Maschen Exercise
Modulopgave 2
2. Semester

Fase 1 - inception

Indledning

I vores modulopgave 2, kunne vi vælge mellem 2 opgaver.

Vi har valgt opgave 1 som er at lave et togsystem som holder styr på vognene som ankommer til Maschen i Hamburg, og sortere dem, så de kan sendes videre.
Her fik vi stillet opgaven at lave tre java programmer der henholdsvis kan;

- Lave toge med op til 10 tilfældige vogne, der ankommer til Maschen
- Splitter togvognene op, så de forskellige togvogne, kommer med de tilhørende fragt-toge
- Lave et program, der bruger en database, til at holde styr på hver togvogn (hvilken type gods fragtes og hvor de befinder sig henne og er den i brug)

Vores rapport bliver udviklet som en UP-proces som er agilt(man kan gå frem og tilbage) med hjælp af diverse UML- analyseværktøjer.
Når man arbejder med UP-processen så har man 4 faser man går i gennem:

1.Inception - I denne her fase overvejer man hvad der skal med i projektet, krav og brainstormer.

2.Elaboration - I denne her fase laver man en mere detaljeret plan over projektet, som man ikke altid holder 100% men man laver nogle deadlines man prøver at overholde.

3.Construction - I denne her fase begynder man at kode og laver nye deadlines.

4.Transition - I denne her fase, finder man en deadline for hvornår projektet skal udleveres til kunden og får systemet etableret hos kunden.

Vi har valgt at brugt Trello-programmet som kanban-værktøj, som hjælper os til at have et overblik over hvor vi er henne i vores proces i vores projekt. Rollen som projektleder varierer under forløbet.

Krav

Vores togsystem skal have et administrationsprogram der skal styre alle vognene, derfor skal vores system kunne varetage sig af CRUD-elementer.(Create, Read, Update, og Delete) Det vi er kommet frem til er noget vi selv har fundet på, på grund af vi ikke har været i kontakt med nogle kunde, for normalt sidder man sammen med sin kunde og kommer frem til de krav ens system skal kunne håndtere.

- C: oprette vogne
- R: få informationer om vogne
- U: sortér vogne

- D: slette vogne

Glossary: Console betyder det der kommer frem på skærmen
Vores metoder returnerer en værdi (0=korrekt, 1=fejl) alt efter om metoden er kørt uden fejl.

FURPS+

Stikordene af akronymet *FURPS+* kan beskrive hvad systemet der udvikles skal kunne.
Functionality: Systemet viser, i en console menu, hvad der er status over vognene der findes i Maschen, som skal sorteres og sendes videre fra Hamburg.
Usability: Console menuen viser status over vogne i Maschen. Brugeren behøver/kan ikke foretage sig nogle beslutninger ud over at lukke programmet.
Reliability: Al data angående vognene gemmes i en MySQL database.
Performance: Systemet kører over tre separate programmer der hvert kører asynkront af hinanden som kan give en smule mere performance.
Supportability: Et krav til at køre programmet er at man har Java, JVM og en version af MySQL kørende til at behandle data som de individuelle programmer bruger.
+: Systemet der udvikles ses som et velgørenhedsprojekt der frigives uden binding mellem nogle specifikke parter. Af denne årsag har projektet GNU General Public License v3 licens, som hovedsageligt giver frihed til at bruge programmet som ønsket, og samtidig sikrer at videredistribuering af ændrede versioner, har samme licens og dermed også sikrer at brugere af dette også har samme rettigheder og friheder.

Fase 2 - elaboration

Use cases

Oversigt over use cases:
UC1: Oprette vogne
UC2: Få info om vogne
UC3: Slette vogne
UC4: Sortér vogne

Vi har valgt at lave én fully dressed use case, og de andre bliver beskrevet som brief use cases.

UC1

Navn: Oprette vogne
Preconditions: At vognene er er i systemet
Main scenario: Aktøren går ind i systemet og får programmet til at genere alle vognene. Konsollen viser vognenes destination, vægt og deres rute.

UC2
Navn: Få info om vogne
Preconditions: Aktøren har alle de oplysninger han skal bruge at få se status på vognene.
Main scenario: Aktøren ønsker at få status/informationer om vognene. Aktøren går ind i systemet under view og der har man så mulighed for at tjekke status på hændelsen.

UC3
Navn: Slette vogne
Preconditions:Vogne er kommet frem til rette destination, og skal kobles fra.
Main scenario:Aktøren går ind i systemet for at få en oversigt over de vognene der er kommet frem til deres destination. Derefter går aktøren ind i systemet og sletter de vogne der ikke er ude og køre mere. Systemet gemmer de opdateringer der er blevet lavet og aktøren lukker programmet igen.

UC4 Fully dressed
Navn: Sortér vogne
Scope: Administrationssystem
Level: Brugerniveau
Primary Actor: Bruger
Stakeholders and Interests: Udviklere, bruger, firma
Preconditions: Der skal genereres vogne før de kan sorteres
Success Guarantee: Vogne sorteres efter destination og uddelegeres til passende fragt-toge
Main Success Scenario:

1. Der genereres op til 10 vogne (use case 1)
2. Vognenes placering og destination læses (use case 2)
3. Vognene sorteres på 3 forskellige spor, efter destination
4. Vognene placeres på nyt spor efter destination, så sidste vogn til afkobling, placeres forrest osv.

Extensions:
1a. Der genereres mindre end 3 vogne

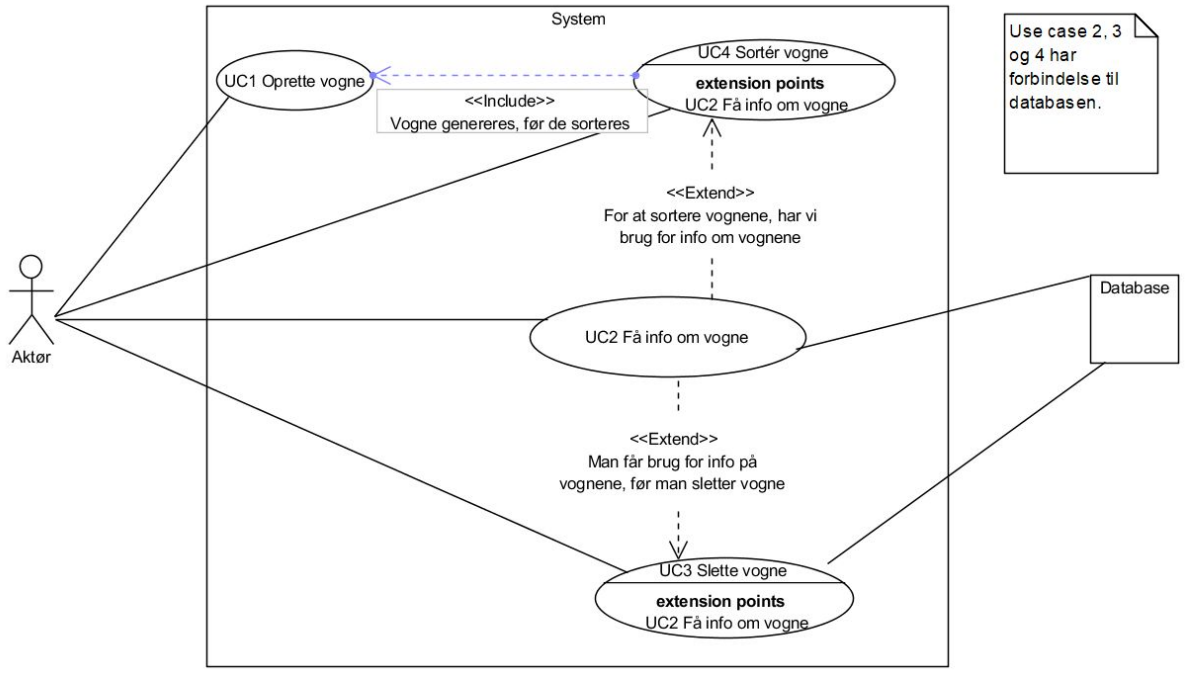
a. Hvis der genereres mindre end 3 vogne, slettes disse, og der genereres nye vogne

1b. Der genereres kun vogne med samme destination
   a. Hvis der kun er én destination på de genererede vogne, slettes disse, og der genereres nye vogne
1c. Der genereres vogne, der ikke kræver sortering
   a. Hvis der genereres vogne, som ikke kræver nogen sortering, slettes disse og der genereres nye vogne

Special Requirements: MySQL Server er påkrævet, til håndtering af databasen

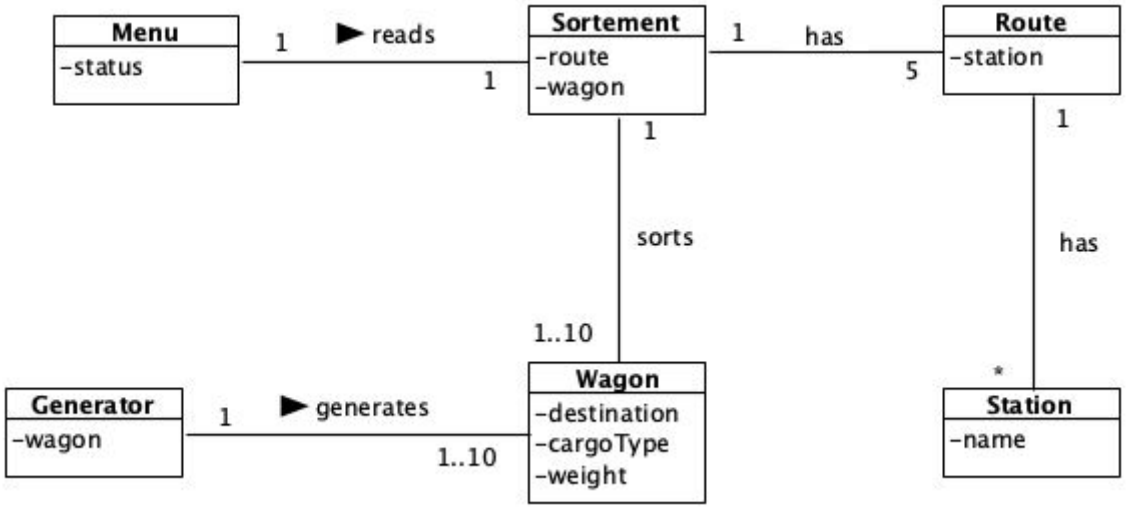
Use Case Diagram

Use Case Diagram visualiserer use cases og viser hvordan de relaterer til hinanden. Dermed visualiserer diagrammet funktionerne i systemet og deres indbyrdes relationer samt hvilken primær aktør er knyttet til hvilken use cases.
I vores tilfæld har vi kun én aktør og det er den bruger som skal betjene systemet. På højre side af systemet finder vi den database der er knyttet til systemet som håndterer alle vores tabeller.
UC1" oprette vogne" er associeret med UC4 "sortér vogne" for vognene generes før de bliver sorteret.
UC3 "slette vogne" har extensions knyttet til UC2 " få info om vogne". Får aktøren skal kunne slette nogle vogne skal han have info fra vores database som tjekker om vognene er i brug eller er kommet frem til deres destination, hvis ja så kan de slettes og hvis ikke så kan de ikke blive slettet og systemet giver besked at de stadig er i brug.
UC4 "sortér vogne" har extensions knyttet til UC2 "få info om vogne", for før aktøren kan sortere vognene skal han vide hvad det er nogle vogne.
UC2, 3 og 4, har forbindelse til databasen.
Vores use case diagram afspejler, på hvilken måde systemet indeholder CRUD-elementerne. Vi kan oprette vogne, læse informationerne om vognene, opdatere vognene og slette vogne.



Domain Model

Domænemodellen viser, hvilken oplysninger vi har brug for i vores system og hvilken forbindelse disse oplysninger har til hinanden. Domænemodellen viser også de use cases der er i vores system og de konceptuelle klasser og deres indbyrdes relationer.
De klasser vi har med viser deres ansvarsfordeling.
Nedenstående er en illustration af vores domænemodel.

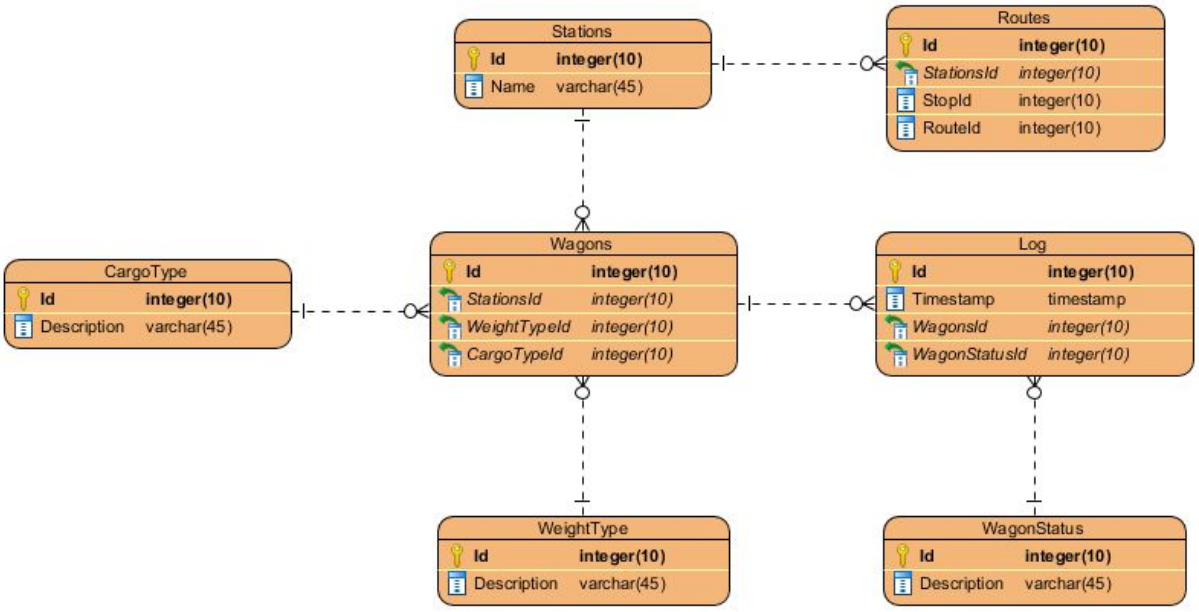


Entity Relationship Diagram

Hvert tabel i databasen har en primary key kaldet ‘Id’, som er sat til auto_increment (automatisk nummerering ved indsætning), med undtagelsen ‘Routes’ som har et særligt ‘Id’ tildelt hvor der er flere rækker i tabellen der beskriver dele af samme rute. Denne tabel har ikke sin primary key sat til auto_increment.

‘StopId’ i ‘Routes’ tabellen refererer til hvilket nummer på den givne rute stationen stopper. Her er foreign key sat til at være ‘Station’ ‘Id’.

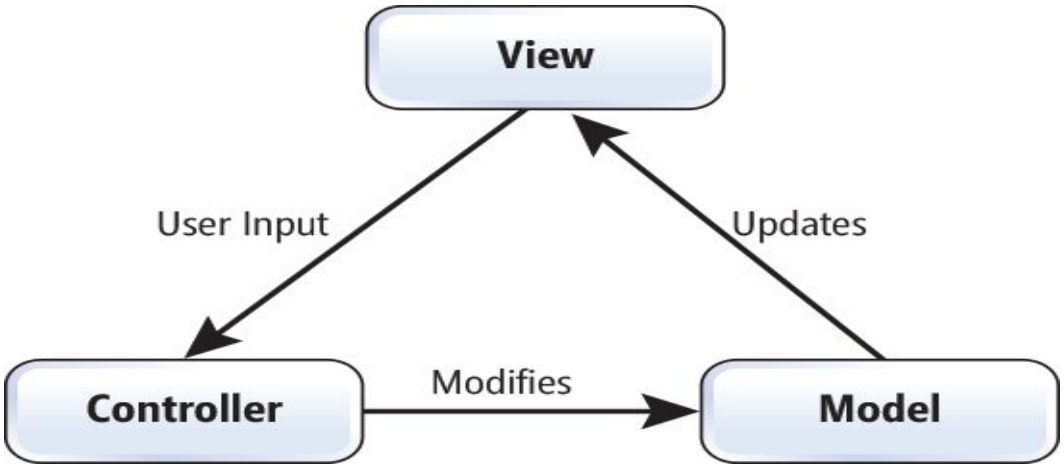
1. normalform: Der er ingen gentagne kolonner i nogle af tabellerne, som gør at første normalform er opfyldt.
2. normalform: Der er ingen delvise afhængigheder, eller sammensatte primære nøgler, som gør at alle tabellerne overholder anden normalform.
3. normalform: Ingen af de givne tabeller har kolonner som er delvis afhængige af hinanden. Dette gør at tredje normalform er opfyldt.



Brug af pattern

Vi har taget brug af MVC pattern i vores system (Model, View og Controller).
MVC viser hvordan ansvaret er fordelt.
Model: Indeholder data strukturen af vores 'knowing' klasser.
View: Sørger for at vise brugergrænsefladen og behandle brugerinput.
Controller: Holder styr på logikken og business delen af programmet. Model klasser manipuleres og sendes til View.

Hver del af MVC har sin egen pakke i vores projekt source filer.



Model
CargoType
Route
Station
Train
Wagon
WagonStatus
WeightType

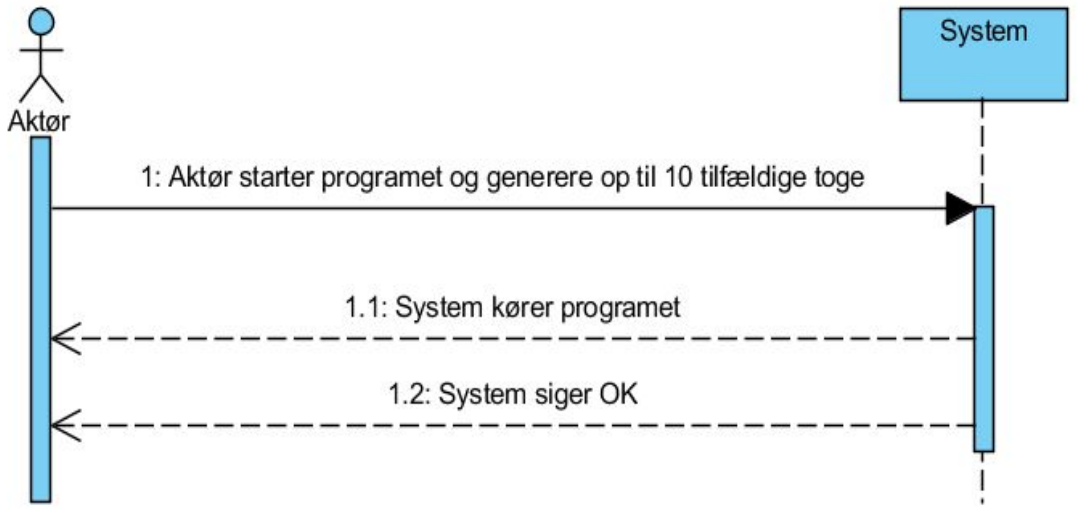
View
Menu

Controller
DatabaseController
SortmentController
WagonGenerator

System Sequence Diagram

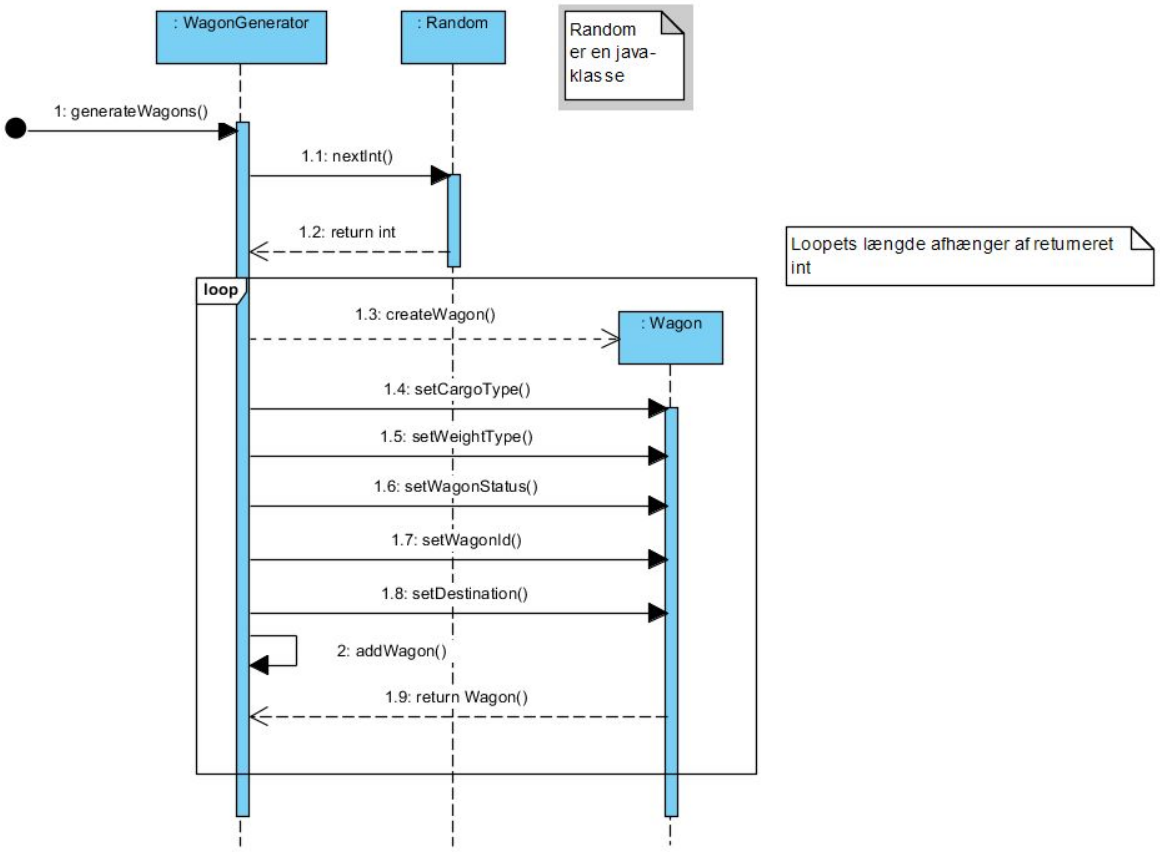
System Sequence Diagram (SSD) er med til at vise et flow mellem aktør og systemet som viser det som et visuelt format.

Man læser det oppefra, aktøren er på den venstr side og systemet til højre.
De linjer der strækker sig imellem dem kaldes aktionslinjer som viser hvordan de kommunikere sammen.
Vores SSD tager udgangspunkt til vores UC1 - oprette vogne.



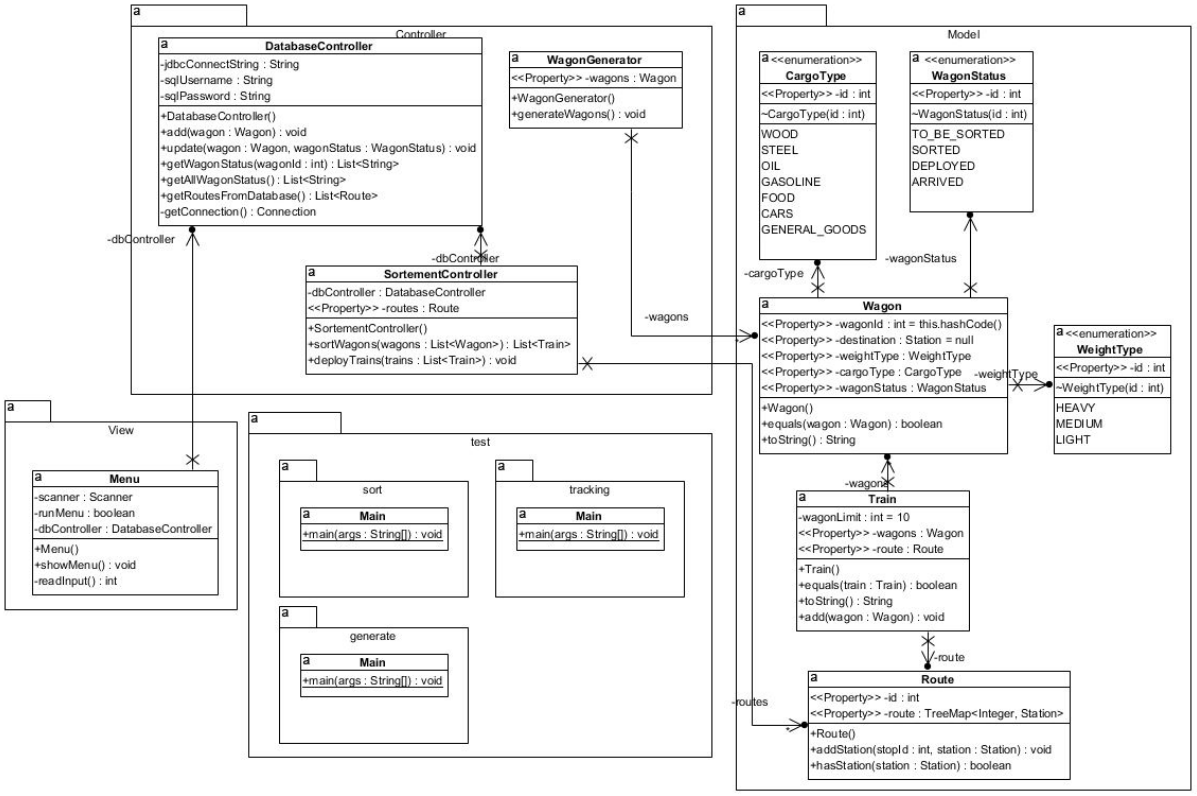
Sequence Diagram

Et Sequence diagram (SD) illustrerer en sekvens af handlinger der sker i et system. Den viser de objekter og klasser der er involveret når en given use case bliver kørt med hvilken rækkefølge metodekald foretages i og hvilken retur-typer.
De parallelle vertikale linjer (livliner), viser hvor længe objektet lever, og de vandrette pile, er meddelelser der bliver udvekslet mellem dem.
Man kan bruge SD til en slags sti for at illustrere hvordan systemet skal respondere og gør det muligt at specificere på en grafisk måde.
Vi har taget udgangspunkt til vores UC01 for at demonstrere hvordan det virker i vores system.



Klassediagram

Klassediagrammet viser hvordan koden ser ud visuel og hvilken attributter og metoder de forskellige klasser har ansvar for. Ved hjælp af de use cases vi er kommet frem til samt domænemodel er vi kommet frem til vores klassediagram som ses nedenstående;



Fase 3 - Construction

(Det er her vi går i gang med at kode vores system)

Fase 4- Transition

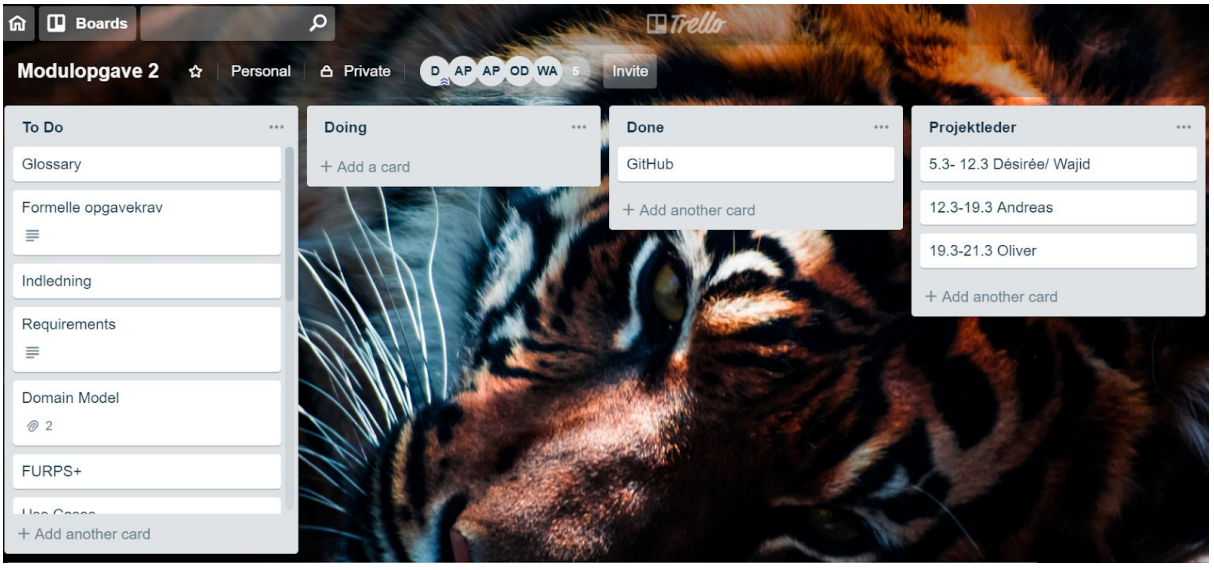
(Det er den sidste fase i UP-processen nemlig overførelse til kunden)

Konklusion

I dette projekt har vi udviklet et administrativt system med 3 java programmer som genererer tilfældige 10 vogne der ankommer til Maschen i Hamborg, samt at udarbejdet hertil knyttede analyser og diagrammer.
Vores gruppe udpeget en projektleder hver uge som skulle via hjælp af Trello hele tiden holde styr på hvor vi var henne i vores proces og hvad vi skulle nå løbet af den uge.(under *milepæl for projektet*- kan I se hvordan det forløb sig.)
Vores projekt blevet udviklet som en UP-proces med de UML- analyseværktøjer. Under den første fase inception - hvor vi i fællesskab fik vi lavet nogle use cases som vi mente at vores system skulle håndtere, men på grund af tidsmangel fik vi desværre ikke etableret UC3 "slette vogne" i vores system.
Den næste fase vi gik ind til var så elaboration, her fik vi styr på alt analysedelen såsom; ERD- der viser hvordan vores tabeller skulle sættes sammen i vores database og fik styr på vores klassediagram inden vi gik i gang med at kode det.
Fase 3 som er construction, er den fase efter vores klassediagram hvor vi gik i gang med vores kode og database.
Fase 4 transition kom vi så ikke til, for det er den fase hvor man overfører programmet til kundens system.

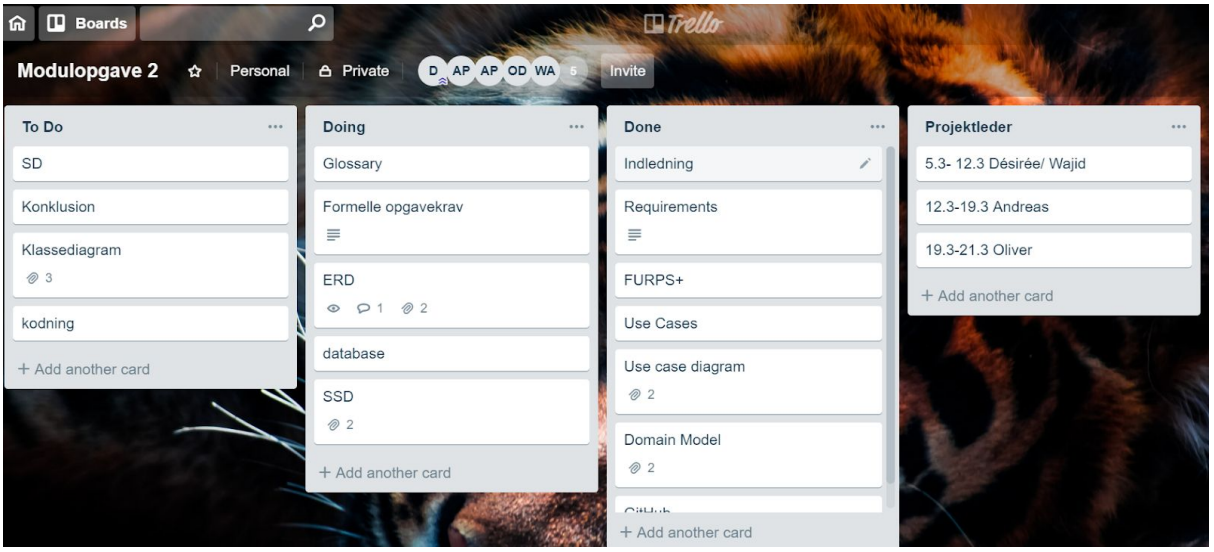
Milepæl for projektet

Uge1- skærmbillede af Trello 08.03.2019



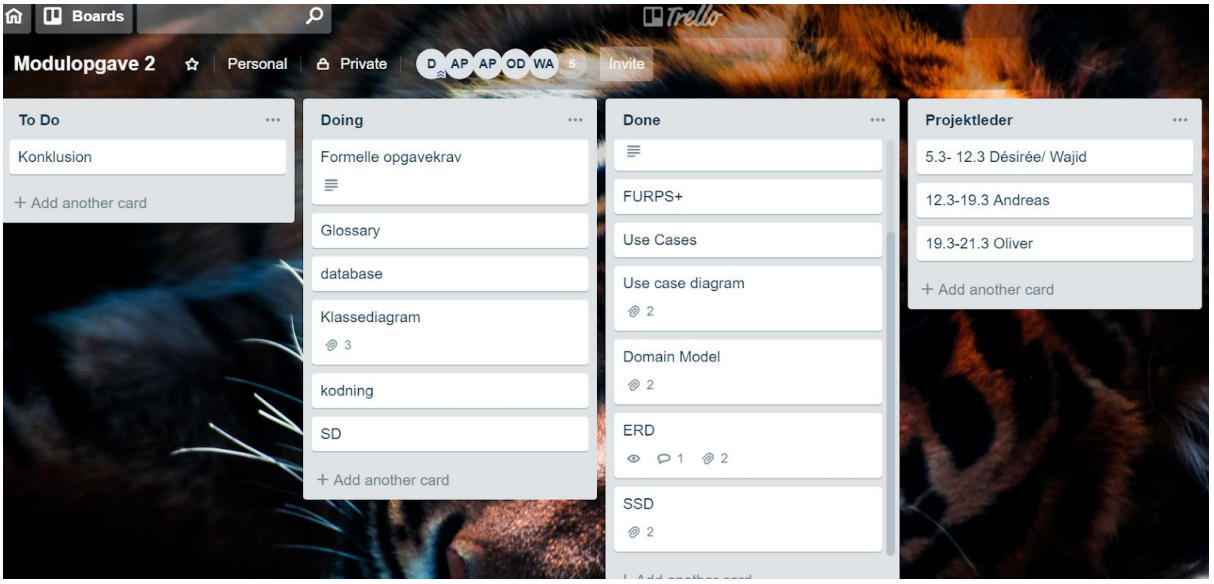
Den første uge fik vi lavet en oversigt i trello over hele vores projekt og hvad der skulle med i vores rapport.
Vi har valgt at vi alle i gruppen skulle være projektleder. I den første uge var det Wajid og Désirée der var projektleder.

Uge 2 - skærmbillede af Trello 15.03.19



Vores 2. uge Var det Andreas som var projektleder. Her var vi godt i gang med rapporten samt med de diverse modeller.
Vores mål for denne uge var at blive færdig med med alt det man skal have med inden man koder.

Uge 3 - skærmbillede af Trello



I vores sidste uge, der gik vi igang med at kigge på vores klassediagram og på koden.
Gennem det her projekt har vi været gode til at holde vores mål og samarbejde.
Det har hjulpet os meget at bruge Trello som værktøj for at har hjulpet os meget til at bevare overblikket og have en fornemmelse med hvor langt vi har i projektet.