THOMAS MORE

Klantenportaal

Realisatiedocument

Thijs Lintermans
Student Bachelor in de Toegepaste Informatica – Applicatieontwikkeling

2024 - 2025

Inhoudsopgave

1. INLEIDING 3

2. ANALYSE 4

2.1. Front-end 4

2.1.1. Technologieën 4

2.1.2. Styling-framework 9

2.2. Backend 12

2.3. Database 17

3. DECA – KLANTENPORTAAL 21

3.1. Opzet van de toepassing 21

3.2. Functionaliteiten 22

3.2.1. Hoofddoel 22

3.2.2. Company Information 23

3.2.3. Checklist dashboard 25

3.2.4. Checklist versies 26

3.2.5. Checklist 28

3.2.6. Notifications 31

3.2.7. FileMaker layout: projectlijst 31

3.2.8. FileMaker layout: versieslijst 32

3.2.9. FileMaker layout: Checklist 32

4. BESLUIT 34

LITERATUURLIJST 35

GENERATIEVE AI 37

1. Inleiding

In dit document wordt de realisatie beschreven van het project dat tijdens mijn stage werd uitgevoerd. Het bouwt voort op het projectplan werd opgesteld tijdens het begin van mijn stage, waarin de doelstellingen, planning en verwachte resultaten van het project werden vastgesteld. In dit realisatiedocument wordt toegelicht hoe het project daadwerkelijk werd aangepakt en uitgevoerd.

Eerst wordt er in dit document een grondige analyse gemaakt van mogelijke tools, technologieën, ontwikkelomgeving en platformen geschikt voor de realisatie van het project. Daarna worden de belangrijkste functionaliteiten van het ontwikkelde klantenportaal besproken. Tot slot volgt er nog een besluit waarin het resultaat van het project wordt geëvalueerd en wordt er teruggeblikt op het verloop van de stage.

2. Analyse

In dit hoofdstuk wordt de voorbereiding op de daadwerkelijke uitvoering van de opdracht uitgeschreven. Hierbij wordt een grondige analyse gemaakt van de mogelijke tools, technologieën, ontwikkelomgeving en platformen geschikt voor de realisatie van het project.

Om mijn keuze te verantwoorden en deze vergelijking te doen, maak ik gebruik van de Weighted Ranking Method.

2.1. Front-end

In dit eerste deel bespreek ik de vergelijking van de verschillende front-end technologieën alsook het styling framework.

Eerst zal ik verschillende technologieën vergelijken die gebruikt kunnen worden om het project tot een mooi en succesvol einde te brengen. Er is aangegeven dat de voorkeur uitgaat naar de technologieën: PHP of Angular. Aangezien de softwareontwikkelaar hiermee ervaring heeft.

Als tweede volgt er een vergelijking van de twee stylingtools, Bootstrap en Tailwind CSS. Hiervoor is er geen bepaalde voorkeur en is er vrij te kiezen.

2.1.1. Technologieën

Angular de opvolger van AngularJS (Wikipedia, 27/03/2023), is een open-source front-end framework voor webapplicaties, ontwikkeld en onderhouden door Google. (Beeproger, n.d.) Het is gebaseerd op TypeScript, ontwikkeld door Microsoft en een superset van JavaScript. (kinsta, 07/06/2023) ontworpen om efficiënt single-page en dynamische webapplicaties te maken. Het biedt een flexibele, schaalbare en onderhoudsvriendelijke architectuur. (Crooks, 19/06/2024)

Het framework gebruikt een component gebaseerde aanpak, waarbij een gebruikersinterface wordt opgebouwd uit herbruikbare componenten die afhankelijk kunnen functioneren. Verder ondersteunt Angular een grote verzameling van bibliotheken die een groot aantal van functies beschikbaar maken, waaronder routing, formulierbeheer, client-server communicatie en meer. (Angular, 15/08/2023)

PHP is een open-source, server-side scripttaal die gebruikt wordt voor het maken van dynamische webapplicaties. PHP is eenvoudig, heeft een brede ondersteuning en biedt veel integratiemogelijkheden. (Mensink, 17/01/2022)

ReactJS is een JavaScript-bibliotheek voor het ontwikkelen van interactieve gebruikersinterfaces. Het is een open-source front-end technologie, die veel flexibiliteit biedt bij het ontwikkelen van gebruikersinterfaces voor webapplicaties. (winacademy, 18/02/2021) (Zedrox, 04/09/2024) (Goralski, 26/05/2022)

Voor- en nadelen front-end technologieën:

Technologie	*Voordelen*	*Nadelen*
Angular	- Veel functionaliteiten en plug-ins beschikbaar, doordat het open-source is - Wijzigingen in data en gebruikersinterface zijn automatisch gesynchroniseerd, door middel van two-way binding - Herbruikbare componenten - Uitgebreide en actieve ontwikkelaarsgemeenschap, wat zorgt voor veel ondersteuning en documentatie - Onderhouden en ontwikkeld door Google, wat bijdraagt aan de betrouwbaarheid (Beeproger, n.d.) (Daan, n.d.)	- Tijdrovend met een steile leercurve - Complex en omslachtig, wat het minder geschikt maakt voor kleinere applicaties (Beeproger, n.d.) (Daan, n.d.) (Siddarth, 09/08/2025)
PHP	- Veel functionaliteiten en plug-ins beschikbaar, doordat het open-source is - Uitgebreide en actieve ontwikkelaarsgemeenschap, wat zorgt voor veel ondersteuning en documentatie - Onderhouden door veel ontwikkelaars, waardoor bugs snel gevonden en opgelost worden (ThePHPAdmin, 12/07/2017) (Mensink, 17/01/2022)	- Het wordt beschouwd als een verouderde taal, zeker nu er veel nieuwere frameworks zijn - PHP maakt gebruik van zwakke types, waardoor onjuiste gegevens aan de gebruiker kunnen worden gepresenteerd - Beperkte debug mogelijkheden, vooral als het gaat om het gebruik van breakpoints en geavanceerde debug tools (Mensink, 17/01/2022)

ReactJS	- Veel ontwikkelaars en een brede gebruikersgroep, dit versnelt het ontwikkelingsproces - Herbruikbare componenten - Virtuele DOM zorgt dat alleen gewijzigde onderdelen worden bijgewerkt en de interface efficiënter wordt aangepast - Onderhouden en ontwikkeld door Facebook, wat bijdraagt aan de betrouwbaarheid (Zedrox, 04/09/2024) (Coleman, 30/08/2023) (Goralski, 26/05/2022)	- Bij het updaten naar een nieuwe versie kunnen er compatibiliteitsproblemen optreden - Complex - Minder gestructureerd (Coleman, 30/08/2023)

Tabel 1: Voor- en nadelen front-endtechnologieën

Samenvatting:

Angular is een krachtig front-end technologie met veel functionaliteiten, herbruikbare componenten en goede ondersteuning dankzij een actieve gemeenschap en Google. Het ondersteunt two-way binding, wat zorgt voor een dynamische gebruikerservaring. Daartegenover staat dat Angular complex is, een steile leercurve heeft en minder geschikt is voor kleine projecten.

PHP is een open-source scripttaal met een grote ontwikkelaarsgemeenschap en veel plug-ins. Het wordt actief onderhouden en is stabiel, maar wordt vaak als verouderd beschouwd. Het zwakke gebruik van types en moeilijk debuggen, maken het foutgevoeliger.

ReactJS biedt snelle en efficiënte gebruikersinterface aanpassingen dankzij de Virtuele DOM en stelt ontwikkelaar in staat om herbruikbare componenten te bouwen en gebruiken. Het heeft een brede gebruikersgroep en wordt ondersteund door Facebook. Nadelen zijn de compatibiliteitsproblemen bij het updaten naar een nieuwe versie, een slechtere structuur dan de rest en hogere complexiteit bij de ontwikkeling van grote applicaties.

Het kiezen van een framework heeft ten slotte vaak te maken met een persoonlijke voorkeur.

WRM front-end technologieën:

Criteria	*Gewicht (0-5)*	*Angular*	*PHP*	*ReactJS*
Ervaring	4	4	3	2
Leercurve	2	2	3	2
Herbruikbaarheid van code	3	3	3	3
Documentatie	4	4	3	4
Integratie met tools	2	3	2	3

Tabel 2: WRM-vergelijking front-end technologieën

Toelichting:

- **Ervaring:** Ik heb de meeste ervaring met het werken in Angular. Met PHP heb ik ook enige ervaring door projecten voor school. Met React heb ik het minste ervaring.

- **Leercurve:** Angular en ReactJS hebben een vergelijkbare complexiteit om het te leren. PHP is eenvoudiger voor beginners.

- **Herbruikbaarheid van code:** Bij alle drie de technologieën is de herbruikbaarheid van de code een sterk argument om te implementeren.

- **Documentatie:** Bij het zoeken naar documentatie is bij ReactJS en Angular deze het meest uitgebreid. Voor PHP wordt hier minder van beschreven.

- **Integratie met tools:** Bij zowel Angular en ReactJS zijn er goede integratiemogelijkheden. Voor PHP wodt hier minder van beschreven.

Berekening:

- **Angular:**
  - Ervaring: (4 × 4) = 16
  - Leercurve: (2 × 2) = 4
  - Herbruikbaarheid van code: (3 × 3) = 9
  - Documentatie: (4 × 4) = 16
  - Integratie met tools: (2 × 3) = 6

Totaal voor Angular: 16 + 4 + 9 +16 + 6 = 51

- **PHP:**
  - Ervaring: (4 × 3) = 12
  - Leercurve: (2 × 3) = 6
  - Herbruikbaarheid van code: (3 × 3) = 9
  - Documentatie: (4 × 3) = 12
  - Integratie met tools: (2 × 2) = 4

Totaal voor PHP: 12 + 6 + 9 + 12 + 4 = 43

- **ReactJS:**
  - Ervaring: (4 × 2) = 8
  - Leercurve: (2 × 2) = 4
  - Herbruikbaarheid van code: (3 × 3) = 9
  - Documentatie: (4 × 4) = 16
  - Integratie met tools: (2 × 3) = 6

Totaal voor ReactJS: 8 + 4 + 9 + 16 + 6 = 43

Conclusie:

Na het bekijken van de vergelijking van deze technologieën, kan geconcludeerd worden dat Angular de hoogste score heeft behaald. Er was op voorhand een indicatie gegeven dat de voorkeur uitgaat naar Angular of PHP. Uiteindelijk is de keuze gemaakt om Angular te gebruiken voor de realisatie van het eindresultaat van het project, aangezien er meer ervaring met is, de leercurve niet groot is en er een uitgebreide documentatie beschikbaar is.

2.1.2. Styling-framework

Tailwind CSS is een uitbreiding op standaard CSS. Het is een utility-first CSS-framework waarmee websites en webapplicaties gestyled kunnen worden. Het maakt gebruik van herbruikbare CSS-klassen die tijdens de ontwikkeling kunnen worden toegepast. Deze klassen maken het eenvoudiger om een visueel aantrekkelijke webapplicatie te bouwen. (Classens, 16/05/2025)

Bootstrap is een open-source front-endbibliotheek die het ontwerpen van websites vereenvoudigt. (webontwikkeling, 17/03/2018)

Voor- en nadelen styling-framework:

Technologie	*Voordelen*	*Nadelen*
Tailwind CSS	- Het staat bekend om zijn hoge snelheid en efficiëntie tijdens ontwikkeling - Er wordt de flexibiliteit aangeboden om elk element eenvoudig toe te passen - Met een basiskennis van HTML en CSS is dit eenvoudig te begrijpen (LJPc hosting, 16/05/2025)	- Minder uitgebreide documentatie dan die van concurrerende tools. Dit bemoeilijkt het leerproces (Tuple, n.d.)
Bootstrap	- Met een basiskennis van HTML en CSS is dit eenvoudig te begrijpen - Een werkend resultaat kan snel opgebouwd worden (webontwikkeling, 17/03/2018)	- Het is een zware tool, wat gevolgen heeft voor de laadtijd - Alles oogt standaard bij het ontwikkelen, omdat het niet veel aangepast wordt (webontwikkeling, 17/03/2018)

Tabel 3: voor- en nadelen styling-frameworks

Samenvatting:

Tailwind CSS staat bekend om zijn snelheid, flexibiliteit en controle tijdens het ontwikkelen. Het biedt de mogelijkheid om eenvoudig specifieke stijlen toe te passen. Hoewel het relatief eenvoudig is te gebruiken, kan het leerproces vertraagd worden door een mindere documentatie.

Bootstrap is een populair CSS-framework dat ontwikkelaars in staat stelt om met minimale kennis een werkend ontwerp op te bouwen. De keerzijde is echter dat het vaak standaarduitziende gebruikersinterfaces zijn, die vaak mindere prestaties en een lange laadtijd hebben doordat de tool relatief zwaar is.

WRM front-end stylingtools:

Criteria	*Gewicht (0–5)*	*Tailwind CSS*	*Bootstrap*
Ervaring	4	4	3
Leercurve	2	3	3
Laadtijd/performance	3	4	3
Community/documentatie	4	4	4
Designvrijheid	3	4	3

Tabel 4: WRM vergelijking front-end styling-frameworks

Toelichting:

- **Ervaring**: ik heb ervaring met zowel Tailwind CSS als Bootstrap, maar met Tailwind heb ik het meeste gewerkt.

- **Leercurve:** Beide frameworks zijn eenvoudig te leren voor wie beschikt over basiskennis van HTML en CSS.

- **Laadtijd/performance:** Bootstrap scoort minder goed op het gebied van laadtijd en performantie, omdat het een zwaarder framework is dat vaak ongebruikte componente meelaadt. Dit is bij Tailwind CSS niet het geval.

- **Community/documentatie:** Beide frameworks beschikken over een sterke community en uitgebreide documentatie.

- **Designvrijheid:** Bootstrap maakt gebruik van vaste componenten, wat de ontwerpvrijheid kan beperken. Tailwind CSS werkt met utility-klassen, die meer flexibiliteit bieden bij het vormgeven van de interface.

Berekening:

- **Tailwind CSS:**
  - Ervaring: (4 × 4) = 16
  - Leercurve: (2 × 3) = 6
  - Laadtijd/ Performance: (3 × 4) = 12
  - Community/ Documentatie: (4 × 4) = 16
  - Designvrijheid: (3 × 4) = 12

Totaal voor Tailwind CSS: 16 + 6 + 12 + 16 +12 = 62

- **Bootstrap:**
  - Ervaring: (4 × 3) = 12
  - Leercurve: (2 × 3) = 6
  - Laadtijd/ Performance: (3 × 3) = 9
  - Community/ Documentatie: (4 × 4) = 16
  - Designvrijheid: (3 × 3) = 9

Totaal voor Bootstrap: 12 + 6 + 9 + 16 + 9 = 52

Conclusie:

Na vergelijking van de verschillende stylingtools heeft Tailwind CSS het best gescoord. Om die reden kies ik voor Tailwind CSS voor de vormgeving van de webapplicatie. Hier is het meeste ervaring mee. Andere redenen zijn dat de applicatie zo licht mogelijk moet zijn, wat lastig te bereiken is met een zware tool zoals Bootstrap. Daarnaast is het belangrijk dat de applicatie een moderne uitstraling heeft, iets wat met Bootstrap's standaard look moeilijker te realiseren valt.

2.2. Backend

In dit onderdeel wordt een vergelijking gemaakt van de mogelijke technologieën voor het bouwen van de backend. Dit wordt ook uitgebreid toegelicht.

Express.js is een flexibel en minimalistisch framework voor Node.js dat functionaliteiten biedt voor het bouwen van onder andere API's. (GeeksforGeeks, 28/02/2025)

ASP.NET core is een open-source, cross-platform web framework voor het ontwikkelen van onder andere API's. (microsoft, n.d.)

Spring boot is een Java-gebaseerd open-source framework. Het is zeer geschikt voor het ontwikkelen van Java-gebaseerde webapplicaties en RESTful API's. (scrums, 10/09/2024)

Voor- en nadelen backend:

Technologie	*Voordelen*	*Nadelen*
Express.js	- Het is minimaal en flexibel, er hoeft geen specifieke structuur of ontwerppatroon gebruikt te worden - Doordat het minimaal is, is het zeer efficiënt en geschikt voor toepassingen met hoge prestaties - Er wordt een overzichtelijke en flexibele manier om routes in te stellen aangeboden - Optimale samenwerking met verschillende HTTP-verzoeken - Inkomende gegevens van aanvragen kunnen in verschillende formaten worden verwerkt (Taha, 10/06/2023) (GeeksforGeeks, 28/02/2025)	- Middleware onderhoud kan zeer ingewikkeld worden - Een beperkt aantal geavanceerde functionaliteiten (Taha, 10/06/2023) (Dataflair, n.d.)
ASP.NET Core	- Zeer sterke prestaties, doordat de focus hierop lag bij de ontwikkeling. - Het platform is ook beschikbaar voor Linux en macOS. - Gemakkelijk integratie met Cloud omgevingen. (Besseling, 18/02/2025)	- Een complex platform voor beginnende ontwikkelaars binnen de .NET-omgeving. (Besseling, 18/02/2025)

Spring Boot	- Uitstekende schaalbaarheid met het spring-ecosysteem - Het platform is ook beschikbaar voor Linux en macOS - Eenvoudige integratie van Spring Boot met het Spring-ecosysteem (Nastase, 10/08/2022) (scrums, 10/09/2024)	- Vanwege de opiniegerichte stijl zijn er veel ongebruikte afhankelijkheden, wat de bestandgrootte van de implementatie vergroot - Verbruikt meer geheugen en systeembronnen, waardoor er meer overhead is in het gebruik van resources. Dit leidt tot extra kosten

Tabel 5: voor- en nadelen backends

Samenvatting:

Express.js is een minimalistisch en flexibel framework, waardoor er geen specifieke structuur of ontwerppatroon nodig is. Het biedt hoge efficiëntie voor toepassingen die hoge prestaties verwachten. Verder biedt het ook een overzichtelijke manier om routes in te stellen en heeft het een optimale samenwerking met verschillende HTTP-verzoeken. Express.js kan inkomende gegevens van aanvragen in verschillende formaten verwerken. Het onderhoud van middleware kan wel ingewikkeld worden en het heeft daarnaast maar een aantal geavanceerde functionaliteiten.

ASP.NET Core is een framework met zeer sterke prestaties, door de focus hierop bij de ontwikkeling. Het is beschikbaar op voor zowel Windows, Linux als macOS, en biedt eenvoudige integratie met Cloud omgevingen. Het framework kan echter complex zijn voor beginnende ontwikkelaar binnen de .NET-omgeving.

Spring Boot is schaalbaar en biedt eenvoudige integratie met het Spring-ecosysteem. Het is ook beschikbaar op Linux en macOS. De opiniegerichte stijl kan echter leiden tot ongebruikte afhankelijkheden, wat de bestandsgrootte vergroot. Daarnaast verbruikt het ook meer geheugen.

WRM backend:

Criteria	*Gewicht (0-5)*	*Express.js*	*ASP.NET core*	*Spring boot*
Ervaring	4	2	4	3
Leercurve	3	4	3	3
Community/documentatie	4	4	4	4
Compatibiliteit met front-end	4	4	3	3
Schaalbaarheid	3	3	4	4

Tabel 6: WRM-vergelijking backend

Toelichting:

- **Ervaring:** Ik heb de meeste ervaring met ASP.NET core, aangezien ik hier al veel met gewerkt heb. Met Java en Spring Boot heb ik ook redelijk wat ervaring. Express.js heb ik weinig tot geen ervaring.

- **Leercurve:** Express.js is de eenvoudigste technologie om te leren, terwijl er voor ASP.NET core en Spring Boot meer kennis vereist wordt.

- **Community/documentatie:** Alle drie de frameworks beschikken over een sterke community en uitgebreide documentatie. Doordat ASP.NET Core ontwikkeld wordt door Microsoft, is de documentatie over het algemeen beter gestructureerd.

- **Compatibiliteit met front-end:** Aangezien het project ontwikkeld wordt met Angular en dus gebruikmaakt van TypeScript, is Express.js een logische keuze door de goede samenwerking met JavaScript. ASP.NET Core en Spring Boot bieden ook

- **Schaalbaarheid:** Spring Boot en ASP.NET core zijn beide goede keuzes voor schaalbare oplossingen. Express.js is ook zeer schaalbaar, maar verwacht extra configuratie om hetzelfde te bereiken als de andere twee.

Berekening:

- **Express.JS:**
  - Ervaring: (4 × 2) = 8
  - Leercurve: (3 × 2) = 6
  - Community/documentatie: (4 × 4) = 16
  - Compatibiliteit met front-end: (4 × 4) = 16
  - Schaalbaarheid: (3 × 3) = 9

Totaal voor Express.js: 8 + 6 + 16 + 16 + 9 = 55

- **ASP.NET core:**
  - Ervaring: (4 × 4) = 16
  - Leercurve: (3 × 3) = 9
  - Community/documentatie: (4 × 4) = 16
  - Compatibiliteit met front-end: (4 × 3) = 12
  - Schaalbaarheid: (3 × 4) = 12

Totaal voor ASP.NET Core: 16 + 9 + 16 + 12 + 12 = 65

- **Spring Boot:**
  - Ervaring: (4 × 3) = 12
  - Leercurve: (3 × 3) = 9
  - Community/documentatie: (4 × 4) = 16
  - Compatibiliteit met front-end: (4 × 3) = 12
  - Schaalbaarheid: (3 × 4) = 12

Totaal voor Spring Boot: 12 + 9 + 16 + 12 + 12 = 61

Conclusie:

Na het vergelijken komt ASP.NET core eruit met de hoogste score. Hoewel dit de hoogste score heeft is dit niet de backend die gebruikt zal worden. Aangezien express.js het best compatibel is met de front-end, er bij de stage ook een voorkeur is voor express.js en hier een template voor was vanuit de stage, wordt deze technologie gekozen.

2.3. Database

In dit deel worden de verschillende database mogelijkheden vergeleken met elkaar. Dit wordt uitbundig toegelicht. De keuze voor de database ligt al wel vast, omdat Deca gebruikmaakt van FileMaker. Dit is een low-code ontwikkelplatform waarmee eenvoudig data gestructureerde applicaties kunnen gebouwd worden.

FileMaker is een low-code platform gebruikt om databaseapplicaties te maken. Op een eenvoudige manier kunnen er door middel van de simpele gebruikersinterface applicaties ontwikkeld worden. Het kan zowel standalone als client-server gebruikt worden. (Wikipedia, 08/07/2023) (The Support Group, 08/09/2020)

MySQL is “open-source relationeel databasesysteem” (joogi, n.d.). Het wordt ondersteund door het bedrijf Oracle en is één van de vele populaire systemen dat helpt bij het opslagen en beheren van gegevens.

MongoDB is een NoSQL-database dat op een documentgeoriënteerde manier te werk gaat. Het wordt gebruikt om gegevens op te slaan in grote volumes. Er wordt niet tewerk gegaan op de traditionele relationele database manier met tabellen en rijen, maar er wordt gebruik gemaakt van collecties en documenten. (cube, n.d.)

Technologie	*Voordelen*	*Nadelen*
FileMaker	- Er kan snel een complex databasesysteem met opgezet worden. Iets waar met een andere applicatie meer tijd voor nodig is - Een integratie maken is simpel, aangezien het een ingebouwde API heeft - Er zit een front-end systeem ingebouwd	- Het is een nichemarkt, waardoor er eerder een beperkte gemeenschap is - De koppeling met webapplicaties is trager - Een betalende licentie is noodzakelijk, wat bij alternatieven niet altijd het geval is
MySQL	- Gratis in gebruik, aangezien het een open-source relationeel databasesysteem is - Het kan gebruikt worden voor zowel kleine als grote projecten - Betrouwbare prestaties voor zowel een kleine en grote hoeveelheid gegevens, alsook grote berekeningen (joogi, n.d.) (kinsta, 16/06/2023)	- Geschikt voor meeste projecten, maar er zijn beperkingen wat betreft de maximale databasegrootte - Beperkingen met geavanceerde functionaliteit
MongoDB	- Flexibele structuur, doordat er geen vaste tabellen zijn - Ideaal voor applicaties waar de datavorm vaak verandert - Ingebouwde schaalbaarheid, waardoor de database kan meegroeien met de groei van de data. Zowel horizontale als verticale schaalbaarheid wordt ondersteund - Het staat bekend voor zijn hoge prestaties en is zeer geschikt voor grote, groeiende en veeleisende applicaties (cube, n.d.)	- Voor gegevensopslag wordt er veel geheugen gebruikt - Bij het opslagen is er een limiet in documentgrootte van 16 MB (Mertz, n.d.)

Tabel 7: voor- en nadelen databases

Samenvatting:

FileMaker is een snel te implementeren databasesysteem met een ingebouwde API en front-end systeem, perfect voor het eenvoudig en snel opzetten van complexe systemen. Het heeft echter wel een beperkte gemeenschap en tragere koppeling met webapplicaties. Daarnaast is een betalende licentie noodzakelijk, wat niet zo is bij alternatieven.

MySQL is een gratis relationeel databasesysteem, dat geschikt is voor zowel grote als kleine projecten. Het is een systeem dat betrouwbare prestaties biedt en is algemeen geschikt voor meeste projecten. Er zijn echter wel beperkingen op het vlak van de maximale databasegrootte en de geavanceerde functionaliteit.

MongoDB is een NoSQL-database die een flexibele en schaalbare structuur aanbiedt, ideaal voor applicaties waar de datavorm vaak verandert. Het biedt hoge prestaties en is geschikt voor grote en groeiende applicaties. Het verbruikt wel veel geheugen en heeft een limiet op documentgrootte.

WRM-databases:

Criteria	*Gewicht (0-5)*	*FileMaker*	*MySQL*	*MongoDB*
Ervaring	4	1	4	2
Leercurve	3	4	3	2
Schaalbaarheid	4	2	4	5
Flexibiliteit	3	2	3	5
Community/documentatie	3	2	5	4

Tabel 8: WRM-vergelijking databases

Toelichting:

- **Ervaring:** Met MySQL heb ik al behoorlijk veel gewerkt. Dit is niet het geval met de andere twee. Met MongoDB ben ik maar een paar keer kort in aanmerking gekomen en met FileMaker nog nooit.

- **Leercurve:** FileMaker is een low-code platform en dus best makkelijk om te leren. MySQL is niet het eenvoudigste om te leren, maar hier heb ik al wel kennis van. Van MongoDB is mijn kennis zeer weinig, dus is de leercurve hier vrij groot.

- **Schaalbaarheid:** MongoDB is sterk in schaalbaarheid, net zoals MySQL. FileMaker is minder geschikt voor grote bedrijven en dus minder schaalbaar.

- **Flexibiliteit:** De datastructuur bij MongoDB is anders en biedt meer vrijheid. MySQL is zeer gestructureerd en FileMaker is beperkt tot wat de UI/lay-out toelaat.

- **Community/documentatie:** Voor zowel MongoDB en MySQL is er een grote community en veel over terug te vinden. Dit is niet het geval voor FileMaker, waar beide eerder beperkt zijn.

Berekening:

- **FileMaker:**
  - Ervaring: (4 × 1) = 4
  - Leercurve: (3 × 4) = 12
  - Schaalbaarheid: (4 × 2) = 8
  - Flexibiliteit: (3 × 2) = 6
  - Community/ Support: (3 × 2) = 6

Totaal voor FileMaker: 4 + 12 + 8 + 6 + 6 = 36

- **MySQL:**
  - Ervaring: (4 × 4) = 16
  - Leercurve: (3 × 3) = 9
  - Schaalbaarheid: (4 × 4) = 16
  - Flexibiliteit: (3 × 3) = 9
  - Community/ Support: (3 × 5) = 15

Totaal voor MySQL: 16 + 9 + 16 + 9 + 15 = 65

- **MongoDB:**
  - Ervaring: (4 × 2) = 8
  - Leercurve: (3 × 2) = 6
  - Schaalbaarheid: (4 × 5) = 20
  - Flexibiliteit: (3 × 5) = 15
  - Community/ Support: (3 × 4) = 12

Totaal voor MongoDB: 8 + 6 + 20 + 15 + 12 = 61

Conclusie:

MySQL komt bij de vergelijking naar boven met de hoogste score. Hier heb ik ook al de meeste ervaring mee en is ook een slimme keuze voor mijn project. Maar bij mijn stage-bedrijf maken ze gebruik van FileMaker, dus wordt deze software gekozen voor het project.

3. Deca – Klantenportaal

In dit hoofdstuk wordt de realisatie en het resultaat beschreven van het stageproject. Er wordt gekeken naar de verschillende functionaliteiten die teruggevonden kunnen worden in het portaal.

3.1. Opzet van de toepassing

De toepassing is opgedeeld in drie verschillende onderdelen die met elkaar samenwerken. Er is de front-end, backend en FileMaker (de database).

Het eerste onderdeel is de front-end, deze heb ik opgezet in Angular in Visual Studio Code. Ook heb ik Tailwind geïnstalleerd om ervoor te zorgen dat alles er mooi en modern uitziet. De front-end is ook nog eens opgedeeld in verschillende delen. Zo bestaat het uit de componenten die samen één geheel vormen, waar je naar kan kijken. Deze kunnen ook hergebruikt worden en dit is gemakkelijk om zo te werk te gaan. Het volgende deel zijn de services. Hier worden de API-calls in gedefinieerd, die data sturen en ophalen naar de backend. Zo kunnen er zaken aangepast, verwijderd, opgehaald en nieuwe data doorgestuurd worden. Het laatste deel zijn de models die helpen met het structureren van de data. Hier kan automatische type controle met gedaan worden tijdens het programmeren. Ook wordt het gebruikt om duidelijk te maken doe de data eruitziet bij het ophalen, updaten, posten van ervan.

Het tweede onderdeel is de backend. Dit is een express server die op Node.js draait. Deze werkt als tussenlaag/backend tussen de front-end en mijn database, FileMaker. De server ontvangt de calls gedefinieerd in de services in de front-end en zet deze om in een juiste call naar FileMaker. Op deze manier kan de data uit de database doorgegeven worden naar de front-end. De tussenlaag zal ervoor zorgen dat dit op een veilige en goede manier gebeurt. De backend bestaat ook nog uit verschillende delen. Het bevat ook models en services. Die hetzelfde doen als in de front-end, structuur brengen aan de data en in de services bevinden zich de API-calls naar FileMaker. Ook bevat de express server router files. Voor elk groot onderdeel van de toepassing heb ik hier een apart bestand voor aangemaakt. Hierin staan alle calls die in de front-end kunnen gedaan worden. Elke call heeft een endpoint, krijgt van de front-end enkele parameters en voert een bepaalde logica uit.

Het derde onderdeel is FileMaker. Dit is de toepassing die gebruikt wordt binnen Deca om al hun data in op te slaan. Ik heb een test database gekregen, waar ik al mijn tabellen in heb staan. In deze tabellen wordt dan alles opgeslagen om het klantenportaal te kunnen voorzien van alle data. FileMaker biedt de mogelijkheid om API-calls te doen. Ook kan men hier lay-outs in maken, wat eigenlijk een visualisatie is van data. Op deze manier is het niet nodig om een extra applicatie laag te bouwen voor een interface. Data kan aangepast worden door de medewerkers van Deca op deze lay-out.

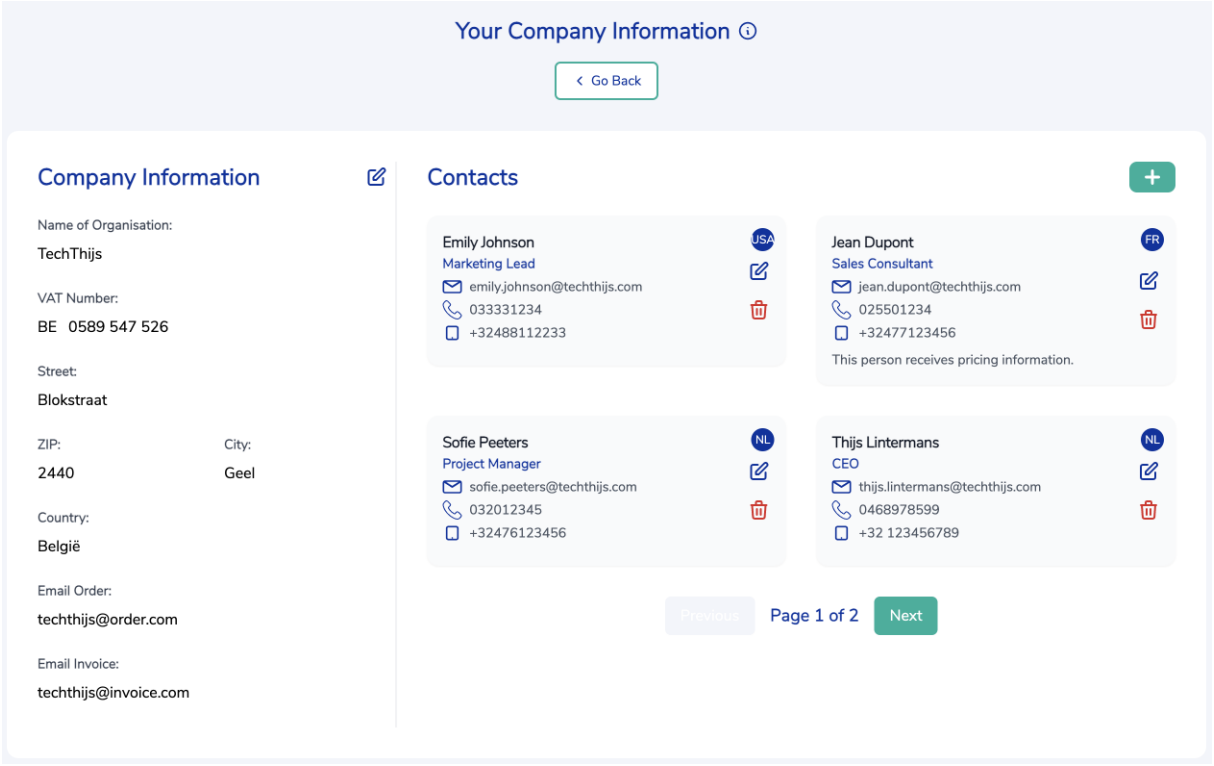
3.2. Functionaliteiten

In dit onderdeel van het hoofdstuk bespreek ik de functionaliteiten van de applicatie dat ik ontwikkeld heb. Deze functionaliteiten worden uitgebreid uitgelegd en waar nodig wordt de link met de backend en FileMaker gelegd en besproken.

3.2.1. Hoofddoel

Het hoofddoel van de webapplicatie is om klanten van Deca een gebruiksvriendelijk portaal aan te bieden waarin zij een project-checklist kunnen invullen. Deze vormt de basis voor het inschatten van de scope van een nieuw project en kan er ook een projectvoorstel volgen door Deca. De toepassing maakt het mogelijk om de checklist in te vullen zonder dat een gebruiker hoeft in te loggen, wat handig is voor nog niet gekende klanten of klanten zonder inloggegevens tot het portaal. Klanten die zich wel kunnen aanmelden, krijgen toegang tot een portaal waar een overzicht beschikbaar is voor hun eerder ingediende projecten (checklistverzamelingen). Deze kunnen ze dan ook op elke moment bekijken of aanpassen. Uiteindelijk is het de bedoeling dat zo'n checklist door zowel de klant als Deca gevalideerd en goedgekeurd wordt. Zo kan er aan een project kan begonnen worden zonder fouten en waar beide partijen volledig op één lijn zitten, alvorens het project van start gaat. De toepassing speelt hierbij een centrale rol als communicatie- en documentatieplatform tussen de klant en Deca.

3.2.2. Company Information



Figuur 1: company information webpagina

Op de home-pagina heeft de gebruiker de optie om op de tegel “Company Information” te klikken. Dan wordt er naar deze pagina genavigeerd. Deze pagina is één van de kernpagina’s van het klantenportaal. Na het inloggen biedt deze pagina de gebruiker de mogelijkheid om bedrijfsspecifieke informatie terug te vinden en te beheren.

Er zijn op deze pagina twee onderdelen. Aan de linkerzijde is er de mogelijkheid om belangrijke informatie te raadplegen. Dit is algemene informatie over het bedrijf die belangrijk is voor Deca om bijvoorbeeld de aankoop mail, mail voor invoices... te weten. Deze data kan door de gebruiker/klant van Deca op deze pagina aangepast worden. Niet alles is aanpasbaar, zoals het land en de btw-nummer, want als dit verandert, worden ze als een nieuwe klant geregistreerd.

Aan de rechterzijde bevinden zich de contacten. Bestaande contacten kunnen bewerkt worden als de informatie veranderd is, verwijderd worden wanneer ze niet langer relevant zijn of aangemaakt worden. Er kan bij elk contact de naam, functie, mail, taal, telefoon- en gsm-nummer ingevuld/aangepast worden. Ook is er de optie om bij een contact aan te duiden of deze persoon prijs informatie moet krijgen. Door dit onderdeel kan er gemakkelijk een lijst opgebouwd worden van aanspreekpunten binnen hun organisatie. Zo gaat er ook geen kostbare tijd verloren bij het zoeken naar een juist contact voor bepaalde communicatie.

Front-end:

```
ngOnInit(): void {
  const userSessionToken = sessionStorage.getItem('userSessionToken') || '';

  this.contactsERPService.getAllCompanyInfo(userSessionToken).pipe(
```

Figuur 2: frontend code inladen bedrijfsinformatie

```
getAllCompanyInfo(userSessionToken: string): Observable<any> {
  const url = environment.express_url + "/companyInfo/getCompanyInfoOfUser";
  return this.http.post<any>(url, { userSessionToken });
}
```

Figuur 3: service in frontend code, inladen bedrijfsinformatie

Backend:

```
companyInfo.post('/getCompanyInfoOfUser', async function (req, res) {
    const { userSessionToken } = req.body;

    if(!userSessionToken) {
        return res.status(400).json({ error: "Missing user session token "});
    }

    try {
        const token = await logInDB();
        console.log("Logged in!");

        const foundSessionTokenUser = await findSessionToken(userSessionToken, token);
        console.log('Session token found: ', foundSessionTokenUser);

        const userData = foundSessionTokenUser.response.data[0];

        if (!userData || !userData.fieldData || !userData.fieldData._k2_customerid) {
            return res.status(400).json({ error: "Invalid session or missing contact ID" });
        }

        const user = new Credentials(userData);
        console.log('user:', user)

        const contactErpId = user.fieldData._k2_customerid;
        console.log('contactErpId', contactErpId)

        const companyInfo = await getContactErp(contactErpId, token);
        const rawContact = companyInfo?.response?.data?.[0];
        const contactErp = new ContactsErp(rawContact);
        console.log('contactErp', contactErp);

        const companyContacts = await getRelations(contactErpId, token);
        const relationRecords = companyContacts?.response?.data || [];
        const relations = relationRecords.map(record => new Relation(record));
        console.log('relations', relations);
```

Figuur 4: backend code, inladen bedrijfsinformatie (deel1)

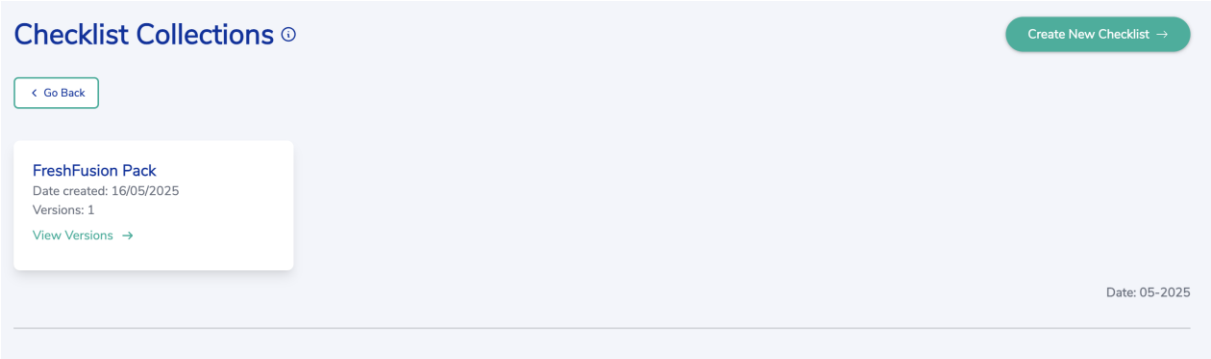
```
        const zLanguagesRaw = await getLanguages(token);
        const languageRecords = zLanguagesRaw?.response?.data || [];
        const languages = languageRecords.map(record => new ZLanguages(record));
        console.log('languages', languages);


        console.log(languages)

        await logOutDB();
        console.log('Logged out!');

        return res.status(200).json({
            message: "Session validated and company info retrieved successfully",
            user: user,
            contactErp: contactErp,
            relations: relations,
            languages: languages
        })
    } catch (error) {
        console.error('Error validating or getting company info: ', error.message);
        return res.status(500).json({ error: error.message });
    }
```

Figuur 5: backend code inladen bedrijfsinformatie(deel2)

Dit is het belangrijkste deel code van deze pagina, omdat er geen data kan getoond worden als dit niet goed werkt. Dit betekent dat er ook niets kan aangepast worden en belangrijke info niet beschikbaar is voor de klant om na te kijken. Hier gebeurt eerst in de front-end de functie, ngOnInit, wat betekent dat dit uitgevoerd wordt bij het laden van de pagina. In deze functie wordt eerst de sessie token van de gebruiker uit de sessie opslag gehaald en doorgegeven aan een API-call in de service: contactsERPService. Dit zal ervoor zorgen dat er een call gedaan wordt naar de backend.

In de backend zien we dan het gedefinieerde endpoint dat opgeroepen wordt. In deze logica wordt er eerst gekeken of de token daadwerkelijk is meegestuurd. Vervolgens wordt er nog een check gedaan of deze token kan gevonden worden in de database. Zo ja dan betekent dit dat de gebruiker is ingelogd en de data mag opgehaald worden. Dus al de rest van de code die volgt haalt uit verschillende tabellen data op. Zo worden uiteindelijk de gebruiker, het contactErp (bedrijfsinformatie), de relations (contacten) en de verschillende talen teruggestuurd naar de front-end. Zo kan deze data getoond worden op deze pagina.

3.2.3. Checklist dashboard



Figuur 6: checklistverzamelingen webpagina

Het checklist dashboard vormt het centrale overzicht voor gebruikers die ingelogd zijn. Op deze pagina worden alle checklistverzamelingen weergegeven. Elke verzameling vertegenwoordigt een project, waarin één of meerdere versies van een checklist kunnen bestaan. De verzamelingen zijn gesorteerd op maand, waarbij de meest recente maand bovenaan wordt weergegeven. Dit zorgt ervoor dat nieuwe of actieve projecten gemakkelijk terug te vinden zijn. Naast het overzicht is er een knop aanwezig waarmee gebruikers eenvoudig een nieuwe checklist kunnen opstarten. Hiermee wordt een nieuw project aangemaakt. Deze pagina biedt dus een gestructureerd en overzichtelijk uitgangspunt voor het beheren van alle lopende en historische projecten binnen het klantenportaal.

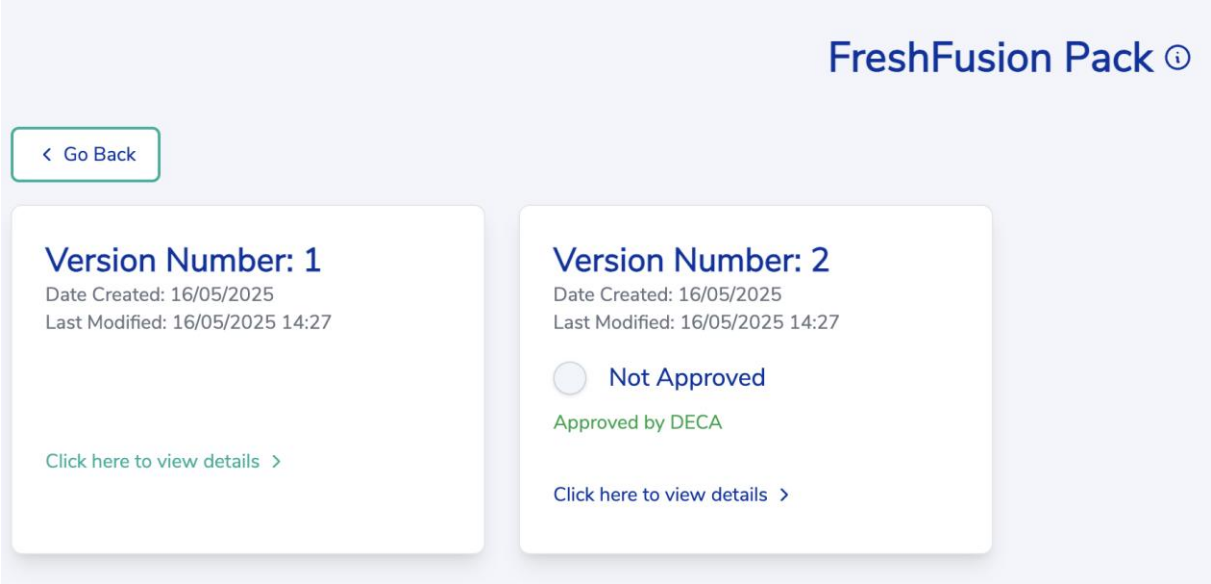
```
const contactErpId = user.fieldData._k2_customerid;

const collections = await getAllChecklistCollectionsForUser(contactErpId, token);
const rawCollections = collections?.response.data || [];
const checklistCollections = rawCollections.map(record => new ChecklistCollection(record));
```

Figuur 7: frontend code, checklistverzamelingen inladen

Het belangrijkste stukje code voor deze pagina is dit deel uit de backend. Dit zorgt ervoor dat de juiste verzamelingen kunnen opgehaald worden voor de gebruiker dat is ingelogd. Hier verloopt het proces hetzelfde als bij de bedrijfsinformatie pagina. Eerst wordt er bij het laden van de pagina een call gedaan naar de backend. Hier wordt er dan eerst geverifieerd wie ingelogd is en dan worden zijn verzamelingen aan de hand van de gebruiker zijn id opgehaald.

3.2.4. Checklist versies

12. Klantenportaal: Checklistversies webpagina



Figuur 8: Checklistversies webpagina

Op de checklistversies-pagina krijgt de gebruiker een overzicht van alle versies die gekoppeld zijn aan één specifiek project (checklistverzameling). Elke versie weerspiegelt een bepaalde staat van de checklist binnen het projecttraject.

De eerste versie van een checklist wordt standaard opgesteld door de klant zelf. Wanneer Deca op hun platform, FileMaker, wijzigingen aanbrengt wordt er automatisch een nieuwe versie toegevoegd aan de verzameling. Zo ontstaat er een chronologische en transparante versiegeschiedenis.

De meest recente versie van de checklist kan vervolgens worden goedgekeurd door zowel de klant als door Deca. Wanneer beide partijen goedkeuring geven, wordt dit duidelijk weergegeven en wordt een versie als definitief beschouwd.

Tot slot is het via deze pagina ook mogelijk om een specifieke versie te openen en de inhoud ervan te bekijken. Hier wordt meer over uitgelegd in het deel ‘Checklist’.

Front-end:

```
toggleApproval(checklistVersion: ChecklistVersion): void {

    let updatedFlag: number;

    if (checklistVersion.fieldData.flag_approved != 1) {
      updatedFlag = 0;
    } else {
      updatedFlag = 1;
    }

    checklistVersion.fieldData.flag_approved = updatedFlag;

    this.checklistVersionService.toggleApprovalState(checklistVersion).pipe(
      concatMap(() => {

        return of(null);
      }),
    ).subscribe({
      next: () => {
        this.ngOnInit();
      },
      error: (error) => {
        console.error('An error occurred: ', error);
      }
    });
  }
```

Figuur 9: front-end code, goedkeuring checklist

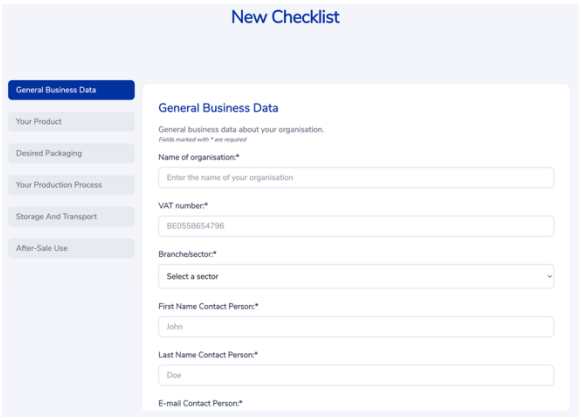
Backend:

```
checklist.post('/toggleApproval', async function (req, res) {
    const { checklistVersion } = req.body;

    if (!checklistVersion?.fieldData || !checklistVersion?.recordId) {
        return res.status(400).json({ error: 'Missing required fields.' });
    }

    try {
        const token = await logInDB();
        console.log('Logged in!');

        const version = new ChecklistVersion({
            fieldData: checklistVersion.fieldData,
            recordId: checklistVersion.recordId
        });

        console.log('Version: ', version);

        const toggleApproval = await toggleChecklistApproval(version, token);
        console.log('Toggle approval result: ', toggleApproval);

        await logOutDB();
        console.log('Logged out.');

        return res.status(200).json({ message: 'Approval state toggled successfully.' });
    } catch (error) {
        console.error('Error toggling approval:', error.message);
        return res.status(500).json({ error: error.message });
    }
});
```

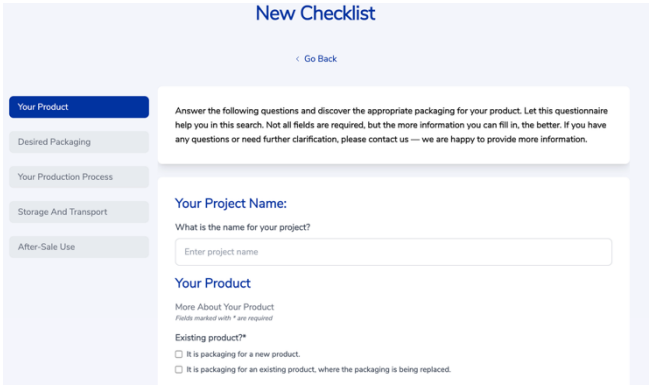
Figuur 10: backend code, goedkeuring checklist

Een belangrijke functionaliteit op deze pagina is het goedkeuren van een checklist. Een gebruiker kan een checklist terug herbekijken en als deze volgens hem/haar dan goed is, kan deze goedgekeurd worden. Dan moet deze door Deca zelf nog wel eens nagekeken worden voor een finale goedkeuring. Telkens er op de knop gedrukt wordt, zal de functie ‘toggleApproval’ uitgevoerd worden. Hier wordt de checklistversie doorgegeven als parameter. Deze bevat een variabele die weergeeft of een checklist al dan niet is goedgekeurd. Vervolgens wordt er gekeken of deze al goedgekeurd is. Als dit het geval is, dan wordt de ‘flag_approved’ terug op nul gezet, wat betekent dat deze terug op niet-goedgekeurd gezet wordt. Dit werkt ook andersom, als het op niet-goedgekeurd staat wordt de variabele op 1 of goedgekeurd gezet. Er wordt nul en één gebruikt, omdat dit in FileMaker handiger werkt. Na deze check wordt er een API-call gedaan naar de backend. Het enige wat er dan nog in de front-end wordt gedaan is het opnieuw laden van de data, na een antwoord van de backend. In de backend wordt er een call gedaan naar de database om van deze versie, de variabele ‘flag_approved’ aan te passen naar wat er is doorgestuurd als parameter naar de backend.

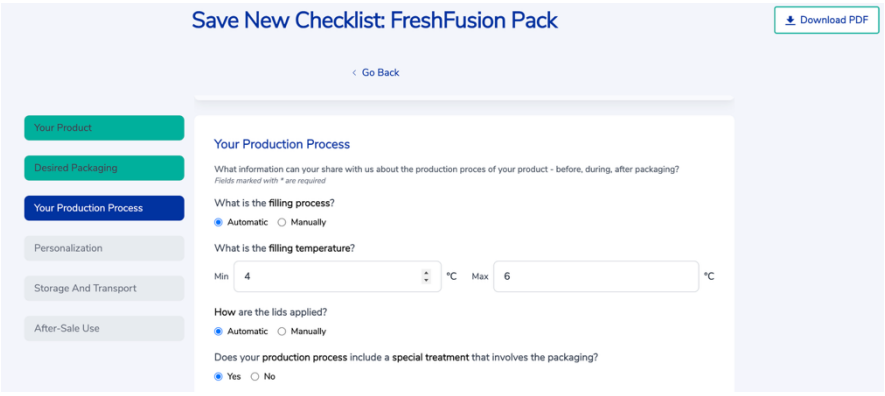
3.2.5. Checklist



Figuur 11: nieuwe checklist, niet-ingelogde gebruiker



Figuur 12: nieuwe checklist, ingelogde gebruiker



Figuur 13: ingevulde checklist, ingelogde gebruiker

Op de checklist pagina zijn er veel functionaliteiten terug te vinden, aangezien dit ook het hoofddoel is van het project dat zeker moest werken. In de ontwikkeling van deze pagina is er veel tijd en moeite geïnvesteerd. Op het eerste zicht lijkt het een simpel formulier, maar in de achtergrond gebeurt er veel om te zorgen dat alles vlot verloopt.

Op deze pagina is het mogelijk om als gebruiker een checklist in te vullen. Dit is mogelijk voor ingelogde gebruikers, maar ook voor niet-ingelogde gebruikers. Als er niet is ingelogd komt er nog een onderdeel bij in de chekclist. Een onderdeel waar meer informatie wordt gegeven over het bedrijf dat de checklist invult.

Buiten het invullen en navigeren doorheen de checklist, is het ook mogelijk voor ingelogde gebruikers om een al ingevulde checklist aan te passen. Bij het saven van deze checklist wordt deze checklist ook aangepast in de database.

Verder kan er op deze pagina ook een pdf gedownload worden van de checklist, om zo het gemakkelijk te maken voor de klant om dit ergens nog op te kunnen slaan.

Inladen checklist

```
ngOnInit(): void {
  this.sortedSectorList = [...this.sectorList].sort();
  const userSessionToken = sessionStorage.getItem('userSessionToken') || '0';

  this.route.paramMap.pipe(
    switchMap((params) => {
      this.checklistId = params.get('checklistId') ?? '';
      this.checklistCollectionRecordId = params.get('checklistCollectionId') ?? '';

      return this.checklistService.getChecklistData(userSessionToken, this.checklistId, this.checklistCollectionRecordId);
    })
```

Figuur 14: front-end code, inladen checklist (deel1)

Het eerste deel zorgt ervoor dat de lijst met sectoren, die nodig is in het begin van de checklist bij het niet inloggen, gesorteerd wordt. Vervolgens wordt er een call gedaan naar de backend om de data op te halen met enkele parameters die meegegeven worden. De parameters die worden meegegeven zijn de eventuele checklistId en checklistcollectionId. Als het een aanpassing is aan een bestaande checklist wordt er op de checklist versie pagina op een versie gedrukt en met de route deze parameters meegegeven.

Als de call succesvol is, dan wordt het antwoord opgeslagen in de variabele data. Uit deze data worden dan eerst al de sectie types en de contactErpId gehaald om te zien of de persoon is ingelogd of niet.

```
).subscribe({
  next: (response) => {
    const data = response as any;

    this.sectionTypes = data.sectionTypes;

    this.contactERPID = data.contactERPID;

    if(this.contactERPID !== null){
      this.rightSessionToken = true;
      this.activeSection = 1;
    }

    this.isNewChecklist = data.isNewChecklist;
    this.checklistCollection = data.checklistCollection;
    this.subseries = data.subseries;
    this.sectionTransport.fieldData.averageTimeTransportUnit = 'hr';

    if(this.isNewChecklist && this.rightSessionToken){
      this.notificationsService.setTriggerNotif(true);
      this.credentialsService.setAuthenticated(true);
    }
```

Figuur 15: front-end code, inladen checklist (deel 2)

Daarna wordt er gekeken of dit een nieuwe checklist is of niet en worden de nodige gegevens al gevuld met data waar het niet uitmaakt of het een nieuwe checklist is of niet.

Verder wordt er als het een nieuwe checklist is en de persoon is ingelogd de meldingen ingeladen en gezorgd dat de menubalk weet dat er iemand is ingelogd.

```
if(!this.isNewChecklist){
  this.checklistVersion = data.checklistVersionInfo;
  console.log('Version get', this.checklistVersion)
  this.sectionProductInfo = data.productInfo;
  this.sectionPackaging = data.packaging;
  this.sectionProductionProcess = data.productionProcess;
  this.sectionPersonalisation = data.personalisation;
  this.designs = data.designs;
  this.sectionTransport = data.transport;
  this.sectionAfterSale = data.afterSale;
  this.sectionTransport.fieldData.averageTimeTransportUnit = 'hr';

  this.copySectionProductInfo = JSON.parse(JSON.stringify(this.sectionProductInfo));
  this.copySectionPackaging = JSON.parse(JSON.stringify(this.sectionPackaging));
  this.copySectionProductionProcess = JSON.parse(JSON.stringify(this.sectionProductionProcess));
  this.copySectionPersonalisation = JSON.parse(JSON.stringify(this.sectionPersonalisation));
  this.copyDesigns = JSON.parse(JSON.stringify(this.pendingFiles));
  this.copySectionTransport = JSON.parse(JSON.stringify(this.sectionTransport));
  this.copySectionAfterSale = JSON.parse(JSON.stringify(this.sectionAfterSale));
```

Als het geen nieuwe checklist is dan worden alle secties gevuld met de data teruggegeven door de backend. Hierna wordt er hier een kopie van gemaakt om bij het drukken op de save knop te vergelijken met de ingevulde gegevens. Zo kan bepaald worden of er wijzigingen zijn aangebracht door de gebruiker.

Figuur 16: front-end code, inladen checklist (deel 3)

Posten/updaten checklist

Bij het indienen of updaten van de checklist wordt er eerst gecontroleerd of de checklist correct is ingevuld. Op basis van de antwoorden worden andere vragen aangepast. Als een keuze wordt gemaakt en er al antwoorden zijn gegeven, maar later een andere keuze wordt geselecteerd, moeten de eerder ingevulde antwoorden worden teruggezet naar de standaardwaarde.

```
let filesArray = [];

for (const file of this.pendingFiles) {
  const fileToUpload = await this.convertFileToBase64(file);
  filesArray.push(fileToUpload);
}
```

Vervolgens worden de mogelijke bestanden die de gebruiker kan uploaden om hun product te personaliseren omgezet naar een base64 bestand. Op deze manier kan het doorgestuurd worden naar de backend.

Figuur 17: front-end code, omzetten bestand naar base64

```
let checklistRequest$;

if (this.isNewChecklist) {
  checklistRequest$ = this.checklistService.postChecklistData(
    this.rightSessionToken,
    this.contactERPID,
    this.sectionGenData,
    this.checklistCollection,
    this.checklistVersion,
    this.sectionProductInfo,
    this.sectionPackaging,
    this.sectionProductionProcess,
    this.sectionPersonalisation,
    filesArray,
    this.sectionTransport,
    this.sectionAfterSale
  );
} else {
  checklistRequest$ = this.checklistService.updateChecklistData(
    this.contactERPID,
    this.checklistId,
    this.checklistCollection,
    this.checklistVersion,
    this.sectionProductInfo,
    this.sectionPackaging,
    this.sectionProductionProcess,
    this.sectionPersonalisation,
    filesArray,
    this.sectionTransport,
    this.sectionAfterSale
  );
}
```

Vervolgens wordt er gekeken of de checklist een nieuwe ingevulde checklist is. Als dit het geval is, betekent het dat er een post wordt gestuurd naar de backend met alle nodige parameters en secties van de checklist om deze in de database te kunnen zetten.

Als dit geen nieuwe checklist is, wordt er een update gedaan van de checklist met ook hier alle nodige parameters en secties van de checklist om deze in de database aan te kunnen passen.

Bij een fout wordt er ook een foutmelding gegeven.

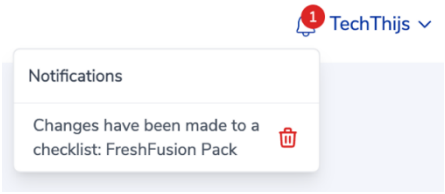
Figuur 18: front-end code, posten of updaten checklist

```
error: (error) => {
  console.error('An error occurred: ', error);

  if (error.status === 400 && error.error?.details) {
    error.error.details.forEach((detail: ErrorDetail) => {
      this.showToast(`⚠ ${detail.name}: ${detail.message}`, 'warning');
    });

    this.submitting = false;
    return;
  }

  this.showToast('❌ Something went wrong while submitting the checklist.');
}
```

Figuur 19: front-end code, foutmelding checklist

3.2.6. Notifications

Om de gebruiker op een eenvoudige manier te informeren dat er wijzigingen zijn aangebracht aan de checklist van één van zijn/haar projecten is er een notificatiesysteem geïmplementeerd. Wanneer een medewerker bij Deca wijzigingen aanbrengt in een checklist en deze opslaat, wordt er via een script in FileMaker een notificatie aangemaakt.



Figuur 20: voorbeeld melding

Frontend code:

```
this.notificationsService.setTriggerNotif(true);
```

Figuur 21: front-end code, trigger inladen meldingen

In de frontend is bovenstaande regel code opgenomen op alle pagina's. Deze regel zorgt ervoor dat een deeltje code in de menu component getriggerd wordt. Deze gaag dan de notificaties ophalen uit de database. Op deze manier wordt er bij het laden van de verschillende pagina's altijd gecheckt of de notificaties er zijn en kunnen ze zo snel mogelijk getoond worden.

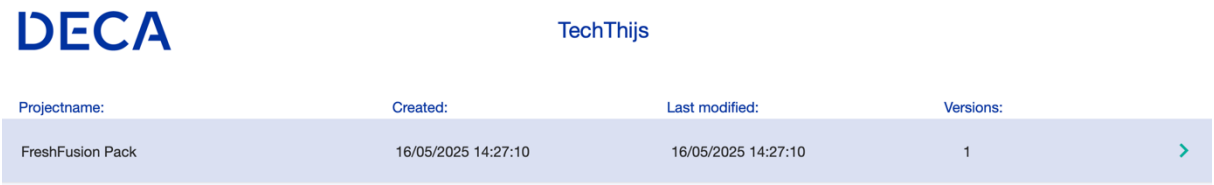
FileMaker script:

```
Ga naar lay-out [ "Notifications" (Notifications) ; Animatie: Geen ]
Nieuwe record/nieuw verzoek
Veld instellen [Notifications::_k2_checklistCollectiondId ; $checklistCollectionId ]
Veld instellen [Notifications::body ;
"Changes have been made to a checklist: " & $checklistCollectionName ]
Veld instellen [Notifications::flag_type ; 1 ]
Veld instellen [Notifications::flag_modified ; 1 ]
Records/verzoeken vastleggen [ Met dialoogvenster: Uit ]
```

Figuur 22: script, aanmaken nieuwe melding

Aan de hand van dit deel uit een script wordt er een notificatie aangemaakt in de database. Eerst wordt er genavigeerd naar de layout/tabel van de notifications. Vervolgens wordt er hier een nieuw record in aangemaakt en worden er enkele velden ingesteld. Zo wordt er een link gelegd met de verzameling van de checklist, het bericht zelf wordt ingesteld en enkele andere velden krijgen een waarde. Ten slotte wordt er nog gezorgd dat alles wat net aangemaakt is wordt vastgelegd.

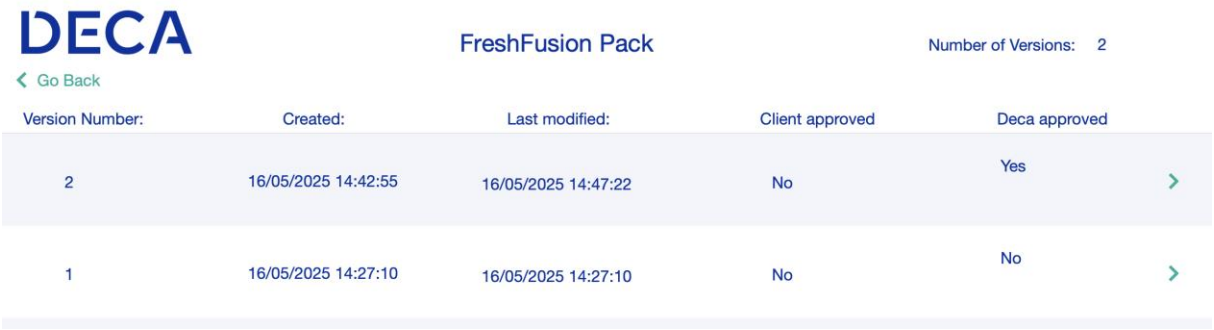
3.2.7. FileMaker layout: projectlijst



Figuur 23: FileMaker layout projectlijst

Dit is de eerste layout in FileMaker waarop een medewerker zal komen. Hierop bevinden zich alle projecten van een bepaalde klant, met extra informatie en het aantal versies bij.

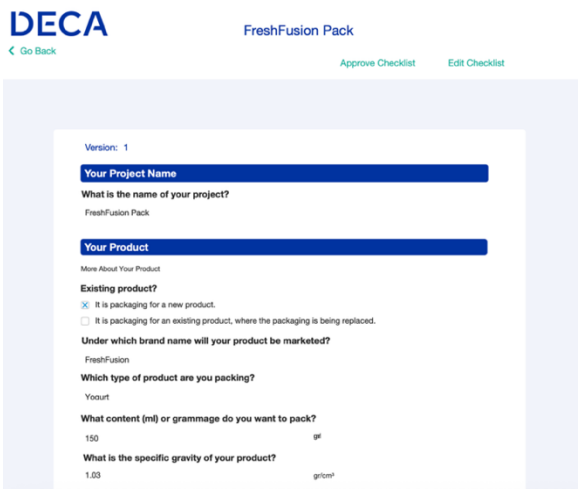
3.2.8. FileMaker layout: versieslijst



Figuur 24: FileMaker layout versieslijst

Als er op het pijltje wordt gedrukt komt men op deze layout terecht. Hier bevinden zich alle versies van het project. Hier kan men zien of deze versie goedgekeurd is door de klant en door Deca. Bij het klikken op het pijltje van de eerste versie, komt men op deze checklist terecht. Dit is enkel om te bekijken. Bij het klikken op het pijltje van de laatste versie komt men op de checklist uit en is er de optie om deze goed te keuren of aan te passen.

3.2.9. FileMaker layout: Checklist



Figuur 25: FileMaker layout ingevulde checklist

Bij het klikken op het pijltje bij de laatste versie op deze layout terecht. Hier is er de mogelijkheid om alles na te kijken, goed te keuren of de checklist aan te passen. Bij het goedkeuren gebeurt er een functie dat lijkt op het script voor de notificatie aan te maken. Bij het klikken op ‘Edit Checklist’ gebeurt er echter veel meer. Aangezien dit dan een nieuwe versie wordt moet er bij elke sectie een nieuw record aangemaakt worden met een kopie van deze data in het nieuwe record. Deze nieuwe checklist kan dan aangepast worden.

```
If [ Get ( LayoutNaam ) ≠ "ChecklistFormReadOnly" ]
    Aangepast dialoogvenster tonen [ Error ; "Wrong layout" ]
    Script afsluiten [ Tekstresultaat: ]
End If
If [ ChecklistVersion::flag_beingEditedByDeca = 1 ]
    Ga naar lay-out [ “ChecklistForm” (ChecklistVersion) ; Animatie: Geen ]
    Script afsluiten [ Tekstresultaat: ]
End If
```

Figuur 26: script, aanmaken checklist (deel 1)

Voordat de logica uitgevoerd kan worden, worden er eerst enkele controles uitgevoerd. Zo wordt onder andere nagegaan of de gebruiker zich op de juistse layout bevindt.

Ook wordt er gecheckt of de variabele, ‘flag_beingEditedByDeca gelijk is aan 1. Als dit het geval is, dan betekent het dat deze checklist al wordt aangepast en dus niet op de read-only layout moet zijn.

Na de check, wordt er eerst in een variabele de id van de checklist opgeslagen waarvan een kopie wordt gemaakt.

Vervolgens wordt de versie zelf gedupliceerd en wordt er gezorgd dat alle velden juist zijn ingesteld. Er moet getoond dat deze versie wordt aangepast, de versie nummer moet naar boven en ‘flag_approved’ moet op 0 aangezien de klant deze versie nog niet heeft kunnen goedkeuren.

```
Variabele instellen [ $checklistIdPrevious ; Waarde: ChecklistVersion::_k1_id ]
Record/verzoek dupliceren
Veld instellen [ ChecklistVersion::flag_beingEditedByDeca ; 1 ]
Veld instellen [ ChecklistVersion::VersionNumber ; ChecklistVersion::VersionNumber + 1 ]
Veld instellen [ ChecklistVersion::flag_approved ; 0 ]
Records/verzoeken vastleggen [ Met dialoogvenster: Uit ]
Variabele instellen [ $checklistId ; Waarde: ChecklistVersion::_k1_id ]
```

Figuur 27: script, aanmaken checklist (deel 2)

Nadat er een nieuwe versie is, wordt er naar de layout van de eerste sectie genavigeerd en een zoekopdracht uitgevoerd. Er moet in deze layout gezocht worden naar de versie van de sectie die we zagen voor het aanpassen hiervan, omdat deze gedupliceerd moet worden.

```
Ga naar lay-out [ “Section_ProductInfo” (Section_ProductInfo) ; Animatie: Geen ]
Zoekopdracht uitvoeren [ Herstellen ]
If [ Get ( GevondenTelling ) < 1 ]
    Aangepast dialoogvenster tonen [ Error ;
    "No record found for this section: section_productInfo" ]
Else
    Record/verzoek dupliceren
    Veld instellen [ Section_ProductInfo::_k2_checklistId ; $checklistId ]
    Records/verzoeken vastleggen [ Met dialoogvenster: Uit ]
End If
```

Figuur 28: script, aanmaken checklist (deel 3)

Vervolgens wordt er eerst gecheckt of dit gevonden is. Als dit niet zo is wordt er een venster getoond met een foutmelding. Als dit wel gevonden is, wordt de sectie gedupliceerd en gezorgd dat er een referentie is met de nieuwe versie van de checklist aan de hand van zijn id. Dit wordt nog vastgelegd en herhaald voor elke sectie.

4. Besluit

Tijdens mijn stage heb ik niet alleen de kans gekregen om een zeer uitgebreid en leerrijk project te realiseren, maar ook om mezelf op persoonlijk en professioneel vlak beter te leren kennen en ontwikkelen. In het begin van dit traject heb ik een grondige analyse uitgevoerd naar verschillende technologieën die het best gebruikt kunnen worden om dit tot een goed einde te kunnen brengen.

De gekozen oplossingen hebben geleid tot een succesvol einde, waarbij gebruiksvriendelijkheid, efficiëntie en schaalbaarheid centraal stonden. Ik heb doorheen het proces heel veel geleerd en mijn technische vaardigheden versterkt, maar ook in probleemoplossend denken en zelfstandig beslissingen nemen ben ik sterk ingegroeid.

Wanneer ik terugkijk op de doelstellingen die op voorhand zijn gesteld, kan ik de conclusie trekken dat ik deze behaald heb. Het eindproduct is er en voldoet aan de verwachtingen van de opdrachtgever.

Om kort te concluderen heeft deze stage en deze opdracht mij zowel op persoonlijk, professioneel en technisch vlak geprikkeld en heel veel bijgeleerd. Ik ben dankbaar dat ik deze kans heb gekregen.

LITERATUURLIJST

Angular. (15/08/2023). *What is Angular.* Opgeroepen op 16/05/2025, van Angular: https://v17.angular.io/guide/what-is-angular

Beeproger. (n.d.). *Angular: wat is het en hoe werkt het?* Opgeroepen op 16/05/2025, van beeproger: https://beeproger.com/blog/angular-wat-is-het-en-hoe-werkt-het/

Besseling, S. (18/02/2025). *.NET Core: wat zijn de voor- en nadelen?* Opgeroepen op 19/05/2025, van Growteq: https://growteq.nl/kennisbank/blogs-nl/maatwerk-development/net/net-core-wat-zijn-de-voor-en-nadelen/

Classens, S. (16/05/2025). *Tailwind CSS.* Opgeroepen op 16/05/2025, van Wux: https://wux.nl/wat-is/tailwind-css

Coleman, D. (30/08/2023). *Beoordeling van het React.js-framework: voordelen, nadelen en use cases.* Opgeroepen op 19/05/2025, van Serverspace: https://serverspace.io/nl/about/blog/review-of-the-react-js-framework-advantages-disadvantages-and-use-cases/

Crooks, M. (19/06/2024). *Angular vs React: Making the Front-End Choice in 2024.* Opgeroepen op 16/05/2025, van Medium: https://medium.com/@melissacrooks/angular-vs-react-making-the-front-end-choice-in-2024-3280a39762b2

cube. (n.d.). *Cube implementeert MongoDB voor flexibele en schaalbare NoSQL database-oplossingen.* Opgeroepen op 19/05/2025, van cube: https://cube.nl/technologie/mongo-db

Daan. (n.d.). *Wat is Angular?* Opgeroepen op 19/05/2025, van ndus3: https://ndus3.com/hub/wat-is-angular/

Dataflair. (n.d.). *Advantages and Disadvantages of ExpressJS.* Opgeroepen op 19/05/2025, van Dataflair: https://data-flair.training/blogs/expressjs-advantages-and-disadvantages/

GeeksforGeeks. (28/02/2025). *What is Express?* Opgeroepen op 19/05/2025, van geeksforgeeks: https://www.geeksforgeeks.org/what-is-express/

Goralski, C. (26/05/2022). *Voor- en nadelen van React.* Opgeroepen op 19/05/2025, van The codest: https://thecodest.co/nl/blog/voor-en-nadelen-van-react/

joogi. (n.d.). *Wat is: MySQL.* Opgeroepen op 19/05/2025, van joogi: https://joogi.nl/kenniscentrum/mysql/

kinsta. (07/06/2023). *Wat is TypeScript? Een uitgebreide gids.* Opgeroepen op 19/05/2025, van kinsta: https://kinsta.com/nl/kennisbank/wat-is-typescript/

kinsta. (16/06/2023). *Wat is MySQL? Een beginnersvriendelijke uitleg.* Opgeroepen op 19/05/2025, van kinsta: https://kinsta.com/nl/kennisbank/wat-is-mysql/

LJPc hosting. (16/05/2025). *Tailwind CSS: een veelzijdig framework voor modern webdesign.* Opgeroepen op 16/05/2025, van LJPc hosting: https://ljpc-hosting.nl/kennisbank/hosting/tailwind-css-een-veelzijdig-framework-voor-modern-webdesign

Mensink, M. (17/01/2022). *Wat is PHP? (complete uitleg).* Opgeroepen op 19/05/2025, van Hypernode: https://www.hypernode.com/nl/blog/wat-is-php/

Mertz, B. (n.d.). *Verschil tussen RDBMS en MongoDB.* Opgeroepen op 19/05/2025, van differbetween: https://nl.differbetween.com/article/difference_between_rdbms_and_mongodb

microsoft. (n.d.). *What is ASP.NET Core?* Opgeroepen op 19/05/2025, van microsoft: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core

Nastase, A. (10/08/2022). *Spring Boot: The good and the bad.* Opgeroepen op 19/05/2025, van medium: https://medium.com/@alexthedev/spring-boot-the-good-and-the-bad-20be1b409f2

scrums. (10/09/2024). *Spring Boot: Open-source Java framework.* Opgeroepen op 19/05/2025, van scrums: https://www.scrums.com/software-development-tools/spring-boot

Siddarth. (09/08/2025). *Angular: pros and cons.* Opgeroepen op 19/05/2025, van dev: https://dev.to/siddharthshyniben/angular-pros-and-cons-m9l

Taha, A. (10/06/2023). *Express.js: The good, the bad, and the Ugly.* Opgeroepen op 19/05/2025, van linkedin: https://www.linkedin.com/pulse/expressjs-good-bad-ugly-aziz-taha

The Support Group. (08/09/2020). *What is FileMaker (and what does it do)?* Opgeroepen op 19/05/2025, van supportgroup: https://blog.supportgroup.com/what-is-filemaker
ThePHPAdmin. (12/07/2017). *Voordelen en nadelen van PHP.* Opgeroepen op 16/05/2025, van PHPBabu: https://www.phpbabu.com/nl/voordelen-en-nadelen-van-php/
Tuple. (n.d.). *Tailwind.* Opgeroepen op 19/05/2025, van tuple: https://www.tuple.nl/nl/kennisbank/tailwind
webontwikkeling. (17/03/2018). *Wat is bootstrap?* Opgeroepen op 19/05/2025, van bronso: https://www.bronso.be/blog/programmeren/wat-is-bootstrap
Wikipedia. (08/07/2023). *FileMaker.* Opgeroepen op 19/05/2025, van wikipedia: https://nl.wikipedia.org/wiki/FileMaker
Wikipedia. (27/03/2023). *Angular.* Opgeroepen op 16/05/2025, van Wikipedia: https://nl.wikipedia.org/wiki/Angular
winacademy. (18/02/2021). *React - wat is het en waarom zou jij het moeten leren?* Opgeroepen op 19/05/2025, van wincadamy: https://www.wincacademy.nl/blog/react-wat-is-het-en-waarom-zou-jij-het-moeten-leren
Zedrox. (04/09/2024). *What is React?* Opgeroepen op 19/05/2025, van Zedrox: https://www.zedrox.nl/blog/alles-over-react-en-hoe-het-werkt/

GENERATIEVE AI

Tijdens het maken van dit realisatie document is er gebruik gemaakt van generatieve AI. Ik heb dit gebruikt als hulpmiddel, niet als definitieve bron van informatie. Outputs van AI kunnen fouten en beperkingen bevatten, daarom is de echtheid altijd nagekeken.

Prompts:

“Geef mij enkele opties voor een backend”

“Wat is Angular”

“Hoe kan ik best de voor- en nadelen tonen in een word document”

“Wat is PHP”

“Wat is React.js”

“Wat is MongoDB”

“Wat is Express.js”

“Wat is .NET core”

“Wat is Spring Boot”

“Voor- en nadelen Express.js met bronnen.”

“Voor- en nadelen ASP.NET core met”

“Geef mij de bronnen”