3.0 Programvarearkitektur og design

[3.1 Organisering, modeller](#h.mhrz8knotp4v)

[3.1.1 Repository modell](#h.uto6epfbddoa)

[3.1.2 Klient-tjener modell](#h.5cam8e823kve)

[3.1.3 Lagdelingsmodell](#h.ha4jwxv5v4rb)

[3.2 Konklusjon](#h.2uxyi9fokstq)

[3.3 Kontrollmodeller](#h.1i4asmd5rvbw)

[4.0 RUP 4+1](#h.s65my3212qce)

[4.1 Logical / implementation view](#h.jl41swtccxlz)

[4.2 Deployment view](#h.x0e6uoin07wg)

[4.3 Process view](#h.1b89ikd86h8a)

[4.3.1 Bruker beveger seg](#h.84ggeihs62uf)

[4.3.2 Bruker logger seg inn](#h.fhepaxjabd6y)

[5 Dypdykk: GUI](#h.402rhs1ak8ph)

[5.1 Virtuell verden](#h.rbahgrnohu20)

[5.2 Minispill](#h.kb3zye6q8mwk)

[Kildereferanser](#h.o90bklqr2ie)

# 

# 

# 3.1 Organisering, modeller

### 3.1.1 Repository modell

*All informasjon/data ligger lagret på en sentral database.*

* For vårt prosjekt er ikke kostnader for drift og vedlikehold et stort problem, derfor gir denne modellens reduserte kostnader liten gevinst.
* Det kunne vært en fordel at alle klienter kun har en enkelt server å forholde seg til, men da prosjektet har et krav om skalerbarhet, vil dette være bli en svakhet, da den sentrale servers kapasitet dikterer den overordnede kapasiteten.
* Da prosjektet også har et krav om høy oppetid, blir risikoen med å ha et SPOF (single point of failure) for stor.

### 3.1.2 Klient-tjener modell

*Beskriver et distribuert system hvor data og oppgaver er delt mellom flere maskiner.*

* Modellen gir best muligheter for skalerbarhet, noe prosjektet krever, da man kan kjøre opp flere servere for å avlaste tjenester med stor pågang.
* En ulempe vil være at en klient må forholde seg til flere servere og distribuert kommunikasjon blir mer komplisert å forholde seg til, men dette kan løses med å introdusere en lastbalanserer som fordeler oppgaver fra klienter til riktig server.
* Systemet vil få et mer komplisert vedlikeholds- og backup regime, da det vil være mange servere som gjør forskjellige tjenester og funksjoner.

### 3.1.3 Lagdelingsmodell

*Systemet er inndelt i presentasjonlag, applikasjonslag og databaselag. Systemet blir delt opp i mindre komponenter.*

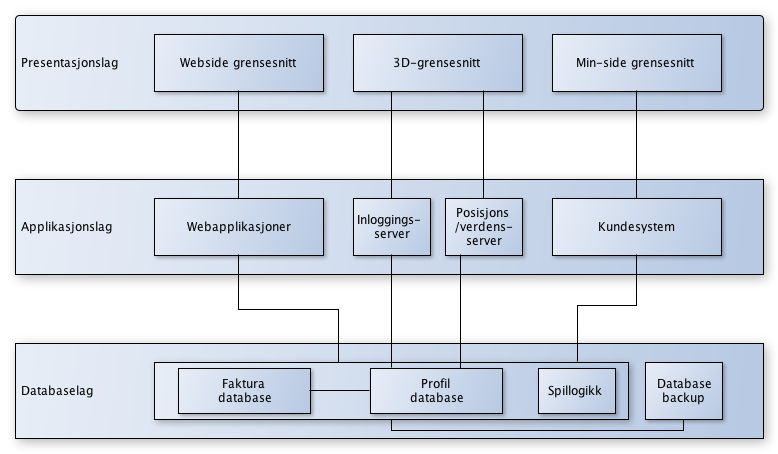
* Lagmodellen vil gi systemet gode muligheter for fremtidige utvidelser og oppgraderinger da et lag kan endres/oppgraderes uten at de underliggende lagene trenger endringer, så lenge ikke grensesnittene mellom lagene endres. Dette er viktig da ChessCom ønsker muligheter for fremtidlige utvidelser og oppgraderinger.
* Modellen tilbyr systemet en god måte å skille de forskjellige komponentene i systemet, men det vil være deler av systemet som ikke passer i modellen, da det er eksterne databaser (faktura, nettbutikk) og kommunikasjon på tvers av lagene.

## 3.2 Konklusjon

Vi valgte en kombinasjon av lagdelingsmodellen og klient-tjener modellen.

Dette er fordi vi da kan ha en sentralisert lokasjon med alle servere og hardware. Vi vil designe det slik at vi har en hovedserver som peker vidre til separate servere for ulike applikasjoner og oppgaver. Ved å ha en slik arkitektur blir det lett å drive vedlikehold og drift, og i tilegg er det lett å utvide kapasiteten da dette er relevant mtp økende kundebase og oppgraderinger i spill, grafikk og ytelse.

Ettersom det vil være kommunikasjon på tvers av lagene vil deler av lagdelingsmodellen ikke helt passe inn, og vi mener derfor at en kombinasjon av lagdelingsmodellen og klient-tjener modellen vil passe best.

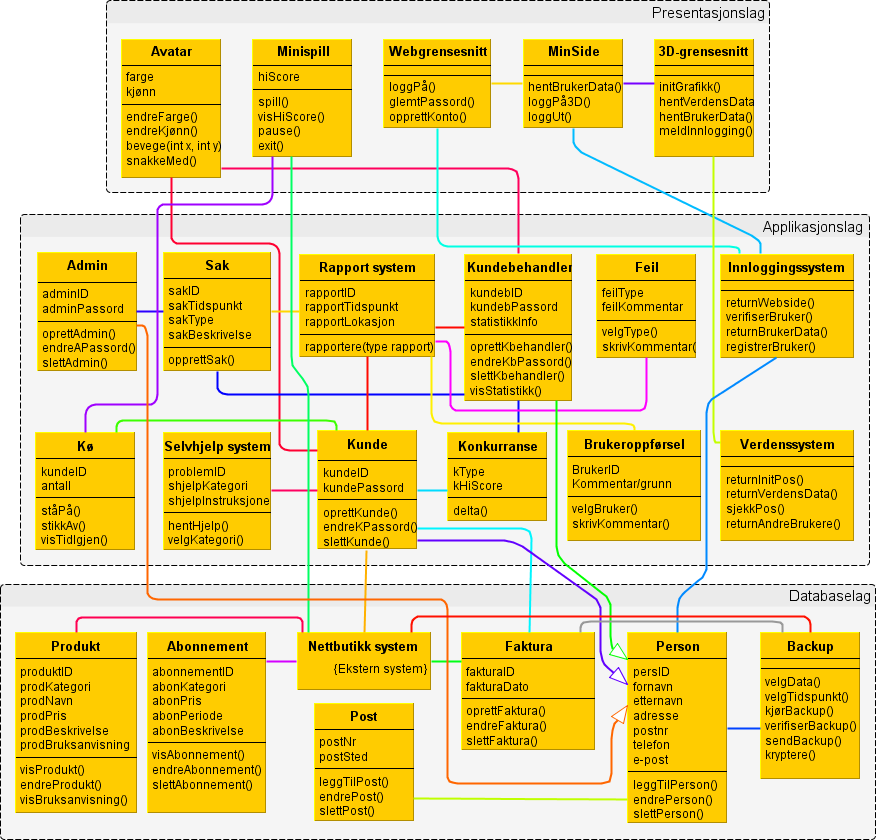


# 

# 

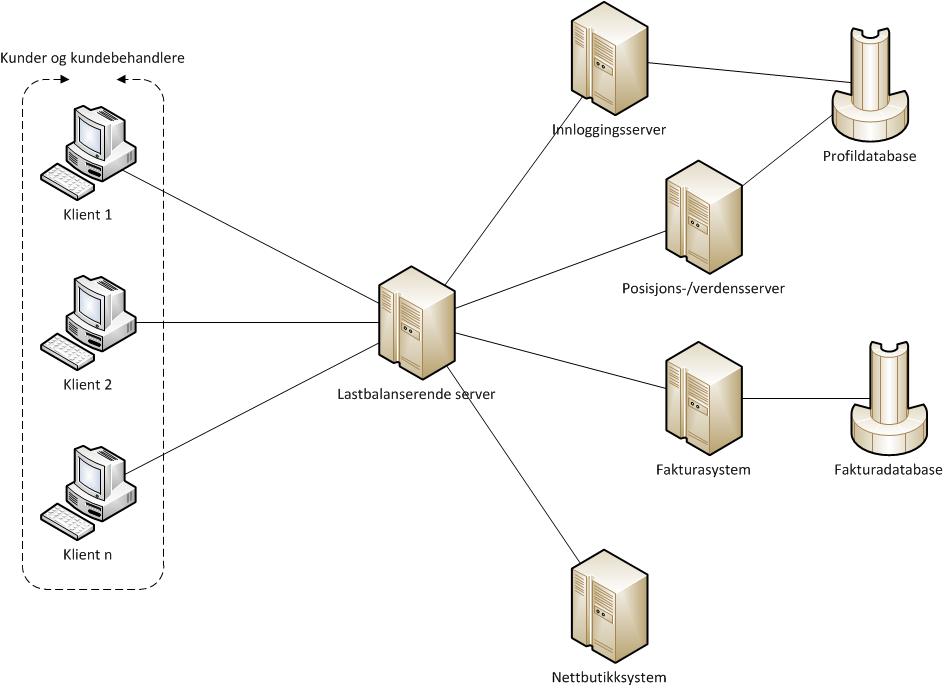
# 4.0 RUP 4+1

## 4.1 Logical / implementation view



Klassediagrammet bygger videre på det konseptuelle klassediagrammet fra kravspesifikasjonen og inneholder klassespesifike funksjoner. Det er også delt inn etter lagmodellen for å enklere forstå hvor de enkelte klasser og funksjoner hører hjemme.

## 4.2 Deployment view



Den sentrale serverens oppgave er å sørge for at alle klienter har ett initielt oppkoblingspunkt for alle funksjoner og deretter fortelle klienten hvilken server som som er tilgjengelig for forespurte funskjon. Dette gjør det mulig å koble til avlastningsservere for belastede funksjoner, skulle det være behov. Den fungerer som en lastbalanserer.

Innloggingsserveren håndterer brukernes innlogginger til systemet og kobler brukere mot sin profil i profildatabasen.

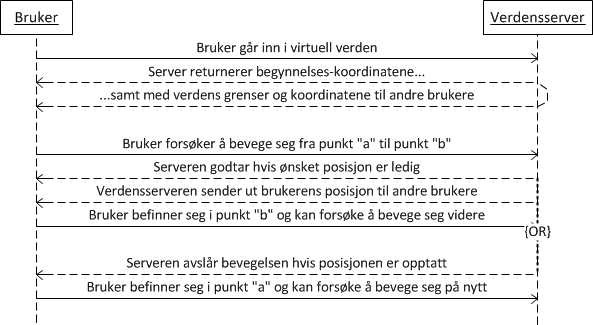
Posisjons- og verdensserveren er serveren som holder styr på hvor alle brukerne er i 3D-verdenen og replikerer disse posisjonene til samtlige brukere. Den håndterer også alle interaksjoner mellom brukere og spillverdenen, slik som kommunikasjon (chat).

Videre er systemet koblet opp mot eksisterende faktura- og nettbutikksystem.

## 

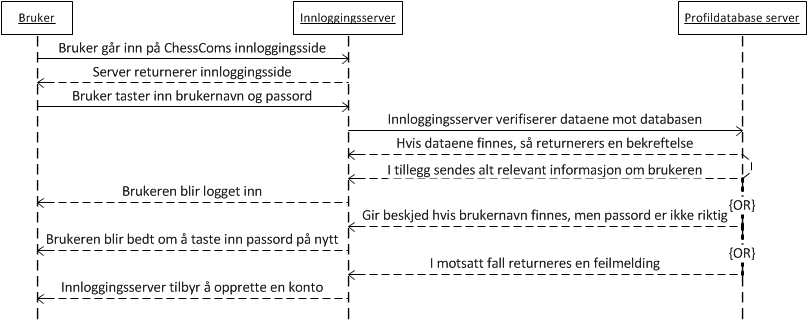
## 4.3 Process view

### 4.3.1 Bruker beveger seg



En bruker beveger avataren sin fra punkt a(x,y) til punkt b(x,y). Dette behandles først av den lokale klienten som flytter 3D-avataren på brukerens skjerm. Det kalkuleres også om denne bevegelsen er lovlig og at det ikke medfører noen kollisjoner med andre objekter i 3D-rommet. Denne bevegelsen blir så meddelt serveren som holder styr på brukeres posisjoner i 3D-rommet med informasjon om at bruker 1 har beveget seg fra *a* til *b*. Denne serveren sender så ut en melding til alle aktuelle klienter med samme melding om bruker 1’s posisjonsendring. Dette gjør at disse klientene oppdaterer sine posisjoner for bruker 1.

### 4.3.2 Bruker logger seg inn



En lokal klient forespør login-serveren om den er tilgjengelig for innlogging. Ved positivt svar gjøres innlogging på serveren mulig for brukeren. Brukeren taster inn brukernavn (klientnummer) og passord (dette kan også hentes fra MinSide hos ChessComs nettsider). Brukeren logges så inn, om brukeren allerede har en profil hentes denne frem fra databasen og lastes til klienten. Dersom ikke overføres brukeren til “lag profil” prosessen. Brukeren initiereres så inn i 3D-rommet.

# 

# 

# 5 Dypdykk: GUI

## 5.1 Inspirasjoner

Prosjektet har hentet noe av inspirasjonen til GUIet fra to online spill, World of Warcraft og EVE Online.



I GUIet til World of Warcraft, og liknende spill, er det et minikart (markert i rødt) over verdenen som lar en bruker enkelere navigere i verdenen og finne frem til interessepunkter. Dette er et nyttig feature for dette prosjektet da ett av kravene er at en bruker enkelt skal finne frem til riktig hjelp innen rimelig tid.



Spillet EVE Online er veldig chat-orientert, slik som også et supportsenter er, og har et godt system for å håndtere samtaler med flere parter på en gang. Chatvinduet benytter et fanesystem der hver fane er et eget chatrom eller privat samtale. Når noe nytt er sagt i en fane vil denne begynne å blinke for å fange brukerens oppmerksomhet.

## 5.2 GUI Konsept



Vi har valgt at brukerens GUI skal være minimalistisk og low-key da brukerens fokus skal være å interaktere med verdenen. Det er derfor fokus på få objekter på skjermen samtidig.

Brukerens avatar representeres som en 3D-figur i tredjepersonsperspektiv midt på skjermen. Andre brukere representeres som liknende 3D-figurer (ikke vist på bildet).

Hoveddelene av GUIet vil være menyen øverst til venstre og kartet øverst til høyre, samt chatvinduet nede til venstre. I tillegg vil aktuell informasjon for brukeren dukke opp som tekst øverst i midten (eventuelt andre steder om det er mer relevant).

Menyen vil inneholde relevante valg for brukeren, som å endre innstillinger for lyd og grafikk, lese hjelp dokumentasjon og logge ut.

Kartet vil hjelpe brukeren å navigere i spillverdenen ved å markere interesseområder og fortelle brukeren hvor en finner et bestemt område i spillverdenen, slik at det er enkelt å navigere til et område.

Chatvinduet vil være brukerens mulighet til å kommunisere med andre brukere i verdenen. Dette vinduet vil være delt inn i separate faner, slik at det er mulig å holde parallelle eller private samtaler med f.eks en kundebehandler.

# Kildereferanser

-Forelesningsnotater

-Slides, Tom Røise

-Pensumbok: Head First Software Development

-Artikkelsamling IMT2243 Systemutvikling (Våren 2012)

# Vedlegg