IT-DIPLOMUDDANNELSEN

AFGANGSPROJEKT

af

Jan Schrøder Hansen

eFTERÅR 2011

**INDHOLD**

[1. Indledning 3](#_Toc310882203)

[2. Opgavebeskrivelse 3](#_Toc310882204)

[3. Krav 4](#_Toc310882205)

[3.1. Funktionelle Krav 4](#_Toc310882206)

[3.2. Ikke funktionelle krav 6](#_Toc310882207)

[4. Analyse 7](#_Toc310882208)

[4.1. Administrative use cases 7](#_Toc310882209)

[4.2. Primære use cases 11](#_Toc310882215)

[4.3. Krav/use case matrix 21](#_Toc310882216)

[4.4. Analysediagram 22](#_Toc310882217)

[5. Design 24](#_Toc310882218)

[5.1. Webside design 29](#_Toc310882219)

[5.2. Frameworks 29](#_Toc310882222)

[5.2.1. Wicket 29](#_Toc310882223)

[5.2.2. Hibernate 30](#_Toc310882224)

[5.3. Sikkerhed 30](#_Toc310882225)

[6. Idriftsættelse 31](#_Toc310882226)

[7. UP Iterationer 32](#_Toc310882227)

[8. Konklusion 32](#_Toc310882228)

[9. Bilag 34](#_Toc310882229)

[9.1. Danske/engelske termer 34](#_Toc310882230)

[9.2. Brugervejledning 35](#_Toc310882236)

[9.3. Kode 35](#_Toc310882238)

[9.4. Indhold på den vedlagte cd 36](#_Toc310882240)

[10. Noter 37](#_Toc310882241)

# Indledning

Dette afgangsprojekt er lavet i forbindelse med IT-Diplomuddannelsen på Ingeniørhøjskolen i København.

# Opgavebeskrivelse

Da min samlever er tegnsprogstolk, og jeg igennem hende ved at der ikke er findes brugerdrevne danske tegnsprogsordbøger tilgængelig på internettet. Der er en officiel ordbog, men denne vedligeholdes fra central side.

Jeg har fundet følgende 2 sider på internettet:

|  |  |
| --- | --- |
| Side | Beskrivelse |
| [www.tegnsprog.dk](http://www.tegnsprog.dk/) | Den officielle danske tegnsprogsordbog. Indeholder pt. ca. 2.000 ord. |
| [www.streetsigners.dk](http://www.streetsigners.dk) | Siden for de unge, med mange hippe tegn. |

Igennem min samlever ved jeg at tolke og døve tit mangler et sted hvor de kan spørge til ord og begreber som er oppe i tiden. Et eksempel kunne være ”Det arabiske forår”, hvordan siger man ”Egypten”, ”Libyen” etc. på tegn.

Det skal først og fremmest være en webløsning, senere kunne de udvides til smartphones.

Alle skal kunne søge efter ord på siden, men for at lave forespørgsel på et ord, eller uploade video forslag til et ord, så skal man være en kendt bruger af systemet. Ord skal kunne grupperes, f.eks. kan der laves en gruppe der hedder ”Det arabiske forår”, eller ”Lande i Nord Afrika”.

Man skal kunne oprette sig som bruger vha. fuldt navn, e-mail. Før man kan logge på første gang, så skal e-mailen verificeres.

Søgningen skal kunne være på ord eller grupper. Så skal man kunne tilmelde sig en e-mail notifikation på en eller flere grupper. Så man får en e-mail når der sker noget på de grupper men interessere sig for.

Løsningens navn bliver ”Tegn til tiden”.

Teknik: Java web løsning, vha. Java[[1]](#endnote-1), Apache Wicket[[2]](#endnote-2), Hibernate[[3]](#endnote-3), JavaDB[[4]](#endnote-4) og Apache Tomcat[[5]](#endnote-5).

Da jeg altid skriver på engelsk i min kode, har jeg valgt at mine UML[[6]](#endnote-6) diagrammer også er på engelsk. Men da rapporten her er på dansk, har jeg vedlagt en dansk/engelsk ordliste under bilag. Se afsnit 8.2. Dette gælder dog ikke for use case[[7]](#endnote-7) diagrammer og use cases. Som jo er det UML værktøj, som kan bruges over for mennesker, som ikke arbejder med it udvikling til dagligt.

Alle diagrammer er udarbejdet vha. af programmer MagicDraw[[8]](#endnote-8).

I de følgende afsnit gennemgås de forskellige udviklingsfaser, startende med krav.

# Krav

Følgende 2 afsnit indeholder lister med krav til systemet, opdelt efter funktionelle og ikke funktionelle krav. De enkelte krav prioriteres efter MoSCoW (Must have, Should Have, Could Have, Wants to have).

## Funktionelle Krav

|  |  |  |
| --- | --- | --- |
| ID | Krav | Prioritet |
| A1 | Systemet skal være tilgængeligt for alle ved søgninger efter ord | M |
| A2 | Systemet skal kræve at men er logget på for at uploade video filer, slette video filer, kommentere og bedømme video filer samt for at deltage i diskussioner. | M |
| A3 | Systemet skal kende en bruges fulde navn og en gyldig e-mail adresse. Dvs. at nye e-mail adresser skal verificeres. | M |
| A4 | Systemet skal give mulighed for at bedømme video sekvenser med 1 til 5 stjerner. En bruger kan kun give en rate pr. video. Men kan dog ændre eller slette sin rate senere. | M |
| A5 | Systemet skal give mulighed for at kommentere video sekvenser. | M |
| A6 | Systemet skal kunne give mulighed for at anmelde video sekvenser som anstødelige. Dette skal medføre en e-mail til en systemansvarlige brugere. | M |
| A7 | Systemet skal kunne håndtere 2 slags indlogget brugere. Admin (superuser) og almindelig. Se efterfølgende rolle matrix. | M |
| A8 | Systemet skal kunne generere en ny adgangskode, hvis en bruger har glemt sin adgangskode. Brugeren skal kunne huske den e-mail som er i systemet | M |
| A9 | Systemet skal give mulighed for at ændre navn, adgangskode og e-mail | M |
| A10 | Systemet skal ikke kunne oprette system brugere. Dette skal gøres direkte i database. | M |
| A11 | Systemet skal give mulighed for at diskutere ord. | M |
| A11 | Systemet skal give mulighed for at gruppere ord samme. | M |
| A12 | Systemet skal give mulighed for at søge efter ord og efter ord grupper | M |
| A13 | Systemet skal give brugerne et hurtig overblik over egne ord/forespørgsler, egne ord grupper og egne uploads af video sekvenser. | M |
| A14 | Systemet skal give brugerne et hurtigt overblik over ord/forespørgsler som mangler forslag samt hvilke ord der indgår i en ord gruppe. | M |

Rolle matrix for krav A7

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ikke logget ind | Alm. | System |
| Søge | Ja | Ja | Ja |
| Uploade filer | Nej | Ja | Ja |
| Slette uploads | Nej | Ja – Kun egne | Ja |
| Bedømme og kommentere uploads | Nej | Ja – Ikke sin egen | Ja |
| Diskutere ord | Nej | Ja | Ja |
| Anmelde som anstødelig | Nej | Ja | Ja |
| Forespørge på ord | Nej | Ja | Ja |
| Oprette nye grupper | Nej | Ja | Ja |
| Slette grupper og uploads som kun er knyttet til denne bruger | Nej | Ja | Ja |
| Slette/rette grupper, ord og uploads for alle brugere | Nej | Nej | Ja |

## Ikke funktionelle krav

|  |  |  |
| --- | --- | --- |
| ID | Krav | Prioritet |
| B1 | Systemet skal være en WEB-Løsning. | M |
| B2 | Systemet skal kunne kører på en Apache Tomcat version 7 ellere nyere webserver. | M |
| B3 | Systemet skal benytte java version 1.6 eller nyere. |  |
| B4 | Systemet skal benytte følgende 2 java frameworks:   * Apache Wicket – version 1.5 – som web framework * Hibernate - version 4.0 – framework som bygger bro mellem den objekt orienteret verden og den rationelle database verden | M |
| B5 | Systemet skal persistere data i en JavaDB version 10 eller nyere, som er en del af standard java. | M |
| B6 | Systemet skal kunne benyttes sammen med HTTPS. Som minimum under login processen. | M |
| B7 | Systemet skal benytte optimistisk låsning, vha. Hibernate. | M |
| B8 | De mest gense videoformater skal kunne oploades, og transformeres til videoformater som understøtter HTML5’s video tag. Pt. er det OGG formatet. | M |
| B9 | FFMPEG benyttes til konvertering af video filer. | M |

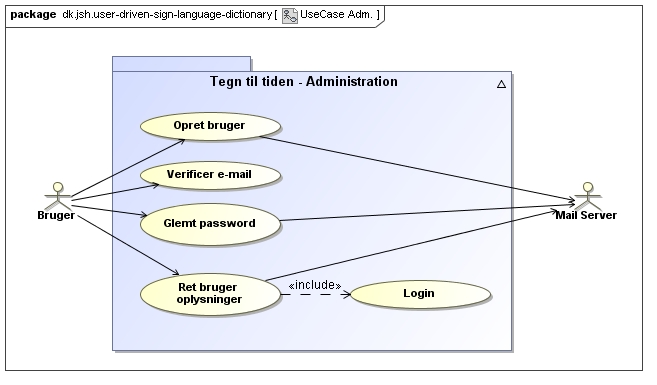
Det afslutter krav indsamlingen, og det bringer os videre til analyse fasen.

# Analyse

Jeg har valg at lave to use case diagrammer, selvom der kun er et system. Diagrammer er opdelt efter administrative use cases og use cases som fortæller hvad systemets primære opgave bliver.

Følgende use case diagram er de administrative use cases. Det er use cases som omhandler brugeroprettelse m.m.

## Administrative use cases



Figur - Administration use case diagram

På de følgende sider er ovenstående use cases beskrevet.

|  |
| --- |
| **Use case: Login** |
| **ID: UC-1** |
| **Kort beskrivelse:**  En bruger logges på systemet. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Brugeren er kendt af systemet. |
| **Hovedforløb:**   1. Brugeren udfylder brugerkode og password felterne og trykker på ”Login” knappen. 2. Systemet verificerer brugerkode og password. Hvis brugerkode og password er i orden, så er brugen ”logget” på systemet, og der er nu en mulighed for at logge af igen. |
| **Slut betingelser:**  Brugeren er logget på systemet. Dette giver adgang til upload af ord, forespørgelser på ord m.m. |
| **Alternative forløb:**   * Brugeren kan ikke logges ind, da brugerkode og password ikke kendes af systemet. Systemet melder at enten brugerkode og/eller password er ukendt for systemet. Og henviser til ”Glemt password” og ”Opret bruger” mulighederne. |

|  |
| --- |
| **Use case: Opret bruger** |
| **ID: UC-2** |
| **Kort beskrivelse:**  En ny bruger oprettes. |
| **Primære aktører:**  Alle. |
| **Sekundære aktører:**  E-mail server. |
| **Start betingelser:**  Brugeren skal have en valid e-mail adresse. |
| **Hovedforløb:**   1. En kommende bruger vælger ”Ny bruger” linket 2. Systemet viser en ny side, som giver mulighed for indtastning af Navn, brugerkode, e-mail, password og gentag password. 3. Brugeren udfylder felter og trykker på ”Gem” knappen. 4. Systemet verificerer alle felterne:    1. Navn skal udfyldes.    2. Brugerkode skal udfyldes    3. E-mail skal udfyldes med et valid e-mail adresse. Dvs. skal indeholde @ m.m.    4. Begge password felter skal være udfyldt med samme tekst. Og teksten skal være mindst 8 tegn lang og indeholde både bogstaver og tal.    5. Hvis alle felter er valide, så gemmer systemet navn, e-mail og password. Og sender en e-mail til den nye bruger. E-mailen indeholder et link til systemet som skal aktiveres før brugeren kan logge på systemet. Dette gøres for at sikre at mail adressen er valid. |
| **Slut betingelser:**  Bruger er delvist oprettet i systemet. Men kan først logge på systemet, når brugers e-mail er verificeret. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Verificer e-mail** |
| **ID: UC-3** |
| **Kort beskrivelse:**  En brugeres mail er verificeret. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  En bruger som er oprettet i systemet, og brugeren har modtaget en e-mail verifikations e-mail fra systemet. |
| **Hovedforløb:**   1. Brugeren åbner e-mailen fra systemet og trykker på linket i denne. 2. Systemet viser en side, som fortæller at brugerens e-mail nu er verificeret. 3. Systemet gemmer oplysninger om verificering af brugers e-mail. 4. Brugeren kan nu fortsætte med at bruge systemet, som en bruger der er logget på systemet. |
| **Slut betingelser:**  Ingen. |

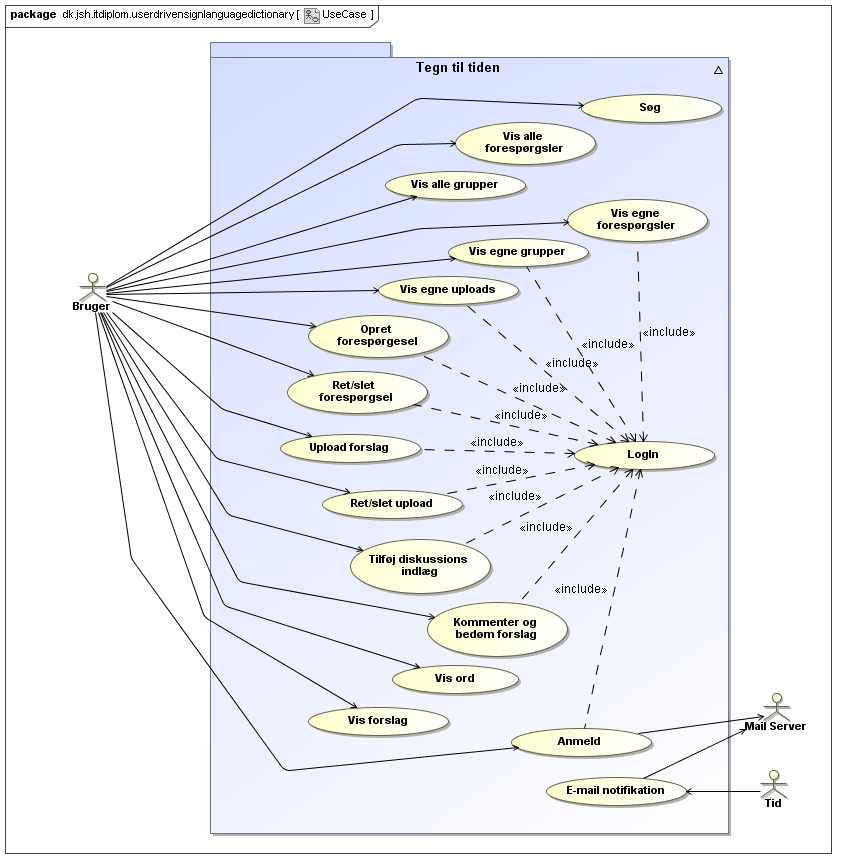
|  |
| --- |
| **Use case: Glemt password** |
| **ID: UC-4** |
| **Kort beskrivelse:**  Bestilling af nyt midlertidigt password. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  E-mail server. |
| **Start betingelser:**  En bruger som er oprettet i systemet med en valid e-mail. |
| **Hovedforløb:**   1. Brugeren trykker på linket ”Glemt password” på Login siden. 2. Systemet viser en side, hvor der skal indtastes den e-mail som er kendt af systemet for den aktuelle bruger. 3. Brugeren indtaster sin e-mail adresse og trykker på knappen ”Dan nyt password” 4. Systemet generer et nyt password til brugeren og sender dette med en e-mail tid denne. I e-mail skal der være en opfordring til at ændre passwordet. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Ret brugeroplysninger** |
| **ID: UC-5** |
| **Kort beskrivelse:**  Rette brugeroplysninger som navn og e-mail. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  E-mail server. |
| **Start betingelser:**  En bruger som er oprettet i systemet. |
| **Hovedforløb:**   1. include (Login) 2. Bruger trykker på linket ”Ret brugeroplysninger” linket. 3. Systemet viser en side med følgende felter: Navn, e-mail, password og gentag password. Navn og e-mail er forudfyldt. 4. Bruger indtaster sine rettelser. 5. Systemet gemmer rettelserne, hvis e-mail adressen er rettet så sender systemet en e-mail verifikations mail. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:**   * Brugeren fortryder. |



## Primære use cases

Følgende use case diagram og use case beskrivelser er systemet primære use cases.



Figur – Use case diagram

På de følgende sider er ovenstående use cases beskrevet.

|  |
| --- |
| **Use case: Søg** |
| **ID: UC-6** |
| **Kort beskrivelse:**  Søg efter et ord. |
| **Primære aktører:**  Alle. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. Brugeren navigerer til systemets startside. 2. Systemets startside indeholder et søgefelt samt et dropdown felt, hvor der vælges mellem ord eller gruppe. Ord er valgt som standard. I søgefeltet kan tegn udskriftes med wildcards \* for et eller flere tegn og ? for et tegn. 3. Brugeren udfylder søgefeltet. 4. Hvis ord er valgt i dropdown listen    1. Så søger systemet efter de ord som opfylder søgningen. 5. Hvis gruppe er valgt i dropdown listen    1. Så søger systemet efter alle de ord, som indgår i grupper, som opfylder søgningen. 6. Systemet viser en liste med links til de enkelte ord, sorteret alfabetisk. For hvert ord vises dato for upload, ord grupper. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Vis alle forespørgsler** |
| **ID: UC-7** |
| **Kort beskrivelse:**  Viser oversigt over alle ord som mangler forslag/uploads. |
| **Primære aktører:**  Alle. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. Brugeren vælger menu linket ”Alle forespørgsler”. 2. System viser en liste med ord som mangler forslag. Sorteret alfabetisk. For hvert ord vises hvornår det er oprettet og hvilke grupper det tilhører. Selve ordet er et link til en side med detaljer om ordet. Derudover er der et link til en upload side, for det aktuelle ord (dog kun for brugere som er indlogget, ellers vises linket som ikke aktivt.). |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:** |

|  |
| --- |
| **Use case: Vis alle Grupper** |
| **ID: UC-8** |
| **Kort beskrivelse:**  Viser oversigt over alle ord grupper. |
| **Primære aktører:**  Alle. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. Brugeren vælger menu linket ”Alle grupper”. 2. System viser en liste med grupper. Sorteret alfabetisk. For hver gruppe vises hvornår gruppen er oprettet. Selve gruppenavnet er et link til en side med detaljer om gruppen, samt en liste over de ord som indgår i gruppen. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:** |

|  |
| --- |
| **Use case: Vis egne forespørgsler** |
| **ID: UC-9** |
| **Kort beskrivelse:**  Viser oversigt over en brugers egne forespørgsler. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren vælger menu linket ”Mine forespørgsler”. 3. System viser en liste med ord som brugeren selv har ønsket forslag til. Sorteret alfabetisk. For hvert ord vises hvornår det er oprettet og hvilke grupper det tilhører. Selve ordet er et link til en side med detaljer om ordet. Derudover er der et link til en ret side, for det aktuelle ord. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:** |

|  |
| --- |
| **Use case: Vis egne grupper** |
| **ID: UC-10** |
| **Kort beskrivelse:**  Viser oversigt over en brugers egne grupper. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren vælger menu linket ”Mine grupper”. 3. System viser en liste med grupper som brugeren selv har oprettet. Sorteret alfabetisk. For hver gruppe vises hvornår gruppen er oprettet. Selve gruppenavnet er et link til en side med detaljer om gruppen, samt en liste over de ord som indgår i gruppen. Derudover er der et link til en ret side, for den aktuelle gruppe. (Ret siden er ikke beskrevet yderligere) |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:** |

|  |
| --- |
| **Use case: Vis egne uploads** |
| **ID: UC-11** |
| **Kort beskrivelse:**  Viser oversigt over en brugers egne uploads. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren vælger menu linket ”Mine uploads”. 3. System viser en liste med ord, som brugeren selv har uploadet forslag til. Sorteret alfabetisk. For hvert ord vises hvornår forslaget er uploadet. Selve ordet er et link til en side med detaljer om ordet. Derudover er der et link til en ret side, for det aktuelle upload. Hvis en bruger har oploadet flere forslag til et ord, så vil det samme ord gå igen på listen. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:** |

|  |
| --- |
| **Use case: Opret forespørgsel** |
| **ID: UC-12** |
| **Kort beskrivelse:**  Oprette en forespørgsel til en et ord |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren vælger linket ”Opret ny forspørgsel” på siden ”Mine forspørgsler”. 3. System viser en side med følgende felter: ”Ord der ønskes forslag til” samt ”Beskrivelse af ordet. 4. Brugeren udfylder begge felter og trykker ”Gem” eller ”Gem og tilknyt grupper”. 5. Hvis brugeren vælger ”Gem” så gemmer systemet forespørgslen, dato/tid for oprettelsen af denne, samt brugeren og returnerer til oversigtssiden. 6. Hvis brugen vælger ”Gem og tilknyt grupper”, så gemmer systemet forespørgslen og viser en ny side, hvor den kan tilknyttes eksisterende eller nye grupper. (Denne side er ikke beskrevet ydereligere) |
| **Slut betingelser:**  En forespørgsel er oprettet med eller uden tilknytning til grupper. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: ret/slet forespørgsel** |
| **ID: UC-13** |
| **Kort beskrivelse:**  Rette eller slette egne forespørgsler til en et ord. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren vælger et ”Ret” link fra listen på ”Mine forespørgsler” siden. Se use case Vis egne forspørgsler. 3. System viser en side med følgende felter: ”Ord der ønskes forslag til”, feltet er låst for rettelser hvis der er uploads til ordet, samt ”Beskrivelse af ordet. 4. Brugeren udfylder felterne med sine rettelser og trykker ”Gem”, ”Slet” eller ”Ret grupper”. Slet er kun vælges hvis, der ikke er uploadet forslag til det aktuelle ord. 5. Hvis brugeren vælger ”Gem” så gemmer systemet rettelserne og returnerer til oversigtssiden. 6. Eller vis brugeren vælger ”Slet” så slettes forespørgslen og der returneres til oversigtssiden. 7. Hvis brugen vælger ”Ret grupper”, så viser en ny side, hvor den kan fjernes og tilknyttes eksisterende grupper eller oprettes nye grupper. (Denne side er ikke beskrevet ydereligere) |
| **Slut betingelser:**  En forespørgsel er rettet/slettet. Tilknytning til grupper er evt. også rettet. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Upload forslag** |
| **ID: UC-14** |
| **Kort beskrivelse:**  Opload af et video forslag til et ord/forespørgsel. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen |
| **Hovedforløb:**   1. include (Login) 2. Brugeren vælger et ”Upload forslag” link fra ”Alle forespørgsler” side oversigten eller linket ”Upload forslag” fra ”Ord” siden, som vises hvis der trykkes på et ord fra søge oversigten m.fl. 3. Systemet viser en side, hvor der kan vælges en videofil ved at trykke på en ”Browse..” knap. Filnavnet vises derefter i et låst felt. Derudover er der et beskrivelsesfelt, som bruges til at beskrive indholdet på videoen. 4. Brugeren vælger en fil og indtaster en beskrivelse og trykker på knappen ”Upload og gem” 5. Systemet konvertere filen, til et HTML5 format og gemmer filen, beskrivelsen, dato/tid for upload og bruger. Derefter returnere systemet til oversigten brugeren kom fra. |
| **Slut betingelser:**  En videofil er uploadet som et forslag til et ord/forespørgsel. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: ret/slet upload** |
| **ID: UC-15** |
| **Kort beskrivelse:**  Rette eller slette egne videofil uploads til en et ord. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren vælger et ”Ret” link fra listen på ”Mine uploads” siden. Se use case Vis egne uploads. 3. System viser en side med selve videoen, samt beskrivelses feltet til denne. 4. Brugeren kan vælge at rette feltet og trykker ”Gem” eller ”Slet”. 5. Hvis brugeren vælger ”Gem” så gemmer systemet rettelsen og returnerer til oversigtssiden. 6. Eller hvis brugeren vælger ”Slet” så slettes videofilen der returneres til oversigtssiden. |
| **Slut betingelser:**  En uploaded videobeskrivelse er rettet eller videofilen og beskrivelsen er slettet. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Tilføj dissussionsindlæg** |
| **ID: UC-16** |
| **Kort beskrivelse:**  Tilføje et diskussionsindlæg til et ord. For at give mulighed for at diskutere den ”rette” måde at gengive ordet på tegnsprog. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren er på siden for et valgt ord. Enten via en søgning (se use casen Søg) eller via andre sider som Alle forespørgsler eller mine forespørgsler. 3. Brugeren vælger et ”Tilføj diskussionsindlæg” link. 4. Systemet viser en side med et tekstboks. 5. Brugeren udfylder tekstboksen med sit indlæg og trykker på en ”Gem” knap. 6. Systemet gemmer diskussionsindlægget, dato/tid for indlægget og brugeren som har oprettet indlægget. |
| **Slut betingelser:**  Et diskussionsindlæg er tilføjet til et ord. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Kommenter og bedøm forslag** |
| **ID: UC-17** |
| **Kort beskrivelse:**  Tilføje en bedømmelse og en kommentar til et forslag. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Det valgte forslag er ikke bedømt for af den aktuelle bruger. Dvs. at et forslag kun kan bedømmes en gang pr. bruger. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren er på siden for et valgt forslag. 3. Brugeren vælger et ”Tilføj bedømmelse” link. 4. Systemet viser en side med en tekstboks til en kommentar og 5 stjerner til bedømmelse. Når en bruger har musen over en af stjernerne så fremkommer en af følgende tekster: ”Dårlig”, ”Under middel”, ”Middel”, ”God” og ”Perfekt”. 5. Brugeren udfylder tekstboksen med sin bedømmelse og trykker på den stjerne som brugeren synes indlægget fortjener og trykker på en ”Gem” knap. 6. Systemet gemmer bedømmelsen, dato/tid og brugeren for denne. |
| **Slut betingelser:**  En bedømmelse er tilføjet til et forslag. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Vis ord** |
| **ID: UC-18** |
| **Kort beskrivelse:**  Vise et ord med beskrivelse. Liste med forslag og en liste med diskussionsindlæg. |
| **Primære aktører:**  Alle. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. Brugeren vælger et ord enten fra en søgning (se use case ”Søg”) eller fra siden ”Alle forespørgsler” (se use case Vis alle forespørgsler) eller fra siden ”Mine forespørgsler (se use case ”Vis egne forespørgsler”) m.fl. 2. Systemet viser en side med ordet, beskrivelse af dette samt en liste med forslag til ordet, sorteret efter bedømmelse (dvs. en gennemsnitlig bedømmelse af forlsaget) og en liste med diskussionsindlæg sorteret efter dato. 3. Hvis det er en bruger som er inlogget, er der mulighed for at afkrydse et felt, som indikere at den aktuelle bruger ønsker e-mail notificering når der er nye forslag til aktuelle ord. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Vis forslag** |
| **ID: UC-19** |
| **Kort beskrivelse:**  Vise et forslag med beskrivelse. Same en liste med bedømmelser. |
| **Primære aktører:**  Alle. |
| **Sekundære aktører:**  Ingen. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. Brugeren vælger et forslag enten fra en ord siden (se use case ”Vis ord”) eller fra siden ”Mine uploads” (se use case ”Vis egne uploads”). 2. Systemet viser en side hvor forslag til ordet kan afspilles, en beskrivelse af forslaget samt en liste med bedømmelser af forslaget sorteret efter dato. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: Anmeld** |
| **ID: UC-20** |
| **Kort beskrivelse:**  Anmeld et forslag/video for at være upassende. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  E-mail server, system administrator. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. include (Login) 2. Brugeren befinder på siden ”Vis forslag” se use case ”Vis forslag”. Her benytter brugeren ”Anmeld” linket. 3. Systemet viser en side, med en tekst boks til beskrivelse af det man finder upassende. 4. Brugeren udfylder ovenstående felt, og trykker på knappen ”Anmeld”. 5. Systemet sender en e-mail til at af systemet kendt system administrator, med følgende oplysninger: Videofile navn, uploadet af brugernavn samt anmeldt af brugernavn. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:**   * Brugeren fortryder. |

|  |
| --- |
| **Use case: E-Mail notifikation** |
| **ID: UC-21** |
| **Kort beskrivelse:**  Brugerne modtager hver mandag morgen ca. kl. 8 en e-mail, med links til de ord som har fået nye forslag og som de har ønsket e-mail notifikation på. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  E-mail server, system administrator. |
| **Start betingelser:**  Ingen. |
| **Hovedforløb:**   1. Systemet starter e-mail notifikation hver mandag kl. 8. 2. For alle brugere som har en eller flere ord de ønkser e-mail notifikation på.    1. Er der en eller flere ord som brugeren ønsker at blive notificeret om, som har nye upload siden forrige mandag.    2. Hvis ja, så danner og sender systemet en mail til brugeren, med links til de ord, som har fået nye uploads. |
| **Slut betingelser:**  Ingen. |
| **Alternative forløb:**  Ingen |

Brugeren rollen ”Administrator” er ikke beskrevet i nogle af de ovenstående use cases. Denne bruger skal kunne alt, det som de alm. brugere kan på deres egne sager,. Dvs. at administrator har ret til at rette på alt. For at gøre hele siden mere strømlinet. En redaktør med andre ord.

## Krav/use case matrix

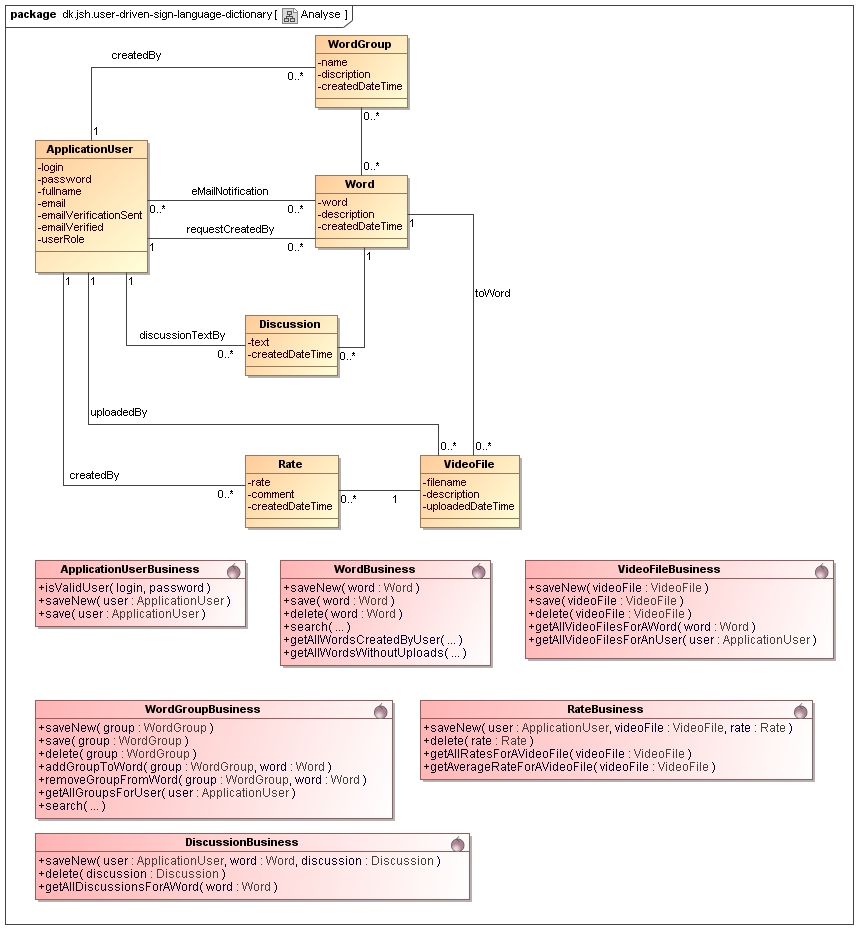
Følgende krav/use case matrix er lavet for at sikre at alle krav er behandlet i en eller flere use cases.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Krav: | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 |
| UC-1 |  | √ |  |  |  |  | √ | √ | √ |  |  |  |  |  |
| UC-2 |  |  | √ |  |  |  | √ |  |  |  |  |  |  |  |
| UC-3 |  |  | √ |  |  |  |  |  |  |  |  |  |  |  |
| UC-4 |  |  |  |  |  |  |  | √ |  |  |  |  |  |  |
| UC-5 |  |  |  |  |  |  |  |  | √ |  |  |  |  |  |
| UC-6 | √ |  |  |  |  |  |  |  |  |  |  | √ |  |  |
| UC-7 |  |  |  |  |  |  |  |  |  |  |  |  |  | √ |
| UC-8 |  |  |  |  |  |  |  |  |  |  |  |  |  | √ |
| UC-9 |  |  |  |  |  |  |  |  |  |  |  |  | √ |  |
| UC-10 |  |  |  |  |  |  |  |  |  |  |  |  | √ |  |
| UC-11 |  |  |  |  |  |  |  |  |  |  |  |  | √ |  |
| UC-12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC-13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC-14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC-15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC-16 |  |  |  |  |  |  |  |  |  |  | √ |  |  |  |
| UC-17 |  |  |  | √ | √ |  |  |  |  |  |  |  |  |  |
| UC-18 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC-19 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC-20 |  |  | √ |  |  | √ |  |  |  |  |  |  |  |  |
| UC-21 |  |  | √ |  |  |  |  |  |  |  |  |  |  |  |

Som det fremgår af ovenstående matrix så er krav 10, ikke beskrevet i nogen use case, dette er også i orden da, det omhandler system brugere, som skal oprettes direkte i databasen.

## Analysediagram

For at komme videre i analysefasen, har jeg udarbejdet følgende klassediagram.



Figur 3 - Analyse klassediagram

Alle de gule klasser er entitetsklasser. Her har jeg valgt ikke at have nogle get og set metoder, da disse ikke giver nogen værdi for diagrammet. Jeg har valgt at opdele data og forretningslogik i hver sit sæt klasser. Bl.a. fordi jeg har valgt at bruge Hibernate, til at mappe mellem den objekt orienteret verden og den relationelle database verden. Mere om det under design.

Den mest centrale entitetsklasserne er ApplicationUser som alle de andre entitetsklasser refererer til. ApplicationsUser indeholder oplysninger og systemes brugere. Word entitetsklassen benyttes til de ord som de enkelte brugere, ønsker forslag til. Det er også den der har e-mail notifikationslink til ApplicationUser. For at gruppere ord i ord grupper benyttes klassen WordGroup. Discussion klassen benyttes til diskussionsindlæg til et ord. Videofil klassen indeholder oplysninger om forslag som er uploadet til et ord. Og sidst men ikke mindst er der Rate som benyttes til at bedømme video uploads af forslag.

De lilla business klasser (også kaldet kontrolklasser, cirklen er en stereotype) er delt op i følgende klasser:

* ApplicationUserBusiness – til håndtering af brugere (opret, ret og login validering)
* WordBusiness – til håndtering af Word (Opret, ret, slet og søgning efter ord og grupper)
* WordGroupBusiness – til håndtering er WordGroup (Opret, ret, slet, tilføj ord til gruppe, fjern ord fra gruppe og find alle grupper som er oprettet af en bruger)
* DiscussionsBusiness – til håndtering af Discussions (Opret, slet og find indlæg til et ord)
* VideoFileBusiness – til håndtering af video file uploads (Opret, ret, slet, find alle upload til et ord og find en brugeres uploads)

Som det fremgår af overstående diagram, har jeg to typer klasser, de gule entitetsklasser og de røde kontrolklasser. Disse to typer har jeg valgt at placere i hver sin pakke, det vil fremgå af mine design diagrammer.

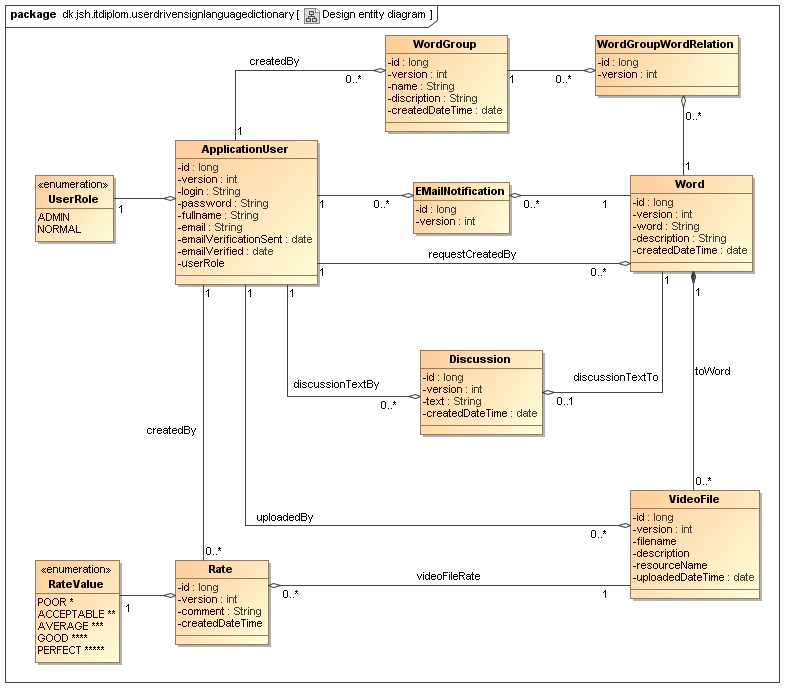
Jeg har valgt at ”låse” analysen, det skal forstås på den måde, at hvis jeg bliver ”klogere” senere i processen, f.eks. under design, så går jeg ikke tilbage og tilretter analysemodellerne, men indarbejder rettelserne i designmodellerne. Det mener jeg godt det kan forsvares da modellen er ret simpel.

For at afgrænse opgaven har jeg ikke medtaget nogen analyse use case realiseringer. Det mener jeg også kan forsvares da de fleste oprationer er meget simple læs, opret, ret og slet operationer også kendt som CRUD (Create, Read, Update, Delete) operationer.

Det afslutter så analyse delen, og bringer os videre til design delen.

# Design

Jeg er kommet frem til følgende klassediagram for entitetsklasserne.



Figur 4 - Design entitetsklassediagram

Ovenstående entitetsklasser placeres i en pakke kaldet:

dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity

Dette er måden man navngiver pakker på i Java.

I forhold til analyseklasse diagrammet, så er der to mange til mange relationer som er blevet til klasser, det et eMailNotification relationen som er blevet til klassen EMailNotification og relationen mellem Word og WordGroup som er blevet til WordGroupWordRelation.

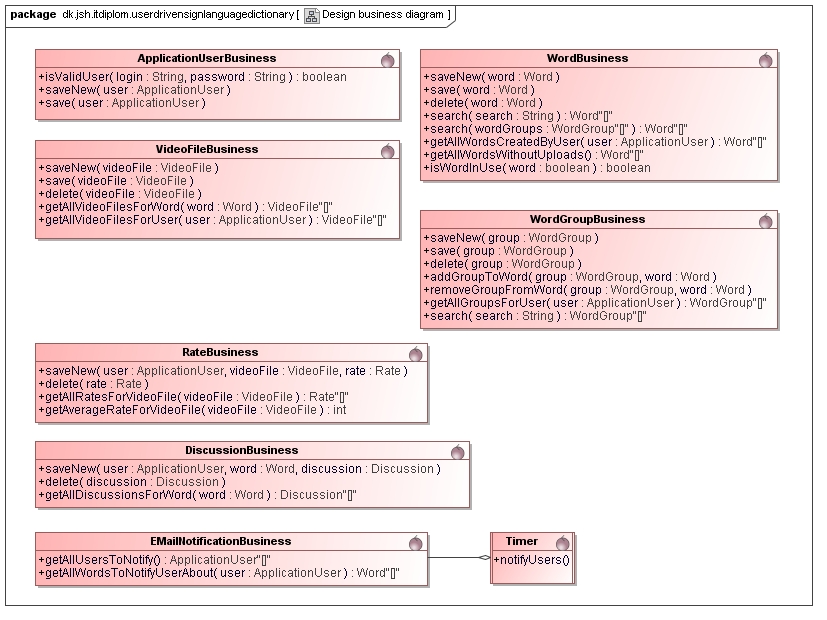
Så er det to attributter som jeg har lavet om til klasser med stereotype ”emumeration”. Det er attributten ”userRole” som er blevet til UserRole klassen, samt attributten rate fra Rate klassen som er blevet til RateValue klassen. Disse klasser laves som Java enums, som er en form for konstanter.

Derudover er alle associationer fra analysediagrammer rettet til enten aggregeringer eller kompositioner.

Alle ovenstående klasser, minus enumeration klasser, skal mærkes med Hibernate/Java annotations[[9]](#endnote-9). Så Hibernate ved hvordan de enkelte klasser, skal mappes til databasetabeller. Disse annotationer benyttes også til at generere et DDL[[10]](#endnote-10) database skema. Alle klasser har også fået it id og version attribut. Id bliver brugt som primær nøgle i database, og version bliver brugt i forbindelse med optimistisk låsning[[11]](#endnote-11).

Mine kontrolklasser kan ses af følgende diagram, som ligger i pakken:

dk.jsh.itdiplom.userdrivensignlanguagedictionary.business.



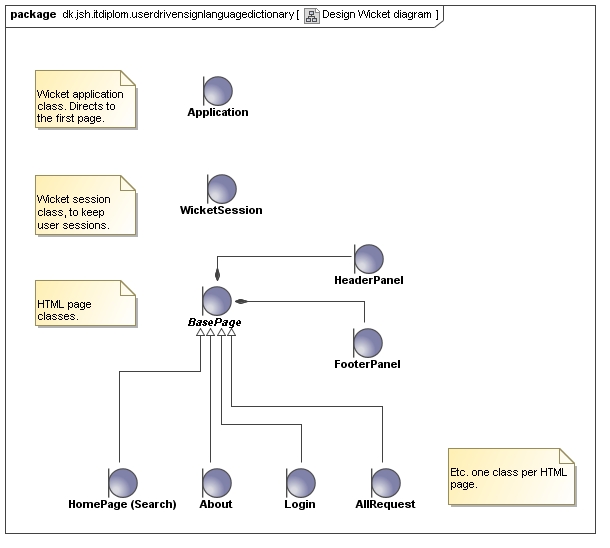
Figur 5 – Design kontrolklassediagram

Her har jeg tilføjet nogle ekstra klasser, bl.a. en Timer, som er en singleton[[12]](#endnote-12) klasse, der vha. EMailNotificationBusiness klassen (som også er ny i forhold til analyseklasse diagrammet) , starter e-mail notifikationen hver mandag morgen kl. 8. Dette skal foregå i en separat programtråd[[13]](#endnote-13).

Derudover har alle metoder nu fået attributter og evt. return værdier. De steder hvor der retuneres arrays, vil i java blive implementeret som lister, dvs. at der vil blive returneret List<type>. F. eks metoden search(String search) i klassen WordBusiness er i diagrammet beskrevet som om den retunere en array af Word objekter. Det vil i java blive implementeret som List<Word>.

Den næste pakke som jeg vil beskrive er de klasser som skal bruges til brugerflade programmeringen, disse klasser kommer til at ligge i pakken:

dk.jsh.itdiplom.userdrivensignlanguagedictionary.wicket



Figur – Design userinterface (boundary) diagram

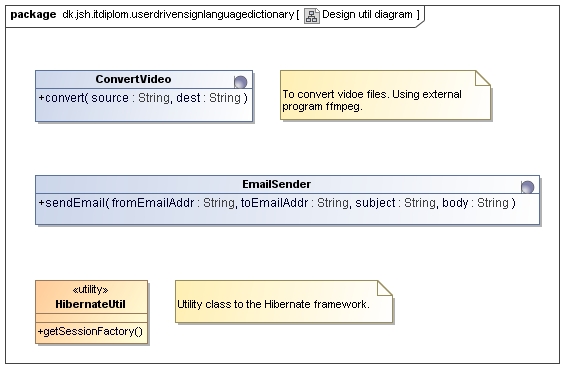
Her har jeg brug stereotypen ”boundary”, da det er brugerblade klasser. Jeg har ikke beskrevet nogle attributter til de enkelte klasser, diagrammet skal bruges til at beskrive strukturen i mine brugerflade klasser. Pakkenavet ender med .wicket for at indikere at det er dette java framework som benyttes. Jeg kommer lidt mere ind på dette framework senere.

En kort beskrivelse af de enkelte klasser:

* Application – Er Wicket’s main klasse, benyttes bl.a. til at pege på den første side brugerne skal møde.
* WicketSession – Er Wicket’s session klasse, til at holde de enkelte brugeres web sessioner.
* BasePage – Er en abstrakt klasse som de enkelte sider aver fra. Med følgende to klasser, som alle sider benytter
  + HeaderPanel – som beskriver toppen af alle sider
  + FooterPanel – som beskriver bunden af alle sider
* Og i bunden af diagrammet komme de enkelte sider, som alle aver fra BasePage. Ikke alle sider er med. Men Wicket frameworket ligger op til en klasse per side.

Den sidste pakke som jeg vil beskrive er pakken:

dk.jsh.itdiplom.userdrivensignlanguagedictionary.util



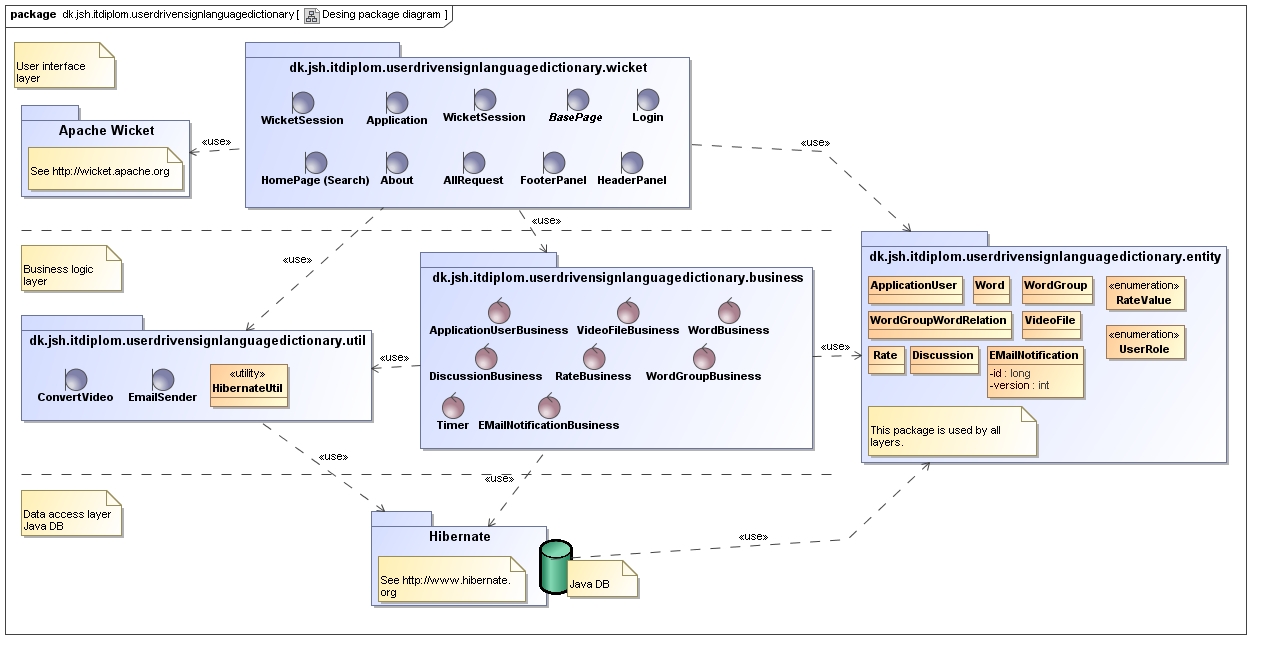
Figur – Design utility diagram

Denne pakke er en ”rode” kasse for klasser som ikke tilhøre de tre centrale pakker, som er beskrevet ovenover.

Følgende er en kort beskrivelse af de enkelte klasser:

* ConvertVideo – klasse til konvertering af videofiler fra f.eks. avi formatet til ogg formatet. Vha. det eksterne program FFMPEG[[14]](#endnote-14).
* EmailSender – klasse til at sende e-mails til en e-mail server. Jeg har benyttet Apache James[[15]](#endnote-15) som e-mail server under udviklingen. Denne server har jeg sat op til at rute alle e-mail modtagere til den samme e-mail modtager.
* HibernateUtil – klasse til Hibernate frameworket, som bruges til at få en Hibernate session (ikke en web session). Disse sessioner bruges til at samle database transaktionen, så det kan laves en commet eller en rollback hvis der er fejl.

Det samlede design kommer til at se således ud.



Figur 8 - Design pakkediagram

Som det ses af pakke diagrammet, er det opdelt i følgende 3 lag:

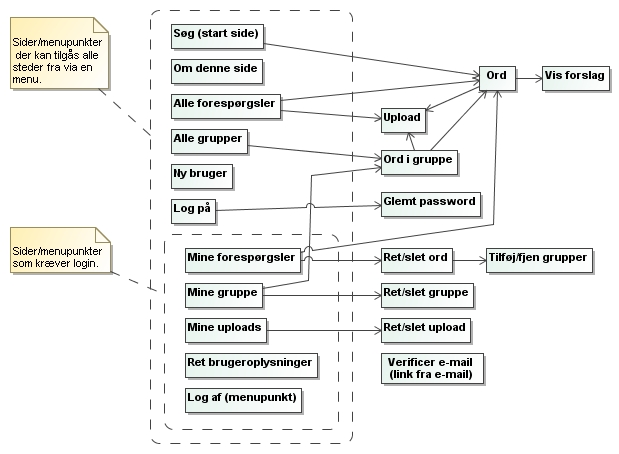
* User interface – til håndtering af websiderne.
* Business logic – som er til håndtering af forretningslogik.
* Data Access – som håndterer database delen. Vha. Hibernate.

Pakken og dk.jsh.itdiplom.userdrivensignlanguage.entity, er til transport klasser mellem lagene. Derudover skal klasserne i entity pakkes have tilføjet Hibernate annotations, så Hibernate kan finde ud at mappe disse klasser til tabeller i databasen. Hibernate kan også ud fra disse annotations danne database DDL skemafiler.

Pakkerne Apache Wicket og Hibernate, er med for at illustrere at disse to framework benyttes.

## Webside design

Jeg er kommet frem til følge webside flow i Tegn til tiden.

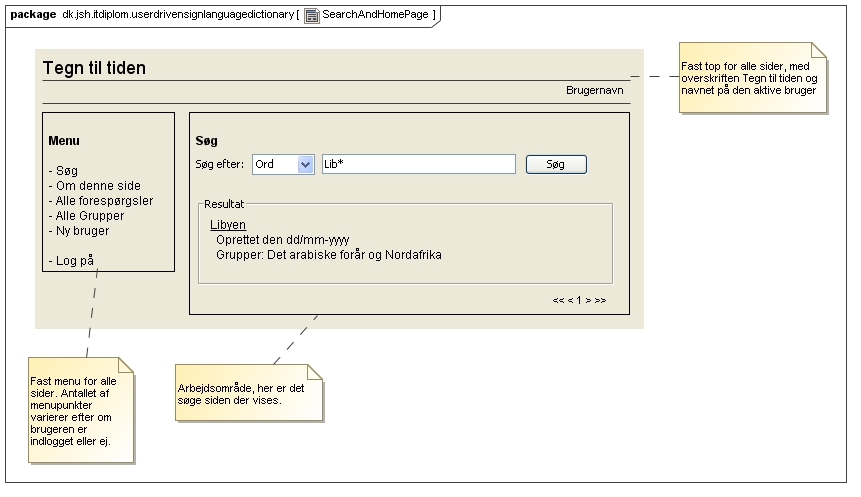


Figur 9 - Side flow

Som det fremgår af diagrammet, kan nogle sider/menupunkter tilgås af alle. Mens andre kræver login. Upload siden kræver også login. Verificer e-mail siden kan kun nås via et link fra en e-mail, som systemet har sendt til brugeren.

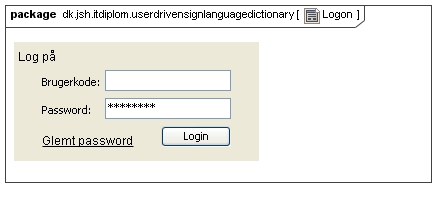
I det følgende vil jeg beskrive sideopbygningen for de vigtigste sider. Til dette har jeg også brugt MagicDraw, som har mulighed for at designe brugerflader. Det er derfor det minder lidt om en Windows brugerflade.

Det første side layout er startsiden, som også er søge siden.



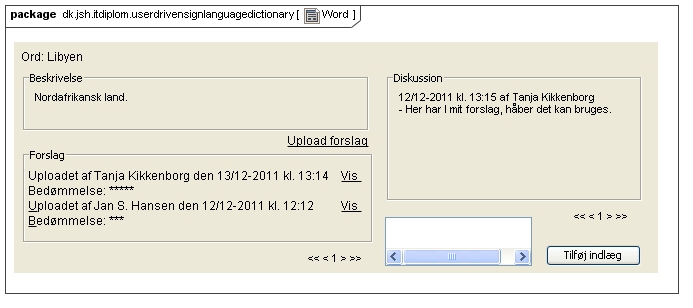
Figur – Start side

I det efterfølgende følger nogle eksempler på hvordan arbejdsområdet kan se ud. For at afgrænse opgaven, har jeg ikke medtaget alle sider. Det første arbejdsområde er ”Log på”.



Figur - Login side

Login siden, linker til en glemt password side, hvor burger bliver bed tom at indtaste sin e-mail adresse. Hvis denne er kendt af systemet, så sendes der en e-mail med et nyt password. Næste arbejdsområde er ”Ord”



Figur – Ord siden

Næste sider, vis siden og upload siden som ovenstående side linker til.



## Frameworks

Følgende afsnit er en kort gennemgang af de to frameworks jeg har valgt at benytte. Dvs. Wicket frameworket til webside programmering og Hibernate frameworket som bruges til at komme fra den objekt orienteret domaine til en relationel database.

### Wicket

På Wickets hjemmeside kan man læse om de mål, udviklerne har haft med Wicket[[16]](#endnote-16). Jeg vil her komme ind på de punkter, jeg selv som udvikler lægger vægt på.

Som udvikler har jeg arbejdet med Struts[[17]](#endnote-17) 1.3, samt med ASP.NET[[18]](#endnote-18), og en af de problemer som er i disse frameworks, er bl.a. at man blander kode med html.

I Struts arbejdes der med Java, JSP[[19]](#endnote-19) tags, HTML[[20]](#endnote-20) tags, JavaScript[[21]](#endnote-21) i en og samme fil. Det samme gør sig også gældende i ASP.NET, hvor det så bare er C#[[22]](#endnote-22) kode, ASP tags, html tags. Og det giver nogle filer som er svære at overskue, svære at genbruge og vedligeholde.

Helt grundlæggende prøver Wicket, at adskille html og kode. Det giver mulighed for, at det faktisk er muligt at få en HTML/CSS[[23]](#endnote-23) specialist, til at lave selve HTML/CSS koden. Som så kan overtages af en programmør. Det er ikke rigtigt muligt med f.eks. Struts og ASP.NET, da der er så mange specielle tags, som en html specialist ikke kender til, og som ikke kan håndteres af webdesigners værktøjer.

Wicket benytter også nogle gamle Java dyder som, en klasse, en Java fil. I Wicket kan en Wicket webside, beskrives vha. en HTML fil og en Java fil. Begge ligger i samme katalog og hedder det samme, på nær fil endelsen. Det eneste krav Wicket stiller til HTML filerne er, at de html elementer der skal være dynamiske, skal have en entydigt Wicket identifikation.

### Hibernate

Hibernate bruges til at komme fra den objekt orienterer verden til den relationelle databasen verden. Med Hibernate kan man selv bestemme om men vil starte med et database design eller med en nogle entitets klasser. Hvis man vælger at starte med et DDL skema, så kan Hibernate danne Java eller C# klasser som matcher skemaet. Eller omvendt som jeg har valgt at få lavet et DDL skema (se skema under Bilag TODO) ud fra mine Java entitetsklasser vha. annotations. (Se Bilag TODO for eksempler for dette).

En anden fordel ved Hibernate er at det er nemt at lave optimistisk låsning, som i korte træk består i, at systemet går ud fra, at de enkelte brugere ikke arbejder på samme data. Hvis der så er konflikter, så er det først til møllen princippet, der bestemmer, hvem der kommer af med sin opdatering. Det er derfor alle entitetsklasserne under design alle fik attributter version tilføjet. Så kan Hibernate vha. af denne attribut samt en annotation selv lave optimistisk låsning. Derudover giver Hibernate mulighed for at bruge HQL[[24]](#endnote-24), som er det samme som SQL[[25]](#endnote-25) med den krølle at i stedet for tabelnavne og kolonnenavne benyttes der klassenavne og klasse attributter. Se eksemplet i næste afsnit om sikkerhed.

F. eks. hvis bruger A læser en række med id = 1 og version = 1, og en bruger B læser den samme række. Bruger A opdatere rækken, hvor Hibernate øger version med 1, så den nu er 2. Nu vil bruger B også opdatere denne række, men bruger B får en fejl. Fordi Hibernate prøver at lave en update med følgende where sætning: ”where id = 1 and version = 1”. Da denne række ikke findes mere, vil Hibernate returnere en exception, som fanges af systemet, og fortæller brugeren at de data han forsøgte at gemme, er rettet af en anden bruger i mellemtiden.

Men som min applikation er designet, vil der ikke være mange konflikter. Da brugerne kun kan rette og slette egne ord, grupper m.fl. Det er kun i de sjældne tilfælde hvor en administrator retter i det samme som en alm. bruger.

## Sikkerhed

Wicket er ifølge wickets hjemmeside[[26]](#endnote-26) ”secure by default”. Og jeg har heller ikke kunne fremprovokere diverse ”Injection flaws”[[27]](#endnote-27) angreb.

SQL injections forebygger også vha. Hibernate’s måde at lave HQL statement på, f.eks som i følgende kode:

StringBuilder hql = new StringBuilder();

hql.append("select word from ");

hql.append("dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.");

hql.append("Word word ");

if (useLike) {

hql.append("where lower(word.word) like :search ");

}

else {

hql.append("where lower(word.word) = :search ");

}

hql.append("order by word.word");

Query query = session.createQuery(hql.toString());

query.setString("search", search);

Som det fremgår af koden, bliver ingen variabler direkte indsat i SQL strengen, men via metoden query.setString(”search”, search).

Med hensyn til adgangskontrol vil jeg anbefale at benytte HTTPS/SSL[[28]](#endnote-28) under selve login processen, for at sikre at login og password ikke bliver opsnappet. Men derudover mener jeg ikke, at der er noget i denne Webapplikation, som er så følsomt at det kræver HTTPS.

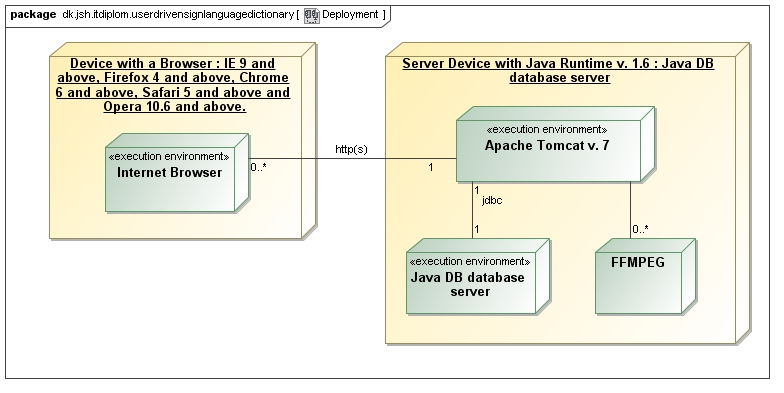
I databasen bør password ikke stå i klar tekst, men som en hashed[[29]](#endnote-29) værdi, som kun kan benyttes til at validere et password.

OWASP siden [www.owasp.org](https://www.owasp.org/index.php/Main_Page) er et rigtigt god side om sikkerheds problematikker i webapplikationer.

# Idriftsættelse

Da Tegn til tiden er en Java webapplikation, uden afhængigheder til andre systemer. Så er selve idriftsættelsen enkel. Det kræver en server med installeret Java og en Apache Tomcat Web Server, samt en Java DB. Java DB er en del af Java. Dvs. at der ikke er nogen specielle krav til operativsystem, da Tomcat og Java DB kan køre på bl.a. Linux, Unix og Windows. Andre webservere kunne også benyttes. Det eneste som ikke er en del af Java verdenen er video konverteringsprogrammer FFMPEG. Programmet er open source og findes både til Linux og Windows.

Følgende er et deployment diagram.



Figur 13 - Deployment diagram

For at deploye en webapplikation i en Tomcat server, bygges der en WAR[[30]](#endnote-30) fil, vha. et udviklingssystem. Og denne WAR fil kopieres over i et specielt Tomcat applikationskatalog. Første gang systemet installeres skal databasen klargøres. Dette gøres med et SQL-script, som opretter alle tabellerne, samt indsætter en administratorbruger. Så denne bruger kan begynde at oprette brugere og produkter i systemet.

Som det også fremgår af ovenstående diagram, skal Tegn til tiden virke sammen med gængse browsere på markedet. Dvs. Internet Explore, Firefox, Opera, Chrome og Safari. Det vil kræve noget test. Men da antallet af antal websider er begrænset, er det en overkommelig opgave. Men for at HTML5 video tagget skal virke, kræver det nyere Browsere. Versioner for disse fremgår af diagrammet.

# UP Iterationer

todo

Indtil nu har denne rapport fuldt den meget udskældte vandfaldsmodel. Dvs. først indsamles der krav, disse analyseres, hvorefter systemet designes. Dette er også en logisk form, hvis et system skal beskrives i rapportformat. Men for at komme tilbage til UP, har jeg valgt at beskrive hvilke indledende iterationer, som BaRI kunne starte sin udvikling efter.

|  |  |
| --- | --- |
| **Iteration** | **Beskrivelse** |
| 1 | Helt basal forretnings-logik til at hente brugere og produkt/modul oplysninger fra DB. Data til disse kan indsættes manuelt i databasen til at starte med. |
| 2 | Oprette et ønske, samt simpel mulighed for at fremsøge denne igen. Herfra kan de enkelte iterationer demonstreres. |
| 3 | Som 2 bare med en fejl. |
| 4 | Tilføje diskussionsindlæg til fejl og ønsker. Og vise disse igen. |
| 5 | Tilføje filer til fejl og ønsker. Og vise disse igen. |
| 6 | Tilføje filer til diskussionsindlæg. Og vise disse igen. |
| 7 | Tilføje e-mail notifikation. |
|  | etc. |

En produktejer kan løbende justere disse iterationer, efter hvad der giver mest værdi. Derfor bør alle iterationer ikke planlægges på forhånd, for at give mulighed for at justere undervejs. For hver af disse iterationer skulle BaRI løbende analyseres, designes, implementeres og testes.

# Konklusion

Denne opgave har mere været en øvelse i den gamle vandfaldsmodel. For som jeg har skrevet tidligere, at få opgaven til passe ind i rapportformen. Der nødvendigvis må starte med krav og analyse og fortsætte med design og implementering. En anden grund til at UP ikke er fulgt, er at der ikke har været noget krav om udvikling. Og noget kørende kode, må være en forudsætning for iterative udviklingsprocesser. Man bør kunne vise et kørende program, til interessenterne, for at blive klogere på, om man er på rette vej.

Jeg er heller ikke sikker på, at UP er den rette proces for alle projekter. I dette tilfælde kunne XP[[31]](#footnote-2) eller SCRUM[[32]](#footnote-3) godt være brugt. UP har efter min mening mere sin berettigelse, hvis der er høje dokumentationskrav til projektet/produktet. Fordi der ligges stor vægt på UML diagrammer og tegninger i alle faser, samt hvilke dokumenter der er en forudsætning for at komme videre til næste fase i processen. Men til forskel for vandfaldsmodellen, så er UP dokumenterne noget der tilrettes for hver iteration.

Processer som XP og SCRUM ligger mere sin vægt på tests, simpelt design, refactoring[[33]](#footnote-4) og kørende kode, som kan bruges her og nu ellers i det mindste demonstreres.

Til slut har jeg et enkelt kritikpunkt til bogen UML and the Unified Process. Da denne helt forbigår brugerfladedesign, som jeg mener, er en vigtigt fase. Da det er her man, som designer, opdager mange fejl og mangler ved sit design.

# Bilag

## Danske/engelske termer

Følgende tabel er en liste af de termer som benyttes af systemet. Alle termer får et dansk og et engelsk navn. Bl.a. for at sikre overgangen fra use-cases til analyse, design og programmering hvor de engelske termer benyttes.

|  |  |  |
| --- | --- | --- |
| Dansk term | Engelsk term | Beskrivelse |
| Bruger | User | En bruger af systemet. |
| Ord | Word | Et ord som der kan uploades forslag til |
| Forespørgsel | Request | Forespørgsler til ord, som en bruger ønsker forslag til. |
| Forslag | Proposals | Forslag til et ord |
| Video fil | Video file | Video forslag til et ord |
| Bedømmelse | Rate | Video bedømmelse |
| Ord gruppe | Word group | Ord gruppe |
| Diskussion | Discussion | Diskussions indlæg |
| Brugerrolle | User role | Bruger roller |

## 



## Brugervejledning

Følgende er brugervejlidening/gemmengang af det jeg har nået at programmere.

TODO



## Kode

todo



## Indhold på den vedlagte cd

Indholdet på den vedlagte CD er inddelt i følgende 3 kataloger:

* Rapport – Indeholder denne rapport i Word 2007 og PDF format.
* MagicDraw – Indeholder 3 MagicDraw projekter. Et for analyse, et for Design og implementering samt et for websider.
* Kode – Indeholder al kode til projektet

# Noter

1. Java – Programmeringssprog, se [en.wikipedia.org/wiki/Java\_%28programming\_language](http://en.wikipedia.org/wiki/Java_%28programming_language%29). [↑](#endnote-ref-1)
2. Apache Wicket - Java Web framework, se [en.wikipedia.org/wiki/Apache\_Wicket](http://en.wikipedia.org/wiki/Apache_Wicket). [↑](#endnote-ref-2)
3. Hibernate - Java Object til database framework, se [en.wikipedia.org/wiki/Hibernate](http://en.wikipedia.org/wiki/Hibernate_%28Java%29). [↑](#endnote-ref-3)
4. JavaDB - Java Database, som er en del af standard java, se [en.wikipedia.org/wiki/Apache\_Derby](http://en.wikipedia.org/wiki/Apache_Derby). [↑](#endnote-ref-4)
5. Apache Tomcat – Web server, se [en.wikipedia.org/wiki/Apache\_Tomcat](http://en.wikipedia.org/wiki/Apache_Tomcat). [↑](#endnote-ref-5)
6. UML – Unified Modeling Language, se [en.wikipedia.org/wiki/Unified\_Modeling\_Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language). [↑](#endnote-ref-6)
7. Use case – System/krav beskrivelser, se [en.wikipedia.org/wiki/Use\_case\_diagram](http://en.wikipedia.org/wiki/Use_case_diagram). [↑](#endnote-ref-7)
8. MagicDraw – UML tegneprogram, se [en.wikipedia.org/wiki/MagicDraw](http://en.wikipedia.org/wiki/MagicDraw). [↑](#endnote-ref-8)
9. Java Annotations – mærkning af java klasser, se [en.wikipedia.org/wiki/Java\_annotation](http://en.wikipedia.org/wiki/Java_annotation). [↑](#endnote-ref-9)
10. DDL – Data Definition Language, se [en.wikipedia.org/wiki/Data\_Definition\_Language](http://en.wikipedia.org/wiki/Data_Definition_Language). [↑](#endnote-ref-10)
11. Optimistisk låsning, se <http://en.wikipedia.org/wiki/Optimistic_concurrency_control>. [↑](#endnote-ref-11)
12. Singleton – Design mønster, som sikre at der kun kan findes en instans af klassen, se [en.wikipedia.org/wiki/Singleton\_pattern](http://en.wikipedia.org/wiki/Singleton_pattern). [↑](#endnote-ref-12)
13. Programtråd – en parallel program tråd, som kører på samme tid som hovedtråden, se [en.wkipedia.org/wiki/Thread\_(computer\_science)](http://en.wikipedia.org/wiki/Thread_%28computer_science%29). [↑](#endnote-ref-13)
14. FFMPEG – et video format konverteringsprogram, se [ffmpeg.org](http://ffmpeg.org/). [↑](#endnote-ref-14)
15. Apache James – Email server, se [james.apache.org](http://james.apache.org/). [↑](#endnote-ref-15)
16. Why Wicket, se [wicket.apache.org/introduction.html](http://wicket.apache.org/introduction.html). [↑](#endnote-ref-16)
17. Struts - Web framework, se [struts.apache.org](http://struts.apache.org/) [↑](#endnote-ref-17)
18. ASP.NET – Web framework, se [en.wikipedia.org/wiki/ASP.NET](http://en.wikipedia.org/wiki/ASP.NET) [↑](#endnote-ref-18)
19. JSP – Java Server Pages, se [en.wikipedia.org/wiki/JSP](http://en.wikipedia.org/wiki/JSP) [↑](#endnote-ref-19)
20. HTML – Hyper Text Makeup Language, se [en.wikipedia.org/wiki/HTML](http://en.wikipedia.org/wiki/HTML) [↑](#endnote-ref-20)
21. JavaScript – programmeringssprog, se [en.wikipedia.org/wiki/JavaScript](http://en.wikipedia.org/wiki/JavaScript) [↑](#endnote-ref-21)
22. C# - C Sharp programmeringssporg, se [en.wikipedia.org/wiki/C\_Sharp\_programming\_language](http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29) [↑](#endnote-ref-22)
23. CSS – Cascading Style Sheets, se <http://en.wikipedia.org/wiki/Cascading_Style_Sheets> [↑](#endnote-ref-23)
24. HQL - Hibnernate Query Language, se [docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html](http://docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html) [↑](#endnote-ref-24)
25. SQL - Structured Query Language, se [en.wikipedia.org/wiki/SQL](http://en.wikipedia.org/wiki/SQL) [↑](#endnote-ref-25)
26. Wicket secure by default - se mere [wicket.apache.org/meet/features.html](http://wicket.apache.org/meet/features.html). [↑](#endnote-ref-26)
27. Se OWASP Top 10, se [www.owasp.org/index.php/Top\_10\_2007](http://www.owasp.org/index.php/Top_10_2007). [↑](#endnote-ref-27)
28. HTTPS/SSL, se [en.wikipedia.org/wiki/Transport\_Layer\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security). [↑](#endnote-ref-28)
29. Password Hashed – se [www.owasp.org/index.php/Guide\_to\_Authentication#Password\_Guidelines](https://www.owasp.org/index.php/Guide_to_Authentication#Password_Guidelines). [↑](#endnote-ref-29)
30. WAR – Web application ARchive, se [en.wikipedia.org/wiki/WAR\_file\_format\_(Sun)](http://en.wikipedia.org/wiki/WAR_file_format_%28Sun%29) [↑](#endnote-ref-30)
31. XP – eXtreme Programmering se mere her [en.wikipedia.org/wiki/Extreme\_Programming](http://en.wikipedia.org/wiki/Extreme_Programming) og [www.extremeprogramming.org/](http://www.extremeprogramming.org/) [↑](#footnote-ref-2)
32. SCRUM – se mere her [en.wikipedia.org/wiki/Scrum\_(development)](http://en.wikipedia.org/wiki/Scrum_%28development%29) og [www.scrumalliance.org/learn\_about\_scrum](http://www.scrumalliance.org/learn_about_scrum) [↑](#footnote-ref-3)
33. Refactoring – se mere [en.wikipedia.org/wiki/Code\_refactoring](http://en.wikipedia.org/wiki/Code_refactoring) [↑](#footnote-ref-4)