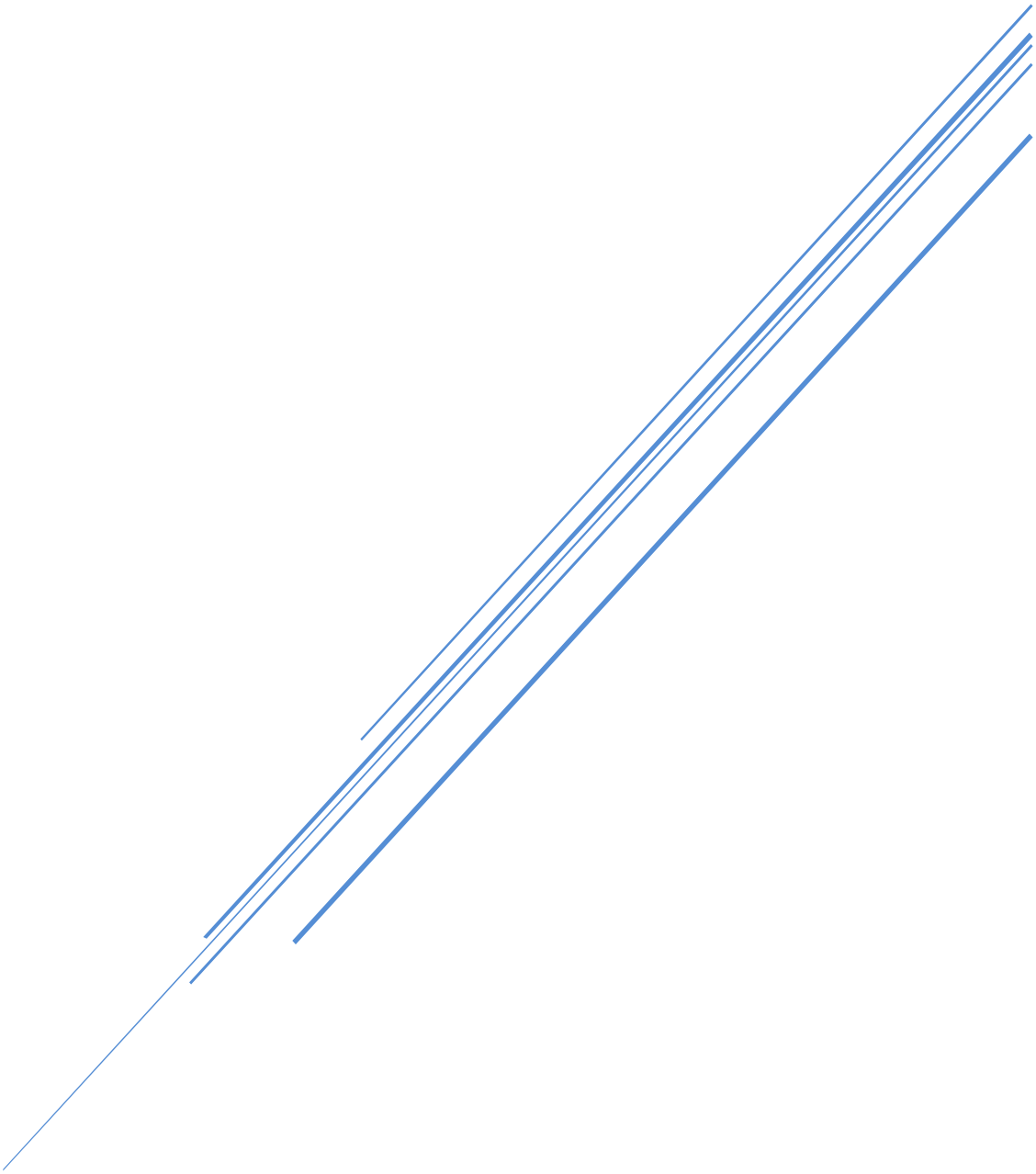
REDCARS CASUS

Case Study OSM-Modeling



Door: Evren Kilic, Tom Sievers en Raymond van Sommeren
Docent: Richard Holleman Klas: ITA-OSM-M-a

Inhoudsopgave

1. Inleiding

RentIt wil een systeem laten maken om huurauto's aan te bieden. Deze auto's kunnen van tevoren gereserveerd worden en dan op een parkeerplaats opgehaald worden. Voor een complete opdracht omschrijving zie de bijlage "Opdrachtomschrijving". De opdracht om dit systeem te ontwerpen is gegeven aan studenten van de HAN. Om dit systeem in zijn geheel te kunnen ontwerp moet als eerste een overzicht gemaakt worden van de mogelijke functionaliteiten en door wie deze gebruikt zullen worden. Deze functionaliteiten worden in dit document benoemd als "use cases" en de personen die hiervan gebruik maken heten "actors". Hierop volgt een beschrijving van wat de acties van het systeem en de actor zijn in de use cases. Hierna volgt een beschrijving van het domein waarin het systeem moet werken. Dit wordt gevolgd door uitleg van welke software op welk stuk hardware uitgevoerd wordt. Hierop volgen de relaties tussen meerdere software component. Hierna volgen de specificaties van de hiervoor opgestelde softwarecomponenten. Daarop volgt de interne uitwerking van een use case binnen en tussen de verschillende softwarecomponenten. Als laatste volgt een stuk uitleg over hoe de code georganiseerd is. Bij al deze hoofdstukken worden de gemaakt keuzes over het systeem uitgelegd.

2. Brief use cases

2.1. Use Case: Beschikbare auto’s inzien

Brief Description	De klant kan op de website van RedCars gaan en hier naar het aanbod van auto’s kijken waarbij hij de gebruiks tijd opgeeft en hiervan de locatie en reservaties ziet.

Tabel 1 Use Case: Beschikbare auto's inzien

2.2. Use Case: Registreren

Brief Description	Een mogelijke klant kan zich als klant registreren bij RedCars

Tabel 2 Use Case: Registreren

2.3. Use Case: Ophalen gereserveerde auto

Brief Description	Klant haalt auto op die door hen gereserveerd is

Tabel 3 Use Case: Ophalen gerserveerde auto

2.4. Use Case: Auto terugbrengen

Brief Description	Klant brengt de gebruikte auto terug naar de RentIt parkeerplaats en checkt uit.

Tabel 4 Use Case: Auto terugbrengen

2.5. Use Case: Auto reserveren

Brief Description	De klant reserveert een auto.

Tabel 5 Use Case: Auto Reserveren

2.6. Use Case: Klantgegevens beheren

Brief Description	Gegevens van een klant worden aangepast door een systeembeheerder van RedCars.

Tabel 6 Use Case: Klantgegevens beheren

2.7. Use Case: Autogegevens beheren

Brief Description	Gegevens van een auto worden aangepast door een systeembeheerder van RedCars.

Tabel 7 Use Case: Autogegevens beheren

2.8. Use Case: Auto openen

Brief Description	De auto wordt geopend met de pas van de klant als deze op zijn naam staat.

Tabel 8 Use Case: Auto openen

2.9. Use Case: Auto afsluiten

Brief Description	De auto wordt afgesloten met de pas van de klant.

Tabel 9 Use Case: Auto afsluiten

2.10. Use Case: Huur betalen

Brief Description	Een verzoek op incasso wordt doorgegeven aan het banksysteem

Tabel 10 Use Case: Huur betalen

2.11. Use Case: Abonnement kiezen

Brief Description	De klant kan het type van zijn abonnement aanpassen

Tabel 11 Use Case: Abonnement kiezen

2.12. Use Case: Inloggen

Brief Description	De klant kan inloggen op de website van RentIt.

Tabel 12 Use Case: Inloggen

3. Requirements

3.1. Usability

Naam	Omschrijving
Gebruiksvriendelijkheid website	De website moet voor een gebruiker van gemiddelde leeftijd van 40 een auto te reserveren zijn binnen 5 minuten.

Tabel 13 Usability

3.2. Performance

Naam	Omschrijving
Informatie snel ophalen	De informatie van de diverse auto's moet binnen 1 sec geladen worden bij een ongeveer last van 10000 bezoekers.

Tabel 14 Performance

3.3. Supportability

Naam	Omschrijving
Toevoegen nieuwe categorieën	Nieuwe type auto's en abonnementen moeten toegevoegd kunnen worden.

Tabel 15 Supportabilty

4. Fully dressed use cases

4.1. Use Case: 3. Ophalen gereserveerde auto

Primary actor: Klant	
Stakeholders and Interests: - Klant - RentIt	
Brief description: Klant haalt de gereserveerde auto op bij een van de RentIt parkeerplaatsen	
Preconditions: - Klant heeft een auto gereserveerd - De gereserveerde auto is aanwezig	
Postconditions (Success Guarantee): De klant heeft de gereserveerde auto opgehaald en kan deze met zijn pas openen	
Main Success Scenario (Basic Flow):	
Actor Action	**System Responsibility**
1. Klant gaat naar de locatie van de gereserveerde auto.	
2. Klant checkt in bij in/uitcheck paal	
	3. Systeem controleert validiteit van de gegevens
	4. De gegevens zijn valide, de klant is ingecheckt
	5. Het systeem verandert de status van de auto
6. De klant kan doorgaan met de use case “Auto openen”.	
Extensions (Alternative Flow):	
4-> Bij de gebruiker is een betalingsachterstand ontdekt	Gebruiker kan niet inchecken en de auto openen

Tabel 16 Use Case: 3. Ophalen gereserveerde auto

4.2. Use Case: 5. Auto reserveren

Primary actor: Klant	
Stakeholders and Interests: Klant, RentIt	
Brief description: Klant reserveert een auto op de pagina van RentIt	
Preconditions: - Klant heeft een reserveerbare auto gevonden (Use case: Beschikbare auto's inzien) - Klant is ingelogd (Use case: Inloggen)	
Postconditions (Success Guarantee): - Klant heeft een auto gereserveerd - De reservering is in het systeem afgehandeld	
Main Success Scenario (Basic Flow):	
Actor Action	**System Responsibility**
1. Klant kiest om te reserveren	
	2. Systeem toont de reservatie pagina
3. Klant kiest datum van ingang en einddatum.	
4. Klant bevestigd reservering	
	5. Systeem slaat reservering op.
Extensions (Alternative Flow):	
5-> Er is een betalingsachterstand ontdekt, reservering wordt niet opgeslagen.	Systeem toont een foutmedling

Tabel 17 Use Case: 5. Auto reserveren

4.3. Use Case: 4. Auto terugbrengen

Primary actor: Klant	
Stakeholders and Interests: Klant, RentIt	
Brief description: Klant brengt de gebruikte auto terug naar een RentIt parkeerplaats en checkt uit.	
Preconditions: - Klant heeft een gereserveerde auto op zijn naam. - De auto is geparkeerd op een RentIt parkeerplaats. - De auto is vergrendeld	
Postconditions (Success Guarantee): - Klant heeft de auto teruggebracht - De auto is weer beschikbaar gesteld	
Main Success Scenario (Basic Flow):	
Actor Action	**System Responsibility**
1. Klant heeft de auto op slot gedaan volgens de use case “Auto afsluiten”	
	2. Systeem registreert dat de auto op slot is en de auto op de parkeerplaats staat
	3. Auto en in/uitcheck paal geven lichtsignalen om aan te geven dat er nog niet is uitgecheckt
4. Klant checkt uit bij in/uitcheckpaal	
	5. Systeem registreert afronding van terugbrengen, ook worden terugbreng datum en km stand geregistreerd.
	6. Systeem gaat door met de use case “Huur betalen”.
Extensions (Alternative Flow):	
4. Klant checkt niet uit bij de paal	
	5. Het lichtsignaal wordt gestopt na 5 minuten als nog niet uitgecheckt is wordt een boete toegevoegd bij de klant. Een verlaagd vertrouwen in klant wordt in het systeem geregistreerd.
	6. Systeem gaat door met de use case “Huur betalen” met een boete.

Tabel 18 Use Case: 4. Auto terugbrengen

4.4. Use Case: 8. Betaling huur

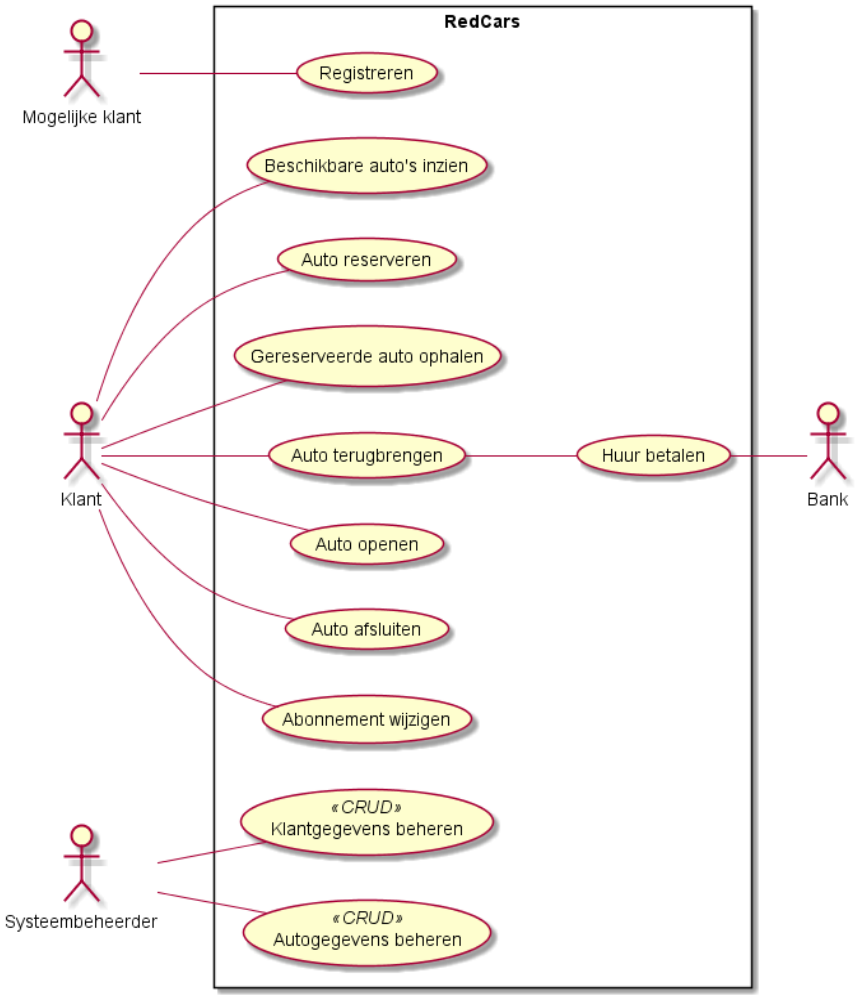
Primary actor: Bank	
Stakeholders and Interests: Klant, RentIt	
Brief description: Een verzoek op incasso wordt doorgegeven aan het banksysteem	
Preconditions: De use case “Auto terugbrengen” moet hiervoor uitgevoerd zijn.	
Postconditions (Success Guarantee): Incasso is betaald	
Main Success Scenario (Basic Flow):	
Actor Action	**System Responsibility**
	1. Systeem berekent bedrag aan de hand van abonnement, datum van terugbrengt en/of aantal gereden kilometers.
	2. Systeem geeft een incasso bericht door aan het banksysteem en registreert bij de gebruiker een betalingsachterstand.
3. Banksysteem handelt het incasso bericht af.	
4. Banksysteem geeft aan dat incasso gelukt is.	
	5. Incasso is gelukt betalingsachterstand wordt verwijderd van het account van de gebruiker.
Extensions (Alternative Flow):	
2. Incasso is mislukt.	

Tabel 19 Use Case: 8. Betaling huur

5. Use Case

Om gebruik te maken van het RedCars netwerk moeten mogelijke klanten zich eerst registreren. Wanneer iemand zich registreert krijgt hij/zij een pas thuisgestuurd. Met deze pas krijgt de gebruiker toegang tot de andere functionaliteiten van het systeem. Hiernaast zijn er binnen het RedCars systeem ook systeembeheerders. Systeembeheerders zijn geen gebruikers aangezien zij op een andere wijze zullen inloggen en anders met het systeem te werk zullen gaan. Indien men van het systeem gebruik gaat maken als een beheerder heeft hij alleen de mogelijkheid om de gegevens van auto's en klanten te beheren. Natuurlijk is het voor een beheerder ook mogelijk om een auto te huren maar in dat geval nemen ze deel aan het systeem als "Klant". De aan de klant gekoppelde use cases beschrijven alle manieren voor een klant om gebruik te maken van het RedCars systeem, hierin zijn er acties welke de klant vanuit huis doet met een applicatie, zoals het reserveren van een auto en het wijzigen van het abonnementstype. Daarnaast zijn er ook interacties die de klant in de fysieke wereld onderneemt, zoals het ophalen van de auto en het openen van de auto.

Het betalen van de huur volgt op het terugbrengen van de auto hierin speelt naast de klant ook de bank een rol. De bank zal namelijk de, door het systeem verstuurde, betaling afhandelen. Het systeem zou eventueel ook uitgebreid kunnen worden om openstaande betalingen periodiek af te handelen, hier is in het huidige ontwerp geen rekening mee gehouden.



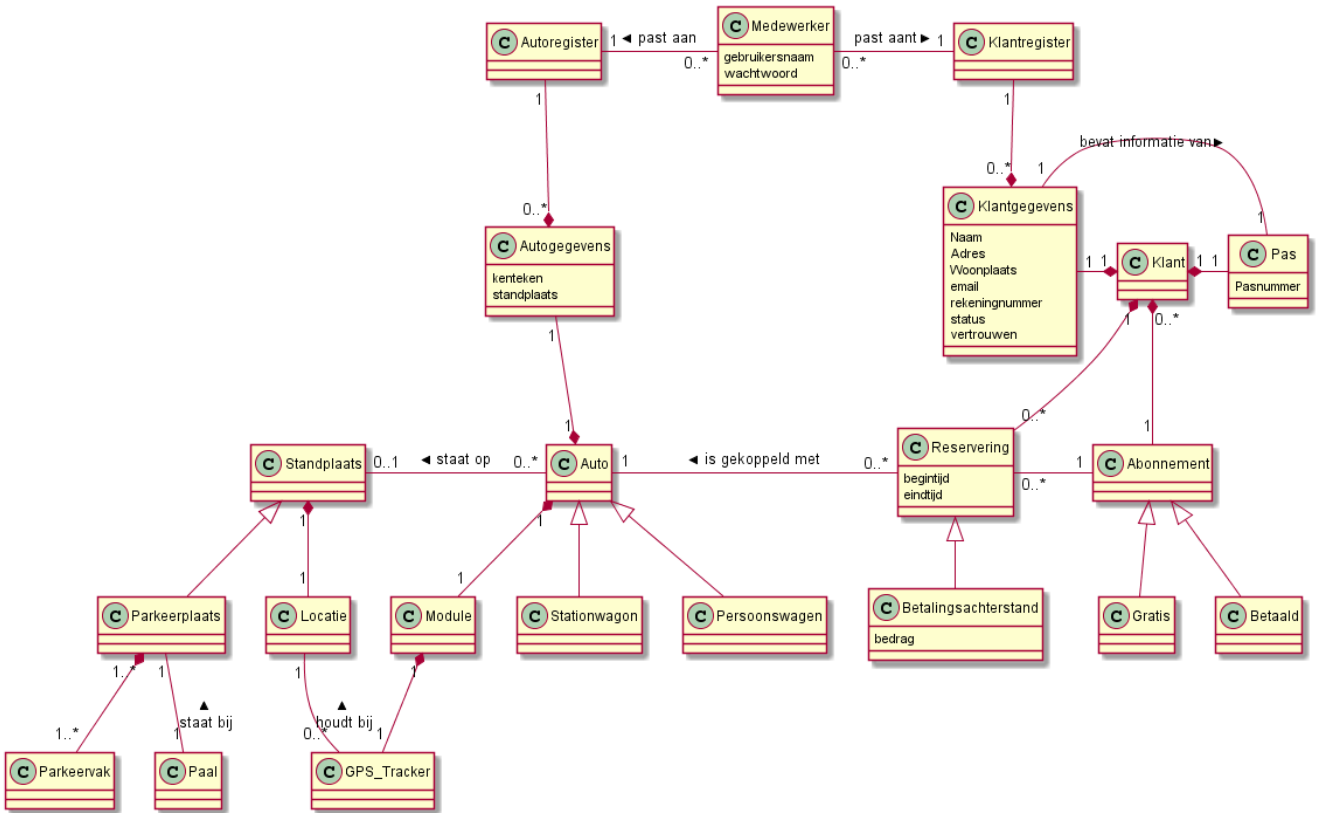
Figuur 1 Use Case Diagram

6. Domain model

Binnen het RedCars domein kunnen de medewerkers van RentIt zowel de Klantgegevens als de Autogegevens aanpassen, deze gegevens horen respectievelijk bij klanten en auto's. De medewerkers passen dit aan via de bijbehorende registers. Auto's zijn verder uitbreidbaar met een type, hier kunnen nog meer types aan toegevoegd worden maar dat is binnen het huidige domein niet gemodelleerd. Verder is er in het huidige domein rekening gehouden met twee soorten abonnementen.

De module in de auto bevat nu alleen nog een GPS-Tracker om de locatie van de auto bij te houden, deze zou later ook nog uitgebreid kunnen worden indien er meer functionaliteit wordt toegevoegd. Naast dat de auto een locatie heeft, hebben in dit domein standplaatsen ook een locatie. Een standplaats wordt hier gedefinieerd als een willekeurige plek voor de auto om te parkeren en hoeft dus geen RedCars parkeerplaats te zijn. Indien het wel over een RedCars parkeerplaats gaat wordt dit gemodelleerd met het woord "Parkeerplaats", bij een dergelijke parkeerplaats hoort ook een paal waar kan worden in- en uitgecheckt. Een "Parkeerplaats" kan uit een of meer "Parkeervakken" bestaan. Op een parkeervak kan een auto van RedCars geparkeerd zijn.

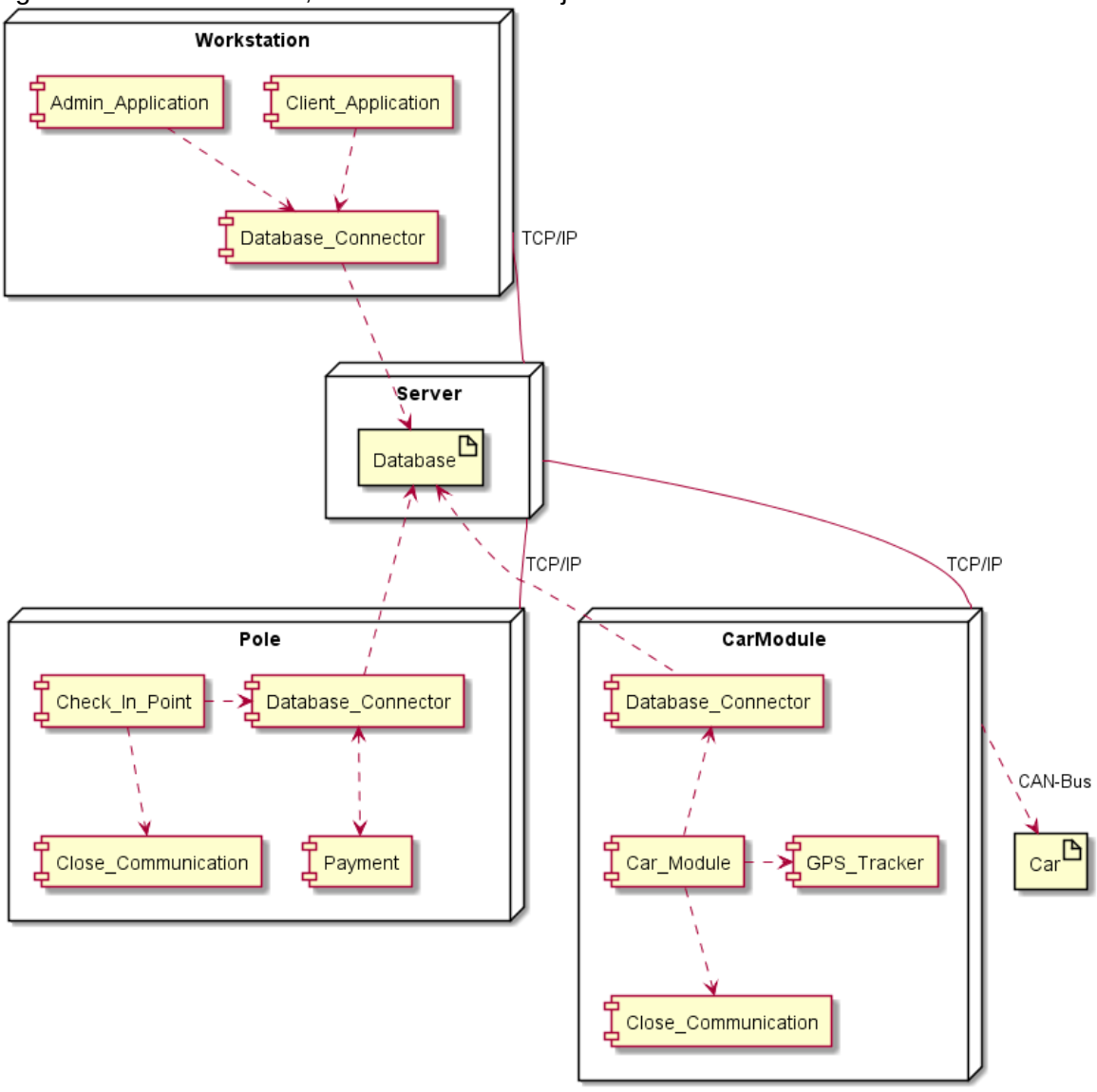
Als laatste wordt er aan een reservering zowel een auto als een klant gekoppeld, zo is dit domein concept een koppeling tussen die twee. Verder moet een reserving een abonnement bijhouden, zodat het bekend is hoe de betaling moet verlopen wanneer de auto geretourneerd is. Verder wordt er uit een reservering een betalingsachterstand gecreëerd wanneer een auto is geretourneerd. Deze achterstand houdt het bedrag bij wat de klant nog verschuldigd is en weerhoud de klant ervan om een nieuwe reservering te maken.



Figuur 2 Domein model

7. Deployment Diagram

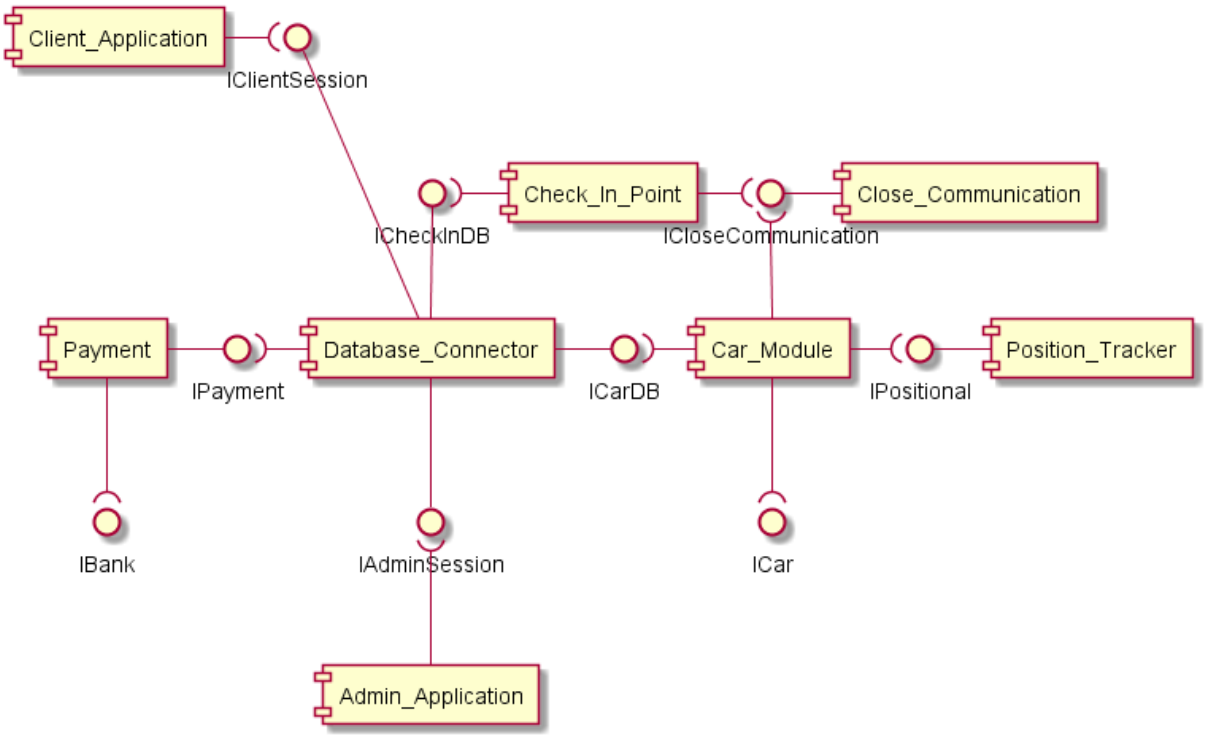
Om het systeem te kunnen gebruiken moet de software op diverse hardware ingezet worden. De interactie van de diverse onderdelen met de server wordt geregeld via TCP/IP. Hiervoor is gekozen, omdat bij de auto voor een draadloze optie gekozen moet worden. Verder is het draadloos netwerk via 2G/3G/4G al zover uitgebreid dat binnen Nederland alles afgedekt kan worden. Hierdoor hoeft er geen geld gestoken worden in de opbouw van een infrastructuur. Er is geen communicatie direct tussen de paal en de auto. De keuze hiervoor is gemaakt om zo mogelijke problemen te voorkomen die op zouden treden als de communicatie met de server niet mogelijk is, maar de communicatie tussen paal en auto wel aanwezig is. De module in de auto moet natuurlijk ook met de auto kunnen communiceren om bijv. de deuren te openen of te sluiten, dit wordt gedaan m.b.v. het CAN-bus protocol. Op de server draait een database die benaderd kan worden via een connector. De paal, auto module en de applicaties hebben allemaal een database connector. De afhandeling van de betaling is afhankelijk van de database connector waarvoor er gekozen moet worden om deze in de paal te zetten. Een betere oplossing was geweest om tussen de database connector en bijv. de paal een eigen component te maken wat verantwoordelijk is voor de communicatie. Dit component zou dan twee interfaces aanbieden, een voor de paal en een voor de database connector. Dit zou het effect hebben dat het protocol waarmee naar de server gecommuniceerd wordt, uitwisselbaar zou zijn.



Figuur 3 Deployment diagram

8. Component Diagram

De Database_Connector heeft meerdere verschillende interfaces. Dit is zo gedaan om zo, het andere component wat hiervan gebruik maakt, alleen de functies ter beschikking te stellen die dan ook echt nodig zijn. De Car_Module en de paal maken allebei gebruik van hetzelfde interface, omdat deze met de pas moeten kunnen communiceren. Hiernaast wordt dit component ook gebruikt om te bevestigen dat de autosleutel nog aanwezig is. Daarnaast is de keuze gemaakt om de Position_Tracker en de Close_Communication als losse componenten te maken zodat het makkelijker is om de hardware hierachter te kunnen vervangen zonder veel aanpassingen in de code van de Car_Module te moeten maken.

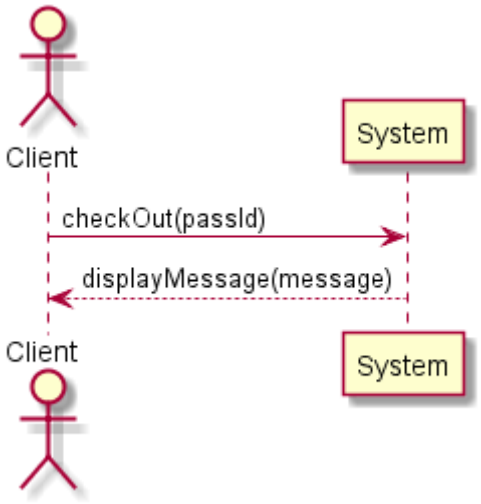


Figuur 4 Component diagram

9. System Sequence Diagrams

9.1. Auto terugbrengen

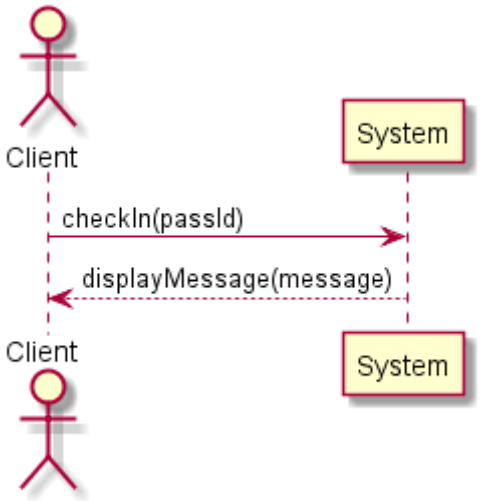
Het auto terugbrengen moet door de klant geïnitieerd worden bij een paal door daar zijn pas tegen te houden. Het systeem handelt dit af en laat dan een bericht aan de gebruiker zien of dat dit gelukt is of niet.



Figuur 5 System sequence: Auto terugbrengen

9.2. Auto ophalen

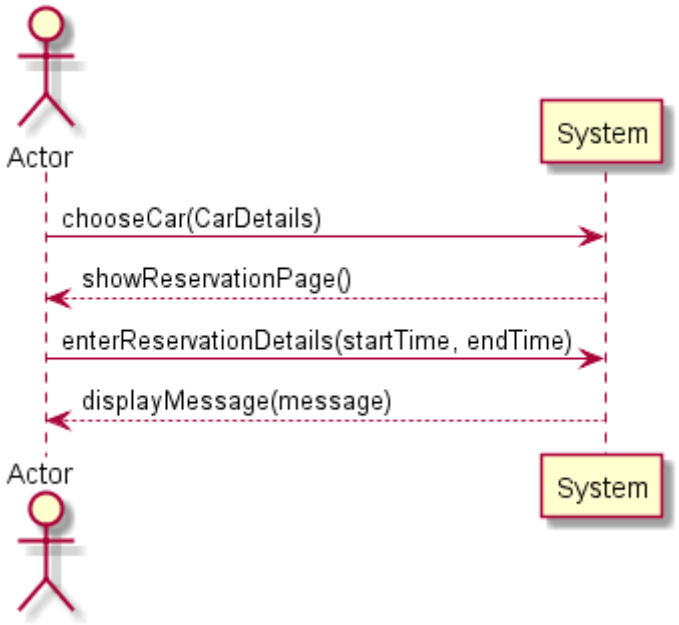
Het ophalen van de auto wordt door de klant geïnitieerd, waarbij hij zijn pas tegen de paal aanhoudt. Het systeem handelt het af en laat zien of de klant is ingechecked of niet.



Figuur 6 System sequence: Auto ophalen

9.3. Auto reserveren

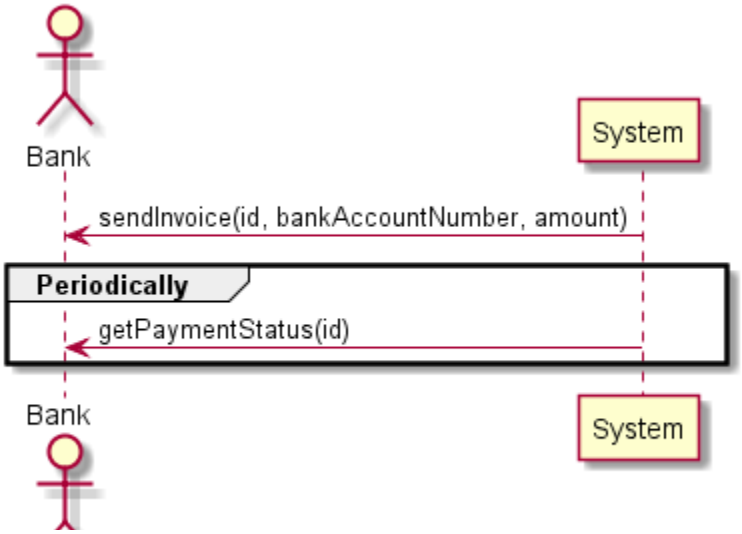
Het reserveren van een auto wordt geïnitieerd met een klant die een auto kiest om te reserveren. Het systeem toont de reservatiepagina van de auto waar de reservatiegegevens ingevuld kunnen worden. De klant vult deze gegevens in en deze gegevens worden dan opgestuurd naar het system. Het systeem toont dan met een bericht of dit succesvol verlopen is.



Figuur 7 System sequence: Auto reserveren

9.4. Huur betalen

Het betalen van de huur wordt geïnitieerd door het systeem zelf. Hierbij wordt dan een incasso naar de bank gestuurd die betaald moet worden. Hierna zal het systeem periodiek opvragen wat de statussen zijn van de betaling

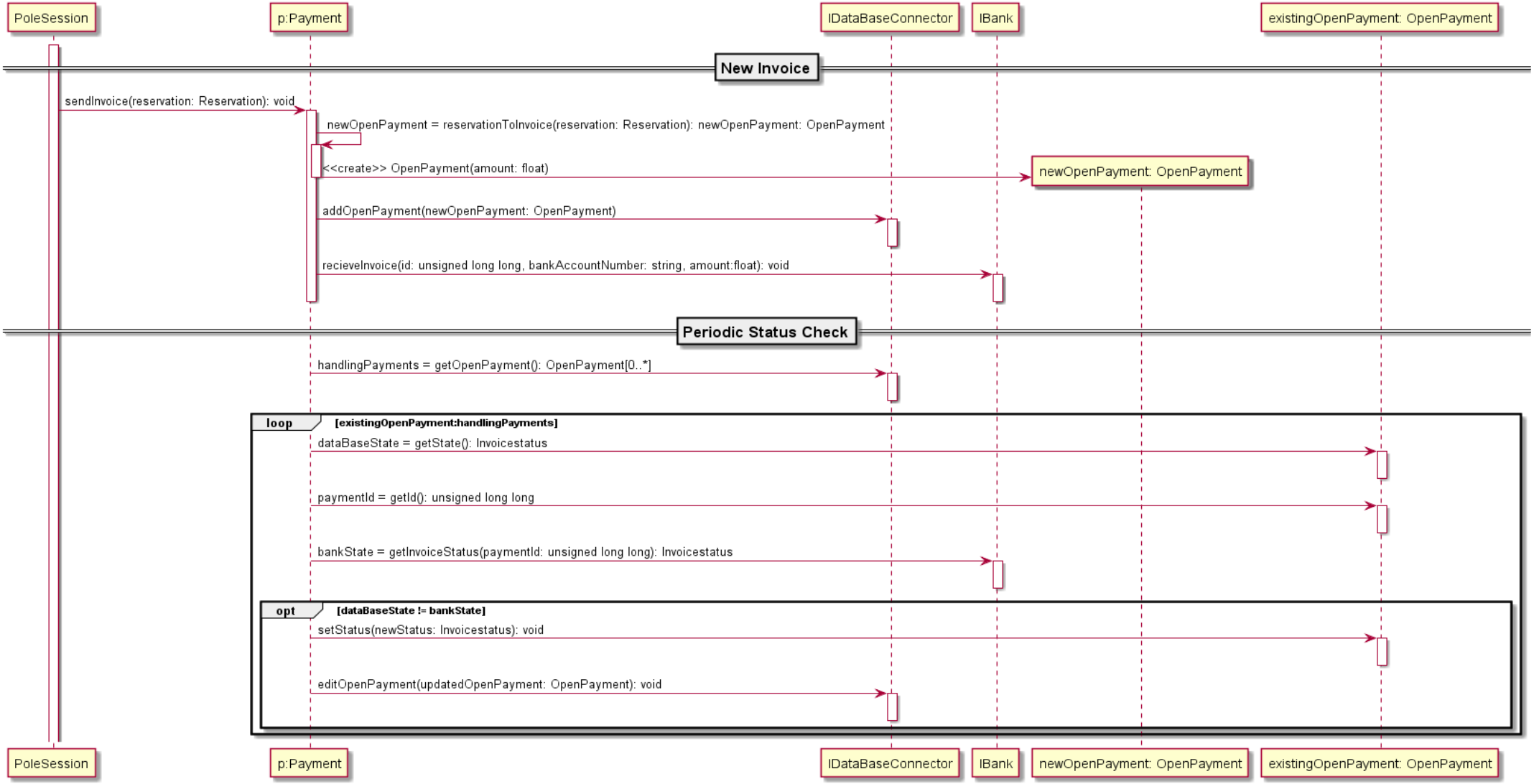


Figuur 8 System Sequence: Huur betalen

10. Sequence diagrams

10.1. Betaling huur

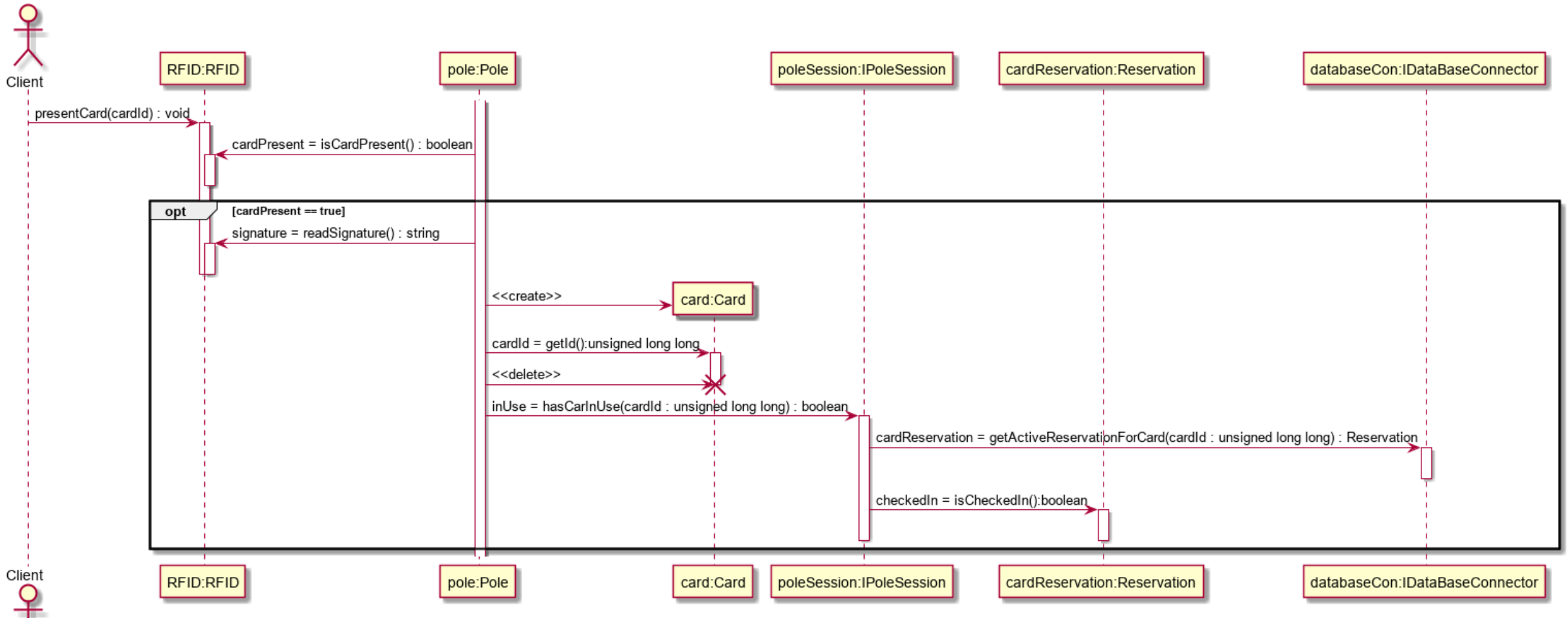
De betaling van de huur begint bij een PoleSession. De Polesession legt de verantwoordelijkheid om een incasso af te handelen bij het Payment object. Het Payment object is toegankelijk voor elke PoleSession en vertaalt de afgeronde reservatie naar een OpenPayment. Deze OpenPayment wordt toegevoegd aan de database, hiermee wordt bijgehouden wat de status is van de betaling en dat de gebruiker nog een openstaande betaling heeft. Dat laatste wordt gebruikt om een klant ervan te letten dat hij/zij een nieuwe reservatie kan maken. Het Payment object zal naast de taak van het maken van nieuwe betaling ook verantwoordelijk zijn voor het bijhouden van de status van de betalingen in de database. Dit wordt gedaan door bij de bank, via het bank interface, op te vragen wat de status is van de betaling en deze, indien nodig, aan te passen in de eigen database. Wanneer een betaling dan voltooid is kan een klant weer een nieuwe reservering maken.



Figuur 9 Sequence: Betaling huur

10.2. Paal interactie

Het ophalen en terugbrengen van de auto begint bij de klant, namelijk wanneer deze zijn pas tegen de paal aan houdt. De paal zelf controleert steeds of er een kaart aanwezig is. Als een kaart gedetecteerd wordt, wordt deze uitgelezen. Van de kaart wordt een signature uitgelezen, uit deze signature wordt dan een cardId gehaald. Hierna wordt met deze cardId opgehaald of de bijbehorende gebruiker een auto in gebruik of gereserveerd heeft.

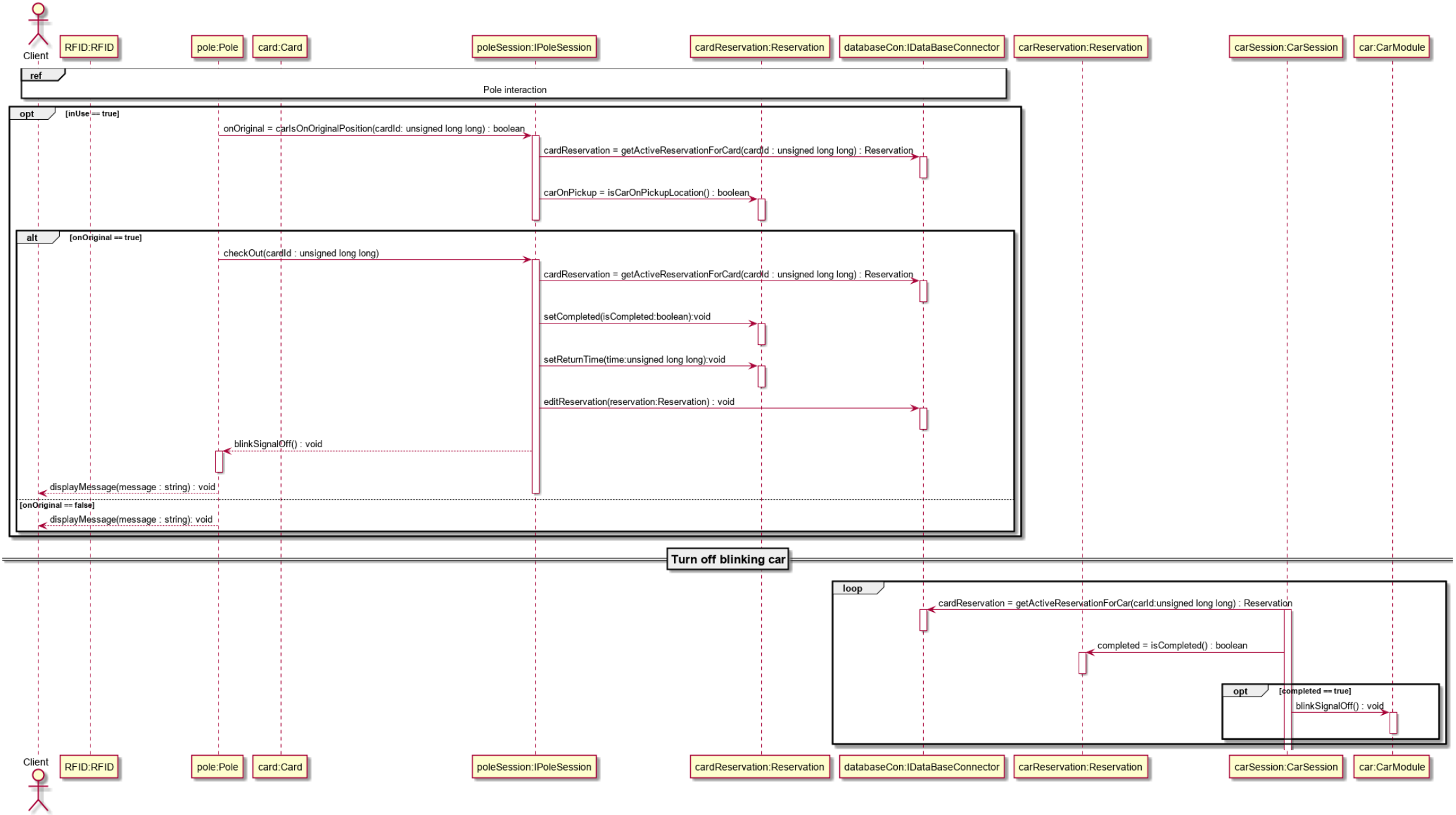


Figuur 10 Sequence: Paal interactie

10.3. Auto terugbrengen TODO Diagram

De interactie begint zoals beschreven hierboven bij “paal interactie”. Als de gebruiker een auto in gebruik heeft dan wordt gecontroleerd of deze auto op dezelfde positie staat als toen deze opgehaald werd. Als de auto op dezelfde positie staat wordt de gebruiker uitgecheckt en wordt het knippersignaal van de paal uitgezet. Hiernaast wordt de reservatie aangepast waardoor het knippersignaal van de auto uit gaat. Na deze sequentie gaat het systeem door met de use case “Betaling huur”.

(Voor diagram zie volgende pagina)

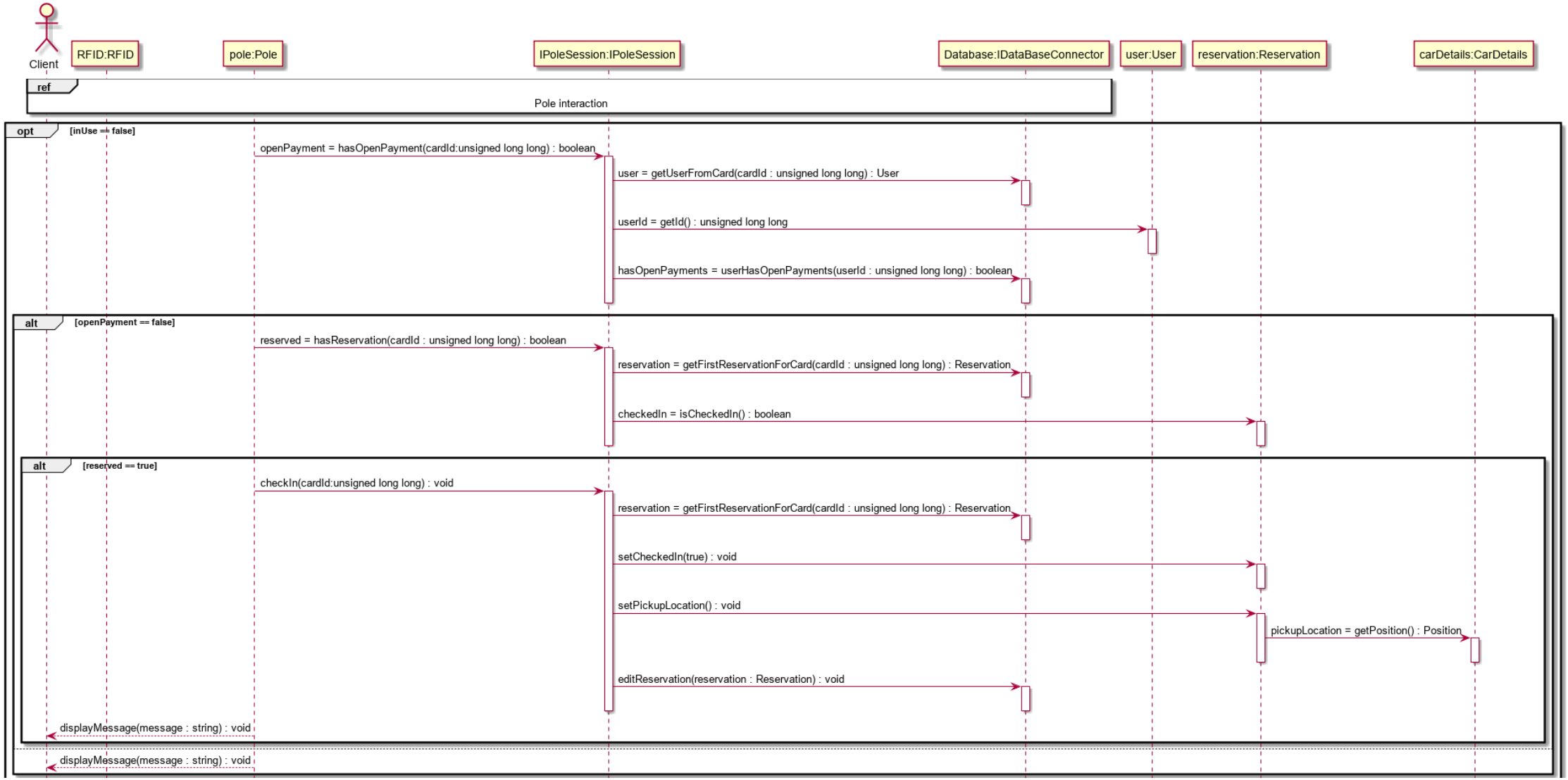


Figuur 11 Sequence: Auto terugbrengen

10.4. Auto ophalen TODO Diagram

Deze interactie begint met de acties zoals beschreven bij “paal interactie”. Als de gebruiker geen auto in gebruik heeft, wordt er bij de database opgevraagd of de gebruiker geen openstaande betaling heeft. Als deze gebruiker geen openstaande betalingen heeft wordt opgevraagd of deze gebruiker een reservatie heeft waarvan de starttijd verstreken is en de eindtijd nog niet bereikt is. Als deze checks allemaal voldoen wordt de gebruiker ingecheckt en wordt de reservatie aangepast in de database.

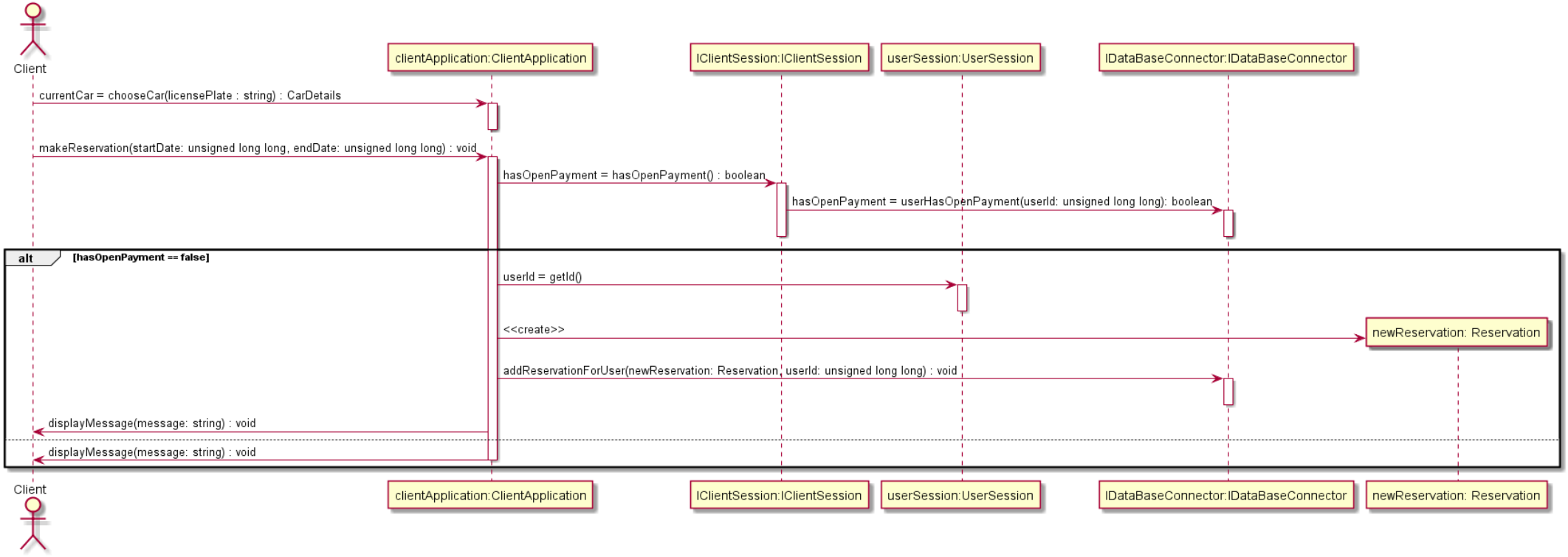
(Voor diagram zie volgende pagina)



Figuur 12 Sequence: Auto ophalen

10.5. Auto reserveren

Het reserveren begint op het moment dat een klant een auto kiest. Als er een auto is gekozen kan de gebruiker de start- en eindtijd opgeven en wordt dit doorgegeven aan het systeem. Hier wordt dan eerst gecontroleerd of deze gebruiker een openstaande betaling heeft. Als hij een openstaande betaling heeft dan wordt een foutmelding teruggegeven. Anders wordt een reservering voor deze gebruiker aangemaakt waaraan dan ook de auto wordt toegevoegd die gekozen is. Deze gegevens worden dan aan de database toegevoegd.

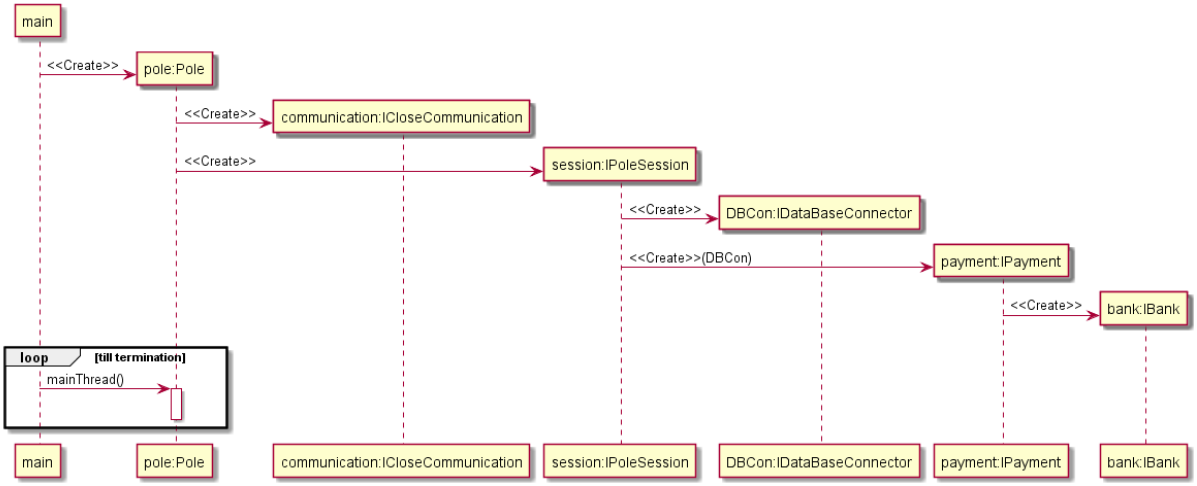


Figuur 13 Sequence: Auto reserveren

11. Creation Sequence Diagrams

11.1. Pole creation

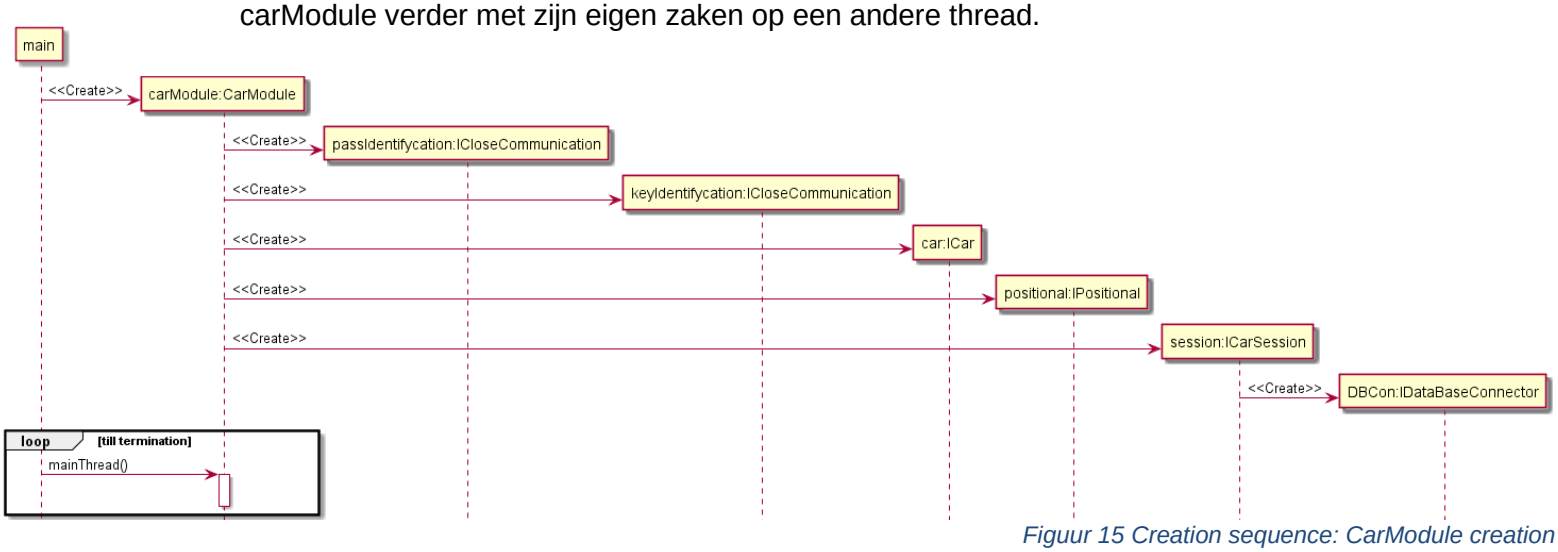
Vanuit de main wordt een Pole (pole) instantie gemaakt, die pole maakt een ICloseCommunication (communication) instantie aan en een IPoleSession (session) instantie. Die session maakt een IDataBaseConnector (DBCon) instantie en een IPayment (payment) instantie aan, bij het aanmaken van payment wordt de DBCon meegegeven. En payment maakt een IBank (bank) instantie aan. Nadat alles is aangemaakt gaat de pole verder met zijn eigen zaken op een andere thread.



Figuur 14 Creation sequence: Pole creation

11.2. CarModule creation

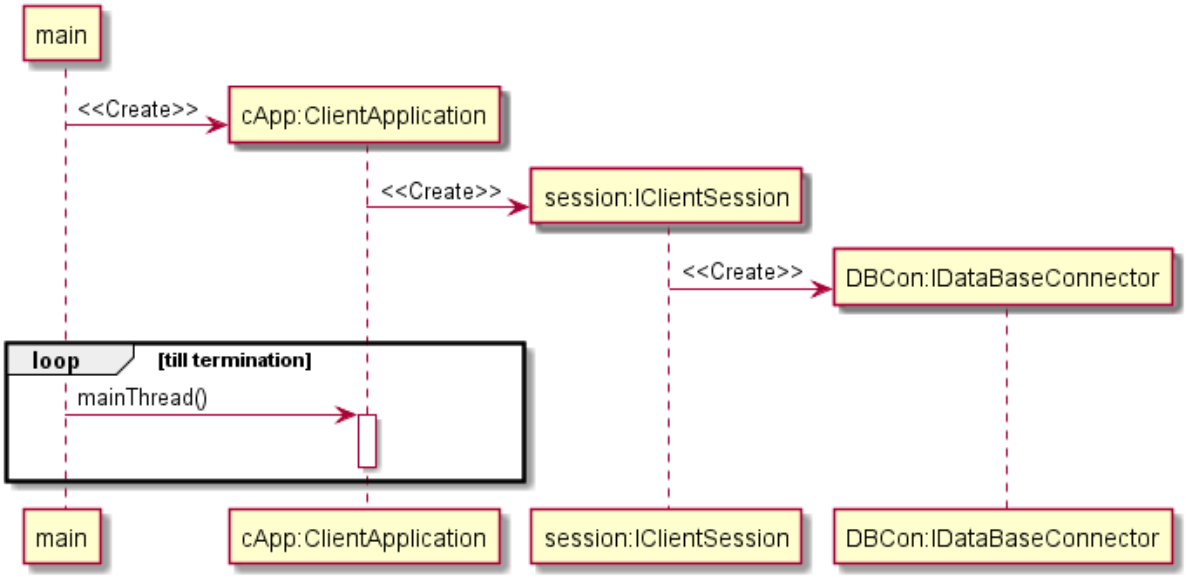
Vanuit de main wordt een CarModule (carModule) instantie gemaakt, die carModule maakt twee ICloseCommunication (passIdentifycation en keyIdentifycation). CarModule maakt ook een ICar (car) instantie en een IPositional (positional) instantie aan. Ook maakt carModule een ICarSession (session) aan waarbij die session een IDataBaseConnector (DBCon) instantie aanmaakt. Nadat alles is aangemaakt gaat de carModule verder met zijn eigen zaken op een andere thread.



Figuur 15 Creation sequence: CarModule creation

11.3. ClientApplication creation

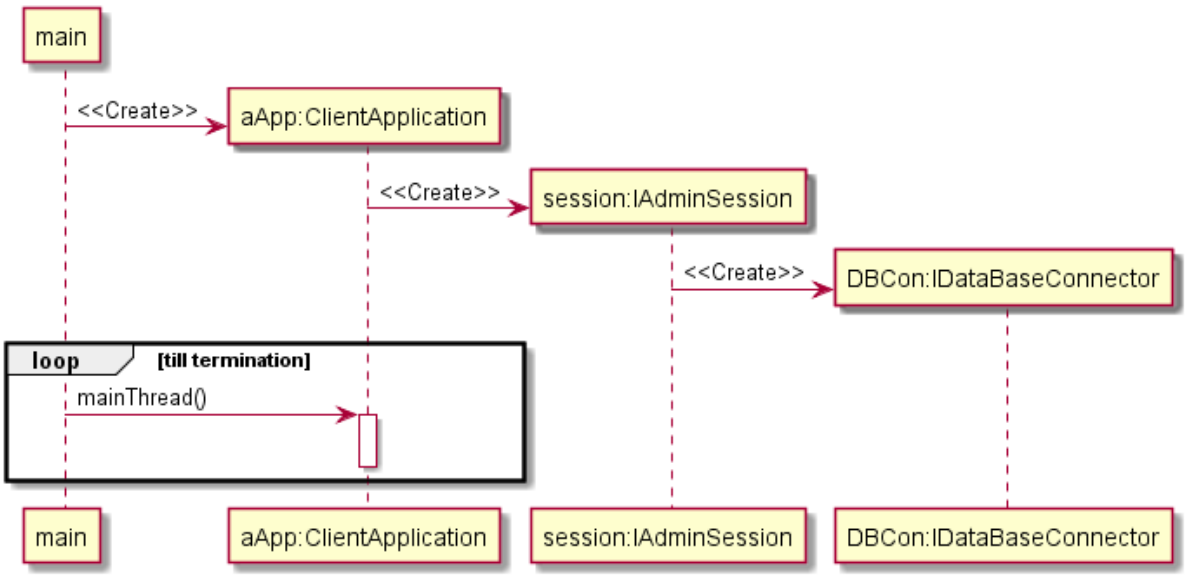
Vanuit de main wordt een ClientApplication (cApp) instantie gemaakt, die cApp maakt een IClientSession (session) instantie aan. Die session maakt een IDataBaseConnector (DBCon) instantie aan. Nadat alles is aangemaakt gaat de cApp verder met zijn eigen zaken op een andere thread.



Figuur 16 Creation sequence: ClientApplication creation

11.4. AdminApplication creation

Vanuit de main wordt een AdminApplication (aApp) instantie gemaakt, die aApp maakt een IClientSession (session) instantie aan. Die session maakt een IDataBaseConnector (DBCon) instantie aan. Nadat alles is aangemaakt gaat de aApp verder met zijn eigen zaken op een andere thread.

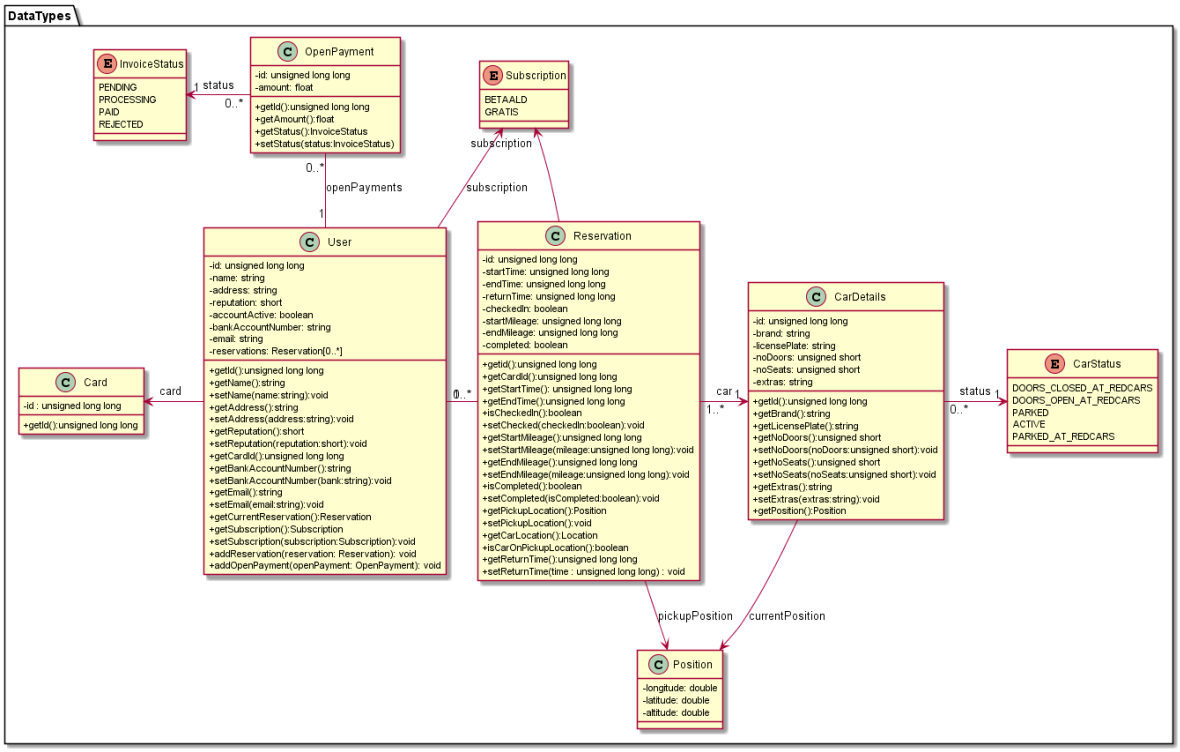


Figuur 17 Creation sequence: AdminApplication creation

12. Class Diagram

12.1. DataTypes

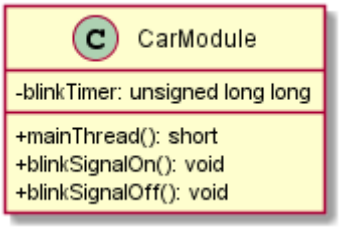
DataTypes is een verzameling van data klassen waarin een groep van gegevens staan die van belang zijn voor dit systeem. Van alle data in een klasse zijn getters aanwezig, maar niet voor alle data is ook een setter aanwezig. Een functie waarbij de data gemanipuleerd kan worden. Hierbij is een keuze gemaakt om een Card als los object van een gebruiker te maken zodat deze bij het uitlezen van een kaart gebruikt kan worden om een signature om te zetten naar een id.



Figuur 18 Class: DataTypes

12.2. Car_Module

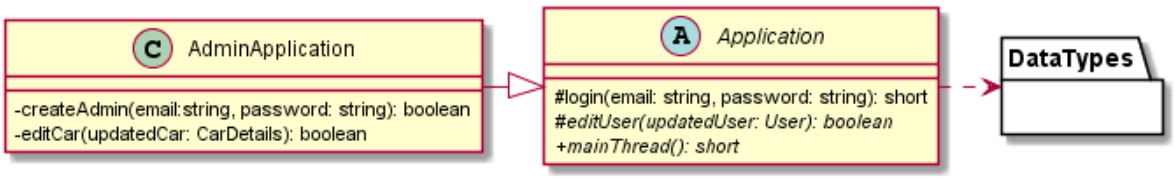
Het Car_Module component bestaat uit een klasse. Deze klasse zal het merendeel van de tijd aan het wachten zijn op een taak. Deze taak wordt vervolgens uitgevoerd met behulp van andere componenten. Zo zal deze klasse een beroep doen op interfaces om zaken in de auto te regelen, passen uit te lezen, of met de database te communiceren.



Figuur 19 Class: Car_Module

12.3. Admin_Application

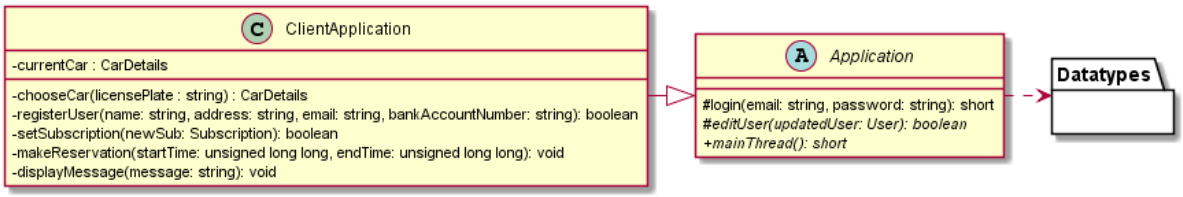
De Admin_Application is het component wat binnen het RedCars systeem gebruikt zal worden door de werknemers van RedCars om voornamelijk gegevens van klanten en auto's te wijzigen. Om dit te doen maakt dit component gebruik van de standaard functionaliteit welke beschikbaar worden gesteld via "Application" en breidt hij deze uit met functionaliteiten die alleen voor de administrators beschikbaar zijn.



Figuur 20 Class: Admin_Application

12.4. Client_Application

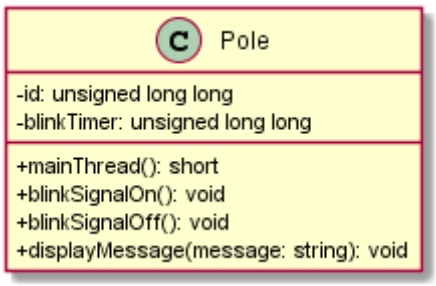
De Client_Application maakt net als de Admin_Application gebruik van standaard functionaliteiten, welke worden aangeboden door de Application klasse. Daarnaast wordt dit weer uitgebreid met functionaliteiten waardoor klanten bijvoorbeeld een reservering kunnen maken.



Figuur 21 Class: Client_Application

12.5. Check_In_Point

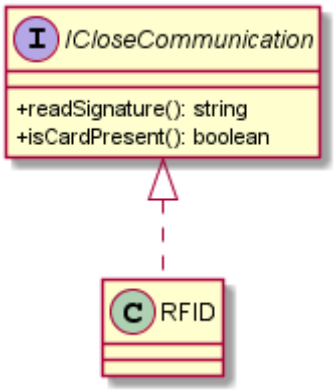
Het Check_In_Point component bestaat uit een klasse, deze klasse wacht binnen de mainThread op een gebruiker om actief te worden. Afhankelijk van of een gebruiker in- of uitcheckt zal deze klasse gebruik maken van interfaces om veranderingen in de database te initiëren.



Figuur 22 Class: Check_In_Point

12.6. Close_Communication

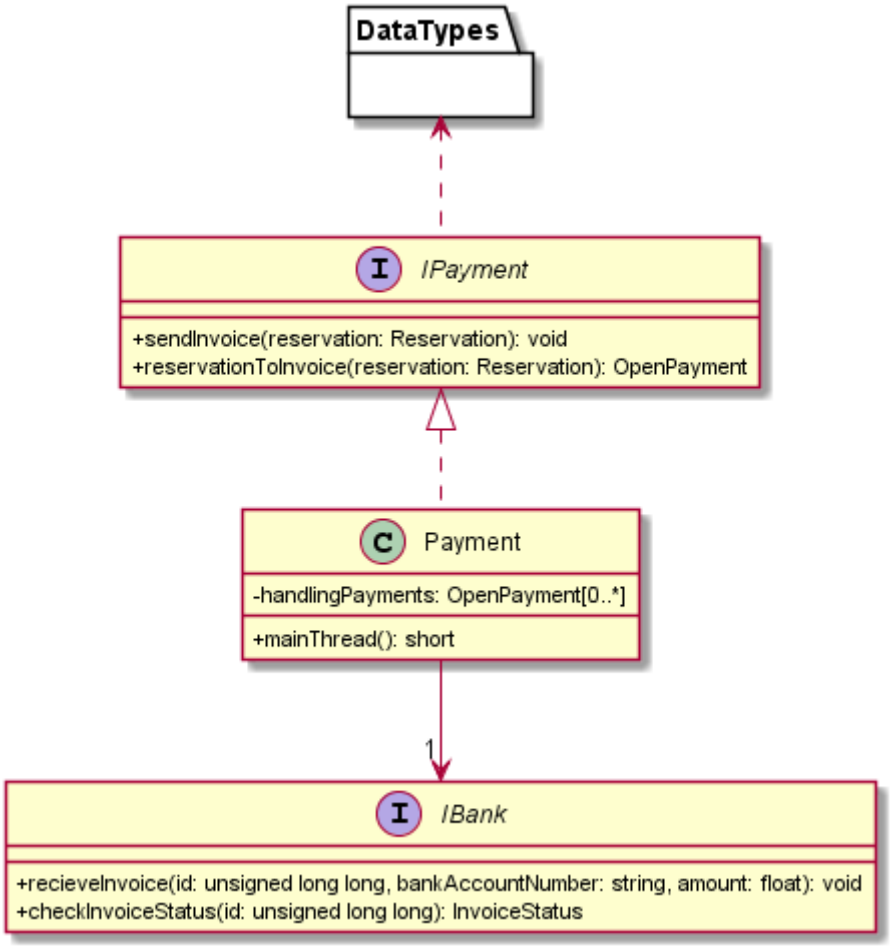
Het component wat de pas gaat uitlezen in zowel de auto als de paal bestaat uit twee klasses. Van deze klasses zal een ervan een interface aanbieden aan een paal of een auto module. De andere klasse biedt de implementatie van deze interface, in ons ontwerp is hiervoor RFID als mogelijkheid gegeven.



Figuur 23 Class: Close_Communication

12.7. Payment

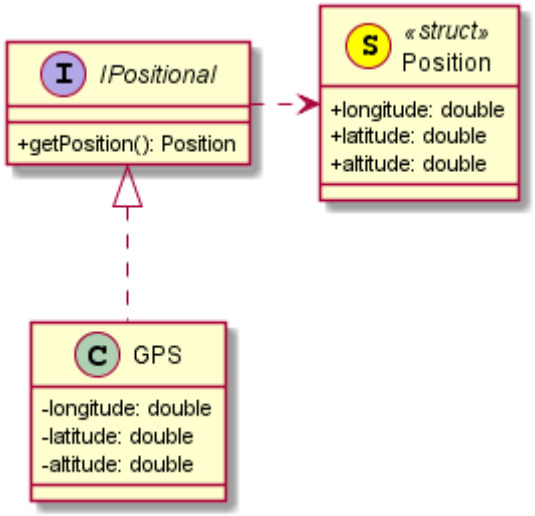
Het Payment component zal, op verzoek van een Check_In_Point verantwoordelijk zijn voor het afhandelen van openstaande betaling en voor het bijhouden van de status van deze betalingen in de database. Het Payment object wordt benaderd via zijn interface en maakt zodoende nieuwe openstaande betaling en voegt deze toe aan de database. Wanneer een betaling nog niet is afgerond maakt het object gebruik van een bank interface om op te vragen of de status van een betaling is veranderd, zo ja kan deze bijgewerkt worden in de database. De bank zelf zal verantwoordelijk zijn voor het afhandelen van ontvangen betalingen.



Figuur 24 Class: Payment

12.8. Position_Tracker

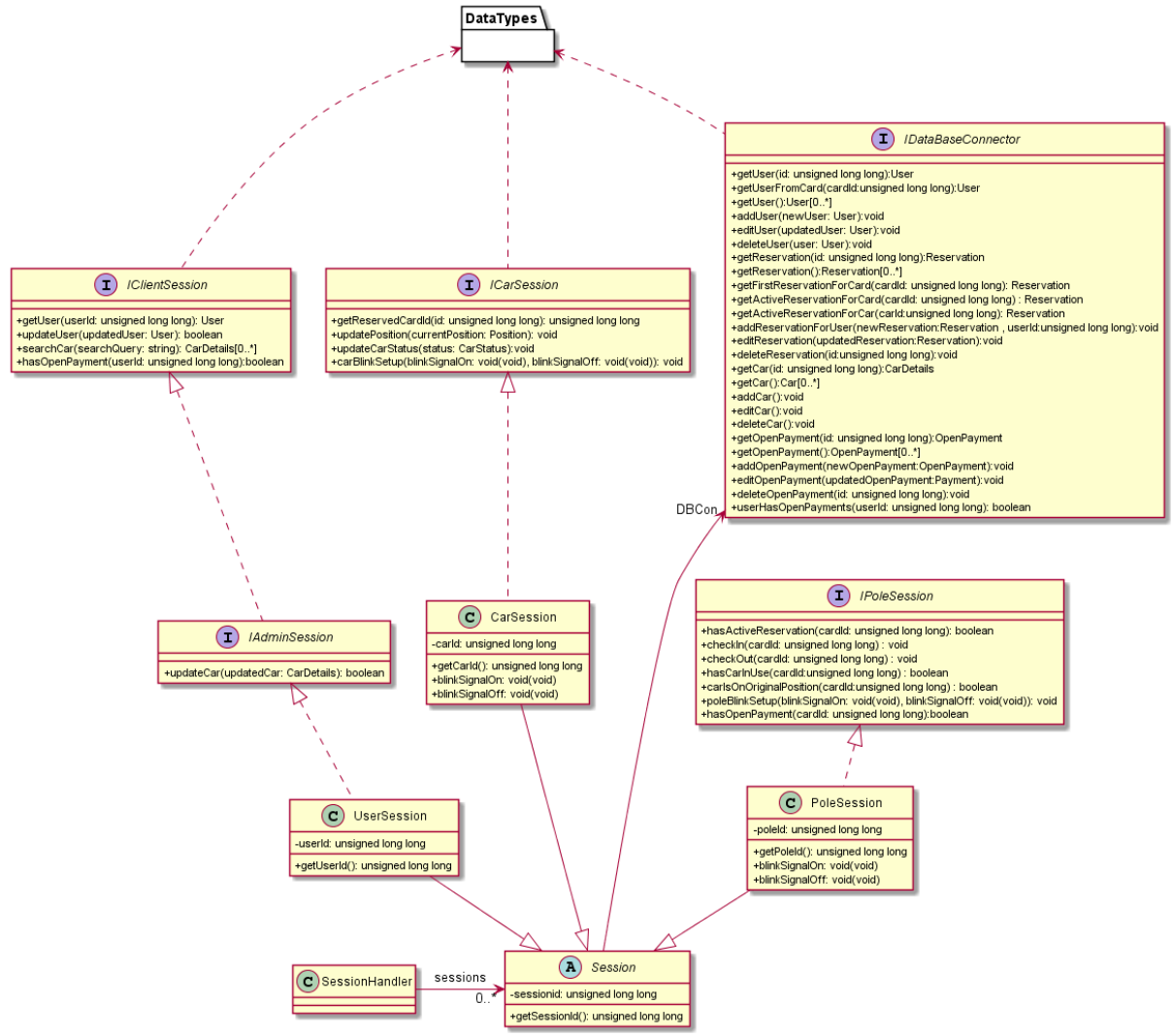
De Position_Tracker biedt een interface aan aan de auto module waarmee de auto zijn locatie kan bepalen. Binnen ons model is hier GPS als implementatie gegeven. De verkregen positie zal door gecommuniceerd worden naar andere componenten in de vorm van een simpele struct.



Figuur 25 Class: Position_Tracker

12.9. Database_Connector

De Database_Connector is verantwoordelijk voor het afhandelen van alle binnenkomende gegevens uit andere componenten in het systeem en verstrekt ook informatie daar waar nodig. Dit wordt gedaan door aan elke gebruiker van dit component ook een sessie toe te wijden. Zo is het niet noodzakelijk voor componenten om zich steeds te identificeren bij de DataBaseConnector. Elk component moet echter andere mutaties uitvoeren op de database en zodoende is heeft elk component een eigen implementatie van sessie om deze functionaliteit te beschrijven. De Database_Connector moet aangezien hij gegevens doorspeelt tussen componenten ook weet hebben van de verschillende classes welke gebruikt worden als verzameling van data.



Figuur 26 Class: DataBase_Connector