OSM-Modeling Casus

Tim Klaassen(568583) & Jochem Grootherder(598765)

Hogeschool van Arnhem en Nijmegen

OSM-M Richard Holleman

01-11-2018

Versie 3.2

Inhoud

[1. Inleiding 2](#_Toc528849943)

[2. Use case diagram 3](#_Toc528849944)

[2.1 Brief description 3](#_Toc528849945)

[3. Niet functionele eisen 5](#_Toc528849946)

[4. Deployment Diagram 6](#_Toc528849947)

[5. Component diagram 7](#_Toc528849948)

[6. Fully dressed use cases 8](#_Toc528849949)

[Use Case: Reserveren van een auto 8](#_Toc528849950)

[Use Case: Ophalen van een auto 9](#_Toc528849951)

[Use Case: Terugbrengen van RedCars auto 10](#_Toc528849952)

[Use Case: Betalen van de huur 11](#_Toc528849953)

[7. Domeinmodel 12](#_Toc528849954)

[8. Sequence diagrammen 14](#_Toc528849955)

[9. Design class diagram 21](#_Toc528849956)

# Inleiding

In dit document staan alle gemaakt opdrachten voor de casus van OSM-M.

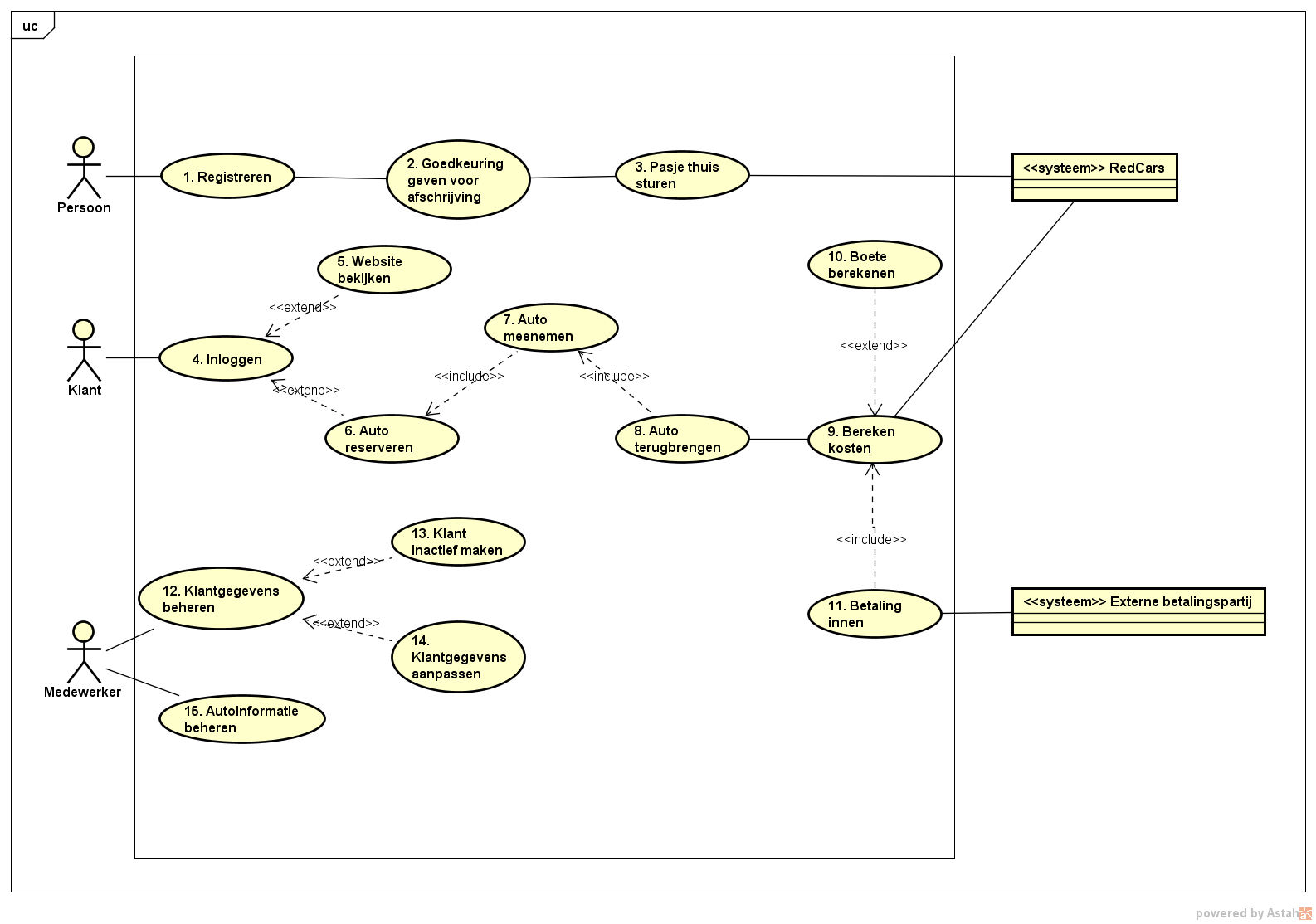
Bij alle gemaakte opdrachten staat een beknopte uitleg over wat het doel van de opdracht is en in welke vorm deze gemaakt is.

Na het lezen van dit document is de lezer bekend met alle gemaakte opdrachten en het beschreven systeem in de casus en zijn werking. Voor het systeem zijn extra vragen gesteld aan de opdrachtgever. Wanneer er naar aanleiding van de klant iets is aangepast word dit aangegeven. Ook wanneer er aannames gemaakt worden, wordt dit aangegeven

# Use case diagram

Een use case diagram is een simpele representatie van hoe een gebruiker op het systeem kan inwerken. Een use case diagram kan de verschillende types van gebruikers die het systeem zullen gebruiken identificeren. Een use case diagram wordt gebruikt om een “higher level view” van het systeem te krijgen. Dit betekent dat het een versimpelde en grafische versie van wat het systeem moet doen.

Het doel van deze opdracht was om een simpel beeld te krijgen van wat er uiteindelijk allemaal moet gebeuren.



Figuur : Use case diagram

## 2.1 Brief description

1. Registreren. Een persoon kan zich registreren om een klant te worden en gebruik te maken van RedCars.

2. Goedkeuring geven voor afschrijving. Omdat een klant moet kunnen betalen bij het gebruik van een pas moet er ook nog een goedkeuring gegeven worden voordat er een pas gestuurd kan worden.

3. Pasje naar klant sturen. Nadat alle vorige use cases zijn uitgevoerd word er een RedCars klantenpas thuis gestuurd en is de persoon een klant

4. Inloggen. Voordat er gebruik gemaakt kan worden van de site moet er ingelogd zijn.

5. Website bekijken. Om te kijken waar er RedCars plekken zijn en of er auto’s beschikbaar zijn op deze plek, moet er op de site gekeken worden.

6. Auto reserveren. Nadat er ingelogd is kan de klant een auto reserveren wanneer er een aanwezig is op de desgewenste plek. Als de klant een betalingsachterstand heeft is dit niet mogelijk.

7. Auto meenemen. Als er van te voren gereserveerd is kan de auto opgehaald worden met de klantenpas op een RedCars parkeerlocatie.

8. Auto terug brengen. Nadat de auto is meegenomen kan deze ook weer op de zelfde parkeerlocatie worden teruggebracht door de klant.

9. Bereken kosten. Nadat de auto is terug gebracht word door het systeem de kosten berekend op basis van het tarief en waar toepasselijk het aantal gereden kilometers.

10. Boete berekenen. Als een klant zich niet aan de bepaalde tijdsduur houd word er een boete in rekening gebracht door het systeem.

11. Betalingen innen. Nadat alle kosten zijn berekend word het bedrag geint door een extern bedrijf.

12. Klantgegevens beheren. Een medewerker kan de gegevens van klanten beheren en controleren

13. Klant inactief maken. Een medewerker kan een klant inactief maken waardoor deze geen auto’s meer kan reserveren en niet meer kan inloggen.

14. klantgegevens aanpassen. Een medewerker kan de gegevens van een klant aanpassen wanneer nodig. Er kunnen geen wachtwoorden gewijzigd worden op deze manier.

15. Auto informatie beheren. Een medewerker kan de informatie over een auto, zoals kosten en beschikbaarheid, bewerken en beheren.

# Niet functionele eisen

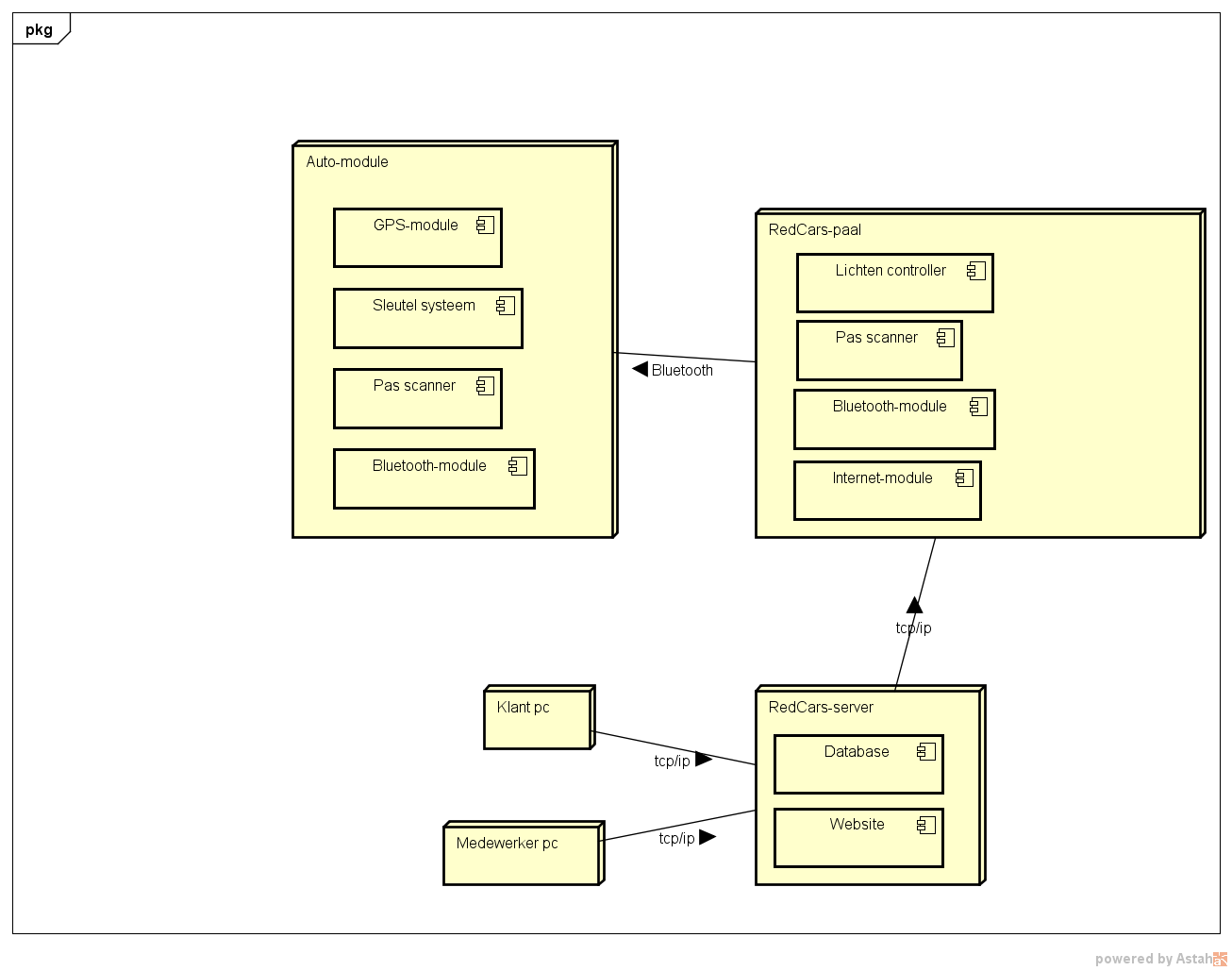
Hieronder staan een aantal niet functionele eisen die geïnterpreteerd kunnen worden door het lezen van de casus. Deze eisen zijn georganiseerd per alinea van de tekst en ze zijn volgens FURPS gerangschikt. FURPS is een acroniem dat een model weergeeft om kwaliteitseigenschappen van software in te delen in functionele en niet functionele eisen. Hierin staat de F voor functionaliteit, de U voor Usability, de R voor reliability, de P voor performance en de S voor supportability.

|  |  |  |
| --- | --- | --- |
| ID | Niet functionele eisen | Soort eis |
| 1 | Op 10% van de plaatsen is een stationwagen met trekhaak. | U |
| 2 | Klanten kunnen zonder eerdere ervaring met de RedCars site, binnen 5 minuten, een auto reserveren. | U |
| 3 | Op locaties waar veel vraag blijkt te zijn, en auto’s vaak allemaal weg zijn, worden extra auto’s geplaatst. | S |
| 4 | De RedCars website is minimaal 90% van de tijd bruikbaar. | R |
| 5 | Er is minimaal 75% van de tijd een auto beschikbaar bij elke RedCars paal. | R |
| 6 | Beschikbaarheid van auto’s wordt elk uur geüpdatet op basis van reserveringen. | U |
| 7 | Typen auto’s moeten binnen 10 minuten toegevoegd kunnen worden aan het systeem. | U |
| 8 | Typen abonnementen moeten binnen 15 minuten toegevoegd kunnen worden aan het systeem. | U |
| 9 | Een klant kan nadat er gereserveerd is binnen 5 minuten een auto meenemen. | U |
| 10 | Loop tijd van paal naar parkeerplaats bedraagt minder dan 2 minuten. | U |
| 11 | Klantenpas wordt binnen 2 weken thuis afgeleverd. | P |

# Deployment Diagram

Een deployment diagram laat de configuraties zien van de run time processing nodes en de componenten die daarin bestaan. Deployment diagrammen worden gebruikt voor het modeleren van het statische deployment overzicht van een systeem. Dit is grotendeels het modeleren van de topologie van de hardware waar het systeem op uitvoert.

Hieronder staat het deployment diagram voor RedCars. Hierin staat welke fysieke objecten die geheugen of processing geschiktheid hebben. Ook staat erin hoe alle objecten met elkaar verbonden zijn en welke componenten in de objecten bestaan.

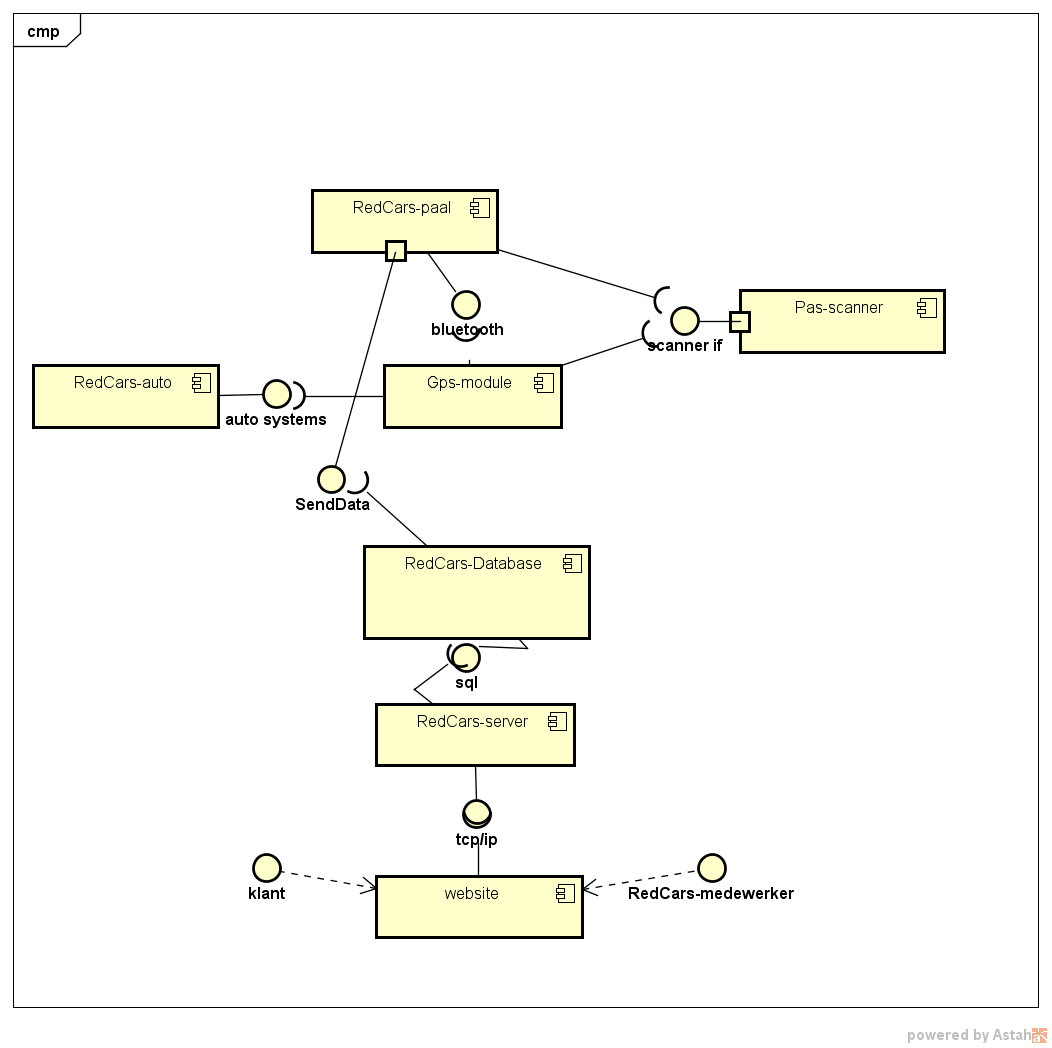


Figuur : Deployment diagram

# Component diagram

Een component diagram is een diagram waarin de structurele relaties tussen componenten van een systeem weergeven worden. Componenten zijn gemaakt van software objecten die hetzelfde doel hebben. Alle componenten communiceren met elkaar via interfaces.

Hieronder staat het componenten diagram voor Redcars. Hierin staat welke componenten er allemaal benodigd zullen zijn voor het bedrijf RedCars om alles te realiseren wat er bereikt moet worden.



Figuur : Component diagram

# Fully dressed use cases

In dit hoofdstuk worden de usecases rondom het reserveren en ophalen van de auto uitgewerkt in fully-dressed usecases.

## Use Case: Reserveren van een auto

|  |  |
| --- | --- |
| **Primary actor: Klant** | |
| **Stakeholders and Interests: RedCars, klant** | |
| **Brief description:**  **Een klant wil een auto reserveren via de website van RedCars.** | |
| **Preconditions:**  **Klant is op de website van RedCars.**  **Klant heeft een account.**  **Klant is ingelogd.**  **Klant heeft geen betalingsachterstand.** | |
| **Postconditions (Success Guarantee):**  **De klant heeft betaald.**  **De klant mag de auto ophalen.** | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. Klant gaat naar zoekpagina van RedCars’ website. | 1. Systeem toont auto’s per stad en de beschikbaarheid van deze auto’s. |
| 1. Klant selecteert een auto. | 1. Systeem toont gegevens van de auto. |
| 1. Klant kiest ervoor om de auto te reserveren. | 1. Systeem vraagt naar begin- en eindtijd van de reservering. |
| 1. Klant vult begin- en eindtijd in. | 1. Systeem reserveert auto voor klant. |
| **Extensions (Alternative Flow):** | |
| 5A1 klant wilt andere auto selecteren. | 5A2 Terug naar stap 2. |
|  | 6A1 auto is niet beschikbaar. |
|  | 6A2 website zorgt ervoor dat de klant deze auto niet kan reserveren. |

Tabel Fully dressed use case reserveren auto

## Use Case: Meenemen van een auto

|  |  |
| --- | --- |
| **Primary actor: Klant** | |
| **Stakeholders and Interests: RedCars, klant** | |
| **Brief description:**  **Een klant heeft een auto gereserveerd en haalt deze nu op.** | |
| **Preconditions:**  **De klant heeft een klantenpas.**  **De klant heeft de auto gereserveerd.**  **De klant is niet ingecheckt.**  **De klant is op de parkeerplaats waar de auto geparkeerd staat.** | |
| **Postconditions (Success Guarantee):**  **De klant heeft de auto opgehaald.**  **De klant is ingecheckt.** | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. Klant houdt pas tegen de in/uitcheck paal. | 1. Systeem check klant in en koppelt de pas aan de auto. |
| 1. Klant gaat naar de auto. |  |
| 1. Klant ontsluit de auto met de pas. | 1. Auto gaat van slot af. |
| 1. Klant stapt in. |  |
| 1. Klant start auto. |  |
| 1. Klant rijdt weg. |  |
| **Extensions (Alternative Flow):** | |
| 3A1 klant gaat naar verkeerde auto. |  |
| 3A2 klant probeert auto te ontsluiten. | 3A3 auto gaat niet van slot af. |

Tabel Ophalen van een auto

## Use Case: Terugbrengen van RedCars auto

|  |  |
| --- | --- |
| **Primary actor: Klant** | |
| **Stakeholders and Interests: RedCars, klant** | |
| **Brief description:**  **Een klant wil een auto terugbrengen bij de parkeerplaats waar de auto vandaan komt** | |
| **Preconditions:**  **Klant is op de parkeerplaats van de RedCars auto.**  **Klant heeft een klantenpas bij zich.**  **Klant heeft een abonnement voor de auto.**  **Klant heeft auto geparkeerd op RedCars parkeerplaats** | |
| **Postconditions (Success Guarantee):**  **De klant heeft RedCars auto teruggebracht.**  **De klant krijgt heeft betaling ontvangen in account.** | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. Klant houdt klantenpas tegen kaartlezer buiten de auto. | 1. Systeem kijkt of de auto op de correcte plek staat met gps-data. |
|  | 1. Systeem sluit auto af. |
| 1. Klant houdt klantenpas tegen de kaartlezer bij de paal. | 1. Systeem controleert type abonnement en hoeveel/hoelang er gereden is. |
|  | 1. Systeem stuurt data naar website voor maken factuur. |
| **Extensions (Alternative Flow):** | |
|  | 2A1 Systeem toont dat auto niet op correcte plek staat geparkeerd. |
| 2A2 Klant gebruikt pas om weer gebruik te maken van de auto en deze correct te parkeren | 2A2 Gaat verder bij stap 2. |

Tabel Terugbrengen van een auto

## Use Case: Betalen van de huur

|  |  |
| --- | --- |
| **Primary actor: Klant** | |
| **Stakeholders and Interests: RedCars, klant, Externe partij voor betalingsafhandeling** | |
| **Brief description:**  **Een klant heeft een RedCars auto gehuurd en teruggebracht en wilt zijn huur betalen.** | |
| **Preconditions:**  **Klant heeft RedCars auto gehuurd.**  **Klant heeft RedCars auto ingeleverd.**  **Klant is op betalingspagina van RedCars website**  **Klant is ingelogd.** | |
| **Postconditions (Success Guarantee):**  **De klant heeft betaald.**  **Openstaande betaling is afgesloten.** | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. Klant vraagt overzicht openstaande betalingen op. | 1. Systeem toont een overzicht van openstaande betalingen van huur. |
| 1. Klant selecteert de gewenste betaling. | 1. Systeem toont informatie over de betaling en geeft mogelijkheid om te betalen. |
| 1. Klant vult informatie in en selecteert “betalen”. | 1. Systeem maakt betaling aan en stuurt deze door naar externe partner. |
|  | 1. Systeem toont betaling geslaagd wanneer externe partner aangeeft dat betaling geslaagd is. |
| **Extensions (Alternative Flow):** | |
|  | 6A1 Systeem toont betaling niet geslaagd. |
|  | 6A2 Systeem gaat verder vanaf 1. |

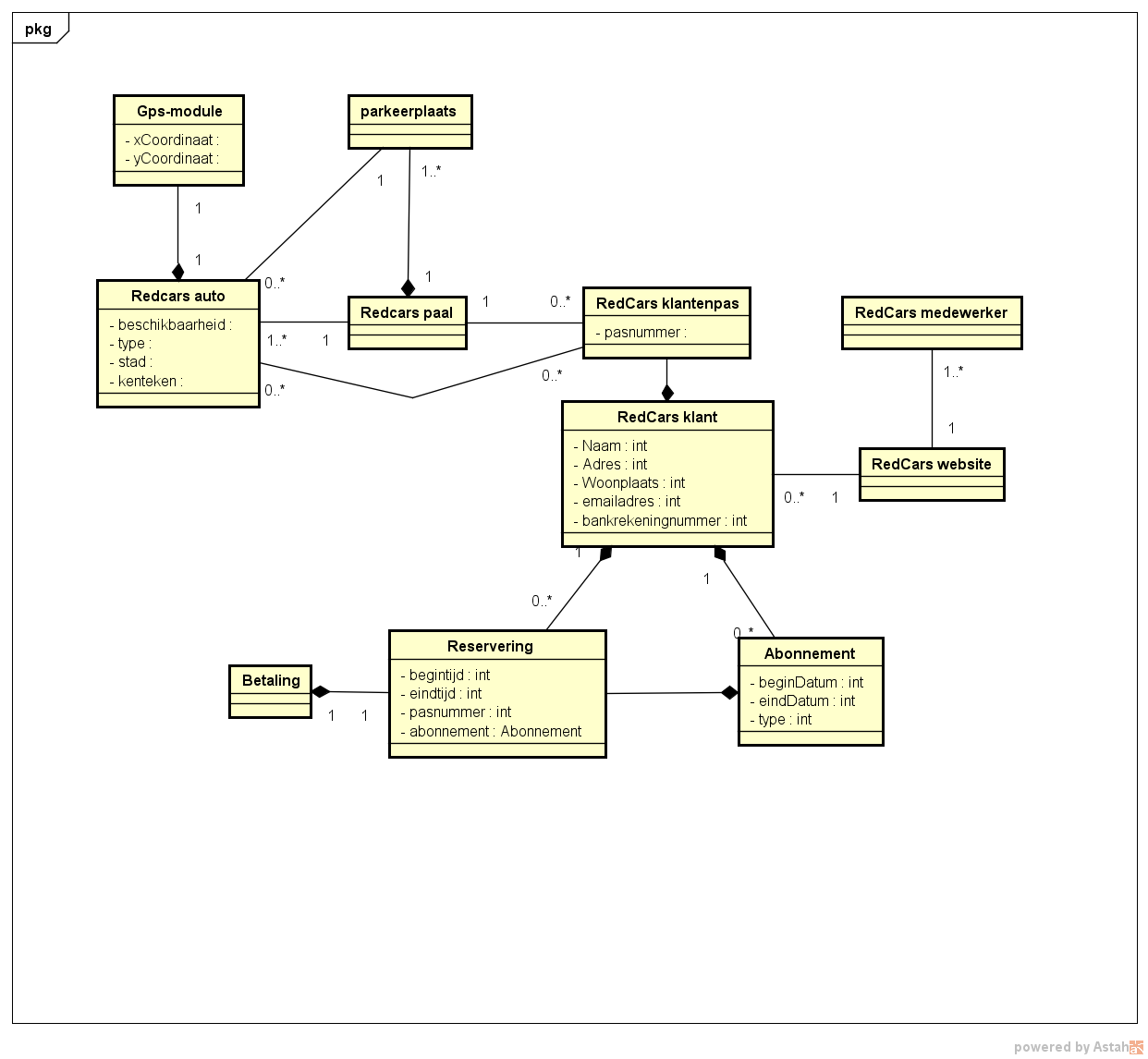
Tabel Betalen van een auto

r het

# Domeinmodel

Een domein model is een manier om dingen die in de echte wereld bestaan te beschrijven en modeleren en de relaties ertussen te tonen.

Hieronder staat het domein model voor RedCars. Hierin is alles wat binnen het domein van RedCars valt opgenomen met de relaties en multipliciteit ertussen.

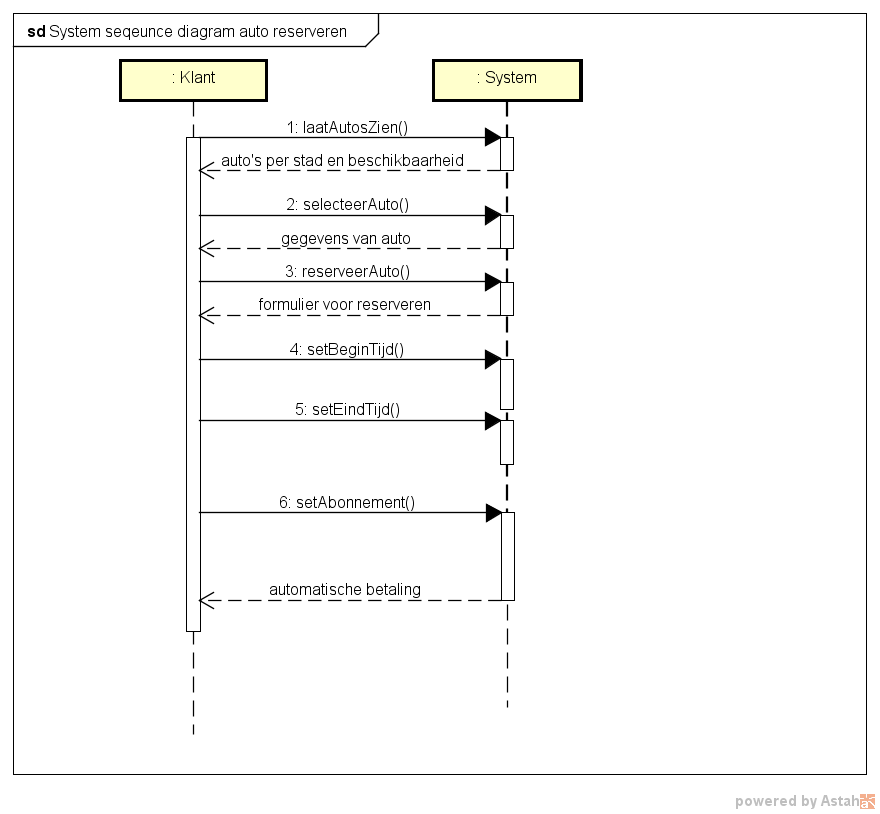


Figuur Domein model

Voor het domein van RedCars is het bovenstaande model gemaakt. Hierin zijn een aantal beslissingen gemaakt die toegelicht worden.  
Elke auto heeft een Gps-module, dit is zodat het systeem bij kan houden waar de auto op alle momenten is. De Gps-module zal ook gebruikt worden voor het vergrendelen en ontgrendelen van de auto. De Gps-module kan niet bestaan als de auto ook niet bestaat. Ook heeft elke auto een parkeerplaats waar hij op moet staan wanneer hij niet in gebruik is. Sommige van deze parkeerplaatsen hebben mogelijkheid om meerdere RedCars auto’s op kwijt te kunnen. Elke parkeerplaats heeft een paal. Dit is zodat wanneer de auto geparkeerd wordt op een parkeerplaats de gebruiker via de paal uit kan checken. De paal kan ook gebruikt worden voor meerdere auto’s tegelijkertijd. Een auto kan alleen wel maar een paal tegelijk hebben. Met de klantenpas kan ingecheckt worden via de paal, dit zorgt ervoor dat de pas gebruikt kan worden om de auto te ontgrendelen. Een pas kan maar bij een paal tegelijkertijd gebruikt worden. Een klantenpas hoort bij een specifieke klant, hierom kan deze ook niet bestaan als de klant niet bestaat. Vandaar de compositite. Een klant is in staat om een reservering te plaatsen, hij kan ook meerdere reserveringen hebben tegelijkertijd. Alleen een reservering kan niet bestaan zonder dat de klant bestaat. Hetzelfde geldt voor een abonnement. Zonder een reservering kan er ook geen betaling bestaan want dan is er niks om te betalen. Verder kan een reservering ook niet bestaan zonder dat er een abonnement is anders kunnen er geen kosten berekend worden. In het model staat ook een website, dit is een onderdeel van het domein omdat de klant via de website reserveringen aan kan maken. Deze website wordt onderhouden door medewerkers.

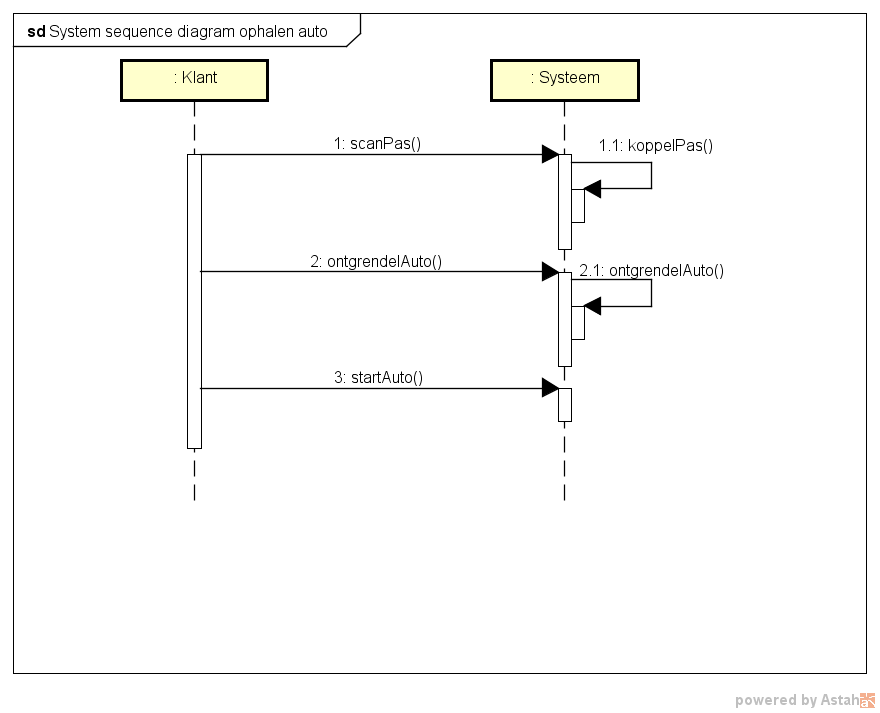
# Sequence diagrammen

Voor Redcars zijn er twee verschillende soorten sequence diagrammen ontworpen, ‘system sequence diagrammen’ en ‘component sequence diagrammen’. Een system sequence diagram is een diagram waarin de events getoond worden die een actor genereert. Hierin wordt het systeem als black box gezien. In een component sequence diagram wordt het systeem niet meer gezien als black box, maar als white box. Dit houdt in dat alle klasses en functies die gebruikt worden om de events die door de actor genereert worden weergeven worden.



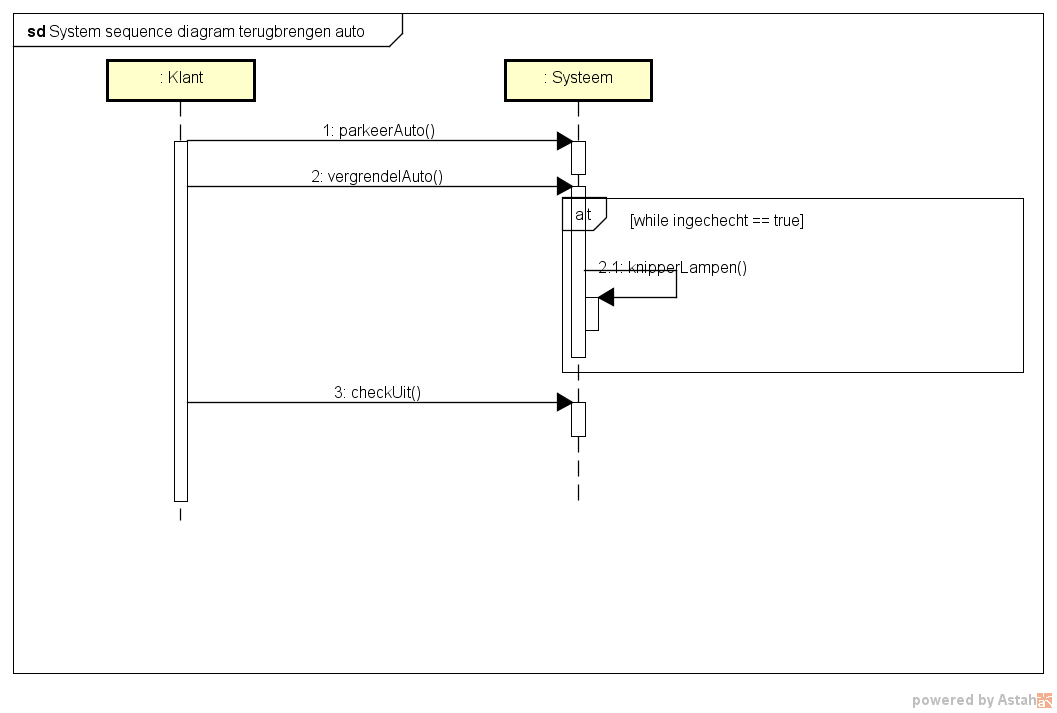
Figuur System sequence diagram auto reserveren

Hierboven staat het system sequence diagram voor het reserveren van een auto. Hier wilt de klant eerst zien welke auto’s er allemaal zijn. Deze worden getoond door het systeem. Vervolgens kiest de klant ervoor om een auto te selecteren, het systeem zal dan de gegevens van de auto tonen. Als de klant de geselecteerde auto wilt reserveren zal hij een formulier in moeten vullen. In dit formulier moet de begintijd, de eindtijd en het abonnement dat de klant wilt ingevuld worden. Wanneer dit allemaal gedaan is zal er een automatische betaling plaats moeten vinden.



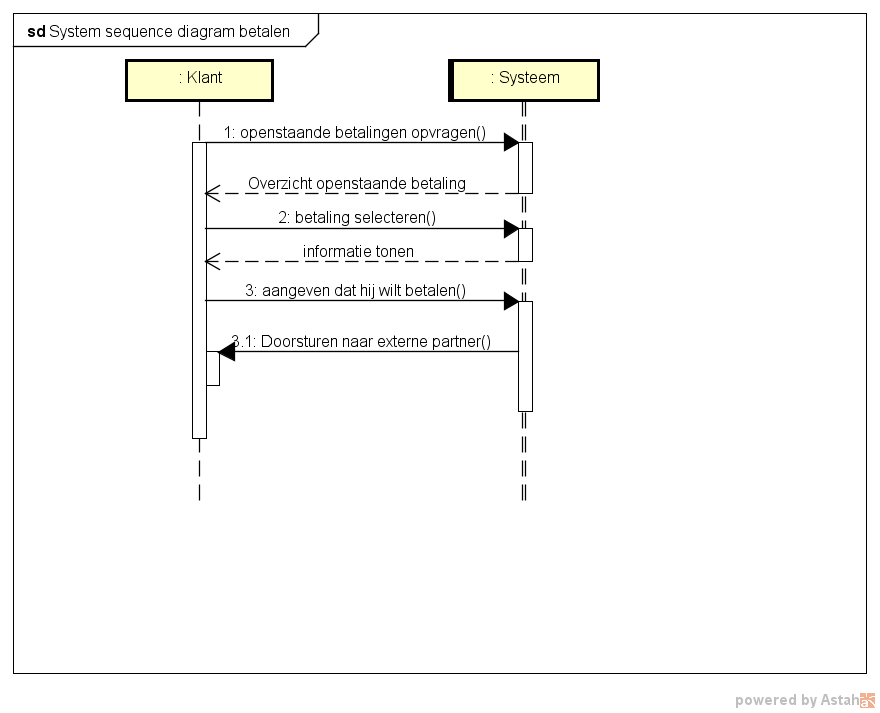
Figuur System sequence diagram ophalen auto

Hierboven staat het system sequence diagram van het ophalen van een auto. Hier wilt de klant een auto ophalen, hiervoor moet hij zijn klantenpas scannen en koppelt het systeem zijn pas aan een auto. Vervolgens kan de klant naar de auto toe gaan en deze opendoen en kan hij er gebruik van maken.



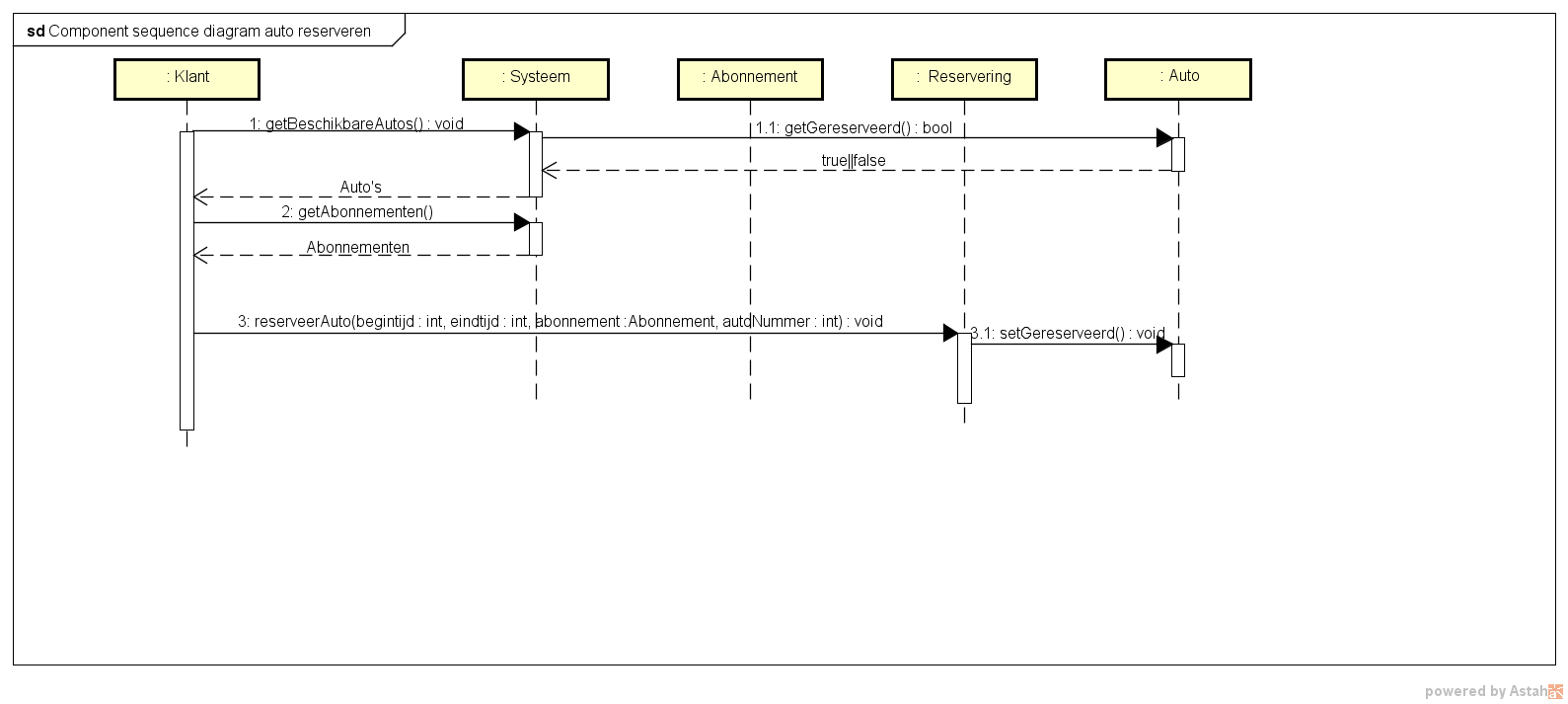
Figuur System sequence diagram terugbrengen auto

Hierboven staat het system sequence diagram voor het terugbrengen van een auto, om dit te doen zal de auto geparkeerd moeten worden op een RedCars parkeerplaats. Vervolgens moet de auto op slot gedaan worden. Wanneer de klant van de auto wegloopt en nog niet uitgecheckt is zullen er lampen gaan knipperen als de klant nog ingecheckt is zodat hij niet vergeet uit te checken. Vervolgens checkt de klant uit.



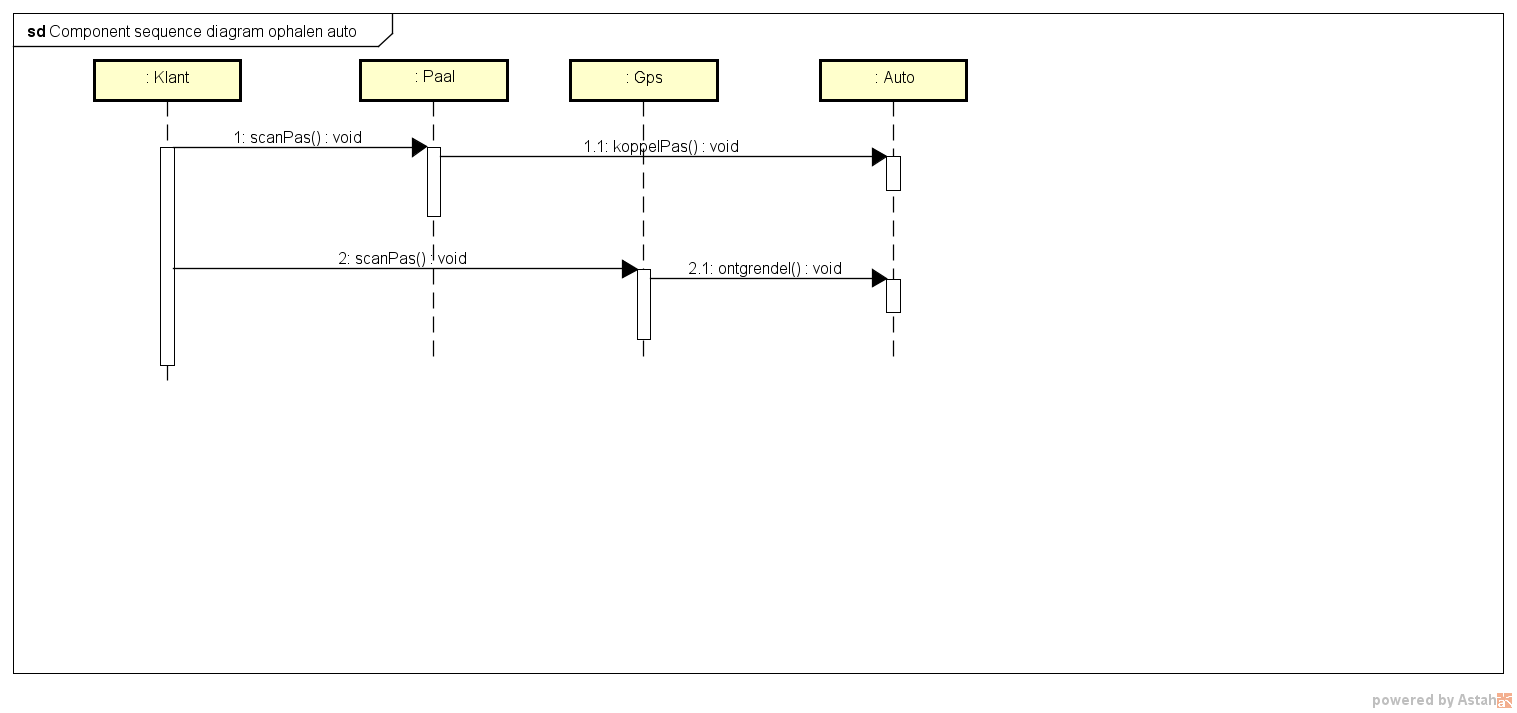
Figuur System sequence diagram betalen

Hierboven staat het system sequence diagram voor het betalen. De klant vraagt aan het systeem de openstaande betalingen op zodat hij een kan kijken hoeveel hij er allemaal heeft. Het systeem geeft alle openstaande betalingen terug. Vervolgens kan de klant ervoor kiezen om hem te selecteren, waarna het systeem de informatie van de gekozen openstaande betaling toont. Hierna kan de klant aangeven dat hij wilt betalen en zal het systeem hem doorsturen naar de externe partner die de betalingen verzorgt.



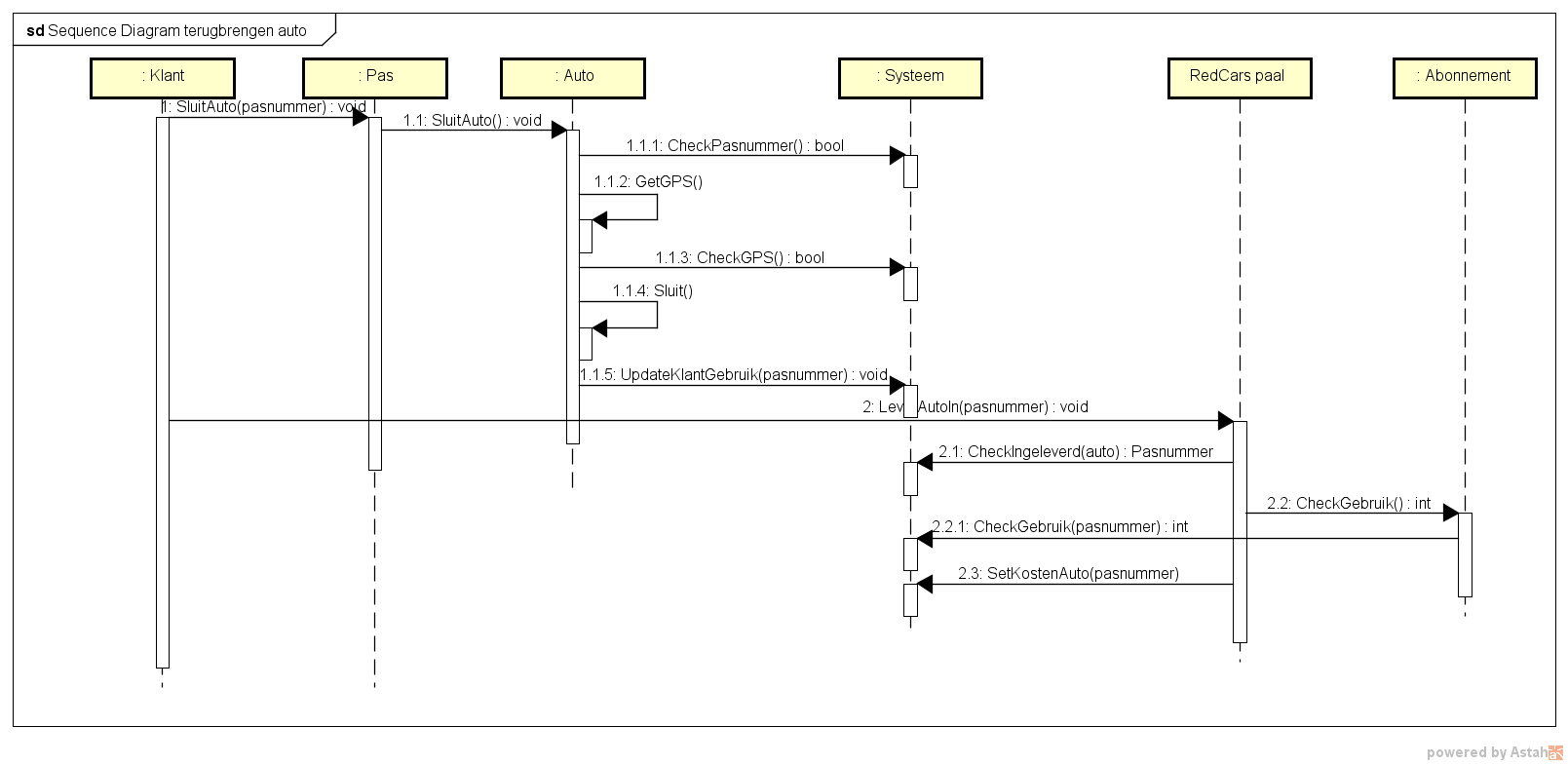
Figuur Component sequence diagram auto reserveren

Hierboven staat het component sequence diagram voor het reserveren van een auto. Om een auto te reserveren moet de klant eerst weten welke auto’s er allemaal beschikbaar zijn. Hiervoor wordt functie getBeschikbareAutos() van Systeem aangeroepen. Deze returned de autos die niet gereserveerd zijn. Dit doet het systeem door voor elke auto die bestaat op te vragen of deze wel of niet gereserveerd is. Vervolgens moet de klant weten uit welke abonnementen hij allemaal kan kiezen voor om een reservering te plaatsen. Hiervoor vraagt hij aan het systeem op welke abonnementen hij allemaal uit kan kiezen. Als de klant dit allemaal eenmaal weet kan hij een reservering plaatsen. Dit doet hij door de functie reserveerAuto(begintijd, eindtijd, abonnement, autonummer) van Reservering aan te roepen. Wanneer dit succesvol is wordt de auto gereserveerd.



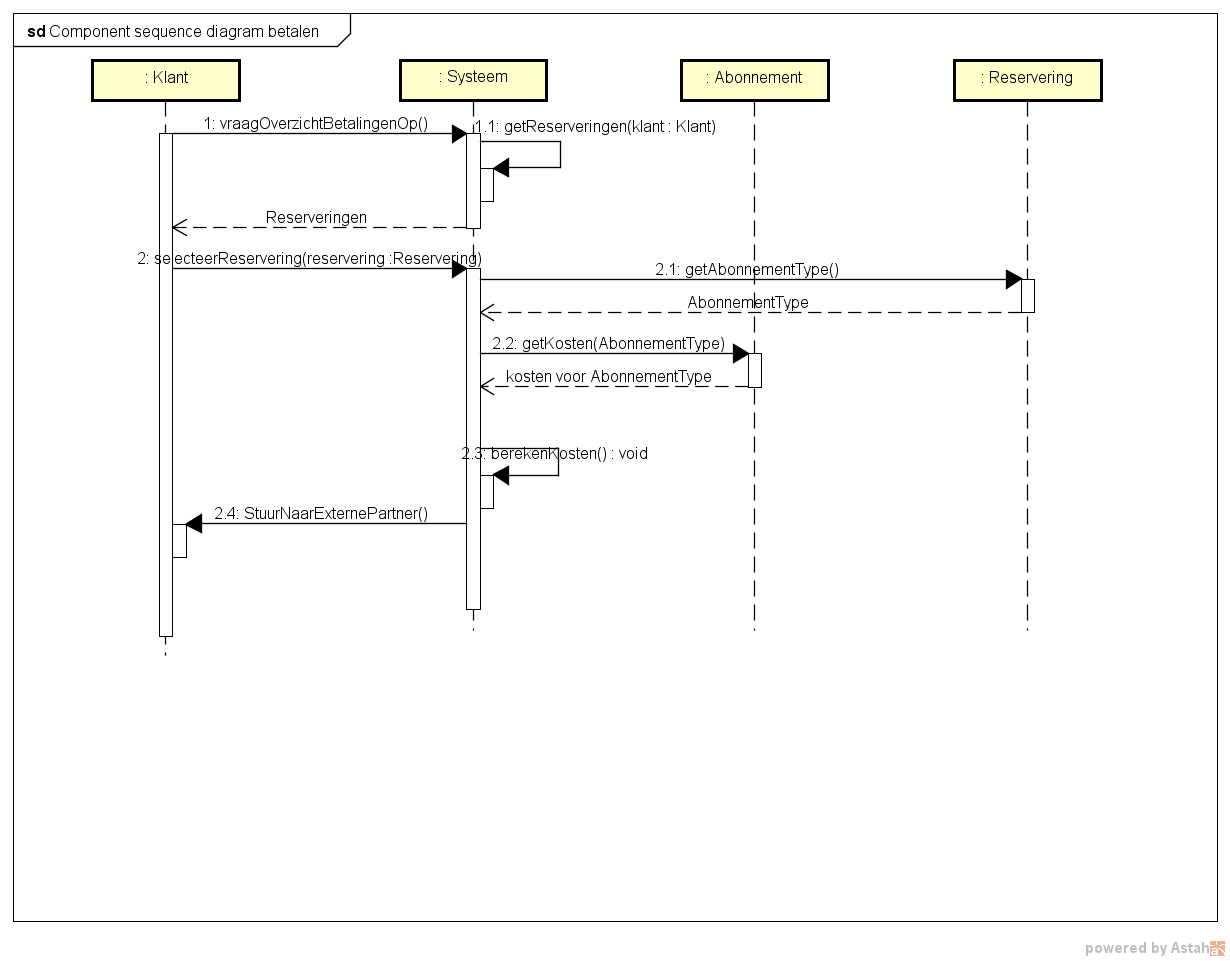
Figuur Component sequence diagram ophalen auto

Hierboven staat de component sequence diagram voor het ophalen van een auto. Dit kan de klant doen door wanneer hij een auto gereserveerd heeft zijn pasje tegen de paal aan te houden. Hierdoor wordt de klant ingecheckt en wordt de pas aan de auto gekoppelt. Vervolgens moet de klant zijn pas bij de auto scannen waardoor de auto ontgrendeld wordt en kan de klant instappen en wegrijden.



Figuur Component sequence diagram terugbrengen auto

Hierboven staat het component sequence diagram voor het terugbrengen van een auto. Hiervoor moet de klant de auto afsluiten, dit doet hij door de pas te gebruiken bij de deur van de auto te houden wanneer deze gesloten is. Vervolgens wordt er gecontroleerd of de pas daadwerkelijk aan de auto gekoppeld was. Vervolgens kijkt wordt de locatie van de auto opgehaald, zodat er gecontroleerd kan worden of de auto op een RedCars parkeerplaats. Vervolgens gaat de auto daadwerkelijk op slot. Hierna wordt het klantgebruik geupdatet zodat het systeem weet hoe lang de auto in gebruik is geweest. Vervolgens gaat de klant met de pas naar de paal en houdt de pas tegen de paal. Vervolgens wordt het gebruik gecheckt om te kijken of dit overeenkomt met het afgesloten abonnement. Hierna worden de kosten bepaalt.

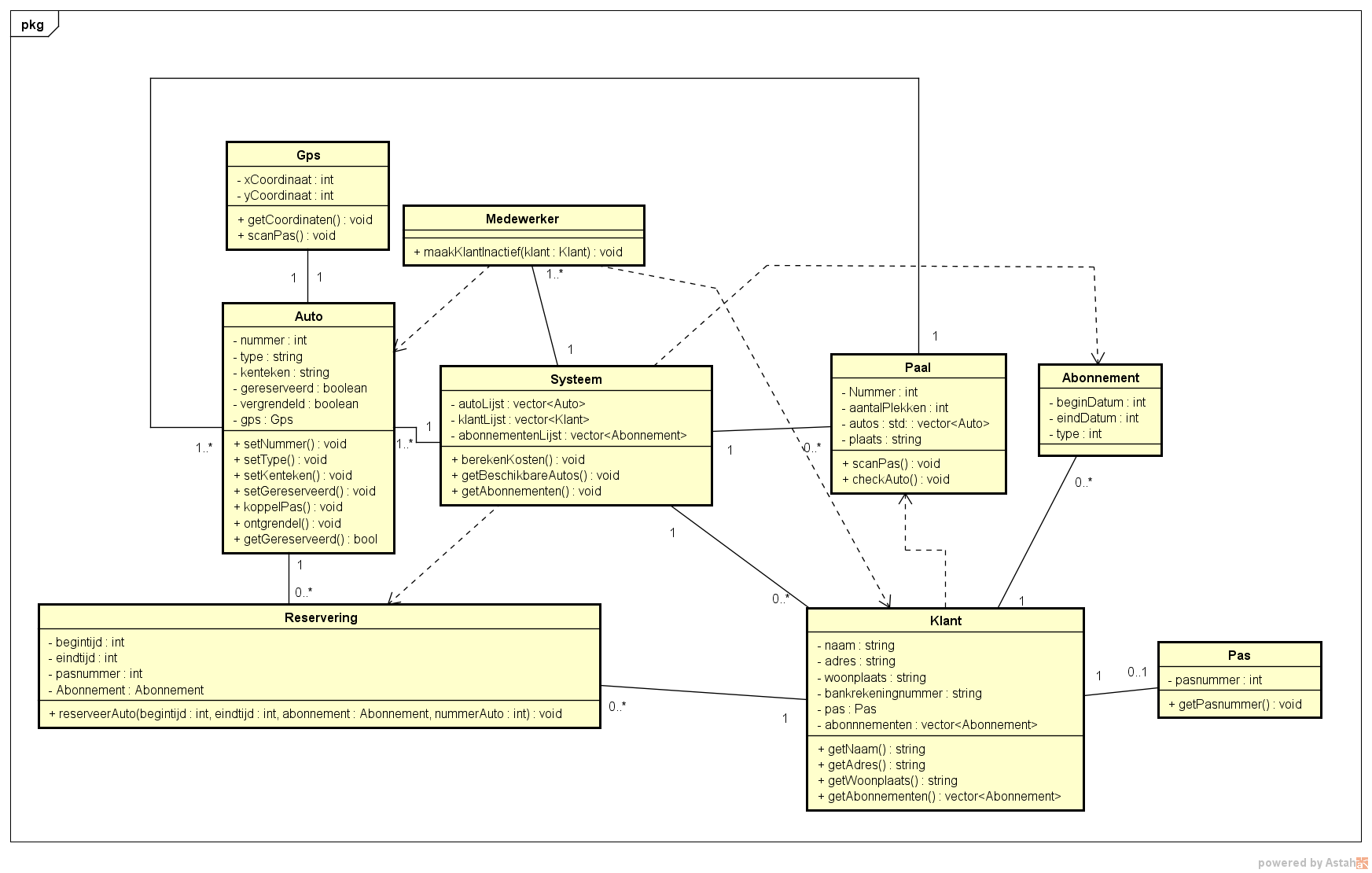


Figuur Component sequence diagram betalen

Hierboven staat het component sequence diagram voor het betalen van openstaande betalingen. Allereerst zal de klant aan systeem opvragen welke betalingen hij allemaal heeft. Om dit te doen kijkt het systeem welke reserveringen er allemaal van de klant zijn en geeft deze terug aan de klant. Vervolgens kan de klant een van deze reserveringen selecteren en zorgt het systeem ervoor dat hij via de reservering het type abonnement krijgt wat bij deze reservering hoort. Vervolgens vraagt het systeem aan het systeem op hoe duur het abonnement is. Hierna berekent het systeem de kosten met door middel van het type abonnement te kijken en naar hoe de klant de auto gebruikt heeft. Wanneer deze kosten berekent zijn zal de klant doorgestuurd worden naar de externe partner die de betaling regelt en dan regelt de externe partner alles verder af.

# Design class diagram

Een klasse diagram beschrijft de structuur van een object georiënteerd systeem door de klasses in het systeem en de relaties ertussen te tonen. Ook laat het constraints, attributen en methoden zien.



Figuur Design class diagram

Bronvermelding

Scaled Agile. (4 augustus 2017). Domain Modeling  
Geraadpleegd op 29-10-2018  
<https://www.scaledagileframework.com/domain-modeling/>

Wikipedia. (8 november 2017). FURPS  
Geraadpleegd op 29-10-2018  
<https://nl.wikipedia.org/wiki/FURPS>

Hogeschool van Arnhem en Nijmegen. (datum onbekend). Class diagrams.pdf  
geraadpleegd op 28-10-2018  
<https://onderwijsonline.han.nl/elearning/lessonfile/AN7lLlry/eyJpdiI6Im5aNVRWUXc0dUVjUTF3eDRobTdWK2c9PSIsInZhbHVlIjoiQzNWVVUzdEdtTkNtd0JUVFNoOUx5dUp2cmRXOTNaZXdPQkVvaWNOXC9DVE09IiwibWFjIjoiMmEzNGJkNGIyNTViNjFjMjcxMDdkZGY4ZjYxZTUwMmUzYWI1YjU4ODY2MGQzMWZhMzY3NDc4ZTQ2MDljMGM5YyJ9>