Architectuur document

PTS-ESD

Remi Arts, Bas Rheiter, Tim Lodder, Ferdie Jonkers, Jelle v.d. Heijden

Inhoud

[Inleiding 2](#_Toc381347633)

[Context 2](#_Toc381347634)

[Applicatie 2](#_Toc381347635)

[Doel van dit document 2](#_Toc381347636)

[Domeinmodel 3](#_Toc381347637)

[Klassendiagram 3](#_Toc381347638)

[Persistentie 8](#_Toc381347639)

[Eigenschappen van objecten m.b.t. persistentie Verplaatsingsysteem 8](#_Toc381347640)

[Eigenschappen van objecten m.b.t. persistentie Rekeningadministratiesysteem 8](#_Toc381347641)

[Eigenschappen van objecten m.b.t. persistentie Rekeningrijderswebsitesysteem 8](#_Toc381347642)

[Eigenschappen van objecten m.b.t. persistentie Autosysteem 8](#_Toc381347643)

[Eigenschappen van objecten m.b.t. persistentie Filesysteem 9](#_Toc381347644)

[Eigenschappen van objecten m.b.t. persistentie Politiesysteem 9](#_Toc381347645)

[Data opslag 9](#_Toc381347646)

[Communicatie 10](#_Toc381347647)

[Attributen m.b.t. communicatie 10](#_Toc381347648)

[Manier en momenten van communicatie 11](#_Toc381347649)

[Realisatie niet-functionele eisen 12](#_Toc381347650)

[Betrouwbaarheid 12](#_Toc381347651)

[Performance 12](#_Toc381347652)

[Beveiliging 12](#_Toc381347653)

[Schaalbaarheid 12](#_Toc381347654)

[Component 13](#_Toc381347655)

[Component diagram 13](#_Toc381347656)

[Koppeling tussen componenten 13](#_Toc381347657)

[Services per component 13](#_Toc381347658)

[Allocatie van objecten 13](#_Toc381347659)

[Remote objecten 13](#_Toc381347660)

[Deployment 14](#_Toc381347661)

[Deployment diagram 14](#_Toc381347662)

# Inleiding

## Context

De Nederlandse probeert een oplossing te vinden voor het fileprobleem. Het idee van de overheid is om de vaste wegenbelasting af te schaffen en een kilometerheffing in te voeren.

Op die manier zal de weggebruiker betalen per gereden kilometer. Het kilometer tarief zal o.a. afhangen van de milieuvriendelijkheid van de auto.

Iedere auto zal een gps systeem krijgen ingebouwd waarmee het aantal gereden kilometers wordt bijgehouden. Hiervoor zijn een aantal applicaties nodig die hieronder worden beschreven.

## Applicatie

De applicatie bestaat uit 6 systemen(verplaatsingssysteem, filesysteem, politiesysteem, rekeningadministratie, autosysteem, rekeningrijderswebsite) die samen zijn geïntegreerd.

## Doel van dit document

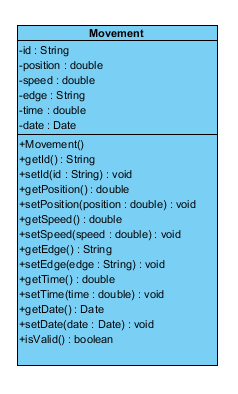
In dit document gaan we de architectuur(technische ontwerpen van software) voor het project verantwoorden. Daarna gaan we alle ontworpen diagrammen en interfaces voor het project beschrijven met betrekking tot de mogelijkheid om onderling (de verschillende onderdelen) te communiceren. Verder gaan we zeven aspecten van de architectuur beschrijven:

* Domeinmodel
* Persistentie
* Communicatie
* Realisatie van niet-functionele eisen
* Componenten
* Deployment

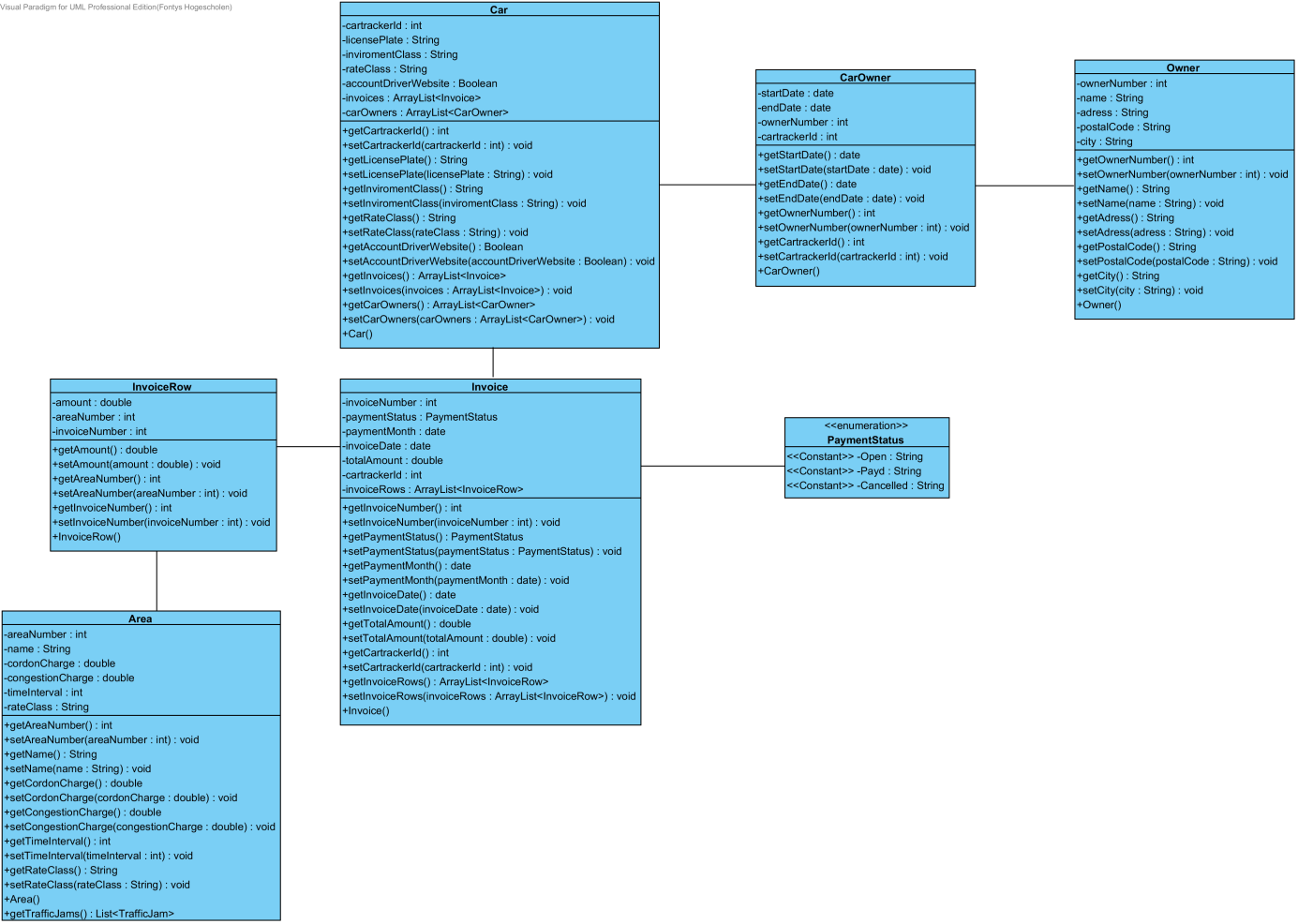
# Domeinmodel

In dit hoofdstuk gaan we toelichting geven op onze gemaakte keuzes bij het klassendiagram. Het domeinmodel dat in dit hoofdstuk wordt getoond, zullen we ook in onze applicatie gaan toepassen.

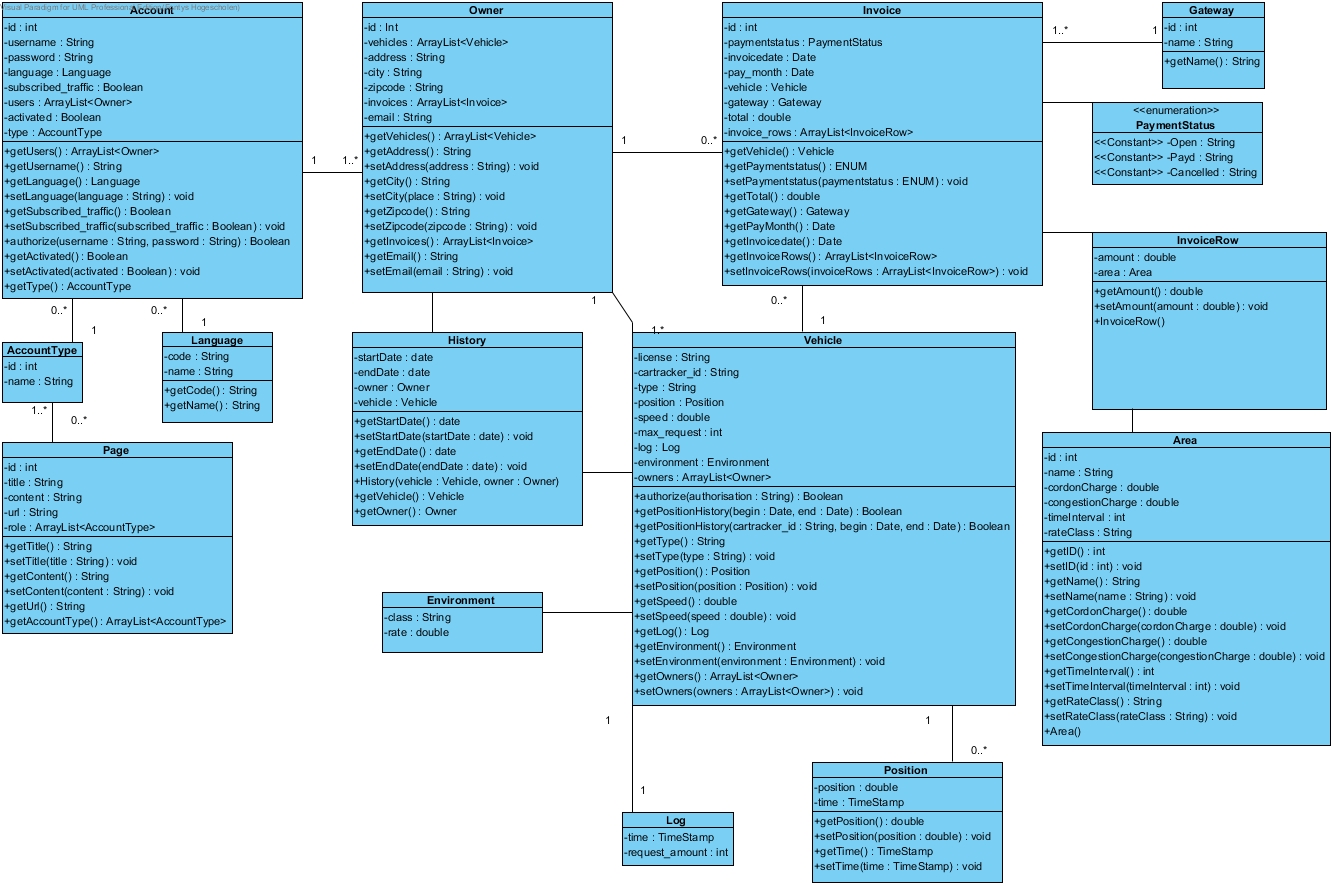
## Klassendiagram



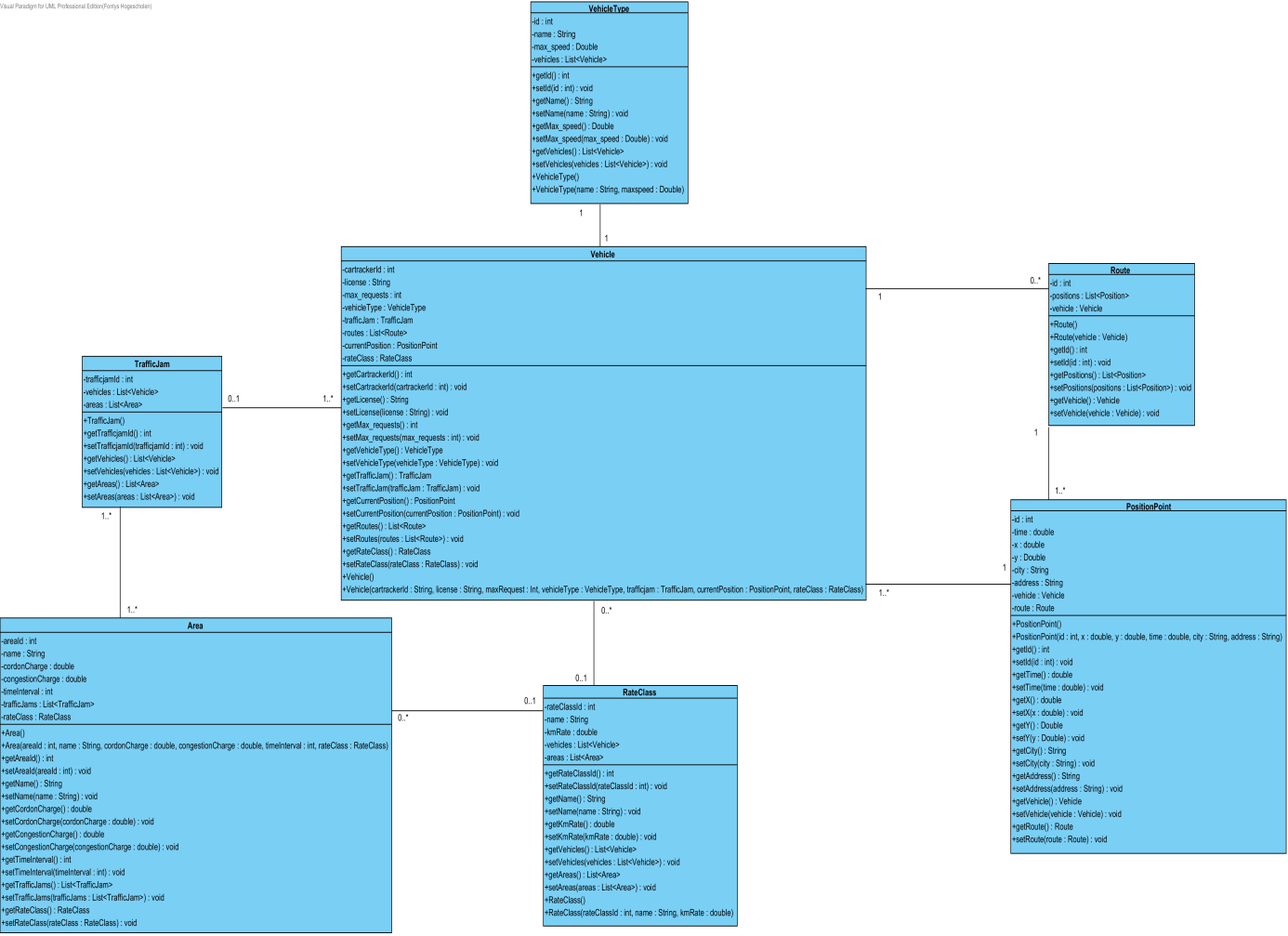
**Figuur 1: Klassendiagram Verplaatsingsysteem.**

****

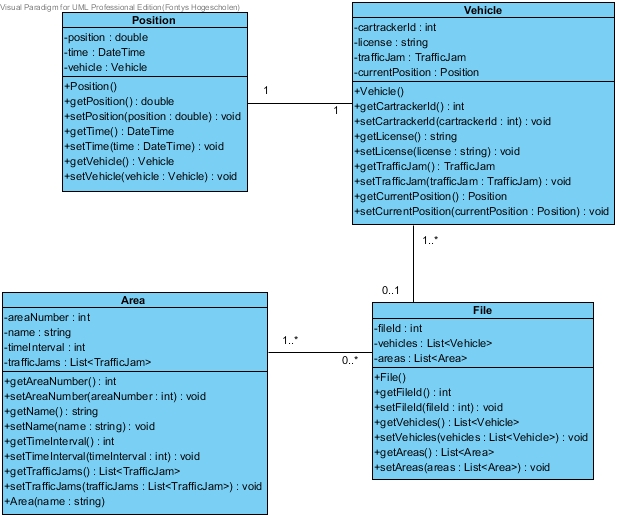
**Figuur 2: Klassendiagram Rekeningadministratiesysteem.**

****

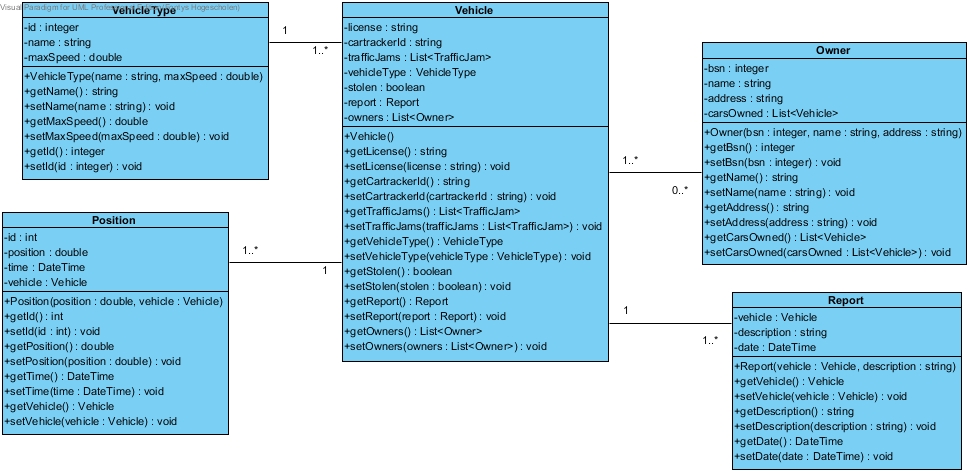
**Figuur 3: Klassendiagram Rekeningrijderswebsitesysteem.**

****

**Figuur 4: Klassendiagram Autosysteem.**

****

**Figuur 5: Klassendiagram Filesysteem.**

****

**Figuur 6: Klassendiagram Politiesysteem.**

# Persistentie

In dit hoofdstuk wordt de persistentie van de objecten beschreven.

## Eigenschappen van objecten m.b.t. persistentie Verplaatsingsysteem

* Movement: **id,** position, speed, edge, time, date.

## Eigenschappen van objecten m.b.t. persistentie Rekeningadministratiesysteem

* Vehicle: license, **cartracker\_id**, enviroment\_class, rate\_class, account\_driver\_website, owner\_id.
* VehicleOwner: **vehicle\_owner\_id**, start\_date, end\_date, owner\_id, cartracker\_id.
* Area: **area\_id**, name, cordon\_charge,congestion\_charge, time\_interval, rate\_class,
* Invoice: **invoice\_id**, payment\_status, payment\_month, date, total\_amount, cartracker\_id.
* InvoiceRow: **invoice\_row\_id**, amount, invoice\_id, area\_id.
* Owner: **owner\_id**, name, address, postal\_code, city.

## Eigenschappen van objecten m.b.t. persistentie Rekeningrijderswebsitesysteem

* Vehicle:license, **cartracker\_id**, max\_request, enviroment\_class, rate\_class, owner\_id.
* VehicleOwner: **vehicle\_owner\_id**, start\_date, end\_date, owner\_id, cartracker\_id.
* VehicleType: **vehicle\_type\_id**, name, max\_speed.
* Language: **language\_id**, name.
* Invoice: **invoice\_id**, payment\_status, payment\_month, date, total\_amount, cartracker\_id, gateway\_id.
* InvoiceRow: **invoice\_row\_id**, amount, invoice\_id.
* Gateway: **gateway\_id**, name.
* Owner: **owner\_id**, name, address, postal\_code, city, email, account\_id, language\_id.
* Account: **account\_id**, username, password, subscribed\_traffic, activated, account\_type\_id.
* AccountType: **account\_type\_id**, name.
* Page: **page\_id**, title, content, account\_type\_id.

## Eigenschappen van objecten m.b.t. persistentie Autosysteem

* Vehicle: **cartracker\_id**, license, max\_request, vehicle\_type\_id, traffic\_jam\_id.
* Position: **position\_id**, position, time, route\_id, cartracker\_id.
* Area: **area\_id**, name, cordon\_charge, congestion\_charge, time\_interval, rate\_class.
* TrafficJam: **traffic\_jam\_id**.
* Route: **route\_id**, cartracker\_id.
* VehicleType: **vehicle\_type\_id**, name, max\_speed.
* TrafficJamArea: traffic\_jam\_id, area\_id, **traffic\_jam\_area\_id**.

## Eigenschappen van objecten m.b.t. persistentie Filesysteem

* Vehicle: lisence, **cartracker\_id**, traffic\_jam\_id
* Position: **position\_id**, position, time, license.
* Area: **area\_id**, name, time\_interval.
* TrafficJam: **traffic\_jam\_id**.
* TrafficJamArea: **traffic\_jam\_area\_id**, traffic\_jam\_id, area\_id.

## Eigenschappen van objecten m.b.t. persistentie Politiesysteem

* Vehicle: **cartracker\_id**, license, vehicle\_type\_id, report\_id.
* Position: position, time, cartracker\_id, **position\_id**.
* VehicleType: **vehicle\_type\_id**, name, max\_speed.
* Owner: **owner\_id**, name, address, phone\_number.
* Report: **report\_id**, date, description, cartracker\_id.
* VehicleOwner: cartracker\_id, owner\_id, **vehicle\_owner\_id**.

## Data opslag

Alle gegevens van de bovenstaande gespecificeerde klassen worden in de database opgeslagen. Alle data moet ieder moment beschikbaar blijven zal voor ieder systeem op hun eigen lokale database opgeslagen worden.

# Communicatie

In dit hoofdstuk wordt de communicatie van en tussen objecten beschreven. Hierbij wordt aangegeven welke attributen van welke objecten moeten worden doorgegeven. Hoe de communicatie wordt gerealiseerd en wanneer de communicatie plaatsvindt. Hoe en wanneer er communicatie plaats vind kan verschillen per attribuut.

## Attributen m.b.t. communicatie

Verplaatsingensysteem

* Movement
* Id
* Position
* Speed
* Edge
* Time
* Date

Rekeningadministratie

* Invoice
  + invoiceNr
  + paymentStatus
  + paymentMonth
  + invoiceDate
  + totalAmount
  + cartracker\_id
  + invoiceRows
* Owner
  + name
  + address
  + zipcode
  + city

Autosysteem

* Vehicle
  + cartracker\_id
  + position

Filesysteem

* TrafficJam
  + trafficjam\_id
  + vehicles
  + areas

## Manier en momenten van communicatie

Er wordt op verschillende manieren gecommuniceerd tussen de applicaties. Hieronder staat deze communicatie beschreven.

#### Verplaatsingensysteem 🡪 Rekening administratie

Het verplaatsingensysteem stuurt informatie over de verplaatsingen van auto’s naar de rekeningenadministratie. De rekening administratie gebruikt deze informatie voor het berekenen van het aantal gereden kilometers. De communicatie gebeurt via een RESTfull webservice.

#### Rekening administratie 🡪 Rekeningrijders website

De rekening administratie genereert facturen op basis van informatie uit o.a. het verplaatsingensysteem. Deze facturen worden doorgestuurd naar de rekeningrijders website. Op het moment dat een gebruiker zijn facturen wil opvragen, haalt de rekeningrijderswebsite deze informatie uit de rekening administratie. Dit maakt dus gebruik van het pull principe en wordt verstuurd m.b.v. JMS.

#### Autosysteem 🡪 Verplaatsingensysteem

Het autosysteem genereert auto’s met een bepaalde route. Deze worden doorgestuurd naar het verplaatsingensysteem via het push principe. De informatie wordt verstuurd SOAP Web services.

#### Filesysteem 🡪 Autosysteem

Het autosysteem heeft de mogelijkheid om file informatie op te vragen uit het filesysteem. Dit wordt via JMS verstuurd met het pull principe.

#### Verplaatsingensysteem 🡪 Politiesysteem

Het politiesysteem bevat functies om een auto realtime te kunnen volgen en om de verplaatsingsgeschiedenis van een auto op te vragen. Deze informatie wordt uit het verplaatsingensysteem gehaald via pull. Het wordt verstuurd via een RESTfull webservice.

# Realisatie niet-functionele eisen

In dit hoofdstuk wordt de realisatie van (overige) niet‐functionele eisen besproken. De volgende onderwerpen dienen ten minste te worden besproken: betrouwbaarheid, performance, beveiliging, schaalbaarheid. Eventueel kunnen nog extra niet‐functionele eisen worden besproken.

## Betrouwbaarheid

De applicaties moeten betrouwbaar zijn en ten alle tijden goed functioneren. Vooral bij het aanmaken van de facturen moet de hoogste betrouwbaarheid gegarandeerd worden. De bedragen moeten natuurlijk ten alle tijden kloppen.

## Performance

Gebruikers mogen niet langer moeten wachten op acties dan 10 seconden. Om de performance van iedere applicatie zo goed mogelijk te krijgen, word er per systeem gekeken wat het meest geschikt is. Door een gepersonaliseerd systeem per applicatie wordt een goede performance gegarandeerd.

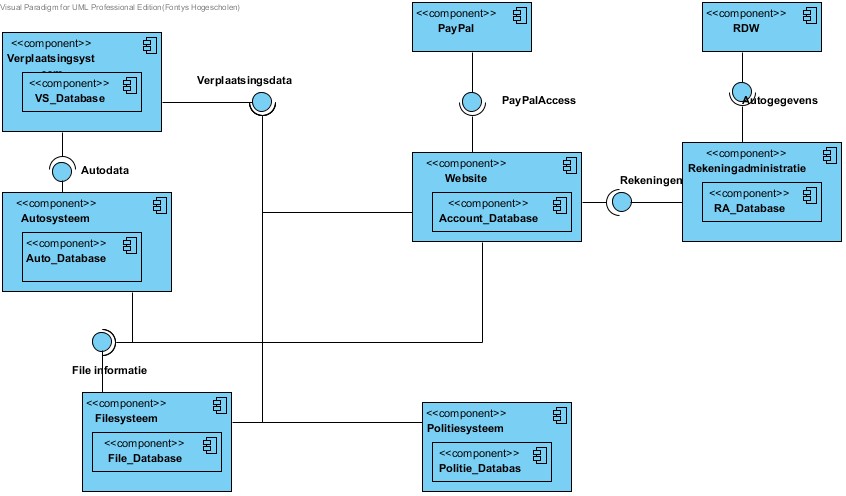
## Beveiliging

De applicaties bevatten gevoelige persoonsgegevens en daarom is een goede beveiliging vereist. Alle persoonsgegevens moeten gecodeerd opgeslagen worden.

# Component

Het componentdiagram geeft de relaties en de afhankelijkheden van de componenten van de systemen schematisch weer met behulp van interfaces. Hierbij wordt duidelijk welk component een bepaalde interface nodig heeft en door welk component die interfaces geleverd wordt.

## Component diagram



**Figuur 7: Component diagram.**

## Koppeling tussen componenten

Het **autosysteem** simuleert de auto’s en de routes die zij ‘volgen’, informatie die cruciaal is voor het verplaatsingssysteem. Middels de interface, welke wordt aangeboden door het autosysteem, wordt deze informatie opgehaald door het verplaatsingssysteem.

Het **verplaatsingssysteem** is het centrale component van het overkoepelende systeem, omdat het verschillende systemen de interface biedt die het mogelijk maakt om de verplaatsingen van auto’s op te halen. Deze informatie wordt – middels de interface – opgevraagd door de rekeningrijderswebsite, het filesysteem en het politiesysteem.

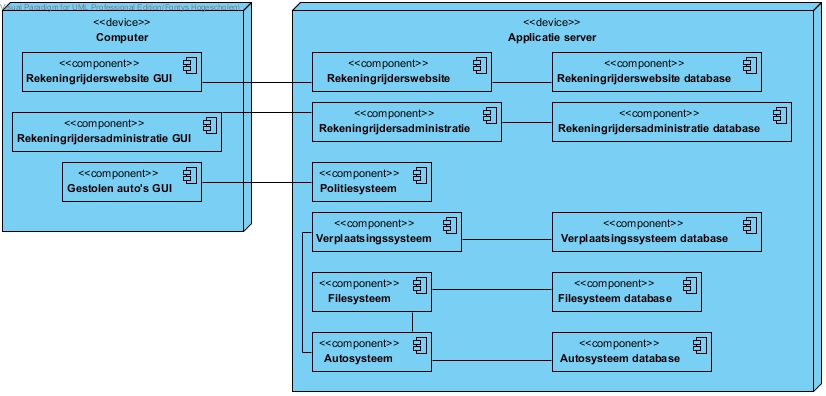
Het **filesysteem** biedt de interface aan die wordt gebruikt door het autosysteem en de rekeningrijderswebsite. Hierbij verschaft het filesysteem informatie betreffende de files.

Het **rekeningadministratiesysteem** zorgt ervoor dat er rekeningen worden opgesteld voor de rekeningrijders. Om ervoor te zorgen dat de rekeningrijders deze rekeningen ook in kunnen zien, biedt het rekeningadministratiesysteem deze gegevens aan middels een interface. De website haalt deze gegevens hiermee op. Voor de verwerking van autogegevens heeft het rekeningadministratiesysteem ook een koppeling nodig met het RDW, welke ook weer middels een interface gerealiseerd wordt, waarbij het RDW de aanbiedende partij is.

Aangezien de **rekeningrijderswebsite** een verbinding nodig heeft met de services van PayPal, zal hier ook een interface worden gebruikt. PayPal is hierbij de aanbiedende partij.

# Deployment

## Deployment diagram

****

**Figuur 8: Deployment diagram.**