel van de DotNet.*invokeMethodAsync*-functie.

Conclusie

Het framework biedt essentiële bouwstenen en maakt gebruik van de solide basis van Razor-pagina's. Hoewel niet alle aspecten van Blazor zijn behandeld, zou verder onderzoek interessant zijn om diepgaander in te gaan op de mogelijkheden van het framework.

Een van de voornaamste voordelen van Blazor is het vermogen om code te schrijven in C#, in tegenstelling tot Javascript. Deze eigenschap maakt de overgang naar webontwikkeling gemakkelijker voor .NET-ontwikkelaars, aangezien zij al bekend zijn met de C#-taal en het .NET-ecosysteem.

De voorbeelden tonen aan dat Blazor een veelbelovende technologie is voor het creëren van geavanceerde webapplicaties, zonder de noodzaak om Javascript-code te schrijven. Het framework biedt ontwikkelaars de mogelijkheid om op een gestroomlijnde en efficiënte wijze interactieve en responsieve applicaties te ontwikkelen.

Een bijkomend voordeel van Blazor is de componentgebaseerde architectuur, die vergelijkbaar is met gerenommeerde frontendframeworks zoals React en Angular. Blazor maakt het mogelijk herbruikbare componenten te bouwen en deze te combineren tot complexe applicaties. Deze component gebaseerde benadering bevordert modulariteit, vereenvoudigt het onderhoud en vergemakkelijkt codehergebruik.

Daarnaast maakt Blazor gebruik van *WebAssembly*, een opkomende webstandaard die ontwikkelaars in staat stelt code in de browser uit te voeren, onafhankelijk van de programmeertaal. Dit stelt Blazor-applicaties in staat om op een efficiënte wijze te functioneren, vergelijkbaar met native applicaties.

Niettemin dient opgemerkt te worden dat Blazor enkele uitdagingen en beperkingen met zich meebrengt waar ontwikkelaars rekening mee dienen te houden. Gezien de relatieve nieuwheid van Blazor, bevindt het ecosysteem zich nog in ontwikkeling en zijn niet alle bestaande Javascript-bibliotheken en -tools direct compatibel met Blazor. Dit kan aanpassingen of alternatieve oplossingen vereisen bij het integreren van externe bronnen.

Op basis van onderzoek kan geconcludeerd worden dat Blazor voldoende volwassen is om in productie te worden gebruikt, met inachtneming van de voor- en nadelen. Niet alle webapplicaties hebben baat bij het gebruik van Blazor, maar als het doel is om een bekend ecosysteem zoals .NET te gebruiken en de afhankelijkheid van JavaScript te verminderen zodat ontwikkelaars sneller aan de slag kunnen of het gebruikt van WebAssembly, is Blazor een uiterst geschikt framework.

Reflectie

Voor dit onderzoek heb ik Blazor gekozen als het passende framework dat voldoet aan de eisen voor mijn stageproject bij Benvitec. In dit project moest ik een webapplicatie ontwikkelen, bij voorkeur in C#, vanwege de reeds aanwezige interne kennis van deze taal. De twee beschikbare C#-frameworks voor webontwikkeling waren ASP.NET MVC en Blazor. Het eerste framework vereiste echter dat ik het grootste deel van de code in JavaScript zou schrijven, wat niet overeenkwam met de voorkeuren van Benvitec. Daarom heb ik besloten om een onderzoek uit te voeren naar Blazor.

In het kader van dit onderzoek waren mijn taken als volgt georganiseerd. Allereerst was het noodzakelijk om een grondige literatuurstudie uit te voeren. Vervolgens ging ik praktisch aan de slag met Blazor-code, die diende als voorbeeldmateriaal om deze paper te schrijven.

Ik ondernam verschillende acties in het kader van dit onderzoek, waaronder een literatuurstudie over het onderwerp en het verkennen van de praktische toepassing van Blazor in een productieomgeving. Mijn doel was om te beoordelen of Blazor voldeed aan de gestelde vereisten. Nadat ik de literatuurstudie had afgerond, begon ik met het schrijven van code in Blazor, waarbij ik me voornamelijk richtte op de basisfunctionaliteiten. Nadat ik voldoende kennis had verzameld, ben ik begonnen met het opstellen van dit eindwerk.

De combinatie van mijn stage en het schrijven van dit eindwerk bleek veel zwaarder te zijn dan ik had verwacht. Desondanks ben ik persoonlijk erg tevreden met het resultaat. Ik had graag meer tijd gehad om dieper in te gaan op Blazor dan alleen de basisprincipes. Het was een uitdaging om te bepalen welke aspecten het meest relevant waren voor dit onderzoek. Maar alle onderzoeksvragen zijn beantwoord en tenslotte heb ik gebruik kunnen maken van Blazor voor de stageopdracht.

Ik ben aanvankelijk gestuit op enige moeilijkheden bij het doorlopen van de literatuurstudie, wat resulteerde in het verlies van veel tijd, voornamelijk vanwege het gebrek aan goede bronnen buiten de documentatie van Microsoft Blazor. Wat ik heb geleerd uit dit project is dat goede documentatie van een framework uiterst belangrijk is. Ik realiseer me nu dat ik zelf te weinig tijd besteed aan dit aspect, en in de toekomst wil ik proberen hier meer energie in te steken. Bovendien heb ik veel over mezelf geleerd, zoals het tonen van mijn vaardigheden in plaats van mezelf de kans te ontnemen om dat te doen.

Bronnenlijst

|  |  |
| --- | --- |
| [1] | Benvitec, „Over ons,” Benvitec, 01 05 2023. [Online]. Available: https://benvitec.be/over-ons/. [Geopend 01 05 2023]. |
| [2] | S. Kazmierczak, „CuttingOptimizer,” 09 06 2023. [Online]. Available: https://github.com/Dtc-Xtreme/CuttingOptimizer. [Geopend 09 06 2023]. |
| [3] | Microsoft, „Create web APIs with ASP.NET Core,” Microsoft, 11 04 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/web-api/?WT.mc\_id=dotnet-35129-website&view=aspnetcore-7.0. [Geopend 10 04 2023]. |
| [4] | Microsoft, „ASP.NET Core Blazor,” 24 02 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/?WT.mc\_id=dotnet-35129-website&view=aspnetcore-7.0. [Geopend 01 05 2023]. |
| [5] | Michiel, „Je gehele site in één taal: met Blazor brengt Microsoft Front-End en Back-End bij elkaar,” ShareValue, 30 03 2020. [Online]. Available: https://www.sharevalue.nl/blogs/je-gehele-site-in-een-taal-met-blazor-brengt-microsoft-front-end-en-back-end-bij-elkaar. [Geopend 03 05 2023]. |
| [6] | Microsoft, „ASP.NET Core Blazor hosting models,” Microsoft, 04 04 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-7.0. [Geopend 04 05 2023]. |
| [7] | Microsoft, „ASP.NET Core Blazor JavaScript interoperability (JS interop),” Microsoft, 27 04 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/javascript-interoperability/?view=aspnetcore-7.0. [Geopend 07 05 2023]. |
| [8] | Microsoft, „Razor syntax reference for ASP.NET Core,” Microsoft, 23 02 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-7.0. [Geopend 04 05 2023]. |
| [9] | S. Kazmierczak, „Blazor,” 10 06 2023. [Online]. Available: https://github.com/Dtc-Xtreme/Blazor. [Geopend 10 06 2023]. |
| [10] | Microsoft, „ASP.NET Core Blazor forms and input components,” Microsoft, 04 04 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/forms-and-input-components?view=aspnetcore-7.0#built-in-input-components. [Geopend 6 10 2023]. |
| [11] | Microsoft, „.NET dependency injection,” Microsoft, 18 03 2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection. [Geopend 07 05 2023]. |