Udvikling af udvidelsesmodul til Drupal

Christopher von Würden F. Eriksen, s072604, Kongens Lyngby d.23. januar 2012.
Diplomingeniør Teknologi og Økonomi afsluttende eksamensprojekt.

Abstract: Opgavens formål er at udvikle et udvidelsesmodul til indholdsstyringssystemet Drupal der integrerer et eller flere eksisterende moduler og gør brug af Facebooks API, projektet udføres ved brug af PHP, MySQL, HTML, CSS og Javascript, og inkluderer herunder teknisk design og udviklingsfaser.

Danmarks Tekniske Universitet
Institut for Informatik og Matematisk Modellering
Bygning 321, 2800 Kongens Lyngby
Telefon: 45 25 33 51, Fax: 45 88 26 73
reception@imm.dtu.dk
www.imm.dtu.dk

Indholdsfortegnelse

1. Prolog

1.1 Tilkendegivelser

Denne opgave er udarbejdet i forbindelse med et afsluttende eksamensprojekt på Danmarks Tekniske Universitet, og afslutter en uddannelse som diplomingeniør i Internetteknologi og Økonomi. Projektet er udført i samarbejdet med B14 ApS, hvor virksomhedsvejleder Martin S. Dahl har været behjælpelig med, at stille gunstige rammer om projektet til rådighed.

Projektet har været vejledt af DTU vejleder Finn Gustafsson, som har hjulpet godt til, at sætte projektet rigtigt i gang. Der skal også lyde en tak til alle medarbejdere hos B14 for støtte og hjælp under hele forløbet, og en tak til venner og familie.

1.2 Forord

Drupal, et af de i dag mest populære indholdsstyringssystemer, leverer i sin standardudgave en lang liste af features, en af systemets mest markante fordele, er muligheden for, at udvide det med udvidelsesmoduler, der tilføjer nærmest enhver tænkelig funktionalitet.

Med udgangspunkt i eksisterende udvidelsesmoduler vil jeg ud fra relevante kriterier udvikle eget modul til Drupal systemet, der integrerer dele af Facebooks API, ikke tidligere integreret, og dermed tilføjer funktionalitet i interaktionen mellem Drupal og Facebook. Projektet belyser hvilke dele der understøttes af Drupal, mulighederne og hvilke funktioner tilbudt af Facebook, som ikke allerede findes i eksisterende moduler, og som kunne være mest interessante at integrere i et udvidelsesmodul.

Der lægges hermed op til en analyse af mulighederne for interaktion med og brug af Facebooks udbudte funktionalitet igennem tredjepart.

1.3 Baggrund

B14 ApS er et lille designbureau med speciale i internetløsninger og grafisk materiale i digitalt format, deres kerneprodukt er komplette hjemmesideløsninger med Drupal CMS som back end. I juni 2011 var en kampagne ved navn Local Heroes startet af Stimorol godt i gang, denne kampagne foregik udelukkende på Facebook og havde Drupal som back end. De ønskede en slags funktionalitet til, at komme i kontakt med de brugere der deltog i konkurrencen, mens B14 tænkte at det kunne være interessant, at se på mulighederne inden for denne problemstilling.

På grund af stigende efterspørgsel efter produkter til Facebook, ønskes Drupal på den baggrund udvidet med funktioner eksisterende udvidelsesmoduler ikke leverer, således at eventuelle produktpakker henvendt Facebook, har mere at tilbyde end på nuværende tidspunkt.

Drupal udviklere der ønsker integration med Facebook bruger på nuværende tidspunkt et modul ved navn ”Facebook for Drupal”, dette modul gør det muligt, at logge ind i Drupal systemet med en Facebook konto, hvor systemet automatisk opretter en Drupal-bruger, som kædes sammen med den pågældende Facebook

profil. Local Heroes ønskede i den forbindelse et system, hvori de via en applikation på Facebook, som brugere ”synes godt om” ville have mulighed for, at sende beskeder til disse brugere igennem Facebooks beskedsystem. Systemet blev ikke realiseret pga. begrænsninger i Facebook Api, det er ikke muligt, at sende beskeder fra en applikation til profiler på Facebook.

For at handle på vegne af en bruger eller få adgang til en brugers profil igennem en applikation, kræves brugerens samtykke, i Facebooks API hedder dette samtykke et ”Access Token”, hvilket egentlig er en adgangsnøgle. APIet stiller fra start en række oplysninger fra profiler til rådighed uden samtykke, men fælles for dem er, at det kun er basale oplysninger der kan læses på den måde. Ved at få adgang til udvidede rettigheder, såkaldte extended permissions, kan en applikation også handle på vegne af brugeren f.eks.

Dette er indbygget i det nuværende modul Drupal for Facebook, således at man i processen ved log-in spørger brugeren om vedkommende vil give samtykke, det Access Token systemet får tilbage bliver herefter gemt i en database, hvis undermodulet til denne funktion er slået til.

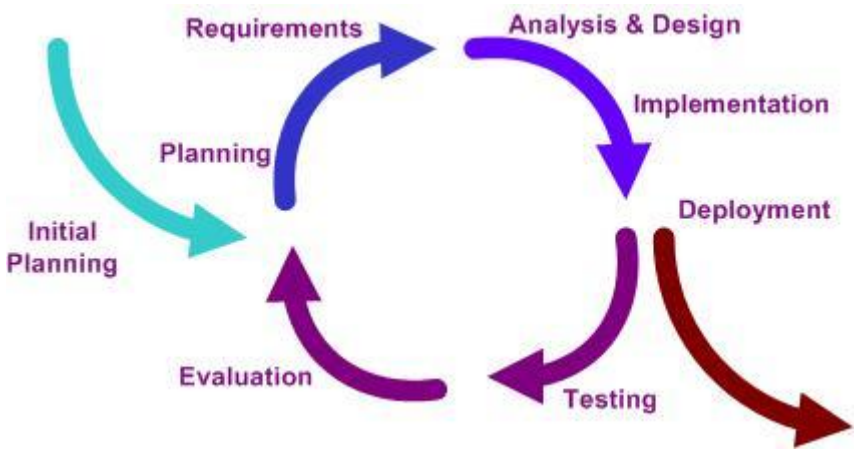
Herefter er det dog trægt, hvad man kan stille op med det Access Token man nu har fået. I alt tilbyder Facebooks API omkring 60 udvidede rettigheder, hvoraf kun få udnyttes på nuværende tidspunkt, det er derfor interessant, at undersøge og analysere sig frem til, hvilke rettigheder der kunne være relevante, at integrere i Drupal, samt bruge nogle af rettighederne som alternativ til føromtalte problemstilling vedrørende udsendelse af meddelelser.

For at vende tilbage til Local Heroes, omhandler dette projekt således udviklingen af et modul til Drupal, der gør brug af Facebooks udvidede rettigheder via det eksisterende modul Facebook for Drupal, og herved bidrager med helt ny funktionalitet til Drupal. Kort sagt der ønskes udviklet funktionalitet, der gør det muligt at sende beskeder til brugere på Facebook fra Drupal.

3. Udviklingsmetoder

Til de fleste softwareprojekter er det som oftest en fordel, at benytte en eller flere modeller til at organisere projektforløbet. En af det mest brugte er Unified Process[1] typerne, der findes i en række udgaver tilpasset forskellige områder, således også softwareudvikling. Unified Process, eller bare UP, er baseret på en iterativ og inkrementel udviklingsproces. UP er ikke et fast sæt regler man skal følge, men nærmere en udviklingsramme, hvori der findes en række metoder. Som brændstof til UP bruges gerne UML, hvoraf Use Cases (Brugsscenarier) er en vigtig del.

Før der gås lidt mere i dybden med UP er det relevant med et indblik i processen bag, den iterative og inkrementelle udviklingsproces. At processen er iterativ vil sige, at processen kører i ring og består af enkelttrin kaldet iterationer, hvilket er procestrin der færdiggøres før cyklussen fortsættes. At den er inkrementel betyder, at hver gang iterationscyklussen genstarter, vil udviklingsprojektet være nået et højere niveau end før.



Figur 1: En iterationscyklus

En iterativ udviklingsmetode er ofte at foretrække i store udviklingsprojekter, hvor små dele af en større konstruktion produceres og testes løbende.

Den iterative og inkrementelle cyklus er den ene del af Unified Process, den anden del er en ramme hvori hele UP projektet indsættes, denne består af 4 faser:

Inception (forberedelse) i hvilken projektets omfang fastslås. Denne inkluderer baggrund for projektet, risikoanalyse, kravspecifikation på baggrund af use cases, projektplan med videre.

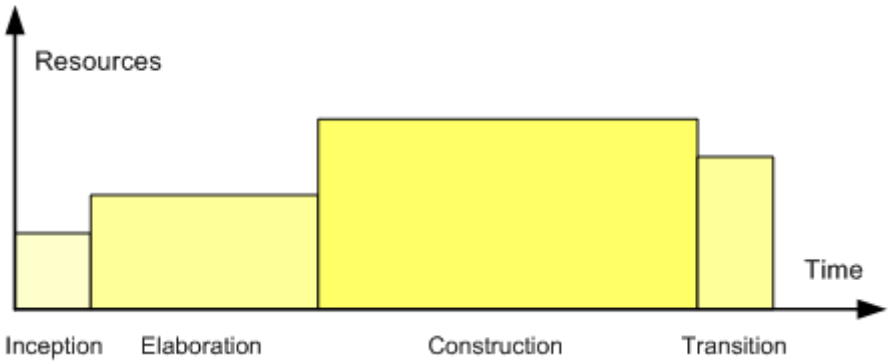
Elaboration (etablering) er anden fase i hvilken der skabes en forståelse af risikofaktorer og hvor der udarbejdes use case diagrammer, klassediagram og pakkediagram fra Unified Modelling Language, UML. I slutningen af etableringsfasen, kan der være udarbejdet en begrænset prototype, hvor de mest relevante risici er adresseret, på denne baggrund kan en mere præcis tidsplan for konstruktionsfasen etableres.

[1] http://da.wikipedia.org/wiki/Unified_Process

Construction (konstruktion) er hermed næste fase, hvori produktionen af produktet udføres, dette er den mest omfattende fase i og med hele systemet konstrueres på baggrund af planudformningen i etableringsfasen. Det er i denne fase iterationscyklussen kommer til sin ret, hvor systemdele udvikles med denne som ramme. Efter hver iteration står en mere færdig version af projektets tilstræbte resultat klar. Typisk tager hver iteration udgangspunkt i en use case. Af UML diagrammer bruges typisk, aktivitets-, sekvens-, kommunikations-, fase- og interaktionsdiagrammer.

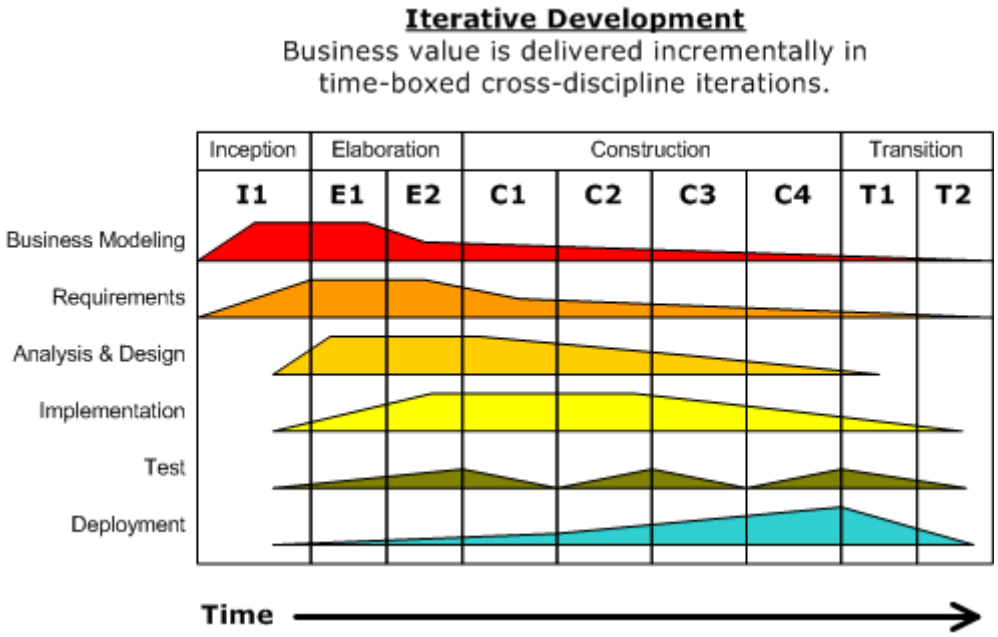
Transition (overdragelse) er den sidste fase, heri bliver systemet kørt ud til brugerne, fra hvilke, feedback kommer tilbage og eventuelle mindre rettelser eller optimeringer kan forekomme. Denne fase inkluderer også oplæring af brugere i systemet, installation og test.

Herunder ses de fire fasers relative andel af tidsforbrug igennem projektforløbet:



Figur 2: Fasernes relative arbejdsbyrde

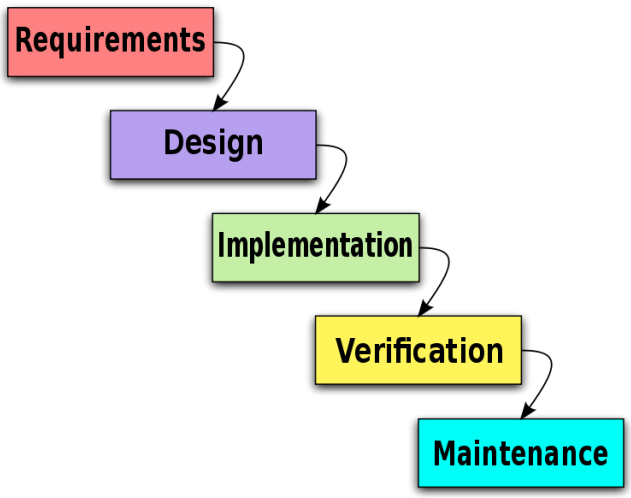
En samlet oversigt over faseforløbet og arbejdsbyrden i hver fase i løbet af projektets løbeperiode illustreres i modellen herunder:



Figur 3: Illustration af projektfasernes vægt over tid

Unified Process kan benyttes i alle større og mindre projekter, om end UP i et lille projekt med relativt få iterationscyklusser med fordel kan bruges kombineret med mere simple udviklingsmetoder.

En sådan er vandfaldsmodellen[2]. I modsætning til UP modellen er vandfaldsmodellen ikke iterativ, men derimod processuel, hvor fremskridt "flyder" nedad i modellen igennem en række faser:



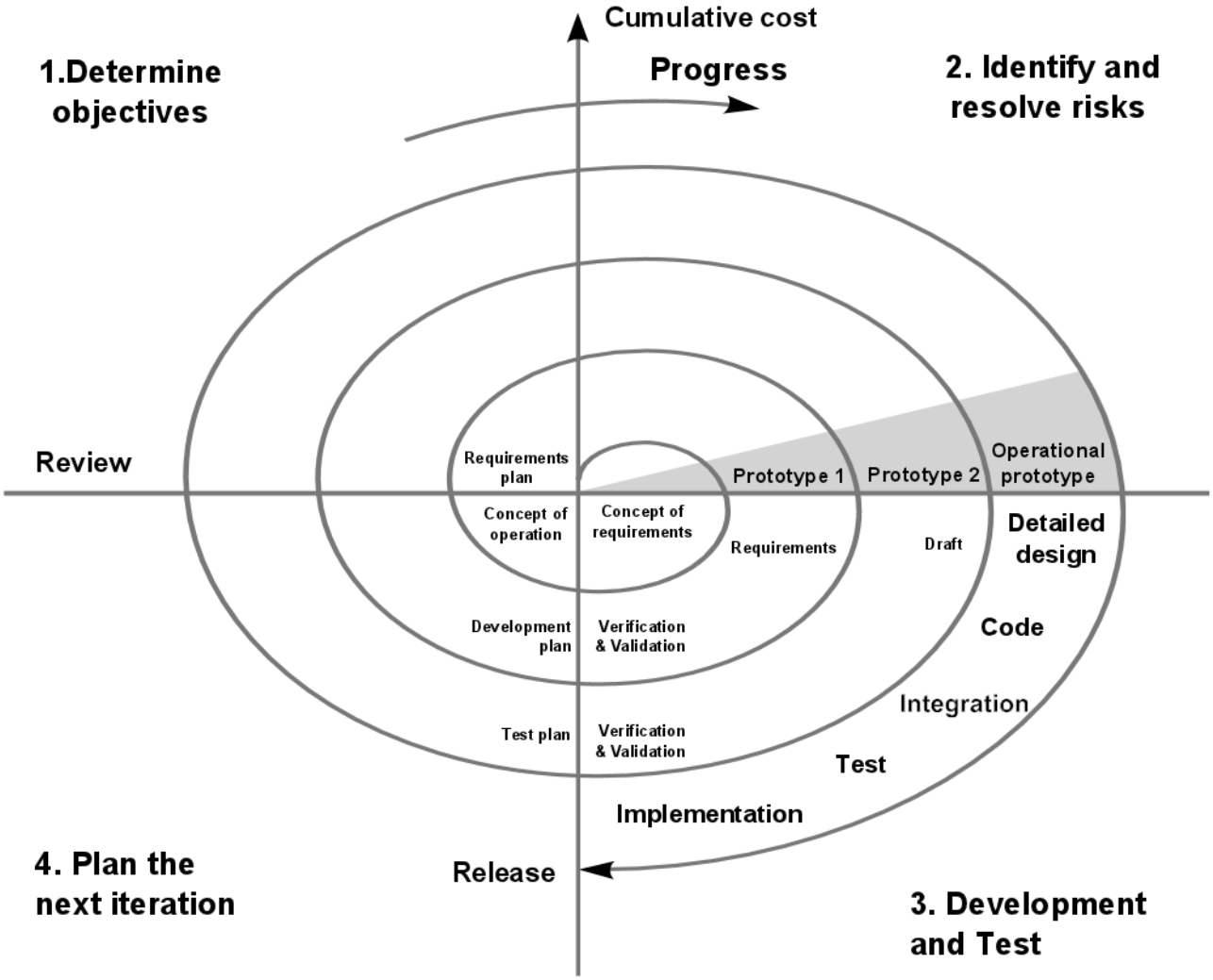
Figur 4: Vandfaldsmodellens faser

Selve faserne er meget lig, hvad der lige er gennemgået i UP modellen, forskellen fra UP består i, vandfaldsmodellens enkleste form, at hver fase afsluttes for hele projektforløbet definitivt efter den er gennemgået, dvs. næste fase påbegyndes ikke, før forrige fase er afsluttet og perfektioneret. Modellen er mest nyttig ved små projekter, da perfekt afslutning af systemdele i hver fase for et stort projekt, kan være svær at opnå.

Vandfaldsmodellen kommer til kort i et projekt med mange ukendte faktorer og UP kan virke noget uoverskueligt i et relativt lille projekt, fordi den er meget stringent og bureaukratisk i sin proces, en kombination af de to kan derfor være at foretrække.

En kombination af de to, hvor man gennemgår iterationscyklussen for hele projektet uden, at dele det op i iterationer som UP foreskriver, men i stedet kombinerer det med de trin, der er i vandfaldsmodellen kunne være spiralmodellen[3], som er simplere end UP modellen, men mere avanceret end vandfaldsmodellen, samtidig med at den tager højde for risikoanalyse. Spiralmodellen er en iterativ model som UP med de faste aspekter som vandfaldsmodellen, den er inkrementel, hvilket vil sige programmet bliver lidt mere færdigt for hvert omløb. Hvert omløb resulterer i en prototype, som der ved næste omløb bygges videre på, således dokumenteres programmet også løbende eftersom der bygges mere og mere på.

[2] http://www.waterfall-model.com/
[3] http://en.wikipedia.org/wiki/Spiral_model



Figur 5: Spiralmodellen

I analyseafsnittet diskuteres udviklingsmetoderne og det vil blive belyst, hvilken der egner sig bedst til projektet.

4. Tidsplan

Der findes en pænere version af denne på Cd'en i form af et Excel-skema, tidsplanen er den faktiske plan, dvs. den viser hvornår enkeltdele af projektet er afsluttet.

Foranalyse		Arbejdstilrettelæggelse		Analyse		Design	
Mål	Dato	Mål	Dato	Mål	Dato	Mål	Dato
Foreløbig indledning	07-sep	Foreløbig litteraturliste	12-sep	Problem-formulering(er)	23-sep	Dokumentation af design	19-dec
Foreløbig Disposition	08-sep			Kravsspecifikation	07-okt		
Forberedelse og afholdelse af vejledermøde	09-sep			Udarbejdelse af use cases	12-okt		
Aktiviteter	**Dato**	**Aktiviteter**	**Dato**	**Aktiviteter**	**Dato**	**Aktiviteter**	**Dato**
Projektoverblik	07-sep	Informationssøgning	08-sep	Risikoanalyse	10-okt	Opsætning i Drupal	09-dec
		Gennemgang af foreløbig disposition	08-sep	Analyse af relevante emner til implementering	06-okt	Forberedelse til implementering	09-dec
		Baggrund for projektet	09-sep	Analyse af Drupal - om CMS	29-sep	Design af database	09-dec
				Løsningsmodeller	28-sep	UML	29-nov
				Simplenews modulet	05-okt	Use cases	25-nov
				Facebook for Drupal modulet	05-okt	Grænseflade	12-dec
				Målgruppe	19-okt	Processuel vs. OOP	14-dec
				Formål	14-sep	Udarbejdelse af grundstruktur	13-dec

Fortsætter næste side.

Implementering		Test		Dokumentation og evaluering	
Mål	Dato	Mål	Dato	Mål	Dato
Dokumentation af implementering	12-jan	Dokumentation af test	19-jan	Evaluering	20-jan
		Black box	18-jan	Korrekturlæsning	21-jan
		White box	17-jan	Aflevering	23-jan
Aktiviteter	Dato	Aktiviteter	Dato	Aktiviteter	Dato
PHP og MySQL	02-jan	Evaluere use cases	12-jan	Skrive konklusion	20-jan
HTML, CSS og javascript	02-jan	black box	18-jan	Revidere disposition	21-jan
Implementering af use cases	23-dec	White box	17-jan	Appendiks	22-jan
Database implementering	23-dec	Unit test	15-jan	Litteraturliste	22-jan
Optimering	04-jan	Feedback på kode	16-jan	Brugervejledning	21-jan
Validering af formularer	05-jan	Gennemgang af kode	03-jan	Layout	22-jan
Drupal implementering	23-dec	Brugervenlighed	19-jan	Kontrollere formalia	22-jan
		Kompatibilitet	15-jan	forord	21-jan
		Fremtid(forbedring)	19-jan		

dato	Noter
24-okt	Risikoanalyse og kravspecifikation udskudt til 26. okt. Pga. arbejde, designafsnit påbegyndes umiddelbart efter.
01-dec	Designafsnit forsinket en uge pga. sygdom, færdiggørelse rykket til 19. december

5. Teknologi

5.1 Introduktion til Drupal

Drupal er et gratis open source (åben kildekode) indholdsstyringssystem, eller CMS, skrevet i PHP og samtidig en af de mest populære back-ends til hjemmesider, hvor pt. 1,7 % af alle hjemmesider benytter Drupal[4], svarende til omkring 700.000[5], hertil skal det siges at 54,4 % af alle hjemmesider benytter en form for CMS.

Drupal har som mange andre open source projekter en enorm brugerbase, og med omkring 12.000 udvidelsesmoduler[6], til at udvide kernen med funktionalitet, er det samtidig et af de CMS med størst fleksibilitet. I modsætning til den nærmeste konkurrent, Joomla, er Drupal kernen meget basal, og udviklingen af udvidelsesmoduler tilskyndes derfor af organisationen. Forskellen består i, hvad du står med efter en frisk installation, Joomla tilbyder en base med meget funktionalitet sat op fra start, i modsætning til Drupal, hvis startside kun tilbyder navigation og log ind. Filosofien er således at man I Drupal kun installerer præcis den funktionalitet man har behov for, samtidig med at udviklingen af udvidelser er gjort så simpel som mulig igennem deres API.

Drupal kernen[7] består af en række funktioner standard for de fleste content management systemer, herunder oprettelse og vedligeholdelse af brugere, styring af navigation, RSS feeds, side layout styring, indholdsstyring og administration. Kerneudgaven kan bruges til såvel små brochureware[8] websider, som til store portaler, community drevne eller e-handelssider.

Indhold i Drupal oprettes som, hvad de kalder, ”noder”. En node kan sidestilles med en artikel i en avis, hvis vi forestiller os at hele avisen sidestilles med Drupal. En node indeholder en række felter (fields), der kan oprettes efter behov, det kan eksempelvis være et felt der viser et billede, hvem der er forfatter el.lign.

Drupal er skrevet i PHP og benytter SQL til database, B14, hvorhos projektet skrives, benytter en Apache webserver kørende under Linux og med MySQL som database, det er derfor i disse rammer projektet udføres.

[4] http://w3techs.com/technologies/overview/content_management/all
[5] http://drupal.org/
[6] Tallet stammer fra drupal.org i september 2011
[7] Kernen henviser til en frisk installation, dvs. uden udvidelsesmoduler udover kernemoduler.
[8] Som navnet hentyder til, en hjemmeside der kun tilbyder samme som en trykt brochure.

5.2 Introduktion til Facebook platformen

Facebook behøver sandsynligvis ikke nærmere introduktion, men overordnet kan det siges at Facebook er en stor interaktiv blå bog.

Det interessante for projektet er naturligvis mulighederne stillet til rådighed af Facebook, kendt som Facebook platformen. Platformen er en række APIer og værktøjer, der giver tredjepartsudviklere mulighed for at integrere gennem applikationer, såkaldte Facebook Applications, igennem eksterne hjemmesider eller mobile enheder.

Kernen i Facebook platformen er Graph APIet, der giver udviklere mulighed for, at læse og skrive data til og fra Facebook, det være sig oplysninger tilgængelige i en persons profil såsom billeder, fødselsdag, uddannelse, venneliste etc.

En oversigt over alle værktøjerne tilgængelige i Facebooks API, kan ses på Facebook developer site[9].

5.2.1 Sociale plug-ins

En anden vigtig del af deres API er de sociale plug-ins, såsom kommentarer, "synes godt om"-knapper, live feed osv. Disse kan indlejres på enhver side med ganske få linjer HTML og således gøre en normalt statisk side mere dynamisk og brugerorienteret selv med små midler.

5.2.2 Open Graph Protocol

Open Graph protokollen gør udviklere i stand til, at integrere deres egne sider med Facebooks graph objekter, det er f.eks. link til egen profil på Facebook, opdateringer til en profils væg, se venneliste og lign. Kort sagt giver protokollen mulighed for, at flytte Facebook over på en ekstern side og lade brugerne benytte mange af de samme funktioner, de allerede kender fra Facebook.

Ved hjælp af Meta-data i en HTML sides header, kan enhver hjemmeside omdannes til Open Graph objekter. Meta data indeholder information såsom titel, type, billede, beskrivelse med mere, når et link til en side indsættes i Facebook, ved Facebook med det samme præcis, hvilket indhold den skal præsentere fra den pågældende side.

5.2.3 Facebook Connect

Facebook Connect er et API der giver udviklere mulighed for, at lade brugere på deres site logge ind igennem Facebook. I stedet for at have et separat log ind system på siden, hvor brugeren skal oprette en bruger først, kan vedkommende i stedet benytte sit eksisterende brugernavn og password fra Facebook. Når en bruger er logget ind, kan han eksempelvis poste indhold fra siden til sin Facebook profil, chatte med venner eller skrive kommentarer til indhold på siden.

Ved at benytte en Facebook applikation, kan udvikleren skabe større interaktion mellem brugere, ved f.eks. at poste beskeder på vedkommendes væg ved ny highscore i et spil eller lignende.

[9] http://developers.facebook.com

Facebook Connect er baseret på teknologien OAuth[10], der er en åben standard for autorisering, som gør brugere i stand til at dele privat indhold fra en side til en anden uden at opgive log ind oplysninger til tredjeparten.

5.2.4 Permissions

En central del af bruger- eller applikationsinteraktionen med Facebook gør brug af bestemte rettigheder, på engelsk permissions[11], disse gør en hjemmeside i stand til at foretage bestemte handlinger på vegne af en bruger, side eller en applikation.

En essentiel del af dette er et access token, der er en slags adgangsnøgle, som giver tredjepart adgang via Facebook Connect og Open Graph protokollen. Der findes forskellige slags access tokens alt efter hvem man ønsker at handle på vegne af, således bruges forskellige tokens henholdsvis for at handle på vegne af en brugerprofil, en applikation eller en side (Facebook page).

Uden access token har tredjepart kun adgang til ganske basale informationer, såsom navn, køn og land, herunder ses et eksempel på oplysninger tilgængelige om mig uden access token:

```
{
   "id": "1545047934",
   "name": "Christopher von W\u00fcrden",
   "first_name": "Christopher",
   "last_name": "von W\u00fcrden",
   "link": "http://www.facebook.com/people/Christopher-von-
W\u00fcrden/1545047934",
   "gender": "male",
   "locale": "da_DK"
}
```

Bemærk at `\u00fc` er en pladsholder for tegnet *ü*.

Facebook opdeler deres permissions i tre grupper, user permissions, friends permissions og extended permissions. User permissions giver adgang til information i en brugerprofil, herunder eksempelvis fødselsdag, ”om mig”, begivenheder brugeren er tilmeldt, hvad brugeren synes godt om etc.

Friends permissions giver rettigheder til det samme som user, men for brugerens venner, frem for brugeren selv, du kan dermed f.eks. se, hvornår en bestemt brugers venner alle har fødselsdag.

Fælles for friends og user permissions er at de kun giver adgang til at læse data fra Facebook, de udvidede rettigheder, extended permissions, giver mulighed for også at skrive data til Facebook samt læsning af nogle brugerspecifikke oplysninger beliggende uden for brugerprofilen, dette er eksempelvis adgang til, at læse indbakken, læse en brugers væg, se indkommende friend requests o. lign. Rettigheder til at skrive omfatter muligheden for, at oprette begivenheder på vegne af en brugerprofil eller poste opdateringer/beskeder på brugerprofilens væg. Af de mere kuriøse rettigheder, der kan spørges om, findes adgang til at rette i vennelister, sende sms beskeder til brugeren eller ændre status på notifikationer.

[10] http://oauth.net/
[11] http://developers.facebook.com/docs/reference/api/permissions/

Der findes også en fjerde type, der bør nævnes, som giver tredjepart adgang til at administrere Facebook pages.

En samlet liste over tilgængelige permissions, i skrivende stund, kan ses i bilag 5, dog henvises til Facebook for en opdateret liste [11], da opdateringer forekommer jævnligt.

5.3 Drupal for Facebook

I forbindelse med udviklingen af et udvidelsesmodul til Drupal er det interessant at se på, hvilke muligheder eksisterende moduler tilbyder, i den kontekst er modulpakken Drupal for Facebook af størst nævneværdighed, eftersom den tilbyder en række af den funktionalitet de fleste kunne ønske sig af en Facebook integration i Drupal.

Nogle af hovedfunktionerne i modul porteføljen er muligheden for, at give brugere af Facebook mulighed til at logge ind på ens Drupal website, hvor de automatisk får oprettet en lokal bruger på Drupal sitet, som er forbundet til en Facebook brugerprofil. Den anden hovedfunktion er muligheden for at integrere Facebook applikationer i en side samt oprette et kanvas[12] på Facebook, således at ens Drupal site kan tilgås og bruges igennem Facebook. Derudover findes en række mindre udvidelser, herunder integration af Facebooks sociale plug-ins, "synes godt om"-knapper, invitation af venner til pågældende side eller publicering af indhold til ens brugerprofils væg. Modulet giver også mulighed for, at adskille ens side fra ens kanvas, dvs. at indhold kan vises forskelligt alt efter om det ses igennem et kanvas eller ej, hvilket er brugbart eftersom Facebook f.eks. har en kanvas begrænsning i bredden på 760 pixels.

5.4 Database

Drupal understøtter en lang række forskellige database teknologier såsom MySQL, MSSQL, PostgreSQL osv. der alle fungerer sammen med PHP4 og PHP5. MySQL og MSSQL er de to mest populære databaseteknologier til web. I dette projekt vil ikke forelægge en analyse af den mest optimale teknologi, da B14 stiller krav om at MySQL bruges, Wikipedia tilbyder dog en omfattende oversigt over forskelle på de forskellige teknologier[13].

5.5 Værktøjer

Modulet udvikles i PHP til Drupal version 6.x på en Linux server med MySQL og Apache webserver kørende, til udvikling benyttes teksteditoren Notepad++ og internet browseren Mozilla Firefox, til databasen benyttes desuden phpMyAdmin til, at kontrollere om data indsættes korrekt. Udviklingscomputeren kører desuden Windows 7 Professional Edition med Service Pack 1.

Til UML diagrammer er programmerne Violet og Microsoft Office Visio 2007 benyttet.

[12] http://developers.facebook.com/docs/guides/canvas/#canvas
[13] http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

6. Kravspecifikation

6.1 Problemområdet

Drupal platformen integrerer Facebook igennem modulpakken Drupal for Facebook, der tilbyder en lang række nyttige tilføjelser til systemet, heriblandt muligheden for at registrere en Facebook applikation i Drupal for igennem denne, at muliggøre brugen af Drupal som CMS til applikationen. Herudover er integrationen af Facebook Connect vigtig, idet systemet får adgang til oplysninger om brugerne via Facebook, frem for, at de selv skal indtastes. I forbindelse med Facebook Connect tilbydes et modul til, at gemme oplysninger om brugere i databasen, hvilket ikke er noget der fra standard er slået til, samt et modul der administrerer, hvilke permissions der ønskes af brugeren under log ind.

Vha. modulerne til at spørge brugeren om de behøvede permissions, og til at gemme oplysningerne, bliver det muligt at benytte endnu et medfølgende undermodul, der indeholder en funktion, der giver brugeren mulighed for, at publicere indhold fra ens side til vedkommendes brugerprofil på Facebook.

I nuværende form tilskyndes en brugerinitieret publicering, altså at modulerne giver mulighed for at brugere selv kan publicere, det er imidlertid ikke muligt som applikation at handle på vegne af brugeren. B14 ApS ønsker at tilbyde et produkt til Drupal, hvori det er muligt at lægge en besked på Facebook til alle sine, på siden registrerede, brugere. Til dette hører naturligvis også en løsning til at administrere udsendte beskeder internt.

6.2 Funktionelle krav

Alle funktionelle krav beskrives ved brug af use cases (brugerscenarier)[14]. Alle use cases er i analysens natur af black box typen, dvs. at de kun betragter, hvad der skal foregå (input og output), og ikke *hvordan* det skal foregå, hvilket i stedet beskrives i designafsnittet. Brugen af use cases er derfor en analysemetode til at identificere hvilken funktionalitet der skal indbygges i programmet, samt hvilke aktører systemet arbejder med.

Figur 6: Blackbox metoden hvor den sorte boks illustrerer systemprocesser skjult for aktørerne

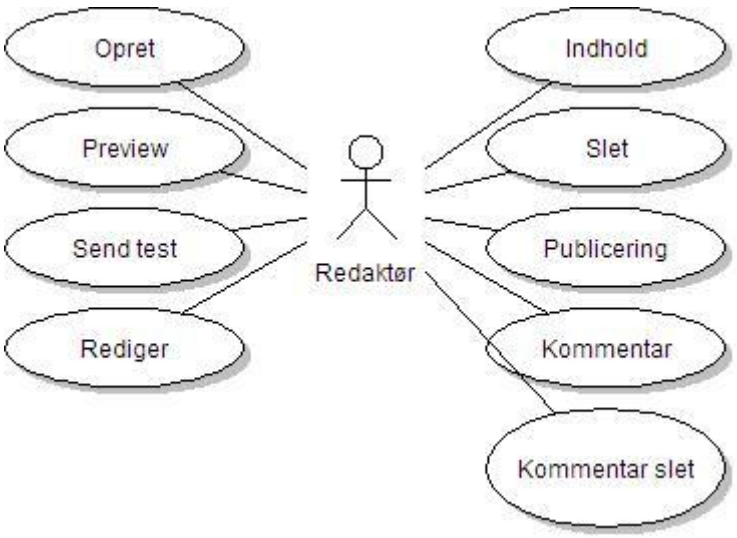
[14] Litteratur brugt hertil: http://www.gatherspace.com/static/use_case_example.html

6.3 Use cases

Der er identificeret 9 use cases baseret på problemområdet og mulighederne stillet til rådighed af Facebook i skrivende stund.

Navn	Beskrivelse
Opret	For at kunne publicere en besked, skal der først være mulighed for at oprette den.
Preview	Det skal være muligt internt, at se en forhåndsvisning af beskeden man har oprettet.
Send test	Det skal være muligt, at sende en test besked til en enkelt profil.
Rediger	Beskeden skal kunne redigeres inden udsendelse efter, den er oprettet.
Indhold	Under oprettelse skal alle felter stillet til rådighed af Facebook, være mulige at oprette.
Slet	Før og efter publicering skal det være muligt at slette beskeden.
Publicering	Publicering af oprettet besked til alle registrerede brugere.
Kommentarer	Publicering af kommentarer til en allerede oprettet og publiceret besked.
Kommantar slet	Sletning af publicerede kommentarer.

I forbindelse med udarbejdelsen af de forskellige use cases blev en aktør identificeret: redaktøren. Dette er en Drupal bruger med redaktør rettigheder (content admin).



Figur 7: Use case diagram

6.3.1 Prioritering af use cases

Alle use cases i projektet er blevet prioriteret efter en subjektiv vurdering af deres vægt i hele projektets tyngde og i forhold til de opstillede krav, men også for at prioritere og dermed planlægge arbejdet under udviklings- og designfaserne.

Der benyttes en skala fra 1 til 3, hvor 1 er absolut nødvendig, 2 er vigtig og 3 er nyttig. Ligeledes er der foretaget en risikovurdering, hvor 1 betyder høj risiko, 2 mellem risiko og 3 lav risiko. Der følger en uddybende risikoanalyse senere i dette afsnit der analyserer hvilke risici, der er opdaget i forbindelse med use casene og mere generelt, risikoen i skemaet herunder er derimod defineret som sandsynligheden for fejl i forbindelse med implementeringen, en forklaring af dette følger i det uddybende afsnit for hver enkelt use case.

Use case ID	Navn	Prioritet	Risiko	Kommentar
UC1	Opret	1	2	Basal use case de andre er afhængige af.
UC2	Preview	2	3	Preview af en oprettet besked internet i systemet, inden udsendelse.
UC3	Send Test	1	2	Skal i princippet vise det samme som preview internt i systemet, visningen her vil dog være ekstern, denne er prioriteret højere end UC2 da denne preview-form er mere præcis.
UC4	Rediger	1	3	Internt skal beskeden kunne redigeres, dog kun inden udsendelse, dette er en standardfunktion i Drupal og burde derfor ikke indeholde større udfordringer.
UC5	Indhold	1	2	I dette indgår en undersøgelse af præcis hvilke felter Facebook stiller til rådighed, tilsvarende felter skal så oprettes i systemet.
UC6	Slet	2	1	Slet er en standardfunktion i Drupal og bør ikke give besvær, men slet af beskeder efter publicering kan vise sig tricky.
UC7	Publicering	1	1	En vigtig use case for systemet, den oprettede besked gemmes i systemet og udsendes efterfølgende til alle tilmeldte brugere, men samtidig risikofyldt pga. Facebook.
UC8	Kommentarer	3	2	En såkaldt nice-to-have funktion, det ideelle er naturligvis at udsende en besked med fyldestgørende information, men oprettelse af kommentarer kan være en naturlig udvidelse, der er dog risiko for store begrænsninger.
UC9	Kommentar slet	3	1	Ligesom UC6

6.4 Ikke-funktionelle krav

Ikke funktionelle krav, eller øvrige krav, er en række krav til projektet der ikke direkte vedrører selve funktionaliteten, men som skal være opfyldt for systemets virke, de ikke funktionelle krav dækker således det operationelle og strukturelle, og definerer dermed også kvaliteten af systemet. Set tilbage på black box modellen et par sider tilbage, dækker de ikke-funktionelle krav den kontekst, hvori modellen befinder sig og beskriver hvordan systemet skal *være*, mens de funktionelle krav beskriver, hvad systemet skal gøre.

Jeg har valgt at dele de ikke-funktionelle krav op i et par kategorier for overblikkets skyld, de første er basale systemkrav:

1. Drupal, systemet skal kunne afvikles igennem Drupal, som et modul der kan aktiveres i Drupals oversigt over installerede moduler.
2. MySQL, systemet skal benytte MySQL til database eftersom det er den, i virksomheden, benyttede databasestandard, principielt set bør systemet dog have mulighed for operation med andre standarder.
3. PHP, systemet skal være skrevet i PHP, hvilket naturligvis er en konsekvens af første punkt, om afvikling i Drupal miljøet.
4. Ikke platformspecifik, skal virke både på Windows såvel som Unix servere.
5. Fejlmeddelelser eller fejl i forbindelse med publicering bør håndteres af Drupals system til formålet.

Herudover er der et par krav om grænseflade:

1. Grænsefladen til systemet bør følge den generelle grænseflade og designretning brugt i Drupal, således at den virker som en naturlig del.
2. Systemet skal i brugervenlighed optimeres efter en bruger med et redaktørniveau i forståelse af den generelle brug af Drupal, altså optimeret efter en bruger med et generelt kendskab til Drupals brugerflade, der tages derfor ikke nødvendigvis hånd om uerfarne brugere til Drupal.
3. For at gøre indgangsbarrieren lavest mulig bruges generelle Drupal hjælpetekster til indtastningsfelter, dette er små linjer tekst under hvert felt, der beskriver hvordan feltet benyttes.

7. Analyse

7.1 Eksisterende muligheder

Som oplægget til problemstillingen i begyndelsen af dette projekt fremlægger, omfatter projektet ikke endnu eksisterende funktionalitet, det bør dog alligevel være relevant, at se på hvilke eksisterende muligheder og alternativer der findes på området. Der tænkes heri kun i Drupal regi.

7.1.1 Drupal Facebook moduler

I den modulpakke der udvikles til, findes faktisk i forvejen et modul som understøtter publicering til Facebook, dette hedder fb_stream. Modulet benytter pt. udelukkende Facebooks JavaScript API og giver mulighed for, at publicere til en brugers væg. Idet det er JavaScript kører det på klientens maskine og følger dermed en brugerinstantieret model, hvor en bruger aktivt foretager en handling for, at opdatere sin status. Denne giver altså ikke mulighed for udsendelse til en gruppe brugere, og kan ej heller udsendes uopfordret dvs. ikke på opfordring af en netop aktiv logget ind bruger.

Noget af det samme, men med en lidt anden indgangsvinkel, findes i et selvstændigt modul ved navn Post to Facebook[15], dette er udvikling af samme person som Drupal for Facebook, men valgt udgivet selvstændigt. Post to Facebook giver en redaktør mulighed for, i forbindelse med oprettelse af en ny node, også at udsende denne på sin egens eller en sides væg, alt dette bygget ind i et overskueligt interface integreret med Drupal, hvor førnævnte blot er et API, der kan bygges ind ved behov. En af de smarte ting ved denne løsning er, at den også giver mulighed for at hente kommentarer til den besked man har udsendt inde i Drupal, og på den måde, at synkronisere mellem kommentarer oprettet i Drupal og på Facebook.

En tredje eksisterende mulighed er modulet Simple Facebook Wall Post[16], der giver udviklere mulighed for at tilføje en knap til bestemte indholdstyper der åbner en dialogboks, hvor brugeren får mulighed for, at poste noget indhold på sin væg. Funktionaliteten er nogenlunde svarende til den almindelige like-knap[17], men mere integreret i Drupal og med bedre kontrol over, hvad præcis der bliver sendt til ens væg.

7.1.2 Simplenews

De tre førnævnte moduler indeholder alle noget af den funktionalitet, som kan bruges i dette projekt, ingen af dem opretter dog en specifik indholdstype beregnet til Facebook eller giver mulighed for andet end publicering til en netop aktiv brugers væg eller en side. Simplenews[18] modulet har umiddelbart intet med Facebook at gøre, idet det bruges til at oprette og udsende nyhedsbreve i Drupal i forskellige abonnementslister. Det der gør det interessant at se på er, hvordan det er integreret med Drupal i forbindelse af udsendelse af store mængder e-mails herunder, hvordan systemet klargør mængden af mails der skal udsendes, og hvordan en potentielt længere tidskrævende proces kører.

[15] http://drupal.org/project/fridge
[16] http://drupal.org/project/simple_fbwall
[17] http://developers.facebook.com/docs/reference/plugins/like/
[18] http://drupal.org/project/simplenews

7.2 Risikoanalyse

I ethvert projekt vil der altid være større eller mindre risici forbundet med dets forløb, disse risici kan inddeles i en lang række kategorier. Som det kunne ses i den uddybende use case tabel, var risici inddelt på en karakterskala fra 1 til 3, med 1 som størst risiko og 3 som mindst. karakteren gives ud fra en vurdering efter en generel formel, i denne risikoanalyse er brugt modellen OWASP[19] (Open Web Application Security Project):

$$Risiko = Sandsynligheden\ for\ en\ hændelse\ x\ omkostninger\ ved\ hændelsen$$

I OWASP modellen inddeles analysen i fem punkter med udgangspunkt i den første formel:

1. Identificer risici
2. Estimer sandsynlighed
3. Estimer omkostninger
4. Determiner alvorlighed
5. Opret prioritetsliste

7.2.1 Identifikation af risici

I denne analyse identificeres og analyseres de størst vurderede risici ud fra denne liste, og sammenlægges med use case skemaet fra tidligere. Risici er alle identificeret ud fra en række typiske risikoområder, dette kan være:

- Menneskelige risici, personer bliver syge, dødsfald.
- Operationelle, tab af adgang til aktiver, manglende adgang til ressourcer.
- Omdømme, tab af tiltro fra partnere eller marked.
- Processuelle, systemnedbrud, organisatoriske fejl, bedrageri.
- Projektmæssige, overskridelse af tidsplan, budget, deadline eller manglende produktkvalitet.
- Finansielle, konkurs, aktie værditab, renteændringer etc.
- Tekniske, forældelse af software eller hardware systemer, tekniske nedbrud.
- Naturlige, naturkatastrofer eller uheld.
- Politiske, lovændringer, skatteregler, offentlig indflydelse, oversøisk indflydelse.

Dette er et udvalg af kategorier risici kan inddeles efter, der findes højst sandsynligt flere, nogle af dem er ikke blevet brugt i dette projekt, eftersom de er irrelevante, dette er eksempelvis finansielle og naturlige, da projektet er ikke afhængigt af finansielle forhold, når alle ressourcerne er stillet gratis til rådighed.

[19] https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

Risiko	Sandsynlighed	Omkostning	Kommentar
Sygdom	Mellem	Mellem	Er vurderet som mellem fordi det kan ske både i kortere og længere tid, omkostningen kan være skred i tidsplan.
Uforudset arbejde	Høj	Lav	I projektforløbet kan der havne uforudset meget arbejde, omkostningen kan være et skred i tidsplan.
Skred i tidsplan	Høj	Lav	Forskellige uforudsete faktorer kan medføre skred, omkostningen kan være mindre tid til dokumentation og implementering.
Internetnedbrud	Lav	Mellem	Netværksnedbrud forekommer og vil betyde at udvikling stopper da projektet kører på en server på internettet.
Systemnedbrud	Lav	Høj	Systemnedbrud forekommer sjældent, men kan stoppe al udvikling og dokumentation samt resultere i skred i tidsplan eller tab af data eller dokumentation.
Facebook opdateringer	Høj	Høj	Facebook opdaterer ofte sit API, som projektet er meget afhængig af, en opdatering kan betyde at eksisterende kode holder op med at virke og skal skrives om, en anden alvorlig omkostning kan være en begrænsning af muligheder, så udsendelse af beskeder umuliggøres.
Drupal opdateringer	Mellem	Mellem	Drupal opdateringer forekommer jævnligt, disse kan undlades installeret, men omkostningen kan være at eksisterende kode holder op med at virke og skal skrives om.
Ukendte begrænsninger	Mellem	Høj	Facebook kan have begrænsninger i deres API som indvirker på projektet, det kan eksempelvis være ukendte begrænsninger i antallet af lovlige API kald, omkostningen kan være at udsendelse af beskeder besværliggøres og kode skal skrives om.

Karakterer:

1 = Lav

2 = Mellem

3 = Høj

7.2.2 Estimering af sandsynlighed

Sandsynlighed og omkostninger er vurderet ud fra en række kriterier. Sandsynligheden er inddelt i to grupper af faktorer der kan hjælpe til at bestemme, hvor stor sandsynligheden er, disse er trusler og sårbarheder. Trussel faktorer er:

- Evner, hvor store er truslens evner til at udnytte risikoen?
- Motiv, hvor motiveret er truslen til at udnytte risikoen?
- Mulighed, hvor mange ressourcer kræves der for at udnytte risikoen?
- Størrelse, hvor stor er gruppen af mennesker der potentielt kunne udnytte risikoen?

Sårbarhedsfaktorer er:

- Opdagelse, hvor nemt er det for truslen at opdage der er en sårbarhed?
- Udnyttelse, hvor nemt er det at udnytte sårbarheden?
- Klarhed, hvor åbenlys er sårbarheden?
- Detektion, hvor stor er sandsynligheden for at en uønsket indtrænger opdages?

7.2.3 Estimering af omkostning

Ligesom med sandsynlighed kan omkostninger inddeles i to overordnede kategorier, formålet her er at estimere omkostningerne ved en given hændelse. Kategorierne er tekniske omkostninger, altså omkostningerne for systemet. Den anden er de forretningsmæssige omkostninger. Tekniske omkostningsfaktorer er:

- Tab af fortrolighed, hvor meget data kan oplyses og hvor følsomt er det?
- Tab af integritet, hvor meget data kan blive ødelagt og hvor ødelagt er det?
- Tab af tilgængelighed, hvor mange tjenester kan gå tabt og hvor vigtige er de?
- Tab af ansvar, kan den/de ansvarlige findes eller spores og hvor nemt?

Forretningsmæssige omkostningsfaktorer er:

- Finansiel skade, hvor meget finansiel skade vil en udnyttelse udgøre?
- Skade på omdømme, vil en udnyttelse skade omdømmet?
- Manglende overholdelse (non-compliance), hvor stor eksponering medfører manglende overholdelse af standarder og normer?
- Persondataovertrædelse, hvor mange personlige oplysninger kunne tilgængeliggøres?

De forretningsmæssige omkostninger vil, i et projekt som dette uden forretningsmæssig kontekst, være af mindre betydning, men for skønnets ægtheds skyld tænkes analysen også ind i en forretningsmæssig kontekst i det følgende afsnit.

7.2.4 Determination af alvorlighed

OWASP modellen inddeler sandsynligheder og omkostninger på en karakterskala fra 1 til 9:

Sandsynligheds- og omkostningsniveauer	
0 til <3	Lav
3 til <6	Mellem
6 til 9	Høj

Den samlede karakter for sandsynlighed og omkostning beregnes ud fra et gennemsnit af alle de individuelt vurderede faktorer. Herunder ses risikoanalysen for en af de angive risici, resten kan findes i bilaget Risikoanalyse, hver karakter er angivet ud fra en subjektiv vurdering af niveau:

Risiko: Systemnedbrud				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
2	4	2	3	2,75
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennemsnit
2	2	2	3	2,25
			Total gennemsnit:	2,5
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
6	7	7	7	6,75
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
6	3	4	8	5,25
			Total gennemsnit:	6
			Risiko:	4,25

Den samlede estimerede risiko er udregnet som et gennemsnit af sandsynligheder og omkostninger og vurderes umiddelbart ud fra denne model:

Risiko alvorlighed				
Omkostninger	Høj	Mellem	Høj	Kritisk
	Mellem	Lav	Mellem	Høj
	Lav	Anføres	Lav	Mellem
		Lav	Mellem	Høj
	Sandsynlighed			

I dette eksempel vil sandsynligheden for systemnedbrud være lav, mens omkostningerne vil være mellem, det giver en overordnet vurdering af risikoen som lav. I beskrivelsen af modellen blev formuleringen umiddelbar vurdering brugt, hvilket skyldes, at der i stort set enhver risikoanalyse, vil være vægtninger, der skal tages højde for. I en virksomhed vil de økonomiske omkostninger sandsynligvis altid, være vægtet højere end de tekniske omkostninger, så selvom modellen måske vurderer risikoen høj som følge af høje tekniske omkostninger, vil risikoens prioritering muligvis blive dømt lav alligevel pga. lave økonomiske omkostninger, der vægtes højere i analysen. I eksemplet og i udregningerne i dette projekt er begge omkostninger vægtet lige, selvom der reelt set ikke vil være økonomiske omkostninger ved nogle af risiciene.

Det sidste punkt er, at oprette en prioritetsliste med de alvorligste risici vægtet højest, og de mindst alvorlige lavest eller måske endda ignoreret. Prioriteringen er altid en vurdering af hvilke risici det mindst kan betale sig at ignorere, også her er det altså en subjektiv vurdering der afhænger af de omkringliggende omstændigheder.

Konklusionen på dette afsnit er hermed en prioriteret risikoliste:

Navn	Gennemsnit	Risiko
Facebook opdateringer	6,31	Kritisk
Ukendte begrænsninger	5,31	Høj
Uforudset arbejde	4,37	Mellem
Systemnedbrud	4,25	Mellem
Drupal opdateringer	3,81	Mellem
Sygdom	3,37	Mellem
Skred i tidsplan	2,93	Lav
Internetnedbrud	2,56	Lav

Modellen giver naturligvis mulighed for tilpasning til det enkelte projekt, og alt efter omstændigheder og krav vil en tilpasning næsten altid være den endelige løsningsmodel. Tilpasninger kan være ændrede sandsynlighedsfaktorer, omkostningsfaktorer og forskellig vægtning af faktorer. En anden idé er at lade forskellige teams udføre analysen, så risici bliver vurderet fra forskellige vinkler og ukendte faktorer måske lettere spottes, projektets begrænsede omkostninger og størrelse taget i betragtning, er det dog her valgt at bruge en standardmodel med ens vægtning.

7.3 Funktionelle krav (Use cases)

I kravspecifikationen blev en række use cases kort beskrevet, i dette afsnit følger en fuldstændig beskrivelse af hver enkelt use case. Der kan forekomme dele i hver use case, som burde høre designafsnittet til, men i alle tilfælde er de medtagne dele blevet vurderet vigtige for det samlede billede, forståelsen af use casens funktion og berettigelse, dette kan også gavne senere i udviklingsforløbet, ved fra starten at definere kravene tydeligere. Som det også fremgår af kravspecifikationen er der kun identificeret en enkelt aktør, denne fremgår derfor som den eneste i alle use cases.

For ordens skyld gennemgås herunder de enkelte dele af en use case:

Use case ID	Navn
Aktør	Aktøren som opererer i denne use case
Prioritet	Hvor højt implementeringen af denne use case er vægtet på en skala fra 1 til 3
Risiko	Risikoen for komplikationer i forbindelse med implementeringen af denne use case vægtet på en skala fra 1 til 3
Beskrivelse	Beskrivelse af use casen
Præmis	Et forhold der skal være opfyldt for at use casen kan udføres
Udløser (trigger)	En handling der igangsætter use casen
Tilstand ved succes	Hvad der sker efter en succesfuld udførelse
Tilstand ved fejl	Hvad der sker efter en fejlslagen udførelse
Primær forløb	Angiver hvilke punkter der gennemgås igennem use casens forløb.
Udvidelser	Foreslåede udvidelser til at udvide use casens handleområde, ekstra funktionalitet.

På næste side følger en udvidet beskrivelse af hver use case fra kravspecifikationen.

7.3.1 UC1: Opret

Use case ID: UC1	Navn: Opret
Aktør	Redaktør
Prioritet	1 (Nødvendig)
Risiko	2 (Middel)
Beskrivelse	Basal use case de andre er afhængige af, dækker oprettelsen af en artikeltype (node) i Drupal, som indeholder de felter der bruges til publicering og understøttes af Facebooks API. Det vil altså sige at under oprettelsen af en ny Facebook post indholdstype vil blive præsenteret for indtastningsfelter som benyttes og understøttes i forbindelse med publicering til Facebook.
Præmis	Indholdstypen hørende til Facebook post skal være oprettet i Drupal
Udløser (trigger)	Brugeren opretter en ny node af den tilhørende indholdstype i Drupal
Tilstand ved success	En ny node er oprettet i Drupal
Tilstand ved fejl	Fejlbesked præsenteres til brugeren
Primær forløb	1. Oprettelse. 2. udfyldning af felter. 3. Gem.
Udvidelser	Ingen

7.3.2 UC2: Preview

Use case ID: UC2	Navn: Preview
Aktør	Redaktør
Prioritet	2 (Vigtig)
Risiko	3 (Lav)
Beskrivelse	Preview af en oprettet besked internt i systemet, inden udsendelse. Visningen baseres på de udfyldte felter fra oprettelsen. Dette giver brugeren mulighed for at få indholdet præsenteret på en realistisk måde uden at der skal kommunikeres med Facebook på nogen måde.
Præmis	Under oprettelse skal body-feltet som minimum være udfyldt
Udløser (trigger)	Brugeren trykker på navigationspunkt til Preview i menu
Tilstand ved success	Preview vises
Tilstand ved fejl	Preview vises, men visning kan være mangelfuld, hvor nogle felter er tomme.
Primær forløb	1. Navigationspunkt trykkes 2. Browseren navigerer til preview siden 3. Preview genereres ud fra gemte oplysninger i Drupal 4. Brugeren præsenteres for preview
Udvidelser	1. Ændret preview baseret på hvilke felter der er udfyldt. 2. Visning af kommentarer i preview.

7.3.3 UC3: Send test

Use case ID: UC3	Navn: Send test
Aktør	Redaktør
Prioritet	1 (Nødvendig)
Risiko	2 (Middel)
Beskrivelse	Skal i princippet vise det samme som preview internt i systemet, visningen her vil dog være ekstern, denne er prioriteret højere end UC2 da denne preview-form er mere præcis. Ved udsendelse af en test angiver brugeren et gyldigt Facebook bruger id, som har godkendt den applikation der bruges i forbindelse med publicering, ved denne type laves et faktisk API kald til Facebook, der bliver dog kun sendt én besked til det angivne id.
Præmis	1. under oprettelse skal body-feltet som minimum være udfyldt. 2. Et Facebook bruger id der på forhånd har godkendt applikationen der benyttes, kræves.
Udløser (trigger)	Brugeren trykker på send knap under navigationspunkt Send
Tilstand ved success	Et API kald foretages til Facebook og beskeden publiceres på væggen tilhørende det indtastede bruger id.
Tilstand ved fejl	Fejlbesked præsenteres for brugeren.
Primær forløb	1. Brugeren indtaster et gyldigt Facebook bruger id 2. Brugeren trykker på Send test knap 3. Et API kald foretages og beskeden forsøges publiceret på Facebook. 4. Brugeren præsenteres for success eller fejlbesked.
Udvidelser	1. Funktion til at hente bruger id automatisk. 2. Funktion til at udsende test af kommentar-besked

7.3.4 UC4: Rediger

Use case ID: UC4	Navn: Rediger
Aktør	Redaktør
Prioritet	1 (Nødvendig)
Risiko	3 (Lav)
Beskrivelse	Internt skal beskeden kunne redigeres, dog kun inden udsendelse, dette er en standardfunktion i Drupal og burde derfor ikke indeholde større udfordringer. I kraft af Drupals status som CMS giver systemet automatisk mulighed for at redigere alle indholdstyper, denne use case vil derfor være understøttet af systemet og skal ikke implementeres.
Præmis	Beskeden skal være oprettet.
Udløser (trigger)	Brugeren trykker på navigationspunkt Rediger (Edit)
Tilstand ved success	Brugeren præsenteres for artikeltypen i redigerbar form
Tilstand ved fejl	Brugeren præsenteres for en fejlbesked (Manglende rettigheder til at redigere)
Primær forløb	1. Brugeren trykker på navigationspunkt til redigering 2. Brugere præsenteres for artikeltypen i redigerbar form 3. Brugeren redigerer og gemmer. 4. Drupal opdaterer den oprindelige artikel.
Udvidelser	Ingen

7.3.5 UC5: Indhold

Use case ID: UC5	Navn: Indhold
Aktør	Redaktør
Prioritet	1 (Nødvendig)
Risiko	2 (Middel)
Beskrivelse	I dette indgår en undersøgelse af præcis hvilke felter facebook stiller til rådighed, tilsvarende felter skal så oprettes i systemet, denne use case vedrører kun installationen. Når modulet skal installeres skal der samtidig oprettes en indholdstype i Drupal der kan benyttes til publicering til Facebook, herved gøres installationen væsentlig lettere end hvis brugeren skulle oprette den påkrævede indholdstype manuelt.
Præmis	For at installere/importere den korrekte indholdstype skal et modul ved navn content_copy være aktiveret på forhånd.
Udløser (trigger)	Modulet aktiveres under modules i sideadministrationen
Tilstand ved success	Modulet aktiveres
Tilstand ved fejl	Modulet aktiveres ikke, brugeren præsenteres for en fejlbesked
Primær forløb	1. Brugeren navigerer til moduler i sideadministrationen 2. modulet vælges og konfigurationen gemmes 3. Systemet kontrollerer om nødvendige tredjepartsmoduler er aktiverede. 4. Modulet aktiveres og indholdstypen importeres.
Udvidelser	1. Bedre dokumentation i forbindelse med installation.

7.3.6 UC6: Slet

Use case ID: UC6	Navn: Slet
Aktør	Redaktør
Prioritet	2 (Vigtig)
Risiko	1 (Høj)
Beskrivelse	Slet er en standardfunktion i Drupal og bør ikke give besvær, men slet af beskeder efter publicering kan vise sig tricky, der skal laves et API kald for hver publicerede besked om sletning dvs. et kald per registreret bruger.
Præmis	For sletning af beskeder på Facebook skal de være publiceret forinden og ikke allerede slettet.
Udløser (trigger)	Brugeren trykker på slet-knap under navigationspunktet Delete
Tilstand ved success	Alle publicerede beskeder slettes fra Facebook
Tilstand ved fejl	Fejlbeskeder logges og vises i Drupals hændelseslog eller brugeren præsenteres for en fejlbesked
Primær forløb	1. brugeren navigerer til Slet 2. Brugeren trykker på slet knap 3. Browseren navigerer til side hvor sletning af hver besked gennemgås 4. Systemet foretager et API kald til Facebook om sletning for hver enkelt publicerede besked 5. API svar logges (succes/fejl) 6. Browseren navigerer til Drupals hændelseslog ved afslutning.
Udvidelser	Ingen

7.3.7 UC7: Publicering

Use case ID: UC7	Navn: Publicering
Aktør	Redaktør
Prioritet	1 (Nødvendig)
Risiko	1 (Høj)
Beskrivelse	Den oprettede besked eller kommentar gemmes i systemet og udsendes efterfølgende til alle tilmeldte facebook brugere. Konkret betyder det at der laves et API kald for hver registreret bruger.
Præmis	Ingen, men det anbefales kraftigt at eftertjekke ved brug af Preview og Send test funktionerne inden faktisk publicering.
Udløser (trigger)	Brugeren trykker på send-knap under navigationspunktet Send
Tilstand ved success	Beskeden publiceres til alle registrerede brugere på Facebook
Tilstand ved fejl	Fejlbeskeder logges og vises i Drupals hændelseslog eller brugeren præsenteres for en fejlbesked
Primær forløb	1. brugeren navigerer til Send 2. Brugeren trykker på send knap 3. Browseren navigerer til side hvor publicering af hver besked gennemgås 4. Systemet foretager et API kald til Facebook om publicering for hver enkelt bruger 1. 5. API svar logges (succes/fejl) 6. Browseren navigerer til Drupals hændelseslog ved afslutning.
Udvidelser	Ingen

7.3.8 UC8: Kommentar

Use case ID: UC8	Navn: Kommentar
Aktør	Redaktør
Prioritet	3 (Valgfri)
Risiko	2 (Middel)
Beskrivelse	Oprettelse af kommentarer til en allerede oprettet besked. Dette er en standard valgfri funktion i Drupal og skal aktiveres for indholdstypen under installationsfasen.
Præmis	Før en kommentar kan publiceres til Facebook skal den oprindelige besked allerede være publiceret.
Udløser (trigger)	Brugeren gemmer en ny kommentar
Tilstand ved success	Kommentaren gemmes i systemet
Tilstand ved fejl	Brugeren præsenteres for en fejlbesked
Primær forløb	1. Brugeren trykker på "kommenter"-link 2. Browseren navigerer til oprettelsesside 3. Brugeren indtaster kommentar og gemmer 4. Kommentaren gemmes i systemet
Udvidelser	Ingen

7.3.9 UC9: Kommentar slet

Use case ID: UC9	Navn: Kommentar slet
Aktør	Redaktør
Prioritet	3 (Valgfri)
Risiko	1 (Høj)
Beskrivelse	Fuldstændig som UC6, eneste forskel er at kommentarer slettes af brugeren ved at trykke på slet-link under kommentaren.
Præmis	Ingen
Udløser (trigger)	Brugeren trykker på slet link under kommentar
Tilstand ved success	Kommentaren slettes
Tilstand ved fejl	Fejlbesked præsenteres for brugeren
Primær forløb	1. bruger trykker på slet link 2. browser navigerer til bekræftelsesside 3. brugeren bekræfter sletning 4. kommentaren slettes fra systemet
Udvidelser	1. Individuel sletning af publicerede kommentarer, denne use case dækker kun sletning af alle publicerede kommentarer.

7.4 Målgruppe

Produktets målgruppe vil være brugere der i forvejen benytter det eksisterende Drupal modul Drupal for Facebook, eftersom projektet her udvider funktionaliteten af og bygger på det eksisterende modul. Ligeledes om målgruppen kan man sige, at det vil være brugere der ønsker, at kunne udsende beskeder til alle deres brugere igennem Drupal og som i forvejen benytter en Facebook applikation med Facebook Connect i en eller anden form. Det vil i flestes tilfælde være aktøren, identificeret i use case analysen, der er brugeren, dermed målrettes der ikke efter den gængse bruger af Facebook, men til en mere teknisk kvalificeret bruger med generelt kendskab til Drupal.

Der vil ikke blive udført en mere dybdegående markedsanalyse, fordi projektets produkt lægges frit tilgængeligt som open source, og i alle tilfælde vil være en del af et slutprodukt, der er integreret med Facebook Connect, en målgruppeanalyse er derfor kun interessant for et slutprodukt. Det skal understreges at der i forbindelse udarbejdelsen af idéen blev foretaget en form for mindre markedsanalyse idet en undersøgelse af eksisterende muligheder blev foretaget, det lykkedes ikke at finde eksisterende funktionalitet til Drupal platformen, og gav derfor dette projekt grundlag.

8. Design

8.1 Grænseflade

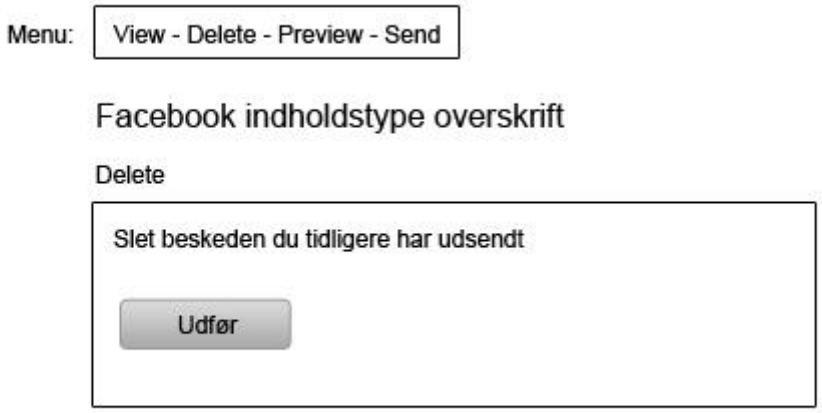
Grænsefladen til modulet skal være forholdsvis enkel og stemme overens med de i kravspecifikationen beskrevne krav, det vil eksempelvis betyde ingen genopfindelser af dybe tallerkener, men udelukkende benyttelse af de standardværktøjer Drupal tilbyder til opsætning af menuer, formularer osv.

Hvis vi forestiller os en indholdstype i Drupal der indeholder de felter en Facebook status opdatering består af, vil der ved en node af denne indholdstype blive vist en særlig menu med mulighederne fra de gældende use cases, det kunne se således ud:

Figur 8: Nodens visning i Drupal.

Designet vil variere meget alt efter hvilket Drupal tema der er brugt, grundstrukturen i grænsefladen som her, vil opfylde kravspecifikationen. Billedet viser visningen under menupunktet der hedder ”View”.

Under menupunktet ”Delete” vil der være en knap til at slette den besked man allerede har udsendt, brugeren skal naturligvis også kende en form for status for beskeden, så vedkommende ved om beskeden rent faktisk kan slettes.



Figur 9: Grænseflade for Delete.

Preview menupunktet vil vise en forhåndsvisning af hvordan beskeden ser ud på Facebook:

Figur 10: Visning af preview.

Den vigtigste er grænsefladen der ligger oven på send funktionerne og giver mulighed for at sende en test, sende reelt eller udsende en kommentar til en allerede udsendt besked, denne grænseflade kunne se således ud under menupunktet ”Send”:

Figur 11: Grænseflade til send

En status bør naturligvis også fremgå så brugeren kan se om beskeden tidligere har været udsendt eller om den er udsendt og senere slettet.

8.2 Funktionalitet

Designet af systemets funktionalitet er naturligvis vigtigt for den endelige implementering. Systemets grundfunktionalitet vil i dette afsnit blive beskrevet med de påkrævede elementer, såsom opbygningen af en Facebook besked og interaktionen mellem systemet og Facebooks API. Her følgende afsnit gennemgår hvordan en besked til Facebook er opbygget, hvordan modulet til Drupal opbygges, hvilke funktioner der skal bygges ind i modulet og en visning af hvordan systemets flow bør være.

8.2.1 En Facebook besked

For at indbygge beskedtypen Facebook benytter i systemet kræver det, at designet afspejler den måde en besked er opbygget på. I figuren herunder ses, hvilke dele en besked består af:

Figur 12: En Facebook beskeds opbygning

Beskeden består af både obligatoriske og valgfrie felter, disse er:

- En hovedbrødtekst, som er obligatorisk hvis ikke et link er vedhæftet.
- En link URL, som er obligatorisk hvis ikke hovedbrødtekst (message) feltet er udfyldt.

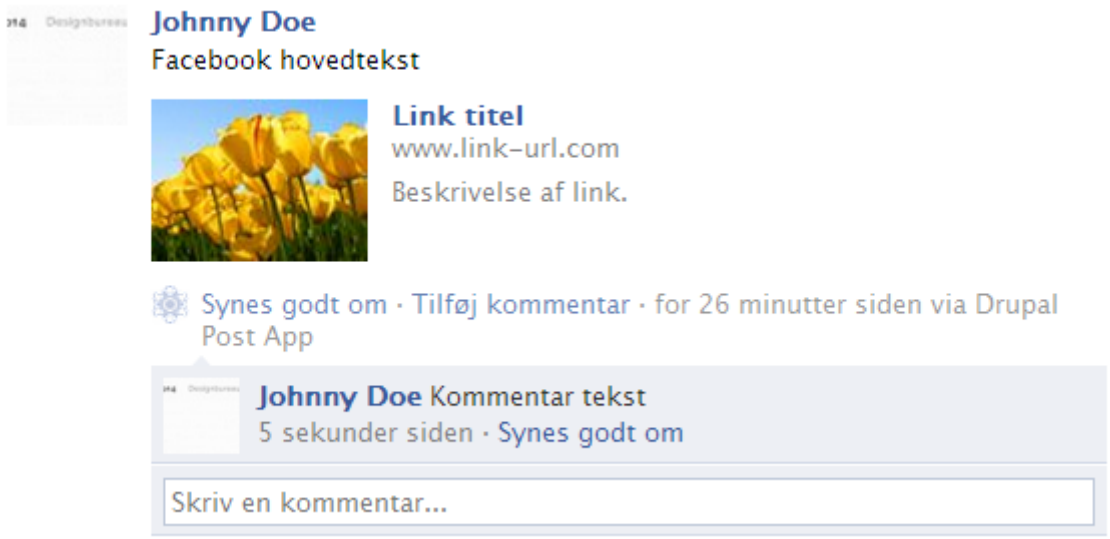
Derudover består et link af nogle felter:

- Link URL, som er web-adressen.
- Link title, som er en valgfri titel til linket.
- Link beskrivelse, som er en valgfri beskrivelse af siden der linkes til.
- Link caption, som er linkets navn i stedet for ”http://www.link-url.com/” kunne man sætte visningen til ”link-url.com” for en mere letlæselig udgave.
- Et billede, som hører til linket.

Derudover ses det at applikationens navn også fremgår af beskeden, det er så brugeren kan se at beskeden er udsendt via en applikation. Til venstre for teksten ”Synes godt om”, ses også et lille ikon, dette er

applikationens ikon, som også kan sættes til at være et specifikt ikon, i eksemplet er brugt Facebooks standard ikon for applikationer.

En kommentar er lidt simplere idet den kun består af et enkelt felt, nemlig selve beskedteksten, som ses på denne figur:



Figur 13: Eksempel på Facebook kommentar

Bemærk at det ikke fremgår af kommentaren, at den er sendt af applikationen, brugeren og især vedkommendes venner har ikke mulighed for at vide, at det ikke er personen selv der har siddet bag tasterne og indsendt kommentaren.

Denne besked har naturligvis også en repræsentation i Facebooks Graph API, som indeholder en del skjulte oplysninger der ikke fremgår af figurerne herover, resultatet forneden er hentet ved at gå til denne adresse: https://graph.facebook.com/me/feed?access_token=indsæt-adgangsnøgle-her, bemærk at der kræves en gyldig adgangsnøgle for at tilgå disse oplysninger, her første del som er selve beskeden:


```
{
   "data": [
      {
         "id": "100002503652851_208540022572785",
         "from": {
            "name": "Johnny Doe",
            "id": "100002503652851"
         },
         "message": "Facebook hovedtekst",
         "picture": "tulips.jpg",
         "link": "http://www.link-url.com/",
         "name": "Link titel",
         "caption": "www.link-url.com",
         "description": "Beskrivelse af link.",
         "icon": "http://www.facebook.com/images/icons/default_app_icon.gif",
```

Der ses to forskellige id numre i starten af resultatet, det første id er beskedens id som identificerer den specifikke besked, nummeret har formen ”profil-id_besked-id”, det næste id er således profilens id, som beskeden er opslået på, dette andet id er en bestanddel af selve beskedens id, som det ses på dens form. Derudover indgår profilens navn ”Johnny Doe” og de tidligere beskrevne besked bestanddele.

Næste del er actions delen, der er en nyskabelse på Facebook som bliver rullet ud første kvartal 2012[20], dette er ikke noget der påtænkes bygget ind i designet, men for god ordens skyld vises, hvilke standard actions der følger med en besked:

```
         "actions": [
            {
               "name": "Comment",
               "link":
"http://www.facebook.com/100002503652851/posts/208540022572785"
            },
            {
               "name": "Like",
               "link":
"http://www.facebook.com/100002503652851/posts/208540022572785"
            }
         ],
```

Heraf ses det at et link til at synes godt om (Like) og et link til at kommentere beskeden (Comment), er de to standard actions der oprettes for en besked. For fremtiden bliver det muligt at oprette egne actions, se fodnoten for eksempler.

Beskeder indeholder også oplysninger om af hvem beskeden kan ses, her kan man angive om beskeden skal kunne ses af alle, kun venner, kun bestemte grupper eller noget helt tredje, selvom disse indstillinger angives vil de altid blive trumfet af den enkelte brugers indstillinger for vedkommendes væg. Hvis ens besked angives til at være offentlig og beskeden publiceres på en profil som har angivet sin væg til kun at blive set af venner, vil den mest indskrænkende indstilling være den gældende.

```
         "privacy": {
            "description": "Public",
            "value": "EVERYONE",
            "allow": "0",
            "deny": "0"
         },
```

Beskedens type fremgår også:

```
         "type": "link",
```

Dette kan være ”status”, ”photo”, ”link” eller noget tredje og generes automatisk.

[20] http://venturebeat.com/2012/01/05/facebook-actions/

Hvis beskeden er publiceret igennem en applikation fremgår dette og efterfølgende tidspunkter for hvornår beskeden er publiceret og evt. opdateret, læg mærke til at applikationen også har sit eget unikke id:

```
"application": {
   "name": "Drupal Post App",
   "id": "285037634867184"
},
"created_time": "2012-01-11T12:06:07+0000",
"updated_time": "2012-01-11T12:32:54+0000",
```

Hvis en eller flere personer synes godt om en den besked der er publiceret fremgår det i likes:

```
"likes": {
   "data": [
      {
         "name": "Johnny Doe",
         "id": "100002503652851"
      }
   ],
   "count": 1
},
```

Count er blot en tæller der angiver antallet af likes. Dette er alle felterne der indgår i en typisk besked, sidst kommer eventuelle kommentarer:

```
"comments": {
   "data": [
      {
         "id": "100002503652851_208540022572785_1065557",
         "from": {
            "name": "Johnny Doe",
            "id": "100002503652851"
         },
         "message": "Kommentar tekst",
         "created_time": "2012-01-11T12:32:54+0000",
         "likes": 1
      }
   ],
   "count": 1
}
},
```

Læg især mærke til at kommentaren også har påhæftet et unikt id til at identificere den specifikke kommentar på formen ”profil-id_besked-id_kommentar-id”. Count er her en tæller der tæller antallet af kommentarer.

Dette er den komplette typiske opbygning af en besked i Facebooks Graph API, det ses af syntaksen at data i APIet består af en lang række indlejrede array (nested arrays). Enhver implementering må nødvendigvis tage højde for denne måde at angive data på.

8.3 Modul funktioner

Til bestemmelse af hvilke essentielle funktioner modulet bør indeholde, kastes der naturligvis et blik på use case beskrivelserne fra analyseafsnittet, heri vil nogle være understøttet af Drupal og størstedelen kræve ekstra, ikke i forvejen eksisterende, funktionalitet, dette følger herunder:

- **UC1**, Opret – Håndteres af Drupal core, det er kun indholdstypen der skal oprettes.
- **UC2**, Preview – Her skal laves en funktion der returnerer et preview af indtastede felter.
- **UC3**, Send Test – Her behøves en række basale funktioner der har med publicering at gøre.
- **UC4**, Rediger – Håndteres udelukkende af Drupal.
- **UC5**, Indhold – Som nævnt under UC1 skal indholdstypen oprettes af modulet, da den ikke vil være eksisterende i forvejen.
- **UC6**, Slet – Slet af selve beskeden i Drupal håndteres af Drupal core, men der behøves en funktion der også udløser en sletning af indhold i den korrekte databasetabel når den slettes fra Drupal. Derudover behøves der selvfølgelig funktioner til sletning af publicerede beskeder fra Facebook.
- **UC7**, Publicering – Vil være en udvidelse til UC3 hvor en række funktioner henter alle brugere og sender til hver enkelt.
- **UC8**, Kommentar – Håndteres også af Drupal core, men der skal laves en speciel funktion der føder UC7s funktioner med kommentarer i stedet for normale beskeder.
- **UC9**, Kommentar slet – Ligesom UC6 hvor det håndteres af Drupal core, der kræves stadig en udløser til at rydde op i den database der bruges.

For at komme nærmere ind på præcis, hvordan disse krav opfyldes, beskrives de funktioner der skal implementeres for at opnå den ønskede funktionalitet, der benyttes så generelle beskrivelser som muligt.

8.3.1 get_facebook

For overhovedet at kommunikere med Facebook, kræver det, at deres API initialiseres først således, at en gyldig session startes. Der behøves derfor en funktion som kan returnere en instans, som er en gyldig session. For at kunne initialisere denne skal funktionen fødes med oplysninger om, med hvilken Facebook applikation der skal initialiseres med, denne applikation kunne givetvis være en af de applikation der i forvejen benyttes af Drupal for Facebook modulet, det kunne derfor tænkes relevant med et menupunkt til at vælge hvilken af ens installerede applikationer der ønskes benyttet til publicering.

8.3.2 previewfb

Denne funktion fødes med et id til at identificere hvilken node artikel der skal benyttes til previewet, funktionen ud fra dette id være i stand til at hente de rette oplysninger der skal bruges til visningen. Funktionen returnerer visningen i form af en HTML side.

8.3.3 post_test

Vil kræve et id til at identificere hvilken node artikel der skal benyttes og skal fødes med et Facebook profil id (fbu), til hvilket test beskeden skal sendes. Ved at benytte Facebook instansen forsøges beskeden sendt i tilfælde af fejl, bør der naturligvis være en form for fejlhåndtering. En eventuel naturlig udvidelse kunne være en funktion som automatisk henter et fbu, hvis redaktør brugeren er logget ind på Facebook, således at vedkommendes profil bruges til testen.

8.3.4 post

Denne funktion vil dybest set køre på samme måde som post_test, men forskellen her er, at denne skal bruges til den faktiske publicering. Dette betyder at funktionen skal bruge en anden form for fejlhåndtering og så skal den opdatere en database ved hver succesfuld publicering, hvori den publicerede beskeds ID nummer på Facebook gemmes. Mht. fejlhåndteringen bør det ikke være noget der foregår i selve post-funktionen, men i stedet sendes fejlen, hvis nogen, til den løkke funktion (post_spool) hvori post-funktionen kaldes, denne kan så med fordel tage sig af eventuelle API fejl.

8.3.5 commentpost

Denne er bygget op ligesom post-funktionen, den skal dog også fødes med et kommentar id (cid), der identificerer præcis hvilken kommentar, der skal publiceres, ud fra dette hentes den rette kommentartekst. En kommentar er, som nævnt, lidt simplere i sin opbygning, fordi den kun indeholder en tekststreng og intet andet, det er derfor kun denne der skal hentes. Fejlhåndteringen vil også her ligge hos post_spool, funktionen sender derfor blot eventuelle fejl videre i systemet. Her behøves ligeledes en databasetabel til at holde styr på publicerede kommentarer, ved et succesfuldt API kald indsættes derfor kommentarens Facebook identifikationsnummer i tabellen.

8.3.6 deletefb

Denne funktion skal bruges til at slette publicerede beskeder fra Facebook (der bruges en anden til at slette selve node artiklen i Drupal). UC6 og UC9 kan her slås sammen, så det bliver en samlet slet-funktion. Funktionen laver dermed et API kald om sletning af en specifik besked, som er identificeret fra databasen hvori tidligere publicerede beskeders Faceboook id er blevet gemt. Efter beskeden er slettet fra Facebook bør databasen naturligvis også ryddes op, funktionen skal derfor efter et succesfuldt kald slette de tilhørende rækker med gemte kommentar- og besked id numre. Fejlhåndteringen her bør ligeledes blive styret af post_spool.

8.3.7 get_status

For at holde styr på om en besked eller en kommentar tidligere er publiceret eller slettet, behøves en database hvori en status kan gemmes for et bestemt node- eller comment id (nid, cid). get_status funktionen laver et opslag i en af disse databaser baseret på et id-nummer og returnerer den tilhørende status. Statusser kunne være: ikke sendt, sendt, slettet.

8.3.8 post_spool

Denne funktion er selve kernen i publiceringssystemet. I postterminologien kunne den kaldes distribueringscentralen, for det er det den fungerer som, en funktion der sørger for at processere beskeder der er fyldt i en tabel og kalde post, commentpost eller deletefb funktionerne alt efter, hvilken type kald der er tale om. Først skal spolen med data hentes, spolen skal indeholde informationer om brugere og beskeder/kommentarer.

Når spolen er hentet ind, kører en løkke der for hver række i tabellen, som udfører den korrekte handling baseret på typen. For hvert gennemløb af en række i tabellen tælles antallet af fejl og succeser, ved denne

fejlhåndtering fås et tal på, hvor mange publiceringer der fejlede og hvor mange der var succesfulde. Fejlene kan så gemmes i Drupals indbyggede log system f.eks.

Efter hvert gennemløb skal rækken slettes fra spolen for, at sikre den ikke bliver gentaget ved en fejl. Når alle rækker i spolen er processeret, skal status opdateres således at det fremgår i Drupal, hvad der er foretaget.

8.3.9 save_spool

Denne funktion bruges til at gemme data i spolen og er blot en forespørgsel til den tilhørende database om indsættelse af data, der er sendt i forbindelse med kaldet af funktionen.

8.3.10 get_spool

Returnerer hele spolen og er blot en forespørgsel til databasen.

8.3.11 clear_spool

Inden publicering kan det være en god idé at tømme spolen i tilfælde af at der ligger gammelt affaldsdata i den, denne funktion laver en forespørgsel om at slette alle rækker i spolen.

8.3.12 checkUser

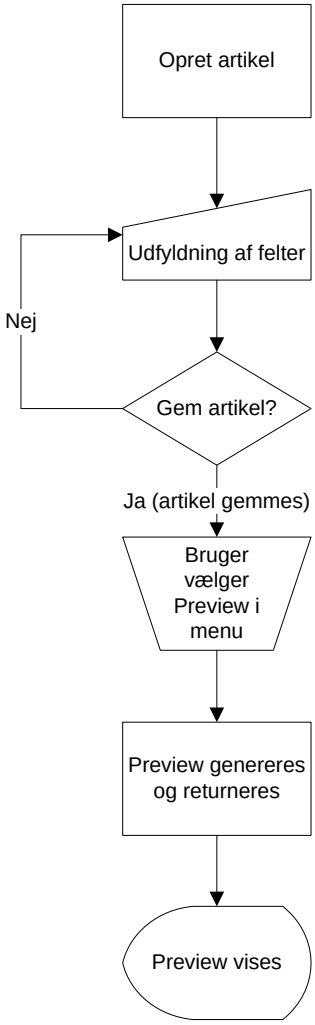
Der kan i mange situationer være tilfælde, hvor en bruger der allerede er oprettet og logget ind ikke i den gældende session, stemmer overens med de data, der ligger i systemet om brugeren, derfor kan det være praktisk at have en funktion der tjekker en bruger og gemmer de gældende oplysninger om vedkommende. Der er fire tænkelige scenarier hvor en gældende session for en bruger ikke stemmer overens med gemte oplysninger, disse er:

1. Hvis brugeren er logget ind uden at være blevet mappet, dvs. oprettet som en lokal bruger i systemet, så skal brugerens Facebook profil id og fbu gemmes.
2. Hvis den adgangsnøgle der er gemt i systemet ikke svarer til den, på nuværende tidspunkt, gældende adgangsnøgle for brugeren, skal den gamle overskrives med den nye.
3. Hvis der for brugeren slet ikke eksisterer en adgangsnøgle i systemet, skal den gemmes.
4. Hvis brugeren ikke har accepteret den applikationen der bruges til publikation, eller har været inde i sin profil og ændre, hvilke rettigheder applikationen har, skal det sørges for, at brugeren accepterer applikationen før videre adgang gives. Et API kald til en bruger der ikke har accepteret publikationsapplikationens extended permissions, vil returnere en fejl.

8.4 Systemflow

I dette afsnit vises en repræsentation af, hvordan forløbet I systemet er fra oprettelse til publicering, det er forsøgt at få så mange use cases som muligt ind i hvert diagram, for dermed at være så dækkende for systemet som muligt.

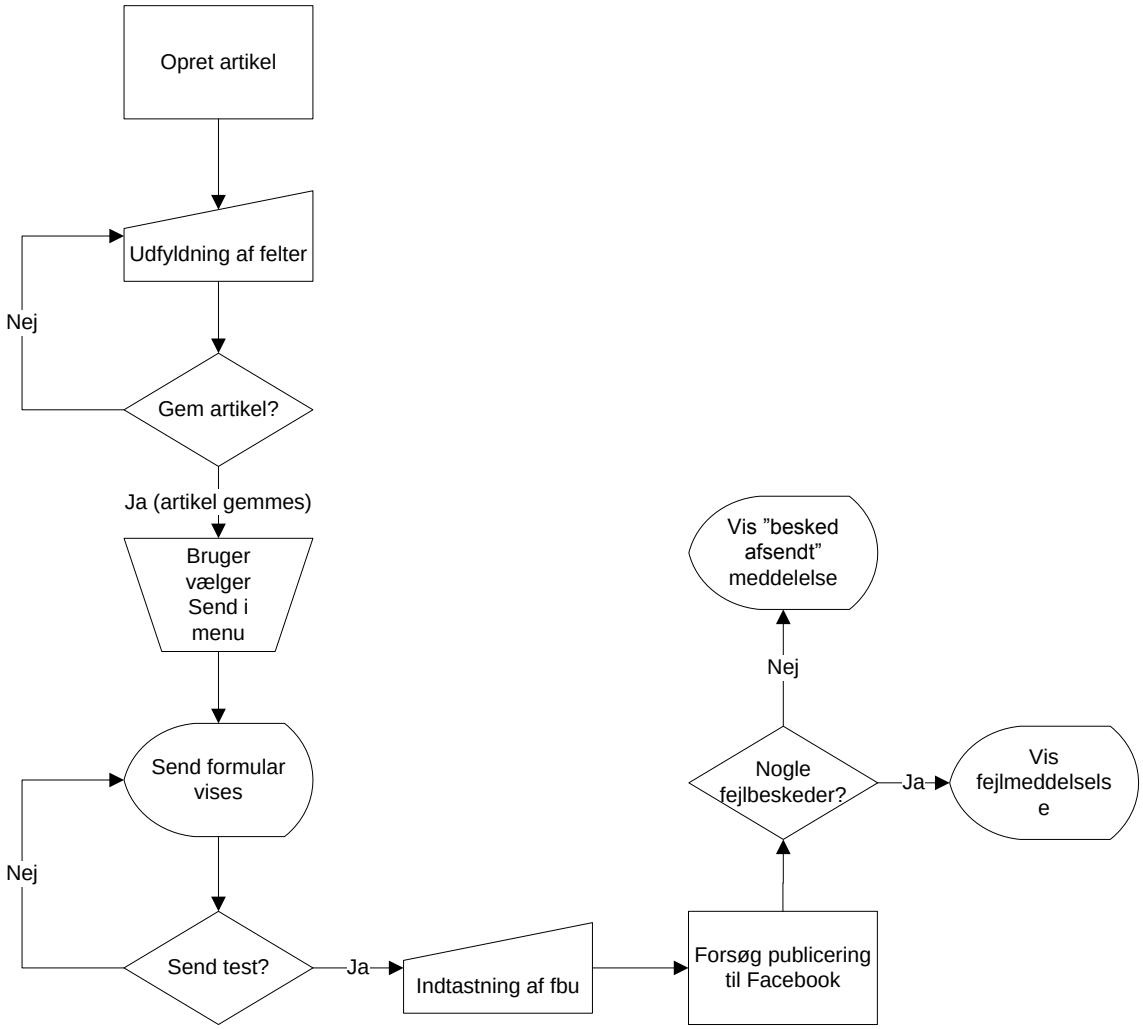
UC2 Preview er yderst simpel da den blot er en forhåndsvisning internt i systemet, herunder ses use casens repræsentation i et flowdiagram:



Figur 14: Flowdiagram over preview

At vise previewet er generelt en simpel funktion, arbejdet ligger mest i at tilpasse previewet til at se ud som Facebooks reelle visning af indholdet, bemærk at use casen UC1 opret også er en del af diagrammet, den del der hører til preview, er punktet hvor brugeren vælger ”Preview” i menuen.

UC3 Send test vil være en simplere form af use casene 6 til 9, dens basale opbygning kan ses i flowdiagrammet herunder:



Figur 15: Flowdiagram over send test use case

Send test er en forholdsvis simpel use case, hvilket afspejles i diagrammet.

Herunder ses et flowdiagram, der viser flowet fra oprettelse af en Facebook post node artikel til den er publiceret og logget, diagrammet dækker i princippet over use casene UC6 til 9, diagrammet er en udvidelse af forrige fra UC3:

Figur 16: Flowdiagram over publicering

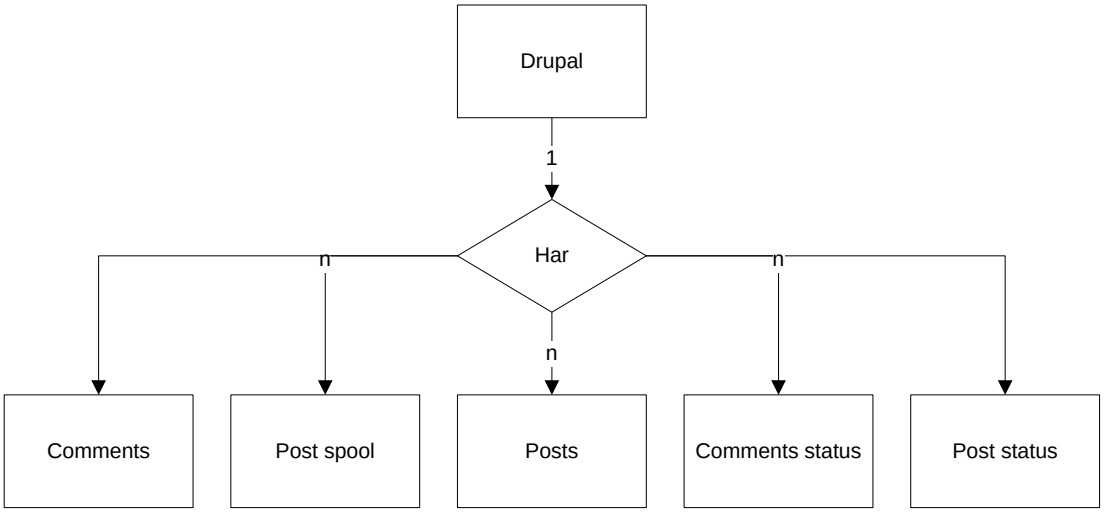
Publicering er den mest omfattende proces, der vil kræve en række forskellige funktioner. Under implementering vil det sandsynligvis være nødvendigt med endnu en række tilføjelser for, at imødekomme specifik implementering i Drupal, diagrammet herover bør som generel betegnelse være minimum under implementering for at opfylde de use cases der har med publicering at gøre (UC6 til 9). I tilfælde af sletning er det en anden side end send-siden der vises, således at sletning foregår på en separat siden, sletning er dog også en form for publicering teknisk set idet der skal gennemføres en række kald til Facebook uanset om der udsendes eller slettes beskeder. Hvis det er en kommentar der oprettes skal opret artikel delen være udført, før en kommentar kan oprettes.

8.5 Database

Systemet udnytter hovedsageligt data fra de moduler, der er tilgængelige i modulpakken Drupal for Facebook, og har dermed ikke behov for større databaser, dog har systemet brug for at gemme oplysninger om de poster og kommentarer der oprettes, derudover kræves der en database til at håndtere beskeder når de skal udsendes, dette er en slags postkasse som fyldes op med alle beskeder og tømmes gradvist som de bliver sendt ud. Et databassedesign på tredje normalform er at foretrække eftersom det sikrer mod inkonsistens i data og yder bedst.

8.5.1 Entitetsrelationer og relationel model

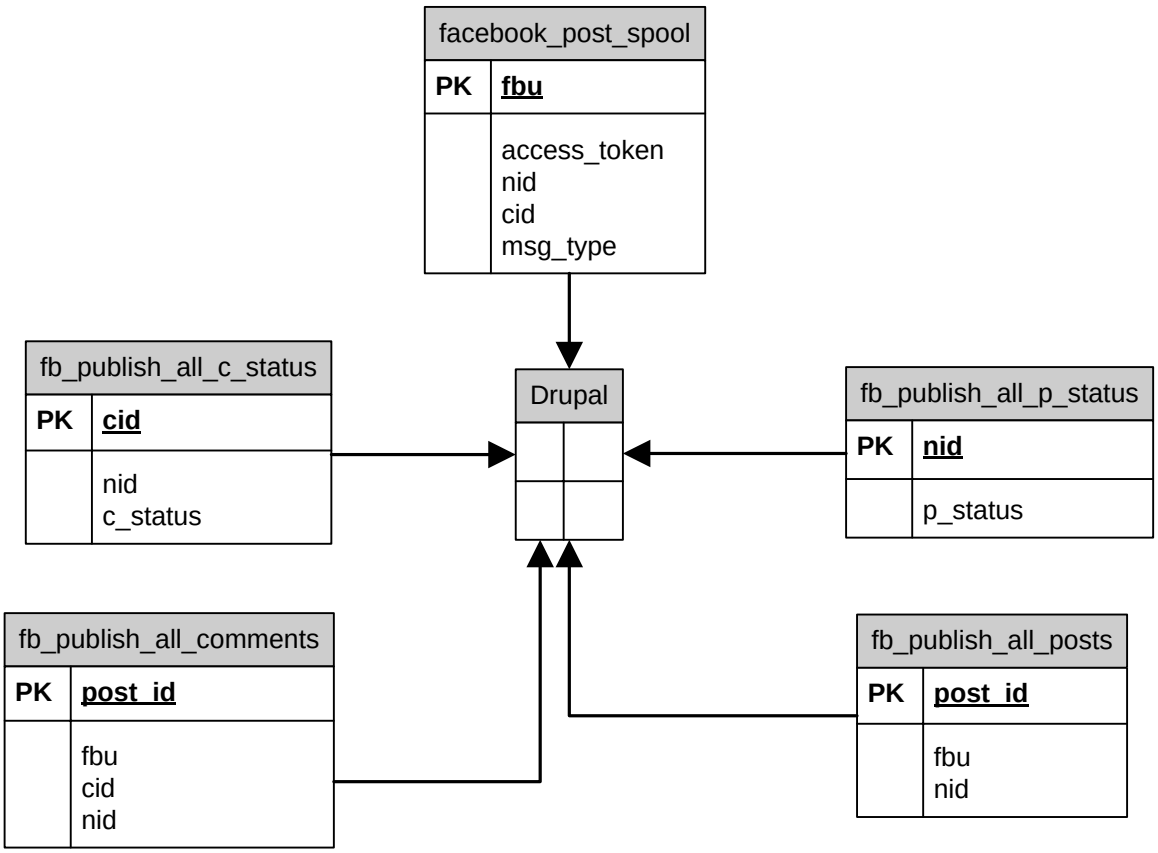
Databasen til systemet er som udgangspunkt fremkommet efter en vurdering af behov i kravspecifikationen hvortil der er forsøgt lavet et ER-diagram (Entity Relationship), der viser en abstrakt præsentation af databasens tabeller.



Figur 17: Entitetsrelation diagram

I den i projektet brugte Drupal installation indgår mere end 80 tabeller, alle dem der ikke har direkte forbindelse til projektet indgår derfor i en entitet kaldet ”Drupal”, som altså dækker over hele systemets basisdatabasestruktur. Modellen viser relationen databaserne imellem og må betragtes som forholdsvis simpel idet der ikke benyttes en særlig avanceret struktur til systemet, koblingen mellem projektentiteterne er derfor også lav da de ikke direkte afhænger af hinanden, men afhænger af kernesystemet (Drupal Core), derfor er det heller ikke noget særligt krav at entiteternes relation er stærk.

Ud fra kravspecifikationen og ER modellen kan en mere præcis relationel databasemodel ligeledes fremstilles:



Figur 18: Relationel databasemodel

Til Forskel fra ER modellen ses det i den relationelle model, at attributternes navne og primærnøgler også indgår (PK = Primary Key), den giver derfor et mere detaljeret overblik over, hvilke dele der er brug for i databasedesignet. Som nævnt tidligere dækker Drupal tabellen i modellen også her faktisk over alle Drupal Core tabeller.

8.5.2 Oversigt over databaser

For bedre forståelse af behovet for disse databaser, følger her en kort beskrivelse af hver tabels formål:

- *fb_publish_all_posts* indeholder et register over Drupal nodes der er publiceret til Facebook, ved hvert succesfuldt API kald om publicering returneres et unikt ID af Facebook, dette ID kan senere bruges til identificere den korrekte post på Facebook, hvis det ønskes at publicere en kommentar eller slette den oprindelige post. Tabellen arkiverer således identifikationsnumre for beskeder der er publiceret til Facebook, tabellen indeholder i øvrigt Drupals ID til beskeden også, dette kaldes ”nid”, som er et node identifikationsnummer.

- *fb_publish_all_comments* er et register over publicerede kommentarer, ligesom med beskeder returnerer Facebook også et kommentar ID ved et succesfuldt API kald, disse ID numre skal gemmes hvis kommentaren senere ønskes slettet igen. Bemærk at Drupals pendant hedder ”cid”

som er et unikt nummer til at identificere en kommentar i Drupal systemet.

- *fb_publish_all_p_status* denne holder blot styr på en nodes status, altså om den er udsendt eller slettet, dette skal bl.a. bruges for at forhindre at kommentarer publiceres når en node ikke er det.

- *fb_publish_all_c_status* fungerer på samme måde som post status, men holder i stedet styr på en kommentars status og kan bl.a. bruges til at forhindre forsøg på sletning af kommentarer der ikke er publiceret.

- *facebook_post_spool* denne tabel benyttes i forbindelse med publicering og fungerer lidt som en postkasse, der fyldes med breve der skal sendes. I dette tilfælde er det dog Facebook jobs postkassen fyldes med og alt hvilken type job der er tale om (msg_type) sørger logikken for at sende og gøre det rigtige, i den forbindelse kræves ofte en adgangsnøgle (access_token) af Facebook for at udføre en bestemt handling, denne varetages af et modul fra Drupal for Facebook pakken, som sendes med, for at blive i post-terminologien kunne man kalde det for frimærket.

8.6 Løsningsmodeller og metoder

8.6.1 Udviklingsmetoder

I starten af opgaven blev nogle bestemte udviklingsmetoder gennemgået bl.a. iterationscyklussen, Unified Process (UP) og den simplere vandfaldsmodel. UP har sin helt klare fordel i store omfattende projekter, der kræver klar koordination og en opdeling i faser (iterationer), for nemmest at opfylde en eventuel omfattende kravspecifikation. Dette projekt har dog en relativ lille kravspecifikation, hvorfor det måske kan virke unødvendig omfattende at benytte, samtidig er vandfaldsmodellen lidt for simpel idet det må antages at hvert step i modellens nedadgående trappe, sjældent vil være helt afsluttet før næste påbegyndes. En anden stor svaghed ved vandfaldsmodellen er, at den forudsætter at alle projektets aspekter kendes fra starten af, men som nævnt i risikoanalysen kan der være en række ukendte faktorer og begrænsninger der spiller ind og dermed gør modellen uhensigtsmæssig.

En mere hensigtsmæssig model er derfor spiralmodellen, da den tager højde for at der kan være uforudsete krav, der skal bygges ind, samtidig med at den er risikostyret, det er også en stor fordel at en iteration kan overstås relativt lettere end i UP modellen, da spiral modellen ikke skal igennem dens bureaukratiske proces for hver iteration. Samtidig er den ligesom UP inkrementel og giver derved et mere færdigt produkt for hvert gennemløb, hvortil dokumentation følger med udviklingen. Derfor er det kun naturligt at benytte spiralmodellen til udviklingen i dette projekt.

8.7 Processuel og objektorienteret programmering i Drupal

I dette afsnit argumenteres for, hvilken implementeringsmetode der er bedst til projektet. I PHP udvikling er der sædvanligvis to gængse måder udvikle på, den ene og mest almindelige i små projekter er processuel udvikling, dette er også den gammeldags fremgangsmåde de fleste PHP udviklere lærer først. Det gængse PHP script vil være processuelt opbygget dvs. at funktionerne kommer en efter en nedad i dokumentet.

Den anden gængse metode er objektorienteret programmering (OOP), hvor man i stedet for en masse funktioner efter hinanden, har klasser med objekter, dvs. datastrukturer bestående af data og metoder, den gængse tanke er at OOP er nemmere at vedligeholde og sætte sig ind i pga. denne opdeling.

Drupal er kodet processuelt, men opbygget objekt orienteret, dvs. at Drupal er opbygget af moduler, temaer, nodes, users osv. der hver især kunne betragtes som objekter, disse er dog set på enkeltvis opbygget processuelt. En node er et dataobjekt der kan indeholde en lang række felter og kan udvides efter behov, disse defineres i et tilhørende modul og bringes sammen af en database tabel, således er systemets datastruktur ”OOP-agtig”, men benytter sig af en tabel i stedet for en klasse. Moduler og temaer opfylder på mange måder rollen som controller-klasser, da hvert modul samler relaterede funktioner og definerer hooks.

8.7.1 Indkapsling, polymorfi og arv

Hook systemet i Drupal laver basis for systemets interface abstraktion, hvor hooks definerer de operationer der kan foretages på og af et modul. Hvis et modul implementerer en hook, udfører det en helt bestemt opgave og returnerer en bestemt type information, når det kaldes.

Drupal indbygger i nogen grad indkapsling idet alle funktionsnavne indeholder et namespace, der som regel er modulets navn, på den måde sikres det at navnene aldrig kommer i konflikt med eksisterende. Drupal definerer dog ikke hvilke klasser der er private eller public.

Polymorfi bruges også ved nodes, hvor et modul kan vise en node ved at kalde en funktion i node-modulet, men den faktiske visning vil være afhængig af, hvilken type node der sendes til funktionen, dette er ligesom at have en klasse af et objekt der bestemmer adfærden når en besked sendes til den. Ligeledes med temaer der modtager en besked og renderer den forskelligt baseret på temaets implementering.

Moduler og temaer kan også ses som klasser der nedarver deres adfærd fra en abstrakt grund klasse. I Drupal core følger en grundimplementering, men med hooks i moduler kan man overskrive disse og ligeledes med temaer, som arver en grundstruktur fra systemet, der kan bygges ovenpå efter behov. Nogle moduler er dog ret omfattende og indeholder et stort antal funktioner, hvoraf de fleste ikke kan føres ind i nye moduler og overskrives.

8.7.2 Designmønstre

Drupals struktur er dog mere avanceret end simpel nedarvning og polymorfi, på meget af systemets kernestruktur kan forskellige designmønstre være på tale.

- **Singleton**: Hvis moduler og temaer betragtes som objekter, så følger de singleton mønsteret idet de ikke indkapsler data, det der separerer moduler er de funktioner de hver især indeholder, så man kunne betragte dem som en klasse med en singleton instans.

- **Decorator**: Drupal gør i høj grad brug af decorator mønsteret igennem en række hooks eksempelvis i en node, hvor vilkårlige moduler vha. af hooks kan udvide adfærden af alle noder. Dette gør det muligt at tilføje en lang række af muligheder til noder uden at skulle skrive sub-klasser. Et i Drupal terminologi klassisk eksempel på dette, er at tilføje ekstra felter til en node, på den måde kan en eksisterende standard node-type udvides med et felt til f.eks. billedupload og vil dermed følge decorator mønsteret.

- **Observer**: Mange hooks følger ligeledes observer mønsteret da mange af dem tillader moduler at registrere sig som observator af Drupals objekter. Et eksempel er taksonomier, der hvis ændres, via en hook automatisk fortæller alle dele der benytter taksonomien, at den er ændret.

- **Bridge**: Drupals database lag er implementeret på en måde svarende til bridge designmønstret, idet moduler skal skrives uafhængigt af databasen der bruges. Man kan tilføje nye databasesystemer uden der behøves at ændres i modulernes kode, således at Drupal kan understøtte en række databasesystemer, pt. understøttes bl.a. MySQL, MSSQL, PostgreSQL samt mange andre.

- **Chain of Responsibility**: Drupals menu system indbygger dette designmønster idet menu systemet undersøger om der er et modul der skal tage sig af den enkelte sideforespørgsel, om brugeren har adgang til den specifikke side og hvilken funktion der skal kaldes i den forbindelse. Hvis menupunktet i sig selv ikke håndterer forespørgslen sendes den videre i kæden (systemet), indtil et modul tager sig af forespørgslen eller kædens ende nås.

Drupal benytter sig altså af en række OOP designmønstre, der normalt ikke ville være brugt i et rent processuelt designet system. Designmønstrene er nævnt fordi det er relevant at kende til Drupals design generelt. Om end designmønstrene ikke nødvendigvis afspejles i implementeringen af projektet, vil de være gældende eftersom implementeringen følger Drupals designstruktur. Designmønstrene benyttes altså ikke på det enkeltstående modul som projektet her er, men modulet er stadig omfattet af de designmønstre der generelt benyttes i Drupal.

Som nævnt er Drupal på overfladen et processuelt system, men det låner en række koncepter normalt kun brugt i OOP og måden hooks kan kaldes på fra vilkårlige moduler, gør at systemets adfærd, for en OOP programmør, næppe vil virke særlig fremmed. Grundet modulers processuelle opbygning, vil implementeringen af projektet følge dette, så produktet vil ikke blive designet objektorienteret, men processuelt og valideret efter Drupals designstandarder.

9. Implementering

I dette afsnit gennemgås, hvordan de 9 use cases er implementeret i Drupal baseret på designafsnittet. Alle use cases er implementeret, hvor nogle er implementeret i selve modulet og andre er et samspil eller udelukkende håndteret af Drupal core systemet. Man kunne derfor også inddele implementeringen i to dele, en som er hoveddelen og selve modulet, der kører på serversiden og den anden samspillet med Drupal. Derudover kører ved publicering et JavaScript på klientens computer, som også gennemgås i selvstændigt afsnit senere.

Først gennemgås hvordan et modul skal bygges, så det spiller sammen med Drupal, hvilke dele det skal indeholde, dernæst gennemgås grænseflade og database implementeringen med udgangspunkt i samme afsnit fra designfasen og til sidst alle aspekter af funktionalitetsimplementeringen alle med henvisninger til relevant kildekode. Der skal gøres opmærksom på, at ikke alle funktioner bliver gennemgået, herunder ikke støttefunktioner til de funktioner der reelt opfylder en use case, dette af hensyn til afsnittets længde.

9.1 Drupal modulet

For at kunne designe moduler til Drupal er det naturligvis nødvendigt at vide noget om, hvordan et sådan bygges op. Drupal.org indeholder i forvejen en temmelig omfattende vejledning (http://drupal.org/developing/modules), der hele tiden holdes opdateret.

Et Drupal modul er en samling funktioner som linker ind i Drupal og tilbyder funktionalitet der ikke i forvejen tilbydes af en standardinstallation. Et basalt modul består af tre filer en *.module*-fil der er selve modulet, en *.install*-fil der fortæller hvad Drupal skal gøre når modulet aktiveres og sidst en *.info*-fil der indeholder informationer om modulet såsom dets navn og en kort beskrivelse.

9.1.1 Info

Det første step i moduludviklingen er naturligvis at finde et navn til ens modul, i Drupal for Facebook hedder modulerne fb, f.eks. fb_stream eller fb_user_app, navnet til projektets modul er efter denne konvention blevet til fb_publish_all. Dette navn skal angives i en .info fil der fortæller Drupal, at modulet eksisterer og navnet skal bruges i navngivningen af alle filer:

```
name = "FB Publish All"
description = "Extends drupal for facebook with publish to all users
capabilities"
dependencies[] = fb
dependencies[] = fb_user_app
dependencies[] = fb_connect
dependencies[] = fb_permission
dependencies[] = content_copy
dependencies[] = imagefield
package = Drupal for Facebook - contrib
core = 6.x
```

Info filen kommer da til at hedde fb_publish_all.info der også angiver hvilken af version af Drupal det er lavet til, i dette tilfælde Drupal version 6.x, dependencies er andre moduler, som modulet er afhængig af og

package angiver en pakke modulet er en del af, denne bruges kun til at bestemme placeringen i visningen i Drupals moduloversigt.

9.1.2 Install

.install-filen er den fil der fortæller Drupal, hvad det skal foretage sig når modulet aktiveres eller deaktiveres, heri kan skrives en række funktioner der kun skal køres en gang, normalvis vil dette være oprettelse af en eller flere til modulet hørende databasetabeller og evt. installation af nye indholdstyper, taksonomier, menuer o. lign. Når modulet deaktiveres kan der givet vis også være funktioner der står for afinstallationen igen.

I projektets implementering forefindes således en funktion der opretter en ny indholdstype ved navn "Facebook Post", denne er afhængig af et modul ved navn content_copy, der følger med enhver Drupal installation, ved hjælp af dette modul kan en eksisterende indholdstype eksporteres eller en ikke eksisterende kan importeres, det er naturligvis det sidste, der her er brug for:

```
/**
 * Implementation of hook_install().
 */
function fb_publish_all_install() {
  // Create tables.
  drupal_install_schema('fb_publish_all');

  //Install the facebook post content type:
  $filename = drupal_get_path('module', 'fb_publish_all') .
"/fb_publish_all.install.inc";
  $content = implode('', file($filename));
  // Build form state
  $form_state = array(
     'values' => array(
        'type_name' => '<create>',
        'macro' => $content,
     ),
  );
  drupal_execute("content_copy_import_form", $form_state);
}
```

Dette er hovedfunktionen i installationsfilen hvor funktionen `drupal_install_schema`, sørger for at installere databasen (forklares nærmere i databaseimplementeringsafsnittet), herefter installeres indholdstypen der er angivet i filen fb_publish_all_install.inc, som er resultatet af en eksport via content_copy modulets eksport funktion, indholdstypen er altså oprettet manuelt i Drupal og derefter eksporteret så den fremover kan oprettes automatisk, hvilket er det der sker til sidst med `drupal_execute`.

9.2 Module

Den vigtigste fil af alle er naturligvis selve modulets kildefil .module-filen. Der vil ikke blive gået i dybden med navnekonventioner og kodestandarder heri, eftersom Drupal dokumentationen allerede går rigeligt i dybden med dette, i testafsnittet vil der dog blive foretaget en validering af om modulet følger disse kodestandarder. Her gennemgås det hvilke dele af projektets modul der er en del af den basis der udgør et basalt modul og dele der direkte vedrører Drupal implementeringen, som ikke er en del af nogle use cases.

9.2.1 Adgang

Det første der inkluderes, hvis der er behov for det, er altid en angivelse af, hvilke rettigheder modulet skal gøre tilgængelige, de bruges til at angive hvilke brugere i Drupal der skal have adgang eller mulighed for hvad. Implementeringen af dette ses herunder:

```
/**
 * Implementation of hook_perm().
 * adds permission to the content type created in the install file
 */
function fb_publish_all_perm() {
  return array(
    'administer facebook_post',
    'create facebook_post',
    'delete facebook_post',
    'delete own facebook_post',
    'edit facebook_post',
    'edit own facebook_post'
   );
}
```

Modulet tilføjer altså seks forskellige rettigheder, normalt vil disse kun være givet til redaktøren i systemet, men der er altså mulighed for at give andre adgang også. Til denne hører en funktion der foretager en handling alt efter brugerens rolle og hvilke adgangsrettigheder vedkommende har ved et forsøg på en given handling, dette håndteres af en tilhørende funktion `fb_publish_all_access($op, $node, $account)`.

9.2.2 Menuer

For at oprette nye sider og tilhørende menupunkter i Drupal bruges en funktion `hook_menu()`, denne gør det muligt for moduler at registrere stier koblet til bestemte URL forespørgsler eller angivelse af nye menupunkter i de eksisterende menusystemer. Definitionen af hver sti inkluderer sædvanligvis et side tilbagekald der kaldes når en bestemt URL forespørges, tilbagekaldet er oftest en funktion der køres.

```
function modulnavn_menu() {
  $items['/abc'] = array(
    'page callback' => 'modulnavn_funktion',
  );
  return $items;
}
```

I ovenstående generelle eksempel vil en forespørgsel på URL'en "/abc" køre funktionen modulnavn_funktion(). Reelt vil arrayet indeholde flere elementer end bare et tilbagekald, et uddrag af fb_publish_all_menu() funktionen ses herunder:

```
  //Node pages
  $items['node/%/sendfbtest'] = array(
   'title' => 'Send facebook',
   'description' => 'Send facebook test message or send to all',
   'page arguments' => array(1),
   'page callback' => 'fb_publish_all_send',
   'access arguments' => array(1),
   'access callback' => 'detect_fb',
   'type' => MENU_LOCAL_TASK
);
```

Ovenstående kode viser siden med formularen til publicering ved at kalde funktionen `fb_publish_all_send()`, typen angiver hvilken type menupunkt der er tale om, `MENU_LOCAL_TASK` er en tilføjelse til den eksisterende menu alle noder har til visning og redigering. Den sidste vigtige ting at have med er et adgangsargument, der fortæller at menupunktet kun skal vises hos en bestemt indholdstype, dette er adgangstilbagekaldet som kører en funktion, `detect_fb()`, der kontrollerer om noden er af typen facebook_post.

Figur 19: Skærmbillede af nodens menu

Der er naturligvis mange måder at implementere en menu på alt efter behov, hvoraf mere omfattende dokumentation kan findes hos drupal.org[21].

[21] http://api.drupal.org/api/drupal/developer--hooks--core.php/function/hook_menu/6

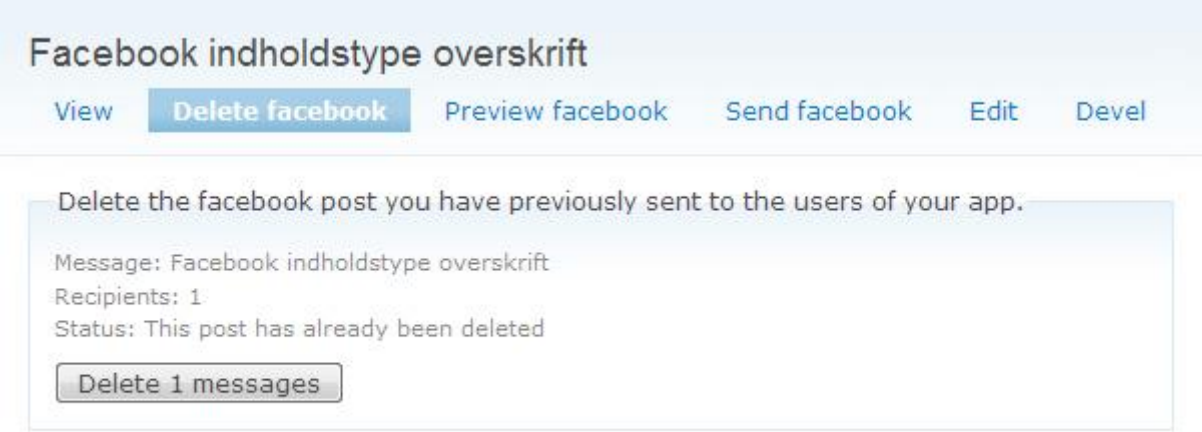
9.3 Grænseflade

Grænsefladen afspejler i høj grad den måde Drupal automatisk genererer indhold på, der er ikke produceret nogen specifik grafisk profil, skønt der eksisterer en pæn grafisk overflade, denne er en del af Drupals grundimplementering, hvor der i modulet blot benyttes et API til at integrere i dette. Farvevalg og styling af elementer håndteres udelukkende af det valgte Drupal tema (i projektet er benyttet standardtemaet fra en frisk installation, som hedder "Garland", dette kan udskiftes efter behov). Modulet bestemmer udelukkende elementernes placering i forhold til hinanden og er altså dermed fuldstændig uafhængig af temaets grafiske overflade.

Hvis udgangspunktet tages i designafsnittets del om grænseflade gennemgås her den faktiske grafiske repræsentation af de skitser der blev lavet tidligere, først ses visningen når Facebook noden er oprettet:

Figur 20: Skærmbillede af nodens visning

Skærmbilledet viser standardvisningen af noden, hvor det ses at der er en gruppe der indeholder felter til et link og forneden er der tilføjet en kommentar til noden. Det næste menupunkt ser ud på en lignende måde, her ses hvordan formularen til slet er implementeret i Drupals brugergrænseflade:



Figur 21: Skærmbillede af interface til sletning

Til forskel fra mock-upen er en status for beskeden også implementeret og en tæller der viser, hvor mange profiler ændringen vil påvirke. En besked kan have tre forskellige stadier:

1. This post has not been sent and cannot be deleted.
2. Active, press delete to delete from all users.
3. This post has already been deleted.

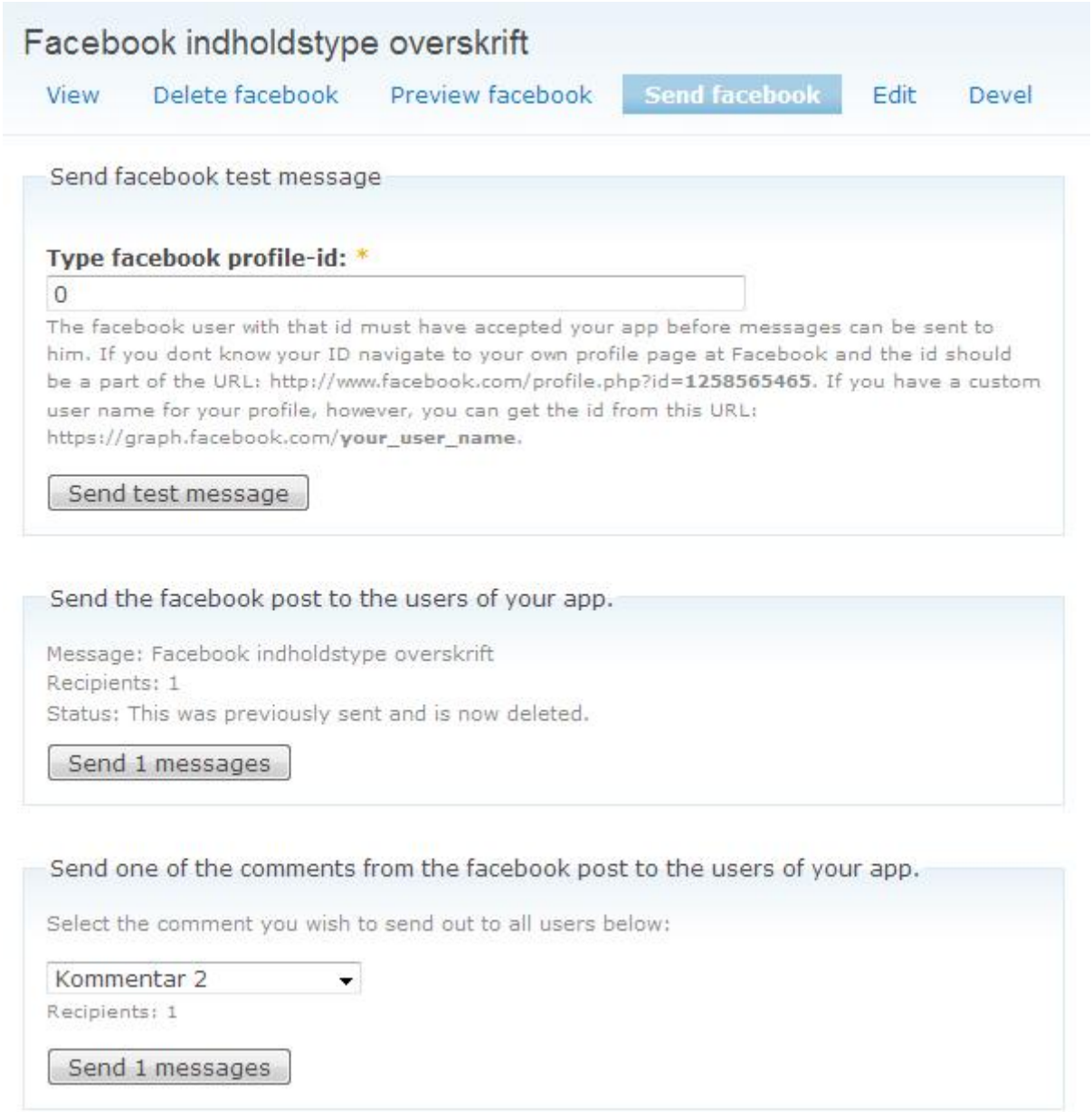
Det vil kun være i andet stadie at sletning giver mening, derfor vil en fejlbesked blive vist, hvis ikke beskedens status er ”aktiv”.

Forhåndsvisningen af hvordan beskeden ser ud, følger den faktiske visning så tæt som muligt, visningen foregår på en separat side, for at sikre at Drupals aktive tema ikke har indvirkning på resultatet, herunder ses øverst modulets implementering (over den sorte streg) og lige under ses, hvordan beskeden faktisk ser ud på Facebook for sammenligning:

Figur 22: Skærmbillede af preview øverst og faktisk repræsentation på Facebook nederst (under den sorte streg)

Det er vigtigt at betragte forhåndsvisningen som en tilnærmelse til den faktiske visning, da den reelle præsentation vil variere efter hvilke felter der er udfyldt, om det er i overensstemmelse med deres (varierende) politik og om det vedlagte billede er i portræt eller landskabsformat. Læg i øvrigt mærke til at navigationen ikke følger med over, så der er tilføjet et link til at gå tilbage til forrige side. Der er også en notits om længden på teksten, begrænsningen i Facebook hedder 420 tegn, denne begrænsning er det forsøgt at tage hånd om under oprettelse vha. et modul der hedder maxlength.

Send-siden følger rimelig præcist mock-upen fra designafsnittet og implementerer de samme tre formulerer, først en til at sende et faktisk preview på Facebook, send test, en formular til faktisk publicering og sidst en formular til at publicere kommentarer:



Figur 23: Skærmbillede af publiceringsinterfacet

Her fremgår også en status på beskeden, således at det er let at følge med i, om beskeden er usendt, sendt eller slettet og ligeledes fremgår antallet af modtagere. Hjælpeteksten under test besked er noget lang, fordi den skal forklare, hvordan et Facebook profil id findes, dette afhænger af, om vedkommende har angivet et brugerdefineret navn til sin profilside eller ej.

I anden formular kan tre forskellige statusbeskeder også være gældende:

1. This post has not been sent out yet.
2. This post has already been sent.
3. This was previously sent and is now deleted.

Ingen af de nævnte vil forhindre yderligere publicering, da det kan forestilles begrundet i alle tilfælde, at publicere på ny. I nederste formular til publicering af kommentarer vil en fejlbesked blive vist, hvis status er enten 1 eller 3, eftersom det ikke er muligt at poste kommentarer til en ikke eksisterende besked.

Der vil ikke blive gået specielt i dybden med "Edit" menupunktet, der også er en use case, da grænsefladen ikke er noget der er implementeret, men er en naturlig del af Drupal systemet. Implementeringen herunder gælder kun de tidligere beskrevne felter, som en Facebook besked består af, hvilke er sat op i Drupals grafiske brugerflade med tilhørende hjælpetekster.

En sidste lettere simpel implementering der er tilføjet til grænsefladen, er en konfigurationsside under Drupal for Facebook modulets konfigurationssider. Det eneste den giver mulighed er, at vælge hvilken applikation der skal benyttes af modulet til publicering, dette vil normalt være den samme applikation, som den der benyttes til Connect, men en anden kan altså vælges, hvis der er mere end én:

Figur 24: Grafisk femstilling af modulets administrationsinterface

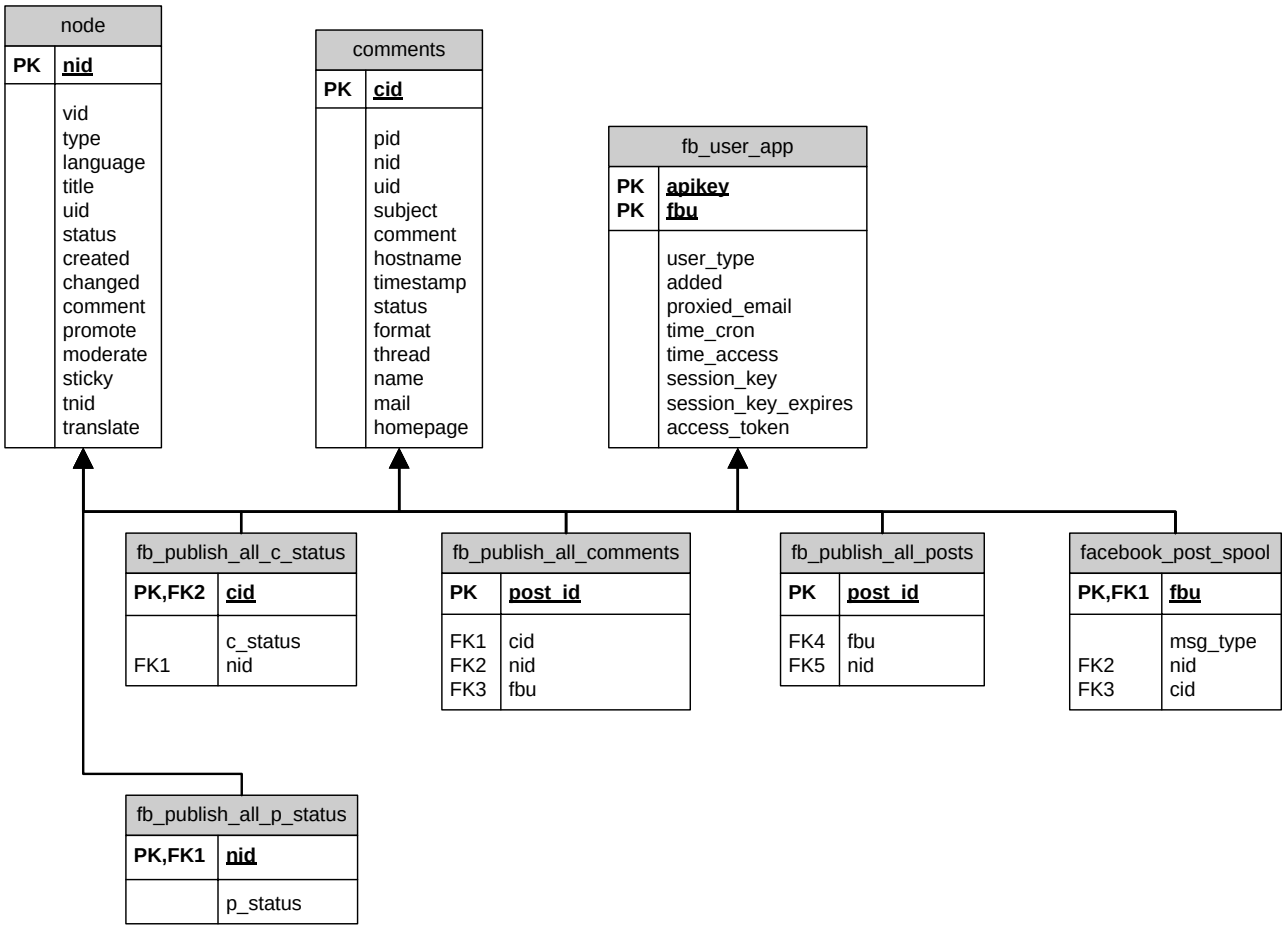
9.4 Database

I afsnittet om installationsfilen under Drupal implementeringen, kunne i koden ses, at der blev kaldt en funktion `drupal_install_schema('fb_publish_all')`, denne funktion sørger for at installerer de databaser, der er angivet nede i selve funktionen. I modsætning til sædvanlig praksis med SQL CREATE TABLE statements, har Drupal sit eget API, hvor hver tabel er angivet som et array af kolonner i tabellen, et eksempel på denne implementering ses herunder:

```
function fb_publish_all_schema() {
  $schema['facebook_post_spool'] = array(
  'description' => 'Spool for temporary storage of facebook posts.',
    'fields' => array(

      'fbu' => array(
        'type' => 'int',
        'size' => 'big',
        'not null' => TRUE,
      ),

      'access_token' => array(
        'type' => 'varchar',
        'length' => 255,
        'not null' => FALSE,
      ),
(...)
    ),
    'primary key' => array('fbu'),
  );

  return $schema;
}
```

Som det ses er hver enkelt kolonne i tabellen også et array, hvori det angives hvilken type data det indeholder, om et dataelement må antage en null-værdi, længde, størrelse og evt. en standardværdi. Som beskrevet tidligere giver denne implementering en fuldstændig uafhængighed fra databaseteknologien. Databaseforespørgsler er også uafhængige når de foretages i modulet, fordi der benyttes Drupals indbyggede funktioner, herunder den mest almindelige `db_query("Query string her");` som kan bruges til de fleste slags SQL forespørgsler, Drupal benytter SQL som det oversætter til den rette syntaks automatisk alt efter hvilken database teknologi der bruges.

I modsætning til designafsnittet er der i denne fase blevet set på præcis hvilke tabeller modulet vil være afhængig af. Det drejer sig om tre forskellige tabeller, modulet bruger primærnøgler fra, der for modulet derfor vil fremgå som fremmednøgler i modellen:



Figur 25: Relationel databasemodel over implementeringen.

Tabellerne node, comments og fb_user_app er de direkte relaterede tabeller, som i designafsnittet gik under viften "Drupal", disse er altså ikke en del af implementeringen, men af det eksisterende system. FK står for Forreign Key og er en fremmednøgle. Tabellen node indeholder alle informationer om en specifik node, ligeledes for comments og kommentarer, mens fb_user_app er Drupal for Facebook modulet tabel over brugere oprettet via Facebook Connect.

Det værd at bemærke at de indfødte tabeller ikke har fremmednøgler i modellen, selvom eksempelvis comments tabellen vil være afhængig af node tabellen i praksis, dette skyldes at Drupal version 6.x slet ikke understøtter fremmednøgler eller benytter en relationel model for dens database. Selvom der i modellen er angivet fremmednøgler for modulets databasetabeller, eksisterer de ikke i den faktiske implementering, fordi det slet ikke er muligt at implementere dem. Dette er naturligvis en væsentlig mangel i Drupal, som gør at redundans i databasen med reel sandsynlighed kan forekomme, de er dog stadig med i modellen for at vise, at der burde være en relation og for at gøre det nemmere at se, hvor værdierne stammer fra.

Hernæst vil hver tabel der indgår som en del af modulet, blive forklaret lidt nærmere. De følger alle det oprindelige design fra designfasen, der henvises således også til beskrivelserne deri. Selve implementeringen findes i .install-filen.

9.4.1 Posts og status

Posts tabellen indeholder informationer om publicerede beskeder herunder beskedens unikke id på Facebook, dens id i Drupal og et id for den bruger beskeden er publiceret til. Tabellen holder altså styr på noder der allerede er publiceret, disse informationer er vigtige at have for de use cases der dækker kommentarer og sletning eftersom funktionerne til udførelse af netop disse bruger informationerne fra posts tabellen.

Status tabellen er egentlig ikke en del af nogen use case, men er implementeret dels af hensyn til brugervenlighed og som en anerkendelse af risici forbundet med publicering, da statussen bruges til at forhindre publicering af kommentarer, på den måde undgås det at en masse unødvendige fejlbeskeder logges i Drupal. Samtidig er det også nyttigt at vide om beskeden er sendt eller ej, hvis ens system indeholder mere end én person med redaktørrolle.

fb_publish_all_p_status	
PK,FK1	**nid**
	p_status

fb_publish_all_posts	
PK	**post_id**
FK4 FK5	fbu nid

Figur 26: Oversigt over posts og p_status tabellerne

9.4.2 Comments og status

Kommentar tabellen bruges næsten nøjagtigt som forrige med den væsentlige forskel at den indeholder informationer om publicerede kommentarer, hvilket inkluderer et id til at identificere kommentaren internt i Drupal og på Facebook for en specifik bruger.

Status er for kommentarer heller ikke en del af kravspecifikationen og indebærer heller ingen risici som sådan, men de er medtaget på en nice-to-have basis, for at lade redaktøren vide hvordan den aktuelle status er på hver enkelt kommentar.

fb_publish_all_c_status	
PK,FK2	**cid**
FK1	c_status nid

fb_publish_all_comments	
PK	**post_id**
FK1 FK2 FK3	cid nid fbu

Figur 27: Oversigt over comments og c_status tabellerne

9.4.3 Post spool

Post spolen fyldes med jobs når publicering påbegyndes og tømmes efterhånden som en besked publiceres til hver enkelt bruger i systemet, hver gang en række slettes i tabellen, vil de øvrige databaser blive fyldt med en række, således at der holdes styr på alle succesfulde kald. Tabellen indeholder de gængse identifikationsnumre der også ses i de førnævnte tabeller samt derudover en slags id til at identificere, hvilken type publikation der skal foretages, kolonnen kan antage værdierne ”post”, ”comment” eller ”delete”. Bemærk i øvrigt, at designafsnittets udgave også indeholder en kolonne kaldet access_token, denne er dog fjernet herfra fordi adgangsnøglen altid er knyttet til et bestemt fbu fra fb_user_app tabellen, tabellen ville derfor ikke være på tredje normalform, hvis den indeholdte en kolonne til adgangsnøglen.

facebook_post_spool	
PK,FK1	**fbu**
	msg_type
FK2	nid
FK3	cid

Figur 28: Oversigt over post spole tabellen

9.5 Funktionalitet

I forbindelse med implementeringen af selve funktionaliteten er som forbillede brugt de udvidede use case beskrivelser, det er derfor også implementeringen af dem der fremgår i dette afsnit. Det skal siges at grundet modulet er opbygget helt processuelt, er det ikke muligt at inkludere et klassediagram, som normal praksis ellers er i denne type opgave. Sekvensdiagrammer er også noget man typisk ville inkludere, men også disse egner sig bedst til OOP projekter, i stedet er der derfor blevet lavet aktivitetsdiagram for de vigtigste use cases, hvoraf use casens flow i systemet afbilledes.

9.5.1 UC1 Opret, UC5 Indhold og UC8 Kommentar

Den pågældende indholdstype "Facebook post" der ligger i Drupal er genereret af content_copy modulet og vedlagt i en separat fil ved navn fb_publish_all.install.inc, som er den egentlige implementering af UC5 use case. På skærmbilledet herunder ses at de felter der skal bruges, jf. designafsnittet, er oprettet. Det er denne opsætning som er eksporteret til filen.

Home » Administer » Content management » facebook post

facebook post

Edit *Manage fields* Display fields link
link description link image link title

Add fields and groups to the content type, and arrange them on content display and input forms.
You can add a field to a group by dragging it below and to the right of the group.

Label	Name	Type	Operations
Title	Node module form.		
Body	Node module form.		
Link	group_fb_link	Standard group	Configure Remove
link	field_fb_link	Text	Configure Remove
link title	field_fb_link_title	Text	Configure Remove
link description	field_fb_link_descr	Text	Configure Remove
link image	field_fb_link_image	File	Configure Remove

Figur 29: Felter som indholdstypen Facebook post indeholder

Den egentlige implementering af dette fremgår faktisk af afsnittet Install under "Drupal modulet" i dette kapitel. UC1 dækker blot over oprettelsen af en node af denne indholdstype. Når indholdstypen er installeret navigerer man til node/add/facebook-post for at oprette noden, dette er implementeret i Drupal og er en standard core funktion. Ligeledes er kommentarer der kan oprettes under den enkelte node, normalvis skal kommentarer aktiveres for hver indholdstype, det er dog bygget ind i installationsimportfilen og dermed vil kommentarfunktionen være aktiveret som standard.

9.5.2 UC2 Preview

Benytter en relativ simpel funktion, `fb_publish_all_previewfb($nid)`, som blot genererer forhåndsvisningen ved at indlæse en node angivet i funktionens kald og printer de gældende felter ud på en realistisk måde. Implementeringen af denne use case er faktisk det eneste sted i projektet, hvor HTML og CSS er benyttet ellers er resten ren PHP.

9.5.3 UC3 Send test

Udover de tilhørende formularer, som implementeringen af vises i et generelt afsnit senere, knyttes kun en funktion til denne use case nemlig, `fb_publish_all_post_test($nid, $fbu)`. Nid bruges til at indlæse de korrekte felter fra noden, og fbu angiver hvilken facebook profil test beskeden skal sendes til. Selve beskeden er i PHP angivet som et array der indeholder de samme felter som selve indholdstypen, felterne fra noden loades således ind i dette array. Selve udsendelsen af testbeskeden sker ved et kald til en funktion i Facebooks PHP API, for at fange eventuelle fejl er denne pakket ind i en try/catch:

```
      try {
        $res = $facebook->api('/'. $fbu .'/feed', 'POST', $post);
      } catch (FacebookApiException $e) {
        drupal_set_message(check_plain("An error occured type: " . $e,
'error'));
      }
```

I tilfælde af fejl fanges fejlbeskeden og printes ud til aktøren.

9.5.4 UC4 Rediger

Denne funktionalitet er en del af Drupals core og har derfor ikke været nødvendig at implementere igennem tredjepart. For at redigere en node klikkes blot på ”Edit” linket i dens menu.

9.5.5 UC6 Slet og UC9 Kommentar slet

Der ligger i virkeligheden to dele funktionalitet i denne use case, det ene er at slette selve noden internt i Drupal systemet og det andet er, at slette publicerede beskeder fra Facebook. Første del er en naturlig del af Drupal core og kan foretages ved at gå ind under menupunktet ”Edit” for nodes menu, nede i bunden findes en slet-knap.

Anden del indebærer et kald til Facebook om sletning for hver publicerede besked gemt i databasen, i praksis foregår det på næsten samme måde som den egentlige publicering, i stedet for:

```
$res = $facebook->api('/'. $fbu .'/feed', 'POST', $post);
```

Bruges kaldet:

```
$res = $facebook->api('/' . $idArray[0], 'DELETE', $post);
```

I modsætning til publicering af en besked, kræver sletning en adgangsnøgle som variablen `$post` indeholder (i publicering indeholder variablen selve den besked der publiceres). Funktionen der foretager sletningen er `fb_publish_all_deletefb($fbu, $access_token, $nid)`, i den køres også en try/catch ved selve kaldet hvor variablen `$res` bliver TRUE hvis kaldet er succesfuldt, i dette tilfælde slettes alle rækker med pågældende nid i tabellen, som holder styr på publicerede beskeder og eventuelt også alle rækker med det pågældende nid i tabellen der holder styr på publicerede tilhørende kommentarer.

Aktivitetsdiagrammet til sletning er stort set magen til det for publicering, triggeren er bare anderledes idet den ligger i anden formular under et andet menupunkt i systemets grafiske brugerflade.

9.5.6 UC7 Publicering

Publiceringsdelen dækker over to funktioner eftersom der er to forskellige publiceringsscenarier. Det ene til selve noden benyttes funktionen `fb_publish_all_post($fbu, $nid)` og det andet til kommentarer benyttes `fb_publish_all_commentpost($fbu, $access_token, $nid, $cid)`. De fungerer dybest set som sletningen udover at der selvfølgelig ikke slettes noget, men derimod publiceres. Efter hvert succesfuldt kald indsættes en ny række i de respektive tabeller, der skal holde styr på publicerede noder og kommentarer. For hvert succesfuldt kald til Facebook antager den førnævnte variabel `$res` her en værdi i form af et id nummer der identificerer den pågældende post eller kommentar i Facebooks system.

Disse er naturligvis nødvendige som selve den basale publiceringsfunktionalitet, men rundt om disse ligger funktioner til at styre en kontinuer strøm af beskeder som fylder og tømmer post spole tabellen. Det sker ved, når aktøren har igangsat publiceringen, at aktøren først sendes til en ny side, hvor der kører et JavaScript og hovedfunktionen `fb_publish_all_post_spool()` der tager sig af distribueringen kaldes. Forinden er spolen blevet fyldt op i formularens submit funktion.

Lidt basalt set henter post_spool funktionen bare informationen om hvilken type publicering det er fra spole tabellen og distribuerer til den rette publiceringslogik alt efter typen. Den har også indbygget logning således at alle fejl bliver logget i Drupals interne logsystem der hedder watchdog. Det logges også når publicering starter og hvornår den slutter, så det fremgår af loggen, hvornår og hvor lang tid den har fundet sted.

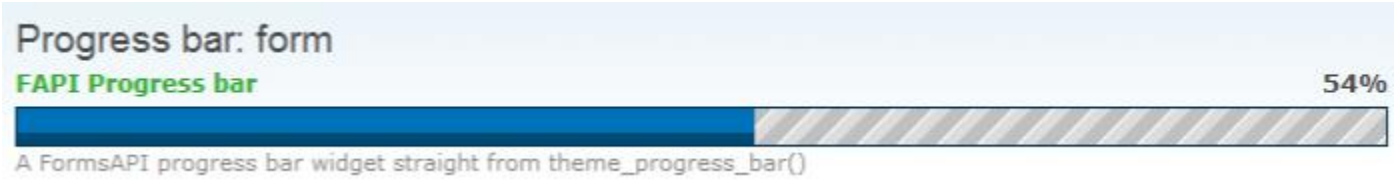
Når publiceringen er afsluttet ledes aktøren automatisk hen til log siden, hvor vedkommende kan se, hvor mange der succesfuldt blev afsendt og evt. hvor mange der fejlede.

9.5.7 Javascript

For at undgå at ramle ind PHPs maksimale udførelsestid for et script, er publiceringen nødt til at blive sat op som en cron job. Cron er en teknologi der muliggør regelmæssig udførelse af funktionalitet, det kan f.eks. være et tjek for opdateringer, indeksering af en hjemmeside eller lignende. I vores tilfælde vil der givetvis være mange brugere at publicere en besked til, det er derfor nødvendigt at køre `fb_publish_all_post_spool()` i et cron job, dette gøres meget enkelt i Drupal ved hjælp af en hook:

```
/**
 * Implementation of hook_cron().
 */
function fb_publish_all_cron() {
  fb_publish_all_post_spool();
  fb_publish_all_clear_spool();
}
```

Her ses at post spool funktionen initialiseres som et cron job, læg i øvrigt mærke til at spolen efterfølgende tømmes, dette er en sikkerhedsforanstaltning. Det i rapporten tidligere nævnte modul Simplenews kører også ved hjælp af cron jobs, hvor Drupal har en fin progress-bar indbygget.



Figur 30: Progress bar i Drupal

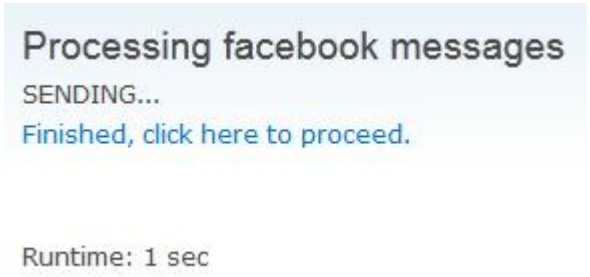
Den er dybest set et JavaScript der ligger ovenpå det cron job der kører i baggrunden og viser, hvor langt systemet er nået med en given opgave. Implementeringen i Simplenews modulet er dog noget mere omstændig, end hvad der er behov for i dette projekt, så i stedet er der blevet udviklet et simpelt AJAX JavaScript der på brugerens skærm viser beskeder sendt, hvor mange der er tilbage, tid der er gået og hvis muligt hvor meget tid der er tilbage (dette regnes ud automatisk). Et script kan også ret nemt indlæses vha. en hook:

```
/**
 * Implementation of hook_init().
 * Initialize the js script for publishing on the send page
 */
function fb_publish_all_init() {
  drupal_add_js(drupal_get_path('module', 'fb_publish_all') .
'/js/fb_publish_all.js');
}
```

Den grafiske repræsentation af scriptet vises på sending-siden, hvor scriptet kører så længe der er beskeder i spolen:

Figur 31: Oversigt over fremskridt under publicering

Scriptet fødes med en information om, hvor mange beskeder der skal sendes, når alle disse er kørt igennem vises et link som leder hen til Drupals log watchdog:

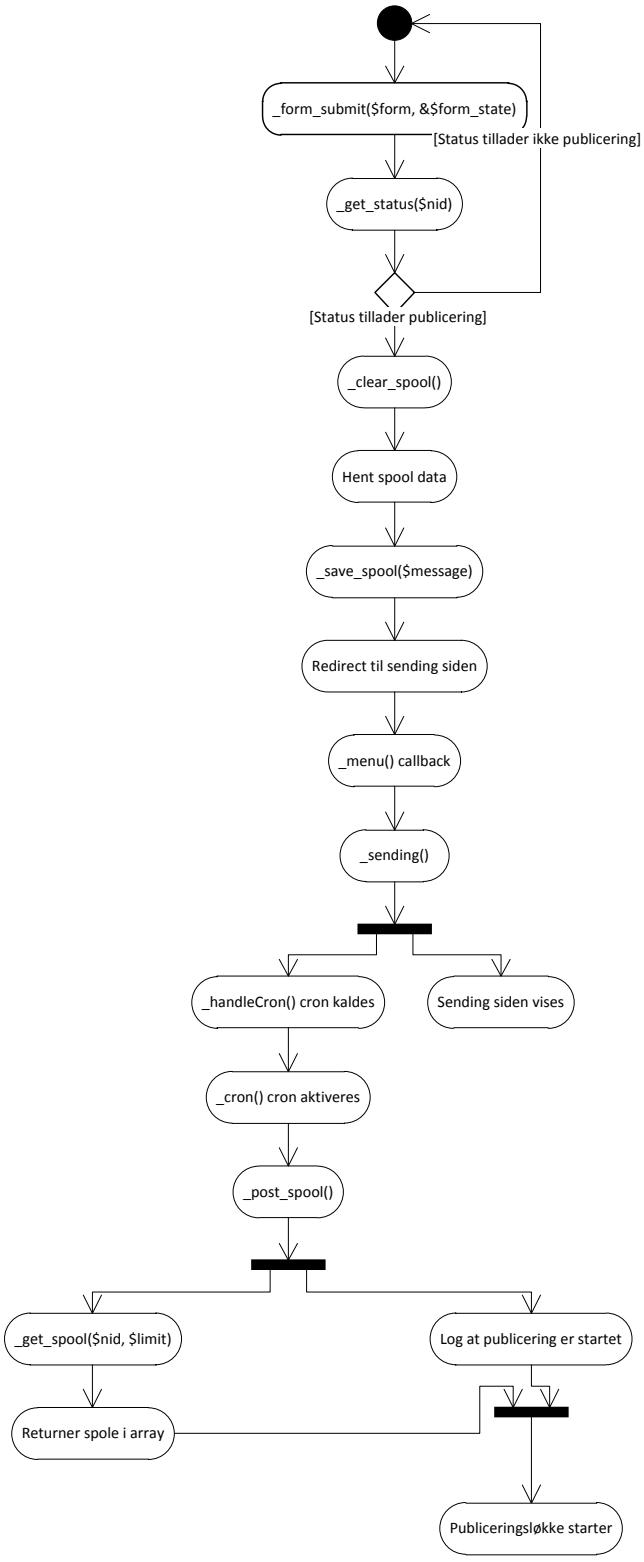


Figur 32: Visning af link til log efter afsluttet publicering

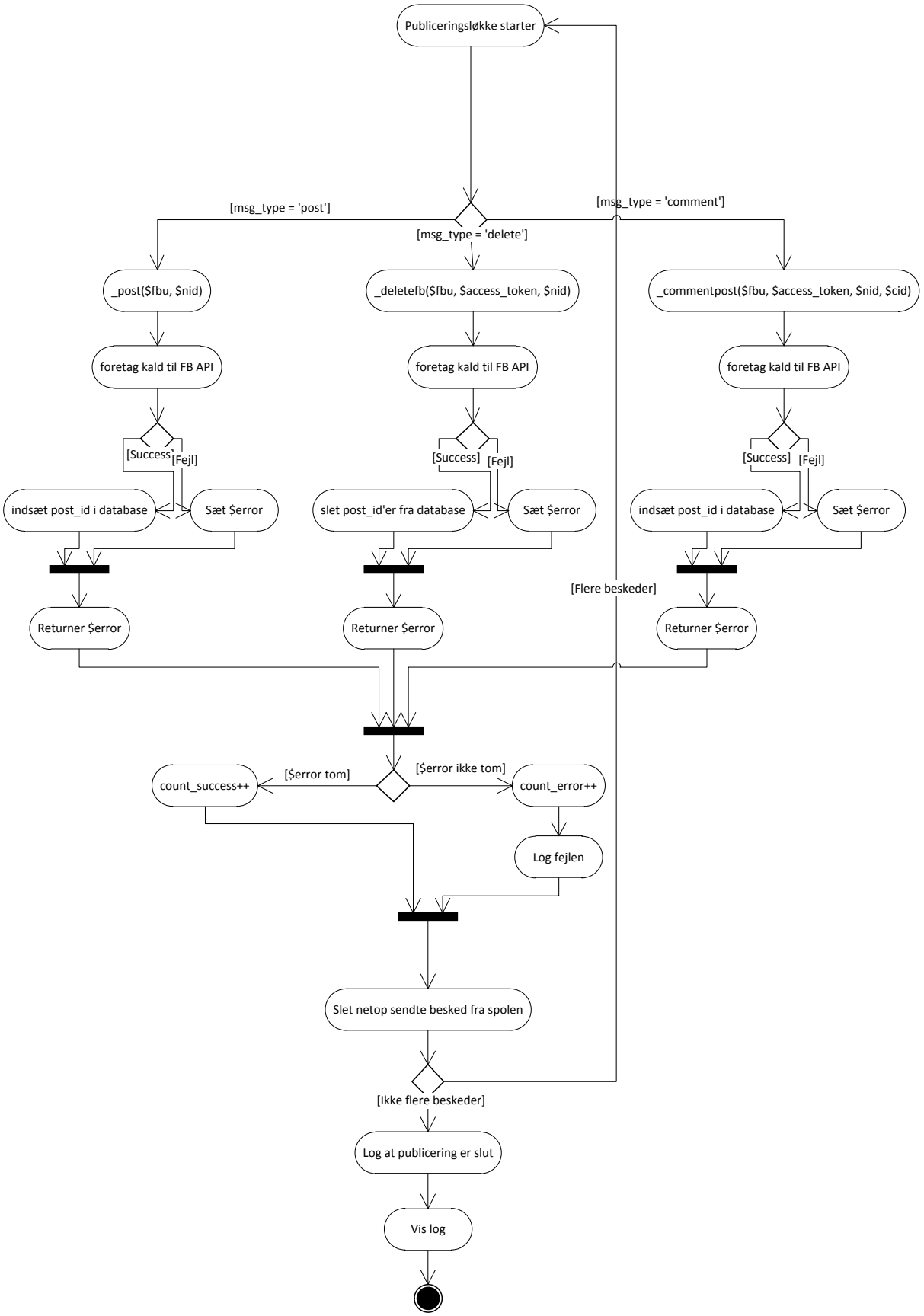
På log siden kan brugeren så se, hvordan publiceringen forløb, herunder hvilken specifik Drupal bruger der startede publiceringen og vedkommendes IP-adresse.

9.5.8 Oversigt over publicering (Aktiviteter)

I dette afsnit ses et aktivitetsdiagram over systemet og de funktioner og kald der indgår i aktiviteten publicering, som er den væsentligste, største og dermed mest relevante at vise i et diagram.



(fortsætter næste side)



Figur 33: Aktivitetsdiagram over publicering (Kan også findes på CD i stort format)

9.5.9 Formularer

Der bruges til den grafiske visning og til at påbegynde publicering en række formularer, disse er implementeret ved brug af Drupals form API som tilbyder tre forskellige funktioner til at bygge og styre formularer. Den første form bruges til at generere formularens indhold, i form_validate kan indbygges funktioner til at validere indtaste indhold og slutteligt er der form_submit der sender formularen. Her et eksempel fra koden der viser hvordan en send-knap genereres:

```
  $form['sendreal']['do_send'] = array(
    '#type' => 'submit',
    '#value' => t('Send !num messages', array('!num' => $numRecipients)),
  );
```

Alle form elementer er et array hvori typer, beskrivelser osv. Kan angives. Submit funktionen sørger for at sende de indtastede oplysninger videre i systemet dertil hvor de skal bruges. I dette projekt foregår selve valideringen længere nede i systemet, i selve formularen kontrolleres status kun for at tjekke om der kan publiceres. En fyldestgørende dokumentation over hele form APIet, kan ses hos drupal.org[22].

[22] http://drupal.org/node/751826

10. Test

I dette afsnit gennemgås forskellige typer test af det færdige produkt herunder gray box, validering og en kompatibilitetstest. Dette har til formål at kvalitetssikre og verificere, at systemet overholder de krav der blev formuleret i kravspecifikationen med henblik på kommercielt brug.

Med udgangspunkt i use cases er der lavet en række test cases, hvor både specificerede og uspecificerede krav indgår. Ligeledes udføres en validering af kodens kvalitet ved brug af Coder modulet i Drupal og en browsertest for at validere kompatibilitet i forskellige miljøer.

10.1 Gray box test

Til test af systemet er metoden gray box test valgt. Gray box er en blanding af white box og black box test, hvor testeren har indblik i systemets virke som i white box metodikken, dog kun de dele der har med test casene at gøre, dermed er der kun delvist indblik fra testerens side af, i black er der intet indblik og testeren ser udelukkende på input og output, dette er ikke optimalt eftersom det blandt andet ønskes testet om de korrekte databaseforespørgsler foretages. Testen er udført af udvikler og foretages ved at se på input og output i Drupal og på Facebook, databasetabellerne bliver kontrolleret igennem phpMyAdmin.

10.1.1 Test af UC1 Opret

Formål med testen:
Undersøge om UC1 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
Modulet skal være aktiveret og installeret med den korrekte indholdstype.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Oprettelse af en facebook post nodetype i admin menuen	Formular til oprettelse med korrekte felter vises til brugeren	Ingen
2	Gem noden og med vilje efterlade body felt tomt	Fejlbesked fortæller brugeren at feltet body skal udfyldes og gemmer ikke	Ingen
3	Gem noden efter indtastning af felter	Noden er gemt i Drupal og vises.	Ingen

10.1.2 Test af UC2 Preview

Formål med testen:
Undersøge om UC2 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
En facebook post node skal være oprettet.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Vis preview menupunkt i den oprettede node	Menupunktet vises i menuen	Ingen
2	Tryk på menupunkt	Preview vises i ny side	Ingen, men visningen kan afvige en smule.
3	Visning af preview	Alle indtastede relevante felter fremgår af previewet, men link oplysninger vises kun, hvis en URL er indtastet.	Ingen, men lange tekster vil blive beskåret af Facebook når der publiceres, dette fremgår kun i en note på Preview siden.

10.1.3 Test af UC3 Send test

Formål med testen:
Undersøge om UC3 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
Noden skal være oprettet og et validt Facebook brugerprofil id skal haves.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Et korrekt id indtastes og send test foretages	Beskeden viser sig på Facebook for pågældende bruger.	Ingen
2	Et ukorrekt id indtastes og send test foretages	Drupal viser en fejlbesked, beskeden sendes ikke.	Succesmeddelelse vises selvom afsendelsen fejlede
3	Hent Facebook profil id automatisk	I indtastningsfeltet vises aktørens Facebook profil id	Virker kun hvis: 1. brugeren er logget ind på Facebook 2. Brugeren har accepteret applikationen og er registreret i Drupal via Facebook Connect
4	Send test af besked uden link	Kun body felt vises på Facebook	Ingen
5	Indtast bogstaver i stedet for tal	Facebook returnerer ukendt id og fejlbesked vises	Succesmeddelelse vises selvom afsendelsen fejlede

10.1.4 Test af UC4 Rediger

Formål med testen:
Undersøge om UC4 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
Noden skal være oprettet og helst ikke publiceret, muligheden vil dog være til stede også efter publicering.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Menupunktet rediger trykkes	Indtastningsfelterne vises igen og kan redigeres.	Ingen
2	Rediger kommentar	Indtastningsfelterne vises igen og kan redigeres.	Ingen
3	Gem redigeret indhold	Noden med det nye indhold vises	Ingen

10.1.5 Test af UC5 Indhold

Formål med testen:
Undersøge om UC5 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
Modulets kildefiler skal være lagt ind i den rigtige mappe i filsystemet.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Afinstaller modulet under moduler	Databasetabellerne der hører til modulet slettes fra databasen	Ingen, men bemærk at den tilhørende indholdstype stadig eksisterer i systemet, dette er med vilje eftersom man kunne ønske at beholde disse oplysninger
2	Aktiver modulet under moduler	Modulet aktiveres, indholdstypen facebook_post installeres	Ingen
3	Aktiver tilhørende moduler der kræves	Brugeren bedes også om at aktivere disse under installation	Ingen
4	Opret korrekte databasetabeller	Tabeller der hører til modulet oprettes i databasen ved aktivering	Ingen

10.1.6 Test af UC6 Slet og UC9 Kommentar slet

Formål med testen:
Undersøge om UC6 og UC9 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
Noden skal være oprettet og hvis den skal slettes fra Facebook skal den være publiceret.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Slet en upubliceret node i Drupal	Noden slettes fra Drupal og fra tabellen som holder styr på dens status	Ingen
2	Slet en publiceret node i Drupal	Noden slettes fra Drupal og fra tabellen som holder styr på dens status samt alle rækker med post_id'er hørende til noden slettes	Rækker med post_id'er der hører til noden slettes ikke, dette kan medføre en database fyldt med affald.
3	Slet en publiceret node på Facebook	Noden slettes på Facebook for hver bruger den er publiceret til, hver række i post-tabellen hørende til den bruger der er slettet for, slettes, det samme med eventuelle kommentarer og status tabellen for noden eller kommentaren opdateres	Ingen
4	Slet en publiceret node på Facebook, brugeren har fjernet applikationens tilladelser	Kaldet vil give en fejl, fejlbeskeden logges og vises ved afslutning	Ingen
5	Browseren navigerer til hændelseslog ved afslutning	Link til log vises	Ingen

10.1.7 Test af UC7 Publicering

Formål med testen:
Undersøge om UC7 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
Noden skal være oprettet og for at sende kommentarer skal mindst en kommentar være tilføjet noden.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Publicer noden til Facebook	Noden publiceres for hver bruger, post_id gemmes i databasen og status opdateres	Ingen
2	Publicer kommentar til publiceret besked	Kommentaren publiceres til hver bruger, post_id gemmes i databasem og status opdateres	Ingen
3	Naviger til sending siden	Sending siden vises med informationer om antallet af sendte beskeder ud af samlet antal, tid tilbage og tid der er gået	Ingen, men opdateringen er sløv idet f.eks. tid der er gået ikke opdateres hvert sekund.
4	Publicering hvor en eller flere af brugerne har fjernet rettigheder til applikationen	Kaldet vil give en fejl, fejlbeskeden logges og vises ved afslutning, der gemmes ikke noget post_id, men status opdateres alligevel	Ingen
5	Browseren navigerer til hændelseslog ved afslutning	Link til log vises	Ingen
6	Publicering af kommentar til en node der er slettet eller ikke publiceret	Fejlbesked vises og publiceringen stoppes	Ingen
7	Vælg kommentar der skal publiceres	Dropdown boks vises, hvor kommentarer (hvis flere) vises, heraf kan en vælges som er den der publiceres	Ingen
8	Publicering af kommentar påbegyndes selvom ingen kommentarer er oprettet	Fejlbesked vises og publiceringen stoppes	Browseren navigerer til sending siden og forsøger at foretage publicering for alle brugere, Facebook returnerer dog en fejl, dette medfører at der logges fejlbeskeder for alle brugere

10.1.8 Test af UC8 Kommentarer

Formål med testen:
Undersøge om UC1 opfylder specificerede krav og reagerer som forventet.

Forudsætninger for testen:
Noden skal være oprettet og kommentarer for indholdstypen aktiveret.

Trin	Aktivitet	Forventet Resultat	Afvigelser
1	Opret kommentar	Formular til oprettelse af kommentar vises	Ingen
2	Gem kommentar (felter udfyldt)	Kommentaren gemmes og vises, kommentaren indsættes i databasen og får angivet en status som usendt	Ingen
3	Gem kommentar felterne ikke udfyldt	Fejlbesked vises	Ingen

10.2 Øvrige krav

I kravspecifikationen fremgik også en række ikke-funktionelle krav, de vil i dette afsnit blive gennemgået således at det er klart om de er opfyldt korrekt.

6. *Drupal, systemet skal kunne afvikles igennem Drupal, som et modul der kan aktiveres i Drupals oversigt over installerede moduler.*
   Der er udviklet .info og .install filer som knytter sig til dette krav og definerer at modulet vises i moduloversigten med tilhørende relevant information.

7. *MySQL, systemet skal benytte MySQL til database eftersom det er den i virksomheden benyttede databasestandard, principielt set bør systemet dog have mulighed for operation med andre standarder.*
   I modulet er brugt Drupals database abstraktionslag som giver mulighed for at bruge mange forskellige databaseteknologier, herunder også MySQL.

8. *PHP, systemet skal være skrevet i PHP, hvilket naturligvis er en konsekvens af første om afvikling i Drupal miljøet.*
   Det er ikke muligt at skrive et modul i andre sprog end PHP, så det er naturligvis opfyldt.

9. *Ikke platformspecifik, skal virke både på Windows såval som Unix servere.*
   Dette krav opfylder Drupal som standard idet PHP kan afvikles på alle gængse platforme.

10. *Fejlmeddelelser eller fejl i forbindelse med publicering bør håndteres af Drupals system til formålet.*
    Til visning af fejlbeskeder bruges altid Drupals indbyggede funktion `drupal_set_message` som

sørger for at fejlbeskeder vises på en pæn måde.

11. *Grænsefladen til systemet bør følge den generelle grænseflade og designretning brugt i Drupal, således at den virker som en naturlig del.*
    Modulet er fuldstændig afhængigt af det brugte tema i Drupal og har ingen særlige visninger udover preview-siden og sending-siden, formularer genereres derfor efter gældende standarder.

12. *Systemet skal i brugervenlighed optimeres efter en bruger med et redaktørniveau i forståelse af den generelle brug af Drupal, altså optimeret efter en bruger med et generelt kendskab til Drupals brugerflade, der tages derfor ikke nødvendigvis hånd om uerfarne brugere til Drupal.*
    Brugerfladen er forsøgt så meget som muligt at være i overensstemmelse med eksisterende interne sider, med andre ord byder brugerfladen ikke på noget der burde være overraskende eller svært at forstå for en bruger med kendskab til Drupal.

13. *For at gøre indgangsbarrieren lavest mulig bruges generelle Drupal hjælpetekster til indtastningsfelter, dette er små linjer tekst under hvert felt, der beskriver hvordan feltet benyttes.*
    Der er for så vidt muligt indsat hjælpetekster og informerende statusbeskeder i alle formularer.

10.3 Validering i Drupal

Drupal har dusinvis af regler for, hvordan kode skal sættes op og se ud, dem vil der ikke blive gået i dybden med her, det skal blot siges at der til valideringen er brugt et modul kaldet Coder, hvilket direkte kan gå ned i ens kode og se om man har fulgt de regler folkene bag Drupal foretrækker, der kodes efter. Denne validering har kun betydning for, hvor let modulet bliver at vedligeholde i fremtiden i forhold til, hvis en anden Drupal udvikler overtog udviklingen.

Resultat:

Coder found 1 projects, 4 files, 2 ***critical*** warnings, 40 ***normal*** warnings, 0 warnings were flagged to be ignored

Efter rettelser:

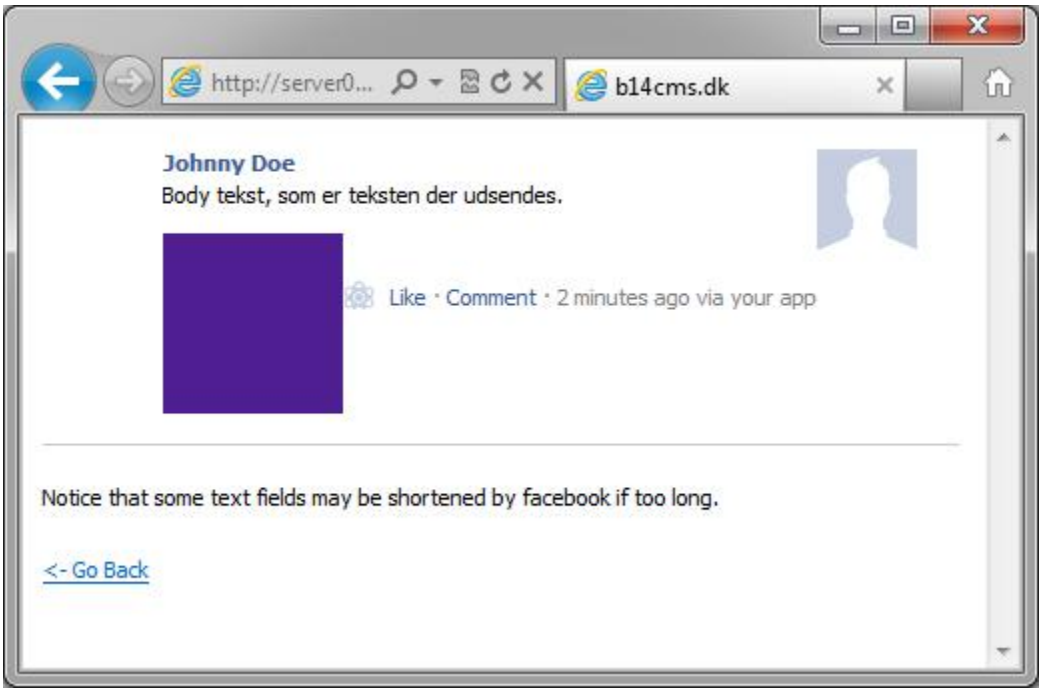
Coder found 1 projects, 4 files, 1 ***normal*** warnings, 0 warnings were flagged to be ignored.

Den eneste advarsel der er tilbage er i fb_publish_all.install.inc, fordi der mangler en beskrivelse af filen i starten, dette er dog udeladt fordi den importeres af content_copy modulet, hvortil der ikke skal angives ”<?php” begyndelses tag, dermed kan kommentarer heller ikke angives i det korrekte Drupal PHP-format, advarslen har dog ingen praktisk betydning.

De fleste af de warnings der oprindeligt var med, var manglende mellemrum mellem parenteser og tuborg-parenteser mm.

10.4 Browsertest og kompatibilitet

Der er udført en browsertest på JavaScriptet og for preview-siden i de tre store browsere Microsoft Internet Explorer version 9, Mozilla Firefox version 9 og Google Chrome version 16. Internet Explorer viser ikke preview siden helt korrekt, mens de andre gør, resultatet ses herunder:



Figur 5: Preview siden i Internet Explorer

Som det tydeligt ses er profilbilledet rykket helt til højre og informationsteksten i bunden står forkert.

JavaScriptet er også testet i alle browsere i forbindelse med publicering og resultatet blev:

- Firefox: scriptet opdaterer som det skal.
- Chrome: scriptet opdaterer som det skal.
- Internet Explorer: scriptet opdaterer som det skal.

I dette afsnit som også hedder kompatibilitet bør det nævnes at modulet udelukkende er udviklet til Drupal version 6.x og altså ikke kan forventes at virke i Drupal 7 eller den snarligt kommende version 8, dette er ikke testet, men fordi modulet i høj grad gør brug af API'et i version 6.x, er det usandsynligt at det virker i nyere versioner, hertil skal foretages en konvertering i henhold til Drupals konverteringsvejledning.

10.5 Evaluering af test

Gennemgangen af test casene fastslår at alle funktionelle krav er opfyldt til fulde, dog mangler der nogle steder lidt finpudsning i form af validering og brugervenlighed, især en bedre funktion til automatisk at hente et profil-id ved send test kunne være smart. Der var et par tilfælde hvor form valideringen slog fejl og systemet fortsatte selvom der skulle være vist en fejlbesked, i det store hele er der dog ingen alvorlige brister fundet.

Facebook har igennem projektforløbet vist sig at byde på nogle underlige begrænsninger, som har været svære at teste og implementere efter. Stimorol Local Heroes som var den oprindelige kunde produktet var tænkt til, havde adskillige tusind registrerede brugere, på alle disse blev chekUser() funktionen med stor succes lagt ind, således at systemet altid var opdateret med brugerens gældende access_token, fbu osv. Her i testfasen har det imidlertid ikke været muligt at teste modulet med så stort et antal brugere, som det er tiltænkt til, det vides derfor ikke om der opstår problemer i den forbindelse, men en simulation ved at køre en løkke og forsøge publicering til den samme bruger igen og igen viste, at køretiden for hver publicering lå på omkring 1 til 2 sekunder, dermed vil køretiden være tilsvarende lang ved mange tusinde brugere.

Netop under denne simulation dukkede en interessant begrænsning fra Facebooks side op. Efter blot 20 kald til Facebooks API begyndte systemet at få fejlbeskeder tilbage. Det viste sig imidlertid at Facebook arbejder med en dynamisk grænse for API kald inden for en given tidsperiode per bruger, grænsen er dynamisk fordi den afhænger af, hvor mange brugere der har blokeret applikationen eller markeret dens beskeder som spam, det kunne derfor forestilles at grænsen kunne blive temmelig lav.

Efter at have dykket lidt ned i denne problemstilling dukkede et mere alvorligt aspekt op, at Facebook muligvis indeholder en skjult begrænsning for, hvor mange API kald der kan foretages per dag, det har dog ikke været muligt at grave præcis dokumentation op for dette, antageligvis vil loftet være over 100.000 kald om dagen, eftersom Facebook dokumentationen angiver, at man bør kontakte dem, hvis man påtænker at køre en applikation med flere kald end dette tal om dagen.

11. Konklusion

11.1 Mål

Det oprindelige formål og den oprindelige problemformulering der blev fremsat under praktikperioden, foreskrev egentlig funktionalitet i Drupal til, at udsende private beskeder til sine brugere, det viste sig imidlertid hurtigt at Facebook slet ikke tilbød applikationer, at sende private beskeder (det der i dag også er chat), der skulle derfor tænkes i alternativer. Projektets mål blev derfor i stedet at udvikle et modul til Drupal, der kunne lægge beskeden på brugerens væg frem for i vedkommendes private indbakke. Systemet skulle fungere lidt ligesom et modul ved navn Simplenews, der bruges til at udsende nyhedsbreve til e-mail adresser der abonnerer, så dertil skulle altså bruges en form for system der kunne gå alle brugere i Drupal igennem og sende en besked til hver enkelt. Efter en analyse af mulighederne stillet til rådighed af Facebooks API blev det også til en række funktionalitetskrav omkring kommentarer og sletning, samt nogle praktiske funktioner indbygget i Drupal.

Som det også fremgår af testafsnittet bliver der delvist konkluderet på disse mål, hvor også en række ukendte faktorer i løbet af projektforløbet har spillet ind. Overordnet set er alle mål nået og produktet opfylder faktisk mere end blot det oprindelige mål om publicering, idet funktionerne om kommentarer og sletning fra Facebook først var noget der kom under idéudviklingen i opstartsfasen. Det oprindelige mål fra kunden er dermed egentlig ikke opfyldt, men der er fundet et rimeligt alternativ og alligevel kommet et yderst funktionelt produkt ud af projektet, som endda koger videre på de oprindelige tanker.

11.2 Projektet

Projektet har kun i nogen grad været risikostyret som spiralmodellen ellers foreskriver, problemet har været at risikoanalysens identificerede risici er en anelse ukonkrete eller for abstrakte til at forebygge imod, dog har vurderingen af risicienes sandsynlighed vist sig at være nogenlunde korrekt, idet i al fald de højest vurderede forekom, og den kritiske op til flere gange i løbet af projektperioden. Den mest kritisk vurderede var pludselige opdateringer af Facebooks API, som der til stadighed bliver udviklet på, denne blev en vigtig del i modellen, fordi det blev kontrolleret om nyeste prototype levede op til det gældende API på pågældende tidspunkt, hvor det faktisk viste sig et par gange, at kode måtte skrives om, fordi Facebooks API netop var opdateret til en nyere version, og dermed havde gjort det brugte utidssvarende, dette gjaldt bl.a. hvornår og hvor der skulle bruges access_token.

Det endelige modul, for nu, virker som det skal og er implementeret efter gældende standarder, så det skulle være lettere at vedligeholde i fremtiden. Alle stillede krav er opfyldt selvom resultatet af dette projekt blev et alternativ til den funktionalitet den oprindelige kunde, Local Heroes, ønskede og faktisk ikke overholder ønsket om at sende beskeder til den private indbakke på Facebook, men det er som sagt endnu en begrænsning i Facebook platformen.

Inden faktisk deployering bør de sidste hager rettes, det drejer som om alle former for validering inden kald til Facebook, da kald har en væsentlig længere svartid end intern validering tager. Der bør også foretages en række ændringer så databasen ikke kan fyldes op med affaldsdata, dette må dog komme i en fremtidig udgave.

11.3 Etik

Under udviklingsfasen stod modulets styrke hurtig klar, her følger et kort eksempel: Et firma opretter en konkurrence som benytter en applikation med modulet slået til, for at være med i konkurrencen, skal Facebook brugeren gå ind på applikationens hjemmeside, hvor vedkommende her vil blive bedt om at acceptere de rettigheder (extended permissions) applikationen forlanger. Herefter vil firmaet have nærmest ubegrænset mulighed for at udsende beskeder på vegne at brugeren, dette er et meget stærkt reklameredskab idet:

1. Konkurrence incitamentet vil givetvis lokke et stort antal brugere til.
2. Når en besked publiceres vil alle brugerens venner kunne se den, hvis det antages at hver bruger har 130 venner, som er et gennemsnitstal der flere gange har været oppe i medierne, og der til applikationen i alt er 1000 tilmeldte giver det potentielt set en mængde på 130.000 modtagere.
3. Dertil skal lægges at beskeder kan kamufleres, så det ser ud som om at det er brugeren selv der har delt indholdet.

Formålet med projektet har naturligvis ikke været at muliggøre denne form for reklame, eftersom det oprindeligt var tiltænkt som et system til at informere en tilmeldt bruger om nyheder i forbindelse med Local Heroes konkurrencen.

Ifølge gældende dansk lov er det ikke tilladt at sende reklame uden at modtageren har givet tilladelse dertil, dette er noget modulet i givet fald ville kunne bruges til, som udgangspunkt må man dog gå ud fra at et professionelt firma altid oplyser brugerne om, hvad det er de siger ja til. Dette vil sædvanligvis være noget man gør opmærksom på under tilmelding.

En ting er gældende dansk lov, der overholdes blot ved at informere korrekt om hvad der siges ja til, en anden ting er den nugældende officielle politik på Facebook. I Facebook Platform Policies[23] står der blandt andet:

- You must not include functionality that proxies, requests or collects Facebook usernames or passwords.
- You will have a privacy policy that tells users what user data you are going to use and how you will use, display, share, or transfer that data and you will include your privacy policy URL in the Developer Application.
- Subject to certain restrictions, including on transfer, users give you their basic account information when they connect with your application. For all other data obtained through use of the Facebook API, you must obtain explicit consent from the user who provided the data to us before using it for any purpose other than displaying it back to the user on your application.
- You will delete all data you receive from us concerning a user if the user asks you to do so, and will provide an easily accessible mechanism for users to make such a request. We may require you to delete data you receive from the Facebook API if you violate our terms.
- You must not present users with the Feed form or publish a Stream story unless a user has explicitly indicated an intention to share that content, by clicking a button or checking a box that clearly explains their content will be shared.

[23] http://developers.facebook.com/policy/

Især sidste statement, der siger at man ikke må foretage en publicering, der ikke er igangsat af brugeren selv, strider imod det koncept projektet her understøtter, så i princippet er Facebook i sin gode ret til at blokere den applikation der bruges igennem modulet. Samtidig bør en række af de andre punkter også overholdes i forbindelse med deployering.

Mange firmaer benytter sig på nuværende tidspunkt også af muligheden for at bede brugerne, at synes godt om-firmaet før de kan deltage i en konkurrence, som lidt lå i kortene i det tænkte eksempel, netop dette har forbrugerombudsmanden fremsat forslag om at forbyde[24], dette kunne være endnu en lovmæssig hindring, hvis den vedtages.

11.4 Perspektiver

Det store mål med projektet i fremtiden vil helt klart være at få det lagt op på Drupal.org, så andre også kan få glæde af det, det ligger lidt i Open Source mentaliteten. Før dette kan ske, skal de sidste fejl selvfølgelig udbedres. Testen med et stort antal brugere kunne også lettere udføres, hvis det blev lagt ud på Drupal.org, fordi, i sin kraft af et stort community, ville give betydelig bedre feedback end det nogensinde ville være muligt i en lukket test.

At lægge det ud offentligt kræver at modulet vedligeholdes, som nævnt har Facebook APIets jævnlige opdateringer i forløbet voldet nogle problemer, derfor skal der holdes øje netop med disse opdateringer. Selvom det kan volde problemer, kan det også åbne nye muligheder, når ny funktionalitet bliver tilføjet til deres API, man kunne godt forestille sig, at det inden for en overskuelig tidsramme blev muligt at poste beskeder som en applikation eller en side, i stedet for kun på vegne af en bruger.

Der er naturligvis allerede nogle interessante funktioner der kunne bygges ind i produktet, som ikke er bygget ind nu. Facebook understøtter at sætte modtagerindstillinger for en publiceret besked, således at det kun er venner, kun familie, kun nære venner eller lignende, der kan se beskeden på væggen, så nærmer funktionaliteten sig det oprindelige mål.

Hvis der er mange brugere i systemet, vil køretiden muligvis blive ret lang, derfor kunne det også være smart med en funktion til at sætte publiceringen på pause og starte igen senere. Dette er noget der er bygget ind i Simplenews mail modulet og kunne med fordel overføres.

11.5 Evaluering,

Projektet er med stor succes udført i samarbejde med B14, der har været en stor støtte igennem processen og givet ro til at arbejde, selvom det har været travlt i firmaet. Det peronslige mål med projektet var at lære om moduludvikling til Drupal i PHP og undersøge mulighederne i Facebooks API, jeg blev meget overrasket over sidstnævnte, hvor meget information man kunne hente ud om den enkelte bruger forholdsvis nemt.

Forinden havde jeg i praktikperioden, kun lige lært Drupal, ved at supportere på forskellige hjemmesider med Drupal som back end. Ved projektets begyndelse havde jeg derfor kun overfladisk kendskab til,

[24] http://nordjyske.dk/artikel/10/5/4/4022183/3/konkurrencer-p%E5-facebook-er-m%E5ske-fortid

hvordan Drupal fungerede, vidste ikke hvordan et modul implementeres og havde kun i forvejen ringe erfaring med PHP udvikling. Især hele aktiviteten omkring publicering, hvor der skulle indbygges ret mange temmelig lange funktioner, var noget der tog tid at få implementeret. Interessant var også at lære Drupals arkitektur og læse om hvordan det er processuelt men samtidig indbygger en lang række begreber fra objektorienteret programmering.

Som det også blev nævnt i testafsnittets evaluering, opstod også en række ukendte begrænsninger, der var en anden identificeret risiko, hvor den ene af disse begrænsninger især var problematisk. En begrænsning lyder at der maksimum kan foretages omkring 20 kald per bruger per dag, eftersom jeg kun testede én bruger under testfasen, blev denne grænse hurtigt nået med alle de test cases der blev gennemgået, så noget af testen måtte skydes til næste dag.

Tidsplanen viste sig at holde ret præcist, som det er vedlagt i en note i dens afsnit, blev kun nogle få ting rykket pga. arbejde og sygdom. Rapporten har også været vigtig at få startet og halvejs i forløbet, var jeg også cirka halvejs med rapporten.

Hvis der var noget, jeg ønskede gjort anderledes i projektforløbet, skulle det nok være at produktet lagde sig tættere op ad Simplenews modulet, hvor man også får den pæne progress bar under udsendelse. Simplenews byder også på den mulighed at sætte udsendelsen på pause, hvor der i send formularen så vises, hvor mange mails der mangler at blive sendt. I simplenews modulet foregår udsendelsen også ved at mails fra dens spole opdeles i batches af f.eks. 20 mails som udsendes på en gang, det er en smart optimering, der også kunne være fordelagtig at benytte.

12. Appendiks

12.1 Indhold på CD

Cd'en indeholder tre mapper:

- Diagrammer, som indeholder Visio filer til alle diagrammer brugt i rapporten.
- Installationsfiler, som indeholder alle filer nødvendige for installation.
    - Herunder tredjepartsmoduler der skal installeres sammen med Drupal
- Kildekode, som indeholder de filer der udviklet i forbindelse med projektet.
- Derudover kan tidsplan og rapporten findes på Cd'en.

12.2 Installationsvejledning

Herunder findes en komplet vejledning til hvordan hele systemet installeres lokalt eller på nettet, hvis man ikke ønsker at køre en lokal udgave kan en præ-installeret opsætning findes på denne adresse: http://server003.b14cms.dk/users/dev/cw6/www til log ind kan bruges brugeren "test" med password "test".

Installation af webserver:

1. Gå på Cd'en ind i mappen installationsfiler.
2. Udpak filen mowes_portable.zip (hvis du allerede har en webserver med PHP og MySQL kan du springe dette over og gå til Drupal opsætningen).
3. Åbn mowes.exe og følg vejledningen på skærmen.
4. Når alt er installeret skulle du gerne se en startside, hvis ikke så naviger til siden: http://127.0.0.1/start/index.php
5. Stop serveren og gå ind i mappen Mowes portable\php5 og åbn filen php.ini, på linje 956 skal semikolon foran denne linje fjernes: extension=php_curl.dll
6. Lokaliser i samme mappe filerne ssleay32.dll og libeay32.dll og kopier dem til apache2/bin folderen.

Drupal:

1. Det er nødvendigt at oprette en database til Drupal før drupal installeres, åbn derfor phpMyAdmin der ligger på adressen http://127.0.0.1/phpmyadmin/ i Mowes, vælg fanen "Databaser" og opret en ny ved navn "drupal".
2. Udpak filen drupal-6.22.zip til roden i din webserver, i Mowes er det folderen www.
3. Åbn INSTALL.txt i den netop udpakkede mappe og følg vedledning deri, hvis den ser underlig ud så brug et program som notepad++ til at åbne den.
4. I settings.php filen som skal laves som noget af det første indtastes under database settings: $db_url = 'mysql://root@127.0.0.1/drupal';
5. I mappen www under Mowes omdøb mappen drupal-6.22 til "drupal".
6. Åbn din browser og naviger til http://127.0.0.1/drupal/install.php
7. Følg vejledningen på skærmen.
8. Opret et navn til din side og en administrator bruger, det er ligemeget hvad du vælger her.

9. Du vil muligvis se om fejl om at mail() ikke virker, se bort fra dette, vi har ikke behov for at sende emails igennem Drupal.

Moduler:

1. Gå igen ind på Cd'en under installationsfiler og dernæst i mappen Moduler til Drupal, gå til den mappe Drupal er installeret i og dernæst ind i mappen "modules" udpak alle zip-filerne i modules mappen.
2. gå til http://127.0.0.1/drupal/admin/build/modules (Administer -> Site Building -> Modules)
3. Aktiver følgende moduler:
    a. **Administration menu** (Giver en sort menu i toppen som gør det nemmere at navigere)
    b. **Content**
    c. **Content Copy**
    d. **Fieldgroup**
    e. **FileField**
    f. **ImageField**
    g. **Text**
    h. **Facebook API**
    i. **Facebook Apps**
    j. **Facebook Connect**
    k. **FB User Management**
    l. **FB Extended Permissions**
    m. **FB User Tracking**
    n. **Maxlength**
4. Når alle ovenstående moduler er aktiveret, aktiver da modulet **FB Publish All.**
5. Drupal for Facebook modulet skal nu sættes op, gå ind i mappen hvor modulet er installeret, dette vil typisk være Mowes portable\www\drupal\modules\fb og åbn filen README.txt, følg nøje vejledningen deri. Det er vigtigt koden i bunden af README.txt tilføjes til settings.php.
6. Ændr denne kode:

```
// Enable URL rewriting (for canvas page apps).

include "sites/all/modules/fb/fb_url_rewrite.inc";
include "sites/all/modules/fb/fb_settings.inc";
```

Til dette:

```
include "modules/fb/fb_url_rewrite.inc";
include "modules/fb/fb_settings.inc";
```

7. Gå ind i mappen Mowes portable\www\drupal\themes\garland og åbn filen page.tpl.php.
   I <html >-tagget i toppen indsæt følgende linje kode:
   xmlns:fb=http://www.facebook.com/2008/fbml
8. Udpak Facebook PHP SDK som det fremgår af FB modulets README, det er vedlagt på Cd'en og hedder facebook-php-sdk-v3.1.1.zip, omdøb om nødvendigt mappen til "facebook-php-sdk".

9. Indsæt følgende linje kode i bunden af settings.php:
   ```
   $conf['fb_api_file'] = 'sites/all/libraries/facebook-php-sdk/src/facebook.php';
   ```
10. Nu skulle FB modulet gerne være køreklar.

Opsætning af modulet:

1. Naviger til http://127.0.0.1/drupal/admin/build/fb/fb_app_create (Administer -> Site Building -> Facebook Applications) og tilføj en ny applikation
2. Følg vejledningen under oprettelse og opret også en Applikation på Facebook der kan bruges til publicering. Under Facebook connect i bunden vælges "Primary".
3. Gem applikationen.
4. Når den er gemt tryk Edit for at gå ind og sætte de extended permissions der skal bruges, dette er følgende under "Required extended permission:
   a. News Feed & Wall - access my News Feed & Wall (read_stream).
   b. Offline Access - access my data when I'm not using the application (offline_access)
   c. Publish to my Wall - publish content to my Wall (publish_stream)
5. Tryk gem.
   Alt skulle nu gerne være installeret og sat op.

12.3 Brugervejledning

For at oprette en ny Facebook post node gå ind under Create content -> Facebook post og udfyld felterne.

Når den er oprettet vises den gemte node.

Menuen i toppen a noden indeholder de muligheder der ligger i grænsefladen, der henvises til de relevante afsnit omkring hvordan hvert punkt virker.

Log ind:
Når en bruger skal oprettes i systemet, skal denne logge ind via Facebook connect, der skulle gerne være en blå knap til dette på forsiden, hvis det er sat korrekt op, hovedsiden vil givetvis være:
http://127.0.0.1/drupal

Hvis der i stedet for denne knap bare står "connect" med tekst, betyder det at modulet Drupal for Facebook ikke er opdateret til nyeste version, eller at Facebook PHP SDK'et er forældet, i de fleste tilfælde er det første der er gældende. I det tilfælde skal nyeste version downloades fra drupal.org, slet den gamle mappe over sæt den opdaterede ind samme sted, hvor alle moduler ligger.

12.4 Risikoanalyse

De udvidede tabeller der er beregningsgrundlag for risikoanalysen.

Risiko: sygdom				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
5	9	2	1	4,25
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
4	1	4	2	5,5
			Total gennemsnit:	4,875
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
1	1	1	6	2,25
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
2	2	1	1	1,5
			Total gennemsnit:	1,875
			Risiko:	3,375

Risiko: uforudset arbejde				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
7	7	8	8	7,5
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
6	9	9	8	8
			Total gennemsnit:	7,75
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
1	1	1	1	1
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
1	1	1	1	1
			Total gennemsnit:	1
			Risiko:	4,375

Risiko: skred i tidsplan				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
5	5	5	4	4,75
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
4	5	4	4	4,25
			Total gennemsnit:	4,5
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
2	1	1	3	1,75
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
1	1	1	1	1
			Total gennemsnit:	1,375
			Risiko:	2,9375

Risiko: Internetnedbrud				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
2	1	2	1	1,5
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
2	2	2	1	1,75
			Total gennemsnit:	1,625
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
2	5	5	3	3,75
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
6	3	2	2	3,25
			Total gennemsnit:	3,5
			Risiko:	2,5625

Risiko: Systemnedbrud				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
2	4	2	3	2,75
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
2	2	2	3	2,25
			Total gennemsnit:	2,5
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
6	7	7	7	6,75
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
6	3	4	8	5,25
			Total gennemsnit:	6
			Risiko:	4,25

Risiko: Facebook opdateringer				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
5	6	6	6	5,75
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
6	7	8	5	6,5
			Total gennemsnit:	6,125
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
4	9	9	5	6,75
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
9	7	6	3	6,25
			Total gennemsnit:	6,5
			Risiko:	6,3125

Risiko: Drupal opdateringer				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
4	5	6	5	5
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
3	6	4	1	3,5
			Total gennemsnit:	4,25
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
6	5	2	2	3,75
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
3	3	4	2	3
			Total gennemsnit:	3,375
			Risiko:	3,8125

Risiko: Ukendte begrænsninger				
Trussel faktorer				
Evner	Motiv	Mulighed	Størrelse	Gennemsnit
2	3	5	4	3,5
Sårbarhedsfaktorer				
Opdagelse	Udnyttelse	Klarhed	Detektion	Gennmsnit
4	3	8	8	5,75
			Total gennemsnit:	4,625
Tekniske omkostninger				
Fortrolighed	Integritet	Tilgængelighed	Ansvar	Gennemsnit
2	6	8	6	5,5
Forretningsmæssige omkostninger				
Finansiel skade	Omdømme	Compliance	Persondata	Gennemsnit
9	8	6	3	6,5
			Total gennemsnit:	6
			Risiko:	5,3125

12.5 Liste over extended permissions

Komplet liste over de udvidede rettigheder en applikation kan foretage en forespørgsel på igennem Facebook, for en opdateret liste se her: http://developers.facebook.com/docs/reference/api/permissions/

User and Friends Permissions

User permission	Friends permission	Description
user_about_me	friends_about_me	Provides access to the "About Me" section of the profile in the about property
user_activities	friends_activities	Provides access to the user's list of activities as the activities connection
user_birthday	friends_birthday	Provides access to the birthday with year as the birthday_date property
user_checkins	friends_checkins	Provides read access to the authorized user's check-ins or a friend's check-ins that the user can see.
user_education_history	friends_education_history	Provides access to education history as the education property
user_events	friends_events	Provides access to the list of events the user is attending as the events connection
user_groups	friends_groups	Provides access to the list of groups the user is a member of as the groups connection
user_hometown	friends_hometown	Provides access to the user's hometown in the hometown property
user_interests	friends_interests	Provides access to the user's list of interests as the interests connection
user_likes	friends_likes	Provides access to the list of all of the pages the user has liked as the likes connection
user_location	friends_location	Provides access to the user's current location as the location property
user_notes	friends_notes	Provides access to the user's notes as the notes connection
user_online_presence	friends_online_presence	Provides access to the user's online/offline presence
user_photos	friends_photos	Provides access to the photos the user has uploaded, and photos the user has been tagged in
user_relationships	friends_relationships	Provides access to the user's family and personal relationships and relationship status
user_relationship_details	friends_relationship_details	Provides access to the user's relationship preferences
user_religion_politics	friends_religion_politics	Provides access to the user's religious and political affiliations

user_status	friends_status	Provides access to the user's most recent status message
user_videos	friends_videos	Provides access to the videos the user has uploaded, and videos the user has been tagged in
user_website	friends_website	Provides access to the user's web site URL
user_work_history	friends_work_history	Provides access to work history as the work property
email	(ikke tilgængelig)	Provides access to the user's primary email address in the email property.

Extended Permissions

Permission	Description
read_friendlists	Provides access to any friend lists the user created. All user's friends are provided as part of basic data, this extended permission grants access to the lists of friends a user has created, and should only be requested if your application utilizes lists of friends.
read_insights	Provides read access to the Insights data for pages, applications, and domains the user owns.
read_mailbox	Provides the ability to read from a user's Facebook Inbox.
read_requests	Provides read access to the user's friend requests
read_stream	Provides access to all the posts in the user's News Feed and enables your application to perform searches against the user's News Feed
xmpp_login	Provides applications that integrate with Facebook Chat the ability to log in users.
ads_management	Provides the ability to manage ads and call the Facebook Ads API on behalf of a user.
create_event	Enables your application to create and modify events on the user's behalf
manage_friendlists	Enables your app to create and edit the user's friend lists.
manage_notifications	Enables your app to read notifications and mark them as read.
offline_access	Enables your app to perform authorized requests on behalf of the user at any time. By default, most access tokens expire after a short time period to ensure applications only make requests on behalf of the user when the are actively using the application. This permission makes the access token returned by our OAuth endpoint long-lived.

publish_checkins	Enables your app to perform checkins on behalf of the user.
publish_stream	Enables your app to post content, comments, and likes to a user's stream and to the streams of the user's friends. With this permission, you can publish content to a user's feed at any time, without requiring offline_access. However, please note that Facebook recommends a user-initiated sharing model.
rsvp_event	Enables your application to RSVP to events on the user's behalf
sms	Enables your application to send messages to the user and respond to messages from the user via text message
publish_actions	Enables your application to publish user scores and achievements.

Page Permission

Permission	Description
manage_pages	Enables your application to retrieve access_tokens for pages the user administrates. The access tokens can be queried using the "accounts" connection in the Graph API. This permission is only compatible with the Graph API.

12.6 Litteraturliste

Dette er en samling over alle eksterne referencer i rapporten.

1. http://da.wikipedia.org/wiki/Unified_Process
2. http://www.waterfall-model.com/
3. http://en.wikipedia.org/wiki/Spiral_model
4. http://w3techs.com/technologies/overview/content_management/all
5. http://drupal.org/
6. http://developers.facebook.com
7. http://oauth.net/
8. http://developers.facebook.com/docs/reference/api/permissions/
9. http://developers.facebook.com/docs/guides/canvas/#canvas
10. http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems
11. http://www.gatherspace.com/static/use_case_example.html
12. http://drupal.org/project/fridge
13. http://drupal.org/project/simple_fbwall
14. http://developers.facebook.com/docs/reference/plugins/like/
15. http://drupal.org/project/simplenews
16. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology
17. http://venturebeat.com/2012/01/05/facebook-actions/
18. http://api.drupal.org/api/drupal/developer--hooks--core.php/function/hook_menu/6
19. http://drupal.org/node/751826
20. http://developers.facebook.com/policy/
21. http://nordjyske.dk/artikel/10/5/4/4022183/3/konkurrencer-p%E5-facebook-er-m%E5ske-fortid