IT-DIPLOMUDDANNELSEN

Objektorienterede metoder

Eksamensprojekt

af

Jan Schrøder Hansen

Forår 2011

**INDHOLD**

[1. Indledning 3](#_Toc293835161)

[2. Opgavebeskrivelse/Krav 3](#_Toc293835162)

[3. Analyse 4](#_Toc293835163)

[4. Design 8](#_Toc293835164)

[4.1. Web side design 11](#_Toc293835165)

[5. Idriftsættelse 11](#_Toc293835166)

[4. UP Iterationer 12](#_Toc293835167)

[5. Konklusion 13](#_Toc293835168)

[6. Bilag 14](#_Toc293835169)

[6.1. Krav 14](#_Toc293835170)

[6.1.1. Funktionelle krav 14](#_Toc293835171)

[6.1.2. Ikke funktionelle krav 16](#_Toc293835172)

[6.2. Dansk/engelsk termer 17](#_Toc293835173)

[6.3. Administrtive use cases 18](#_Toc293835174)

[6.4. Use case beskrivelser 19](#_Toc293835175)

[6.5. Krav/use case matrix 32](#_Toc293835181)

[6.6. Analyse pakke diagram 33](#_Toc293835182)

[6.7. Solskins scenarie aktivitetsdiagram 34](#_Toc293835183)

[6.8. State Machine diagram for en sag 35](#_Toc293835184)

[6.9. Web Side design 36](#_Toc293835185)

[6.10. Indhold på vedlagte CD 42](#_Toc293835186)

# Indledning

Dette eksamensprojekt er lavet i forbindelse med faget Objektorienterede metoder på IT- Diplomuddannelsen, Ingeniørhøjskolen i København.

Faget har taget udgangspunkt i bogen ”UML 2 and the Unified Process” med undertitlen ” Practical Object-Oriented Analysis and Design” af Jim Arlow og Ila NeustadtRaghu Ramakrishnan. Samt materiale udarbejdet af underviser Jacob Nordfalk.

# Opgavebeskrivelse/Krav

For at komme igennem så meget af materialet i faget som muligt, har jeg valgt at lave en webløsning, til håndtering af ønsker og fejlreporter, til et eller flere softwareprodukter.

Der skal være mulighed for at følge status på fejl og ønsker. Er fejlen eller ønsket godkendt, afvist, er det under udvikling, under test etc. Derudover skal der være mulighed for at uploade forskellige filer såsom skærmdumps af fejl, eller prototyper på ny skærmlayouts og andre filer, som kan hjælpe til at belyse en sag.

Programmets navn er BaRI, som står for ”Bugs and Request Interceptor”.

Det er en løsning, jeg startede på under faget Web og Serverprogrammering efteråret 2009 og som jeg videreudviklede i faget Databasesystemer og Web efteråret 2010. Begge rapporten kan findes på den vedlagte cd.

Under faget Web og Serverprogrammering var mit fokus på et web framework kaldet Wicket[[1]](#footnote-2). En lille del af opgaven skulle også demonstrere, at løsningen kunne persistere data i en database, så der blev lavet 2 tabeller. Under faget Databasesystemer og Web var mit fokus på database delen, hvor antallet af tabeller kom op på 11. Begge løsninger kan godt betragtes som prototyper.

I denne opgave vil jeg starte forfra med krav, analyse, design af ovenstående system. Selve UP processen vil ikke fremgå af denne rapport, da det ikke svært at beskrive denne proces i rapport form. Så rapporten vil følge den gamle vandfaldsmodel, med krav, analyse, design, etc.

Funktionelle og ikke funktionelle krav til ovenstående kan ses under bilag, se afsnit 6.1. Dette skal ses som kravspecifikationer, samme med ovenstående opgave beskrivelse.

Jeg vil dog komme ind på hvordan opgaven kunne løses vha. af UP iterationer.

Da jeg altid skriver på engelsk i min kode, har jeg valgt at mine UML diagrammer også er på engelsk. Men da rapporter her er på dansk, har jeg vedlagt en dansk/engelsk ordliste under bilag. Se afsnit 6.2.

Alle diagrammer er udarbejdet vha. af programmer MagicDraw[[2]](#footnote-3).

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

# Analyse

Jeg har valg at lave 2 use case diagrammer, selv om der kun er et system. Diagrammer er opdelt efter administrative use cases og use cases som giver værdi til løsningen. De administrative use cases kan ses under bilag. Se afsnit 6.3. Disse use cases vil jeg ikke komme ind på, da de er ret trivielle CRUD[[3]](#footnote-4) use cases.

Figur 1- BaRI Use Cases

Som det fremgår af overstående, så er alle brugertyper, som også fremgår af kravene med på diagrammet. Der er fordi de enkelte brugerroller er vigtige for valgmulighederne i flere af ovenstående use cases. En Use case som login til systemet og logud, har jeg valgt helt at se bort fra, da jeg ikke mener de giver værdi i analyse fasen. Det samme gælder for en administrators mulighed for at slette ønsker og fejl. Til slut har jeg lavet en krav/use case matrix, for at sikre at alle funktionelle krav er dækket ind af overstående use cases. Se afsnit 2.1.

Herunder er use casen ”Opret ønske” beskrevet. De andre use case beskrivelser, kan ses under bilag. Se bilag afsnit 6.4.

|  |
| --- |
| **Use case:** Opret ønske |
| **ID: 1** |
| **Kort beskrivelse:**  En bruger opretter et nyt ønske |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. |
| **Hovedforløb:**   1. Brugeren vælger ”Opret nyt ønske” link. 2. Systemet viser en tom side, med følgende felter:    1. produkt skal vælges, og hvis tilgængeligt kan et modul også vælges    2. overskrift    3. beskrivelse    4. prioritet (”Skal udvikles”, ”Bør udvikles”, ”Kunne være rart at have”, ”Kan vente”)    5. e-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere    6. synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger) 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet oprettet et nyt ønske, og tildeler dette et entydigt ID, og sætter sags status til ”Oprettet”, udviklingsstatus til ”Ikke påbegyndt” og sætter en oprettet dato/tid på ønsket, samt gemmer hvilke bruger der har oprettet sagen. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. 6. include (Vedhæft filer) |
| **Slut betingelser:**  Et nyt ønske er oprettet. |
| **Alternative forløb:**  Brugeren fortryder. |

For at komme videre i analysefasen, har jeg udarbejdet nedenstående klasse diagram



Figur 2 - Analyse klasse diagram

Alle de gule klasser er entity klasser. 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[[4]](#footnote-5), til at mappe mellem den objekt orienteret verden og den relationelle database verden. Mere om det under design.

Entity klasserne består er Error og Request, som begge arver fra den abstrakte Case. Case indeholde alle de atributter som er fælles for Error og Request. Der kan knyttes diskussionsindlæg til fejl og ønsker vha. klassen Diskussion, og filer på samme måde vha. a File. Som også bruges til at knytte Filer til diskussionsindlæg. Brugere kan knyttes til fejl og ønsker vha. associationerne createdBy, responsibleDeveloper og responsibleTester. Samt 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 et stereotype) er delt om 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, så har jeg 2 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 6.2.

For at illustrere de forskelle tilstande en sag kan være i har jeg udarbejdet et solskins scenarie aktivitetsdiagram. Som viser hvilke brugerroller som er inde over en sag. Se under bilag afsnit 6.3.

Jeg har valgt at ”låse” analysen, det skal forstås på den måde at hvis jeg bliver ”klogere” senere i processen. Så går jeg ikke tilbage og tilretter analyse modellerne. Men indarbejder rettelserne i design modellerne.

TODO: Use case realisering

# Design

Jeg er kommet frem til følgende klasse diagram for mine entitets klasser.



Figur 3 - Design entitets klasse diagram

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 udviklings proces statusser, samt prioritet attributter med enumarations klasser. Som er en form for programmerings konstanter.

De var 2 mange til mange relationer i analyse diagrammer. Mellem Produkt og User har jeg indsat UserProductAccess og mellem User og Role har jeg indsat UserHasRole.

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

Alle ovenstående klasser, minus enumeration klasser, skal mærkes med Hibernate/Java annotations[[5]](#footnote-6). Så Hibernate ved hvordan de enkelte klasser skal mappes til database tabeller. Disse annotationer kan også benyttes til at genere et DDL[[6]](#footnote-7) database skema.

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



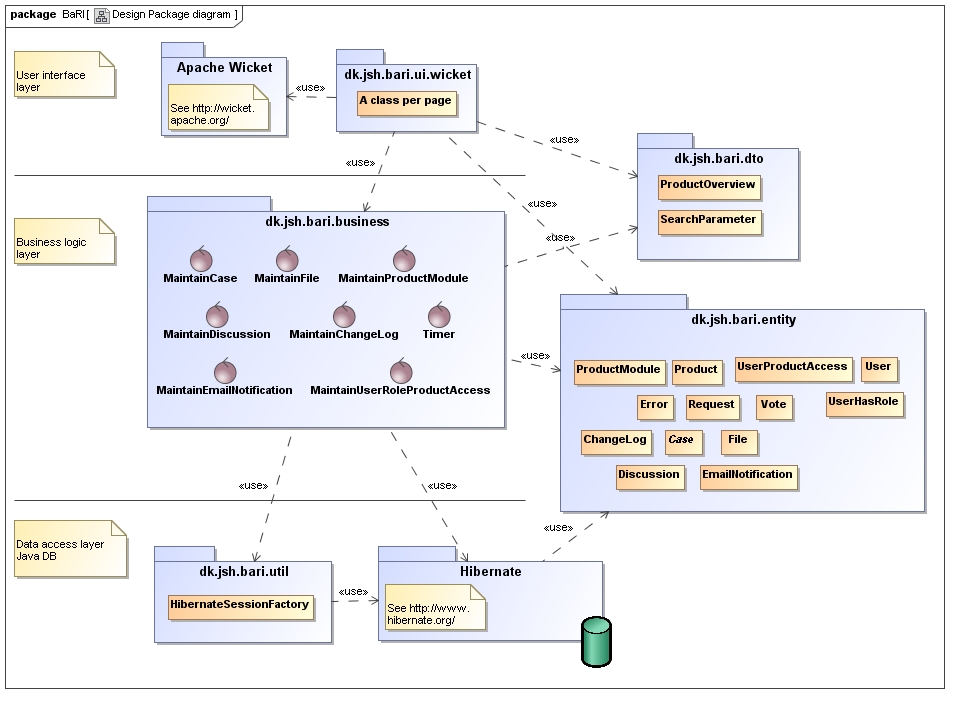
Figur 4 – Design kontrolklasse diagram

Her har jeg tilføjet nogle ekstra klasser, bl.a. en Timer, som er en singleton[[7]](#footnote-8) klasse, som vha. MaintainEMailNotification klassen, starter e-mail notifikationen med faste intervaller. Dette skal foregå i en separat program tråd[[8]](#footnote-9).

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

Under bilag se afsnit todo, har jeg vedlagt et state machine diagram, som beskriver hvilke brugerroller som må ændre fejl og ønske statusserne, sagbehandlings status og udviklings proces status.

Det samlede design kommer til at se således ud:



Figur 5 - Design pakke diagram

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. Dto pakken er til entitets klasser som ikke skal gemmes i databasen. F. eks. en klasse SearchParameter, som 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 som bruges til initiering af Hibernate.

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

* User interface – til håndtering af web siderne
* 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 skema filer.

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

## Web side design

Jeg er kommet frem til følge web side flow i BaRI.



Figur 6 - Side flow

Siderne ”Oversigt”, ”Søg”, ”Opret ønske” og ”Opret Fejl” kan tilgås fra alle sider i BaRI (minus Login) via en fælles menu linje som går igen på alle sider.

Under bilag afsnit 2.4 kan de enkelte layout ses.

# Idriftsættelse

Da BaRI er en java web applikation, uden afhængigheder til andre systemer. Så er selve idriftsættelsen enkel. Det kræver en server med installeret jave og en Apache Tomcat Web Server[[9]](#footnote-10), samt en Java DB. Java DB er en del af Java. Dvs. at der er ikke nogen specielle krav til operativ system, da Tomcat, og Java DB kan køre på bl.a. Linux, Unix og Windows. Andre web servere kunne også benyttes.

Følgende er et deployment diagram.



Figur 7 - Deployment diagram

For at deploye en web. application i en Tomcat server, bygger med en war fil, vha. sit udviklingssystem. Og denne war fil kopieres over i et specielt Tomcat applikations katalog. Første gang systemet installeres, så skal databasen klargøres. Dette gøres med et sql script. Som opretter alle tabellerne, samt indsætter en administrator bruger. Så denne kan begynde at oprette brugere og produkter i systemet.

Som det også fremgår af ovenstående diagram, så skal BaRI virke samme med gængse browsere som er på markedet p.t. Dvs. Internet Explore, Firefox, Opera, Chrome og Safari. Det vil kræve noget test. Men da det er et begrænset antal web, sider der skal laves, er det overkommeligt.

# UP Iterationer

Indtil nu har denne rapport jo 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 rapport format. Men for at komme tilbage til UP, har jeg valgt at beskrive med hvilke iterationer 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 diskussions indlæg til fejl og ønsker |
| 5 | Tilføje filer til fejl og ønsker |
| 6 | Tilføje filer til diskussionsindlæg |
| 7 | Tilføje e-mail notifikation |
|  |  |

En produktejer kunne 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. Undervejs med disse iterationer skulle BaRI løbende analyseres, designes, implementeres og testes.

# Konklusion

Denne opgave her mere være en øvelse i den gamle vandfaldsmodel. Da som jeg har skrevet tidligere, for at få opgaven til passe ind i rapport formen. Som jo 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 forudsætningen for iterative udviklingsprocesser. Man bør kunne vise noget kørende kode, til interessenter 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 kun XP[[10]](#footnote-11) eller SCRUM[[11]](#footnote-12) 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 e til næste fase i processen. Men til forskel for vandfaldsmodellen, så er UP dokumenterne noget der tilrettes for hver iteration.

Processer som XP ligger mere sin vægt på test, simpelt design, refactorering og kørende kode, som kan bruges ellers i det mindste demonstreres.

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

# Bilag

## Krav

Følgende af en liste af krav til systemet, delt op efter funktionelle og ikke funktionelle krav. De enkelte krav prioriteres efter MoSCoW (Must have, Should Have, Could Have, Want to have).

### Funktionelle krav

|  |  |  |
| --- | --- | --- |
| Id | Krav | Prioritet |
| 1 | BaRI skal kunne oprette og vedligeholde ønsker til andre systemer | M |
| 2 | BaRI skal kunne oprette og vedligeholde fejl som er fundet i andre systemer | M |
| 3 | BaRI skal kunne knytte diskussionsindlæg til både ønsker og fejl. Et indlæg kan rettes af den bruger som har oprettet dette, så længe der ikke er nyere indlæg. | M |
| 4 | BaRI skal give mulighed for at uploade forskellige filer som knyttes til enten en fejl, et ønske eller et diskusionsindlæg. |  |
| 5 | BaRI skal kunne håndtere følgende sagsbehandlings statusser for både ønske og fejl: ”Oprettet”, ”Behandles”, ”I bero”, ”Godkendt til udv.”, ”Afvist”, ”Lukket”, ”Godkendt” | M |
| 6 | BaRI skal for godkendte fejl og ønsker kunne håndtere følgende udviklingsproces statusser: ”Ikke påbegyndt”, ”Begyndt”, ”Klar til test”, ”Testet” og ”Fejl fundet” | M |
| 7 | BaRI skal kunne prioritere ønsker med følgende prioriteter: ”Skal udvikles”, ”Bør udvikles”, ”Kunne være rart at have”, ”Kan vente” | M |
| 8 | BaRI skal kunne prioritere fejl med følgende prioriteter: ”Driftkritisk”, ”Kritisk”, ”Kosmetisk” | M |
| 9 | BaRI skal logge, alle ændringer på fejl og ønsker. Skal kunne vises i brugerfladen. | C |
| 10 | BaRI skal give mulighed for at slutbrugere kan stemme hvilke ønsker der skal prioteres | C |
| 11 | BaRI skal efter kunne vise en grafisk overside pr. system. Som viser antallet af ønsker med en given status, og på samme måde for fejl. | M |
| 12 | BaRI skal kunne operere med følgende brugerroller: Administrator, Slutbruger, Udvikler, tester, projektleder. | M |
| 13 | En bruger kan have en eller flere roller | M |
| 14 | En BaRI administrator kan som den eneste oprette og rette systemer, nye brugere, bruger/system grupper | M |
| 15 | BaRI skal kunne sende E-mail notifikationer, når der er ændringer på en fejl eller et ønske. De enkelte brugere skal for ønske eller sag selv tilslutte sig notifikation | C |
| 16 | BaRI skal kunne henvise produktion versioner for fejl. Og kode revisioner for fejl og ønsker. | C |
| 17 | BaRi skal tildele alle fejl og ønsker et entydigt ID, som kan bruges til at fremsøge sagen, samt skal kunne bruges som henvisning/link fra andre sager/diskussionsindlæg. | M |
| 18 | BaRI skal give mulighed for at opdele systemer i moduler, så fejl og ønsker kan henvise til et modul inden for et system. Dette skal dog være valgfrit. | C |
| 19 | BaRI skal give mulighed for at afgive estimater på fejl og ønsker | M |
| 20 | BaRI skal give alle ikke slut-brugere mulighed for at mærke fejl, ønsker og diskussionsindlæg som ”ikke synlige for slut-brugere” | M |
| 21 | BaRI skal give en projektleder mulighed for at tildele en udvikler og en tester en fejl og et ønske. | M |

Følgende er funktionelle krav, som er opstilles i rolle matrixer. Rolle matrix (C=create, R=read, U=Update, D=Delete)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Krav id 100** | Fejl | Ønske | Diskussions-indlæg | Uploads | Bruger | Gruppe | System |
| Administrator | CRUD | CRUD | CRUD | CRUD | CRUD | CRUD | CRUD |
| Projektleder | CRU | CRU | CRU | CRU |  |  |  |
| Udvikler | CRU | CRU | CRU | CRU |  |  |  |
| Tester | CRU | CRU | CRU | CRU |  |  |  |
| Slut-bruger | CRU | CRU | CRU |  |  |  |  |

De enkelte rollers mulighed for at rette fejl og ønske statusser.

Sagsbehandlingsstatusser, brugere kan ændre status til følgende:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Krav id 101** | Oprettet | Overvejes | I bero | Godkendt til udv. | Godkendt | Afvist | Lukket |
| Administrator | J | J | J | J | J | J | J |
| Projektleder | J | J | J | J | J | J | J |
| Udvikler | J | N | N | N | N | N | N |
| Tester | J | N | N | N | N | N | N |
| Slut-bruger | J | N | N | N | J | N | N |

Udviklingsproces status:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Krav id 102** | Ikke pågyndt | Begyndt | Klar til test | Testet | Fejl fundet |
| Administrator | J | J | J | J | J |
| Projektleder | J | J | J | J | J |
| Udvikler | J | J | J | N | N |
| Tester | N | N | N | J | J |
| Slut-bruger | N | N | N | J | J |

Sagsbehandlings status og udviklingsproces status, er også beskrevet vha. af et aktivitetsdiagram, se afsnit 2.3 og et state machine diagram, se afsnit. 2.4.

Ønske prioriteter:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Krav id 103** | Skal udvikles | Bør udvikles | Kunne være rart | Kan vente |
| Administrator | J | J | J | J |
| Projektleder | J | J | J | J |
| Udvikler | N | N | N | N |
| Tester | N | N | N | N |
| Slut-bruger | J | J | J | J |

Fejl prioriteter krav:

|  |  |  |  |
| --- | --- | --- | --- |
| **Krav id 104** | Driftkritisk | Kritisk | Kosmetisk |
| Administrator | J | J | J |
| Projektleder | J | J | J |
| Udvikler | J | J | J |
| Tester | N | N | N |
| Slut-bruger | J | J | J |

### Ikke funktionelle krav

|  |  |  |
| --- | --- | --- |
| Id | Krav | Prioritet (MoSCoW) |
| B1 | BaRI skal være en WEB-Løsning | M |
| B2 | BaRI skal kunne køre på en Apache Tomcat version 6 ellere nyere | M |
| B3 | BaRI skal benytte følgende 2 java frameworks skal benyttes: Apache Wicket and Hibernate | M |
| B4 | BaRI skal persistere data i en JavaDB | M |
| B5 | BaRI skal kunne benyttes sammen med HTTPS | M |
| B6 | BaRI skal i sin brugerflade tilbyde følgende 2 sprog: Dansk og engelsk | M |
| B7 | BaRI skal benytte optimistisk låsning | M |

## Dansk/engelsk termer

|  |  |  |  |
| --- | --- | --- | --- |
| Gruppe | DK term | EN term | Beskrivelse |
| **Bruger roller** | Administrator | Administrator | En BaRI administrator. Som kan oprette, rette og slette produkter og brugere. |
|  | Projektleder | Project Manager |  |
|  | Slut-bruger | End user |  |
|  | Udvikler | Developer |  |
|  | Tester | Tester |  |
| **Entiteter** | Sag | Case | En abstrakt sag. |
|  | Ønske | Request |  |
|  | Fejl | Error |  |
|  | Sagsbehandlings status | Case status |  |
|  | Udviklings proces status | Developing status |  |
|  | Fejl prioritet | Error priority |  |
|  | Ønske prioritet | Request priority |  |
|  | Fil | File |  |
|  | Diskussion | Discussion |  |
|  | Ændrings log | Changelog |  |
|  | Produkt | Product |  |
|  | Produkt modul | ProductModule |  |
|  | E-mail notifikation | E-Mail notification |  |
|  | Stemme | Vote |  |
|  | Rolle | Role |  |
| **Sagsbehandlings statusser** | Oprettet | Created |  |
|  | Behandles | Consider |  |
|  | I bero | Suspended |  |
|  | Godkendt til udvikling | Approved for development |  |
|  | Afvist | Rejected |  |
|  | Lukket | Closed |  |
|  | Godkendt | Approved |  |
| **Udviklingsproces statusser** | Ikke påbegyndt | Not started |  |
|  | Begyndt | Started |  |
|  | Klar til test | Ready to test |  |
|  | Fejl fundet | Failed |  |
|  | Testet | Tested |  |
| **Fejl prioriteter** | Driftkritisk | Operation critical |  |
|  | Kritisk | Critical |  |
|  | Kosmetisk | Cosmetic |  |
| **Ønske prioriteter** | Skal udvikles | Must be developed |  |
| s | Bør udvikles | Should be developed |  |
|  | Kunne være rart at have | Can wait |  |
|  | Kan vente | Nice to have |  |

## Administrtive use cases

Figur - Administrative Use Cases

## Use case beskrivelser

Herunder er de enkelte use case beskrivelser. De fleste use cases er lig med en web side. Disse kan findes under afsnit 6.5.

|  |
| --- |
| **Use case:** Opret ønske |
| **ID: 1** |
| **Kort beskrivelse:**  En bruger opretter et nyt ønske |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. |
| **Hovedforløb:**   1. Brugeren vælger ”Opret nyt ønske” link. 2. Systemet viser en tom side, med følgende felter:    1. produkt skal vælges, og hvis tilgængeligt kan et modul også vælges    2. overskrift    3. beskrivelse    4. prioritet (”Skal udvikles”, ”Bør udvikles”, ”Kunne være rart at have”, ”Kan vente”)    5. e-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere    6. synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger) 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet oprettet et nyt ønske, og tildeler dette et entydigt ID, og sætter sags status til ”Oprettet”, udviklingsstatus til ”Ikke påbegyndt” og sætter en oprettet dato/tid på ønsket, samt gemmer hvilke bruger der har oprettet sagen. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. 6. include (Vedhæft filer) |
| **Slut betingelser:**  Et nyt ønske er oprettet. |
| **Alternative forløb:**  Brugeren fortryder. |

|  |
| --- |
| **Use case:** Opret fejl |
| **ID: 2** |
| **Kort beskrivelse:**  En bruger opretter en ny fejl. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. |
| **Hovedforløb:**   1. Brugeren vælger ”Opret ny fejl”. 2. Systemet viser en tom side, med følgende felter:    1. produkt skal vælges, og hvis tilgængeligt kan et modul også vælges    2. overskrift    3. beskrivelse    4. prioritet (Driftkritisk”, ”Kritisk”, ”Kosmetisk”)    5. e-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når denne fejl rettes af andre brugere    6. produktversion    7. synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger)    8. include (Vedhæft fil) 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet oprettet et nyt ønske, og tildeler dette et entydigt ID, og sætter sags status til ”Oprettet”, udviklingsstatus til ”Ikke påbegyndt” og sætter en oprettet dato/tid på ønsket, samt gemmer hvilke bruger der har oprettet sagen. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. |
| **Slut betingelser:**  En ny fejl er oprettet. |
| **Alternative forløb:**  Brugeren fortryder. |

|  |
| --- |
| **Use case: Opret diskussionsindlæg** |
| **ID: 3** |
| **Kort beskrivelse:**  En bruger opretter et nyt diskussionsindlæg til en fejl eller et ønske. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren ”står” på enten et ønske eller en fejl, som denne ønsker at oprette et diskussionsindlæg på. |
| **Hovedforløb:**   1. Brugeren vælger at ”Tilføj indlæg” linket. 2. Systemet giver mulighed for at indtaste følgende:    1. selve indlægget. Her skal det være muligt at skrive #sagsid for at linke til en anden sag.    2. synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger)    3. include (Vedhæft filer) 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet opretter et nyt diskussionsindlæg, som knyttes til den aktuelle sag (ønske eller fejl) og den aktuelle bruger. Tidspunkt for oprettelse af indlægges gemmes også. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. |
| **Slut betingelser:**  Et diskussionsindlæg er oprettet og knyttet til et ønske eller en fejl. |
| **Alternative forløb:**  Brugeren fortryder. |

|  |
| --- |
| **Use case:** Vis/ret ønske |
| **ID: 4** |
| **Kort beskrivelse:**  En bruger retter oplysninger på et eksisterede ønske. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. Og det pågældende ønske er til et af disse produkter. |
| **Hovedforløb:**   1. Brugeren kommer til ønsket enten via et link fra en anden sag, eller fra et link fra en søge side. (Se use case Søg efter ønsker og fejl) 2. Systemet viser en side, med følgende felter:    1. produkt (låst)    2. evt. modul (låst)    3. overskrift    4. beskrivelse    5. oprettelsesdag (låst)    6. prioritet (”Skal udvikles”, ”Bør udvikles”, ”Kunne være rart at have”, ”Kan vente”)    7. e-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere    8. synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger)    9. konklusion    10. revisions id (Vises kun for slut-bruger og tester, andre brugere kan rette)    11. estimat (Vises kun for slut-bruger, andre brugere kan rette)    12. stemme, har den aktuelle bruger stemt på dette ønske (Kun for slut-brugere)    13. antal stemmer dette ønske har i alt (låst)    14. sagsbehandlingsstatus (”Oprettet”, ”Behandles”, ”I bero”, ”Godkendt”, ”Afvist”, ”Lukket”). Hvis et ønske er Afvist eller Lukket, kan ingen data rettes, medmindre en Administrator eller en projektleder retter status tilbage til enten Oprettet, Behandles eller I bero.    15. udviklingsstatus (”Ikke påbegyndt”, ”Begyndt”, ”Klar til test”, ”Testet”, ”I produktion”)    16. Hvis bruger er projektleder        1. Mulighed for at tildele en ansvarlig udvikler        2. og tester    17. Derudover vises alle eksisterende diskussionsindlæg. Med mulighed for at rette sidste, hvis dette er oprettet af samme bruger.    18. include (Vedhæft filer)    19. include (Slet filer)    20. include (Vis/hent fil)    21. include (Vis ændrings loggen) 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet rettet et ønske. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. |
| **Slut betingelser:**  Et ønske er rettet. |
| **Alternative forløb:**  Brugeren fortryder. |

|  |
| --- |
| **Use case:** Vis/ret fejl |
| **ID: 5** |
| **Kort beskrivelse:**  En bruger retter oplysninger på en eksisterede fejl. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. Og den pågældende fejl er til et af disse produkter. |
| **Hovedforløb:**   1. Brugeren kommer til fejlen enten via et link fra en anden sag, eller fra et link fra en søge side. (Se use case Søg efter ønsker og fejl) 2. Systemet viser en side, med følgende felter:    1. produkt (låst)    2. evt. modul (låst)    3. overskrift    4. beskrivelse    5. oprettelsesdag (låst)    6. prioritet (Driftkritisk”, ”Kritisk”, ”Kosmetisk”)    7. e-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere    8. synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger)    9. konklusion    10. revisions id (Vises kun for slut-bruger og tester, andre brugere kan rette)    11. Estimat (Vises kun for slut-bruger, andre brugere kan rette)    12. Stemme, har den aktuelle bruger stemt på dette ønske (Kun for slut-brugere)    13. Sagsbehandlingsstatus (”Oprettet”, ”Behandles”, ”I bero”, ”Godkendt”, ”Afvist”, ”Lukket”) Hvis en fejl er Afvist eller Lukket, kan ingen data rettes, medmindre en Administrator eller en projektleder retter status tilbage til enten Oprettet, Behandles eller I bero.    14. Udviklingsstatus (”Ikke påbegyndt”, ”Begyndt”, ”Klar til test”, ”Testet”, ”I produktion”)    15. Hvis bruger er projektleder        1. Mulighed for at tildele en ansvarlig udvikler        2. og tester    16. Derudover vises alle eksisterende diskussionsindlæg. Med mulighed for at rette sidste, hvis dette er oprettet af samme bruger.    17. include (Vedhæft filer)    18. include (Slet filer)    19. include (Vis/hent fil)    20. include (Vis ændrings loggen) 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet rettet et ønske. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. |
| **Slut betingelser:**  En fejl er rettet. |
| **Alternative forløb:**  Brugeren fortryder. |

|  |
| --- |
| **Use case: Vis/ret diskussionsindlæg** |
| **ID: 6** |
| **Kort beskrivelse:**  En bruger vises og får mulighed for at rette et diskussionsindlæg, for enten et ønske eller en fejl. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. Og dette indlæg er til et ønske eller en fejl, til et af disse produkter.  Det er kun det sidste indlæg, der kan rettes. Og det er kun den bruger som har oprettet indlægget, som kan rette dette. |
| **Hovedforløb:**   1. Brugeren vælger ret knappen på det sidste diskussionsindlæg. Knappen vil kun være synlig hvis det er den samme bruger som har oprettet dette indlæg. 2. Systemet giver mulighed for at rette følgende:    1. selve indlægget. Her skal det være muligt at skrive #sagsid for at linke til en anden sag.    2. synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger)    3. include (Vedhæft filer)    4. include (Slet fil) 3. Brugeren retter felterne og vælger at gemme. 4. Systemet retter et nyt diskussionsindlæg 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. |
| **Slut betingelser:**  Et diskussionsindlæg er rettet. |
| **Alternative forløb:**  Brugeren fortryder. |

|  |
| --- |
| **Use case: Vis produkt forside** |
| **ID: 7** |
| **Kort beskrivelse:**  En bruger vises en produkt forside. Med en statusoversigt over antallet af fejl og ønsker. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. |
| **Hovedforløb:**   1. Brugeren vælger et af et af de produkter denne har adgang til. Siden er forudfyldt med data for det første produkt i listen. (Sorteret alfabetisk) 2. Systemet viser 4 lagkage diagrammer: (med antal i hvert stykke)    1. Et for alle ønsker, opdelt efter sagsbehandlingsstatus    2. Et for alle ønsker, opdelt efter udviklingstatus    3. Et for alle fejl, opdelt efter sagsbehandlingsstatus    4. Et for alle fejl, opdelt efter udviklingsstatus 3. Under ønske diagrammerne skal det totale antal ønsker stå. 4. Under fejl diagrammerne skal det totale antal fejl stå. 5. Samt en liste med de 10-20 fejl/ønsker til dette produkt, som seneste er oprettet eller rettet. Med direkte link til disse. |
| **Slut betingelser:** |
| **Alternative forløb:**  Brugeren fortryder. |

|  |
| --- |
| **Use case:** Søg efter ønsker og fejl |
| **ID: 8** |
| **Kort beskrivelse:**  En bruger har mulighed for at fremsøge fejl og ønsker. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren er oprettet i systemet. |
| **Hovedforløb:**   1. Brugeren vælger et af et af de produkter denne har adgang til. 2. Hvis det valgte produkt, har tilknyttet moduler    1. Så kan bruger også vælge imellem disse, eller alle moduler som er valgt på forhånd. 3. Brugeren vælger mellem fejl, ønske eller begge. Begge er valgt på forhånd. 4. Brugeren vælger sagsbehandlingsstatus, her er alle også valgt på forhånd. 5. Brugeren vælger udviklingsstatus, her er alle også valgt på forhånd. 6. Brugeren har mulighed for et udfylde et fritekst søgefelt. Som søger i alle tekst felter for fejl, ønsker og diskussionsindlæg. 7. Brugeren trykker på søg knappen 8. Systemet finder alle de sager som opfylder ovenstående søgekriterier, og viser disse en liste med følgende kolonner:    1. Id, type (fejl eller ønske), overskrift, oprettet dato, sidste rettelsesdato, sagsbehandlingsstatus, udviklingsstatus, prioritet og antal stemmer. Listen kan sorteres ved at trykke på de enkelte kolonneoverskrifter. 9. Brugeren kan vælge at trykke på en linje i listen for at få denne vist. (Se use cases: Vis/ret ønske og Vis/ret fejl) |
| **Slut betingelser:** |
| **Alternative forløb:**   * Der skal være en mulighed for at indtaste et sags id, så brugeren har mulighed for at gå direkte til en sag vha. dette id. * Brugeren fortryder. |

|  |
| --- |
| **Use case:** E-mail notification |
| **ID: 9** |
| **Kort beskrivelse:**  Systemet har mulighed for at sende e-mail notifikationer til en bruger |
| **Primære aktører:**  Systemet (Tiden) |
| **Sekundære aktører:**  Alle brugere af systemet |
| **Start betingelser:**  Brugernes e-mail adresser findes og er valide. |
| **Hovedforløb:**   1. En gang i timen mellem 7 og 22    1. For hver fejl og ønske en bruger har ønsket at blive notificeret om       1. Systemet undersøger om der er sket ændringer på sagen, vha. ændringsloggen       2. Hvis dette er tilfældet          1. Systemet sender en e-mail til denne bruger, med følgende oplysninger             1. Fejl/eller ønske id             2. Overskrift på sagen,             3. Hvis det sidste er nye diskussionsindlæg så vises disse             4. Link til sagen |
| **Slut betingelser:**  En eller flere brugere er blevet notificeret. |
| **Alternative forløb:** |

|  |
| --- |
| **Use case: Vedhæft filer** |
| **ID: 10** |
| **Kort beskrivelse:**  En bruger vedhæfter en eller flere filer til et ønske, en fejl eller et diskussionsindlæg. |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Et ønske, en fejl eller en et diskussionsindlæg findes som filerne kan tilknyttes |
| **Hovedforløb:**   1. Brugeren vælger ”upload filer” linket. 2. For hver fil som en bruger ønsker at uploade    1. Aktøren udfylder en kort beskrivelse af filens indhold, samt vælger den fil der skal uploades.    2. Systemet gemmer: beskrivelse, filnavn, dato/tid for upload, samt fil data og knytter filen til enten et ønske, fejl eller et diskussionsindlæg.    3. Systemet opretter en post i ændringsloggen med ovenstående data |
| **Slut betingelser:**  En eller flere filer er uploadet. |
| **Alternative forløb:**  Ingen |

|  |
| --- |
| **Use case:** Vis/hent fil |
| **ID: 11** |
| **Kort beskrivelse:**  Mulighed for at vise eller hente en fil (Så den kan gemmes lokalt). |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren har adgang til produktet. Hvortil filen er knyttet enten via en fejl, et ønske eller et diskussionsindlæg. |
| **Hovedforløb:**   1. Brugeren vælger ”vis fil” linket, på enten en fejl, ønske eller diskussionsindlæg. Der kan være 0 eller mange. 2. Systemet streamer den aktuelle fil til brugens browser 3. Browseren giver kan vælge at vise filen, hvis denne er i et format den kan arbejde med. Som mange billedeformater, html, pdf etc. 4. Ellers giver browseren mulighed for at gemme filen lokalt. |
| **Slut betingelser:**  En fil er vist eller hentet og gemt lokalt. |
| **Alternative forløb:**  Brugeren fortryder |

|  |
| --- |
| **Use case:** Slet filer |
| **ID: 12** |
| **Kort beskrivelse:**  Mulighed for at vise eller hente en fil |
| **Primære aktører:**  Alle aktører har denne mulighed. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren har adgang til produktet. Hvortil filen er knyttet enten via en fejl, et ønske eller et diskussionsindlæg. |
| **Hovedforløb:**   1. Brugeren vælger ”slet fil” linket, på enten en fejl, ønske eller diskussionsindlæg. Der kan være 0 eller mange. På diskussionsindlæg er det kun det sidste indlæg der kan slettes filer fra. Og det er kun af samme bruger som har lavet indlægget. 2. Systemet sletter filen. |
| **Slut betingelser:**  En fil er slettet. |
| **Alternative forløb:**  Brugeren fortryder |

|  |
| --- |
| **Use case:** Vis ændringsloggen |
| **ID: 13** |
| **Kort beskrivelse:**  Se ændringsloggen for en fejl eller et ønske. |
| **Primære aktører:**  Alle aktører har denne mulighed. Dog ikke slut-brugere. Da der i loggen kan være oplysninger som slut-bruger ikke må se. Se Feltet ”synlig for slut-bruger” i tidligere use cases. |
| **Sekundære aktører:**  Ingen |
| **Start betingelser:**  Brugeren har adgang til produktet. Hvortil en fejl, et ønske eller er tilknyttet. |
| **Hovedforløb:**   1. Brugeren er på et ønske eller en fejl. (Se use cases ”Vis/ret fejl” og ”Vis/ret ønske”) 2. Brugeren vælger ”Vis ændringslog linket” 3. Systemet viser en side med følgende oplysninger: (Listeform, nyeste først)    1. dato/tid for ændringen    2. bruger    3. brugers rolle    4. om det er en oprettelse, rettelse eller en sletning    5. type: fejl, ønske eller diskussionsindlæg    6. overskrift 4. Brugeren kan trykke på en af ovenstående ændringer 5. Systemet viser i dfri tekst hvad ændringerne er. (F.eks. før og efter tekster på forskellige felter på en fejl eller et ønske, eller et teksten fra en diskussion indlæg som er tilføjet etc.) |
| **Slut betingelser:**  Brugeren har set ændringsloggen for en fejl eller et ønske. |
| **Alternative forløb:** |



## 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 produkt forside | 1. Søg efter ønser 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 |  |  |  | √ | √ |  |  |  |  |  |  |  |  |  |

## Analyse pakke diagram



Figur 9 - Analse pakke diagram

Som det fremgår af overstående diagram, så har jeg lagt mine klasser i 2 pakker. Business og Entity pakkerne. Business håndterer kald fra brugerfladen til databasen. Og Entity klasserne er databærende klasser.

## Solskins scenarie aktivitetsdiagram

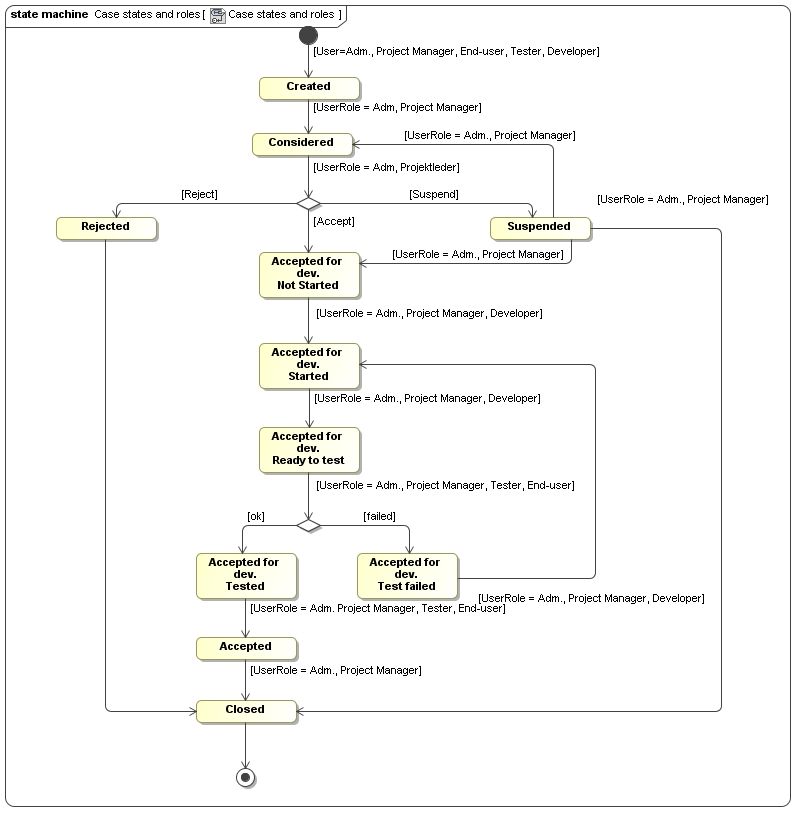


Figur 10 – Solskins scenarie aktivitetsdiagram

Ovenstående aktivitetsdiagram beskriver et solskins scenarie for et ønske. På request objektet kan ses sagsbehandlingsstatus og udviklingsproces status (I den kantede parentes).

## State Machine diagram for en sag

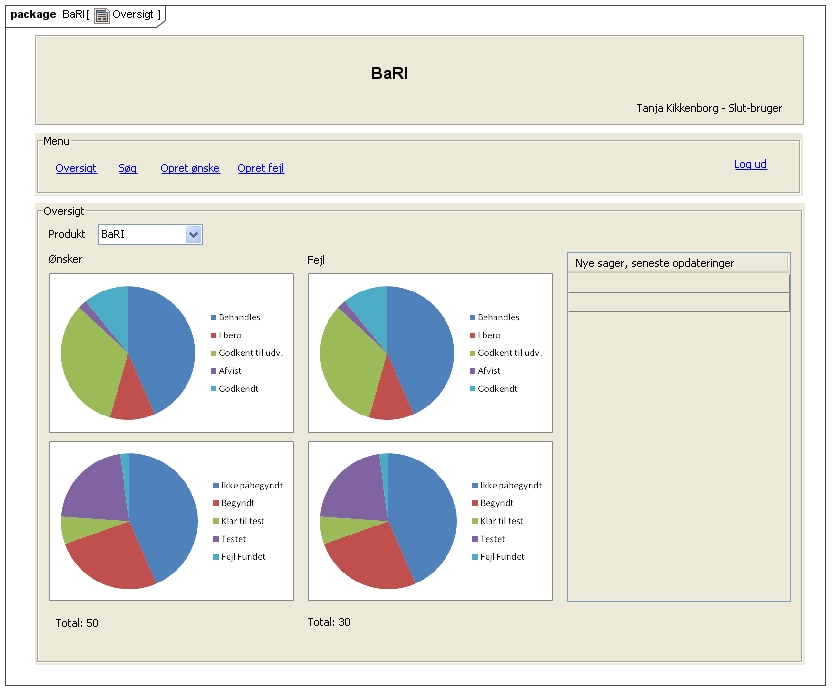
Diagrammet herunder viser hvilke brugerroller som kan ændre de 2 sags statusser som er fælles for både fejl og ønsker. Alle brugere kan oprette fejl og ønsker.



Figur 11 - State machine diagram for Request og Error

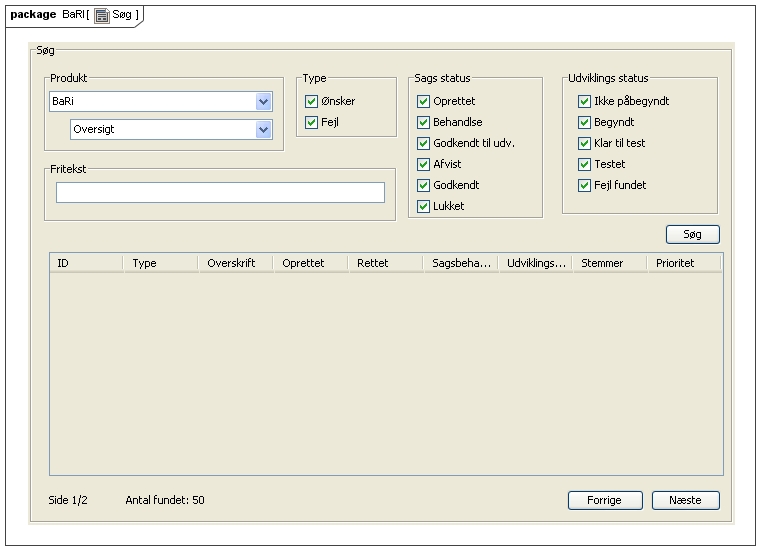
## Web Side design

Følgede web side er den første som vises efter login.

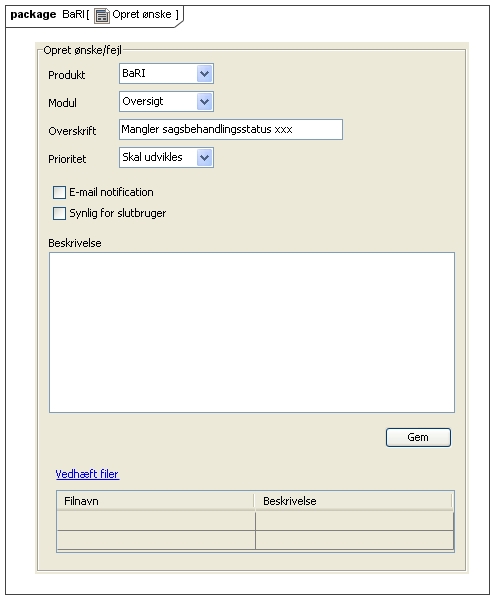


Figur 12 - Oversigt

Overskrift boksen og menuboksen er fælles for alle sider (dog ikke login siden), og vises derfor kun på ovenstående figur.

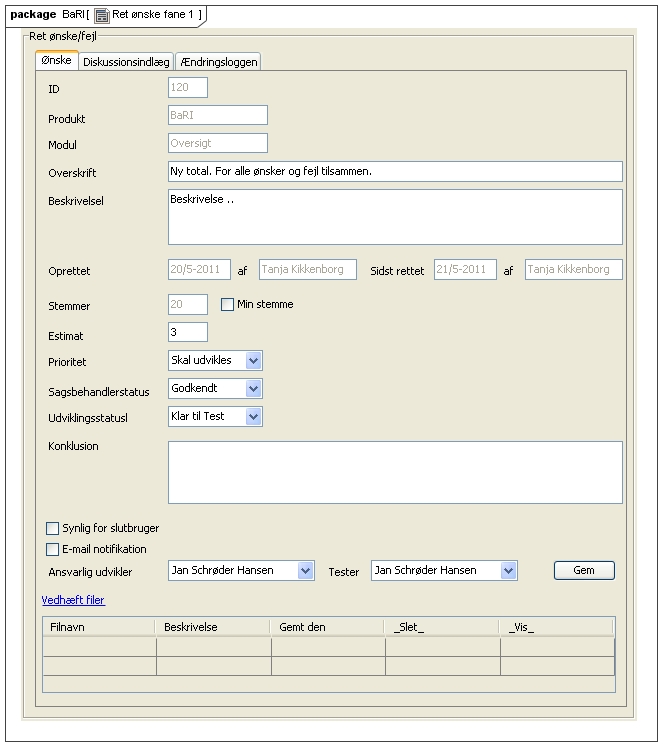


Figur - Søg



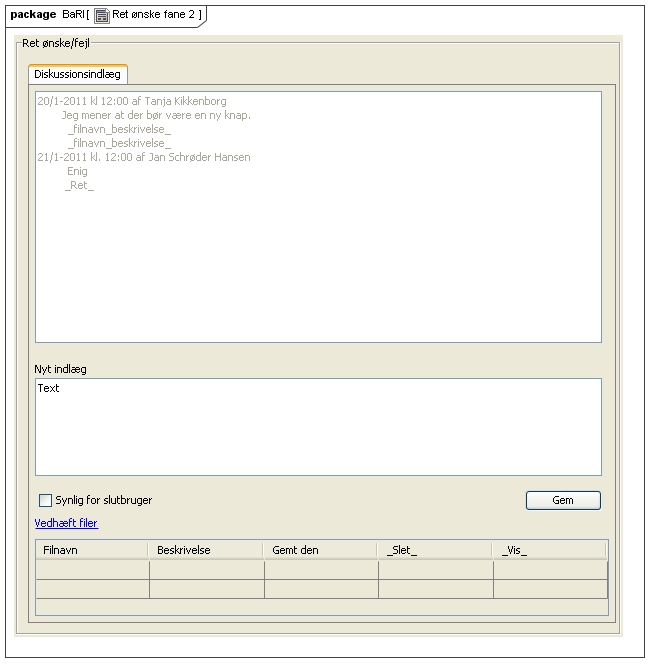
Figur 14 - Opret ønske eller fejl

Opret ønske eller fejl, adskiller sig kun i hvilke prioritet der kan vælges. Linket Vedhæft filer er først synlig efter at sagen er gemt første gang. Dvs. at tryk på gem knappen bliver på siden.



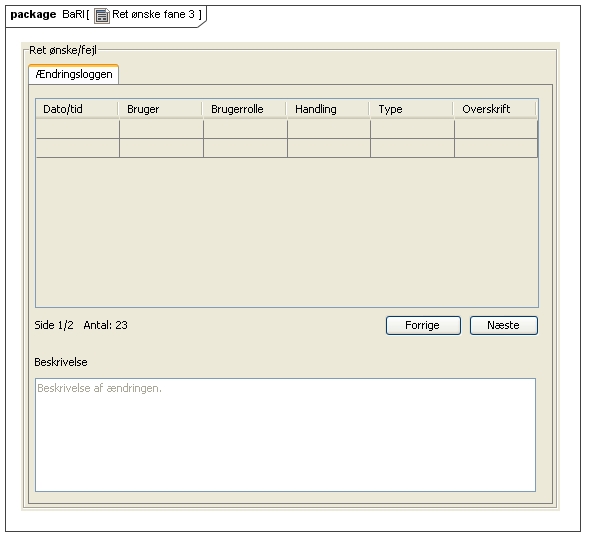
Figur 15 - Ret ønske eller fejl – første fane

For hver fil der er vedhæftet, er der 2 links. Et slet link, som viser en ”er du sikker dialog” før der slettes. Og et vis link, som henter filen. Og browseren giver mulighed for at vise filen hvis den kender formatet, eller at gemme filen lokalt. Hvis brugeren har rollen administrator, så ville der også være en Slet knap på denne side.



Figur 16 - Ret ønske eller fejl – anden fane

Den store tekstboks i toppen, indeholder alle tidligere indlæg. I dato/tid rækkefølgen. Hvert indlæg skrives med dato/tid og bruger. Ny indrykket linje indrykket med selve teksten. Og hvis der er vedhæftet filer vises disse som links med filnavn og beskrivelse. På det sidste indlæg er der et ret link, hvis den aktuelle bruger, er den sammen som har lavet indlægget. Hvis ret linket vælges så kommer teksten og evt. filer ned i nyt indlæg delen. Når et nyt indlæg gemmes, kan teksten ses med det samme i den store tekstboks i toppen.



Figur 17 - Ret ønske eller fejl – tredje fane

I toppen af ændringsloggen er der en tabel med alle ændringer til en fejl eller et ønske. Hver linje i tabellen er et link. Hvis der trykkes på linket så vises selve rettelsen i ”Beskrivelse” tekstenboksen. F. eks. hvis er status er rettet, så skrives før værdien og efterværdien.

F. eks. ”Udviklingsstatus er rettet fra ”Klar til test” til ”Testet”

## 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
* Databasesystemer og WEB – Indeholder rapporten til faget Databasesystemer og WEB i PDF format.
* WEB og Serverprogrammering – Indeholder rapporten til faget WEB og Serverprogrammering i PDF format.
* MagicDraw – Indeholder to MagicDraw projekter. Et for analyse og et for Design og implementering.

1. Wicket – se <http://wicket.apache.org/> [↑](#footnote-ref-2)
2. MagicDraw – se [www.magicdraw.com](http://www.magicdraw.com/) [↑](#footnote-ref-3)
3. CRUD – Create, Read, Update og Delete operationer [↑](#footnote-ref-4)
4. Hibernate – Se [www.hibernate.org](http://www.hibernate.org/) [↑](#footnote-ref-5)
5. Java Annotations – se mere [en.wikipedia.org/wiki/Java\_annotation](http://en.wikipedia.org/wiki/Java_annotation) [↑](#footnote-ref-6)
6. DDL – Data Definition Language – se mere [en.wikipedia.org/wiki/Data\_Definition\_Language](http://en.wikipedia.org/wiki/Data_Definition_Language) [↑](#footnote-ref-7)
7. Singleton – se mere [en.wikipedia.org/wiki/Singleton\_pattern](http://en.wikipedia.org/wiki/Singleton_pattern) [↑](#footnote-ref-8)
8. Program tråd – se mere [en.wikipedia.org/wiki/Thread\_(computer\_science)](http://en.wikipedia.org/wiki/Thread_%28computer_science%29) [↑](#footnote-ref-9)
9. Apache Tomcat – se mere [tomcat.apache.org](http://tomcat.apache.org/) [↑](#footnote-ref-10)
10. 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-11)
11. 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-12)