

**[Fleet management]**

|  |  |  |
| --- | --- | --- |
| REVISION | DATE | AUTHOR |
|  |  |  |
| {complete table on each revision} {clear table for new document} | | |

# Functioneel

## Doel van de applicatie

Met de fleet management app (FMA) moet een bedrijf in staat zijn om zijn wagenpark te gaan beheren. Dit wil zeggen dat de applicatie aan een volgende functionele vereisten moet voldoen:

* Beheren van voertuigen
* Beheren van tankkaarten
* Beheren van chauffeurs
* Inplannen van onderhoud en herstelling

## Wie gebruikt de applicatie

De applicatie zelf wordt gebruikt door een aantal rollen binnen de organisatie.

De beheerder rol kan alles beheren en de nodige instellingen aanpassen, vanuit deze rol kunnen er ook gebruikers worden aangemaakt en aan rollen gekoppeld worden.

De administratieve rol kan de wagens, tankkaarten, chauffeurs beheren en kan de nodige aanvragen doen en goedkeuren.

De werknemer rol kan zijn eigen wagen bekijken en de nodige aanvragen doen.

## Welke data moet worden bijgehouden

### Chauffeurs

Bij een chauffeur moet volgende data worden bijgehouden

* Naam
* Voornaam
* Adres (straat + nummer, Stad, postcode)
* Geboortedatum
* Rijksregisternummer (met validatie <https://nl.wikipedia.org/wiki/Rijksregisternummer>)
* Type rijbewijs

Een chauffeur mag slechts 1 maal voorkomen in het systeem. Er moet dus een controle worden uitgevoerd bij het aanmaken van een nieuwe chauffeur of deze al dan niet in het systeem aanwezig is. Hier moet ook een mogelijkheid zijn om een chauffeur op niet actief te zetten zodat mensen die uit dienst zijn niet langer zichtbaar zijn in de applicatie.

### Tankkaarten

Bij een tankkaart moet volgende data bijgehouden worden

* Kaartnummer
* Geldigheidsdatum
* Pincode
* Authenticatie type
  + PIN/ PIN + Kilometerstand
* Tankaart opties
  + Brandstof type
  + Extra services (Fuel, Shop, Tires, Carwash, ….)

Een tankkaart kan ook slechts eenmaal in het systeem voorkomen. Ook hier moet er dus een controle op komen. Tankkaarten moeten kunnen geblokkeerd worden (bij verlies of diefstal)

De tankkaart is altijd gekoppeld aan een chauffeur en kan mogelijks aan meerdere chauffeurs gekoppeld zijn. Er mag ten allen tijde slechts 1 koppeling als actief gezien worden.

### Voertuigen

Bij een voertuig moeten volgende gegevens worden bijgehouden

* Chassisnummer
* Nummerplaat
  + Hou er rekening mee dat een voertuig meerdere nummerplaten kan hebben (slechts 1 actieve) en dat we hiervan een historiek moeten kunnen opvragen
* Brandstoftype (Benzine, Diesel, hybride Benzine, hybride Diesel, elektrisch)
* Type wagen (personen, lichte vracht, … )
* Kilometerstand
  + Ook hier moet een historiek kunnen opgevraagd worden

#### Onderhoud

Per voertuig moeten alle onderhoudsbeurten bijgehouden worden. Deze worden manueel ingegeven en bevatten volgende gegevens

* Datum
* Prijs
* Garage waar het onderhoud is uitgevoerd
* Factuur (document opladen)

#### Herstellingen

Als er schade is aan een voertuig moet er hiervoor tracking gebeuren in het systeem. Bij schadegevallen moeten volgende gegevens worden bijgehouden

* Datum
* Schade omschrijving
* Verzekeringsmaatschappij
  + Uit een lijst te kiezen van
* Referentienummer v/d verzekeringsmaatschappij
* Meerdere foto’s
* Meerdere documenten (opladen)

### Aanvragen

De werknemer kan een aantal acties via het systeem aanvragen. Om deze allemaal centraal te beheren houden we volgende data bij

* Datum van aanvraag
* Type aanvraag (tankkaart, herstelling, onderhoud, …)
* Gewenst data van inplannen (2 datums)
* Status van de aanvraag
* Gelinkte voertuig (kan ook niet gekend zijn)
* Chauffeur die de aanvraag doet

# Technisch

## Algemene architectuur



Als basis voor deze applicatie gebruiken we .net 5 en MSSQL server. De applicatie zelf wordt opgebouwd uit een 4 lagen systeem. De Data Access Laag maakt gebruik van 2 ORMs EF Core en Dapper. EF Core wordt toegepast voor alle CRUD operations en Dapper wordt gebruikt voor het opstellen van de meer complexe Queries.

Alle Business logica wordt centraal beheerd in de Business laag. Deze laag dient als centraal aanspreekpunt voor de verschillende endpoints die data naar de buitenwereld beschikbaar stellen.

Als endpoints worden er een readonly api, een write api en een gRPC Service voorzien.

De frontend bestaat uit 2 delen. Het eerste deel is een Web front die de gebruikers in staat stelt om hun eigen wagen en tankkaarten op te vragen, maar ook de mogelijkheid bied om aanvragen tot herstellingen en onderhoud

## EF Core DAL

Deze datalaag bevat alle code om de nodige CRUD operations te gaan uitvoeren tegenover de databank. Hier moet er gebruik gemaakt worden van een Generic Repository zodat er geen herhaling van code plaatvind. Om de koppeling met de database te gaan leggen maken we gebruik van de Code first manier van werken.

## Dapper DAL

Deze datalaag bevat de meer complexere queries en maak gebruik van de Dapper ORM

## Business Logic

Dit is het centrale punt van de applicatie. Hier wordt alle logica geschreven die nodig is om de applicatie te doen werken. Deze applicatie laag moet ook volledig afgetest worden door middel van unit tests.

## End-points

Alle end-points kunnen aanzien worden als doorgeefluiken van de Business logic. In de end-points zelf zit geen business logica. Hier zit wel error handling in

### Read API

Dit is het lees end-point van de applicatie. Hier wordt alle data aangeboden die nodig is voor de user front-end.

### Write API

Dit is het schrijf en bewerken end-point van de user front-end.

### gRPC Service

Deze end-point bevat alle functies die nodig zijn om de administratie front-end te faciliteren. Dit zijn zowel lees als schrijf bewerkingen.

# Referenties

* Identity server 4: <https://identityserver4.readthedocs.io/en/latest/>
* Entity Framework Core: <https://docs.microsoft.com/en-us/ef/core/>
* Dot net Core: <https://docs.microsoft.com/en-us/dotnet/core/>
* Design patterns: <https://www.dofactory.com/net/design-patterns>
* Repository Pattern: <https://deviq.com/repository-pattern/>
* MediatR: <https://github.com/jbogard/MediatR/wiki>
* Dapper : <https://dapper-tutorial.net/>