IT-DIPLOMUDDANNELSEN

AFGANGSPROJEKT

af

Jan Schrøder Hansen

eFTERÅR 2011

**INDHOLD**

[1. Indledning 3](#_Toc310524858)

[2. Opgavebeskrivelse 3](#_Toc310524859)

[3. Krav 4](#_Toc310524860)

[3.1. Funktionelle Krav 4](#_Toc310524861)

[3.2. Ikke funktionelle krav 6](#_Toc310524862)

[4. Analyse 7](#_Toc310524863)

[4.1. Administrative use cases 7](#_Toc310524864)

[4. Design 12](#_Toc310524865)

[4.1. Webside design 15](#_Toc310524866)

[5. Idriftsættelse 15](#_Toc310524867)

[6. UP Iterationer 16](#_Toc310524868)

[7. Konklusion 17](#_Toc310524869)

[9. Bilag 18](#_Toc310524870)

[9.1. Danske/engelske termer 18](#_Toc310524871)

[3.5. Krav/use case matrix 19](#_Toc310524879)

[3.6. Indhold på vedlagte CD 21](#_Toc310524880)

[4. NOTER 22](#_Toc310524881)

# Indledning

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

# Opgavebeskrivelse

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

Jeg har fundet følgende 2 sider på internettet:

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

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

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

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

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

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

Løsningens navn bliver ”Tegn til tiden”.

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

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

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

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

# Krav

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

## Funktionelle Krav

|  |  |  |
| --- | --- | --- |
| ID | Krav | Prioritet |
| A1 | Systemet skal være tilgængeligt for alle ved søgninger efter ord | M |
| A2 | Systemet skal kræve at men er logget på for at uploade video filer, slette video filer, kommentere video filer og rate(bedømme) ord (video sekvenser). | 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 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 mail til en eller flere systemansvarlige brugere. Dette kræver ikke at brugeren er indlogget. | 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 adgangskode | 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 |

Rolle matrix for krav A7

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

## Ikke funktionelle krav

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

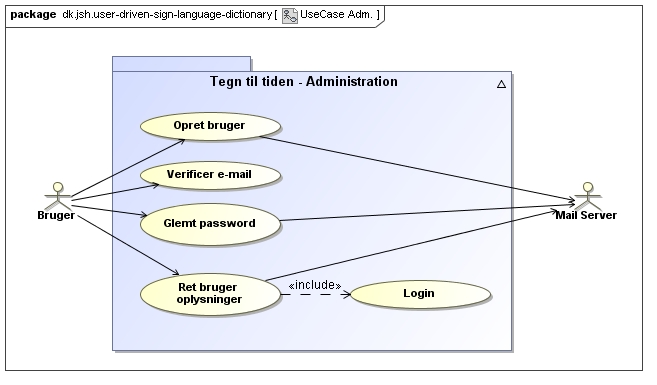
Det afslutter krav indsamlingen, og det bringer os videre til analyse fasen.

# Analyse

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

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

## Administrative use cases



Figur - Administration use case diagram

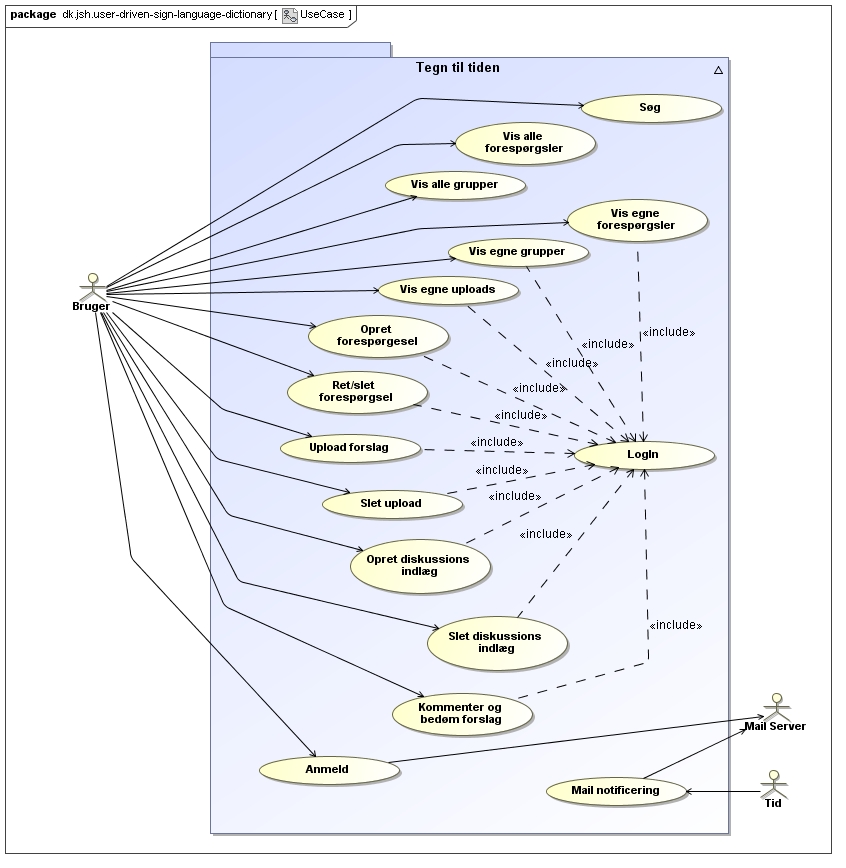
|  |
| --- |
| Use case: LOGIN |
| **ID: UC-1** |
| **Kort beskrivelse:**  En bruger logges på systemet. |
| **Primære aktører:**  Alle kendte brugere. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er kendt af systemet. |
| **Hovedforløb:**   1. Brugeren udfylder brugerkode og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

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

## Primære use cases

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



Figur – Use case diagram

|  |
| --- |
| 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 |
| **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 |
| **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 |
| **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 |
| **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 og returnerer 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 en forespørgsel til en et ord. |
| **Primære aktører:**  Alle |
| **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 |
| **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 og beskrivelsen. |
| **Slut betingelser:**  En videofil er uploadet som et forslag til et ord/forespørgsel. |
| **Alternative forløb:**   * Brugeren fortryder |

diskussionsindlæg anmeld kommenter indlæg

TODO hertil

TODO krav/use case matrix

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



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[[9]](#footnote-2), til at mappe mellem den objekt orienteret verden og den relationelle database verden. Mere om det under design.

Entitetsklasserne består af Error og Request, som begge arver fra den abstrakte Case. Case indeholder alle de attributter som er fælles for Error og Request. Der kan knyttes diskussionsindlæg til fejl og ønsker vha. klassen Diskussion, og filer på samme måde vha. klassen File. File bruges også til at knytte filer til diskussionsindlæg. Brugere kan knyttes til fejl og ønsker vha. associationerne createdBy, responsibleDeveloper og responsibleTester. Brugeren knyttes også til fejl og ønsker indirekte gennem Produkt, som bruges til hvilke Produkter de enkelte brugere har adgang til. En bruger kan have en eller flere roller vha. Role. Vote bruges til håndtering af stemmer til et ønske.

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

* MaintainCase – til håndtering af fejl og ønsker. Oprettelse, rettelse og søgning.
* MaintainDiscussion – til håndtering af diskussionsindlæg
* MaintainFile – til håndtering af filer som knyttes enten til en sag (fejl eller ønske) eller til et diskussionsindlæg
* MaintainChangelog – til håndtering af ændringer til fejl og ønsker.

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. Se pakkediagrammet under bilag afsnit 8.6.

For at illustrere de forskellige tilstande en sag kan være i, har jeg udarbejdet et solskins scenarie aktivitetsdiagram. Dette diagram viser hvilke brugerroller, som er inde over en sag samt sagens statusser igennem forløbet. Se under bilag afsnit 8.7.

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.

For at afgrænse opgaven har jeg ikke medtaget nogen analyse use case realiseringer.

# Design

Jeg er kommet frem til følgende klassediagram for mine entitetsklasser.



Figur 4 - Design entitetsklassediagram

Som det fremgår af diagrammet ligger alle mine entitetsklasser nu i en pakke kaldet dk.jsh.bari.entity, som er måden man navngiver pakker på i Java. Jeg har erstattet alle sagsbehandlings- og udviklingsprocesstatusser, samt prioritet attributter med enumerations klasser. Der er en form for Java konstanter.

De var to ”mange til mange” relationer i analysediagrammet. Mellem Produkt og User, har jeg indsat UserProductAccess og mellem User og Role, har jeg indsat UserHasRole.

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]](#footnote-3). Så Hibernate ved hvordan de enkelte klasser, skal mappes til databasetabeller. Disse annotationer kan også benyttes til at generere et DDL[[11]](#footnote-4) database skema.

Mine kontrolklasser kan ses af følgende diagram, som ligger i pakken dk.jsh.bari.business.



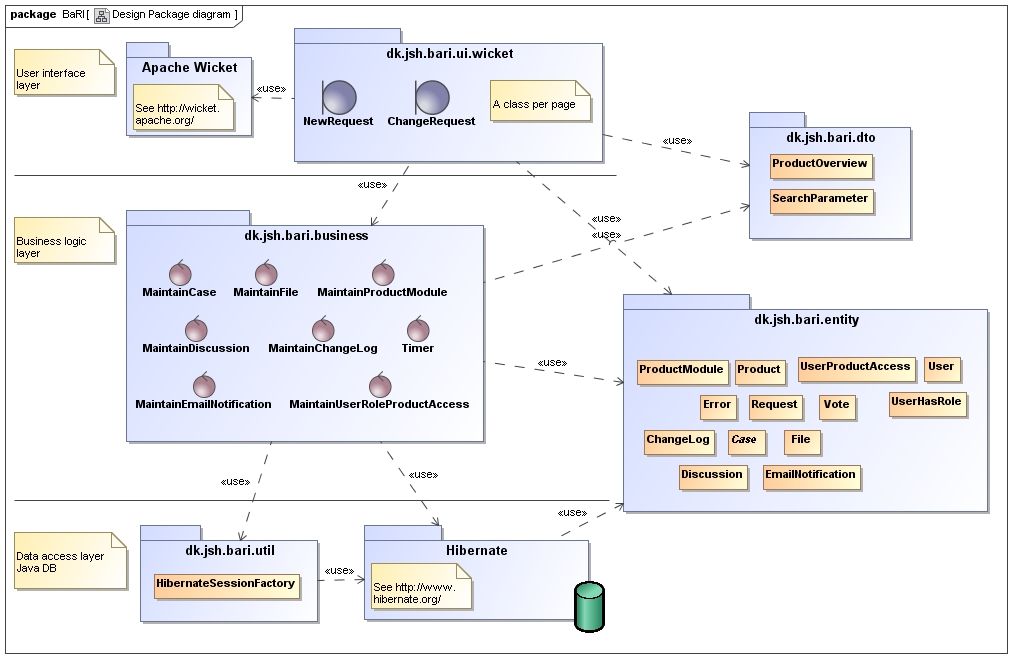
Figur 5 – Design kontrolklassediagram

Her har jeg tilføjet nogle ekstra klasser, bl.a. en Timer, som er en singleton[[12]](#footnote-5) klasse, der vha. MaintainEMailNotification klassen, starter e-mail notifikationen med faste intervaller. Dette skal foregå i en separat programtråd[[13]](#footnote-6).

Derudover er der tilføjet klasserne MaintainProduktModule og MaintainUserRoleProductAccess, som begge mangler en del metoder til at vedligeholde produkter/moduler, brugere og brugeradgang. Disse blev fravalgt i forbindelse med afgrænsning af opgaven under analysefasen. I MaintainProductModule har jeg en enkelt metode kaldet getProductOverview, som bruges til at fremvise en produktforside.

Under bilag, se afsnit 8.8, har jeg vedlagt et state machine diagram, der beskriver hvilke brugerroller, som må ændre fejl- og ønskestatusserne, sagsbehandlingsstatus og udviklingsprocesstatus.

Det samlede design kommer til at se således ud:



Figur 6 - Design pakkediagram

For at afgrænse opgaven, har jeg valgt ikke at gå i dybden med pakkerne: dk.jsh.bari.wicket, dk.jsh.bari.util og dk.jsh.bari.dto. Wicket pakken er til bruger interface klasser. Og der vil komme en klasse pr. web side. Dvs. ca. en klasse pr. use case. Pakken dto er til entitetsklasser, som ikke skal gemmes i databasen. F. eks. klassen SearchParameter, der bruges i forbindelse med use casen ”Søg efter fejl og ønsker”, samt klassen ProductOverview som bruges i forbindelse med use casen ”Vis produkt forside”. Og sidst Util klassen der bruges til initiering af Hibernate.

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

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

Pakkerne dk.jsh.bari.dto og dk.jsh.bari.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.

For at kvalitetssikre designet har jeg lavet nogle use case realiseringsdiagrammer i form af nogle sekvensdiagrammer. Se eksempler på disse under bilag afsnit 8.9.

## Webside design

Jeg er kommet frem til følge webside flow i BaRI.



Figur 7 - Side flow

Siderne ”Oversigt”, ”Søg”, ”Opret ønske” og ”Opret Fejl” kan tilgås fra alle sider i BaRI, dog ikke Login, via en fælles menulinje som går igen på alle sider.

Under bilag afsnit 8.10 kan de enkelte webside layout ses.

# Idriftsættelse

Da BaRI 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[[14]](#footnote-7), 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.

Følgende er et deployment diagram.



Figur 8 - Deployment diagram

For at deploye en webapplikation i en Tomcat server, bygges der en war[[15]](#footnote-8) 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 BaRI 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.

# 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 indledende iterationer, som BaRI kunne starte sin udvikling efter.

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

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

# Konklusion

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

Jeg er heller ikke sikker på, at UP er den rette proces for alle projekter. I dette tilfælde kunne XP[[16]](#footnote-9) eller SCRUM[[17]](#footnote-10) godt være brugt. UP har efter min mening mere sin berettigelse, hvis der er høje dokumentationskrav til projektet/produktet. Fordi der ligges stor vægt på UML diagrammer og tegninger i alle faser, samt hvilke dokumenter der er en forudsætning for at komme videre til næste fase i processen. Men til forskel for vandfaldsmodellen, så er UP dokumenterne noget der tilrettes for hver iteration.

Processer som XP og SCRUM ligger mere sin vægt på tests, simpelt design, refactoring[[18]](#footnote-11) og kørende kode, som kan bruges her og nu ellers i det mindste demonstreres.

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

# Bilag

## Danske/engelske termer

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

|  |  |  |
| --- | --- | --- |
| Dansk term | Engelsk term | Beskrivelse |
| Bruger | User | En bruger af systemet. |
|  |  |  |
|  |  |  |
|  |  |  |



## 

## Krav/use case matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Krav | 1. Opret ønske | 1. Opret fejl | 1. Opret disk. | 1. Vis/ret ønske | 1. Vis/ret fejl | 1. Vis/ret disk. | 1. Vis produktforside | 1. Søg efter ønsker og fejl | 1. E-mail notifikation | 1. Vedhæft filer | 1. Slet fil | 1. Vis/hent fil | 1. Vis ændringsloggen | 1. Adm. Use cases |
| 1 | √ |  |  | √ |  |  |  |  |  |  |  |  |  |  |
| 2 |  | √ |  |  | √ |  |  |  |  |  |  |  |  |  |
| 3 |  |  | √ |  |  | √ |  |  |  |  |  |  |  |  |
| 4 | √ | √ |  | √ | √ |  |  |  |  | √ |  |  |  |  |
| 5 | √ | √ |  | √ | √ |  |  |  |  |  |  |  |  |  |
| 6 | √ | √ |  | √ | √ |  |  |  |  |  |  |  |  |  |
| 7 | √ |  |  | √ |  |  |  |  |  |  |  |  |  |  |
| 8 |  | √ |  |  | √ |  |  |  |  |  |  |  |  |  |
| 9 | √ | √ | √ | √ | √ | √ |  |  |  | √ | √ |  | √ |  |
| 10 |  |  |  | √ |  |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  | √ |  |  |  |  |  |  |  |  |
| 12 |  | √ |  | √ | √ |  |  |  |  |  |  |  |  | √ |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  |  | √ |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  | √ |
| 15 |  |  |  | √ | √ |  |  |  | √ |  |  |  |  |  |
| 16 |  | √ |  | √ | √ |  |  |  |  |  |  |  |  |  |
| 17 | √ | √ | √ |  |  | √ |  |  |  |  |  |  |  |  |
| 18 | √ | √ |  |  |  |  |  |  |  |  |  |  |  | √ |
| 19 |  |  |  | √ | √ |  |  |  |  |  |  |  |  |  |
| 20 |  |  | √ | √ | √ | √ |  |  |  |  |  |  |  |  |
| 21 |  |  |  | √ | √ |  |  |  |  |  |  |  |  |  |

## Indhold på vedlagte CD

Indholdet på den vedlagte CD er inddelt i følgende 4 kataloger:

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

# NOTER

1. Java – Programmeringssprog, se mere [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 mere [en.wikipedia.org/wiki/Apache\_Wicket](http://en.wikipedia.org/wiki/Apache_Wicket) [↑](#endnote-ref-2)
3. Hibernate - Java Object til database framework, se mere [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 mere [en.wikipedia.org/wiki/Apache\_Derby](http://en.wikipedia.org/wiki/Apache_Derby) [↑](#endnote-ref-4)
5. Apache Tomcat – Web server, se mere [en.wikipedia.org/wiki/Apache\_Tomcat](http://en.wikipedia.org/wiki/Apache_Tomcat) [↑](#endnote-ref-5)
6. UML – Unified Modeling Language, se mere [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 mere [en.wikipedia.org/wiki/Use\_case\_diagram](http://en.wikipedia.org/wiki/Use_case_diagram) [↑](#endnote-ref-7)
8. MagicDraw – UML tegneprogram, se mere [en.wikipedia.org/wiki/MagicDraw](http://en.wikipedia.org/wiki/MagicDraw) [↑](#endnote-ref-8)
9. Hibernate – Se [www.hibernate.org](http://www.hibernate.org/) [↑](#footnote-ref-2)
10. Java Annotations – se mere [en.wikipedia.org/wiki/Java\_annotation](http://en.wikipedia.org/wiki/Java_annotation) [↑](#footnote-ref-3)
11. DDL – Data Definition Language – se mere [en.wikipedia.org/wiki/Data\_Definition\_Language](http://en.wikipedia.org/wiki/Data_Definition_Language) [↑](#footnote-ref-4)
12. Singleton – se mere [en.wikipedia.org/wiki/Singleton\_pattern](http://en.wikipedia.org/wiki/Singleton_pattern) [↑](#footnote-ref-5)
13. Programtråd – se mere [en.wikipedia.org/wiki/Thread\_(computer\_science)](http://en.wikipedia.org/wiki/Thread_%28computer_science%29) [↑](#footnote-ref-6)
14. Apache Tomcat – se mere [tomcat.apache.org](http://tomcat.apache.org/) [↑](#footnote-ref-7)
15. WAR – se mere [en.wikipedia.org/wiki/WAR\_file\_format\_(Sun)](http://en.wikipedia.org/wiki/WAR_file_format_%28Sun%29) [↑](#footnote-ref-8)
16. XP – eXtreme Programmering se mere her [en.wikipedia.org/wiki/Extreme\_Programming](http://en.wikipedia.org/wiki/Extreme_Programming) og [www.extremeprogramming.org/](http://www.extremeprogramming.org/) [↑](#footnote-ref-9)
17. SCRUM – se mere her [en.wikipedia.org/wiki/Scrum\_(development)](http://en.wikipedia.org/wiki/Scrum_%28development%29) og [www.scrumalliance.org/learn\_about\_scrum](http://www.scrumalliance.org/learn_about_scrum) [↑](#footnote-ref-10)
18. Refactoring – se mere [en.wikipedia.org/wiki/Code\_refactoring](http://en.wikipedia.org/wiki/Code_refactoring) [↑](#footnote-ref-11)