IT-DIPLOMUDDANNELSEN

AFGANGSPROJEKT

af

Jan Schrøder Hansen

eFTERÅR/Vinter 2011

**INDHOLD**

[1. Indledning 3](#_Toc311399774)

[2. Opgavebeskrivelse 3](#_Toc311399775)

[3. Krav 5](#_Toc311399776)

[3.1. Funktionelle Krav 5](#_Toc311399777)

[3.2. Ikke funktionelle krav 7](#_Toc311399778)

[4. Analyse 8](#_Toc311399779)

[4.1. Administrative use cases 8](#_Toc311399780)

[4.2. Primære use cases 12](#_Toc311399786)

[4.3. Krav/use case matrix 22](#_Toc311399787)

[4.4. Analysediagram 23](#_Toc311399788)

[5. Design 25](#_Toc311399789)

[5.1. Webside design 31](#_Toc311399790)

[5.2. Frameworks 37](#_Toc311399793)

[5.3. Sikkerhed 38](#_Toc311399794)

[6. Idriftsættelse 39](#_Toc311399795)

[7. UP Iterationer 40](#_Toc311399796)

[8. Konklusion 41](#_Toc311399797)

[9. Bilag 42](#_Toc311399798)

[9.1. Danske/engelske termer 42](#_Toc311399799)

[9.2. Brugervejledning 43](#_Toc311399805)

[9.3. Udviklingsmiljø 56](#_Toc311399806)

[9.4. Kode 57](#_Toc311399809)

[9.5. Indhold på den vedlagte cd 81](#_Toc311399811)

[10. Noter 82](#_Toc311399812)

# 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 . 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.

Jeg vil arbejde efter UP[[8]](#endnote-8) (Unified Process), som er en iterativ udviklingsproces. Men denne rapport vil følge den gamle ”Vandfalds model”, da det giver et naturligt flow i beskrivelsen af systemet.

Alle diagrammer er udarbejdet vha. af programmer MagicDraw[[9]](#endnote-9).

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, opret ny bruger og ret bruger. | 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 |

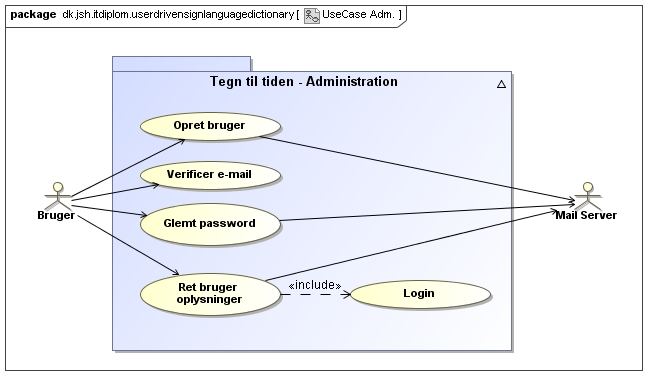
Det afslutter krav indsamlingen, og det bringer os videre til analyse fasen.

# 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. |

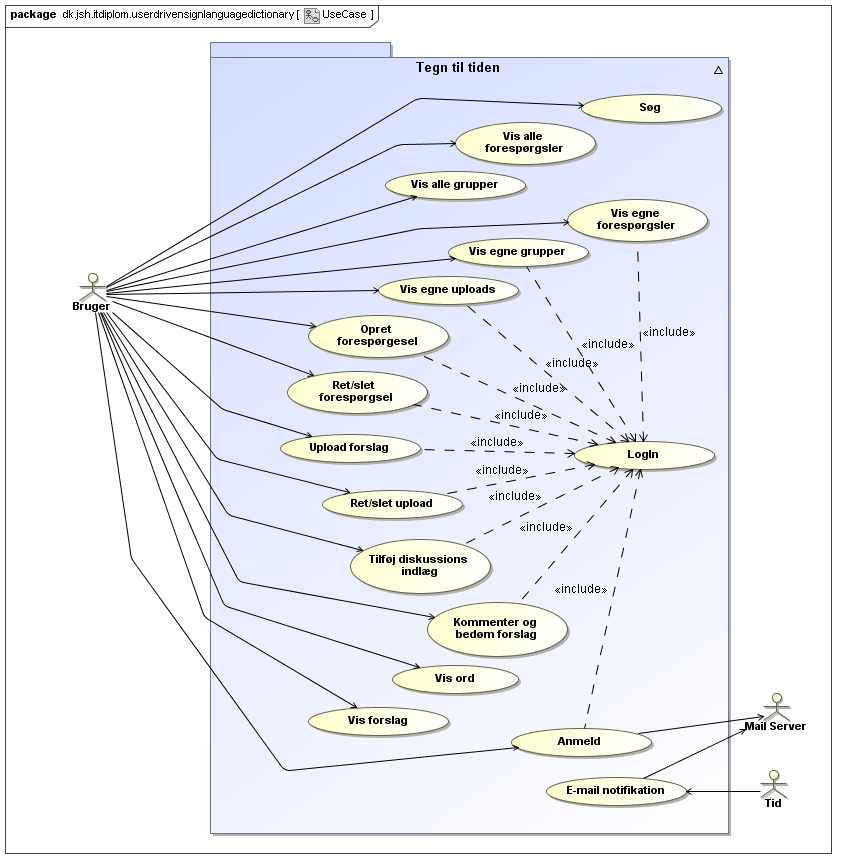
|  |
| --- |
| **Use case: Glemt password** |
| **ID: UC-4** |
| **Kort beskrivelse:**  Bestilling af nyt midlertidigt password. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  E-mail server. |
| **Start betingelser:**  En bruger som er oprettet i systemet med en valid e-mail. |
| **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, brugerkode, e-mail, password og gentag password. Navn, brugerkode og e-mail er forudfyldt. Brugerkode kan ikke rettes. 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 som upassende” 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 ”Tegn til tiden” siden mere strømlinet. En redaktør med andre ord.

## Krav/use case matrix

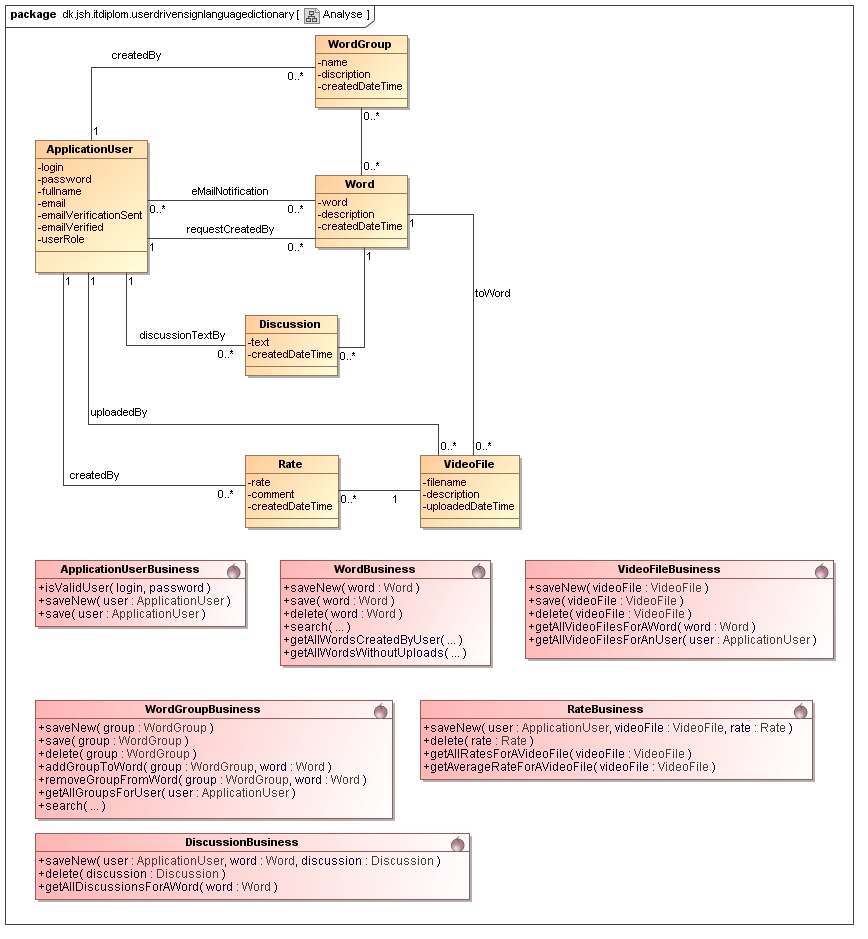
Følgende krav/use case matrix er lavet for at sikre at alle krav er behandlet i en eller flere use cases.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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.

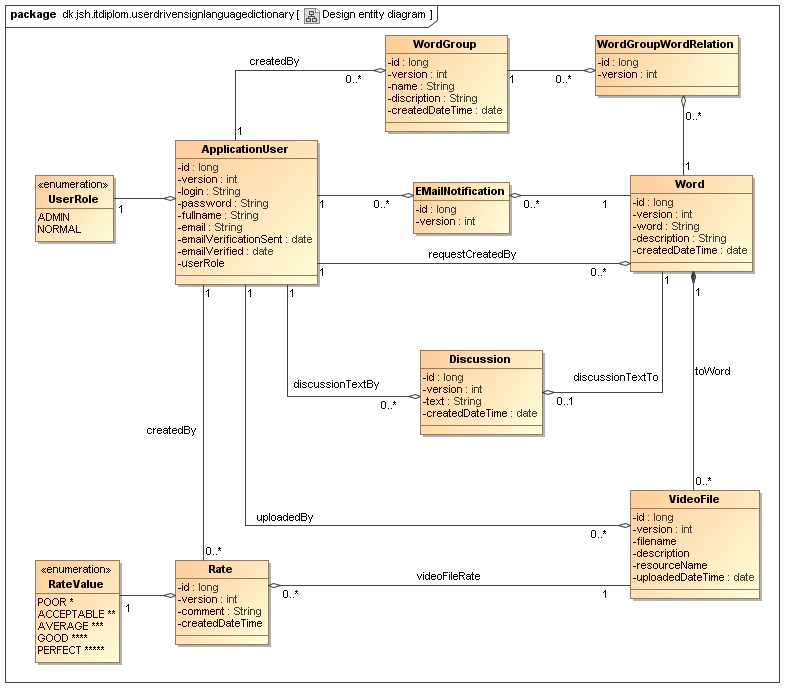
Jeg har valgt at ”låse” analysen, det skal forstås på den måde, at hvis jeg bliver ”klogere” senere i processen, f.eks. under design, så går jeg ikke tilbage og tilretter analysemodellerne, men indarbejder rettelserne i designmodellerne. Det mener jeg godt det kan forsvares da modellen er ret simpel.

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 er eMailNotification relationen som er blevet til klassen EMailNotification og relationen mellem Word og WordGroup som er blevet til WordGroupWordRelation.

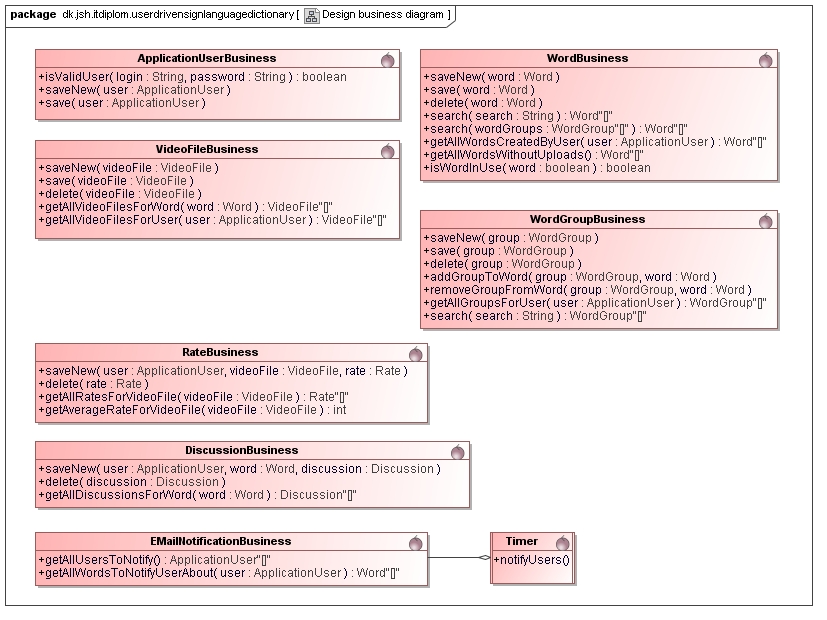
Så er det to attributter som jeg har lavet om til klasser med stereotype ”emumeration”. Det er attributten ”userRole” som er blevet til UserRole klassen, samt attributten rate fra Rate klassen som er blevet til RateValue klassen. Disse klasser laves som Java enums, som er en form for konstanter.

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

Alle ovenstående klasser, minus enumeration klasser, skal mærkes med Hibernate/Java annotations[[10]](#endnote-10). Så Hibernate ved hvordan de enkelte klasser, skal mappes til databasetabeller. Disse annotationer benyttes også til at generere et DDL[[11]](#endnote-11) 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[[12]](#endnote-12).

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

dk.jsh.itdiplom.userdrivensignlanguagedictionary.business.



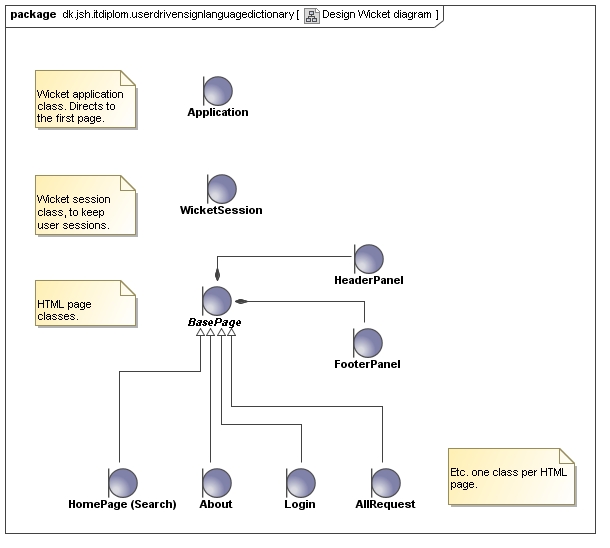
Figur 5 – Design kontrolklassediagram

Her har jeg tilføjet nogle ekstra klasser, bl.a. en Timer, som er en singleton[[13]](#endnote-13) 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[[14]](#endnote-14).

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

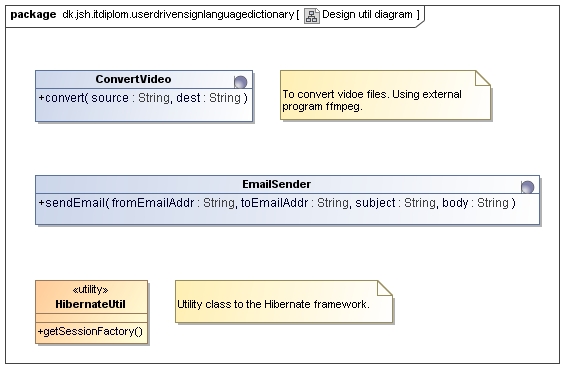
Her har jeg brug stereotypen ”boundary”, da det er brugerblade klasser. Jeg har ikke beskrevet nogle attributter til de enkelte klasser, diagrammet skal bruges til at beskrive strukturen i mine brugerflade klasser. Pakkenavet ender med .wicket for at indikere at det er dette java framework som benyttes.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



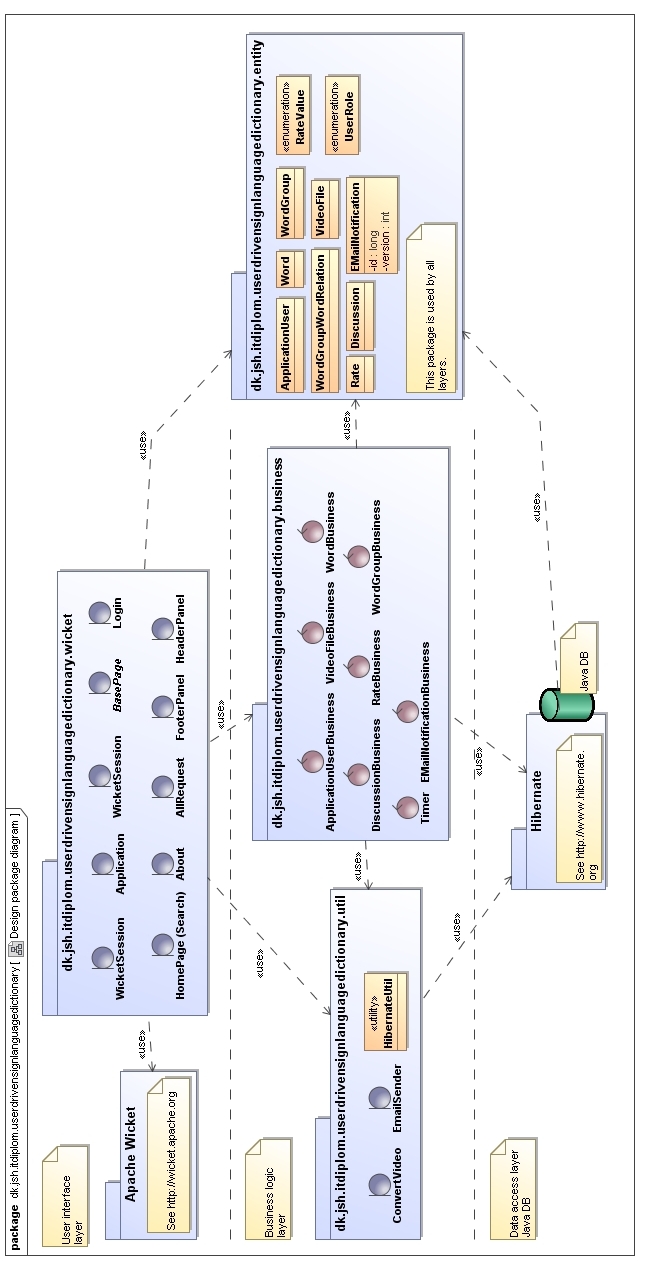
Figur – Design utility diagram

Denne pakke er en ”rode” kasse for klasser som ikke tilhører 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[[15]](#endnote-15).
* EmailSender – klasse til at sende e-mails til en e-mail server. Jeg har benyttet Apache James[[16]](#endnote-16) 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 pakkediagrammet, er det opdelt i følgende 3 lag:

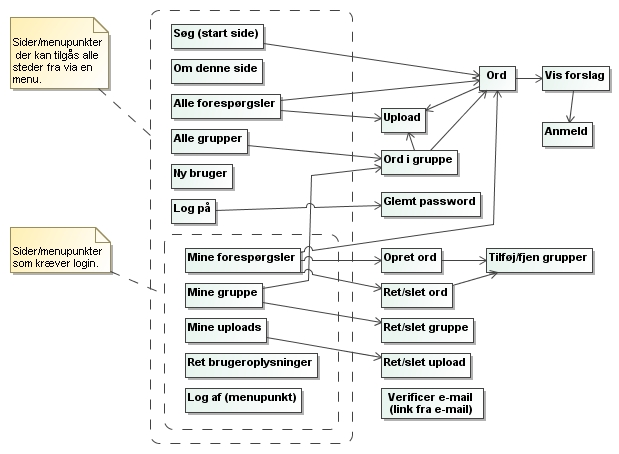
* User interface – til håndtering af websiderne.
* Business logic – som er til håndtering af forretningslogik.
* Data Access – som håndterer database delen. Vha. Hibernate.

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.

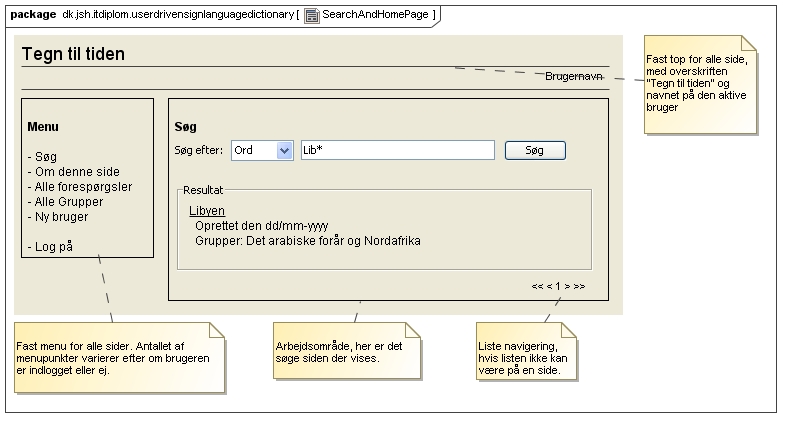


Figur 9 - Side flow

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.

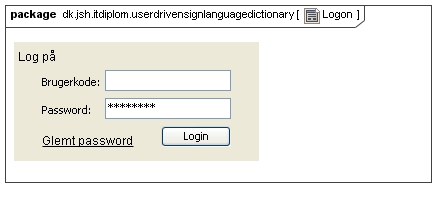
I det følgende vil jeg beskrive sideopbygningen for de vigtigste sider. Til dette har jeg også brugt MagicDraw, som har mulighed for at designe brugerflader. Det er derfor de følgende sidelayouts minder lidt om en Windows brugerflader.

Det første side layout er startsiden, som også er søgesiden.



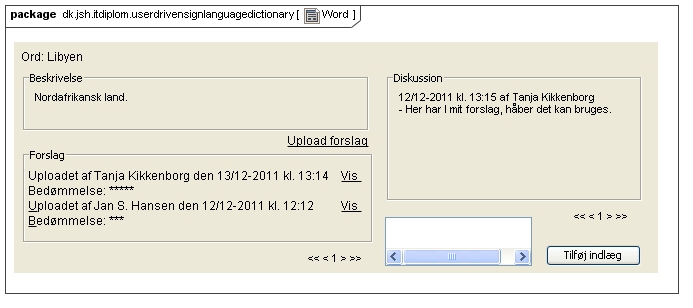
Figur – Start side

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

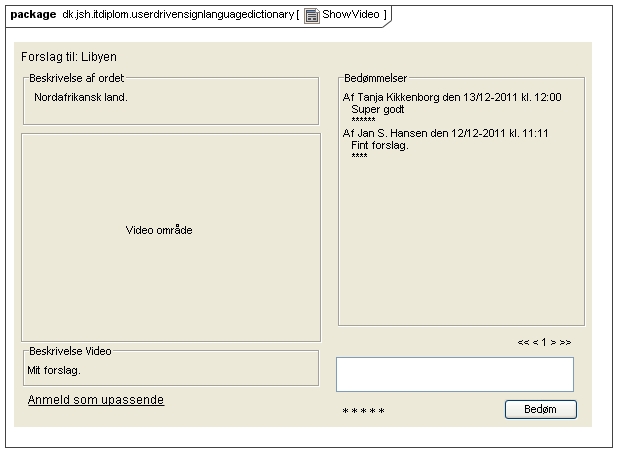
Login siden, linker til en ”Glemt password” side, hvor burger bliver bedt om at indtaste sin e-mail adresse. Hvis denne er kendt af systemet, så sendes der en e-mail med et nyt password. Næste arbejdsområde er ”Ord”.



Figur – Ord side

Bedømmelsen på denne side, er et gennemsnit af alle bedømmelser til den aktuelle video. Man kunne overveje at bruge to faneblade til ovenstående side, så Diskussion delen får en fane for sig selv. På de diskussionsindlæg man selv har lavet, vil der være et slet link. Muligheden for at tilføje indlæg for indloggede brugere.

Næste sider, ”Vis forslag” siden og ”upload” siden som ovenstående side linker til.



Figur – Vis forslag side

Her kunne man også opdele siden i to faneblade, så bedømmelsesdelen får sin egen fane. Også her vil der være et slet link på den aktuelle brugers bedømmelser. Muligheden for at bedømme videoen og anmelde som upassende vil kun være for indloggede brugere. ”Anmeld som upassende” linker til en ny side med en enkelt tekstboks og en Anmeld knap. De 5 stjerner skal indikere hvordan man bedømmer, ved at vælge det antal stjerner som man mener video fortjener. Som man kender det for mange andre web sider. Når man ”hover” med musen over en af stjerner, skal følgende tekster fremkomme: ”Dårlig”, ”Under middel”, ”Middel”, ”God” og ”Perfekt”.



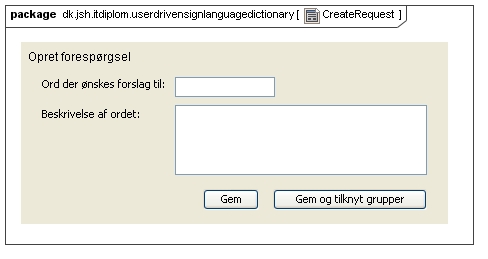
Figur – Upload video side

Den næste side er ”Mine forespørgsler”, den bruges også som udgangspunkt for de andre ”liste” sider, som er ”Mine grupper”, ”Mine uploads”, ”Alle forespørgsler” og ”Alle grupper”.



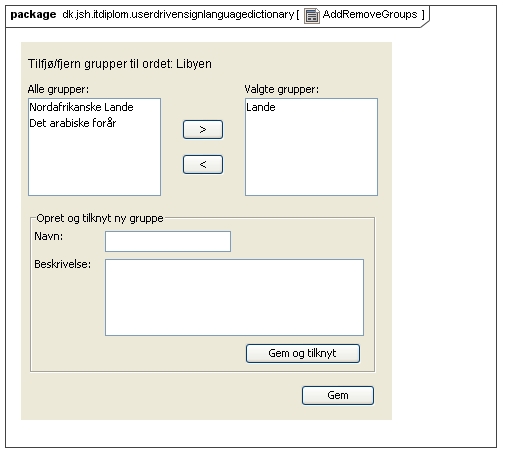
Figur – Mine forespørgsler side

Her har man mulighed for at oprette nye forespørgsler vha. ”Opret ny forespørgsel” linket. I listen har hver forespørgsel to links, selve ordet som linker til ”Ord” siden, og ”Ret” siden som linker til en ny side hvor forespørgslen kan rettes.



Figur – Opret forespørgsel side

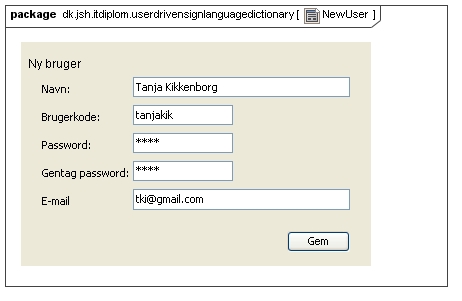
Ovenstående side er til oprettelse af en ny forespørgsel. Ret/slet forespørgsel siden være som denne bare med en ”Slet” knap. Som ikke kan benyttes hvis der er uploads til ordet. Den sidste side jeg vil gennemgå er ”Tilføj/fjern grupper” som man kommer til ved at trykke på knappen ”Gem og tilknyt grupper”.



Figur – Tilføj/fjern grupper side

Denne side benyttes til at tilføje og fjerne grupper til en ord. Vha. to lister, en med alle ikke valgte grupper og en med valgte grupper. Så har brugeren mulighed for at flytte grupper frem og tilbage med de ”<” og ”>” knapperne. Hvis musen holdes stille over en af grupperne i de to lister, så fremkommer gruppebeskrivelsen. Siden giver også mulighed for at oprette nye grupper.

Jeg vil slutte af med ”Opret ny bruger” og verificer e-mail siden.

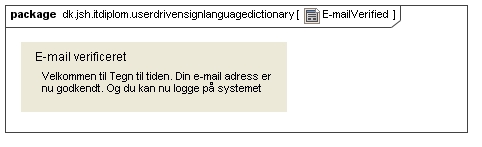


Figur – Opret ny burger side

Siden ”Ret brugeroplysninger” vil være som ovenstående side, bortset fra at ”Brugerkode” ikke kan rettes. Hvis der ikke er problemer på ovenstående side, dvs. alle felter er udfyldt og ”Brugerkode” er ikke brugt af en anden, ”Password” og ”Gentag password” er ens og opfylder password krav, så sendes der en mail til den indtastede e-mail adresse med følgende tekst:

|  |
| --- |
| Velkommen til Tegn til tiden.  Før du kan logge på systemet skal du trykke på følgende link, for at bekrærft din mail adresse.  [Bekræft e-mail](http://www.google.dk)  OBS! - Denne mail kan ikke besvares.  Eventuelle spørgsmål kan rettes til Jan Scrhøder Hansen på e-mail: jan.sch.hansen@gmail.com  Med venlig hilsen  Tegn til tiden |

Når brugeren trykker på linket i mailen, kommer denne til følgende side:



Figur 19 – E-mail verificeret side



## 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[[17]](#endnote-17). Jeg vil her komme ind på de punkter, jeg selv som udvikler lægger vægt på.

Som udvikler har jeg arbejdet med Struts[[18]](#endnote-18) 1.3, samt med ASP.NET[[19]](#endnote-19), 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[[20]](#endnote-20) tags, HTML[[21]](#endnote-21) tags, JavaScript[[22]](#endnote-22) i en og samme fil. Det samme gør sig også gældende i ASP.NET, hvor det så bare er C#[[23]](#endnote-23) 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[[24]](#endnote-24) 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[[25]](#endnote-25), som er det samme som SQL[[26]](#endnote-26) 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

Man kan ikke lave en offentlig webapplikation uden at komme ind på sikkerhed. Så her følger de tanker jeg har gjort mig om sikkerhed.

Wicket er ifølge wickets hjemmeside[[27]](#endnote-27) ”secure by default”. Og jeg har heller ikke kunne fremprovokere diverse ”Injection flaws”[[28]](#endnote-28) 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). Som er en Hibernate metode, som er med til at beskytte mod Injections.

Med hensyn til adgangskontrol vil jeg anbefale at benytte HTTPS/SSL[[29]](#endnote-29) under selve login, opret by bruger og ret brugeroplysninger 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.

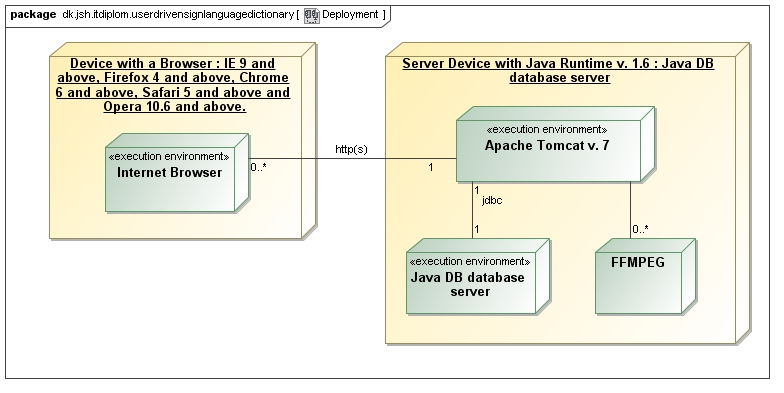
I databasen bør password ikke stå i klar tekst, men som en hashed[[30]](#endnote-30) værdi, som kun kan benyttes til at validere et password med.

OWASP siden (se [www.owasp.org](https://www.owasp.org/index.php/Main_Page)) er et rigtigt god side, om sikkerhedsproblematikker 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 20 - Deployment diagram

For at idriftsætte en webapplikation i en Tomcat server, bygges der en WAR[[31]](#endnote-31) 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

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 iterationer, som kunne benyttes til at udvikle ”Tegn til tiden”. Iterationer jeg har valgt bygger mest på de områder som hører sammen i applikationen, end på hvor lang tid de enkelte iterationer tager.

|  |  |
| --- | --- |
| **Iteration** | **Beskrivelse** |
| 1 | Indsamle krav, starte på use case beskrivelser, som skal bruges til login, brugeroprettelse/rettelse og e-mail verificering. |
| 2 | Starte på analyse og design for bruger vedligeholdelse. |
| 3 | Udvikle en web ramme med header, menu, login og brugeroprettelse forretningslogik og database adgang. |
| 4 | Use case, analyse, design og udvikling af ”Mine forespørgsler” siden samt ”Opret ny forespørgsel” og ”Ret forespørgsel”. Men ikke noget med grupper. I de følgende iterationer er alle faser med, dvs. use case, analyse, design, programmering og test. |
| 5 | Tilføje ”Søg” siden. |
| 6 | Tilføje ”Ord” siden, som udbygges når ”Upload” siden er færdig. |
| 7 | Tilføje ”Mine grupper”, ”Ret grupper” og ”Tilføj/fjern grupper” siderne. |
| 8 | Tilføje ”Upload”, ”Mine uploads” og ”Ret upload”. |
| 9 | Tilføje ”Alle forespørgsler” siden. |
| 10 | Tilføje ”Alle grupper” siden. |
| 11 | Tilføje E-mail notifikation. |
| 12 | Afsluttende test. |
| 13 | Idriftsættelse. |
| 14 | Vedligeholdelse. |

Jeg kan desværre ikke påstå at jeg har fulgt ovenstående. Men de iterative processer som UP, SCRUM[[32]](#endnote-32) og XP[[33]](#endnote-33). Ligger alle op til at man bliver klogere undervejs og må omprioriterer.

# Konklusion

Jeg er desværre ikke blevet helt færdig med selve programmeringen. For at illustrere hvor meget jeg har nået, har jeg under bilag lavet en brugervejledning, med skærmdumps. Men kun af det jeg har udviklet indtil nu. Håber jeg får lidt tid mellem aflevering af denne opgave og til eksamen, til at udvikle lidt mere. Se brugervejledningen under bilag afsnit 9.2. Ligeledes under bilag har jeg koden og lidt om det udviklingsmiljø jeg har brugt. Se bilag afsnit 9.3 for udviklingsmiljø og afsnit 9.4 for selve koden.

Jeg har været så heldig, at jeg kunne genbruge lidt fra mine tidligere eksamensopgaver på IT-Diplomuddannelsen. Bl.a. fra faget ”Objektorienterede metoder” hvor jeg har genbrugt selve rapportopbygningen. Derudover har jeg genbrugt lidt fra opgaven til faget ”Databasesystemer og Web”, hvor jeg har lånt lidt om Hibernate og optimistisk låsning. Og til slut har jeg lånt lidt fra opgaven til faget ”Web og serverprogrammering” hvor jeg har lånt lidt om Wicket og sikkerhed.

Det er altid god skik at komme med ”billige” alternative løsninger, og i dette tilfælde burde man undersøge diverse CMS[[34]](#endnote-34) løsninger. For at se om de ikke kunne bruges. Det er desværre ikke noget jeg har haft tid til. En anden, efter min mening oplagt mulighed er Wikipedia, eller en løsning som bygger på Wiki teknologi. Men mig bekendt understøtter disse ikke video. Men det ville være oplagt at på f.eks. den danske Wikipedia, kunne der være en lille videosekvens for de opslag/ord, som det giver mening at kunne på tegnsprog.

En oplagt vider udviklings mulighed for denne løsning, ville være at udvikle smartphone applikationer (apps). Dette er der mange grunde til. En er at døve har taget smartphonen til sig, specielt dem med et kamera, som vender mod brugeren selv. Da de på denne måde kan kommunikere på tegnsprog. Og selve telefonen kan jo også bruges til at optage forslag med.

Jeg stod over for at flere muligheder mht. hvordan jeg ville vise video i web applikationen. Mit valgt faldt på HTML 5’s video tag. Dette medførte dog nogle problemer, bl.a. at det kun er de nyeste browsere som understøtter dette HTML tag. Dertil skal siges at HTML 5 ikke er en færdig standard endnu. En af de ting som de forskellige browser leverandører ikke er blevet enige om endnu, er hvilke formater der skal benyttes. Ifølge W3School siden (se siden [www.w3schools.com/html5/html5\_video.asp](http://www.w3schools.com/html5/html5_video.asp)), kan man nøjes med OGG[[35]](#endnote-35) og MPEG-4[[36]](#endnote-36) formatet, så skulle de gængse browsere været dækket ind. HTML 5 video tag’et kan dog linke til flere filer, så browseren selv kan vælge hvad den foretrækker. Indtil videre understøtter jeg kun OGG formatet. Vis man ønsker at understøtte ældre browsere, så er man nød til at overveje Adobe Flash[[37]](#endnote-37). Det har jeg dog fravalgt, da meget tyder på at det er på vej ud. Bl.a. fordi Apple ikke understøtter det, i mange af deres produkter.

Så fik Steve Jobs det sidste ord igen.

# 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 brugervejledning til ”Tegn til tiden”. Dog har jeg kun medtaget det som jer har nået et udvikle.

Den første side brugerne af ”Tegn til tiden” møder er ”Søge siden”.

å 

Som det fremgår af siden, er man ikke logget på systemet endnu. Til venstre er der en menu med de muligheder, der er for ikke indloggede brugere. På selve siden er der mulighed for at søge efter ord eller efter ord i grupper. I søgefeltet kan der bruges følgende wildcards:

|  |  |
| --- | --- |
| **Wildcard** | **Beskrivelse** |
| \* | Erstatter 1 eller flere tegn. F. eks. hvis man ønsker alle ord som starter med ned ”Nord” så kan man skrive ”Nord\*” i feltet. |
| ? | Erstatter 1 og kun et tegn. |

Hvis der f.eks. søges efter grupper som starter med ”Nord” så kunne følgende side fremkomme.



Som det fremgår her er der to ord som indgår i en gruppe som hedder ”Nordafrika”.

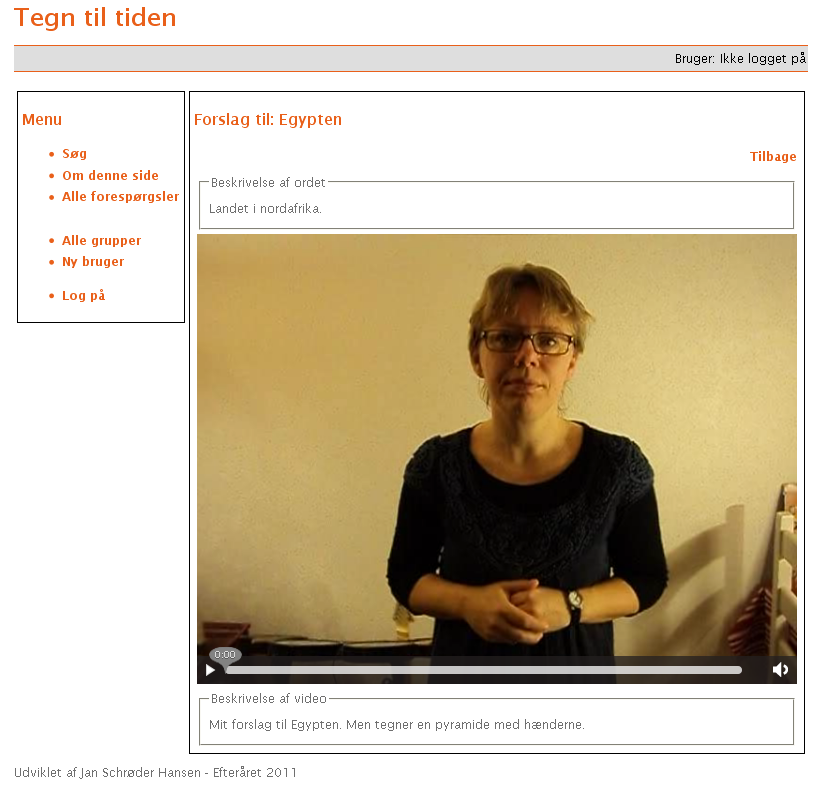
For alle lister i ”Tegn til tiden” er der max. 4 ”linier” per side. Og man kan navigere i siderne ved at trykke på symbolerne i højre hjørne af listerne.

|  |  |
| --- | --- |
| **Symbol** | **Beskrivelse** |
| << | Gå til første side. |
| < | Gå til forrige side. |
| 1 | Gå til den valgte side. Der kan f.eks. stå 1 2. Det vil side at der er to sider. |
| > | Gå til næste side. |
| >> | Gå til sidste side. |

Hvis man så trykker på ”Egypten” linket så kommer følgende side frem.



Her fremgår det at der kun er et forslag til ordet. Hvis man trykker på ”Vis” linket kommer følgende side.



Her kan forslaget til ordet så ses ved af afspille videoen. Derudover er der en beskrivelse af ordet og af videoen.

**Oprettelse af en ny bruger**

Ved at trykke på menulinket ”Opret ny bruger” kommer følgende side.



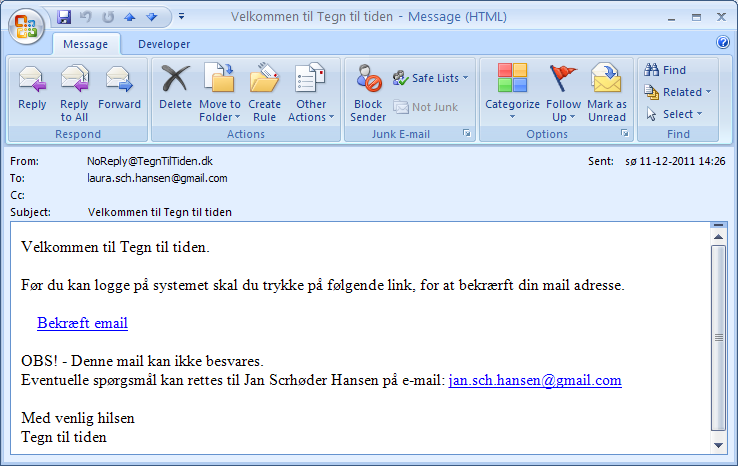
Denne side udfyldes.

****

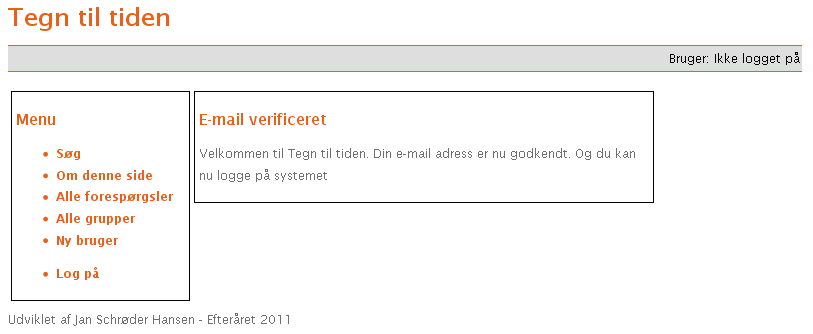
Ved tryk på ”Gem” kommer følgende side.



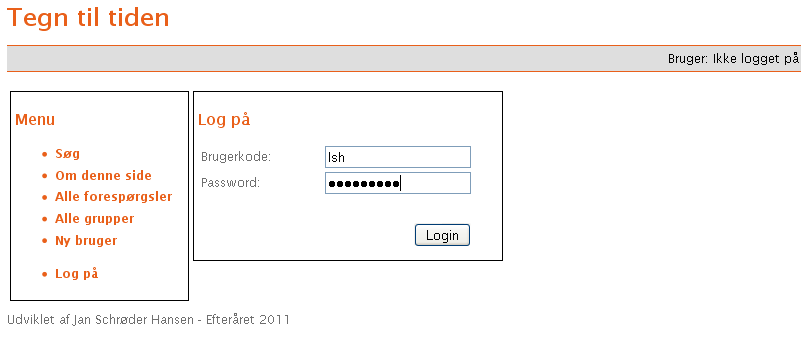
Som det fremgår af siden, så har systemet sendt en mail til brugen. Så e-mail adressen kan verificeres.



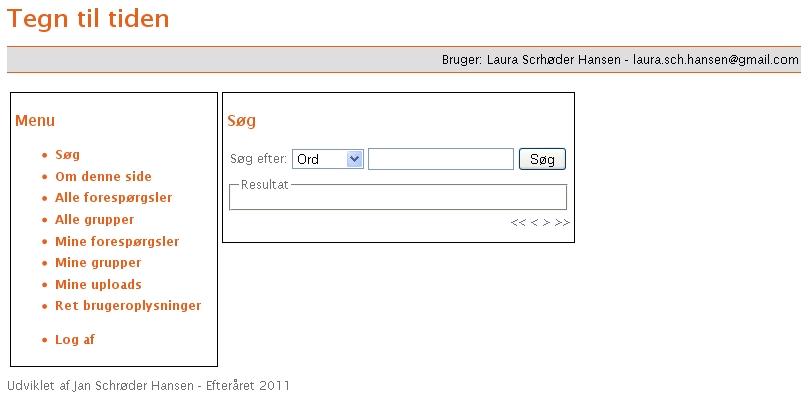
Her skal brugeren så trykke på ”Bekræft email” linket. som viser følgende side.



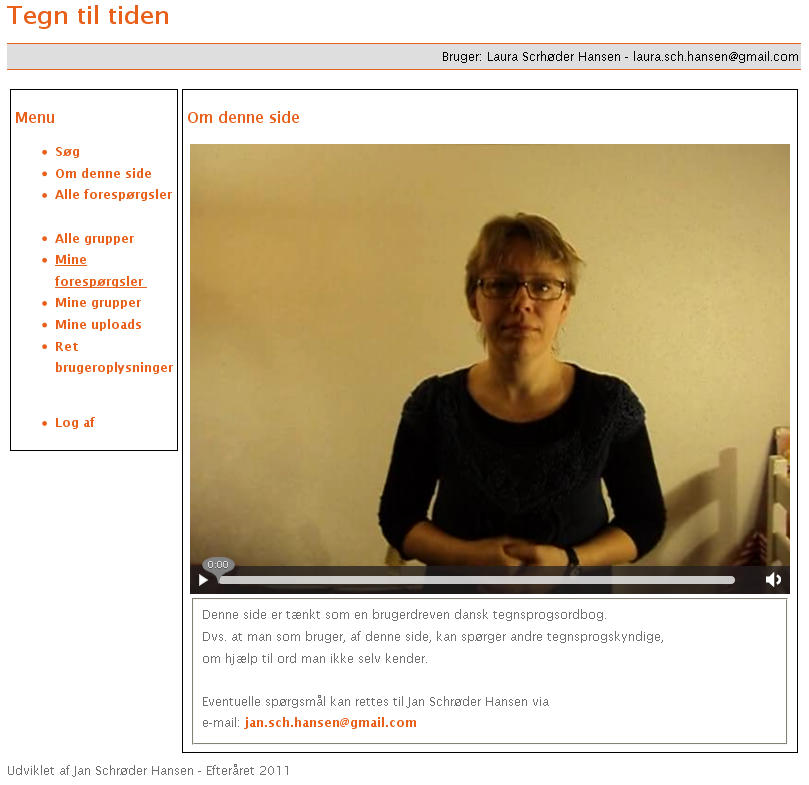
Så er brugeren klar til at logge sig på systemet. Ved at trykke på ”Log på” linket.



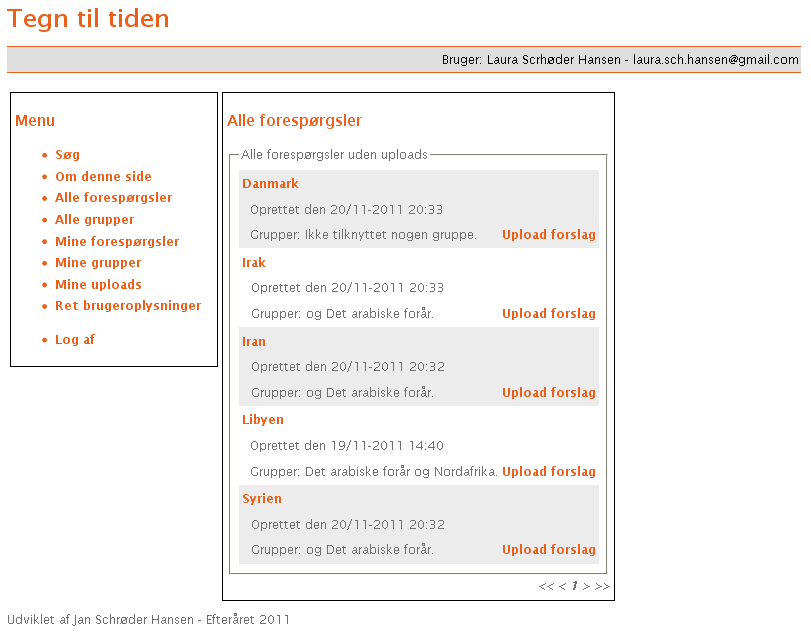
Brugeren trykker på ”Login” knappen og kommer til Søge siden, men nu som indlogget bruger med flere valgmuligher i Menu boksen.



”Om denne side” menulinket viser følgende side. Med en lille velkommen video, samt en kort beskrivelse af formålet med denne applikation.



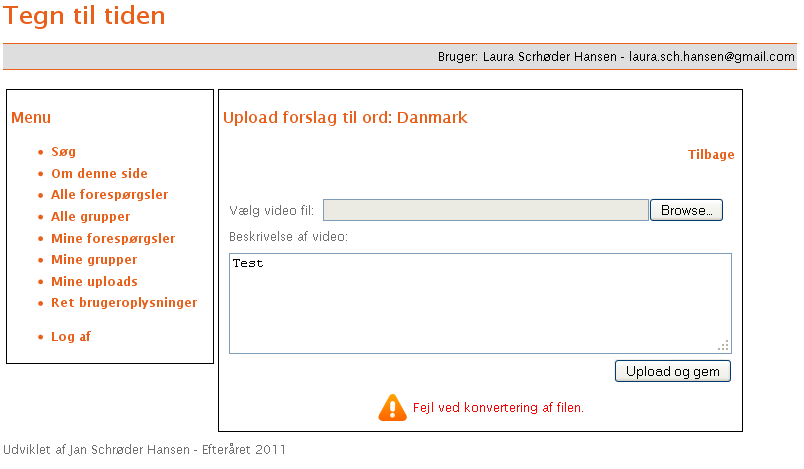
”Alle forespørgsler” menulinket viser følgende side.



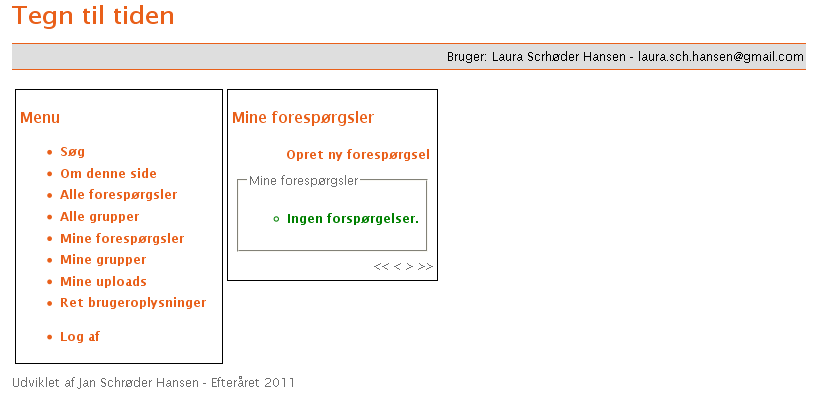
Denne liste indeholder alle de ord som mangler forslag. Hvis brugeren trykker på et af ”Upload forslag” linkene, så kommer følgende side.



Her vælger brugeren en videofil vha. af ”Browse…” knappen, som viser en alm. fildialog, hvor en fil kan vælges, brugeren udfylder beskrivelse feltet og trykker på ”Upload og gem” knappen. Systemet henter filen og konvertere denne til et HTML5 video format og gemmer den. Denne proces tager lidt tid. Hvis systemet ikke kan konverter filen kommer den en fejl besked.



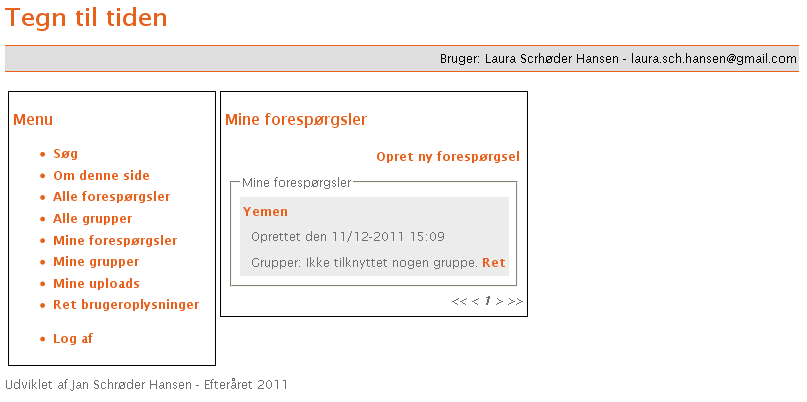
”Mine forespørgsler” menulinket viser følgende side.



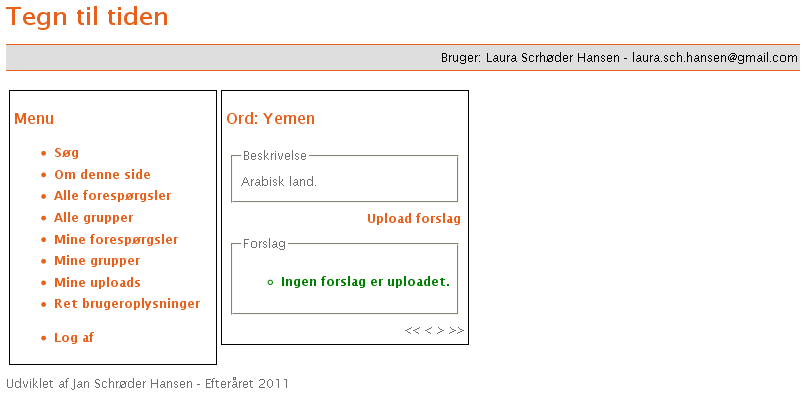
Her fremgår det at brugeren ikke har oprettet nogle forespørgsler endnu. Dette gøres ved at trykke på ”Opret ny forespørgsel” linket, som viser følgende side.



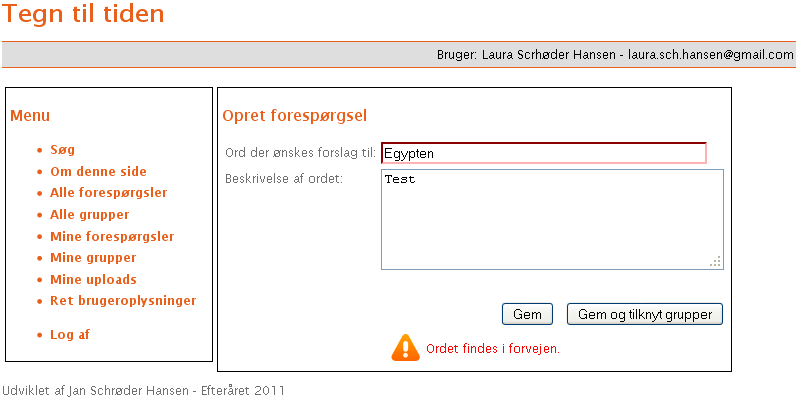
Her kan brugeren så ønske forslag til et ord, ved at udfylde de to felter og trykke på ”Gem” knappen. ”Gem og tilknyt grupper” er ikke implementeret endnu. Når der gemmes så vender systemet tilbage til ”Mine forespørgsler”



På listen er der et ”Ret” link som ikke er implementeret endnu. Hvis brugeren trykker på ord linket, så vises følgende side, nu med et …



Hvis man forsøger at oprette et ord som findes i forvejen, så kommer følgende fejl.



Til slut kan brugeren logge af ved at trykke på menulinket ”Log af”, og følgeden side vises, hvor det fremgår at brugeren ikke er logget på mere.



## Udviklingsmiljø

Under udviklingen har jeg brugt følgende værktøjet, programmer og services:

* NetBeans[[38]](#endnote-38) 7.0.1 – Java udviklingsmiljø, i dette miljø kan man starte følgende servere, som jeg har benyttet mig af:
  + JavaDB server.
  + Apache Tomcat 7 server.
* Apache James som er en mailserver. Denne har jeg sat op til, at sende alle mails til den samme modtager, som er min arbejdsmail.
* Microsoft Outlook, mail program, for at læse de mails, systemet genererer.
* Apache SubVersion[[39]](#endnote-39), kode versionskontrol system, som jeg bruger sammen med Google’s GoogleCode[[40]](#endnote-40) service. Så jeg har alle mine kodefiler, dokumenter m.m. på GoogleCode. Se [code.google.com/p/user-driven-sign-language-dictionary](http://code.google.com/p/user-driven-sign-language-dictionary/). Så alt hvad jeg har lavet ligger lokalt på en pc, og på denne GoggleCode side.



## Kode

Koden vises pakkevis i følgende rækkefølge:

* dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity
* dk.jsh.itdiplom.userdrivensignlanguagedictionary.business
* dk.jsh.itdiplom.userdrivensignlanguagedictionary.wicket
* dk.jsh.itdiplom.userdrivensignlanguagedictionary.util

Til sidst følger nogle enkelte Hibernate setup, SQL DDL og Ant scripts filer.

HTML findes i wicket pakket.

**Pakken: dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity**

Filen: ApplicationUser.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity;

2

3 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.Constants.UserRole;

4 import java.io.Serializable;

5 import java.util.Date;

6 import javax.persistence.\*;

7

8 /\*\*

9 \* Application user entity class.

10 \*

11 \* @author Jan S. Hansen

12 \*/

13 @Entity

14 public class ApplicationUser implements Serializable {

15 private static final long serialVersionUID = 1L;

16

17 @Id

18 @GeneratedValue(strategy = GenerationType.IDENTITY)

19 protected Long id;

20 @Version

21 @Column(nullable = false)

22 protected Integer version;

23 @Column(length=20, nullable = false, unique=true)

24 protected String login;

25 @Column(length=20, nullable = false)

26 protected String password;

27 @Column(length=50, nullable = false)

28 protected String fullname;

29 @Enumerated(EnumType.STRING)

30 @Column(length=50, nullable = false)

31 protected String email;

32 @Column(nullable = true)

33 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

34 protected Date emailVerificationSent;

35 @Column(nullable = true)

36 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

37 protected Date emailVerified;

38 @Enumerated(EnumType.STRING)

39 @Column(length=10, nullable = false)

40 protected UserRole userRole;

41

42 public ApplicationUser() {

43 }

44

45 public ApplicationUser(String login, String password, String fullname,

46 String email, Date emailVerificationSent,

47 Date emailVerified, UserRole userRole) {

48 this.login = login;

49 this.password = password;

50 this.fullname = fullname;

51 this.email = email;

52 this.emailVerificationSent = emailVerificationSent;

53 this.emailVerified = emailVerified;

54 this.userRole = userRole;

55 }

56

57 public Long getId() {

58 return id;

59 }

60

61 public void setId(Long id) {

62 this.id = id;

63 }

64

65 public Integer getVersion() {

66 return version;

67 }

68

69 public void setVersion(Integer version) {

70 this.version = version;

71 }

72

73 public String getFullname() {

74 return fullname;

75 }

76

77 public void setFullname(String fullname) {

78 this.fullname = fullname;

79 }

80

81 public String getLogin() {

82 return login;

83 }

84

85 public void setLogin(String login) {

86 this.login = login;

87 }

88

89 public String getEmail() {

90 return email;

91 }

92

93 public void setEmail(String email) {

94 this.email = email;

95 }

96

97 public String getPassword() {

98 return password;

99 }

100

101 public void setPassword(String password) {

102 this.password = password;

103 }

104

105 public Date getEmailVerificationSent() {

106 return emailVerificationSent;

107 }

108

109 public void setEmailVerificationSent(Date emailVerificationSent) {

110 this.emailVerificationSent = emailVerificationSent;

111 }

112

113 public Date getEmailVerified() {

114 return emailVerified;

115 }

116

117 public void setEmailVerified(Date emailVerified) {

118 this.emailVerified = emailVerified;

119 }

120

121 public UserRole getUserRole() {

122 return userRole;

123 }

124

125 public void setUserRole(UserRole userRole) {

126 this.userRole = userRole;

127 }

128 }

129

Filen: Constants.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity;

2 import java.util.ArrayList;

3 import java.util.List;

4

5 /\*\*

6 \* Constants and enums.

7 \*

8 \* @author Jan S. Hansen

9 \*/

10 public class Constants {

11 private Constants() {}

12

13 /\*\*

14 \* BaRI user role: ADMIN, DEVELOPER, NORMAL.

15 \*/

16 public enum UserRole {

17 ADMIN("Administrator"),

18 NORMAL("Alm. bruger");

19

20 private String name;

21

22 UserRole(String name) {

23 this.name = name;

24 }

25

26 public String getName() {

27 return name;

28 }

29

30 public static List<String> getNames() {

31 List<String> names = new ArrayList<String>();

32 names.add(ADMIN.name);

33 names.add(NORMAL.name);

34 return names;

35 }

36

37 public static UserRole getName(String name) {

38 if (ADMIN.name.equals(name)) {

39 return ADMIN;

40 }

41 return NORMAL;

42 }

43 }

44 }

45

Filen: Video.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity;

2

3 import java.io.Serializable;

4 import java.util.Date;

5 import javax.persistence.\*;

6

7 /\*\*

8 \* File entity class.

9 \*

10 \* @author Jan S. Hansen

11 \*/

12 @Entity

13 public class VideoFile implements Serializable {

14 private static final long serialVersionUID = 1L;

15

16 @Id

17 @GeneratedValue(strategy = GenerationType.IDENTITY)

18 protected Long id;

19 @Version

20 @Column(nullable = false)

21 protected Integer version;

22 @Column(length=100, nullable = false)

23 protected String fileName;

24 @Column(length=250, nullable = true)

25 protected String description;

26 @Column(length=50, nullable = false)

27 protected String resourceName;

28 @Column(nullable = false)

29 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

30 protected Date uploadedDateTime;

31 @ManyToOne(optional=false)

32 @org.hibernate.annotations.ForeignKey(name="fk\_file\_applicationuser")

33 protected ApplicationUser uploadedBy;

34 @ManyToOne(optional=false)

35 @org.hibernate.annotations.ForeignKey(name="fk\_file\_word")

36 protected Word toWord;

37

38 public VideoFile() {

39 };

40

41 public VideoFile(String fileName, String description, String resourceName,

42 Date uploadedDateTime,

43 ApplicationUser uploadedBy, Word toWord) {

44 this.fileName = fileName;

45 this.description = description;

46 this.resourceName = resourceName;

47 this.uploadedDateTime = uploadedDateTime;

48 this.uploadedBy = uploadedBy;

49 this.toWord = toWord;

50 }

51

52 public Long getId() {

53 return id;

54 }

55

56 public void setId(Long id) {

57 this.id = id;

58 }

59

60 public Integer getVersion() {

61 return version;

62 }

63

64 public void setVersion(Integer version) {

65 this.version = version;

66 }

67

68 public String getFileName() {

69 return fileName;

70 }

71

72 public void setFileName(String fileName) {

73 this.fileName = fileName;

74 }

75

76 public String getDescription() {

77 return description;

78 }

79

80 public void setDescription(String description) {

81 this.description = description;

82 }

83

84 public String getResourceName() {

85 return resourceName;

86 }

87

88 public void setResourceName(String resourceName) {

89 this.resourceName = resourceName;

90 }

91

92 public Word getToWord() {

93 return toWord;

94 }

95

96 public void setToWord(Word toWord) {

97 this.toWord = toWord;

98 }

99

100 public ApplicationUser getUploadedBy() {

101 return uploadedBy;

102 }

103

104 public void setUploadedBy(ApplicationUser uploadedBy) {

105 this.uploadedBy = uploadedBy;

106 }

107

108 public Date getUploadedDateTime() {

109 return uploadedDateTime;

110 }

111

112 public void setUploadedDateTime(Date uploadedDateTime) {

113 this.uploadedDateTime = uploadedDateTime;

114 }

115 }

116

Filen: Word.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity;

2

3 import java.io.Serializable;

4 import java.util.ArrayList;

5 import java.util.Collections;

6 import java.util.Date;

7 import java.util.List;

8 import javax.persistence.\*;

9

10 /\*\*

11 \* Word entity class.

12 \*

13 \* @author Jan S. Hansen

14 \*/

15 @Entity

16 public class Word implements Serializable {

17 private static final long serialVersionUID = 1L;

18

19 @Id

20 @GeneratedValue(strategy = GenerationType.IDENTITY)

21 protected Long id;

22 @Version

23 @Column(nullable = false)

24 protected Integer version;

25 @Column(length=50, nullable = false, unique=true)

26 protected String word;

27 @Column(length=250, nullable = true)

28 protected String description;

29 @Column(nullable = false)

30 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

31 protected Date createdDateTime;

32 @ManyToOne(optional=false)

33 @org.hibernate.annotations.ForeignKey(name="fk\_word\_applicationuser")

34 protected ApplicationUser requestCreatedBy;

35 @ManyToMany(fetch = FetchType.EAGER)

36 @OrderBy("name")

37 @JoinTable(

38 name="WordGroupWordRelation",

39 joinColumns={@JoinColumn(name="WORD\_ID")},

40 inverseJoinColumns={@JoinColumn(name="WORDGROUP\_ID")}

41 )

42 protected List<WordGroup> wordGroups;

43

44 public Word() {

45 }

46

47 public Word(String word, String description, Date createdDateTime,

48 ApplicationUser requestCreatedBy) {

49 this.word = word;

50 this.description = description;

51 this.createdDateTime = createdDateTime;

52 this.requestCreatedBy = requestCreatedBy;

53 }

54

55 public Long getId() {

56 return id;

57 }

58

59 public void setId(Long id) {

60 this.id = id;

61 }

62

63 public Integer getVersion() {

64 return version;

65 }

66

67 public void setVersion(Integer version) {

68 this.version = version;

69 }

70

71 public String getWord() {

72 return word;

73 }

74

75 public void setWord(String word) {

76 this.word = word;

77 }

78

79 public String getDescription() {

80 return description;

81 }

82

83 public void setDescription(String description) {

84 this.description = description;

85 }

86

87 public ApplicationUser getRequestCreatedBy() {

88 return requestCreatedBy;

89 }

90

91 public void setRequestCreatedBy(ApplicationUser requestCreatedBy) {

92 this.requestCreatedBy = requestCreatedBy;

93 }

94

95 public Date getCreatedDateTime() {

96 return createdDateTime;

97 }

98

99 public void setCreatedDateTime(Date createdDateTime) {

100 this.createdDateTime = createdDateTime;

101 }

102

103 public List<WordGroup> getWordGroups() {

104 return wordGroups;

105 }

106

107 public List<String> getSortedWordGroups() {

108 List<String> list = new ArrayList<String>();

109 for (WordGroup wordGroup: wordGroups) {

110 list.add(wordGroup.name);

111 }

112 Collections.sort(list);

113 return list;

114 }

115 }

116

Filen: WordGroup.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity;

2

3 import java.io.Serializable;

4 import java.util.Date;

5 import javax.persistence.\*;

6

7 /\*\*

8 \* Word group entity class.

9 \*

10 \* @author Jan S. Hansen

11 \*/

12 @Entity

13 public class WordGroup implements Serializable{

14 private static final long serialVersionUID = 1L;

15

16 @Id

17 @GeneratedValue(strategy = GenerationType.IDENTITY)

18 protected Long id;

19 @Version

20 @Column(nullable = false)

21 protected Integer version;

22 @Column(length=30, nullable = false, unique=true)

23 protected String name;

24 @Column(length=250, nullable = true)

25 protected String description;

26 @Column(nullable = false)

27 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

28 protected Date createdDateTime;

29 @ManyToOne(optional=false)

30 @org.hibernate.annotations.ForeignKey(name="fk\_wordgroup\_applicationuser")

31 protected ApplicationUser createdBy;

32

33 public WordGroup() {

34 }

35

36 public WordGroup(String name, String description, Date createdDateTime,

37 ApplicationUser createdBy) {

38 this.name = name;

39 this.description = description;

40 this.createdDateTime = createdDateTime;

41 this.createdBy = createdBy;

42 }

43

44 public Long getId() {

45 return id;

46 }

47

48 public void setId(Long id) {

49 this.id = id;

50 }

51

52 public Integer getVersion() {

53 return version;

54 }

55

56 public void setVersion(Integer version) {

57 this.version = version;

58 }

59

60 public String getName() {

61 return name;

62 }

63

64 public void setName(String name) {

65 this.name = name;

66 }

67

68 public String getDescription() {

69 return description;

70 }

71

72 public void setDescription(String description) {

73 this.description = description;

74 }

75

76 public ApplicationUser getCreatedBy() {

77 return createdBy;

78 }

79

80 public void setCreatedBy(ApplicationUser createdBy) {

81 this.createdBy = createdBy;

82 }

83

84 public Date getCreatedDateTime() {

85 return createdDateTime;

86 }

87

88 public void setCreatedDateTime(Date createdDateTime) {

89 this.createdDateTime = createdDateTime;

90 }

91 }

92

Filen: WordGroupWordRelation.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity;

2

3 import java.io.Serializable;

4 import javax.persistence.\*;

5

6 /\*\*

7 \* Word group/Word many to many relation.

8 \*

9 \* @author Jan S. Hansen

10 \*/

11 @Entity

12 @Table(name="WordGroupWordRelation",

13 uniqueConstraints = {

14 @UniqueConstraint(columnNames={"wordGroup\_id", "word\_id"})

15 })

16 public class WordGroupWordRelation implements Serializable {

17 private static final long serialVersionUID = 1L;

18

19 @Id

20 @GeneratedValue(strategy = GenerationType.IDENTITY)

21 protected Long id;

22 @Version

23 @Column(nullable = false)

24 protected Integer version;

25 @ManyToOne(optional=false)

26 @org.hibernate.annotations.ForeignKey(name="fk\_wordgroupwordrelation\_wordgroup")

27 protected WordGroup wordGroup;

28 @ManyToOne(optional=false)

29 @org.hibernate.annotations.ForeignKey(name="fk\_wordgroupwordrelation\_word")

30 protected Word word;

31

32 public WordGroupWordRelation() {

33 }

34

35 public WordGroupWordRelation(WordGroup wordGroup, Word word) {

36 this.wordGroup = wordGroup;

37 this.word = word;

38 }

39

40 public Long getId() {

41 return id;

42 }

43

44 public void setId(Long id) {

45 this.id = id;

46 }

47

48 public Integer getVersion() {

49 return version;

50 }

51

52 public void setVersion(Integer version) {

53 this.version = version;

54 }

55

56 public Word getWord() {

57 return word;

58 }

59

60 public void setWord(Word word) {

61 this.word = word;

62 }

63

64 public WordGroup getWordGroup() {

65 return wordGroup;

66 }

67

68 public void setWordGroup(WordGroup wordGroup) {

69 this.wordGroup = wordGroup;

70 }

71 }

72

**Pakken: dk.jsh.itdiplom.userdrivensignlanguagedictionary.business**

Filen: ApplicationUserBusiness.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.business;

2

3 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.ApplicationUser;

4 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.util.HibernateUtil;

5 import java.util.Date;

6 import java.util.List;

7 import org.hibernate.Query;

8 import org.hibernate.Session;

9 import org.hibernate.Transaction;

10

11 /\*\*

12 \* Business metods for ApplicationUser.

13 \*

14 \* @author Jan S. Hansen

15 \*/

16 public class ApplicationUserBusiness {

17

18 private ApplicationUserBusiness(){};

19

20 /\*\*

21 \* Gets a applicationUser from login and password.

22 \*

23 \* @param login user login

24 \* @param password password

25 \* @return a ApplicationUser or null if login or password is wrong.

26 \*/

27 public static ApplicationUser isValidUser(String login, String password) {

28 ApplicationUser appUser = null;

29 Session session = HibernateUtil.getSessionFactory().openSession();

30 String hql = "select appUser from "

31 + "dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity."

32 + "ApplicationUser appUser "

33 + "where appUser.login = :login "

34 + "and appUser.password = :password "

35 + "and appUser.emailVerified is not null";

36 Query query = session.createQuery(hql);

37 query.setString("login", login);

38 query.setString("password", login);

39 List<ApplicationUser> appUsers = query.list();

40 if (appUsers.size() == 1) {

41 appUser = appUsers.get(0);

42 }

43 else if (appUsers.size() > 1) {

44 throw new RuntimeException("More then one user with login " +

45 login);

46 }

47 session.close();

48 return appUser;

49 }

50

51 /\*\*

52 \* Persist a new Application user.

53 \*

54 \* @param newUser a new ApplicationUser

55 \*/

56 public static void saveNew(ApplicationUser newUser) {

57 Session session = HibernateUtil.getSessionFactory().openSession();

58 Transaction tx = session.beginTransaction();

59 session.save(newUser);

60 tx.commit();

61 session.close();

62 }

63

64 /\*\*

65 \* Test if a user login is in use.

66 \*

67 \* @param login user login to test

68 \* @return true is user login is in use.

69 \*/

70 public static boolean isUserLoginInUse(String login) {

71 boolean inUse = false;

72

73 Session session = HibernateUtil.getSessionFactory().openSession();

74 String hql = "select appUser "

75 + "from dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity."

76 + "ApplicationUser appUser "

77 + "where appUser.login = :login";

78 Query query = session.createQuery(hql);

79 query.setString("login", login);

80 if (query.list().isEmpty()) {

81 inUse = false;

82 }

83 else {

84 inUse = true;

85 };

86 session.close();

87 return inUse;

88 }

89

90 /\*\*

91 \* Set e-mail verified to true.

92 \*

93 \* @param login user login

94 \*/

95 public static void setEmailVerified(String login) {

96 Session session = HibernateUtil.getSessionFactory().openSession();

97 String hql = "select appUser "

98 + "from dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity."

99 + "ApplicationUser appUser "

100 + "where appUser.login = :login";

101 Query query = session.createQuery(hql);

102 query.setString("login", login);

103 ApplicationUser user = (ApplicationUser)query.list().get(0);

104 user.setEmailVerified(new Date());

105 Transaction tx = session.beginTransaction();

106 session.save(user);

107 tx.commit();

108 session.close();

109 }

110 }

111

Filen: VideoFileBusiness.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.business;

2

3 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.VideoFile;

4 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.Word;

5 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.util.HibernateUtil;

6 import java.util.ArrayList;

7 import java.util.List;

8 import org.hibernate.Query;

9 import org.hibernate.Session;

10 import org.hibernate.Transaction;

11

12 /\*\*

13 \* Business metods for VideoFile.

14 \*

15 \* @author Jan S. Hansen

16 \*/

17 public class VideoFileBusiness {

18 private VideoFileBusiness(){};

19

20 /\*\*

21 \* Persist a new video file

22 \*

23 \* @param newWord a new video file

24 \*/

25 public static void saveNew(VideoFile newFile) {

26 Session session = HibernateUtil.getSessionFactory().openSession();

27 Transaction tx = session.beginTransaction();

28 session.save(newFile);

29 tx.commit();

30 session.close();

31 }

32

33 /\*\*

34 \* Get all video files for a word.

35 \*

36 \* @return A list of vidoe files

37 \*/

38 public static List<VideoFile> getAllVideoFilesForAWord(Word word) {

39 List<VideoFile> videoFileList = new ArrayList<VideoFile>();

40 Session session = HibernateUtil.getSessionFactory().openSession();

41 String hql =

42 "select videofile from "

43 + "dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity."

44 + "VideoFile videofile "

45 + "where videofile.toWord.id = :wordid "

46 + "order by videofile.uploadedDateTime desc";

47 Query query = session.createQuery(hql);

48 query.setLong("wordid", word.getId());

49 videoFileList = query.list();

50 session.close();

51 return videoFileList;

52 }

53 }

54

Filen: WordBusiness.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.business;

2

3 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.ApplicationUser;

4 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.Word;

5 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.WordGroup;

6 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.util.HibernateUtil;

7 import java.util.ArrayList;

8 import java.util.List;

9 import org.hibernate.Query;

10 import org.hibernate.Session;

11 import org.hibernate.Transaction;

12

13 /\*\*

14 \* Business metods for Word.

15 \*

16 \* @author Jan S. Hansen

17 \*/

18 public class WordBusiness {

19 private WordBusiness(){};

20

21 /\*\*

22 \* Persist a new Word.

23 \*

24 \* @param newWord a new Word

25 \*/

26 public static void saveNew(Word newWord) {

27 Session session = HibernateUtil.getSessionFactory().openSession();

28 Transaction tx = session.beginTransaction();

29 session.save(newWord);

30 tx.commit();

31 session.close();

32 }

33

34 /\*\*

35 \* Test if a word exists.

36 \*

37 \* @param word word to test

38 \* @return true is the word exists

39 \*/

40 public static boolean isWordInUse(String word) {

41 boolean inUse = false;

42

43 Session session = HibernateUtil.getSessionFactory().openSession();

44 String hql = "select word "

45 + "from dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity."

46 + "Word word "

47 + "where word.word = :word";

48 Query query = session.createQuery(hql);

49 query.setString("word", word);

50 if (query.list().isEmpty()) {

51 inUse = false;

52 }

53 else {

54 inUse = true;

55 };

56 session.close();

57 return inUse;

58 }

59

60 /\*\*

61 \* Get all words created by a specific user.

62 \*

63 \* @return A list of Word.

64 \*/

65 public static List<Word> getAllWordsCreatedByUser(ApplicationUser user) {

66 List<Word> wordList = new ArrayList<Word>();

67 Session session = HibernateUtil.getSessionFactory().openSession();

68 String hql =

69 "select word from "

70 + "dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity."

71 + "Word word "

72 + "where word.requestCreatedBy.id = :userid "

73 + "order by word.word";

74 Query query = session.createQuery(hql);

75 query.setLong("userid", user.getId());

76 wordList = query.list();

77 session.close();

78 return wordList;

79 }

80

81 /\*\*

82 \* Search for words.

83 \*

84 \* @param search search string

85 \* @return a list of words that match the search criteria

86 \*/

87 public static List<Word> search(String search) {

88 List<Word> wordList = new ArrayList<Word>();

89 search = search.toLowerCase();

90 search = search.replace("\*", "%");

91 search = search.replace("?", "\_");

92 boolean useLike = false;

93 if (search.indexOf("%") != -1 || search.indexOf("\_") != -1) {

94 useLike = true;

95 }

96 Session session = HibernateUtil.getSessionFactory().openSession();

97 StringBuilder hql = new StringBuilder();

98 hql.append("select word from ");

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

100 hql.append("Word word ");

101 if (useLike) {

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

103 }

104 else {

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

106 }

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

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

109 query.setString("search", search);

110 wordList = query.list();

111 session.close();

112 return wordList;

113 }

114

115 /\*\*

116 \* Search for words with specific groups.

117 \*

118 \* @param search search string

119 \* @return a list of words that match the search criteria

120 \*/

121 public static List<Word> search(List<WordGroup> wordGroups) {

122 List<Word> wordList = new ArrayList<Word>();

123 Session session = HibernateUtil.getSessionFactory().openSession();

124 StringBuilder hql = new StringBuilder();

125 hql.append("select word from ");

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

127 hql.append("Word word ");

128 hql.append("join word.wordGroups wordGroup ");

129 hql.append("where wordGroup.id in (");

130 boolean first = true;

131 for (WordGroup wordGroup : wordGroups) {

132 if (first) {

133 first = false;

134 }

135 else {

136 hql.append(", ");

137 }

138 hql.append(wordGroup.getId());

139 }

140 hql.append(") ");

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

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

143 wordList = query.list();

144 session.close();

145 return wordList;

146 }

147

148 /\*\*

149 \* Get all words without uploads

150 \*

151 \* @return a list of all words without uploads

152 \*/

153 public static List<Word> getAllWordsWithoutUploads() {

154 List<Word> wordList = new ArrayList<Word>();

155 Session session = HibernateUtil.getSessionFactory().openSession();

156 StringBuilder hql = new StringBuilder();

157 hql.append("select word from ");

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

159 hql.append("Word word ");

160 hql.append("where word.id not in (");

161 hql.append("select distinct (videoFile.toWord.id) from ");

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

163 hql.append("VideoFile videoFile ");

164 hql.append(") ");

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

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

167 wordList = query.list();

168 session.close();

169 return wordList;

170 }

171 }

172

Filen: WordGroupBusiness.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.business;

2

3 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.WordGroup;

4 import dk.jsh.itdiplom.userdrivensignlanguagedictionary.util.HibernateUtil;

5 import java.util.ArrayList;

6 import java.util.List;

7 import org.hibernate.Query;

8 import org.hibernate.Session;

9

10 /\*\*

11 \* Business metods for word groups.

12 \*

13 \* @author Jan S. Hansen

14 \*/

15 public class WordGroupBusiness {

16 private WordGroupBusiness() {}

17

18 /\*\*

19 \* Get all word groups.

20 \*

21 \* @return A list of WordGroup.

22 \*/

23 public static List<WordGroup> getAllWordGroups() {

24 List<WordGroup> wordGroupList = new ArrayList<WordGroup>();

25 Session session = HibernateUtil.getSessionFactory().openSession();

26 String hql =

27 "select wordGroup from "

28 + "dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity."

29 + "WordGroup wordGroup "

30 + "order by wordGroup.name";

31 Query query = session.createQuery(hql);

32 wordGroupList = query.list();

33 session.close();

34 return wordGroupList;

35 }

36

37 /\*\*

38 \* Search for groups.

39 \*

40 \* @param search search string

41 \* @return a list of word groups that match the search criteria

42 \*/

43 public static List<WordGroup> search(String search) {

44 List<WordGroup> wordGroupList = new ArrayList<WordGroup>();

45 search = search.toLowerCase();

46 search = search.replace("\*", "%");

47 search = search.replace("?", "\_");

48 boolean useLike = false;

49 if (search.indexOf("%") != -1 || search.indexOf("\_") != -1) {

50 useLike = true;

51 }

52 Session session = HibernateUtil.getSessionFactory().openSession();

53 StringBuilder hql = new StringBuilder();

54 hql.append("select wordGroup from ");

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

56 hql.append("WordGroup wordGroup ");

57 if (useLike) {

58 hql.append("where lower(wordGroup.name) like :search ");

59 }

60 else {

61 hql.append("where lower(wordGroup.name) = :search ");

62 }

63 hql.append("order by wordGroup.name");

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

65 query.setString("search", search);

66 wordGroupList = query.list();

67 session.close();

68 return wordGroupList;

69 }

70 }

71

**Pakken: dk.jsh.itdiplom.userdrivensignlanguagedictionary.wicket**

Filen: todo.java

Filen: todo.java

**Pakken: dk.jsh.itdiplom.userdrivensignlanguagedictionary.util**

Filen: ConvertVideo.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.util;

2

3 import java.io.BufferedReader;

4 import java.io.IOException;

5 import java.io.InputStream;

6 import java.io.InputStreamReader;

7 import java.util.Date;

8 import java.util.logging.Level;

9 import java.util.logging.Logger;

10

11 /\*\*

12 \* Convert video files to the OGG video format.

13 \*

14 \* @author Jan S. Hansen

15 \*/

16 public class ConvertVideo {

17 static final Logger logger = Logger.getLogger(ConvertVideo.class.getName());

18

19 /\*\*

20 \* Convert video file til ogg file format.

21 \*

22 \* @param source Source file name

23 \* @param dest Destination file name

24 \*/

25 public boolean convert(String source, String dest) {

26 Runtime runtime = Runtime.getRuntime();

27 //TODO: location of ffmpeg2theora-0.28.exe in a properties file

28 StringBuilder cmdLine = new StringBuilder();

29 cmdLine.append("\\GoogleCode\\user-driven-sign-language-dictionary");

30 cmdLine.append("\\ffmpeg2theora-0.28.exe ");

31 cmdLine.append(" -o ");

32 cmdLine.append(dest);

33 cmdLine.append(" ");

34 cmdLine.append(source);

35 try {

36 logger.info("Convert video: " + cmdLine.toString());

37 Process process = runtime.exec(cmdLine.toString());

38

39 //Start thread to read standard error

40 ProcessOutput err = new ProcessOutput(process.getErrorStream(),

41 "ERR");

42 Thread errThread = new Thread(err);

43 errThread.start();

44

45 //Start thread to read standard output

46 ProcessOutput std = new ProcessOutput(process.getInputStream(),

47 "STD");

48 Thread stdThread = new Thread(std);

49 stdThread.start();

50

51 process.waitFor();

52 } catch (Exception exception) {

53 logger.log(Level.SEVERE, "Error converting video", exception);

54 return false;

55 }

56 return true;

57 }

58

59 /\*\*

60 \* Create an OGV wicket resource name. Format UserId\_xx\_wordId\_yy\_mm.ogv.

61 \* xx = user Id (Database PK), yy = word Id and mm = milliseconds since

62 \* 1/1-1970.

63 \*

64 \* @param userId User id

65 \* @param wordId word id

66 \* @return a OGV wicket resource name.

67 \*/

68 public String createOgvResourceName(String userId, String wordId) {

69 StringBuilder sb = new StringBuilder("UserId\_");

70 sb.append(userId);

71 sb.append("\_wordId\_");

72 sb.append(wordId);

73 sb.append("\_");

74 Date now = new Date();

75 sb.append(now.getTime());

76 sb.append(".ogv");

77 return sb.toString();

78 }

79

80 /\*\*

81 \* Creates an OGV filename with full path.

82 \*

83 \* @param resourceName resource name

84 \* @return filename

85 \*/

86 public String createOgvFilename(String resourceName) {

87 //TODO: Get full path, should not be hard coded.

88 StringBuilder sb = new StringBuilder("C:\\GoogleCode\\"

89 + "user-driven-sign-language-dictionary\\Code\\"

90 + "UserDrivenSignLanguageDictionary\\build\\web\\"

91 + "WEB-INF\\classes\\dk\\jsh\\itdiplom\\"

92 + "userdrivensignlanguagedictionary\\wicket\\uploadedvideo\\");

93 sb.append(resourceName);

94 return sb.toString();

95 }

96

97 private class ProcessOutput implements Runnable {

98 private InputStream inputStream;

99 private String type;

100

101 public ProcessOutput(InputStream inputStream, String type) {

102 this.inputStream = inputStream;

103 this.type = type;

104 }

105

106 @Override

107 public void run() {

108 BufferedReader procesOutput = new BufferedReader(new

109 InputStreamReader(inputStream));

110 String line = null;

111 try {

112 while ((line = procesOutput.readLine()) != null) {

113 logger.info(type + ": " + line);

114 }

115 } catch (IOException ex) {

116 logger.log(Level.SEVERE, "Error reading output of type " + type,

117 ex);

118 }

119 }

120 }

121 }

122

Filen: EMailSender.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.util;

2

3 import java.util.Properties;

4 import javax.mail.Message;

5 import javax.mail.MessagingException;

6 import javax.mail.Session;

7 import javax.mail.Transport;

8 import javax.mail.internet.InternetAddress;

9 import javax.mail.internet.MimeMessage;

10

11 /\*\*

12 \* E-Mail sender. Singleton pattern.

13 \*

14 \* @author Jan S. Hansen

15 \*/

16 public class EMailSender {

17 private static Properties emailProperties = new Properties();

18 private static EMailSender singletonInstance;

19

20 /\*\*

21 \* Private constructor to prevent use of new keyword outside this class.

22 \*/

23 private EMailSender(){};

24

25 public static EMailSender getInstance() {

26 if (singletonInstance == null) {

27 //TODO: Get host from a property file.

28 emailProperties.put("mail.smtp.host", "localhost");

29 singletonInstance = new EMailSender();

30 }

31 return singletonInstance;

32 }

33

34 /\*\*

35 \* Send an e-mail.

36 \*

37 \* @param fromEmailAddr from e-mail address

38 \* @param toEmailAddr to e-mail adress

39 \* @param subject e-mail subject

40 \* @param body e-mail body text

41 \* @return true if no errors occurs.

42 \*/

43 public boolean sendEmail(String fromEmailAddr, String toEmailAddr,

44 String subject, String body) {

45 Session session = Session.getDefaultInstance(emailProperties, null);

46 MimeMessage message = new MimeMessage(session);

47 try {

48 message.setFrom(new InternetAddress(fromEmailAddr));

49 message.addRecipient(Message.RecipientType.TO, new InternetAddress(

50 toEmailAddr));

51

52 message.setSubject(subject);

53 message.setContent(body, "text/html");

54 Transport.send(message);

55 return true;

56 } catch (MessagingException ex) {

57 System.err.println("Cannot send email. " + ex);

58 return false;

59 }

60 }

61

62 /\*\*

63 \* Send a no-reply e-mail.

64 \*

65 \* @param toEmailAddr to e-mail adress

66 \* @param subject e-mail subject

67 \* @param body e-mail body text

68 \* @return true if no errors occurs.

69 \*/

70 public boolean sendNoReplyEmail(String toEmailAddr, String subject,

71 String body) {

72 return sendEmail("NoReply@TegnTilTiden.dk", toEmailAddr, subject, body);

73 }

74 }

75

Filen: HibernateUtil.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.util;

2

3 import org.hibernate.\*;

4 import org.hibernate.cfg.\*;

5

6 /\*\*

7 \* Hibernate session factory.

8 \*

9 \* @author Jan S. Hansen

10 \*/

11 public class HibernateUtil {

12 private static SessionFactory sessionFactory;

13

14 static {

15 try {

16 sessionFactory =

17 new AnnotationConfiguration().configure().buildSessionFactory();

18 }

19 catch (Throwable ex) {

20 throw new ExceptionInInitializerError(ex);

21 }

22 }

23

24 /\*\*

25 \* Get a Hibernate session factory.

26 \*

27 \* @return a SessionFactory

28 \*/

29 public static SessionFactory getSessionFactory() {

30 return sessionFactory;

31 }

32

33 /\*\*

34 \* Close SessionFactory.

35 \*/

36 public static void shutdown() {

37 getSessionFactory().close();

38 }

39 }

40

Filen: Text.java

1 package dk.jsh.itdiplom.userdrivensignlanguagedictionary.util;

2

3 import java.util.List;

4

5 /\*\*

6 \* Text utilities.

7 \*

8 \* @author Jan S. Hansen

9 \*/

10 public class Text {

11

12 /\*\*

13 \* Make at list of word groups.

14 \*

15 \* @param wordGroupList a Word group list

16 \* @return a string of wordGroups

17 \*/

18 public static String makeWordGroupString(List<String> wordGroupList) {

19 StringBuilder groups = new StringBuilder();

20 int noOfGroups = wordGroupList.size();

21 if (noOfGroups > 0) {

22 for (int i = 0; i < noOfGroups; i++) {

23 String wordGroup = wordGroupList.get(i);

24 if (i > 0 && i < noOfGroups - 1) {

25 groups.append(", ");

26 }

27 else if (i == noOfGroups -1) {

28 groups.append(" og ");

29 }

30 groups.append(wordGroup);

31 }

32 }

33 else {

34 groups.append("Ikke tilknyttet nogen gruppe");

35 }

36 groups.append(".");

37 return groups.toString();

38 }

39 }

40

**Diverse scripts og setup filer.**

Filen: hibernate.cft.xml

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

3 <hibernate-configuration>

4 <session-factory>

5 <property name="hibernate.dialect">org.hibernate.dialect.DerbyDialect</property>

6 <property name="hibernate.connection.driver\_class">org.apache.derby.jdbc.ClientDriver</property>

7 <property name="hibernate.connection.url">jdbc:derby://localhost:1527/sample</property>

8 <property name="hibernate.connection.username">app</property>

9 <property name="hibernate.connection.password">app</property>

10

11

12 <property name="hibernate.dialect">org.hibernate.dialect.DerbyDialect</property>

13 <property name="hibernate.connection.driver\_class">org.apache.derby.jdbc.ClientDriver</property>

14 <property name="hibernate.connection.url">jdbc:derby://localhost:1527/UDSLD</property>

15 <property name="hibernate.connection.username">app</property>

16 <property name="hibernate.connection.password">app</property>

17

18 <!-- Show and print nice SQL on stdout -->

19 <property name="hibernate.show\_sql">true</property>

20 <property name="hibernate.format\_sql">true</property>

21

22 <!-- List of annotated classes -->

23 <mapping class="dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.ApplicationUser" />

24 <mapping class="dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.WordGroup" />

25 <mapping class="dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.Word" />

26 <mapping class="dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.WordGroupWordRelation" />

27 <mapping class="dk.jsh.itdiplom.userdrivensignlanguagedictionary.entity.VideoFile" />

28

29 </session-factory>

30 </hibernate-configuration>

31

32

Filen: build.xml

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!-- You may freely edit this file. See commented blocks below for -->

3 <!-- some examples of how to customize the build. -->

4 <!-- (If you delete it and reopen the project it will be recreated.) -->

5 <!-- By default, only the Clean and Build commands use this build script. -->

6 <!-- Commands such as Run, Debug, and Test only use this build script if -->

7 <!-- the Compile on Save feature is turned off for the project. -->

8 <!-- You can turn off the Compile on Save (or Deploy on Save) setting -->

9 <!-- in the project's Project Properties dialog box.-->

10 <project name="UserDrivenSignLanguageDictionary" default="default" basedir=".">

11 <description>Builds, tests, and runs the project UserDrivenSignLanguageDictionary.</description>

12 <import file="nbproject/build-impl.xml"/>

13 <!--

14

15 There exist several targets which are by default empty and which can be

16 used for execution of your tasks. These targets are usually executed

17 before and after some main targets. They are:

18

19 -pre-init: called before initialization of project properties

20 -post-init: called after initialization of project properties

21 -pre-compile: called before javac compilation

22 -post-compile: called after javac compilation

23 -pre-compile-single: called before javac compilation of single file

24 -post-compile-single: called after javac compilation of single file

25 -pre-compile-test: called before javac compilation of JUnit tests

26 -post-compile-test: called after javac compilation of JUnit tests

27 -pre-compile-test-single: called before javac compilation of single JUnit test

28 -post-compile-test-single: called after javac compilation of single JUunit test

29 -pre-dist: called before archive building

30 -post-dist: called after archive building

31 -post-clean: called after cleaning build products

32 -pre-run-deploy: called before deploying

33 -post-run-deploy: called after deploying

34

35 Example of pluging an obfuscator after the compilation could look like

36

37 <target name="-post-compile">

38 <obfuscate>

39 <fileset dir="${build.classes.dir}"/>

40 </obfuscate>

41 </target>

42

43 For list of available properties check the imported

44 nbproject/build-impl.xml file.

45

46

47 Other way how to customize the build is by overriding existing main targets.

48 The target of interest are:

49

50 init-macrodef-javac: defines macro for javac compilation

51 init-macrodef-junit: defines macro for junit execution

52 init-macrodef-debug: defines macro for class debugging

53 do-dist: archive building

54 run: execution of project

55 javadoc-build: javadoc generation

56

57 Example of overriding the target for project execution could look like

58

59 <target name="run" depends="<PROJNAME>-impl.jar">

60 <exec dir="bin" executable="launcher.exe">

61 <arg file="${dist.jar}"/>

62 </exec>

63 </target>

64

65 Notice that overridden target depends on jar target and not only on

66 compile target as regular run target does. Again, for list of available

67 properties which you can use check the target you are overriding in

68 nbproject/build-impl.xml file.

69

70 -->

71

72 <target name="schemaexport">

73

74 <path id="project.classpath">

75 <fileset dir="build/web/WEB-INF/lib">

76 <include name="\*\*/\*.jar"/>

77 <include name="\*\*/\*.zip"/>

78 </fileset>

79 </path>

80

81 <taskdef name="hibernatetool"

82 classname="org.hibernate.tool.ant.HibernateToolTask"

83 classpathref="project.classpath"/>

84

85 <hibernatetool destdir=".">

86 <classpath path="build/web/WEB-INF/classes" />

87 <annotationconfiguration configurationfile="src/java/hibernate.cfg.xml" />

88 <hbm2ddl

89 drop="true"

90 create="true"

91 export="true"

92 outputfilename="ddl.sql"

93 delimiter=";"

94 format="true" />

95 </hibernatetool>

96 </target>

97 </project>

98

99

Filen: ddl.sql

1

2 alter table VideoFile

3 drop constraint fk\_file\_applicationuser;

4

5 alter table VideoFile

6 drop constraint fk\_file\_word;

7

8 alter table Word

9 drop constraint fk\_word\_applicationuser;

10

11 alter table WordGroup

12 drop constraint fk\_wordgroup\_applicationuser;

13

14 alter table WordGroupWordRelation

15 drop constraint fk\_wordgroupwordrelation\_wordgroup;

16

17 alter table WordGroupWordRelation

18 drop constraint fk\_wordgroupwordrelation\_word;

19

20 drop table ApplicationUser;

21

22 drop table VideoFile;

23

24 drop table Word;

25

26 drop table WordGroup;

27

28 drop table WordGroupWordRelation;

29

30 create table ApplicationUser (

31 id bigint not null generated always as identity,

32 email varchar(50) not null,

33 emailVerificationSent timestamp,

34 emailVerified timestamp,

35 fullname varchar(50) not null,

36 login varchar(20) not null unique,

37 password varchar(20) not null,

38 userRole varchar(10) not null,

39 version integer not null,

40 primary key (id)

41 );

42

43 create table VideoFile (

44 id bigint not null generated always as identity,

45 description varchar(250),

46 fileName varchar(100) not null,

47 resourceName varchar(50) not null,

48 uploadedDateTime timestamp not null,

49 version integer not null,

50 toWord\_id bigint not null,

51 uploadedBy\_id bigint not null,

52 primary key (id)

53 );

54

55 create table Word (

56 id bigint not null generated always as identity,

57 createdDateTime timestamp not null,

58 description varchar(250),

59 version integer not null,

60 word varchar(50) not null unique,

61 requestCreatedBy\_id bigint not null,

62 primary key (id)

63 );

64

65 create table WordGroup (

66 id bigint not null generated always as identity,

67 createdDateTime timestamp not null,

68 description varchar(250),

69 name varchar(30) not null unique,

70 version integer not null,

71 createdBy\_id bigint not null,

72 primary key (id)

73 );

74

75 create table WordGroupWordRelation (

76 id bigint not null generated always as identity,

77 version integer not null,

78 word\_id bigint not null,

79 wordGroup\_id bigint not null,

80 primary key (id),

81 unique (wordGroup\_id, word\_id)

82 );

83

84 alter table VideoFile

85 add constraint fk\_file\_applicationuser

86 foreign key (uploadedBy\_id)

87 references ApplicationUser;

88

89 alter table VideoFile

90 add constraint fk\_file\_word

91 foreign key (toWord\_id)

92 references Word;

93

94 alter table Word

95 add constraint fk\_word\_applicationuser

96 foreign key (requestCreatedBy\_id)

97 references ApplicationUser;

98

99 alter table WordGroup

100 add constraint fk\_wordgroup\_applicationuser

101 foreign key (createdBy\_id)

102 references ApplicationUser;

103

104 alter table WordGroupWordRelation

105 add constraint fk\_wordgroupwordrelation\_wordgroup

106 foreign key (wordGroup\_id)

107 references WordGroup;

108

109 alter table WordGroupWordRelation

110 add constraint fk\_wordgroupwordrelation\_word

111 foreign key (word\_id)

112 references Word;

113

114

Filen: db\_init.xml

1 -- Extra constrains

2 alter table ApplicationUser

3 add constraint valid\_user\_roles

4 check (userrole in ('ADMIN', 'NORMAL'));

5

6 alter table ApplicationUser

7 add constraint password\_length\_ge\_3

8 check (length(password) >= 3);

9

10 -- Test data

11 ---------------------------------------

12 -- Users

13 insert into applicationuser

14 (login, password, userrole, fullname, email, emailverificationsent,

15 emailverified, version)

16 values

17 ('jsh', 'jsh', 'NORMAL', 'Jan Schrøder Hansen', 'jsh@jsh.dk',

18 '2011-10-29 10:10:10', '2011-10-29 10:10:12', 1);

19

20 insert into applicationuser

21 (login, password, userrole, fullname, email, emailverificationsent,

22 emailverified, version)

23 values

24 ('tki', 'tki', 'NORMAL', 'Tanja Kikkenborg', 'tki@tki.dk',

25 '2011-10-29 10:10:10', '2011-10-29 10:10:12', 1);

26

27



## 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 webside brugerflade desing.
* Kode – Indeholder al kode til projektet

# Noter

De fleste af mine noter, henviser (linker) til den engelske Wikipedia, da den engelske version af Wikipedia, ofte er mere beskrivende end den danske, og fordi Wikipedia som regel er neutral i sine beskrivelser.

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. UP – Unified Process, se [en.wikipedia.org/wiki/Unified\_Process](http://en.wikipedia.org/wiki/Unified_Process). [↑](#endnote-ref-8)
9. MagicDraw – UML tegneprogram, se [en.wikipedia.org/wiki/MagicDraw](http://en.wikipedia.org/wiki/MagicDraw). [↑](#endnote-ref-9)
10. Java Annotations – mærkning af java klasser, se [en.wikipedia.org/wiki/Java\_annotation](http://en.wikipedia.org/wiki/Java_annotation). [↑](#endnote-ref-10)
11. DDL – Data Definition Language, se [en.wikipedia.org/wiki/Data\_Definition\_Language](http://en.wikipedia.org/wiki/Data_Definition_Language). [↑](#endnote-ref-11)
12. Optimistisk låsning, se <http://en.wikipedia.org/wiki/Optimistic_concurrency_control>. [↑](#endnote-ref-12)
13. 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-13)
14. 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-14)
15. FFMPEG – et video format konverteringsprogram, se [ffmpeg.org](http://ffmpeg.org/). [↑](#endnote-ref-15)
16. Apache James – Email server, se [james.apache.org](http://james.apache.org/). [↑](#endnote-ref-16)
17. Why Wicket, se [wicket.apache.org/introduction.html](http://wicket.apache.org/introduction.html). [↑](#endnote-ref-17)
18. Struts - Web framework, se [struts.apache.org](http://struts.apache.org/) [↑](#endnote-ref-18)
19. ASP.NET – Web framework, se [en.wikipedia.org/wiki/ASP.NET](http://en.wikipedia.org/wiki/ASP.NET) [↑](#endnote-ref-19)
20. JSP – Java Server Pages, se [en.wikipedia.org/wiki/JSP](http://en.wikipedia.org/wiki/JSP) [↑](#endnote-ref-20)
21. HTML – Hyper Text Makeup Language, se [en.wikipedia.org/wiki/HTML](http://en.wikipedia.org/wiki/HTML) [↑](#endnote-ref-21)
22. JavaScript – programmeringssprog, se [en.wikipedia.org/wiki/JavaScript](http://en.wikipedia.org/wiki/JavaScript) [↑](#endnote-ref-22)
23. 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-23)
24. CSS – Cascading Style Sheets, se <http://en.wikipedia.org/wiki/Cascading_Style_Sheets> [↑](#endnote-ref-24)
25. 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-25)
26. SQL - Structured Query Language, se [en.wikipedia.org/wiki/SQL](http://en.wikipedia.org/wiki/SQL) [↑](#endnote-ref-26)
27. Wicket secure by default - se mere [wicket.apache.org/meet/features.html](http://wicket.apache.org/meet/features.html). [↑](#endnote-ref-27)
28. Se OWASP Top 10, se [www.owasp.org/index.php/Top\_10\_2007](http://www.owasp.org/index.php/Top_10_2007). [↑](#endnote-ref-28)
29. HTTPS/SSL, se [en.wikipedia.org/wiki/Transport\_Layer\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security). [↑](#endnote-ref-29)
30. 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-30)
31. 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-31)
32. SCRUM - Udviklingsproces, se [en.wikipedia.org/wiki/Scrum\_(development)](http://en.wikipedia.org/wiki/Scrum_%28development%29). [↑](#endnote-ref-32)
33. XP – eXtreme Programing, se [en.wikipedia.org/wiki/Extreme\_programming](http://en.wikipedia.org/wiki/Extreme_programming). [↑](#endnote-ref-33)
34. CMS – Content Management System, se [en.wikipedia.org/wiki/Content\_management\_system](http://en.wikipedia.org/wiki/Content_management_system). [↑](#endnote-ref-34)
35. OGG – Videoformat, se [en.wikipedia.org/wiki/Ogg](http://en.wikipedia.org/wiki/Ogg). [↑](#endnote-ref-35)
36. MPEG-4 – Videoformat, se [en.wikipedia.org/wiki/MPEG-4\_Part\_14](http://en.wikipedia.org/wiki/MPEG-4_Part_14). [↑](#endnote-ref-36)
37. Adobe Flash – Multimedieplatform fra Adobe, kan bl.a. vise video, se [en.wikipedia.org/wiki/Adobe\_Flash](http://en.wikipedia.org/wiki/Adobe_Flash). [↑](#endnote-ref-37)
38. NetBeans – Java udviklingsmiljø, se [en.wikipedia.org/wiki/NetBeans](http://en.wikipedia.org/wiki/NetBeans) [↑](#endnote-ref-38)
39. Apache SubVersion – kode versionerings system, se [en.wikipedia.org/wiki/Apache\_Subversion](http://en.wikipedia.org/wiki/Apache_Subversion). [↑](#endnote-ref-39)
40. GoogleCode - Google service der gratis hoster udviklingsprojekter, mod at de udgives som open source, se [en.wikipedia.org/wiki/Google\_Code](http://en.wikipedia.org/wiki/Google_Code). [↑](#endnote-ref-40)