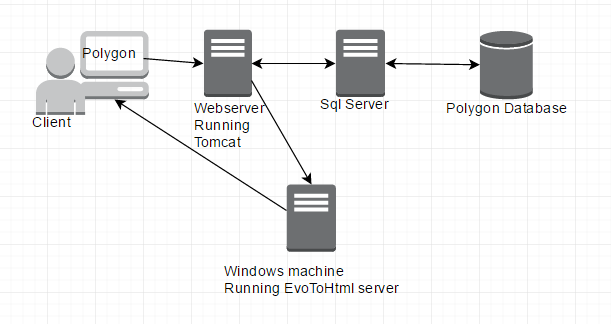
|  |
| --- |
| Sundebygninger |
| Semester 2. Datamatiker |
|  |
| En gennemgang af oplysningerne om et system til sundebygninger.dk |
|  |



# System opbygning (Forfatter: Joacim)

Vores system er opbygget som det ses på nedenstående figur;



En klient tilgår sundebygninger.tk som der hostet af en Tomcat web server som kører på en Ubuntu server, Tomcat web serveren tilgår så vores sql server, der har en database med alt data til systemet.

Vi har også en Windows maskine der har en pdf converter server kørende, Windows maskinen bliver kaldt direkte fra web serveren, og returnerer en pdf fil til klienten, som der er konverteret fra en html side.

## Performance

Grunden til at vi både har en server til sql og en til Tomcat, er at vores Web server kun har 500mb ram, hvilket gjorde at vores side havde det med at crashe lidt tilfældigt, dog er det en fordel, da det gør det lettere at skalere fx sql serveren op nu, da den kun har et formål.

Dog er vores web server og sql server ikke på samme netværk, hvilket kan medføre nedsat hastighed når web serveren skal hente data.

## Sikkerhed

Serveren der kører Tomcat, er placeret i Frankfurt derfor skal den kunne tilgås udefra, det vil sige at SSH er slået til på port 22.

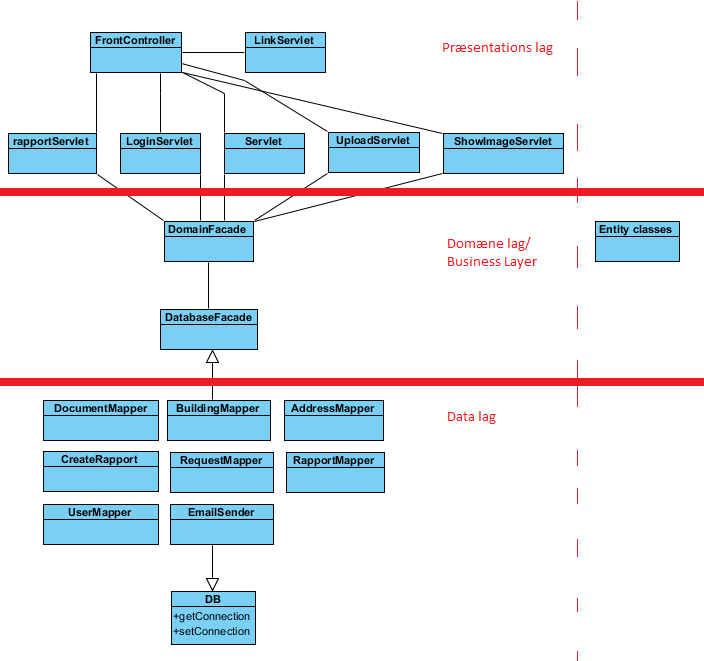
Dog er login med password slået fra, login som root slået fra, så det kun er muligt at logge ind via en public key som vi har genereret, samt installeret en private key på serveren. Vi kunne også kun have tilladt login fra specifikke IP’er, men det har vi valgt ikke at gøre da vi stoler på vores ssh keys.

Med hensyn til Tomcat har vi ikke taget nogle specielle forbehold, udover selvfølgelig at have en stærk kode til web manageren.

Serveren i Frankfurt forbinder til en Server placeret i Danmark, den server tillader kun trafik på port 3306(standard sql port) fra ip adressen: 138.68.66.172(Serveren i Frankfurt).

Den SQL bruger der bliver forbundet med har kun adgang til polygon databasen på serveren og intet andet.

# Klassediagram (Forfatter: Joacim)



## Tre lags struktur

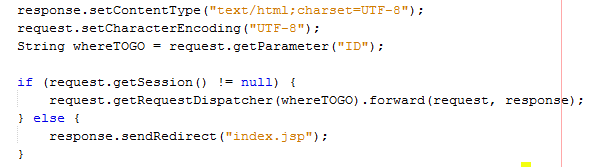
Som det ses på ovenstående figur, har vi gjort brug af en tre lags arkitektur

En af fordelene ved tre lag strukturen, er at det skaber en meget lavere kobling i mellem alle klasser i programmet, det gør det ved at hvert lag kun behøves tale sammen igennem en klasse fx, DomainFacade eller DatabaseFacade klasserne.

Det gør det også nemmere at skifte et lag ud, for eksempel hvis vi gerne ville lave et program i stedet for en hjemmeside, eller evt begge dele skulle vi ikke til at bygge hel ”bunden” op igen, men kun opbygge et nyt lag i fx java Swing i stedet for servlets og jsp’er.

## Præsentations lag

”Den øverste del” af vores program består af et præsentations lag, der indeholder vore FrontController der taler videre til alle andre Servlets, denne model valgte vi for at forsimple måden vi kontrollerer at man har fået sat en session/er logget ind, nedenstående billede viser et lille billede af koden i den:



Response er standard i Servletter generet af netbeans og gør at svaret der kommer bliver i UTF-8.

Ved linje 2 viste der sig et stort plus ved at gøre brug af en frontcontroller, her kunne vi sætte requestet til også at være UTF-8, dette medførte at ved hjælp af en linje, kunne vi gemme UTF-8 i vores database, hvis vi ikke havde haft en frontcontroller, havde vi været nødt til at dykke ned i hver servlet der gemte text med æ,ø,å eller andre UTF-8 tegn og tilføje .setCharacterEncoding(”UTF-8”);

Lidt længere nede i koden kontrollerer vi om der er blevet oprettet en session, og sender vores nu modificerede response og request videre.

Udover FrontController klassen, indeholder dette lag alt andet der relateret til vores view.

## Business lag

Når vi dykker ned i koden, eller kigger på vores klasse diagram, ses det at der ikke sker meget i vores forretningslag, vi kunne have brugt det til at lave noget som vi ikke har i databasen, fx en estimerer pris på en bygning, ved at hive adresse, størrelse på bygning osv ud af databasen, og herefter lave en algoritme i vores forretningslag som der lavede en estimeret pris.

Ser man lidt tilbage på vores valg, kunne vi godt have nøjes med 2 lag et præsentations lag og et data lag, dog er det meget godt at have dette lag hvis man skulle få brug for at lave et pris estimat på bygninger eller andre finurlige ting.

## Data lag

Data laget består af mange forskellige ting, hvis vi kigger fra toppen igen, har vi vores facade som der indeholder alle metoder som de øvre lag har brug for.

Herefter kommer vores entity klasser, der bliver brugt til fx at oprette objektet Address, der kan indeholde en adresse, et id og et postnummer. Disse klasser bliver brugt af vores mappers, der henter forskelligt data fra vores database via sql kommandoer og en jdbc driver.