Professionele Bachelor Toegepaste Informatica

# eFenKa Progressive Web App

Matthias Van Den Bergh

Promotoren:

Joeri Jans — eMenKa NV
Nele Custers — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

# eFenKa Progressive Web App

Matthias Van Den Bergh

Promotoren:

Joeri Jans — eMenKa NV
Nele Custers — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

# Dankwoord

Voor het voltooien van mijn stageopdracht zou ik mijn bedrijfspromotor Joeri Jans willen bedanken. Dankzij zijn ervaring en expertise heb ik mijn stage tot een goed einde kunnen brengen. Ik zou hem in het bijzonder willen bedanken voor de duidelijke richtlijnen en voor de snelle reacties op al mijn vragen. Verder wil ik hem ook bedanken voor de vele nieuwe technologieën die ik doorheen mijn stage heb kunnen leren.

Daarnaast wil ik ook mijn hogeschoolpromotor Nele Custers bedanken voor alle feedback op zowel mijn eindwerk als mijn stageportfolio. Dankzij haar feedback en bijsturingen heb ik mijn eindwerk tot een goed einde kunnen brengen.

Verder zou ik eMenKa NV en al zijn werknemers willen bedanken voor de interessante en leerrijke stage en de steeds beschikbare hulp. In het bijzonder zou ik nog Marcel Aerts willen bedanken voor het mogelijk maken van deze stage en om ervoor te zorgen dat deze periode zo aangenaam mogelijk was voor mij.

Tot slot zou ik mijn ouders willen bedanken om mij de kans te geven deze opleiding te doorlopen en voor de steun doorheen deze jaren. En zou ik mijn vrienden willen bedanken, mede dankzij hen ben ik steeds gemotiveerd gebleven.

# Abstract

eMenKa, opgericht in 2007, is een onafhankelijk softwarebedrijf gespecialiseerd in .NET. De corebusiness van eMenKa is .NET-consultancy. eMenKa staat voor M en K, Mankracht en Kennis. Mankracht leveren aan bedrijven die een tijdelijk tekort hebben aan .NET-ontwikkelaars, en kennis leveren in de zin van expertiseopdrachten zoals securityaudit, SQL-prestatie-audit, architectuuropzet, etc.

Hoewel de focus op consultancy ligt, heeft eMenKa ook enkele interne projecten waaronder eFenKa. eFenKa is een applicatie ontwikkeld voor het beheer van wagenparken. Momenteel worden er ongeveer 2500 wagens van verschillende klanten in dit systeem beheerd.

De applicatie is ongeveer tien jaar oud en is aan vernieuwing toe. De huidige app is gebouwd met Visual WebGui en .NET framework 4.5 en wordt niet meer ondersteund. Ze wordt daarom opnieuw ontwikkeld met moderne technologieën om zo de applicatie weer onderhoudbaar en uitbreidbaar te maken**.** Het uiteindelijke doel is dat de toepassing als een Progressive Web App gereleaset wordt.

Een PWA of Progressive Web App is een webapplicatie die ook kan worden gedownload als mobiele toepassing. Doordat ze kan worden gedownload, werkt de toepassing ook offline en kan ze pushberichten sturen. Een PWA combineert de flexibiliteit van het web met de ervaring van een native applicatie.

eFenKa bestaat uit zeven modules. Tijdens de stageperiode wordt gefocust op de Voertuigenmodule. Deze module zorgt voor het beheer van de verschillende voertuigen binnen een bedrijf.

De technologieën die hiervoor gebruikt worden zijn ASP.NET Core 2.2 met Entity Framework Core voor de API-backend en Angular 7 voor de frontend. Authenticatie werkt via het SSO-principe dat toegepast wordt op alle applicaties van eMenKa door het gebruik van IdentityServer 4. Het volledige onderhouds- en ontwikkelproces van de app gebeurt volgens het DevOps-principe met behulp van Azure DevOps.

Gedurende de stage wordt ook de vergelijking gemaakt tussen de gebruikte frontendtechnologie Angular en Vue.js. Er wordt gekeken wat de voor- en nadelen van deze technologieën zijn en welke wanneer de betere keuze is. Daarnaast wordt onderzocht hoe ze beide geoptimaliseerd zijn voor het maken van een PWA en welke invloed het Reduxpatroon heeft op de prestaties van eenzelfde applicatie gemaakt met deze technologieën.

Uit onderzoek blijkt dat beide frameworks uitstekend geoptimaliseerd zijn voor het maken van PWA. Angular bleek voor eFenKa de beste optie van de twee vanwege de omvang van de applicatie. Vue.js is het overwegen waard voor kleinere applicaties. Verder toont het onderzoek aan dat Redux de prestaties van de applicatie kan bevorderen, maar de belangrijkste reden om deze technologie te gebruiken is de leesbaarheid verhogen en de code “loosely coupled” houden zodat deze makkelijk testen is.

# Inhoudsopgave

# Lijst van gebruikte figuren

# Lijst van gebruikte tabellen

# Lijst van gebruikte afkortingen

| | |
|---|---|
| CLI | Command-line interface |
| DTO | Data Transfer Object |
| HR | Human Resources |
| MVC | Model-View-Controller |
| Npm | Node.js package manager |
| OIDC | OpenID Connect |
| ORM | Object-relational mapping |
| OS | Operating System |
| PBI | Product Backlog Item |
| PWA | Progressive Web App |
| R&D | Research and Development |
| SaaS | Software as a Service |

# Inleiding

eMenKa NV is een .NET-consultancybedrijf met enkele interne projecten, waaronder eFenKa. eFenKa is een wagenparkbeheersysteem dat ongeveer tien jaar geleden gebouwd is. De technologieën die daarvoor gebruikt zijn, .NET Framework en Visual WebGui, zijn verouderd en de laatstgenoemde wordt zelfs niet meer ondersteund.

Daarom bestaat mijn stageopdracht erin deze webapplicatie weer op te bouwen met nieuwe technologieën. Deze technologieën zijn Angular 7 voor de frontend en .NET Core 2.2 voor de backend-API. Verder wordt het vernieuwde eFenKa een Progressive Web App. Dit wil zeggen dat de website geïnstalleerd kan worden als applicatie om vervolgens te werken als native applicatie op mobiele toestellen.

Omdat eFenKa een grote applicatie is die bestaat uit zeven verschillende modules, wordt er gefocust op één module, namelijk die voor de voertuigen. Binnen deze module is het mogelijk een overzicht van alle beschikbare voertuigen te bekijken en deze voertuigen te beheren.

Zoals gezegd heeft eMenKa besloten Angular te gebruiken voor de frontend. Maar is dit framework de beste optie? In het onderzoek wordt Angular vergeleken met een ander populair framework, Vue.js. Aan de hand van een literatuurstudie worden de twee frameworks vergeleken op verschillende vlakken zoals populariteit, algemene ondersteuning, Progressive Web App-ondersteuning en verschillende andere domeinen. Naast de literatuurstudie worden ook metingen uitgevoerd met de Chrome DevTools en een Chrome-extensie genaamd Lighthouse. Met deze tools worden de laadtijden van de websites, gemaakt met bovenstaande frameworks, gemeten. Met Lighthouse wordt ook onderzocht hoe goed de frameworks uitgerust zijn voor het creëren van Progressive Web Apps.

Daarnaast wordt ook gekeken naar de invloed van het Reduxpatroon binnen deze twee frameworks aan de hand van hun framework-specifieke Redux-implementaties, namelijk NgRx voor Angular en Vuex voor Vue.js.

Aan de hand van deze gegevens wordt er geconcludeerd welk framework in welke situatie de beste keuze is.

# I. Stageverslag

# 1 Bedrijfsvoorstelling

## 1.1 Situering

eMenKa, opgericht in 2007, is een onafhankelijk softwarebedrijf gespecialiseerd in .NET. De hoofdzetel van eMenKa bevindt zich in Antwerpen. In Gent is er een bijkantoor en in Kortrijk en Hasselt heeft eMenKa een flexkantoor ter beschikking. In dit laatste vindt de stage plaats.

eMenKa heeft een platte organisatiestructuur. eMenKa's overhead bestaat uit de drie oprichters en twee managers. Steven Claus (oprichter) staat in voor de financiën en administratie, Marcel Aerts (oprichter) voor interne ICT en HR. Joeri Jans (9 jaar bij eMenKa) is de competencemanager van eMenKa. Hij is het aanspreekpunt voor alle technische aangelegenheden en hij is ook de scrummaster en teamlead van de interne .NET-ontwikkelingen (hoofdzakelijk de SaaS-oplossing eFenKa, maar ook de applicaties voor de administratieve vereenvoudiging binnenin eMenKa). Tjorven Denorme (oprichter), sales- en accountmanager, en Luk Vanderstraeten (11 jaar bij eMenKa), accountmanager, staan in voor de opvolging van de klanten en de medewerkers.

eMenKa heeft vanaf dag één gekozen voor een structurele groei door mensen te binden met dezelfde waarden die het management ambieert.

## 1.2 Focus eMenKa

### 1.2.1 .NET

De corebusiness van eMenKa is .NET-consultancy. eMenKa staat voor M en K, Mankracht en Kennis. Mankracht leveren aan bedrijven die tijdelijke tekorten hebben aan .NET-onderhoud of meehelpen aan nieuwe .NET-oplossingen. Kennis leveren in de optiek van korte expertiseopdrachten zoals securityaudit, SQL-prestatie-audit, architectuuropzet, etc.

### 1.2.2 Regio

Bij eMenKa is werk-levensbalans een heel belangrijk element. Ze volgen hun contractoren nauw op en spelen in op hun veranderende interesses en privé- of gezinsleven. Daarom werken ze steeds met regionale consultants en volgen ze hun mensen van nabij op. eMenKa is van mening dat de energie in het project moet zitten en niet al deels verloren mag gaan in lastige woon-werkverplaatsingen.

## 1.3 Partners

eMenKa is een Microsoft partner en sinds 2018 zijn ze lid van AZUG. Verder wordt er werk gemaakt van partnerships met andere *user groups* zoals ViSUG, MADN etc. Sinds 2019 stellen zij ook een medewerker ter beschikking voor hét .NET-event van het jaar, Techorama. eMenKa is ook erkend door Belspo als Young Innovative Company voor zijn R&D-activiteiten rond eFenKa en specifiek R&D-werk bij de huidige klanten.

# 2 Voorstelling stageopdracht

## 2.1 eFenKa

Naast hun corebusiness, .NET-consultancy, heeft eMenKa zelf een SaaS-oplossing ontwikkeld en gecommercialiseerd voor het beheer van wagenparken, nl. eFenKa. Onder andere Baloise Insurance gebruikt eFenKa voor het managen van hun wagenpark.

De applicatie bestaat uit de volgende zeven modules:

- Dashboard: Deze module is de landingspagina van de website. Hij geeft een overzicht van de belangrijkste informatie uit de andere modules. Dat zijn vooral deadlines of limieten die dreigen overschreden te worden. Enkele voorbeelden hiervan zijn wagens die hun toegelaten kilometers dreigen te overschrijden of een te hoog verbruik hebben.
- Dossiers: Deze module geeft een overzicht van de voertuigen die in gebruik zijn met hun bestuurder, nummerplaat, etc.
- Bestuurders: Alle personen die in het bezit kunnen zijn van een wagen en hun persoonlijke informatie.
- Leveranciers: Alle bedrijven die instaan voor onder andere onderhoud en verzekeringen.
- Tankkaarten: Een overzicht van alle tankkaarten met hun eigenaar.
- Voertuigen: Een tabel met alle beschikbare merken en hun modellen, motortypes, etc.
- Rapporten: Deze module zorgt voor het genereren van rapporten van de verschillende modules over verschillende periodes.

## 2.2 Huidige situatie

eFenKa is ongeveer tien jaar geleden ontwikkeld. De toepassing is gemaakt met Visual WebGui, SQL Server, Entity Framework en .NET Framework 4.5. Omdat Visual WebGui niet meer wordt ondersteund en .NET Framework geleidelijk aan vervangen wordt door .NET Core, moet de applicatie opnieuw opgezet worden met moderne technologieën.

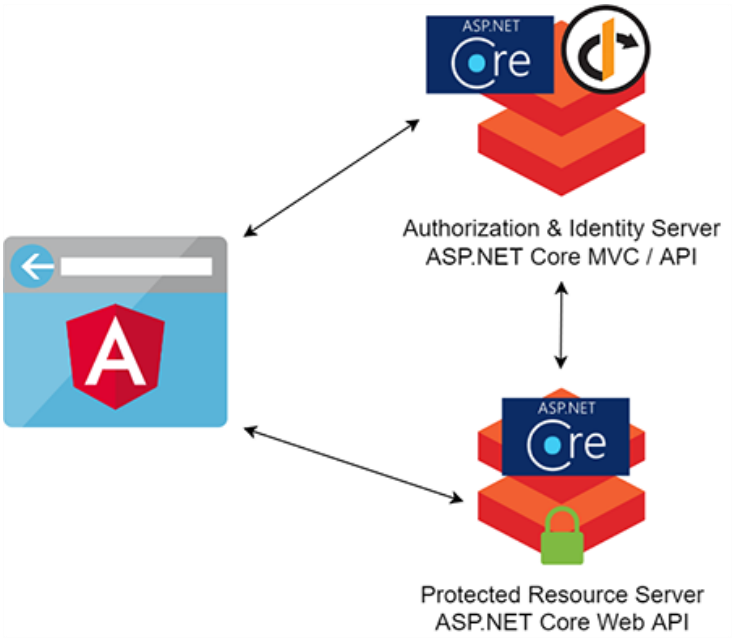
## 2.3 Nieuwe situatie

Voor de nieuwe versie van eFenKa is gekozen voor Angular 7 als frontendtechnologie. De backend-API wordt ontwikkeld met ASP.NET Core 2.1 en Entity Framework als ORM. De huidige SQL-database blijft behouden. Bovendien wordt de nieuwe applicatie een Progressive Web App waardoor die ook geoptimaliseerd is voor mobiele toestellen. Omdat het een Progressive Web App is, werkt de applicatie ook offline en bestaat de mogelijkheid tot installatie als applicatie.

Net als in de huidige situatie is de nieuwe toepassing voorzien van authenticatie. De nieuwe situatie maakt hiervoor gebruik van het Single sign-on-principe. Het komt erop neer dat de gebruiker slechts eenmaal moet inloggen om toegang te krijgen tot alle applicaties van eMenKa waarvoor hij geautoriseerd is.

## 2.4 Architectuur eFenKa

Onderstaande afbeelding geeft schematisch de architectuur van eFenKa weer.




*Figuur 1 Algemene architectuur eFenKa [1]*

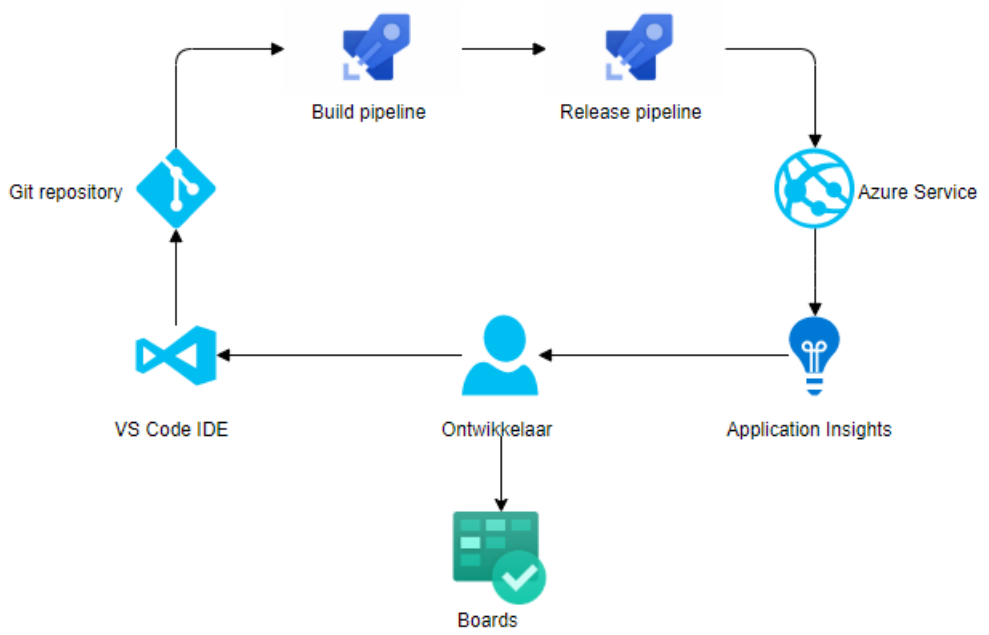
Als de gebruiker naar de eFenKa-website surft komt hij terecht op de eMenKa-inlogpagina afkomstig van de Identity Server. Als het inloggen geslaagd is, wordt de gebruiker doorverwezen naar eFenKa. Hier krijgt hij de Angular-applicatie te zien. De data binnen de applicatie komt van een ASP.NET Core-API.

# 3 Uitwerking stageopdracht

## 3.1 Azure DevOps

Het managen van het eFenKa-project gebeurt in Azure DevOps. Azure DevOps is een Microsofttool die verschillende hulpmiddelen bevat voor het bouwen van software. Deze hulpmiddelen bestaan uit de Boards, Pipelines, Repos, Test Plans en Artifacts. De Boards bevatten alle Agile hulpmiddelen zoals de *backlog*, het Kanbanbord en de verschillende sprints. De Pipelines bevatten alle Build en Release *pipelines*. Deze Build *pipelines* zorgen voor het uitvoeren van de nodige processen om een *build* te maken van het project dat vervolgens gereleaset wordt via de Release *pipeline*. De Repos bevatten alle *repositories*. Hier bevindt zich alle code en gebeurt het versiebeheer, in het geval van mijn stage, met behulp van Git. De Test Plans bevatten verschillende tools voor softwaretesters en onder Artifacts is het mogelijk packages te maken, hosten en delen binnen het project. Deze laatste twee hulpmiddelen worden niet gebruikt tijdens het stageproces. De implementaties van de overige tools worden in dit hoofdstuk toegelicht.

### 3.1.1 Algemene flow

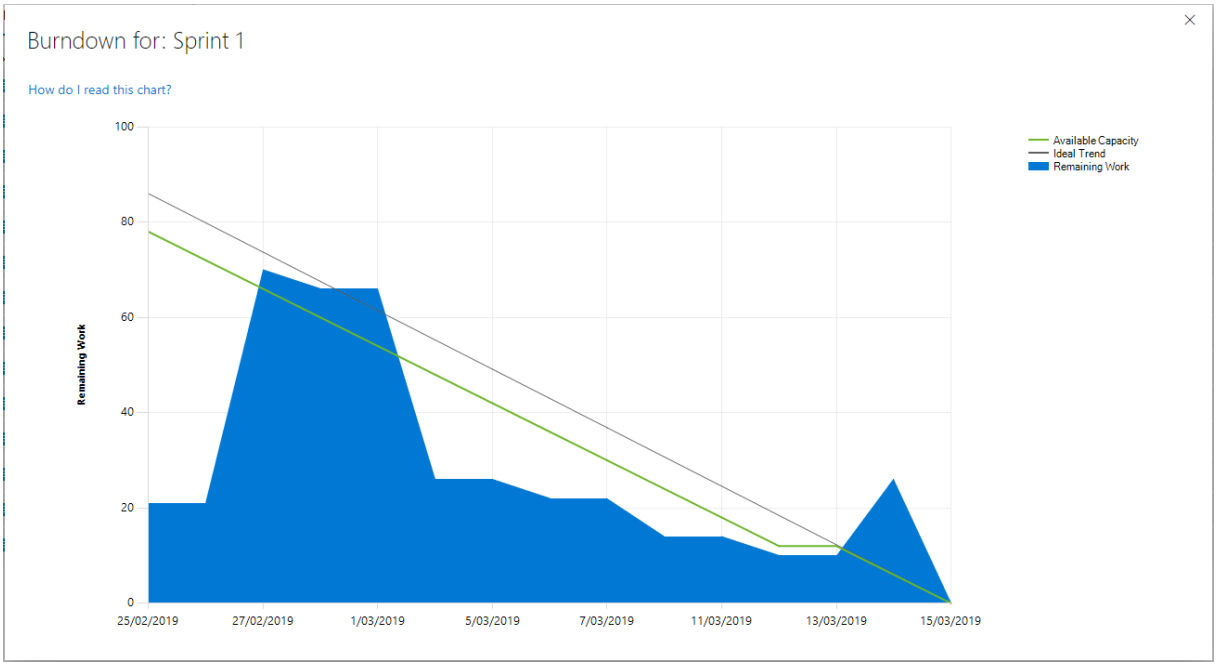



*Figuur 2 Azure DevOps flow*

Bovenstaande afbeelding toont schematisch de flow van Azure DevOps binnen het eFenKa-project. Voor lokaal versiebeheer is gebruikgemaakt van de ingebouwde tools van Visual Studio Code. Van hieruit wordt code naar de Git *repository* binnen Azure DevOps gepusht. Van deze code wordt vervolgens een *build* gemaakt in de Build *pipeline* en van het backend-project worden de testen gerund. Als het maken van de *build* geslaagd is, wordt het verkregen Build Artifact naar de Release *pipeline* gestuurd. Hier wordt van beide projecten (front- en backend) een *production build* gemaakt. Deze wordt vervolgens geïmplementeerd op een Azure Service. Via de Application Insights kan de ontwikkelaar eventuele problemen herkennen. De analyse van het project en opvolgen van de sprints en *backlog* gebeuren via de Azure Boards.

### 3.1.2 Agile

Het project verloopt volgens een agile-methodologie met behulp van de Azure Boards. De verschillende *user stories* worden doorheen het verloop van de stage toegevoegd aan het Kanbanbord. De stage wordt opgedeeld in sprints van telkens drie weken. Elke sprint bevat verschillende *user stories*, in Azure DevOps worden ze Product Backlog Items genoemd, die aan het einde van de sprint af moeten zijn. Elk PBI bevat één of meer “Tasks” of taken die de programmeur moet voltooien om een PBI als afgewerkt te beschouwen. Elk PBI bevat een verwacht aantal werkuren. Op basis van die uren kan de vooruitgang per sprint opgevolgd worden aan de hand van een *burndown chart*. Deze toont het verwachte verloop, de maximaal haalbare capaciteit en het effectieve verloop van de sprint. Onderstaande figuur toont de *burndown chart* van sprint 1.




*Figuur 3 Burndown chart sprint 1*

### 3.1.3 Versiebeheer

Ook het versiebeheer gebeurt in Azure DevOps. Het heeft hiervoor een eigen implementatie van Git. Omdat er maar één ontwikkelaar aan het project werkt, is er beslist geen gebruik te maken van verschillende branches, enkel een Master- en Developmentbranch. Alle *commits* komen op de Developmentbranch terecht.

### 3.1.4 Pipelines

Om *build*- en deployproblemen snel op te sporen wordt er gebruikgemaakt van twee Build en Release *pipelines*. Zowel het Angular- als ASP.NET Core-project zitten elk in een eigen Build en Release *pipeline*. Om dit mogelijk te maken voor het Angular-project is het toegevoegd aan een .NET Solution die dient als een container voor dit project. Deze techniek wordt onder 3.4.2 .NET Solution als container voor het Angular-project verder toegelicht.

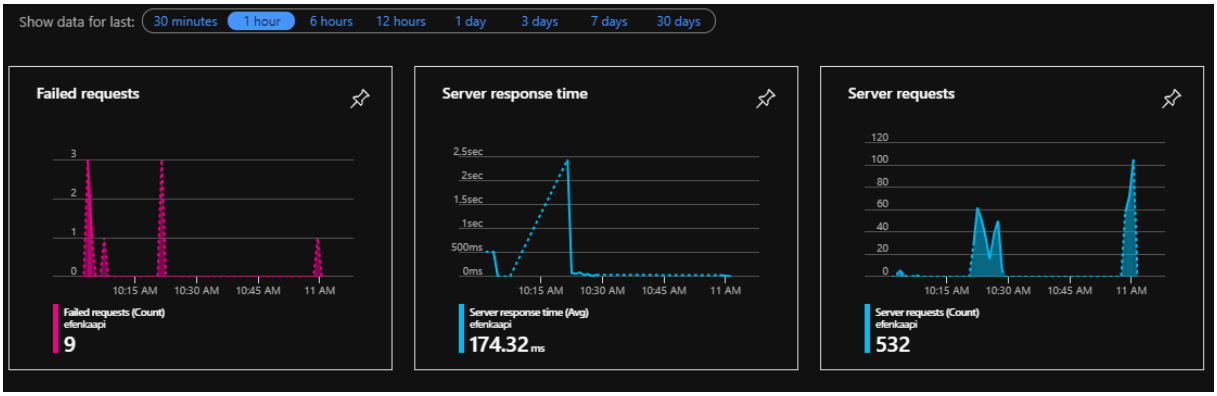
Het maken van een *build* gebeurt de eerste sprints ongeveer één keer per sprint, wanneer een groter onderdeel van de applicatie voltooid is en de bijhorende unittesten geschreven zijn. Naarmate de deadline nadert, wordt er frequenter een *build* en release gemaakt.

De Release *pipeline* zorgt voor het creëren van een release naar een Azure Service. Zo kunnen problemen bij het maken van een *production build* van de Angular-applicatie ook vroegtijdig opgelost worden.

De laatste weken van de stage wordt de volle kracht van Azure pipelines gebruikt door het aanzetten van Continuous Integration en Continuous Deployment. Als er een push gebeurt naar de Development *branch* zal hiervan automatisch een *build* gemaakt worden. Als deze afgelopen is, wordt een melding naar Slack gestuurd met de status van de *build* om eventuele problemen meteen op te kunnen lossen. Als de *build* slaagt wordt ook een release gemaakt van het Build Artifact naar de Azure Service. Als de Release start en eindigt wordt eveneens een bericht naar Slack gestuurd met de status van de release.

### 3.1.5 Azure Portal

De API is uitgerust met een NuGet *package* dat informatie over de API-*requests* naar Azure Portal stuurt. Hier kan onder de Application Insights handige informatie afgelezen worden van de grafieken zoals het aantal mislukte *requests*, de gemiddelde reactietijd van de server en het totaal aantal *requests*. Onderstaande afbeelding toont deze informatie over een periode van een uur.




*Figuur 4 API-requests over een periode van een uur*

## 3.2 Code Quality Control

Om een zekere garantie te hebben van kwaliteitsvolle code wordt er gebruikgemaakt van volgende technieken.

### 3.2.1 Unittesten ASP.NET Core

In de API-backend worden er unittesten geschreven voor de verschillende services. Deze bevatten de logica van de backend en zijn dus een belangrijk punt om te testen. De services vormen de verbinding tussen de *controllers* en *repositories*. Onderstaande afbeelding toont de *code coverage* binnen de services.

Voor het schrijven van de unittesten wordt gebruikgemaakt van xUnit. Voor het maken van *mocks* is gebruikgemaakt van NSubstitute.

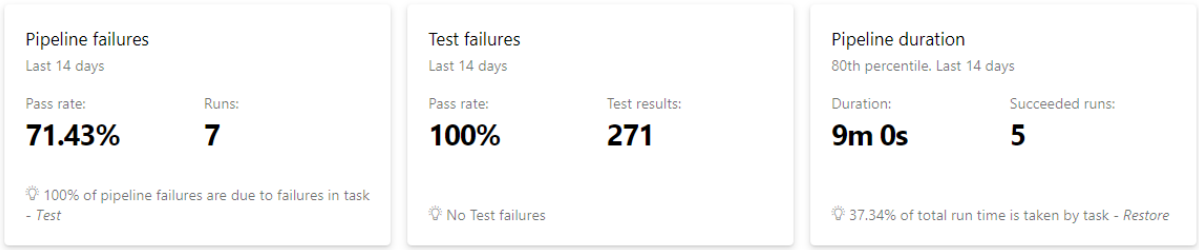
| | | | | |
|---|---|---|---|---|
| efenka.services.dll | 83 | 11,81% | 620 | 88,19% |
| Efenka.Services.Vehicles.Vehi... | 8 | 7,55% | 98 | 92,45% |
| Efenka.Services.Vehicles.Seri... | 10 | 15,63% | 54 | 84,38% |
| Efenka.Services.Vehicles.Mo... | 10 | 12,82% | 68 | 87,18% |
| Efenka.Services.Vehicles.Inte... | 10 | 12,82% | 68 | 87,18% |
| Efenka.Services.Vehicles.Exte... | 10 | 12,82% | 68 | 87,18% |
| Efenka.Services.Vehicles.Eng... | 10 | 12,82% | 68 | 87,18% |
| Efenka.Services.Vehicles.Do... | 9 | 13,24% | 59 | 86,76% |
| Efenka.Services.Vehicles.Bra... | 8 | 11,27% | 63 | 88,73% |
| Efenka.Services.Shared.DefC... | 8 | 12,90% | 54 | 87,10% |
| Efenka.Services.Shared.Corp... | 0 | 0,00% | 17 | 100,00% |
| Efenka.Services | 0 | 0,00% | 3 | 100,00% |

*Figuur 5 Test coverage API*

Het Services-project is uitgekomen op een *coverage*-percentage van ongeveer 88%. In de praktijk komt dit erop neer dat alle methodes getest zijn, behalve de blokken die een *exception* opgooien.

Deze unittesten worden overigens automatisch uitgevoerd bij het maken van een *build* in de Build Pipeline. De *pipeline* houdt informatie zoals het slagingspercentage van de unittesten bij.

Onderstaande afbeelding toont de slagingspercentages van de *builds* en *testruns*, en de gemiddelde duur van het *build*-proces van de API.




*Figuur 6 Slagingspercentages build en testen en gemiddelde duur van een build*

### 3.2.2 Unittesten Angular

Het frontendproject bevat unittesten gemaakt met het Jasmine testframework die uitgevoerd worden met Karma. Deze testen controleren de *API-calls* binnen de services. Ze checken de hoeveelheid *calls* en de verwachte output. De HttpClient wordt hiervoor vervangen door een *mock*-object gemaakt aan de hand van een Jasmine SpyObject.

Er zijn tests geschreven voor alle services van de Voertuigenmodule. Deze services zorgen voor de API-calls en het filteren van de data. Onderstaande afbeelding bevestigd de 100% *test coverage* voor de services onder de Vehicles-module. Verder is ook de service getest die instaat voor afhandelen van de verschillende errorstatuscodes.

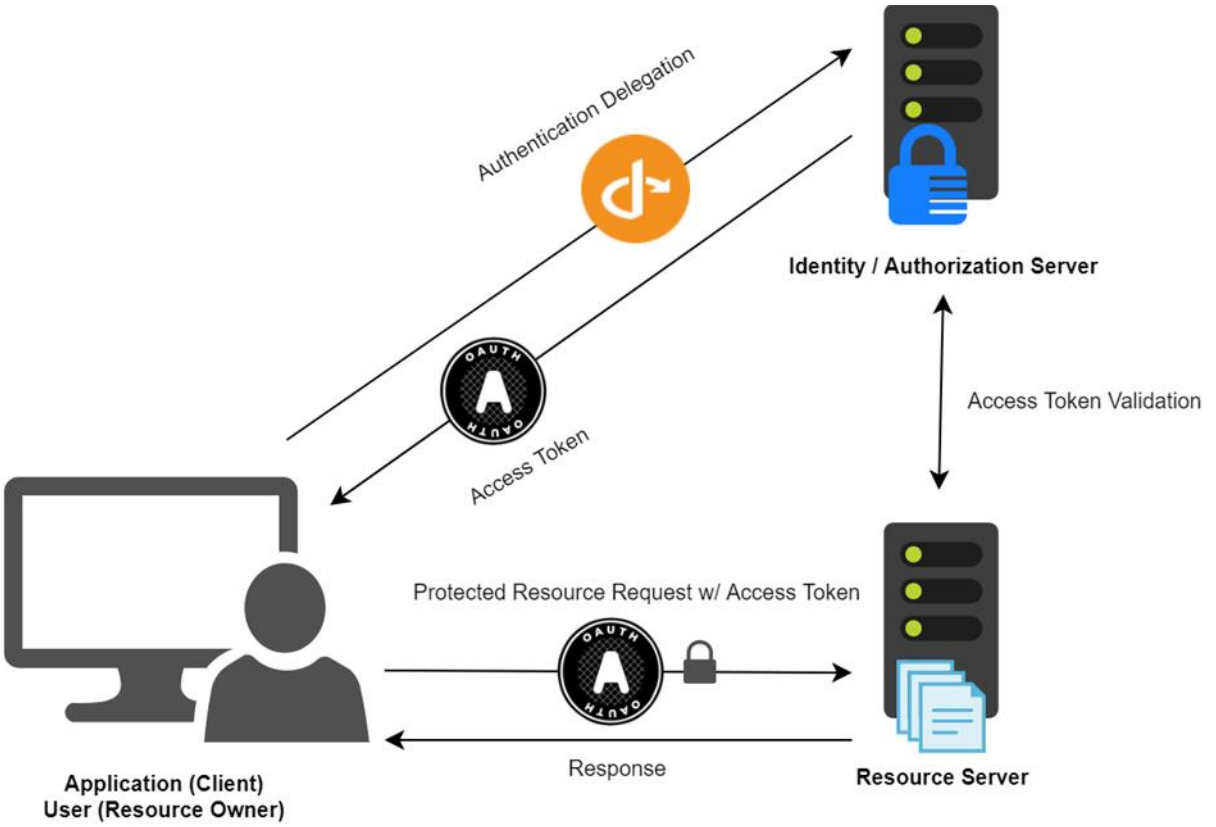
| File | | Statements | | Branches | | Functions | |
|---|---|---|---|---|---|---|---|
| src | | 100% | 3/3 | 100% | 0/0 | 100% | 0/0 |
| src/app/shared/models | | 100% | 1/1 | 100% | 0/0 | 100% | 0/0 |
| src/app/shared/services | | 85.71% | 18/21 | 100% | 0/0 | 100% | 4/4 |
| src/app/shared/testing | | 50% | 2/4 | 100% | 0/0 | 50% | 2/4 |
| src/app/vehicles/shared/services | | 100% | 180/180 | 90.91% | 20/22 | 100% | 70/70 |
| src/environments | | 0% | 0/0 | 0% | 0/0 | 0% | 0/0 |

*Figuur 7 Test coverage Angular*

### 3.2.3 Code reviews

Ook voert de bedrijfspromotor op regelmatige tijdstippen Code Reviews uit. De feedback die hieruit komt, wordt toegepast in het verdere ontwikkelproces.

## 3.3 Authenticatie en autorisatie via Identity Server




*Figuur 8 Schematische voorstelling Identity Server [1]*

Voor het authentiseren van gebruikers werkt eMenKa met Identity Server. Dit heeft als voordeel dat alle authenticatie en autorisatie op een centrale server gebeurt. Hierdoor kunnen gebruikers hergebruikt worden over de verschillende applicaties en kan centraal beslist worden welke gebruiker toegang krijgt tot welke applicatie. Bovenstaande afbeelding toont schematisch de werking van dit principe.

Als de gebruiker naar de applicatie surft wordt gecontroleerd of hij nog een geldig Access Token heeft. Als de gebruiker een tijd niet meer actief was of nog niet ingelogd is, wordt hij via een Authentication Delegation verwezen naar de Identity Server van eMenKa en krijgt hij een inlogpagina te zien. Dit gebeurt aan de hand van een *route guard* op de *base url* van de Angular-applicatie die checkt of er nog een geldig token aanwezig is. Heeft de browser wel nog een geldig token, dan wordt de gebruiker doorverwezen naar de startpagina van de applicatie. Als de client vervolgens data opvraagt uit de API, bevat de request steeds het Access Token. De API checkt met de Identity Server of de gebruiker geautoriseerd is om de gevraagde data te bekijken of te manipuleren en geeft vervolgens toegang tot de data als de client de juiste rechten heeft.

Deze manier van authentiseren en autoriseren is gebaseerd op OpenID Connect en OAuth. OpenID Connect (OIDC) is een eenvoudige laag van identiteits- en authenticatieprotocollen die is gebouwd bovenop het OAuth-protocol waarmee clients de identiteit van gebruikers kunnen verifiëren. OAuth is een open standaard voor autorisatie die veilige, gedelegeerde toegang biedt, wat betekent dat een client namens een gebruiker acties kan ondernemen of toegang tot data via een API kan krijgen, zonder dat de gebruiker ooit zijn authenticatiegegevens met de applicatie zelf deelt. Dit laatste is te danken aan de geïmplementeerde vorm van delegatie. [1]

Een vorm van autorisatie is nog niet geïmplementeerd tijdens de stage. De autorisatie van gebruikers binnen eFenKa gebeurt zoals in de andere applicaties van eMenKa aan de hand van *claims*. Door bijvoorbeeld het geven van een "admin"-claim aan een bepaalde gebruiker krijgt deze toegang tot administratieonderdelen van de applicatie. Zo kan er makkelijk toegang verleend worden tot bepaalde functies volgens de verschillende functies binnen een bedrijf zoals onderhoud, HR, etc.

## 3.4 Frontend-development

Onder frontend-development wordt het Angular-project verder toegelicht.

### 3.4.1 *Npm packages*

Eén van de grote sterktes van Angular en JavaScript in het algemeen, is het gebruik van *npm packages.* De *Node.js package manager* is een packagemanager speciaal voor de JavaScript-programmeertaal. De *package manager* kan zowel afhankelijkheden van een project als van het systeem beheren. De npm zorgt voor een package.json-bestand waarin alle afhankelijkheden opgeslagen staan. Via één commando kan elke ontwikkelaar alle benodigde afhankelijkheden installeren.

### 3.4.2 .NET Solution als container voor het Angular-project

Zoals eerder vermeld is gekozen om het Angular-project aan een .NET Solution toe te voegen. Het grote voordeel hiervan is dat het project kan beschouwd worden als een .NET-project wanneer het geïmplementeerd wordt op een server. Omdat de solution dient als container voor het project, wordt het hosten op Azure vergemakkelijkt. Ook in de *build* en release *pipeline* kan het project hierdoor beschouwd worden als een .NET-applicatie en kan er gebruikgemaakt worden van de bestaande templates die de verschillende dotnet-commando's uitvoeren voor het *build*-proces. Dit maakt het dan weer makkelijk om het Continuous Integration- en Continuous Deployment-principe toe te passen op zowel de backend als de frontend.

### 3.4.3 Progressive Web App-ondersteuning in Angular

Om de gebruikerservaring te verbeteren van de applicatie wordt er PWA-ondersteuning toegevoegd aan de applicatie. Deze ondersteuning voorziet de website van een Manifestbestand en een Service Worker. Dankzij het Manifestbestand kan de website geïnstalleerd worden en werken als een gewone mobiele applicatie. De Service Worker zorgt ervoor dat de website lokaal in de browser gecachet wordt en werkt zonder internetverbinding. De website zal hierdoor als sneller ervaren worden aangezien de pagina's niet van een externe server moeten geladen worden. De meer diepgaande en technische uitleg over Progressive Web Apps staat in het onderzoekgedeelte van de paper.

Het toevoegen van deze ondersteuning in Angular kan eenvoudig via het commando: "ng add @angular/pwa". Dit commando zorgt voor de configuratie van het Manifestbestand en de Service Worker. In de praktijk worden er twee afhankelijkheden toegevoegd aan het project: "@angular/pwa" en "@angular/service-worker", respectievelijk voor de PWA-configuratie en de Service Worker-configuratie. Er worden dus twee belangrijke bestanden toegevoegd, met name "manifest.json", het Manifestbestand en "ngsw-config.json", de configuratie van de Service Worker. Verder worden een referentie naar het Manifestbestand en een "noscript"-tag toegevoegd aan het "index.html"-bestand. De "noscript"-tag waarschuwt de gebruiker dat JavaScript verplicht is om de applicatie te doen werken, moest hij dit niet geïnstalleerd hebben. Als laatste worden er ook

verschillende iconen toegevoegd die worden gebruikt als de website als applicatie geïnstalleerd wordt.

## 3.5 Backend-development

Onder backend-development wordt het ontwikkelproces van de ASP.NET Core-API verder toegelicht.

### 3.5.1 NuGet *packages*

NuGet is net zoals de npm een packagemanager, maar dan voor .NET. De NuGet-clienttools bieden de mogelijkheid om *packages* te creëren en te gebruiken. Het backendproject gebruikt verschillende NuGet *packages*. Onder ander Identity Server voor de authenticatie en autorisatie, en AutoMapper voor het mappen van de entiteiten naar DTO's zijn belangrijke NuGet *packages* binnen het eFenKaproject. Daarnaast bevat het project ook een *package* van eMenKa zelf dat verschillende herbruikbare klassen bevat die zij in al hun projecten gebruiken. In eFenKa zorgt het voor het linken van de *request*-statistieken aan de Azure Portal Insights.

### 3.5.2 ORM

Om met een domein model te kunnen werken en data uit de database te halen met zo weinig mogelijk code is het gebruik van een ORM een logische keuze. Voor de eFenKa API wordt daarom gebruikgemaakt van Entity Framework. eFenKa bestond al en had daarom een bestaande database. Ook hiervoor is Entity Framework een handige tool door een techniek genaamd "Code first existing database" genereert Entity Framework Entiteit-klassen gebaseerd op de database. Aan de hand van AutoMapper konden deze entiteiten makkelijk omgevormd worden naar DTO's om zo objecten te creëren met de benodigde eigenschappen.

### 3.5.3 OData

Om het ophalen en tonen van de data te optimaliseren, wordt er bij het ophalen van de voertuigen gebruikgemaakt van OData. OData is een NuGet *package* van Microsoft dat ontwikkelaars de mogelijkheid biedt ingebouwde queryparameters te gebruiken. De GetVehiclesPaged-methode heeft een EnableQuery-tag zoals te zien op onderstaande afbeelding die ervoor zorgt dat je de data per x-aantal records kan afhalen. Daarnaast bevat OData ook implementaties voor bijvoorbeeld het filteren en ordenen van data, allemaal via de url. Zo hoeft er geen extra functie voorzien te worden voor het filteren van de voertuigen op merk.

```
[HttpGet]
[EnableQuery]
0 references | Matthias Van Den Bergh, 15 days ago | 1 author, 1 change | 0 requests | 0 exceptions
public ActionResult<List<Vehicle>> GetVehiclesPaged()
{
    return Ok(_readVehicleService.GetAll());
}
```

*Figuur 9 OData implementatie op de GetVehiclesPaged-methode*

Het gebruik van OData bracht enkele problemen met zich mee voor de frontend. De tabel die gebruikt wordt in de frontend is gemaakt om alle data te ontvangen en deze dan vervolgens zelf in te delen in pagina's met een zelfgekozen hoeveelheid records. Het filteren van de data gebeurde oorspronkelijk ook in de frontend, maar dit is met de implementatie van OData niet meer mogelijk aangezien niet alle data tegelijk toegankelijk is in de frontend. Daarom wordt voor het filteren

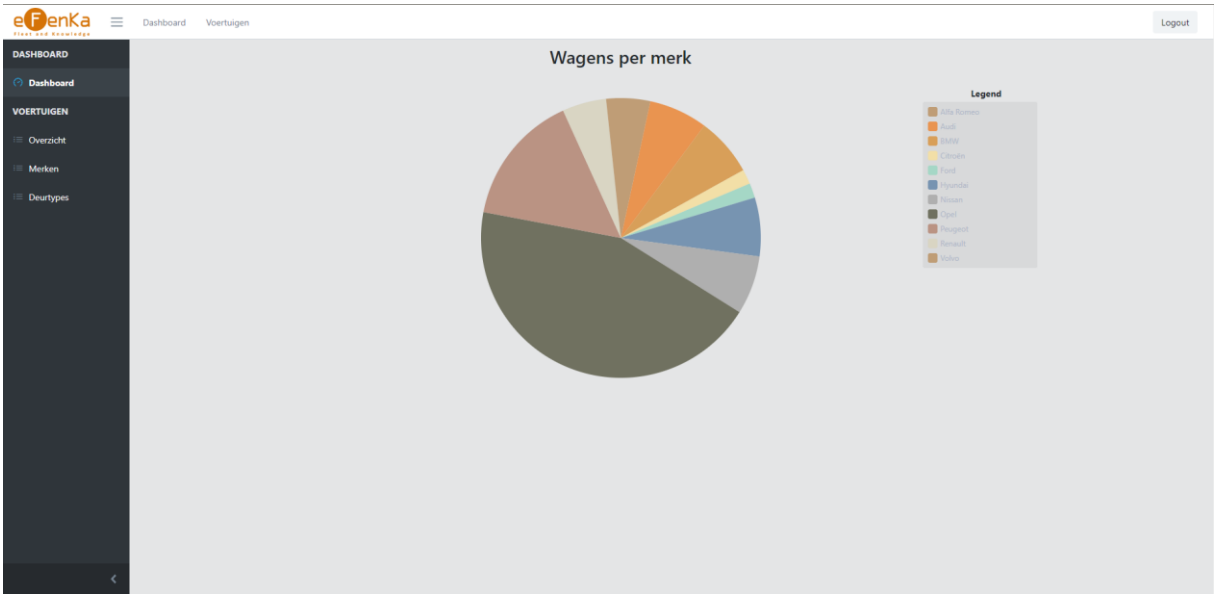
gebruikgemaakt van de ingebouwde OData-filter. Zo wordt de data al in de backend gefilterd. Het probleem met de pagina's was in de frontend te verhelpen door een ingebouwde eigenschap van de tabel te activeren ("Server pagination") en bij het initialiseren van de tabel de totale hoeveelheid records mee te geven. Dan was er enkel nog het problemen met de "Exporteer naar Excel"-functie waarbij de gebruiker verschillende records kan selecteren om te exporteren. Dit is opgelost door alle data af te halen op het moment dat de checkbox van een rij wordt aangevinkt.

## 3.6 Eindresultaat

Onder eindresultaat wordt in detail naar de gemaakte applicatie gekeken. Het eindresultaat geeft een overzicht van hoe eFenKa er aan het einde van de stageperiode uitziet en focust op enkele bijzondere features.

### 3.6.1 Het dashboard

De Voertuigenmodule is tijdens de stage volledig afgewerkt. Als de gebruiker naar de website van eFenKa surft, komt hij eerst op de dashboardpagina terecht. Hier staat voorlopig enkel een cirkeldiagram met een overzicht van het aantal wagens per merk dat een bedrijf heeft. Als de andere modules worden gebouwd na de stageperiode, zullen hier meer overzichten en grafieken komen. Onderstaande afbeelding toont de staat van de dashboardpagina.

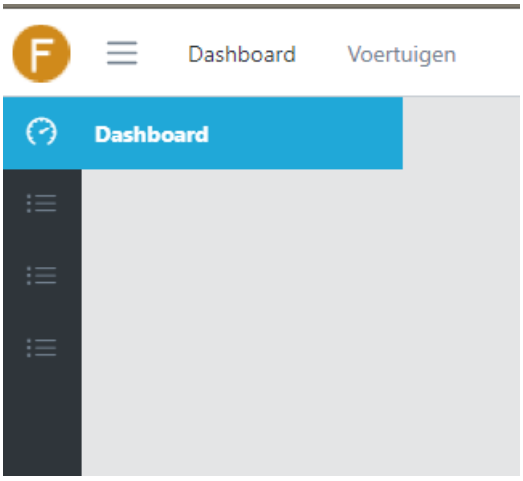



*Figuur 10 Dashboard eFenKa*

### 3.6.2 Navigatiebalken

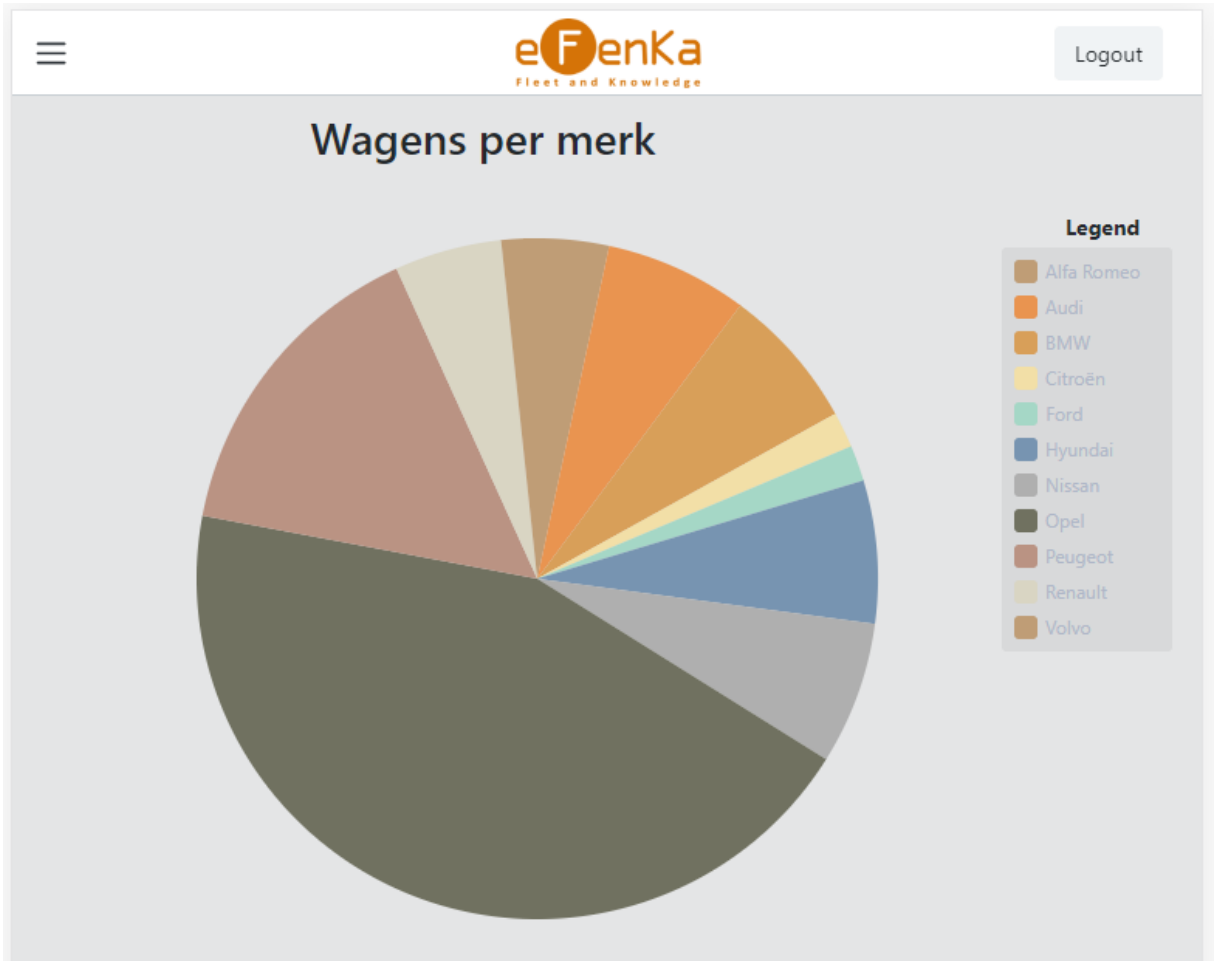
Op de figuur zijn eveneens de navigatiebalken te zien. In de linkse navigatiebalk staan onder de "Voertuigen" drie links: één naar het overzicht van de voertuigen, één naar het overzicht van de merken en één naar het overzicht van de verschillende deurtypes. Deze worden later nog toegelicht. De twee links in de bovenste navigatiebalk navigeren naar de dashboardpagina en het overzicht van de voertuigen.

Onderaan het menu is er een pijl te zien die het menu kan minimaliseren, maar toch zichtbaar houden. Met de muis kan er dan over de verschillende elementen gezweefd worden om de volledige tekst te zien. Onderstaande afbeelding toont dit principe. Het logo wordt eveneens vervangen door een kleine versie. Via het hamburgermenu bovenaan kan de navigatiebalk volledig ingeklapt worden.




*Figuur 11 Navigatiebalk in geminimaliseerde vorm*

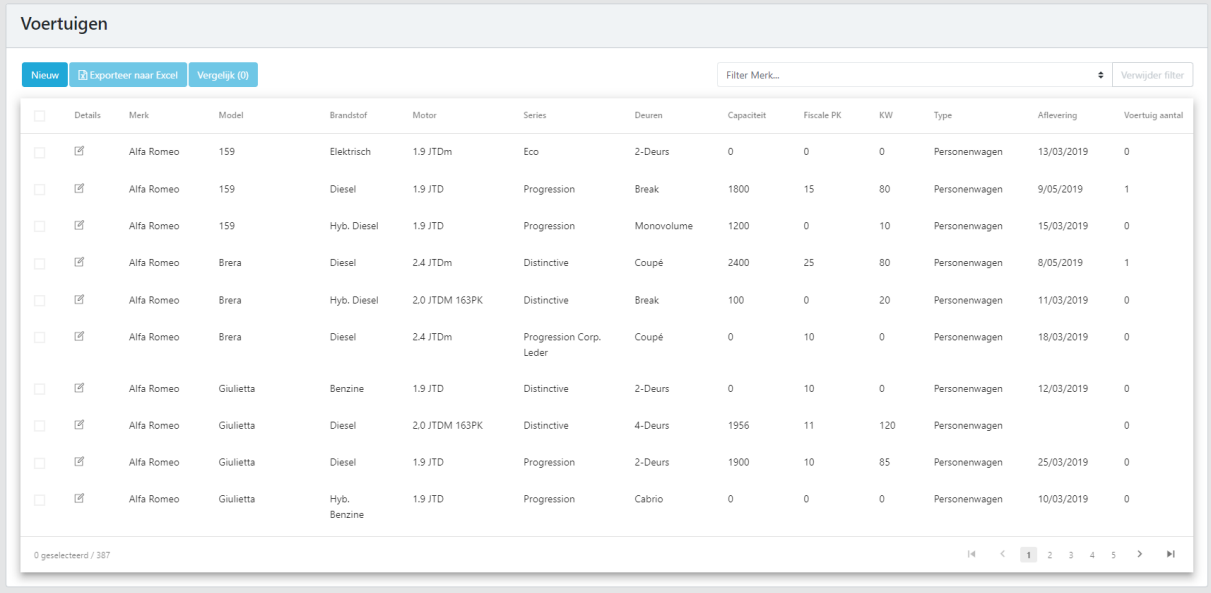
Daarnaast is de navigatiebalk ook *responsive* en past deze zich aan op mobiele apparaten zoals te zien op onderstaande afbeelding die een virtuele versie van de IPad simuleert. Het hamburgermenu klapt hier automatisch in om extra ruimte te creëren voor de eigenlijke content.




*Figuur 12 eFenKa op een mobiel apparaat*

### 3.6.3 Overzichten

De drie links onder de Voertuigenmodule navigeren allemaal naar een overzicht. Onderstaande afbeelding toont zo een overzicht, in dit geval van de voertuigen.

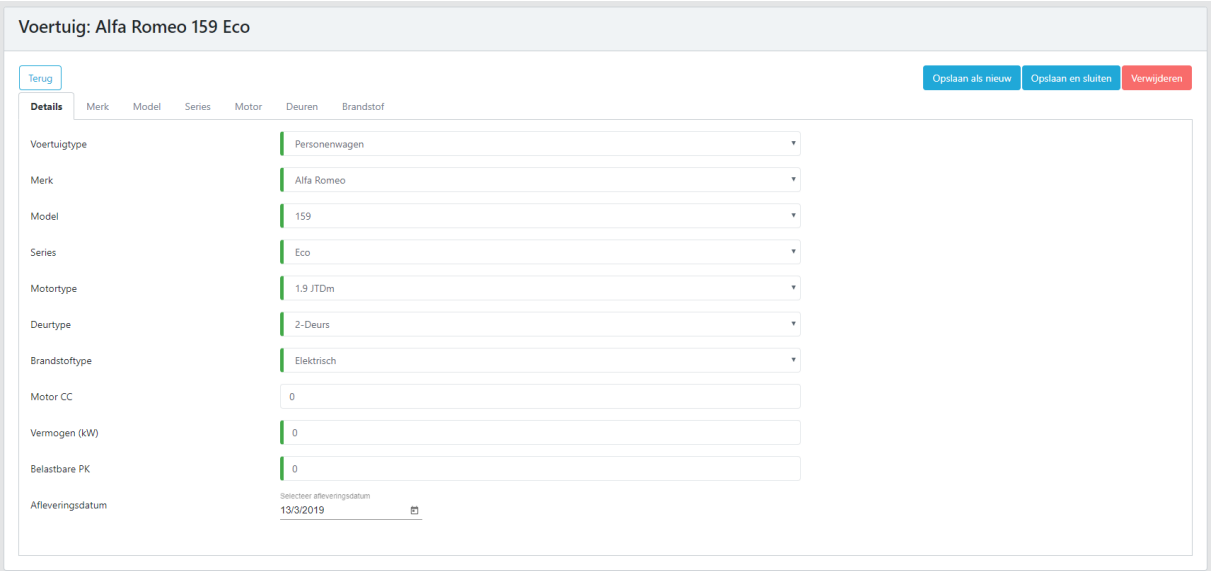



*Figuur 13 Overzicht voertuigen*

Hier zijn verschillende mogelijkheden waaronder het navigeren naar een pagina om een nieuw voertuig aan te maken. Naast die knop staan nog een knop voor het exporteren naar Excel en een knop voor het navigeren naar een pagina om voertuigen te vergelijken. Deze knoppen worden actief als er rijen geselecteerd worden. Rechts bovenaan de pagina is er de mogelijkheid tot filteren op merk. Als er op een rij geklikt wordt, wordt de gebruiker naar een detailpagina verwezen. Afhankelijk van de geklikte kolom wordt het overeenkomend tabblad getoond.

### 3.6.4 Detailpagina

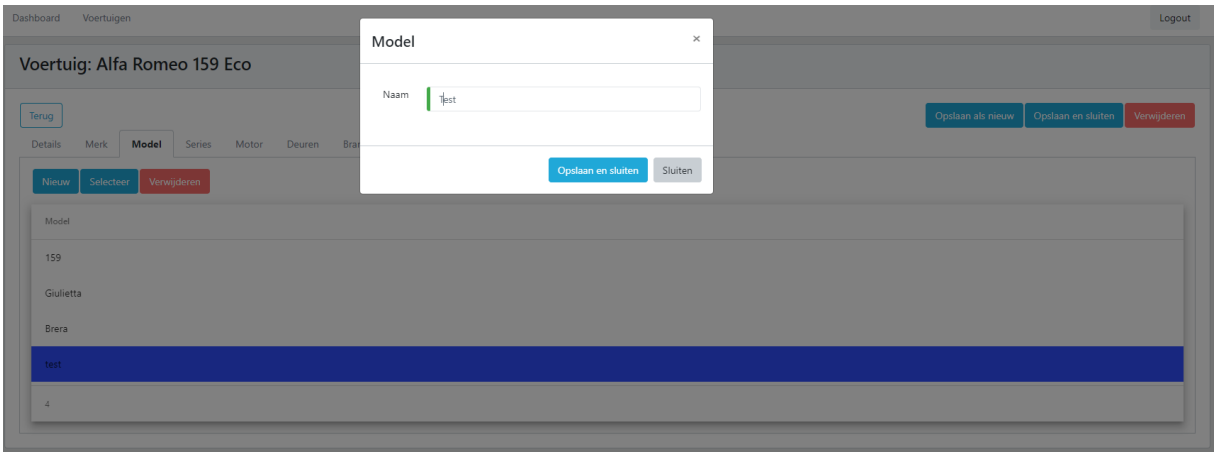
De detailpagina van een voertuig ziet er als volgt uit.




*Figuur 14 Detailpagina voertuig*

De pagina bestaat uit een “tabset” die verschillende tabbladen bevat. Afhankelijk van de geklikte kolom komt de gebruiker in het overeenstemmende tabblad terecht. Het eerste tabblad geeft een overzicht van het voertuig en geeft de mogelijkheid tot updaten van het voertuig. De overige tabbladen bevatten een overzicht van alle mogelijke eigenschappen van een voertuig zodat deze ook

aangemaakt of aangepast kunnen worden zoals te zien op onderstaande afbeelding. De beschikbare modellen, series en motortypes veranderen eveneens mee met het geselecteerde merk.

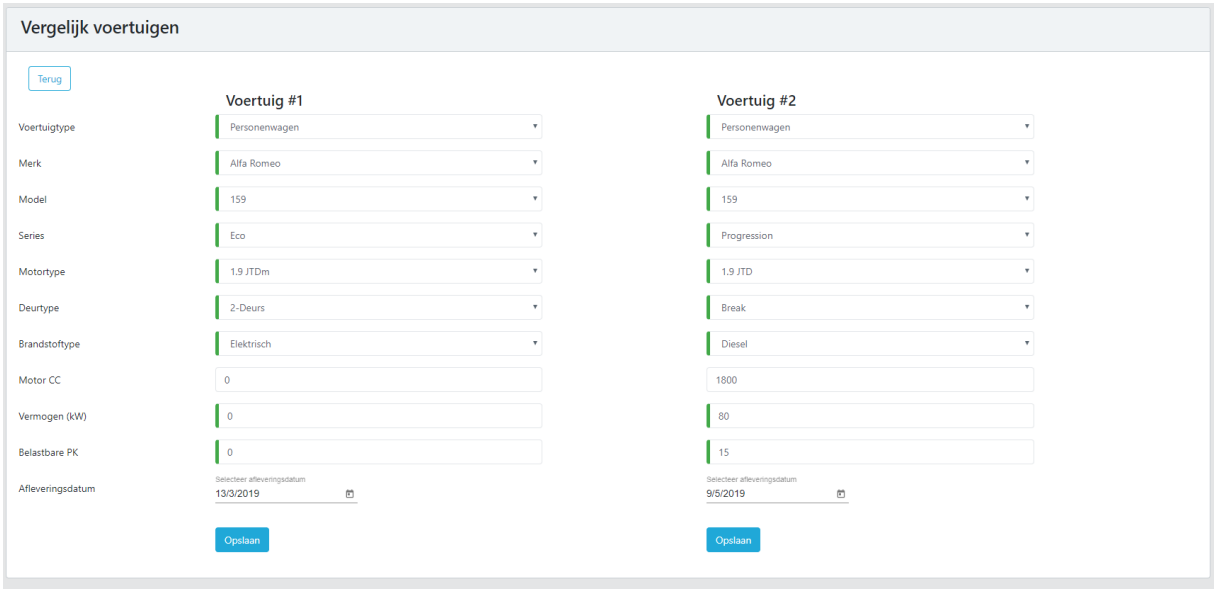



*Figuur 15 Pop-upscherm om een eigenschap van een voertuig aan te passen*

De detailpagina van het voertuig heeft rechtsboven drie knoppen: de eerste voor een bestaand voertuig aan te passen en dit op te slaan als nieuw voertuig, de tweede voor een voertuig te updaten en een derde om het voertuig te verwijderen. De tabbladen van de eigenschappen bevatten functionaliteit voor het creëren, updaten, verwijderen en selecteren van een eigenschap.

### 3.6.5 Vergelijkingspagina

Omdat het in de oude eFenKa-applicatie mogelijk is meerdere voertuigen tegelijk te bekijken, is er ook functionaliteit ingebouwd om voertuigen te vergelijken met elkaar. Dit is te zien op onderstaande afbeelding.

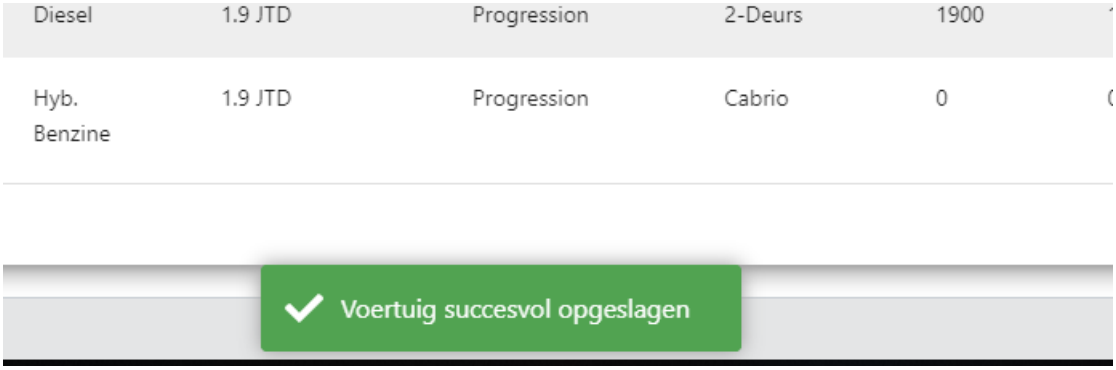



*Figuur 16 Pagina voor het vergelijken van voertuigen*

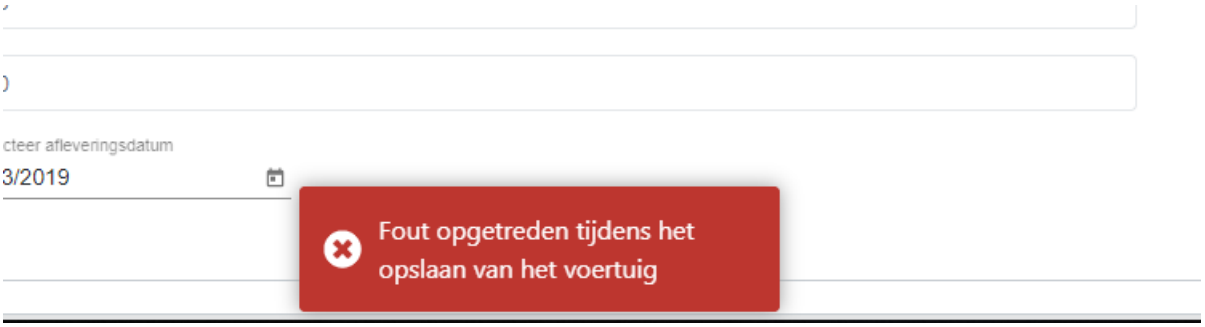
Het aantal voertuigen om te vergelijken is gelimiteerd op vier om het overzicht te bewaren. Elk voertuig kan hier eveneens aangepast worden.

### 3.6.6 Toasts

Om de gebruiker duidelijk te maken dat een actie geslaagd of gefaald is, worden er pop-upberichten of *toasts* getoond. Onderstaande afbeeldingen tonen een geslaagde en gefaalde operatie. De *toasts* verdwijnen na 5 seconden vanzelf.

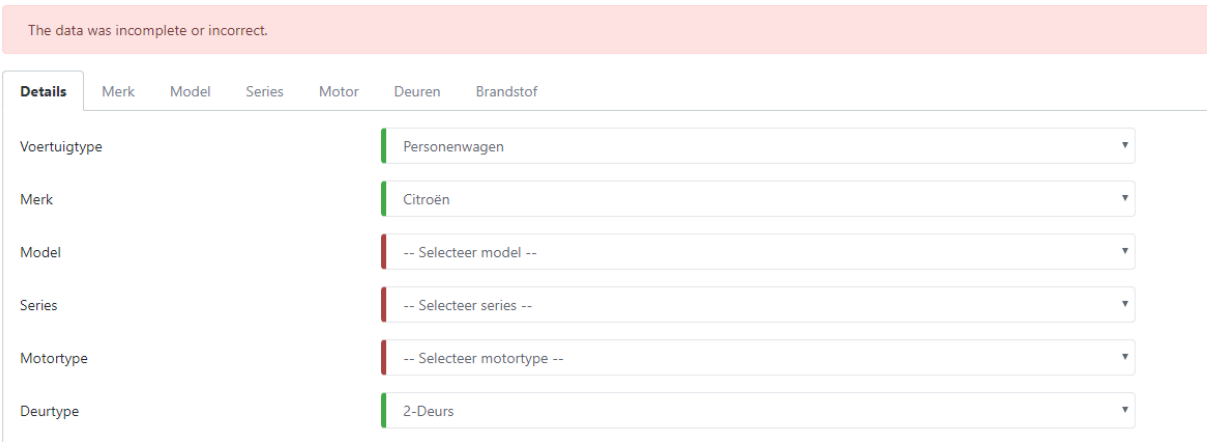



*Figuur 17 Toast geslaagde operatie*




*Figuur 18 Toast gefaalde operatie*

Het formulier maakt de gebruiker overigens duidelijk wat er precies misliep en waar de data incorrect is zoals op onderstaande afbeelding.




*Figuur 19 Foutboodschap na indienen ongeldig voertuig*

# 4 Reflectie stageopdracht

Vooraf was beslist dat de stageopdracht bestond uit het uitwerken van één of twee modules uit de eFenKa-applicatie. De Voertuigenmodule stond hier als belangrijkste module op de planning. De

Bestuurdersmodule was extra in het geval dat er tijd over was en was oorspronkelijk voor in het geval er een tweede stagestudent zou geweest zijn.

De Voertuigenmodule is helemaal afgewerkt tijdens de stage. De functionaliteit heb ik op relatief korte tijd kunnen realiseren. Nadat deze af was, heb ik in overleg met mijn bedrijfspromotor beslist de module verder te perfectioneren en heeft hij mij laten uitbreiden in onder andere Identity Server. De laatste weken ben ik veel bezig geweest met mijn onderzoek waarvoor ik veel heb kunnen bijleren zoals Vue.js, Redux en Redux-toepassingen binnen Angular en Vue.js (NgRx en Vuex).

Doorheen de stageperiode heb ik veel ervaring opgedaan op vlak van *hard* én *soft skills*. Zo heb ik mij verder kunnen specialiseren in .NET Core en meer specifiek het ASP.NET Core-framework door het maken van een API. Op frontend-vlak heb ik mij dan weer kunnen verdiepen in Angular en dankzij mijn onderzoek ook in Vue.js. Op vlak van *soft skills* heb ik vooral leren zelfstandig werken en vragen stellen wanneer nodig. Vooraf twijfelde ik of ik een groot project als dit wel alleen zou kunnen realiseren. Schoolopdrachten waren vaak zeer klein of wanneer ze toch van grotere omvang waren, moesten ze in team voltooid worden. Ik ben achteraf gezien wel blij dat ik zelfstandig aan dit project mocht werken. Ik heb er enorm veel uit geleerd en het heeft mij ook bevestiging gegeven naar de toekomst toe. Voor ik aan de stage begon was ik niet zeker of ik hierna ging werken of verder studeren, maar het maken van een applicatie van zulke omvang heeft mij overtuigd om na mijn opleiding te beginnen als .NET-ontwikkelaar. Hoewel het een zeer leuk proces was, ben ik ook tegen verschillende problemen aangelopen. Omdat ik tijdens de stageperiode met verschillende nieuwe technologieën bezig ben geweest, brachten deze veel nieuwe problemen met zich mee. Het ene probleem was al wat makkelijker op te lossen dan het andere. Dankzij de grote hoeveelheid programmeurs over de hele wereld waren veel oplossingen online te vinden. Sommige problemen waren helaas niet zo makkelijk te vinden, maar op die momenten kon ik steeds bij mijn bedrijfspromotor terecht die mij dan snel in de juiste richting kon sturen.

Ik vind dat ik een mooi eindresultaat kan afleren aan het einde van mijn stage. Ik heb een volledige module van eFenKa kunnen afwerken met login-systeem inbegrepen. Dankzij deze opdracht heb ik verschillende technologieën geleerd met als voornaamste: .NET Core, Progressive Web App, Redux (inclusief NgRx en Vuex), Vue.js, Identity Server en OData. Daarbij heb ik zelfstandig leren werken, vragen durven stellen wanneer nodig en voel ik dat ik helemaal klaar ben om mijn carrière in de IT te starten.

# II. Onderzoekstopic

# 5 Vraagstelling onderzoek

Er bestaan tegenwoordig verschillende frontendframeworks. Voor dit project wordt gebruikgemaakt van Angular, maar is dit de beste optie voor het maken van een Progressive Web App? Op welke manier vergelijkt Angular ten opzichte van Vue.js in het creëren van zo een PWA? Welk framework is hier het best voor geoptimaliseerd? En welke invloed heeft Redux op de prestaties van de applicatie?

# 6 Onderzoeksmethode

Voor de stageopdracht is er een verouderde webapplicatie opnieuw opgezet met nieuwe technologieën. De backend bestaat uit een API die is opgezet met behulp van ASP.NET Core, als frontendtechnologie is Angular 7 gebruikt. Voor het onderzoek ligt de focus op deze frontendtechnologie en op een mogelijk alternatief hiervoor, namelijk Vue.js. Daarbij wordt de invloed van het Reduxpatroon op een applicatie gemaakt met deze technologieën onderzocht. Deze twee applicaties worden gecreëerd als Progressive Web Apps.

Om te bepalen welke technologie het meest aangewezen is, wordt eerst en vooral onderzocht wat de voor- en nadelen zijn van Angular en Vue.js. Aan de hand van een literatuurstudie wordt dan bepaald wanneer Angular meer aangewezen is en wanneer Vue.js. Bij dit deel van het onderzoek dient ook rekening gehouden te worden met hoe beide technologieën geoptimaliseerd zijn voor het ontwikkelen van PWA's en hoe goed ze samenwerken met Redux.

Vervolgens wordt met beide technologieën een PWA gebouwd. De prestaties van beide applicaties worden getest aan de hand van verschillende tools. Hier wordt bij beide technologieën ook de vergelijking gemaakt tussen een applicatie met Redux en een applicatie zonder. Om meetbare resultaten te bekomen wordt het open source project Lighthouse, gemaakt door de ontwikkelaars van Google Chrome, gebruikt. Beide applicaties worden met deze tool getest op prestatie en of ze voldoen aan alle normen van een goede Progressive Web App. Verder wordt gebruikgemaakt van de profiler die ingebouwd is in de Google Chrome DevTools om te onderzoeken wat de prestaties van de toepassing beïnvloedt, of dit verschillend is bij de onderzochte technologieën en welke invloed het gebruik van Redux hierop heeft.

Om relevante resultaten te bekomen wordt een kleine Single Page Application opgezet die een formulier zal afhandelen en een tabel zal vullen met informatie verkregen van een web-API. De applicatie zal zowel *GET, POST, PUT* als *DELETE requests* uitvoeren om te onderzoeken hoe snel de verschillende applicaties deze afhandelen en vooral hoe Redux de prestaties mogelijk kan beïnvloeden door *state management*.

# 7 Literatuurstudie

## 7.1 Inleiding

In de inleiding worden de verschillende technologieën die in dit onderzoek aan bod komen toegelicht.

### 7.1.1 Angular

Angular is een platform dat webapplicaties ontwikkelen makkelijk maakt. Het combineert declaratieve templates, *dependency injection*, end-to-endhulpmiddelen en geïntegreerde best practices om ontwikkelingsuitdagingen op te lossen. Angular stelt ontwikkelaars in staat applicaties te creëren die werken op zowel web, mobiel als desktop. [2]

Het is een frontendframework gebaseerd op TypeScript, dat zelf dan weer compileert naar JavaScript en dat wordt onderhouden door het Angular Team van Google en een gemeenschap van individuen en bedrijven. Angular is de opvolger van AngularJS en heeft op 18 oktober 2018 een versie-update gehad naar Angular 7. [3]

### 7.1.2 Vue.js

Vue is een progressief, open source, JavaScript framework voor het maken van gebruikersinterfaces. Anders dan monolithische frameworks, is Vue vanaf nul ontworpen om incrementeel te kunnen worden aangepast. De *core library* is gefocust op de weergavelaag en is makkelijk te leren en te integreren met andere *libraries* of bestaande projecten. Aan de andere kant is Vue.js ook perfect in staat gesofisticeerde Single Page Applications aan te sturen wanneer het gecombineerd wordt met moderne hulpmiddelen en ondersteunende *libraries*. [4]

### 7.1.3 Progressive Web App

Progressive Web Apps (PWA's) zijn webapplicaties die laden als gewone webpagina's of websites, maar de gebruiker extra functionaliteit bieden zoals offline werken, pushberichten sturen, en toegang tot apparaathardware die normaal enkel beschikbaar is voor *native* applicaties. PWA's combineren de flexibiliteit van het web met de ervaring van een *native* applicatie. [5]

In 2015 bedachten ontwerper Frances Berriman en Google Chrome-ingenieur Alex Russell de term "Progressieve Web App" om apps te beschrijven die profiteren van nieuwe functies die worden ondersteund door moderne browsers, waaronder *service workers* en webapp manifestbestanden, waarmee gebruikers webapps kunnen upgraden tot progressieve webapplicaties in hun *native* besturingssysteem (OS). [6]

### 7.1.4 Redux

Redux is een voorspelbare statuscontainer voor JavaScript-apps. Het helpt applicaties te schrijven die zich consistent gedragen, in verschillende omgevingen worden uitgevoerd (client, server en *native*) en die eenvoudig te testen zijn. Redux werkt met Angular, Vue.js en elke andere *view library*. Redux is klein (2 kB, inclusief afhankelijkheden), maar heeft een grote hoeveelheid aan add-ons ter beschikking. [7]

Redux is eigenlijk een opslagplaats die de staat van de toepassing bevat. Het kan in applicaties, die bijvoorbeeld gemaakt zijn met Angular, de staat van elke component bijhouden en updaten. Hierdoor wordt de interactie tussen verschillende componenten bij grote applicaties overzichtelijker. Het managen van deze staten gebeurt door Reducers die aangestuurd kunnen worden door acties. [8]

## 7.2 Progressive Web Apps in Angular en Vue.js

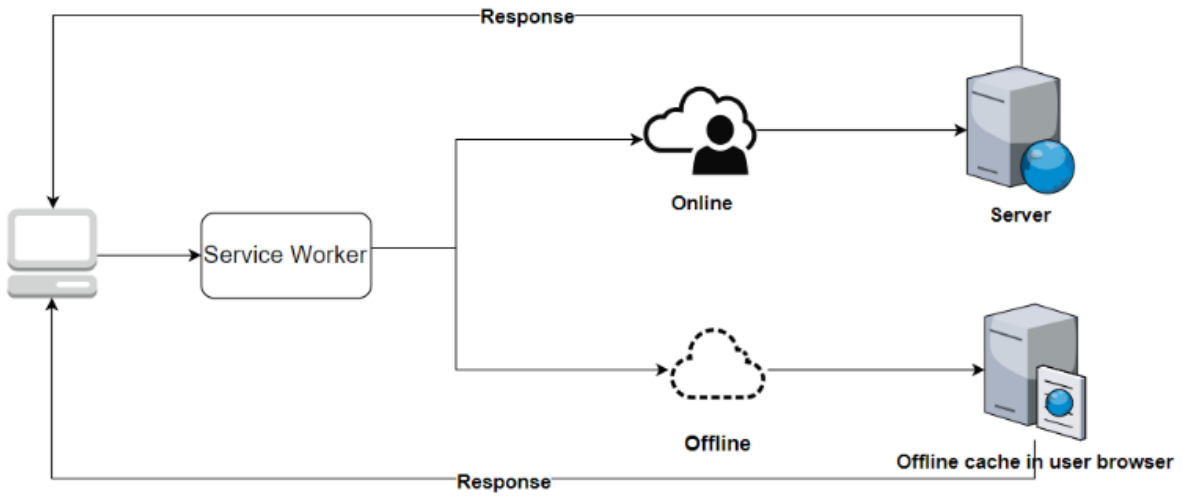
Er zijn verschillende opties voor het maken van PWA's. Het is mogelijk zelf de benodigde componenten toe te voegen zoals een Manifestbestand en Service Worker. Maar Angular en Vue

hebben beide ook ingebouwde ondersteuning voor het maken van Progressive Web Apps. Angular heeft dit sinds Angular 5 en bij Vue zit het in de Vue-CLI. [9]

### 7.2.1 Service Worker

Eén van de belangrijkste elementen van een Progressive Web App is de Service Worker. Een artikel op DZone verklaart Service Workers als volgt: Service Workers breiden het webwerkersmechanisme uit, waardoor scripts op de achtergrond en in een thread kunnen starten die gescheiden is van de hoofdthread van de browser. Service Workers worden gebruikt als een soort van proxy tussen de client- en server-side-applicatie. Dankzij Service Workers kunnen statische bronnen in de cache opgeslagen worden of specifieke *API-calls* geactiveerd worden (bijvoorbeeld door een API te vragen voor een inkooplijst die op de server is opgeslagen en deze te downloaden, zelfs zonder toegang tot het internet). Verder ondersteunen ze mechanismen als pushberichten en achtergrondsynchronisaties. [10]

Onderstaande afbeelding toont schematisch waar de Service Worker wordt gebruikt.




*Figuur 20 Mechanisme Service Worker voor het werken als interceptor van requests van een applicatie*

In de praktijk zorgt de Service Worker ervoor dat de website zowel online als offline zal werken. De Service Worker checkt bij het bezoeken van de webpagina of er een internetconnectie tot stand gebracht kan worden. Als dit het geval is, zal er een *call* gebeuren naar de server waarop de website gehost wordt. Zo krijgt de gebruiker de laatste versie van de webpagina te zien.

Als er geen connectie met het internet is, gaat de Service Worker na of de webpagina zich nog in browsercache bevindt. Als dit het geval is, krijgt de gebruiker die pagina te zien en kan er offline gebruikgemaakt worden van de website. Deze techniek heet “Network or cache”.

Andere veel gebruikte technieken zijn “Cache only” en “Cache and update”.

“Cache only” wordt normaal gebruikt voor statische content die nooit zal veranderen. Een *web call* telkens er naar de site gesurft wordt, is in dit geval niet nodig.

“Cache and update” is een mix van bovenstaande twee technieken. Zowel de standaard PWA-implementatie binnen Vue.js als binnen Angular gebruikt deze techniek overigens. Deze strategie laat de gebruiker eerst de pagina in cache zien zodat de website meteen reageert. Vervolgens gebeurt er een call naar de externe server als er een internetverbinding is. Als er een nieuwe versie van de site

beschikbaar is, zal deze getoond worden de volgende keer de url bezocht wordt, of bij het herladen van de huidige pagina.

### 7.2.2 Manifestbestand

Het manifestbestand in een PWA is een eenvoudig JSON-bestand dat de browser informeert over de webtoepassing en hoe de app zich moet gedragen wanneer het geïnstalleerd is op een mobiel apparaat of het bureaublad van de gebruiker. Chrome heeft een manifestbestand nodig om de prompt “Aan beginscherm toevoegen” weer te geven.

Een typisch manifestbestand bevat informatie zoals de applicatienaam, pictogrammen die moeten worden gebruikt, de start-url die moet worden aangeroepen bij het starten en meer. [11] Volgende afbeelding toont een voorbeeld van een basis manifestbestand in Angular.


*Figuur 21 Voorbeeld manifestbestand in Angular*

## 7.3 Redux in Angular en Vue.js

Zowel Angular als Vue.js hebben een eigen *library* voor *state management*. Voor Angular is de voornaamste *library* NgRx, voor Vue.js is dit Vuex.

NgRx Store zorgt voor *reactive state management* voor Angularapplicaties en is geïnspireerd door Redux. Vuex is een *state management pattern* en *library* voor Vue.js-applicaties. Het dient als een gecentraliseerde opslagplaats voor alle componenten in een toepassing, met regels die ervoor zorgen dat de staat alleen op een voorspelbare manier kan worden gemuteerd. [12] [13]

Gedetailleerde toelichting van Redux, NgRx en Vuex staan onder 7.6 Redux in Angular versus Redux in Vue.js.

## 7.4 Angular versus Vue.js

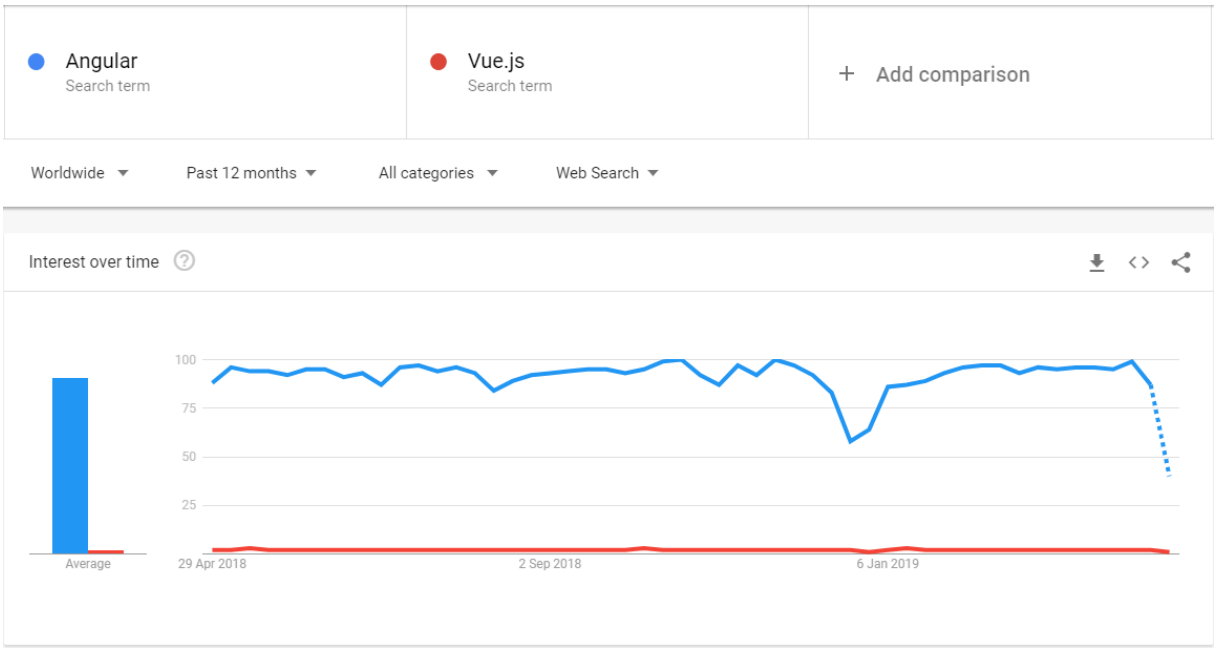
Om een vergelijking te maken tussen beide frameworks wordt onderzoek gedaan naar de populariteit van beide frameworks, de ondersteuning ervan, gebruiksgemak en efficiëntie, prestaties en hun leercurve.

### 7.4.1 Populariteit

Om een idee te krijgen van de populariteit van Angular en Vue.js wordt er een vergelijking gemaakt op Google zoekopdrachten, GitHub *repository* sterren, de Stack Overflow populariteit en LinkedIn vacatures.

#### 7.4.1.1 Google Trends

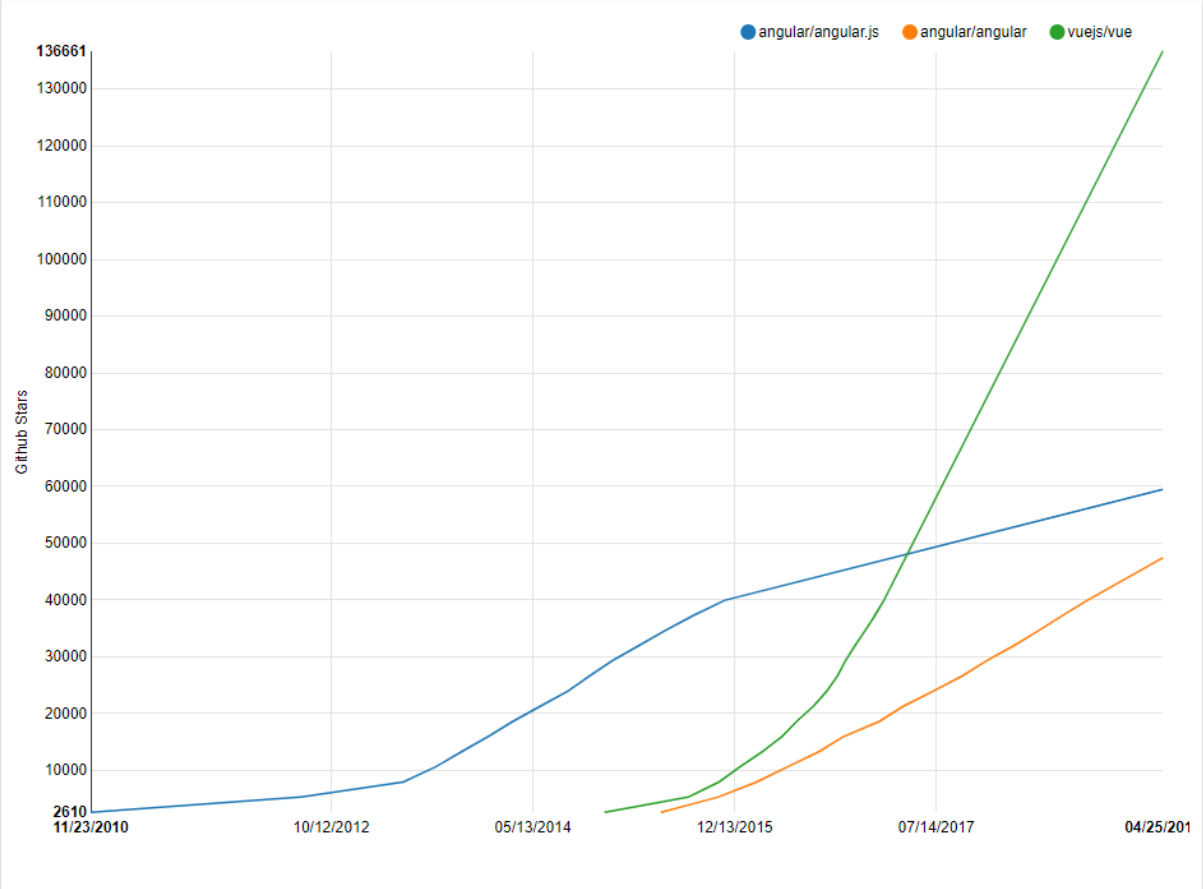
Als we kijken naar de populariteit van beide frameworks over het afgelopen jaar op Google, is Angular duidelijk de winnaar. Uit een vergelijking van Google Trends blijkt dat Angular tot 45 keer meer gezocht wordt. Onderstaande grafiek geeft de vergelijking van Google Trends weer.




*Figuur 22 Angular vs Vue.js Google zoekopdrachten*

#### 7.4.1.2 GitHub *repository* sterren

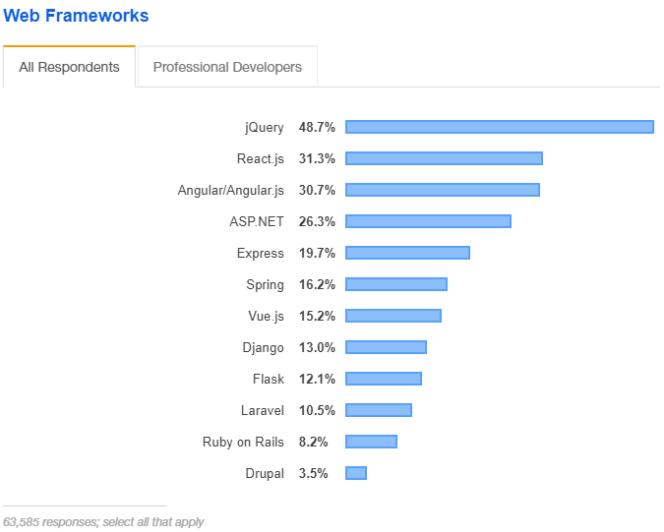
Als we dezelfde vergelijking naar populariteit maken op GitHub, geeft dit een heel ander beeld. Uit een vergelijking van het aantal gekregen sterren tussen Vue.js, Angular en zijn voorloper Angular.js blijkt dat Vue.js veel populairder is op GitHub. Zelfs na het optellen van de sterren van Angular en Angular.js blijft Vue.js de duidelijke winnaar. Volgende grafiek geeft het totaal aantal sterren weer sinds het ontstaan van de frameworks op GitHub.



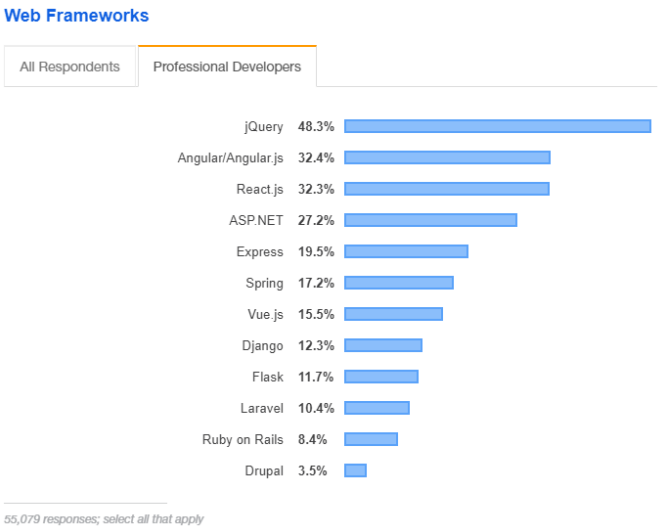

*Figuur 23 Angular en Angular.js vs Vue.js GitHub sterren*

#### 7.4.1.3 Stack Overflow populariteit

Uit een jaarlijkse enquête op Stack Overflow, te zien op de twee onderstaande afbeeldingen, met de vraag welk webframework programmeurs gebruiken, kwam Angular eruit als duidelijke winnaar. Ongeveer 30,7% van de webontwikkelaars en 32,4% van de professionele webontwikkelaars zeggen Angular of Angular.js te gebruiken. Voor Vue.js is dit slechts 15,2% van de webontwikkelaars en 15,5% van de professionals. [14]




*Figuur 24 Populairste webframeworks bij webontwikkelaars in 2019*




*Figuur 25 Populairste webframeworks bij professionele webontwikkelaars in 2019*

#### 7.4.1.4 LinkedIn vacatures

Volgens onderzoek uit december 2018 naar het aantal vacatures op LinkedIn is Angular dan weer de duidelijke winnaar met ongeveer 70.000 jobaanbiedingen. Vue.js is met ongeveer 10.000 aanbiedingen voor ontwikkelaars heel wat minder populair. [15]

#### 7.4.1.5 Conclusie

Angular is het oudste framework van de twee als we zijn voorloper Angular.js meerekenen en wordt gebruikt in vele grote organisaties zoals Google, Forbes, WhatsApp en Instagram. Angular wordt door het onderhoud van Google meer als vaste waarde beschouwd. Vue.js daarentegen is de sterkste groeier en wint snel aan populariteit. Er zijn ook enkele grote bedrijven die inzetten op Vue.js zoals Alibaba en GitLab, maar het heeft nog niet de status van Angular. [16]

### 7.4.2 Ondersteuning

Van de twee frameworks geniet Angular duidelijk de beste ondersteuning. Het wordt ondersteund en onderhouden door Google wat de groei van Angular nog een tijd garandeert. Met een grote update elke zes maanden ben je als ontwikkelaar snel up-to-date met nieuwe features binnen Angular. Daarnaast bezorgt het Angularteam ook regelmatige updates over nieuwe versies, releases en afschrijvingstermijnen.

Vue.js heeft een tool voor het helpen migreren naar nieuwe versies, maar heeft een onduidelijker overzicht van opkomende versies en toekomstplannen. Daarnaast geniet het ook niet de onmiddellijke ondersteuning van een groot bedrijf als Google omdat Vue.js gestart is door één persoon, Evan You, een voormalig werknemer van Google. [16]

### 7.4.3 Gebruiksgemak en efficiëntie

Angular is een volwaardig framework omdat het een goede basis biedt om een applicatie te bouwen zonder externe toepassingen. Er moet niet naar *libraries*, routingsoplossingen en de structuur gekeken worden. Dit heeft als gevolg dat Angular ook het grootste is van de twee met een bestandsgrootte van ongeveer 500KB. Vue.js aan de andere kant is flexibeler en universeler dan Angular.

Vue.js situeert zich ergens tussen een *library* en een volwaardig framework. Dit helpt code efficiënt te houden met een betere balans tussen interne afhankelijkheden en flexibiliteit. Dit maakt de bestandsgrootte ook aanzienlijk kleiner (ongeveer 80 KB). Dit heeft wel als gevolg dat het gebruiksgemak in het algemeen als minder ervaren wordt ten opzichte van Angular.

Omdat Vue.js het jongste framework is, heeft het zich kunnen baseren op Angular.js en React (eveneens een JavaScript framework). Het heeft zo van de start al veel geleerd uit de fouten en problemen van beide frameworks. Dit heeft ervoor gezorgd dat Vue.js veel goede eigenschappen van beide frameworks bevat en toch kleiner is in formaat en efficiënter te werk kan gaan met resources. [16] [15]

### 7.4.4 Prestaties

Als het op prestaties aankomt van applicaties gebouwd met beide frameworks, is er geen duidelijke winnaar aan te duiden. De prestatie van de applicatie zal vooral afhankelijk zijn van de grootte van de applicatie en de optimalisatie van de geschreven code en slechts in zeer kleine mate van het gebruikte framework. [16]

### 7.4.5 Leercurve

Als bedrijven kiezen voor Vue.js is het meestal vanwege zijn kleine leercurve. Vue.js biedt hogere aanpasbaarheid en is daarom gemakkelijker te leren dan Angular. Daarnaast biedt het standaard JavaScript-ondersteuning, terwijl Angular TypeScript ondersteunt. Webontwikkelaars hebben meestal al een JavaScript-achtergrond wat de stap naar Vue.js kleiner maakt. Verder heeft Vue.js een overlap met zowel Angular als React met betrekking tot hun functionaliteit zoals het gebruik van componenten. Vandaar dat de overgang naar Vue.js van één van beide frameworks relatief gemakkelijk is. De eenvoud en flexibiliteit van Vue.js heeft ook een nadeel, het nodigt meer uit tot het schrijven van slechte code, waardoor het moeilijk wordt om te debuggen en te testen.

Angular heeft een grotere leercurve. Voor het beheersen van Angular moet je concepten zoals TypeScript en MVC leren. Hoewel het tijd kost om Angular te leren, is het de investering waard omdat het een goed beeld geeft over hoe frontend in het algemeen werkt. [15] [16]

### 7.4.6 Conclusie

Beide frameworks hebben hun voor- en nadelen. Het kiezen tussen de twee frameworks komt vooral neer op persoonlijke voorkeur. Belangrijke aspecten in deze beslissing zijn:

- TypeScript of JavaScript: Angular is volledig gebaseerd op TypeScript en programmeren gebeurt ook in deze taal. Vue.js daarentegen is volledig op JavaScript gebaseerd, maar het is wel de overgang aan het maken naar TypeScript.
- Leercurve: Moet er snel een project gemaakt worden in één van de twee frameworks en is er geen kennis aanwezig van één van deze twee, dan is de beste keuze Vue.js door zijn kleinere leercurve.
- Ondersteuning en zekerheid: Angular is een meer gevestigde waarde en veiligere gok naar de toekomst toe dan Vue.js doordat het onderhouden wordt door Google.
  Hoewel het niet ondersteund wordt door een groot bedrijf, kent Vue.js een zeer sterke groei en geniet het steeds meer interesse van enkele grote bedrijven.

## 7.5 PWA's in Angular versus PWA's in Vue.js

Welk framework is het best geoptimaliseerd voor Progressive Web Apps? In dit onderdeel wordt de ingebouwde ondersteuning voor PWA's in Angular en Vue.js vergeleken. Ook wordt gekeken naar frameworks die worden ondersteund door Angular en Vue.js.

### 7.5.1 PWA-ondersteuning in Angular

PWA-ondersteuning toevoegen aan een Angular 7 project is relatief eenvoudig. Eerst en vooral dient de ontwikkelaar de Angular CLI te installeren en een eerste project gemaakt te hebben. Vervolgens kunnen de benodigdheden voor een PWA eenvoudig toegevoegd worden via het commando "ng add @angular/pwa". Achterliggend zorgt dit commando ervoor dat er afhankelijkheden naar de @angular/pwa *library* en @angular/service-worker *library* worden toegevoegd aan het Angular-project. Daarnaast wordt de Service Worker geactiveerd in het angular.json-bestand. Er wordt eveneens een ngsw-config.json-bestand toegevoegd dat de configuratie bevat over het cachen van data. Als laatste wordt de Service Worker geïmporteerd en geregistreerd in de App Module. [10]

Om van de applicatie een volwaardige Progressive Web App te maken, moet er nog een manifest.json-bestand opgesteld worden. Zoals eerder uitgelegd bevat dit bestand alle informatie over de applicatie. Voeg dit bestand toe aan het index.html-bestand.

Omdat dit geen applicatie is uit de App Store en de gebruiker best op de hoogte gehouden wordt van updates, is het mogelijk onderstaande code toe te voegen aan het initialisatieblok van de App Component.

```typescript
import {Component, OnInit} from '@angular/core';
import {SwUpdate} from "@angular/service-worker";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {

  constructor(private swUpdate: SwUpdate) {
  }

  ngOnInit() {
    if (this.swUpdate.isEnabled) {
      this.swUpdate.available.subscribe( next: () => {
        if (confirm("New version available. Load New Version?")) {
          window.location.reload();
        }
      });
    }
  }
}
```

*Figuur 26 Code voor het informeren over een update van de PWA*

### 7.5.2 PWA-ondersteuning in Vue.js

Ook Vue CLI 3 bevat ingebouwde PWA-ondersteuning. Het toevoegen hiervan is bijna identiek aan Angular.

Na het installeren van de Vue CLI en het aanmaken van een project, kunnen via het commando “vue add @vue/pwa” de benodigde elementen voor een PWA toegevoegd worden. Bij het aanmaken van een nieuw project via de Vue CLI is het ook mogelijk meteen de “PWA support”-optie aan te vinken in de wizard.

Anders dan Angular wordt in een Vue.js-project meteen een manifest.json-bestand aangemaakt in de “public”-map. Ook wordt er een registerServiceWorker.json-bestand gegenereerd. Dit bestand is enkel voor het registreren van de Service Worker. Daarnaast is er ook een bestand nodig met de configuratie van de Service Worker, meestal genaamd service-worker.js. Dit bestand bevat eveneens de configuratie nodig voor het cachen van de juiste data. [17]

### 7.5.3 Conclusie

Zowel Angular als Vue.js hebben zeer goede en zeer gelijkaardige Progressive Web App-ondersteuning. In beide gevallen is het veranderen van een standaard project in een PWA-project mogelijk door het uitvoeren van enkele commando’s in elk hun CLI’s.

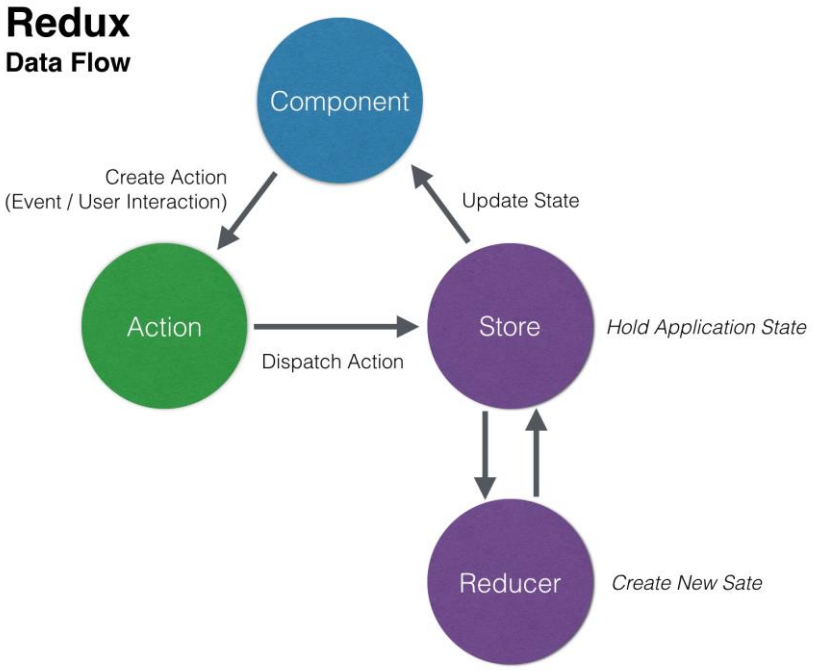
## 7.6 Redux in Angular versus Redux in Vue.js

Om Redux toe te passen in Angular en Vue.js wordt in dit onderzoek gebruikgemaakt van respectievelijk NgRx en Vuex. Het zijn beide *libraries* die gebaseerd zijn op Redux en alle nodige componenten voor het toepassen van Redux bevatten.

### 7.6.1 Werking van Redux

Bij de algemene werking van Redux zijn er vier belangrijke elementen: de Component, de Action, de Store en de Reducer. De Component is een fragment van de gebruikersinterface. Verschillende componenten kunnen samen een pagina vormen. Vanuit de Component kan een *event* getriggerd worden dat vervolgens vertaald wordt naar een Action die de staat van de applicatie verandert. Vervolgens worden de Action en de oude Application State uit de Store naar de Reducer gestuurd. De Reducer maakt aan de hand van deze twee parameters een State Object dat de oude Application State in de Store vervangt. De staat van de Component is nu gewijzigd naar de nieuwe staat. Onder

volgende afbeelding, die schematisch de werking van Redux toont, worden de belangrijkste concepten verder toegelicht. [18]




*Figuur 27 Gegevensstroom in Redux*

#### 7.6.1.1 Actions

Actions binnen Redux zijn JavaScript-objecten die informatie uit de applicatie naar de Store verzenden. Zij zijn de enige bron van informatie voor de Store. Via het "dispatch"-commando spreken ze de Store aan. Actions bevatten een type dat gedefinieerd wordt door een *string*. Dit type geeft aan om welk soort actie het gaat. [19]

#### 7.6.1.2 Reducers

Reducers zijn functies die de Action en de huidige State van de applicatie ontvangen uit de Store. Zij zijn de enige objecten die een nieuwe State kunnen aanmaken. De Action-objecten geven enkel de nodige informatie om te weten welke actie ondernomen dient te worden. De reducer-functie bestaat normaal uit een switch-case-statement dat switcht over alle mogelijke Action-types. [20]

#### 7.6.1.3 Store

De Store staat in voor verschillende zaken:

- Hij bevat de staat van de applicatie of de State
- Hij geeft componenten toegang tot deze State.
- De Store registreert *subscribers* (componenten die "luisteren" of er een nieuwe staat beschikbaar is)
- Hij behandelt het afmelden van deze "*subscribers*"

Een applicatie kan maar één Store bevatten. Om toch opsplitsing te hebben in de data is het mogelijk de State op te splitsen. Dit principe heet "reducer composition". Deze techniek zorgt dat het overzicht in grote applicaties behouden blijft dankzij de opsplitsing van data in logische blokken. [21]

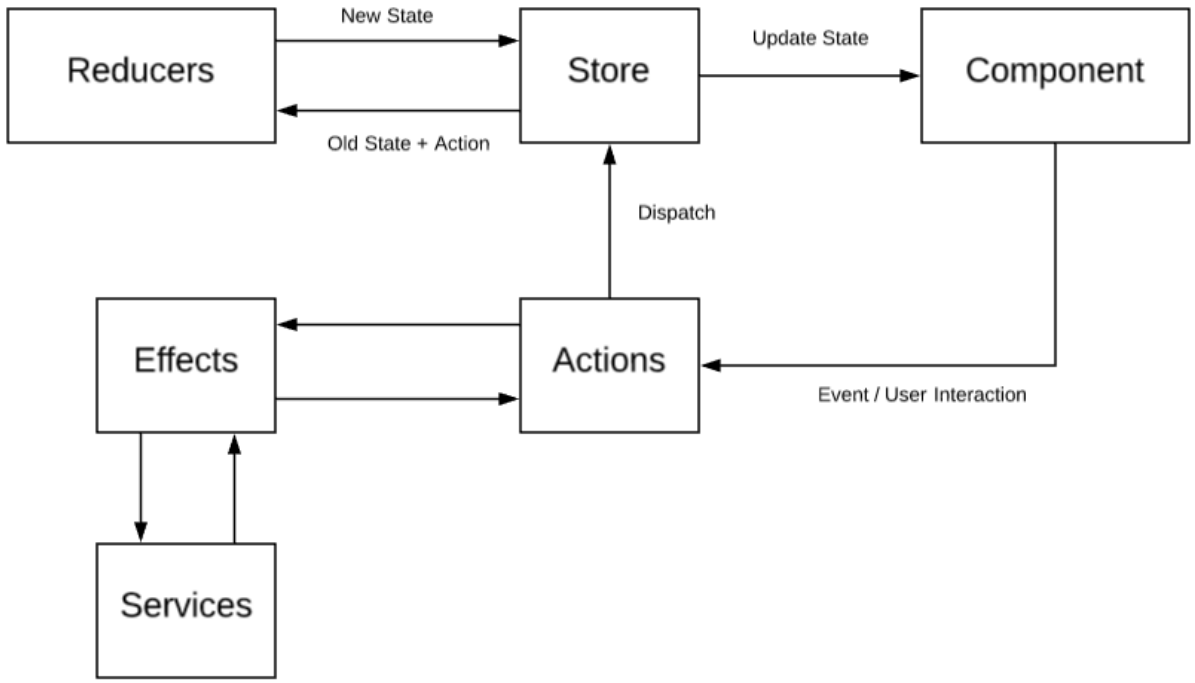
### 7.6.2 NgRx

NgRx is een Reduxtoepassing specifiek ontworpen voor Angular. NgRx of het Reduxpatroon gebruiken heeft verschillende voordelen. Zo zal de applicatie zich consistenter gedragen omdat de staat van de applicatie niet wordt aangepast, maar vervangen door een nieuwe. Het testen van de NgRx-implementatie zal ook vergemakkelijken omdat de staat aangepast wordt door zuivere functies. Er zijn ook nadelen verbonden aan NgRx. Voor elke verandering zijn er Actions nodig en wordt de Application State in de Store vervangen door een nieuwe. Dit kan resource-intensief zijn.

NgRx is ook niet in elke applicatie aangeraden. NgRx zal vooral nuttig zijn in medium tot grote projecten wanneer veel componenten dezelfde staat delen en waarbij data door veel verschillende componenten opgevraagd en bewerkt wordt.

Zoals onderstaand schema toont, is de flow van NgRx gelijkend aan de algemene Reduxflow. NgRx heeft, anders dan algemene Redux, nog een extra kenmerk, namelijk Effects. Effects werken op Angular services en andere elementen die geen componenten zijn in Angular. In de meeste gevallen spreken de Effects de methoden van de services aan die instaan voor de API-*calls*. Zij zorgen ervoor dat de data binnen de applicatie up-to-date blijft. Dit is vooral van belang als andere gebruikers dezelfde data manipuleren.

Daarnaast bezit NgRx ook Selectors. Omdat de State Tree in de Store een groot object kan worden is het handig als men enkel het benodigde onderdeel eruit kan halen. Hiervoor worden Selectors gebruikt. [22]




*Figuur 28 Gegevensstroom in NgRx*

### 7.6.3 Vuex

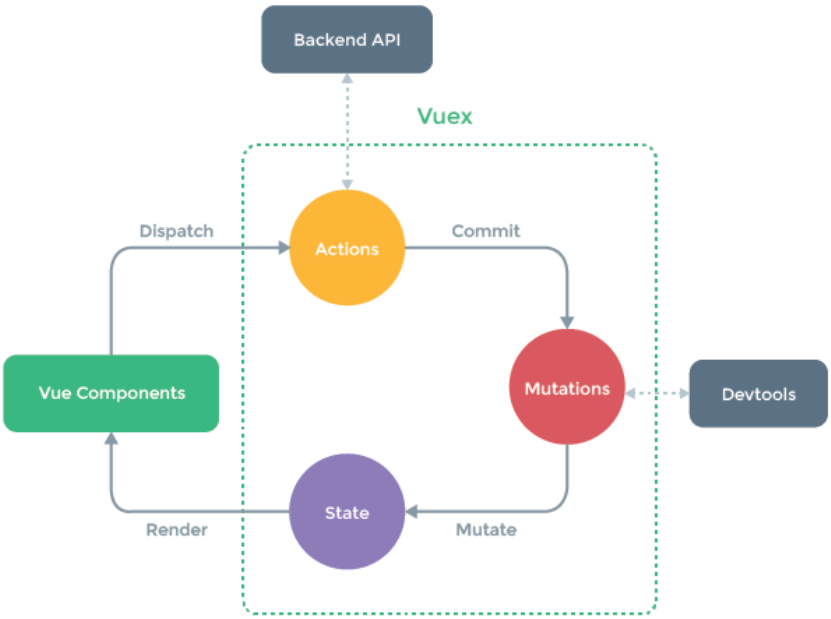
Vuex is minder gelijkend aan het typische Reduxpatroon. Het maakt ook gebruik van Components, Actions, een Store en Reducer, maar bevat enkele fundamentele verschillen. Op de officiële Vuex-pagina en op onderstaand schema worden Store en Reducer respectievelijk State en Mutations genoemd, maar in essentie hebben ze dezelfde functie.

Vuex is geen rechtstreekse implementatie van Redux, maar meer een mix van Redux en Flux, het patroon waarop Redux gebaseerd is. Zoals onderstaand schema toont, gaan Actions meteen naar de Mutations zonder eerst langs de Store (of State) te passeren zoals dat normaal gebeurt in Redux. Dit komt doordat de State niet wordt vervangen door een compleet nieuw object. In plaats daarvan zorgen Mutations, zoals de naam zegt, voor het muteren of aanpassen van de huidige staat. De Mutations houden wel steeds een oude versie van de staat bij zodat "time travel debugging", teruggaan tijdens het debuggen om de oude staat te bekijken, mogelijk is. [23]

Net als in NgRx is het mogelijk om de State op te splitsen, maar ook dit gebeurt hier op een andere manier. In Vuex gaat dit aan de hand van Modules. Deze modules lijken op het gebruik van een tweede Store. Ze bezitten elk hun eigen State, Actions, Mutations en Getters. Deze Getters zijn vergelijkbaar aan de Selectors binnen NgRx.

Een ander groot verschil met NgRx is het ontbreken van Effects. In Vuex is dit ook niet echt nodig omdat hier geen gebruikgemaakt wordt van services zoals in Angular. In plaats daarvan gebeuren API-*calls* rechtstreeks in de Actions.

Ten slotte bevat het schema ook een verbinding van Mutations naar DevTools. Deze pijl visualiseert de Mutations die ook de Store in de Vue-DevTools aanspreken om het bovenvermelde "time travel debugging" mogelijk te maken.




*Figuur 29 Gegevensstroom in Vuex*

### 7.6.4 Conclusie

Zowel Angular als Vue.js hebben een eigen *library* om het Reduxpatroon toe te passen, respectievelijk NgRx en Vuex.

De NgRx-*library* is uitgebreider dan de Vuex-*library* wat vooral in grote applicaties verschillende voordelen biedt. Hoewel NgRx snel leidt tot veel meer code en bestanden dan in Vuex voor dezelfde situatie, is de leesbaarheid beter. Daarnaast zorgen Effects er ook voor dat API-*calls* en andere

services afgescheiden blijven waardoor de Actions geen functionaliteit bevatten wat de testbaarheid dan weer vergroot.

Voor kleine applicaties, waar de State maar rond één model draait, kan Vuex wel aangenamer zijn doordat er minder code dient geschreven te worden. Maar als de applicatie zo klein is, is *state management* waarschijnlijk overbodig.

## 7.7 Algemene conclusie Angular versus Vue.js

| | **Angular** | **Vue.js** |
|---|---|---|
| **Release datum** | September 2016<br>Oktober 2010 (AngularJS) | Februari 2014 |
| **Grondlegger** | Google | Evan You |
| **Wanneer gebruiken** | Grote applicaties | Kleine SPA's |
| **Taal** | TypeScript | JavaScript met TypeScript-ondersteuning |
| **Leercurve** | Groot | Klein |
| **Populariteit** | Zeer populair bij ontwikkelaars | Grote populariteit op GitHub |
| **Bedrijven** | Google, Forbes | Alibaba, GitLab |
| **PWA-ondersteuning** | Uitstekend | Uitstekend |
| **Redux-implementatie** | NgRx | Vuex |

*Tabel 1 Vergelijkingsmatrix Angular versus Vue.js*

Bovenstaande vergelijkingsmatrix toont de voornaamste gelijkenissen en verschillen tussen Angular en Vue.js.

Zo is Vue.js ouder dan Angular, maar omdat de voorganger van Angular, AngularJS, ouder is dan Vue.js is het een meer gevestigde waarde dan Vue.js. Een andere belangrijke factor in de grotere populariteit van Angular is de ondersteuning door Google. Doordat een gespecialiseerd team met een groot budget achter Angular zit, kan het veel makkelijker doorbreken en zijn plaats op de markt opeisen dan Vue.js. Vue.js is opgericht door één individu en wordt onderhouden door de community. Vue.js heeft doorheen de jaren wel sterk aan populariteit gewonnen doordat grote bedrijven zoals Alibaba zich achter de technologie gezet hebben.

De leercurve van Angular wordt in het algemeen aanzien als groter dan die van Vue.js. Dit heeft veel te maken met het feit dat Angular TypeScript gebruikt en Vue.js gefocust is op typische JavaScript. Veel van de webontwikkelaars hebben een achtergrond in JavaScript wat de stap naar Vue.js kleiner maakt. De laatste jaren stijgt de interesse in TypeScript wel steeds meer waardoor Vue.js besloten heeft dit ook te ondersteunen.

De grote sterkte van Vue.js was dat het heeft kunnen leren van de fouten die de grote concurrenten AngularJS en React gemaakt hadden doorheen de jaren. Sindsdien is AngularJS vervangen door Angular en is dit voordeel gedeeltelijk weggevallen.

Progressive Web App-ondersteuning is in beide frameworks wel zeer gelijkend en even uitstekend en makkelijk implementeerbaar.

Ook voor Redux hebben ze beide framework-specifieke implementaties die inspelen op de sterktes van de frameworks.

# 8 Prototype

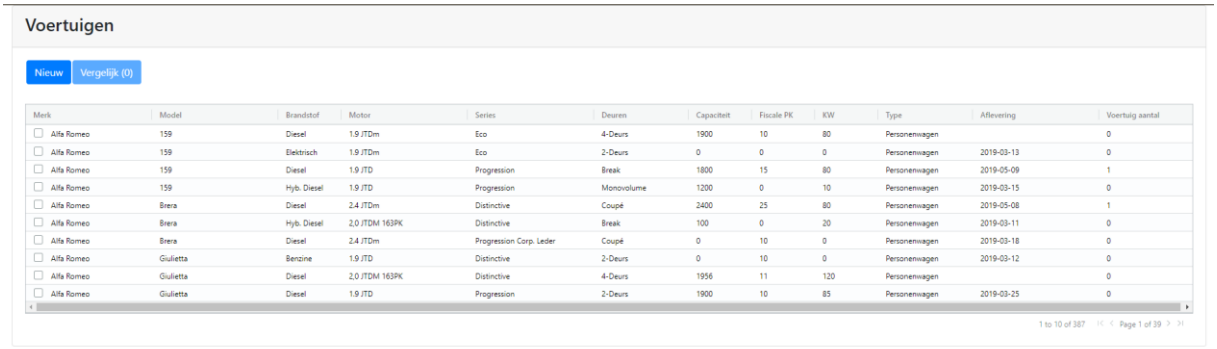
Om de bevindingen in de literatuurstudie te staven met de praktijk worden er prototypes gemaakt waarop verschillende metingen zullen gedaan worden.

## 8.1 Overzicht prototypes

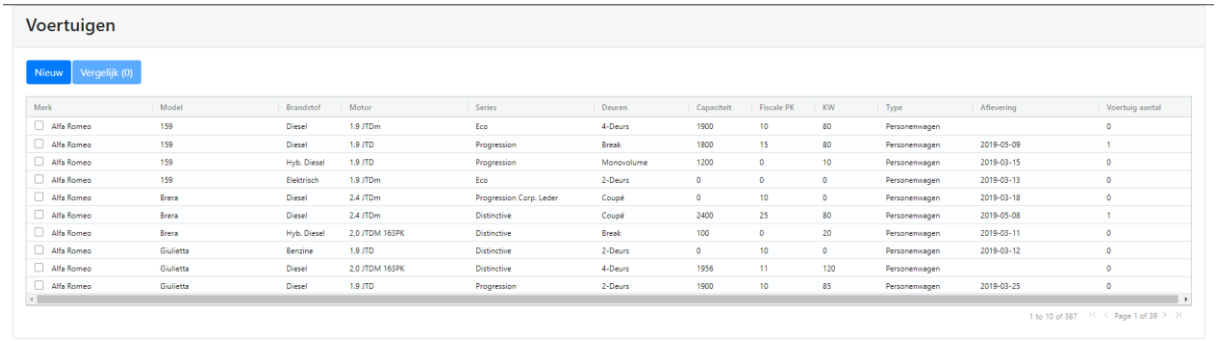
Om de vergelijking tussen Angular en Vue.js, met en zonder Redux, te kunnen maken, worden er vier prototypes gebouwd. De vier applicaties hebben een identieke functionaliteit.

### 8.1.1 Startscherm

Zo tonen ze alle vier een overzicht van alle voertuigen in een tabel. Voor deze tabel wordt zowel in Vue.js als Angular gebruikgemaakt van het *npm package* Ag-grid om de invloed van externe factoren te beperken. Onderstaande afbeeldingen tonen respectievelijk het eerste scherm in de Angular en de Vue.js applicatie. De applicaties met Redux zijn visueel identiek.



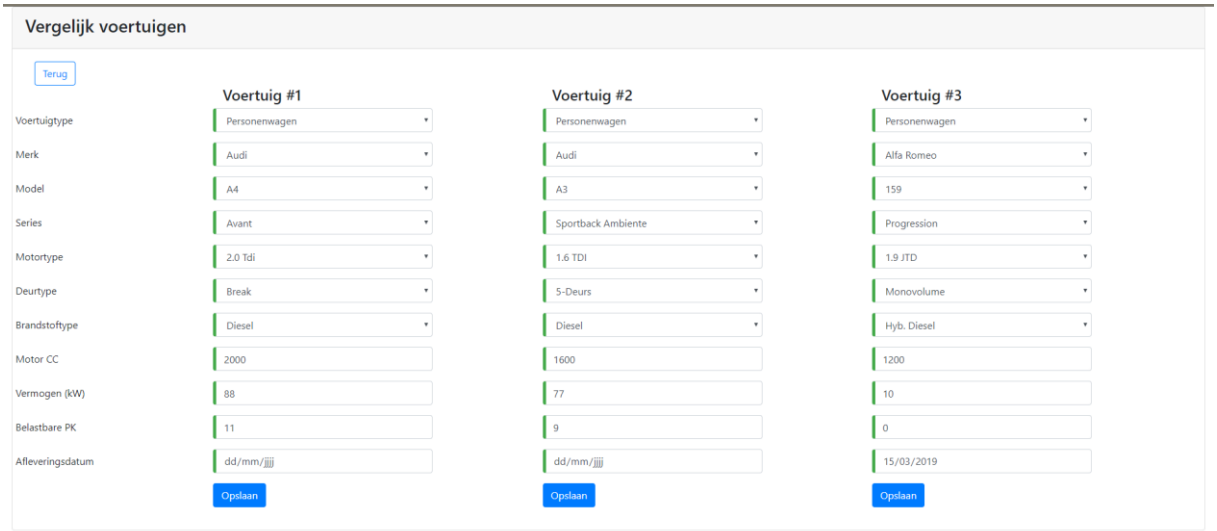

*Figuur 30 Startscherm Angular applicatie*



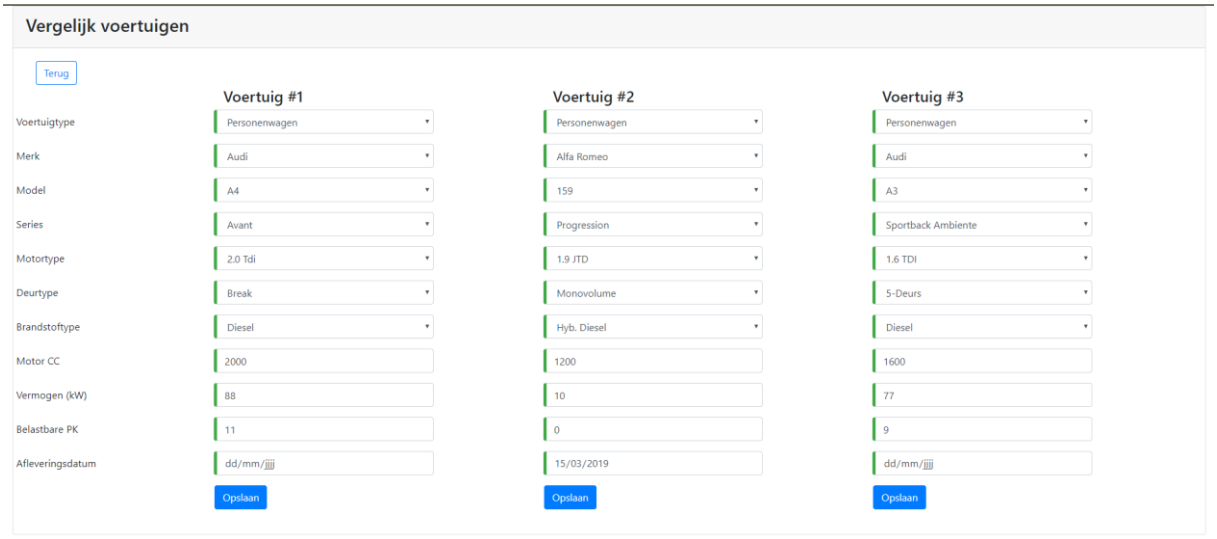

*Figuur 31 Startscherm Vue.js applicatie*

### 8.1.2 Vergelijkingsscherm

Het overzicht heeft ook de mogelijk verschillende voertuigen te selecteren om deze vervolgens te vergelijken. De knop is functioneel bij een selectie van één tot vier voertuigen. In dit overzicht is het mogelijk elk voertuig individueel aan te passen. Onderstaande afbeeldingen tonen het scherm in respectievelijk Angular en Vue.js.



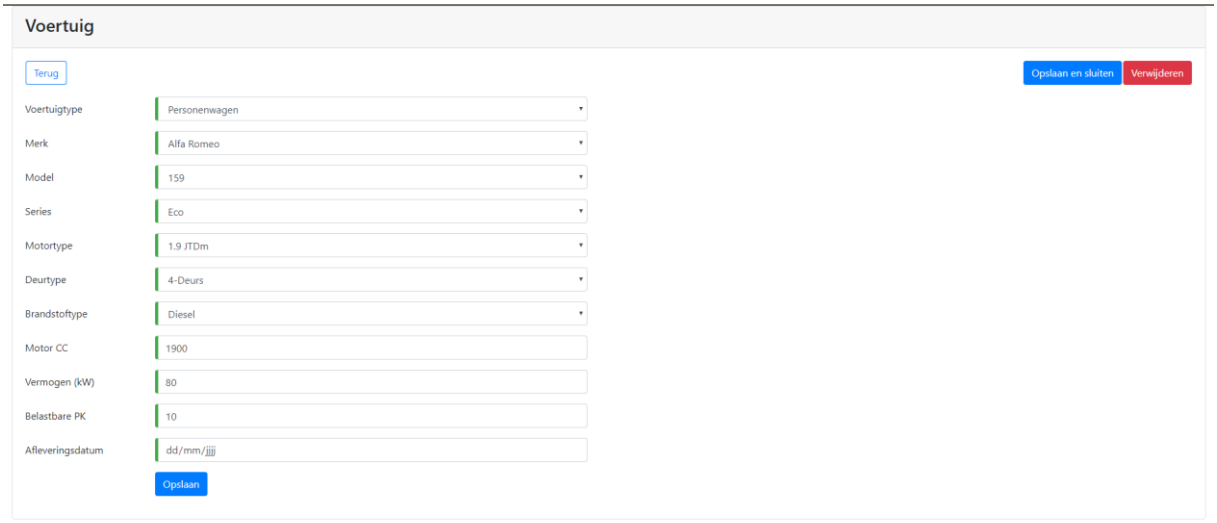

*Figuur 32 Vergelijkingsscherm Angular applicatie*



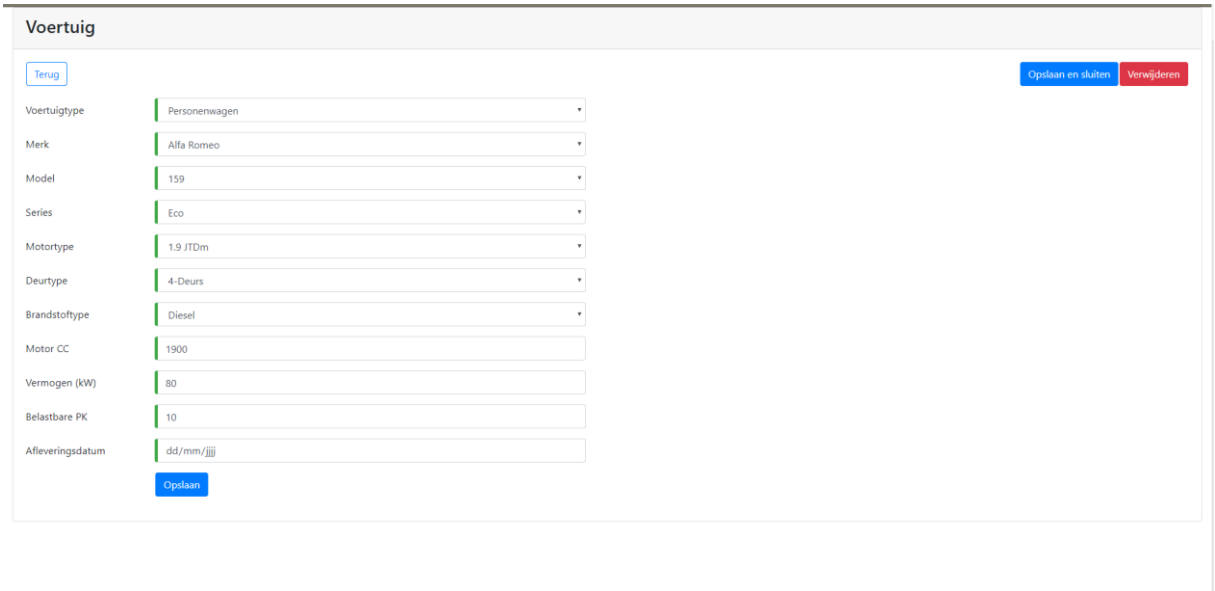

*Figuur 33 Vergelijkingsscherm Vue.js applicatie*

### 8.1.3 Detailscherm

Als laatste is er ook de mogelijkheid een nieuw voertuig aan te maken door op de “Nieuw”-knop te klikken of een bestaand voertuig aan te passen of te verwijderen door op de rij te klikken. In beide gevallen kom je op het detailscherm terecht zoals te zien op onderstaande afbeeldingen.




*Figuur 34 Detailscherm Angular applicatie*




*Figuur 35 Detailscherm Vue.js applicatie*

## 8.2 Werkwijze

Om conclusies te kunnen trekken uit de prestaties van de verschillende applicaties zullen er verschillende metingen uitgevoerd worden. Al deze metingen worden uitgevoerd op de applicaties onder gelijkaardige omstandigheden. Om de invloed van de laptop zelf en de browser te beperken zal enkel het nodige openstaan. Tijdens het uitvoeren van de metingen staan enkel volgende programma's open: een Excelsheet voor het noteren van de resultaten, de API in Visual Studio (deze draait lokaal) en twee Google Chrome tabbladen: één met de applicatie en één met de Dev Tools of Lighthouse om de metingen uit te voeren. Er wordt steeds een Incognitovenster gebruikt zonder extensies om de invloed van de browser te beperken.

## 8.3 Resultaten

Er worden metingen uitgevoerd over de opstarttijd, gebruiksprestaties en PWA-optimalisatie van beide frameworks.

### 8.3.1 Opstarttijd van de applicaties

Eerst en vooral wordt onderzocht of de opstarttijd van de applicaties wordt beïnvloed door het framework of hun implementaties van Redux.

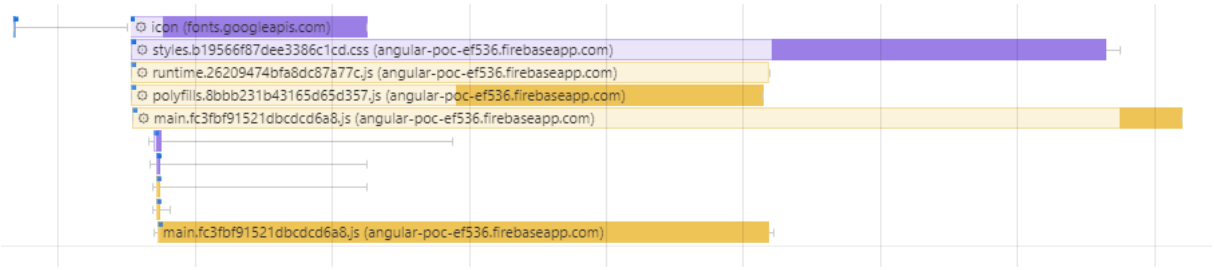
Onderstaande tabel toont de opstarttijd van de applicaties gemeten met de Chrome Dev Tools. Met de opstarttijd wordt bedoeld de tijd tussen het opvragen van de pagina en de "First Contentful Paint" of eerste inhoud die op de pagina verschijnt. Er werden telkens tien metingen na elkaar uitgevoerd. Onderstaande tabel toont de duur in milliseconden per meting met daaronder de gemiddelde duur over de tien metingen.

| | Angular | Angular Redux | Vue.js | Vue.js Redux |
|---|---|---|---|---|
| **1** | 3480 | 4980 | 3342 | 3905 |
| **2** | 3552 | 3335 | 3195 | 2686 |
| **3** | 3419 | 3499 | 3992 | 2920 |
| **4** | 3568 | 3594 | 4114 | 3085 |
| **5** | 3480 | 3424 | 3812 | 3937 |
| **6** | 3568 | 2325 | 4070 | 4132 |
| **7** | 3478 | 3594 | 4184 | 3803 |
| **8** | 3443 | 3503 | 3955 | 3969 |
| **9** | 3506 | 3373 | 3983 | 3944 |
| **10** | 3572 | 3124 | 3973 | 3820 |
| **GEMIDDELDE** | **3506,6** | **3475,1** | **3862** | **3620,1** |

*Tabel 2 Vergelijking in opstarttijd (in ms) per applicatie over tien metingen*

Over tien metingen kunnen we concluderen dat de opstarttijd zeer gelijkend is. Wat wel opvalt is dat de opstarttijd van beide Vue.js applicaties hoger is dan die van de Angular applicaties. Maar dit verschil is toch te klein om conclusies te trekken. Een belangrijke factor die de opstarttijd beïnvloedt, is de internetsnelheid. De metingen gebeuren op een netwerk waar verschillende gebruikers actief zijn. De snelheid kan daarom verschillen afhankelijk van de activiteit van de andere gebruikers.

Wat ook opvalt zijn de eerste meting bij de opstellingen met Redux. Deze zijn te verklaren door cachen. Bij de toepassingen zonder Redux was het cachen al gebeurt voor het uitvoeren van de eerste meting, daarom is daar geen verschil te merken. Door het cachen van bestanden moeten de framework-specifieke bestanden slechts eenmaal geladen worden. Dat deze bestanden framework-specifiek zijn, verklaart wel waarom de eerste meting zo aanzienlijk verschillend is tussen Angular en Vue.js. Angular steunt als meer volwaardig framework op meer en groter bestanden dan Vue.js. Onderstaande afbeeldingen tonen de bestanden die ingeladen worden van respectievelijk Angular en Vue.js.




*Figuur 36 Angular-specifieke bestanden*


*Figuur 37 Vue.js-specifieke bestanden*

Vooral de gele lijnen, de Javascript-bestanden, zijn hier belangrijk. Op deze bestanden steunen de frameworks. Bij Angular zijn dit een “runtime.js”-, “polyfills.js”- en “main.js”-bestand en bij Vue.js een “app.js”- en “chuck-vendors.js”-bestand.

### 8.3.2 Gebruiksprestaties

Vervolgens wordt de invloed van Redux gemeten op de prestaties van de applicaties “in gebruik”. Om dit te testen worden de Chrome Dev Tools gebruikt om de tijd te berekenen tussen het laden van alle componenten en het vullen van deze componenten met data.

Bij het meten wordt gekeken naar het inladen van de data in het overzicht, de verschillende invulvelden bij het updaten van een voertuig en de invulvelden bij het vergelijken van vier voertuigen. De tijd wordt telkens tweemaal gemeten om Redux aan het werk te zien. Een eerste keer als er nog data in de Reduxstore zit en een tweede keer als dit wel het geval is.

Onderstaande tabel toont het resultaat van de metingen in milliseconden.

| | Angular | Angular Redux | Vue.js | Vue.js Redux |
|---|---|---|---|---|
| **Inladen data overzicht eerste keer** | 370 | 400 | 660 | 770 |
| **Inladen data overzicht tweede keer** | 270 | 0 | 610 | 0 |
| **Inladen data detailscherm eerste keer** | 250 | 310 | 300 | 810 |
| **Inladen data detailscherm tweede keer** | 160 | 0 | 280 | 0 |
| **Inladen data 4 voertuigen eerste keer** | 510 | 860 | 260 | 360 |
| **Inladen data 4 voertuigen tweede keer** | 260 | 0 | 290 | 0 |

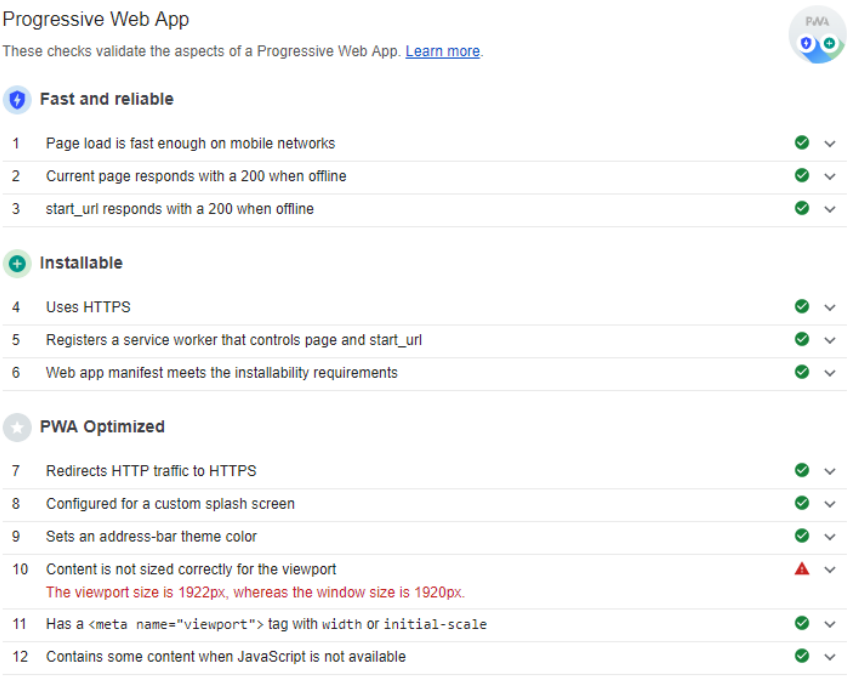
*Tabel 3 Laadtijden (in ms) tussen lege en met data gevulde componenten*

Wat meteen opvalt aan de metingen is de invloed van Redux in zowel de Angular- als Vue.js-applicatie. Wanneer de data is opgenomen in de *application state* die in de *store* opslagen is, en dus niet meer uit de API moet komen, onderscheiden de Chrome Dev Tools geen twee pagina’s meer en is de tijd tussen een data met en zonder data niet meer waarneembaar. De Chrome Dev Tools nemen namelijk telkens een screenshot wanneer een verandering van de pagina gedetecteerd wordt. In de andere gevallen werden er wel screenshots genomen met en zonder data. Tussen deze screenshots kon vervolgens de tijd berekend worden. In sommige gevallen zijn deze tijden aanzienlijk verschillend. Dit is te wijten aan verschillende factoren waaronder internetconnectie en kwaliteit van de code zoals het afhandelen van de binnengekregen data.

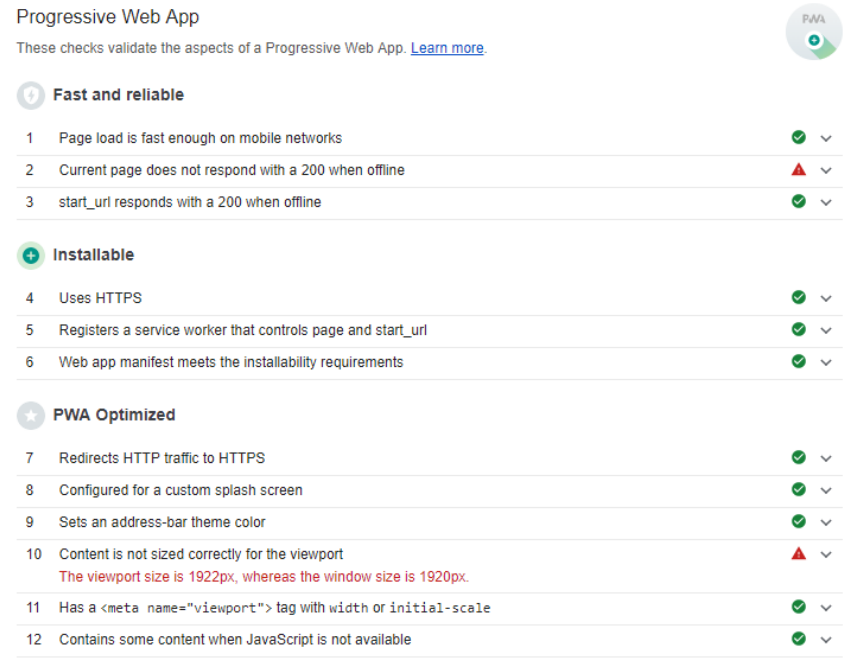
### 8.3.3 Progressive Web App-optimalisatie

Alle vier de prototypes zijn uitgerust om als Progressive Web App te werken. Hiervoor is enkel de configuratie doorlopen die Angular en Vue.js zelf voorstellen. De tool Lighthouse controleert een website op de verschillende aspecten van een PWA en toont aan welke voldaan zijn. Zo is het mogelijk te achterhalen welk framework het best uitgerust is voor het bouwen van een PWA.

Uit de test op onderstaande afbeelding blijkt dat Angular aan één vereiste meer voldoet dan Vue.js. Beide voldoen aan alle andere vereisten behalve het correct schalen. Dit probleem doet zich voor op alle geteste PWA’s op het internet en is vermoedelijk een probleem in de instellingen van de gebruikte browser.




*Figuur 38 PWA-optimalisatie Angular*




*Figuur 39 PWA-optimalisatie Vue.js*

Het enige andere aspect waaraan Vue.js niet voldoet volgens Lighthouse is dat de huidige pagina (het overzicht van de voertuigen of "/vehicles"-url) niet reageert bij de afwezigheid van een internetconnectie. Maar na een manuele test wordt deze meting toch ontkracht. Als er gesurft wordt naar de "/vehicles"-url van beide applicaties krijgt de gebruiker telkens het overzicht van de voertuigen. In conclusie zijn beide frameworks uitstekend uitgerust voor PWA-ondersteuning en is een onderscheid voor dit aspect tussen de twee verwaarloosbaar.

# Conclusie

Voor eFenKa wordt Angular gebruikt voor het maken van de frontend. Is dit de beste optie? Of is Vue.js een beter alternatief? Hoe vergelijken beide frameworks zich ten opzichte van elkaar en hoe goed zijn ze uitgerust voor het creëren van een Progressive Web App? En welke invloed heeft een implementatie van Redux op de prestaties?

Uit onderzoek is gebleken dat Angular en Vue.js op vele vlakken verschillen van elkaar. Zo is Vue.js ouder dan Angular, maar omdat de voorganger van Angular, AngularJS, ouder is dan Vue.js is het een meer gevestigde waarde dan Vue.js. Daarnaast geniet Angular ook betere ondersteuning dan Vue.js doordat het eigendom is van Google. Vue.js daarentegen is opgericht door één individu en wordt onderhouden door de community. Vue.js heeft doorheen de jaren wel sterk aan populariteit gewonnen doordat grote bedrijven zoals Alibaba zich achter de technologie gezet hebben.

De leercurve van Angular wordt in het algemeen aanzien als groter dan die van Vue.js. Dit heeft veel te maken met het feit dat Angular TypeScript gebruikt en Vue.js gefocust is op typische JavaScript. Veel webontwikkelaars hebben een achtergrond in JavaScript wat de stap naar Vue.js kleiner maakt. De laatste jaren stijgt de interesse in TypeScript wel steeds meer waardoor Vue.js besloten heeft dit ook te ondersteunen.

De grote sterkte van Vue.js was dat het heeft kunnen leren van de fouten die grote concurrenten AngularJS en React gemaakt hebben in hun beginjaren. AngularJS is ondertussen vervangen door Angular en dat heeft ervoor gezorgd dat dit voordeel gedeeltelijk weggevallen is.

Er zijn ook enkele gelijkenissen binnen deze frameworks. Zo is de Progressive Web App-ondersteuning in beide frameworks zeer gelijkend en even uitstekend en makkelijk implementeerbaar.

Uit de metingen is gebleken dat een applicatie in het ene framework niet beter is dan in het andere. Alle gemeten verschillen waren verwaarloosbaar. Het meest significante verschil was te zien in de opstarttijd of initiële laadtijd van de websites. Angular is een uitgebreider framework dan Vue.js wat betekent dat Angular meer framework-specifieke bestanden bevat die langer duren om te laden. In de praktijk is dit verschil niet merkbaar.

Ook de Reduxtoepassingen waren in beide frameworks verschillend. Angular heeft met NgRx een Reduxtoepassing die volledig gebaseerd is op het Reduxpatroon terwijl Vuex in Vue.js een mix van Redux en zijn voorganger Flux.

Alle resultaten in acht genomen is Angular de betere optie voor eFenKa. Omdat Angular een uitgebreider framework is met betere ondersteuning, is het een veiligere optie voor applicaties van grote omvang zoals eFenKa. Voor kleine applicaties is Vue.js het overwegen zeker waard.

Tot slot hebben de metingen naar de impact van Redux aangetoond dat het de prestaties wel degelijk ten goede kan komen, maar het blijft voornamelijk een implementatie om de code beter leesbaar en “loosely coupled” te houden. Het vermindert de communicatie tussen de componenten. Voor eFenKa is het gebruik van NgRx zeker aangewezen.

# Bibliografie


[1] M. Macneil, „User Authentication and Identity with Angular, Asp.Net Core and IdentityServer," FullStack Mark, 28 april 2019. [Online]. Available: https://fullstackmark.com/post/21/user-authentication-and-identity-with-angular-aspnet-core-and-identityserver. [Geopend 22 mei 2019].

[2] Angular, „What is Angular?," Angular, [Online]. Available: https://angular.io/docs. [Geopend 2 april 2019].

[3] Wikipedia, „Angular," Wikipedia, 9 januari 2019. [Online]. Available: https://nl.wikipedia.org/wiki/Angular. [Geopend 2 april 2019].

[4] Vue, „Introduction," Vue, [Online]. Available: https://vuejs.org/v2/guide/. [Geopend 3 april 2019].

[5] Wikipedia, „Progressive web applications," Wikipedia, 2 april 2019. [Online]. Available: https://en.wikipedia.org/wiki/Progressive_web_applications. [Geopend 3 april 2019].

[6] A. Russell, „Progressive Web Apps: Escaping Tabs Without Losing Our Soul," Infrequently, 15 juni 2015. [Online]. Available: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/. [Geopend 4 april 2019].

[7] Redux, „Getting Started with Redux," Redux, 9 december 2018. [Online]. Available: https://redux.js.org/introduction/getting-started. [Geopend 4 april 2019].

[8] S. Kapoor, „What is Redux and who uses it?," Quora, 16 november 2018. [Online]. Available: https://www.quora.com/What-is-Redux-and-who-uses-it. [Geopend 4 april 2019].

[9] S. Lahoti, „Top frameworks for building your Progressive Web Apps (PWA)," Packt Hub, 15 mei 2018. [Online]. Available: https://hub.packtpub.com/top-frameworks-for-building-your-progressive-web-application-pwa/. [Geopend 21 april 2019].

[10] R. Witkowski, „Developing a PWA Using Angular 7," DZone, 26 november 2018. [Online]. Available: https://dzone.com/articles/developing-pwa-using-angular-7. [Geopend 25 april 2019].

[11] M. Gaunt en P. Kinlan, „The Web App Manifest," Google, 12 maart 2019. [Online]. Available: https://developers.google.com/web/fundamentals/web-app-manifest/. [Geopend 25 april 2019].

[12] NGRX, „NGRX Reactive State for Angular," NGRX, [Online]. Available: https://ngrx.io/. [Geopend 22 april 2019].

[13] Vue, „What is Vuex?," Vue, [Online]. Available: https://vuex.vuejs.org/. [Geopend 22 april 2019].

[14] Stack Overflow, „Developer Survey Results 2019: Most Popular Technologies," Stack Overflow, 2019. [Online]. Available: https://insights.stackoverflow.com/survey/2019#most-popular-technologies. [Geopend 17 mei 2019].

[15] S. Daityari, „Angular vs Vue vs React," Codeinwp, 15 april 2019. [Online]. Available: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/. [Geopend 25 april 2019].

[16] SPEC INDIA, „React vs Angular vs Vue.js: A Complete Comparison Guide," Medium, 16 augustus 2018. [Online]. Available: https://medium.com/front-end-weekly/react-vs-angular-vs-vue-js-a-complete-comparison-guide-d16faa185d61. [Geopend 25 april 2019].

[17] T. Żdanuk, „How to use PWA plugin in Vue CLI 3.0," Naturaily, 6 augustus 2018. [Online]. Available: https://naturaily.com/blog/pwa-vue-cli-3. [Geopend 25 april 2019].

[18] S. Eschweiler, „Learn Redux – Introduction To State Management With React," Coding The Smart Way, 15 mei 2017. [Online]. Available: https://codingthesmartway.com/learn-redux-introduction-to-state-management-with-react/. [Geopend 26 april 2019].

[19] Redux, „Actions," Redux, [Online]. Available: https://redux.js.org/basics/actions. [Geopend 29 mei 2019].

[20] Redux, „Reducers," Redux, [Online]. Available: https://redux.js.org/basics/reducers. [Geopend 29 mei 2019].

[21] Redux, „Store," Redux, [Online]. Available: https://redux.js.org/basics/store. [Geopend 29 mei 2019].

[22] S. G. d. Rosa, „Angular: NGRX a clean and clear Introduction," Medium, 7 oktober 2018. [Online]. Available: https://medium.com/frontend-fun/angular-ngrx-a-clean-and-clear-introduction-4ed61c89c1fc. [Geopend 25 april 2019].

[23] Vue, „Getting Started," Vue, [Online]. Available: https://vuex.vuejs.org/guide/. [Geopend 26 april 2019].