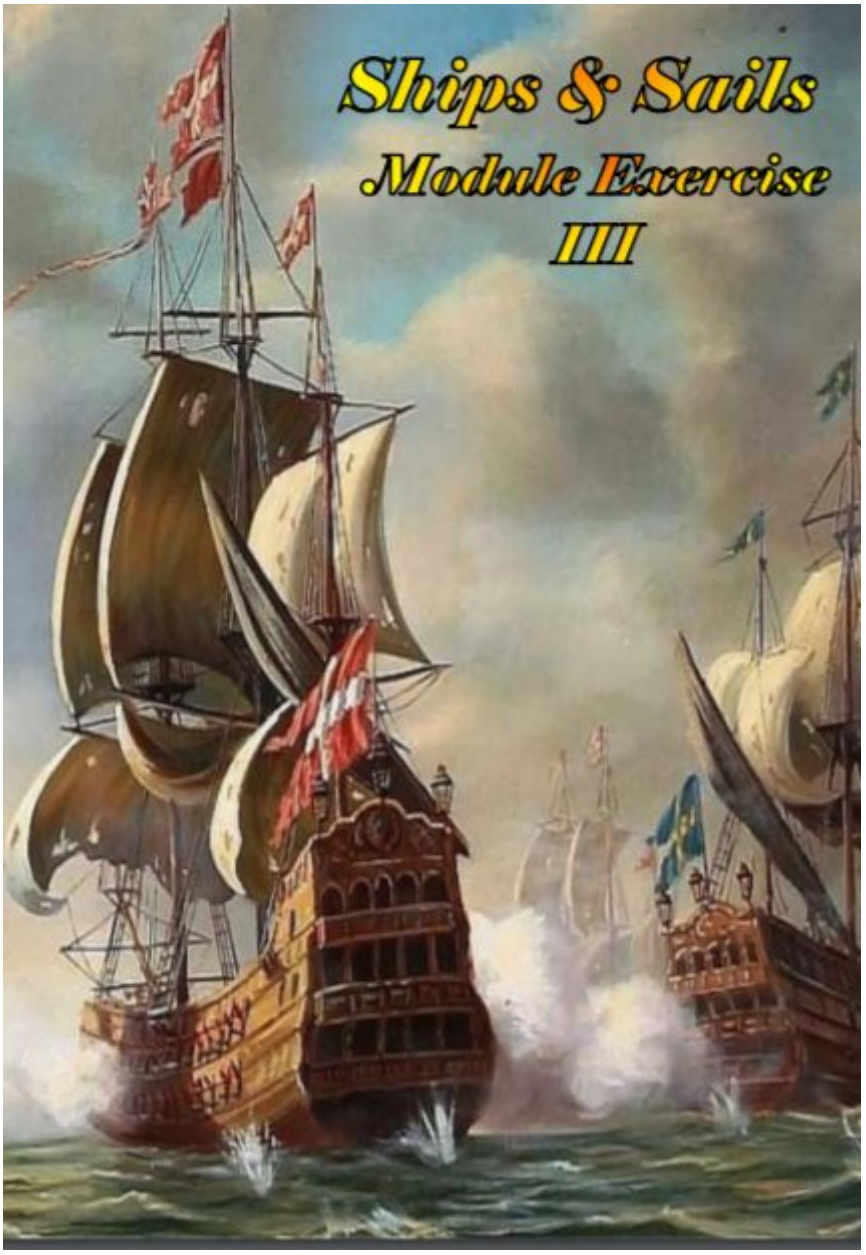


Modulopgave 3

Klasse DAT18C

Udarbejdet af; Andreas Petersen, Daniel, Désirée og Sofie

MODULOPGAVE 3 - SHIPS AND SAILS

Indledning
Ships & Sail er et rundebaseret multiplayerspil med server/klient-forhold, hvor to spillere får lov at udkæmpe søslag mod hinanden på hver deres computer.

Vi har hver især kodet en del af programmet og har dokumenteret vores arbejde i et logsystem, der findes i bilaget. Overordnet har vi inddelt implementeringen af programmet på følgende måde:

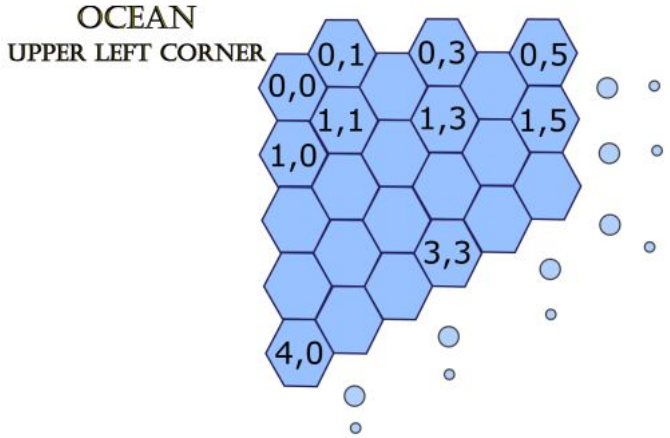
Communication: Daniel
UI: Desirée
Movement: Andreas
Gun: Sofie

Alle diagrammer forefindes også i originale vvp-filer i github.

Krav
Vi skal lave et webbrowserbaseret multiplayerspil (client/server, peer to peer).

Den første spillers computer agerer både server og klient, den anden spiller er blot klient. Både server og klient skal have en relationel database, hvor data om skibe og scenarier er gemt. Databasen kan udvides til at holde data om påbegyndte scenarier.

Bræt
Spillet spilles på et hav, dækket af heksagoner (sekskanter), der fungerer som felter skibe kan bevæge sig på.
Havets størrelse kan variere fra et scenarie til et andet.



Havets størrelse er forudbestemt af spilværten og sejler et skib ud over havets kant, forsvinder det. Der kan kun være ét skib per felt, og hvis to skibe på et givent tidspunkt befinder sig på samme felt, sker der en kollision og begge skiber rykker tilbage til hvor de kom fra og deres fart sættes til 0.

Vind

Over havet blæser en vind, med hastighed og retning, der skifter fra tid til anden. Vind påvirker skibes bevægelsesevne. Det er severen, der fortæller om vejret.

Skibe
Begge spillere kontrollerer et antal skibe af forskellige slags. Der er tre forskellige typer skibe. Alle skibe flytter på én gang.

Kanoner
Skibe har kanoner med forskellige typer ammunition. Kanonerne er ligeligt fordelt på begge sider af skibet (man kan derfor kun affyre halvdelen af kanonerne per affyringstur). Der skal tre mand til at affyre én kanon ligemeget type ammunition. Kanonerne tager én runde for at lade med samme type ammunition og to runder for ny ammunition. Alt efter skibstype kan man sigte forskelligt.

Kanonkugle: ødelægger skroget. Mulighed for at skibet eksploderer! Kort til lang rækkevidde.

Kæde: ødelægger sejlet. Middel rækkevidde.

Metal: dræber mænd. Kort rækkevidde.

Bevægelse
Et skibs fart er afhængig af skibstype, vejret og skrogskade.

Et skibs fart kan gå fra 0 til sin types topfart, set i tabellen nedenfor.
Tag højde for vind og skrogskade.

Type	top fart	max sejl	# besætning / sejl	Sejl <= 50% → fart	Sejl <= 25% → fart	Sejl <= 10% → fart	Sejl = 0% → fart
Brig	2	30	6	1	1	1	0
Ship of the Line	5	60	6	3	2	1	0
Man at War	4	80	6	3	2	1	0

FURPS+

Functional Requirements

Id	Krav	Kommentar
FR001	Systemets backend skal kodes i Java med Spring.	
FR002	Systemets frontend skal laves med html, css og javascript.	Har brugt Javascript - det gav mere mening
FR003	Systemets skal kunne spilles fra en internetbrowser.	http protokol
FR004	Systemet bruger MySQL til at bevare data.	
FR005	Fullstack applikation.	Frontend og backend og datalag ligger i samme applikation.
FR006	Implementering af MVC pattern.	
FR007	Implementering af Peer to peer dataudveksling.	Server/Client kommunikationsservice
FR009	En spiller (Server) skal kunne starte en ny spil-lobby og vente på en modstander. (Et spil har to spillere)	
FR010	En spiller (klient) kan tilslutte sig en spil-lobby (Server), der venter på en modstander og spillet starter.	

Non-Functional Requirements

Id	Krav	Kommentar
NFR001	Spillet bruger 2d-grafik.	
NFR002	Håndter tab af forbindelse	
NFR003	Et spil er rundebaseret og består af 2 spillere og et givent scenarie.	

NFR004	Projektet skal beskyttes med licens.	MIT License https://opensource.org/licenses/MIT

Use cases

Oversigt over use cases:
UC1: Start af spil
UC2: Spille en runde
UC3: Lade
UC4: Skyde
UC5: Beregne skade efter kanon
UC6: Flytte
UC7: Beregne skade efter kollision
UC8: Beregne om man bliver ramt af skud
UC9: Spille spillet

UC1: Start af spil
Precondition: Der er en spiller, der er en server og anden spiller, der er en client. Der er en forbindelse mellem dem.
Main scenario: Spilleren loader et spilscenarie fra databasen. Spilleren starter spillet ved at gå i gang med UC9.

UC2 Fully dressed	**Spille en runde**
Level	Brugerniveau
Scoop	Onlinespil
Primary Actor	Spiller
Stakholders and Interests	Spiller - vil starte et spil, men for at starte spillet skal man have forbindelse til en modspiller. - Modspiller for at kunne spille spillet skal man have en anden spiller og man skal have en forbindelse med server og client. Udvikler- se om spillet kører optimalt. Asger og Jarl - se at vi har forstået opgaven korrekt.
Precondition	UC1

Success Guarantee	Spillet kan spilles, og det bliver gennemført og slutter af med en vinder.
Main Success Scenario	1.Spillerne får vejrforholdene at vide af serveren. 2. Spillerne sætter eventuelle bevægelser, ladninger og angreb op. 3. Serveren og klienten udveksler deres træk (se UC6, UC3, UC4). 4. Dernæst beregnes først eventuelle kollisionsskader (se UC7) og sidenhen eventuelle angrebsskader (se UC5). 5.Trækkene vises grafisk. 6. Spillerne udveksler oplysninger om tilbageværende skibe. 7. Spillet viser en eventuel vinder.
Extensions	*a. Spilleren afgiver en ulovlig ordre 1. Spilleren sætter sin ordre op igen. *b. Spillet tager ikke imod ens ordre. 1. Spilleren angiver ordren igen.

UC3: Lade
Precondition: Spilleren har minimum tre mænd tilbage.
Main scenario: Spilleren vælger først, hvilke skibe, der skal lade. Der skal tre mænd per kanon. Dernæst vælger spilleren type af ammunition. Ladningen tager én runde med samme ammunition.
Extension:
Valg af ny type ammunition: ladningning tager to runder.

UC4: Skyde
Precondition: UC3 er sket, og spilleren har minimum tre mænd tilbage.
Main scenario: Spilleren vælger først hvilke skibe, der skal skyde. Spilleren sigter på et punkt alt efter skibstype og ammunitionstype. Skibet affyrer ammunition, hvilket tager én runde.

UC5: Beregne skade efter kanon
Precondition: UC8 er sand.
Main scenario: Spilleren får at vide, hvilken type ammunition skibet er ramt af og med hvilken sandsynlighed og om det er kritisk og antallet af kanoner, der blev affyret (firing value). Derefter beregnes skaden på den relevante del af skibet. Dette kan have indflydelse på UC3, UC4, UC6 og UC2 alt efter ammunition.

Extension:

Skrogkvalitet <= 0: skibet synker.

Kritisk ramt af kanon: skibet synker med en sandsynlighed på 1/20

Antal sejl <= 0: skibet kan ikke flytte sig.

Antal mænd <= skibet kan ikke lade eller skyde.

UC6: Flytte

Main scenario: Spilleren vælger først hvilke skibe, der skal flytte sig. Dernæst vælger spilleren retning, drejning og fart på skibene. Skibene flytter sig, hvilket tager én runde.

Extension:

Vind: vinden kommer i fronten → skibet kan ikke flytte sig frem.

Vind: vinden kommer fra frontsiderne → skibet flytter sig muligvis langsommere alt afhængig af farten og maksfarten.

Kollision: Kolliderer skibet med andre skibe, ryger skibet tilbage til udgangspunktet for runden, farten sættes til 0 og skade beregnes (se UC7).

UC7 : Beregne skade efter kollision

Precondition: to eller flere skibe er stødt sammen på ét felt (se UC2).

Main scenario: Skibet tager skade i skroget afhængig af hvor mange skibe, der kolliderer. Skaden beregnes ud fra ⅓ af skrogkvaliteten på alle andre skibe. Skibenes skrogkvalitet er altid den kvalitet, skibet havde før kollision.

Extension:

Skrogkvalitet <= 0: skibet synker.

UC8: Beregne om man bliver ramt af skud

Main scenario: Skibet sejler igennem en krigszone, der består af 7 felter, et 40% risikofelt i midten og 6 10% risikofelter rundt om. Skibet kan sejle over flere af felterne i løbet af en tur. Der slås med en 10-sidet terning for at udregne sandsynligheden for at blive ramt.

Extension:

Skibet rammes på 40% risikofeltet: der rulles en 20-siders terning for at finde ud af, om skibet bliver ramt kritisk.

UC9: Spille spillet

Main scenario: Spilleren udfører UC2 igen og igen, indtil der er fundet en vinder. Vinderen findes ved, at der enten ikke er nok mænd tilbage til at skyde eller sejle, eller der ikke er flere skibe tilbage. Spillet kan også slutte efter x antal runder bestemt af spilscenariet, og vinderen er i så tilfælde spilleren med mindst skadede skibe.

Extension:

Uafgjort: Hvis begge spillers sidste skib går ned i samme runde, bliver spillet uafgjort.

Use case diagram

UC1 " Start af spil" Både modstander og spiller skal være tilkoblet for at spillet kan spilles.
UC2 "Spille en runde" er tilkoblet til spilleren og er associeret med use cases: UC3,UC6,UC7 og UC8 for de kan kun blive udført når man starter spillet og spiller det.
Spilleren har kun forbindelse til UC1 "Start af spil" og UC9 "Spille spillet".
UC5 " Beregne skade efter kanon" har en extend-relation til UC8 "Beregne om man bliver ramt af skud" for UC5 er kun aktuelt hvis man bliver ramt er kanon.
UC9 "Spille spillet" er associeret med UC2 "Spille en runde"
Nedenstående en oversigt over vores use case diagram.

Glossary

Ord	Beskrivelse

Hull quality	Et skib har et skrog med et forudbestemt antal livspoint.
Sail quality	Et skib har N antal sejl med forudbestemt antal livspoint.
Brig, Ship of the Line, Man at War	Navne på de skibe der er med i spillet.
Kanonkugle, Kæde, Metal	Typer ammunition til kanoner. Beskrevet i afsnittet “Kanoner”.

Risikoanalyse
Risikoanalysen omhandler projektet og hvilke elementer der er i fare for at udgøre en risiko mod projektet og dets overholdelse af tidsplanen.

S = Sandsynlighed for at det der kan gå galt, faktisk sker.
K = Konsekvensens grad for hvor alvorligt det er, det der går galt.
R = Risikotal som samlet vurdering

Sandsynligheds grader:
- 1: Meget lav
- 2: Lav
- 3: Moderat
- 4: Høj
- 5: Meget høj

Konsekvens grader:
- 1: Næsten ubetydeligt
- 3: Meget lidt vigtig
- 5: Vigtig
- 7: Mindre alvorlig
- 9: Meget alvorlig

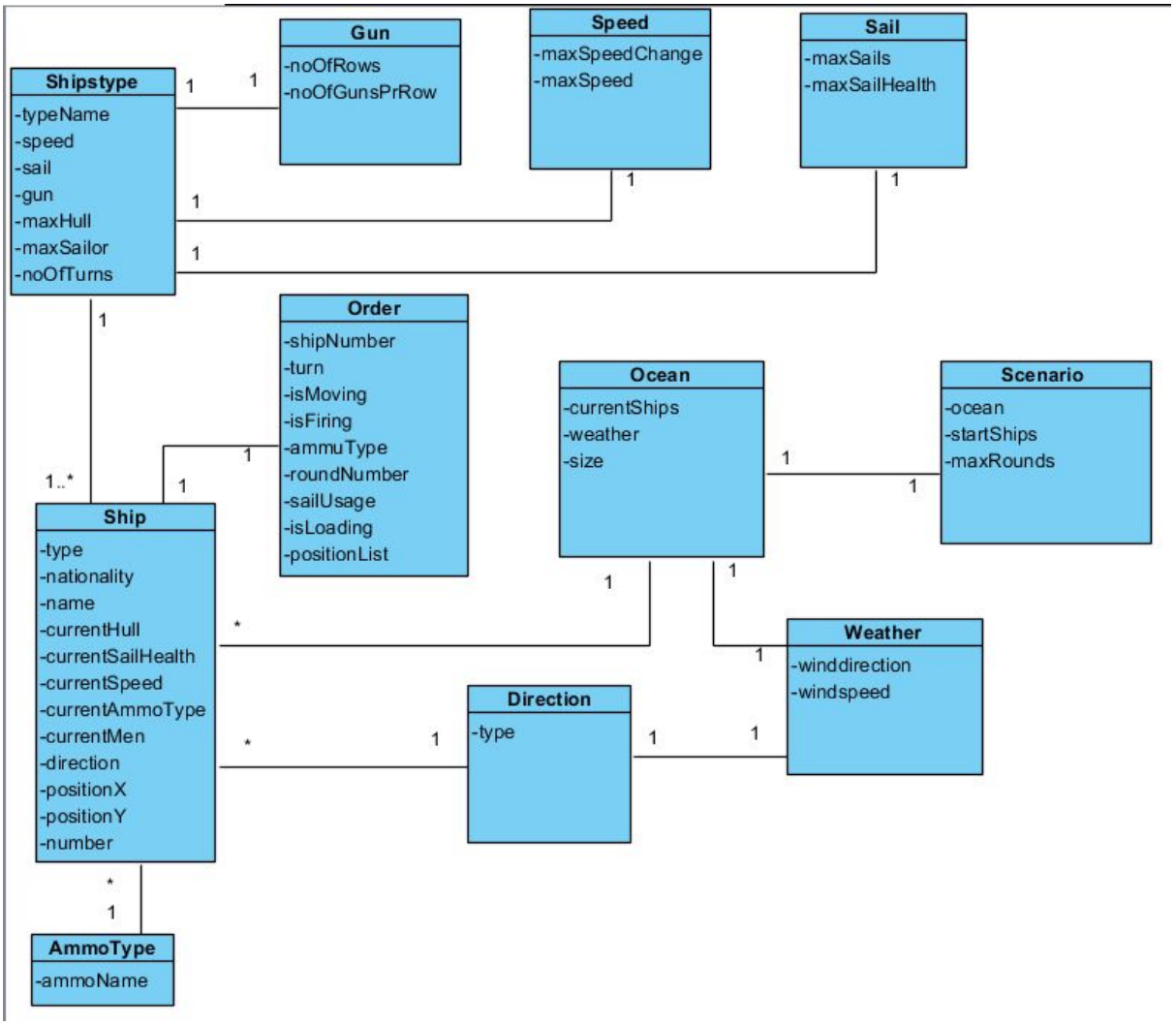
Projekt : Modulopgave 3 - Ships and sails		
Hvad kan gå galt?	**Risiko** ***S* x *K* = *R***	**Tiltag der reducerer sandsynligheden**
Et gruppemedlem bliver syg, eller på anden måde ude af stand til at bidrage.	3 x 5 = 15	Mere kommunikation omkring projektets status og medlemmernes helbred. Forventningerne til tidsplanen justeres og

		der afgrænses muligvis yderligere.
Udvikler forstår ikke den opgave de er blevet stillet- forstår ikke kundens ønske ud fra det materiale de har fået udleveret	5 x 5 = 25	At man får en bedre beskrivelse af kunden hvad det er lige præcis hvad kunden ønsker sig af givne opgave.
Udvikler kan ikke blive enige/forståelsen af opgaven de er blevet stillet.	2 x 9 = 18	Sætter sig ned og gennemgå opgaven sammen - og diskuterer hvad de hver især har forstået ud fra opgaven de er blevet stillet.
Der er samarbejdsvanskeligheder i gruppen. Man ender i tidspres og man kan måske ikke overholde den deadline man har fået.	2 x 7 = 14	Man kan lave en intern kontrakt hvor alle bliver enige om fremgangsmåden fremadrettet.
Kunden kommer med modstridende krav og ønsker til programmet.	5 x 9 = 45	Det er vigtigt at udrede kundens krav og ønsker fra starten og derfor kan udviklerne være nødt til at stille mange enslydende spørgsmål til kunden indtil udviklerne er tilfredse med svaret.
Der er uenighed internt hos kunden om projektet.	5 x 5 = 25	Det er vigtigt at udrede kundens krav og ønsker fra starten og derfor kan udviklerne være nødt til at stille mange enslydende spørgsmål til kunden indtil udviklerne er tilfredse med svaret.
Udviklerne har fået en for kort deadline.	4 x 9 = 36	Der kan enten ansættes flere udviklere eller skæres i kravene. I denne opgave kan der kun skæres i kravene.

Modellerne

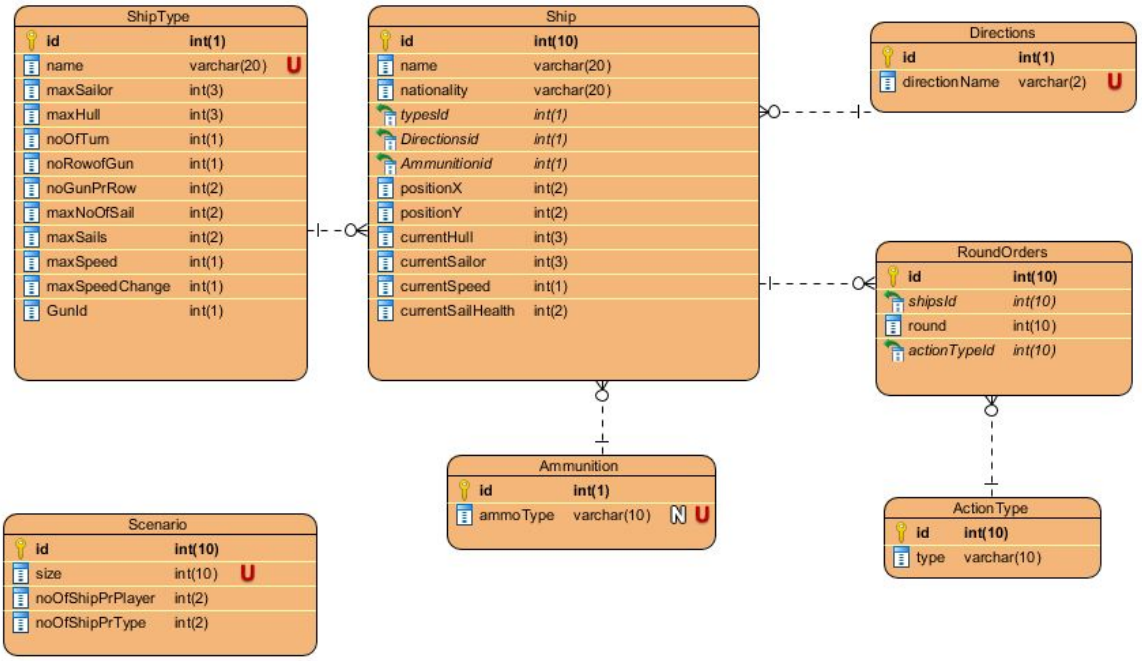
Domænemodel

Nedenstående er et billede af vores domænemodel. Det er denne model, vi har bygget vores design op om og det er hvad vi forsøger at arbejde os hen imod. Dette er vores konceptuelle idé af de forskellige spilelementer.



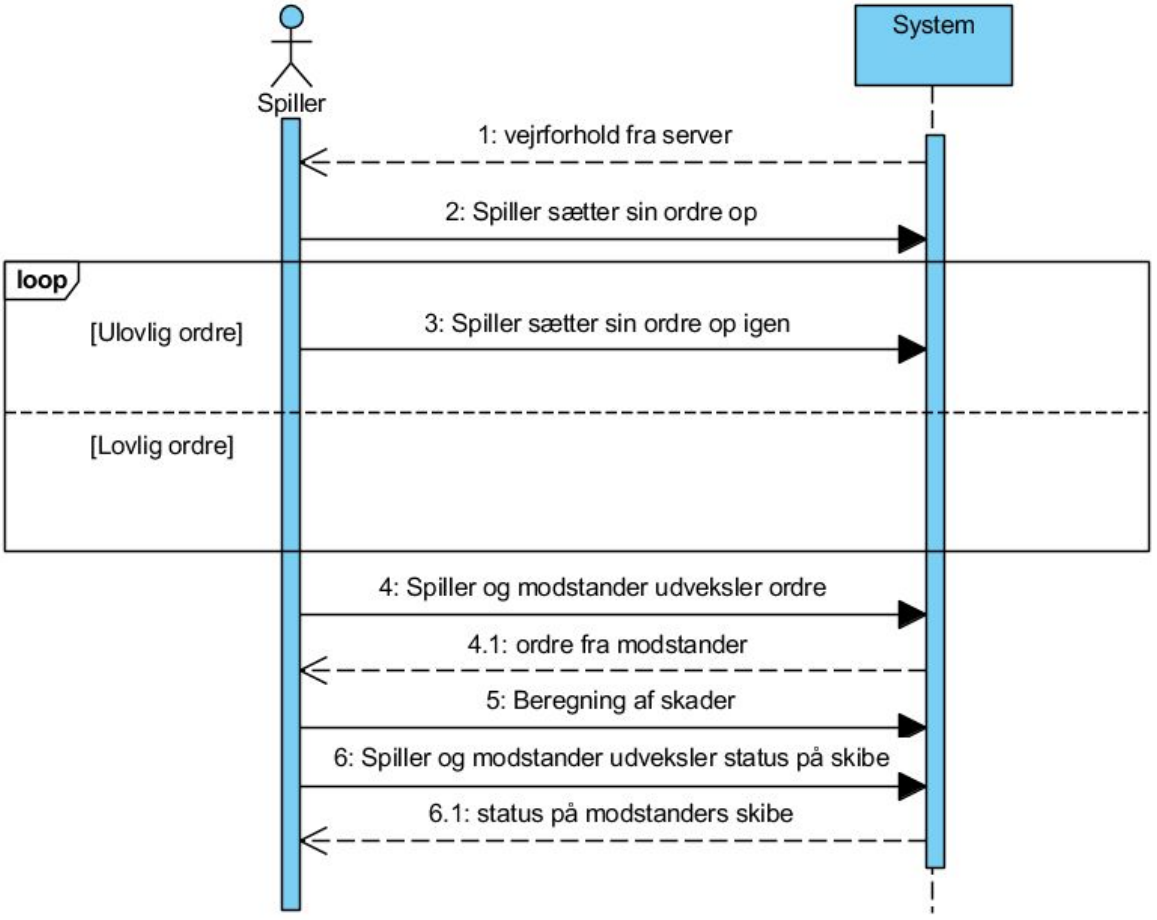
ERD

Nedenstående er et billede af vores ERD. Vi har valgt at lave det forholdsvis simpelt og det er derfor ikke i 3. normalform. Da vi kun har tre skibstyper med faste variabler af fx max speed og max sails, der ikke ændrer sig undervejs, har vi valgt at lægge dem sammen i shiptype, selvom de konceptuelt i vores domænemodel ligger i tre forskellige “klasser”.



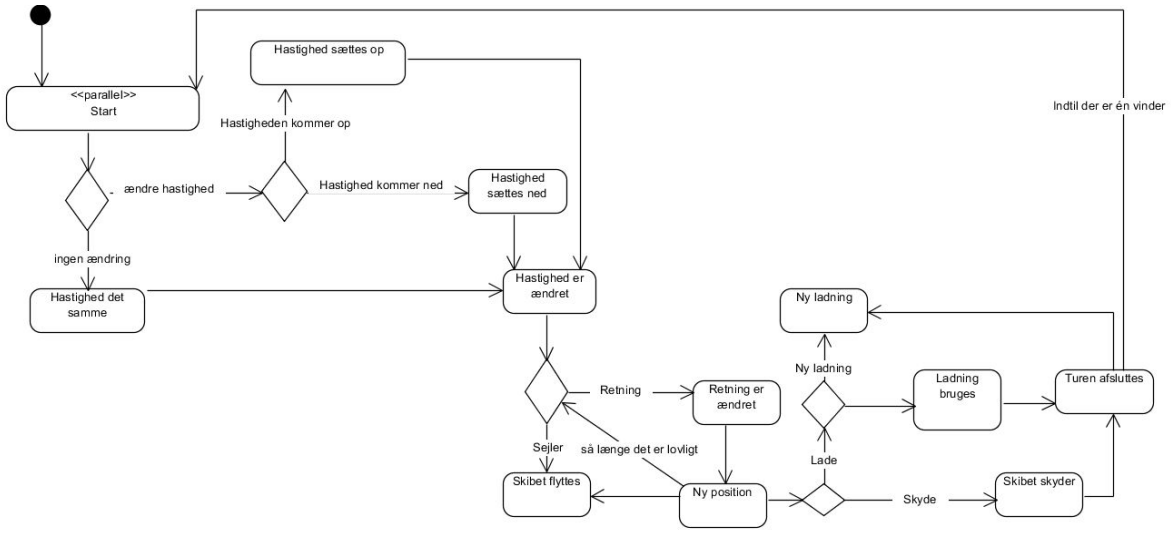
SSD

Nedenstående er et billede af et SSD over UC2 “Spille en runde”. .



Tilstandsmodel

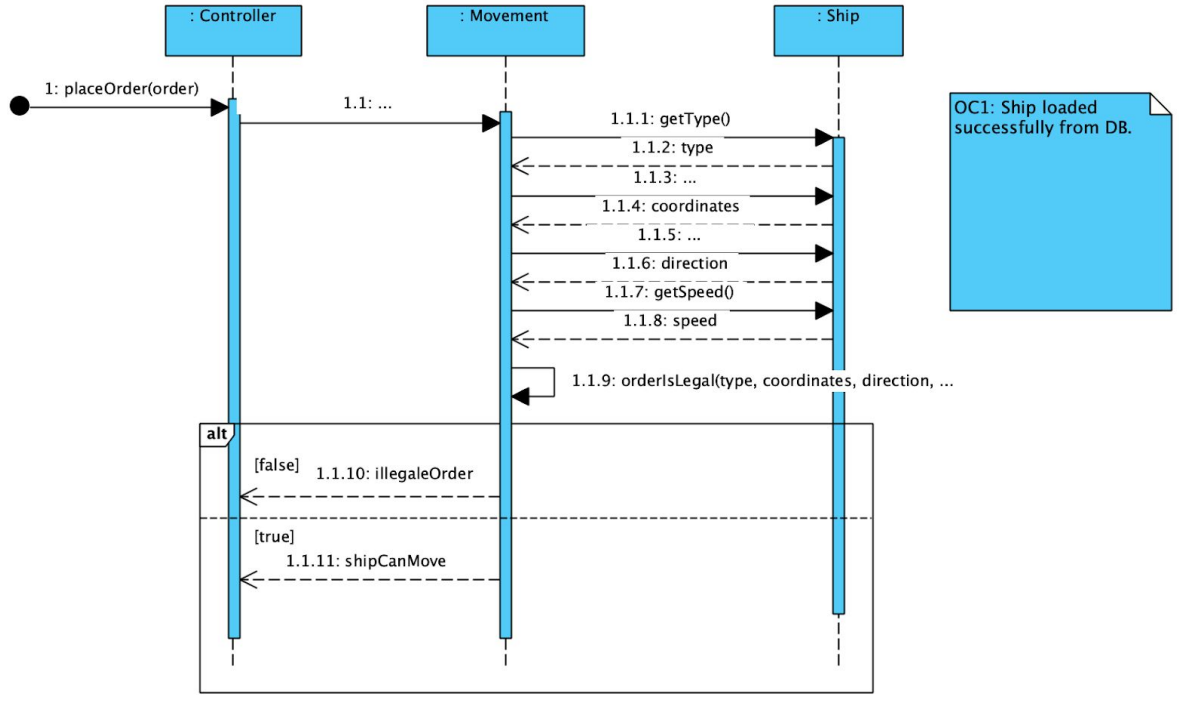
Nedenstående er vores tilstandsmodel over skibet i spillet. Modellen viser et skibs tilstand gennem en runde, der gentager sig indtil en vinder er fundet.



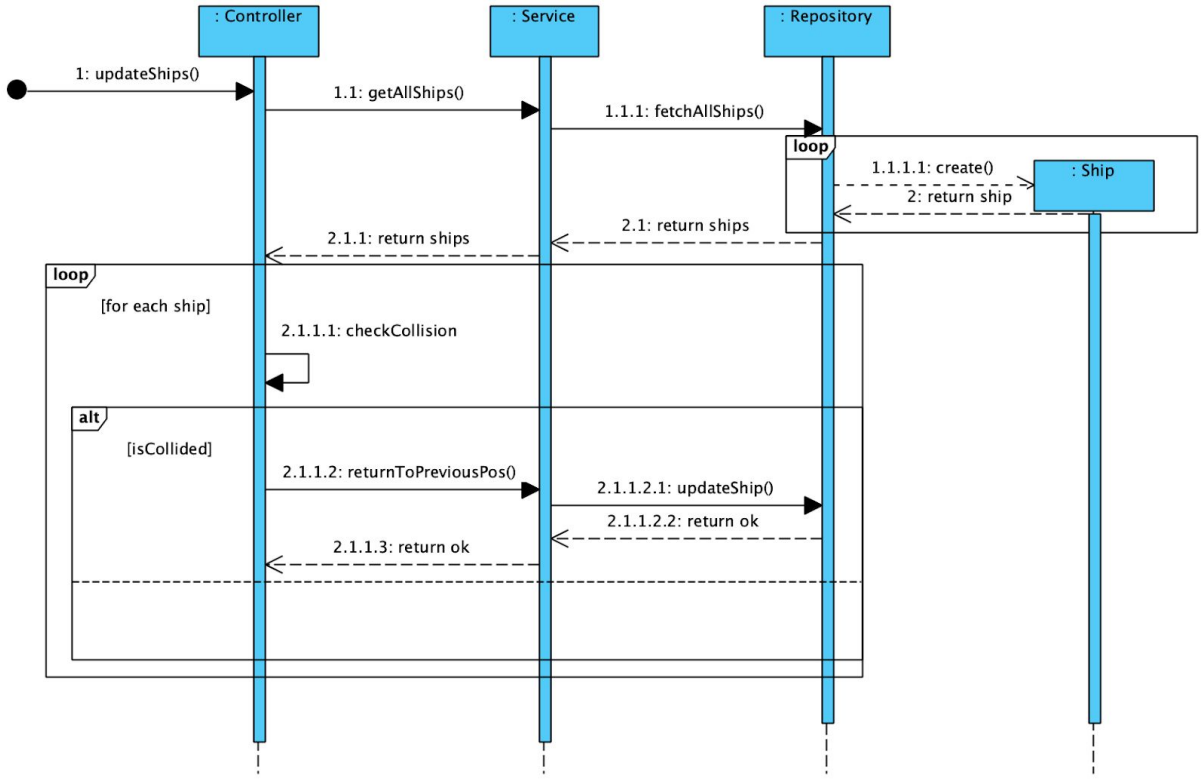
SD

Vores to SD'er over movementcontrol-delen svarer til vores første iteration af movementcontrol og svarer altså IKKE til den egentlige implementering.

SD over at flytte ét felt:

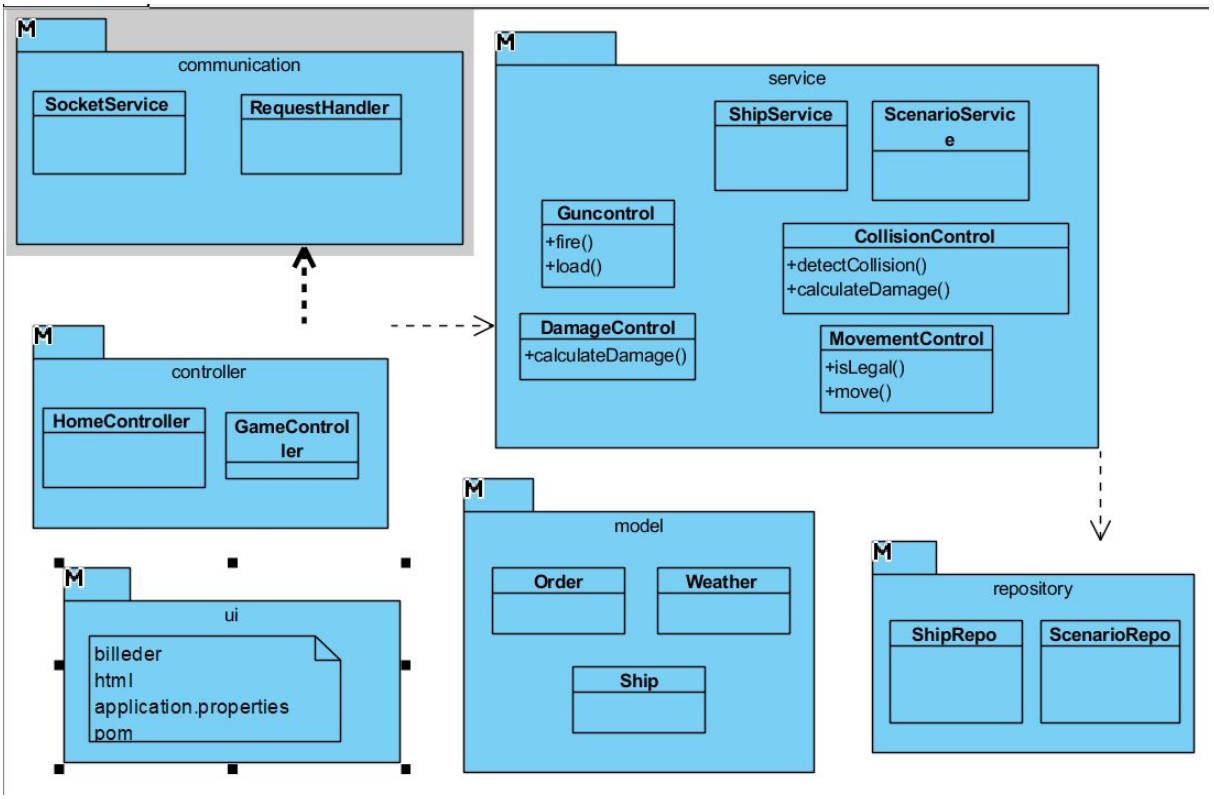


SD over at udregne kollisionsskade:

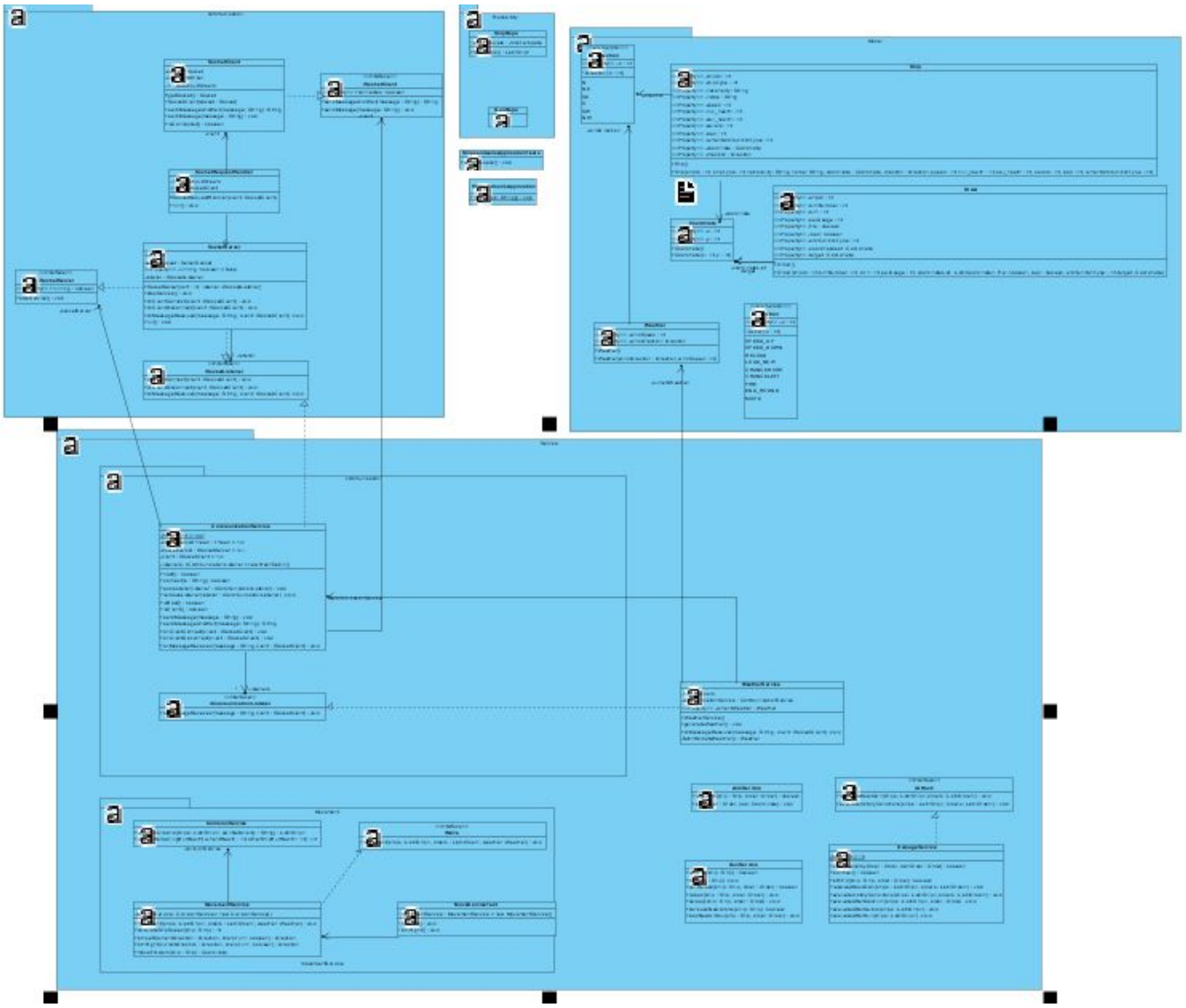


Klassediagram

Nedenstående er to udgaver af et klassediagram. Det første viser vores overordnede første idé af inddeling af klasser, der viser vores MVC-mønster. Vi har brugt dette til at opdele implementeringsansvaret imellem os og få en overordnet idé af programmets struktur, og klasserne har derfor ikke metoder og attributter.



Det andet diagram er reversed fra IntelliJ og viser vores egentlig implementering. Her er det tydeligt at se, at vi har set os nødsaget til at lave flere klasser. Bl.a. var der ikke tænkt på at sigte og selve kommunikationsdelen var svær at forstå, før vi gik i gang med at kode den. Det er også tydeligt at se, at vi ikke er færdige, da der findes en del klasser, der ikke er forbundet med andre.



Konklusion

Denne opgave er løst i Java Spring boot, der er et framework for webapplikationer og ved hjælp af Java server/klient. I et virkelig scenarie ville man ikke løse denne form for multiplayerspil ved at lade en spiller være både server og klient. Det ville i stedet være mere rigtigt at lave selve programmet i Spring, lægge det op som en webserver og lade hver computer være klienterne.

For at løse denne opgave skulle alle grupper ideelt samarbejde om ændringer i model- og interfaceklasserne, men dette er efter vores mening ikke sket helt nok. Der dukker konstant uafklarede designmæssige spørgsmål op, når man koder, som man ikke havde fantasi til at forestille sig i designfasen. Der er en vis uenighed grupperne imellem om, hvordan reglerne skal fortolkes.

Vi fokuserede vores arbejde på iterationer af design, da design er helt afgørende for at få spillet til at virke. Vi brugte lang tid på først at tale om hvert enkelt element i reglerne.

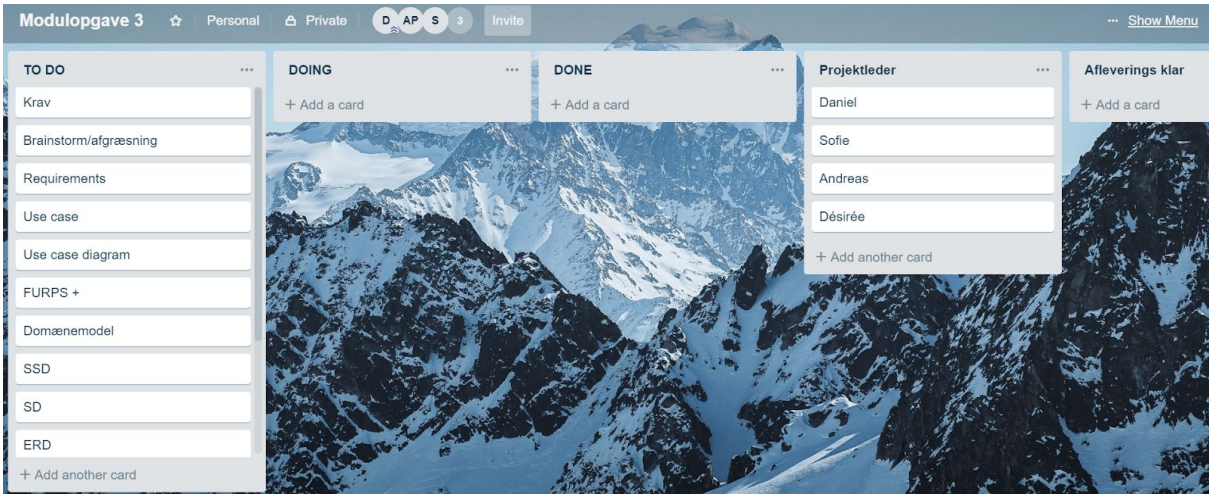
Dernæst gik vi i gang med forskellige designmodeller, men før vi fik lavet dem færdige, gav de anledning til nye diskussioner af reglerne, og dermed opstod en ny iteration. Samme mønster gentog sig i implementeringsfasen, hvor vi knap var gået i gang med at skrive en metode, før der opstod nye syn på og spørgsmål til reglerne.

Alt dette betyder, at vi ikke afleverer et færdigt produkt, der kan spilles. Der er stadig uafklarede spørgsmål grupperne imellem, og implementeringen behøver flere iterationer. Vi har dog alle været med til at designe, og vi har også alle kodet en del af programmet.

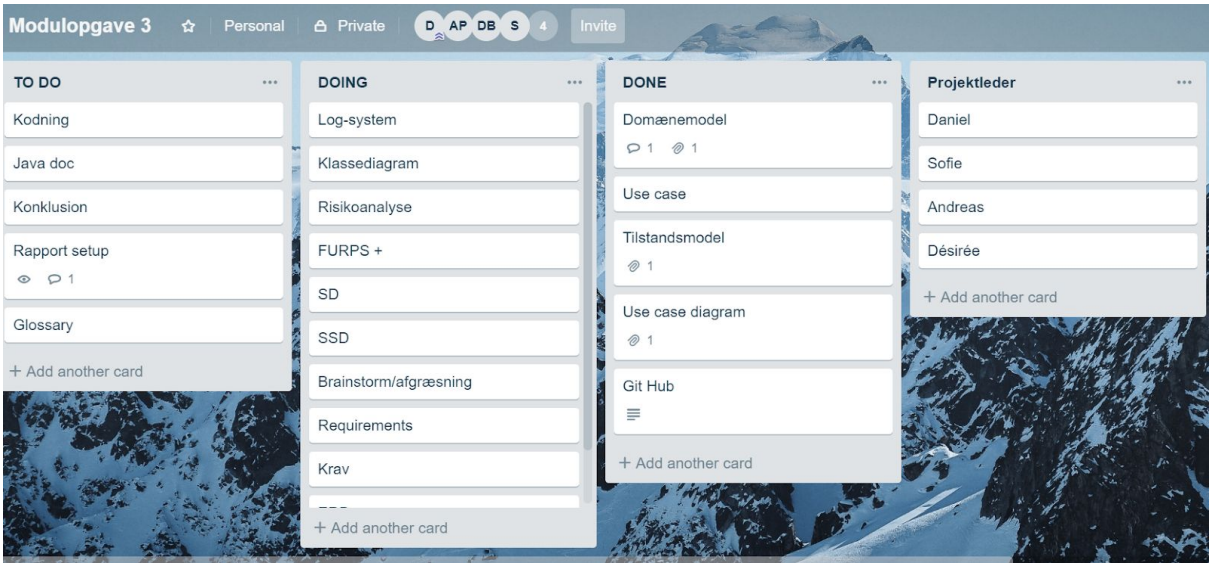
Bilag

Trello

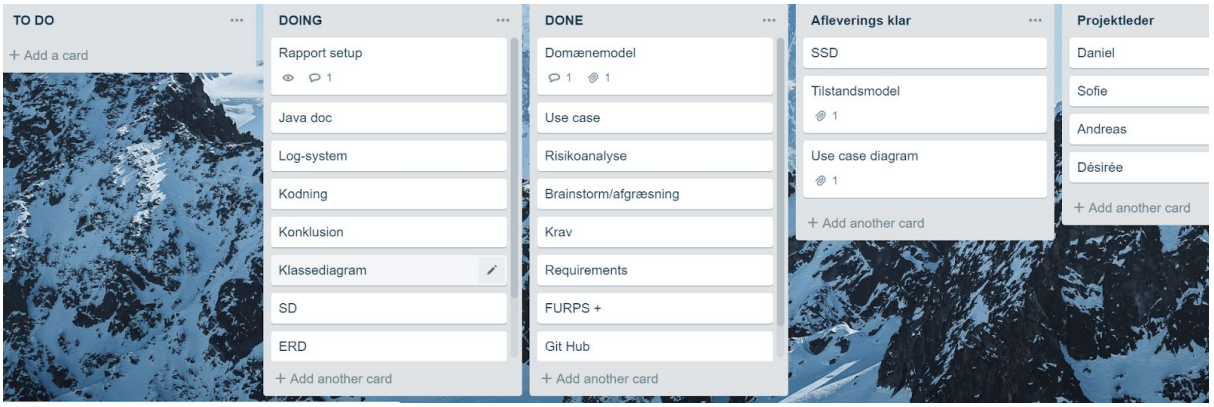
Uge 14.



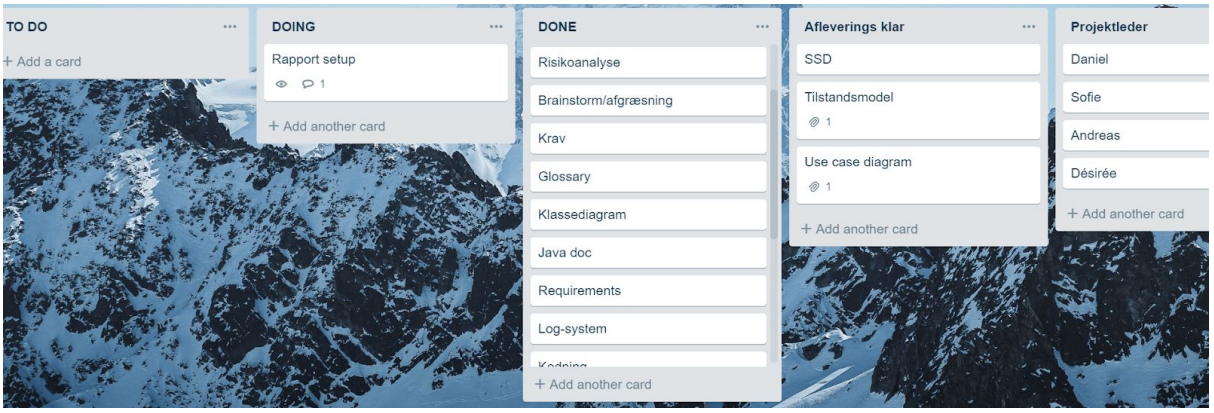
Uge 15.



Uge 16 (Påskeferie)

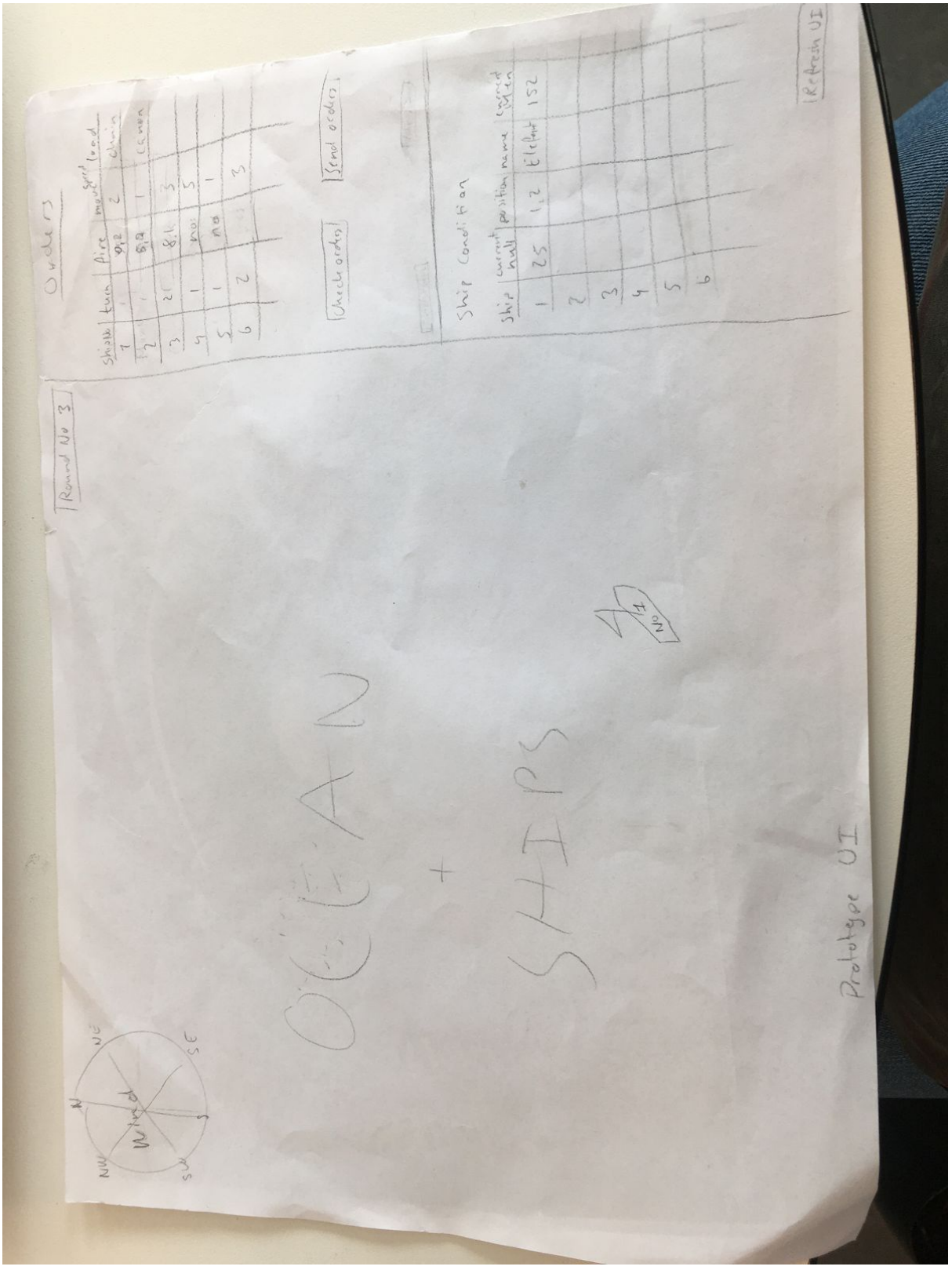


Uge 17



UI-prototype

Vi har lavet en prototype over UI for at visualisere brugerinterfacet på hjemmesiden. Vi har lavet det for bedre at kunne designe hjemmesiden og for at opdage fejl og mangler i vores design.



LOGSYSTEM:

Dag 1- 01.04:
Sofie og Desiree: Brainstorm, oprettet Trello,Git Hub, google doc, krav, og use cases.
Andreas: Oprettet rapportskabelon i LaTeX - pushet til github.
Dag 2- 02.04:
Alle: Møde om uddeling af opgaver
Sofie: data på skibe til tredje normalform
Désirée: Use case diagram
Andreas:Risikoanalyse
Daniel:Furps+
Dag 3- 03.04:
Alle mødes kl.9.12 - kigge på Domænemodel, lave UI-prototype
Dag 4 - 04.04: ITO og andre lektier
Dag 5: Designsnak

Weekend:
Sofie: SSD over UC2
Desiree: gøre Use case færdig samt use case diagram
Daniel: FURPS+
Andreas: Risikoanalyse

Dag 6 - 08.04: Andreas,Sofie og Désirée har lavet tilstandsmodel og gået i gang med SD.
Dag 7: Undervisning
Dag 8: Daniel,Sofie og Désirée har kigget på klassediagram - men mangler oplysninger fra Jarl og Asger for at komme videre med det. Andreas har lavet udkast til et SD.
Dag 9: Undervisning
Dag 10. 12.4 Har fordelt hvem der skal lave hvad. Kigger på det hver især i vores påskeferie.

Påskeferie:
Sofie: guncontrol-implementering.
Andreas: Movement implementering
Desirée: UI
Daniel: communication + database

Dagene efter påske har vi prøvet at sætte vores opgave sammen og rettet de sidste ting i vores rapport.