TSS Rapport

Gruppe 1

Martin Kassoe Fredslund (05705)

Mikkel Rosendahl (05877)

Esben Sune Rasmussen (05831)

Claus Andersen (04296)

# Designovervejelser

Første design havde en authenticate klasse som indgangsvinkel til serveren. Denne returnerede ved login en facade der afspejlede de muligheder en given bruger måtte havde. Altså så en facade factory.  
Dette princip kunne dog ikke lade sig gøre fordi web services ikke kan overføre referencer.  
Det endelige design benytter en login metode på de enkelte services til at håndtere sikkerheden. De servicefacadeobjekter som klienterne opretter benytter sig af en række objekter internt. Disse objekter bliver først instantieret når brugeren med succes kalder login metoden. Forsøges andre metoder end login at kaldes før dette sker, vil der blive kastet en null pointer exception da et af førnævnte objekter anvendes.

Heterogenitet er opnået ved at serveren er skrevet i java og klienterne i C#. Java siden blev lavet først og C# delen blev herefter genereret ved først at anvende værktøjet WSDL.exe i .net og siden hen den indbyggede web reference funktionalitet i visual studio.

Access transperency er i høj grad opnået og fx som beskrevet omkring login så kastes en exception opstået på serversiden i det distribuerede objekt på klientsiden nøjagtig som var den opstået lokalt i et ”normalt” objekt.

Systemet forudsætter at der indtastes en host adresse for at hente WSDLen. Dette er herefter hardcoded i programmet.

Der har fra starten været et ønske om at det skulle være nemt senere at udskifte enkeltdele i systemet. Fx ville vi gerne kunne udskifte middelwaren senere uden at det påvirkede de andre dele i systemet.

# Patterns

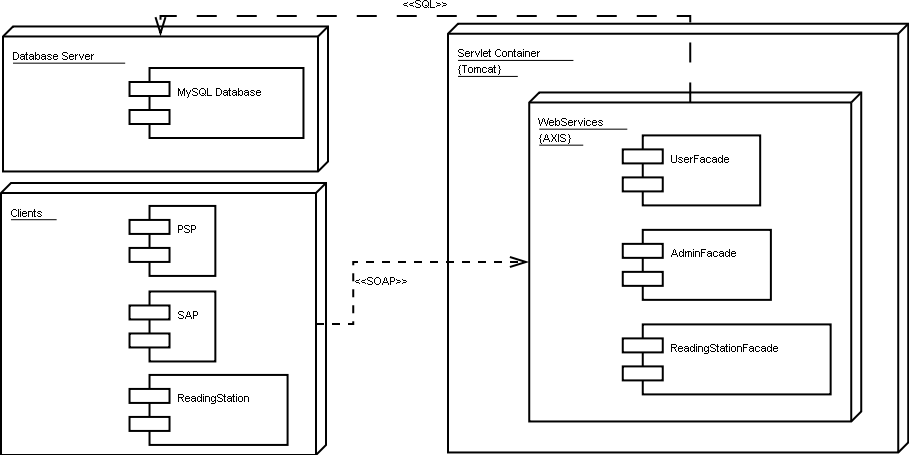
Der er anvendt **facadepattern** ud mod klienterne. De services som klienterne benytter, er facade klasser som stiller et interface til rådighed alt efter hvilket login der bliver brugt.

Til kommunikation mellem klienterne og serveren anvendes data transfer objects (**DTOpattern**). Dette gør at de konkrete metoder der kaldes blot tager imod en specifik DTO og ikke en række forskellige datatyper. Hvis der senere ændres i hvilke data en metode skal tage i mod tages dette blot fra DTOen og metodens signatur forbliver derved uændret.

I første omgang blev der lavet en driver til Pico TH-03 temperaturmåleren og for at kunne bruge den uden at have den fysiske enhed blev der lavet en simulator. Dette benyttede **proxypattern** og gjorde det derved transparent for applikationen hvad den egentlige kilde til data var. Dette blev dog droppet da den fysiske enhed helt blev droppet.

# Klassediagram

# Deployment diagram



# Refleksion

Det viste sig at være svære end umiddelbart antaget at designe systemet og der gik lang tid med nogle detaljer. Skulle et lignende projekt laves igen ville mange af disse detaljer tage væsentlig kortere tid men der ville stadig gå en del tid med det.

Undervejs er vi stødt på følgende plusser og minusser:

* Der kan kun overføres objekter der lever op til java bean standarden. Altså ikke fx facader som vi først havde håbet på.
* Håndtering af sessions sker ved blot at definere at scope skal være lig session i WSDLen og ved at oprette en cookiecontainer i klienten. Det må siges at være enkelt.
* Overførelsen af tids typer er lidt kompleks da man ved distribuerede systemer nemt kan blive konfronteret med forskellige kalendertyper, tidszoner og sommertid. Der findes dog typer der håndtere disse problemstillinger så længe man er dem bevidst.
* Importering/genering fra WSDL filer har mange problemstillinger, en af disse vi er stødt på er ”brækkede” schema links (sikkert på grund af skolens proxy udad til).
* Det er ikke muligt at overføre interfaces vha axis hvilket kan medføre nogle mindre logiske konstruktioner.
* For at kunne bruge observer pattern skal alle computere have installeret axis. Da kun serveren kørte axis i vores setup var vi nødsaget til at polle for ny data fra klienterne. Dette giver generelt en del unødvendig trafik men kan ikke undgås her.

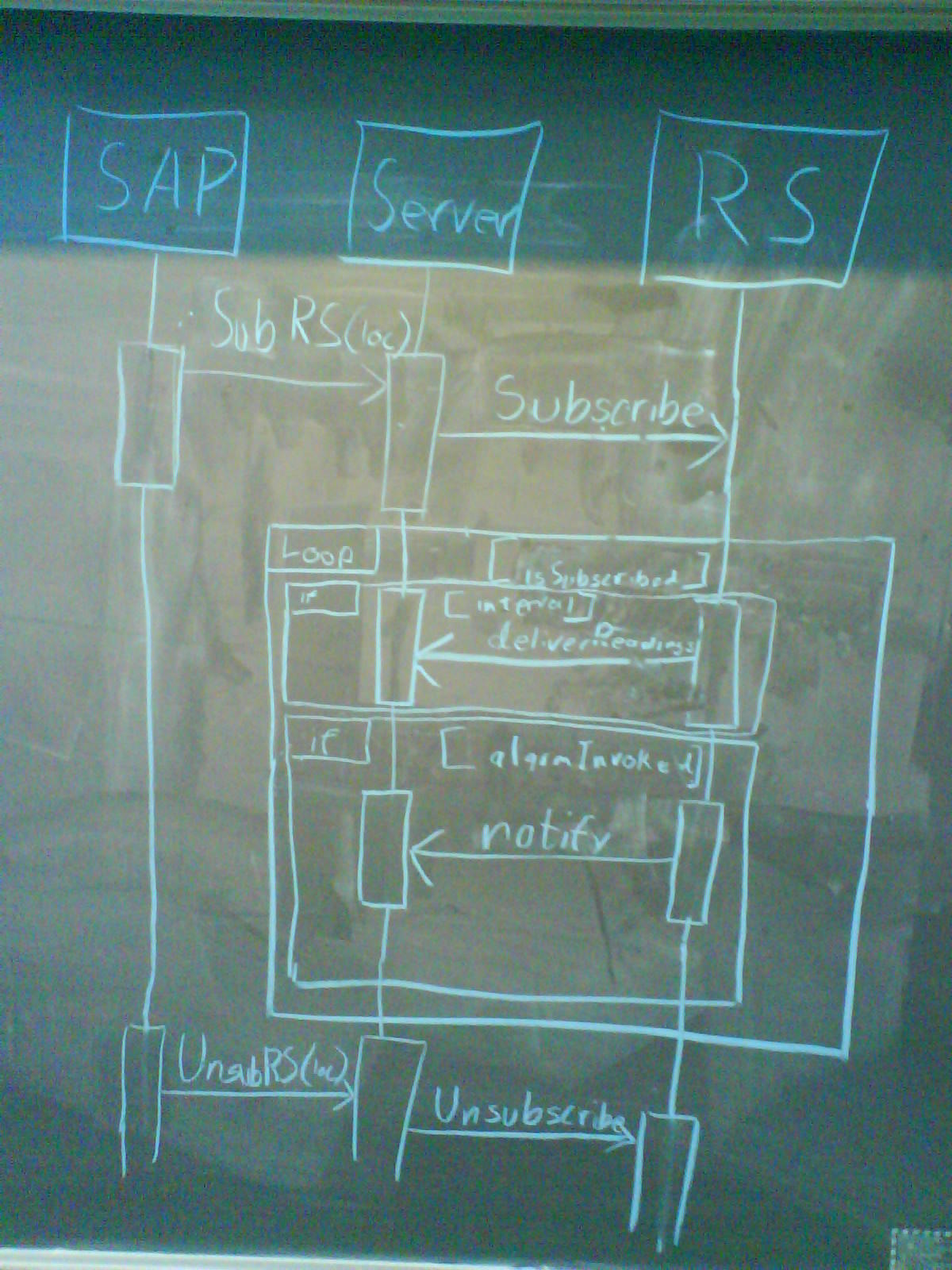
## Hvorfor vi ikke fik fuldført projektet

Vi må nok indse at planlægningen ikke har været alt for heldig. Resultatet var en stor mængde arbejde kort før deadline. Derudover har vores arbejde været for individuelt og ukoordineret så da vi endelig satte os sammen var der stadig lang vej før vores kode kunne sætte sammen.

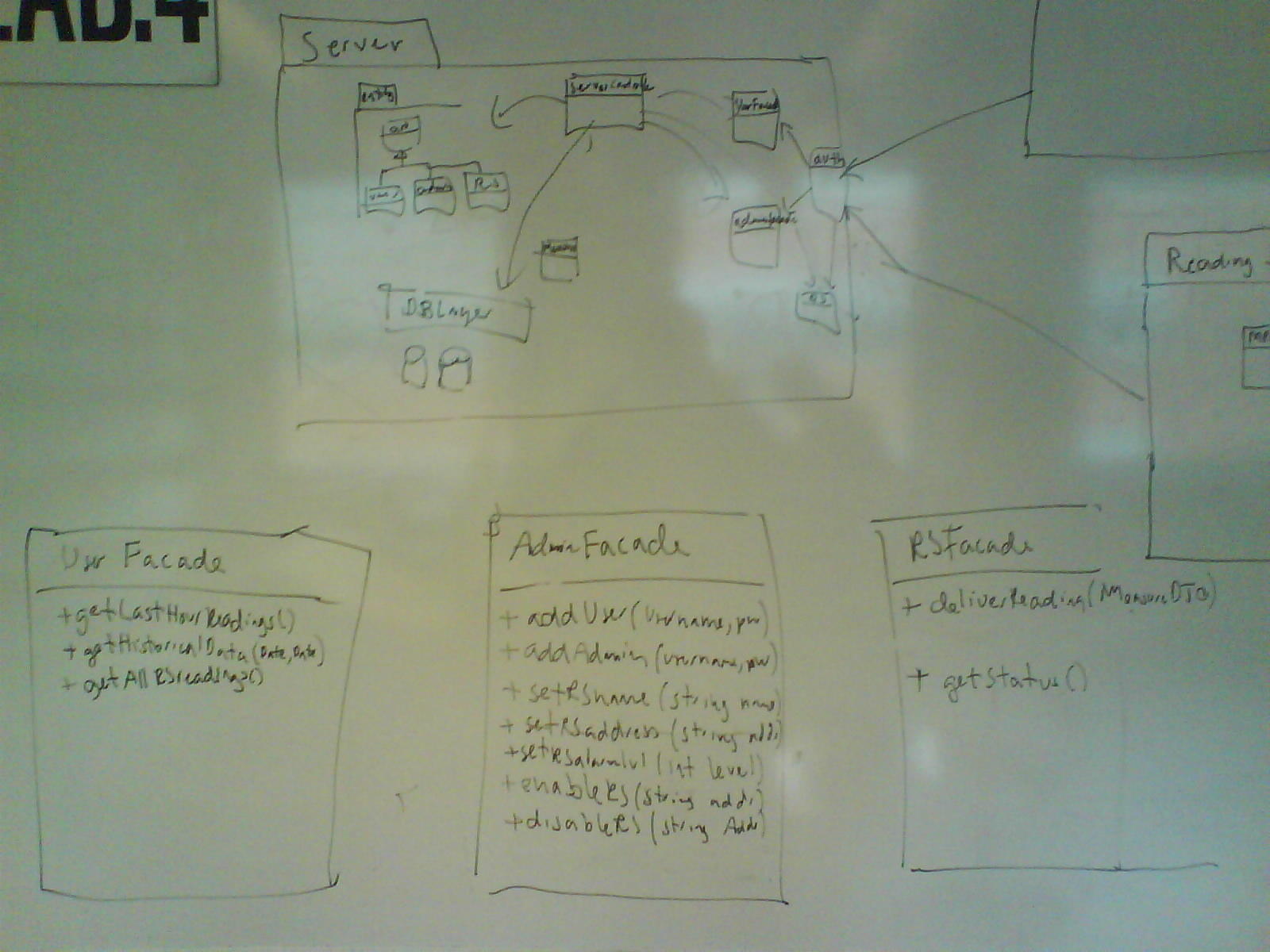
I starten af projektet lagde vi meget stor vægt på web servicen og antog at databasedelen var nem og at den nærmest gav sig selv. Det viste sig ikke at være tilfældet, fordi vi ikke var enige om vores mål. Vi havde nogle lækre interfaces og det var det.

Enkelte dele som fx login kører men sammenkoblingen af alle delene blev ikke færdig.

## Bilag A



## Bilag B



## Bilag C

