**User Stories**

Als gebruiker wil ik een zoekscherm, zodat ik kan invullen waar ik heen wil vliegen en met hoeveel personen ik wil vliegen.

Als gebruiker wil ik een keuzescherm, zodat ik kan kiezen op welk tijdstip ik wil vliegen en kan kiezen met welke maatschappij wil vliegen.

Als gebruiker wil ik een verwerkingsportaal, zodat ik kan zien of er stoelen beschikbaar zijn voor de gekozen vlucht.

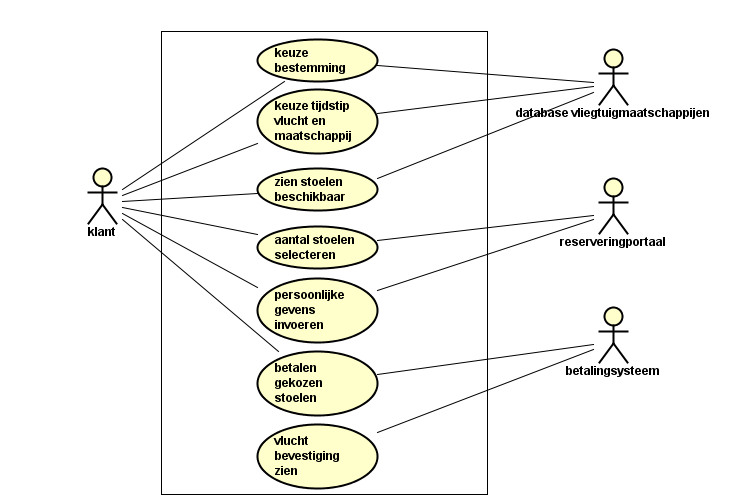
Als gebruiker wil ik een het aantal stoelen kan kiezen, zodat ik deze stoelen kan bestellen.

Als gebruiker wil ik mijn persoonlijke gegevens zoals paspoort enzovoorts kunnen invoeren, zodat ik kan opgeven wie ik en mijn eventuele medepassagiers zijn.

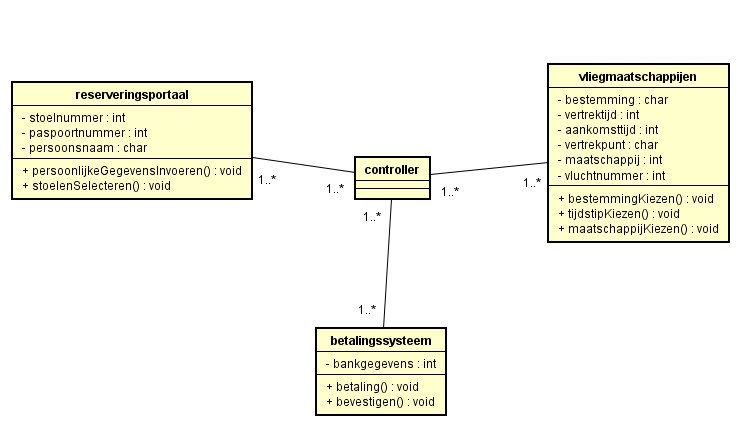
Als gebruiker wil ik mijn gekozen stoelen aan boord van de vlucht kunnen boeken en betalen, zodat ik kan reserveren.

Als gebruiker wil ik een stoel van een vlucht kunnen reserveren als deze vrij is en betaald is, zodat ik de plek aan boord de vlucht gereserveerd heb en de plaats niet meer beschikbaar is.

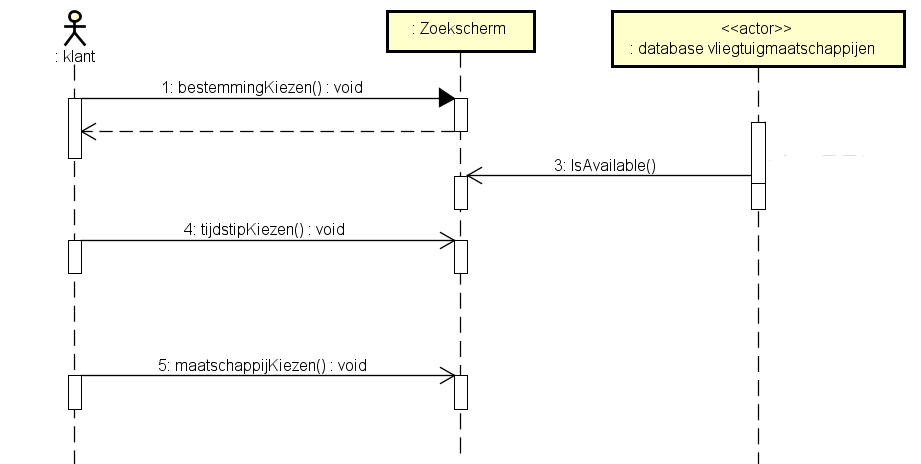
**Use Case Diagram**



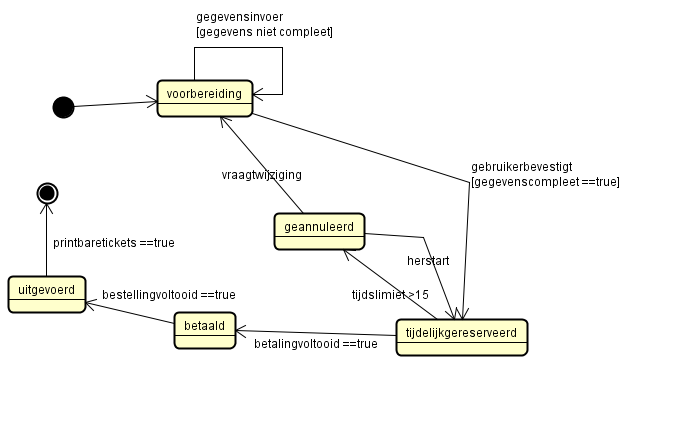
**Klassediagram**

****

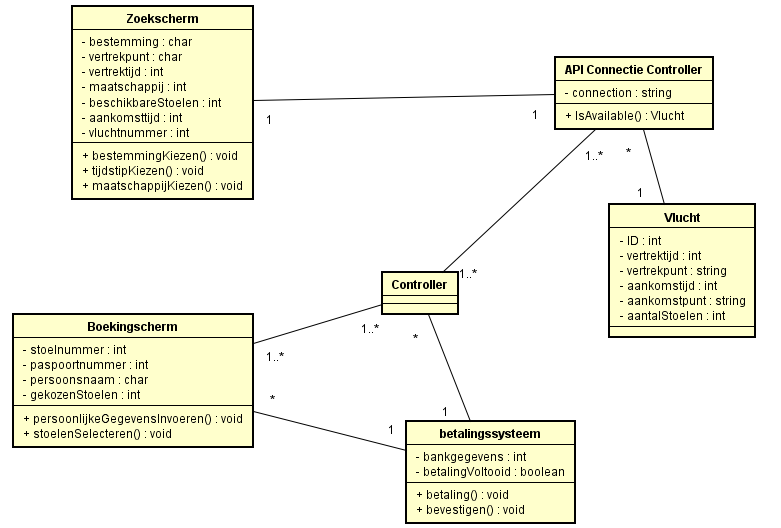
**Sequencediagram**



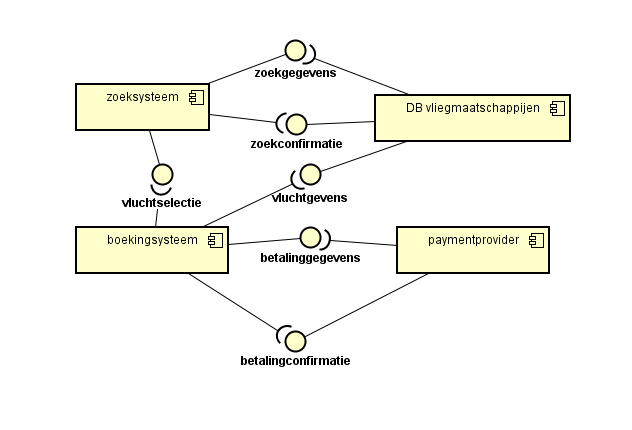
**Toestandsdiagram**

****

**Klassediagram aangevuld vanuit Sequencediagram en Toestandsdiagram**

****

**Componentdiagram**

****

**Controle via Criteria Diktaat**

Controle van het eerste klasse diagram voor de SD en STD laat zien dat niet alle acties gedekt worden. Door middel van logica was bijvoorbeeld de confirmatie terug nog vergeten bij een aantal use cases. Dat werd vooral duidelijk bij het seqiencediagram, want hoe ziet het asynchroon met de weg terug naar het zoekscherm.Het scherm moet iets ontvangen aan attributen uit een method, maar die method was er niet. Een confirmatie. Het is nog zo of al het externe wel gedekt wordt bij de boundary klassen. Want waar zit precies de grens? En moet model klasse Vlucht wel een verbinding hebben? M.a.w. klopt de multipliciteit.

**Gegeneerde code vanuit Klassediagram**

public class Zoekscherm

{

private char bestemming;

private char vertrekpunt;

private int vertrektijd;

private int maatschappij;

private int beschikbareStoelen;

private int aankomsttijd;

private int vluchtnummer;

public void bestemmingKiezen()

{

}

public void tijdstipKiezen()

{

}

public void maatschappijKiezen()

{

}

}

public class Vlucht

{

private int ID;

private int vertrektijd;

private string vertrekpunt;

private int aankomstijd;

private string aankomstpunt;

private int aantalStoelen;

}

public class Boekingscherm

{

private int stoelnummer;

private int paspoortnummer;

private char persoonsnaam;

private int gekozenStoelen;

public void persoonlijkeGegevensInvoeren()

{

}

public void stoelenSelecteren()

{

}

}

public class betalingssysteem

{

private int bankgegevens;

private boolean betalingVoltooid;

public void betaling()

{

}

public void bevestigen()

{

}

}

public class API Connectie Controller

{

private string connection;

public Vlucht IsAvailable()

{

return null;

}

}