Formål

Der skal laves et website til brugerdefineret semi-struktureret materiale. Semi-struktureret er i denne sammenhæng fritekst, som bliver beriget med klart afgrænsede entiteter. For at konkretisere systemet vælger jeg at lave en madopskriftsamling, hvor man har ingredienser og mængder som afgrænsede entiteter, og en fremgangsmåde i fritekst. Derudover skal det være muligt at berige opskrifter med tags. Alle brugere skal frit kunne oprette og editere i opskrifter, samt give allerede eksisterende opskrifter rating og tags. Man skal kunne melde sig til som bruger af systemet, og herefter logge ind. Indloggede brugere kan vælge at gemme opskrifter i en favoritliste.

Begrundelse

Mange af de opskriftssamlinger som man finder på nettet i dag er præget af lukkethed. Almindelige brugere er begrænset til at læse opskrifter medmindre de gennemgår en flertrinsprocess, for at blive bruger. Dette medfører helt naturligt at der er få som publiserer deres opskrifter. Da rigtig mange mennesker ligger inde med spændende opskrifter og næsten alle har en mening om hvordan man bedst kan tilberede sine råvarer, mener jeg at denne lukkede fremgangsmåde ikke er optimal. Alle ville få meget større udbytte hvis opskriftssiderne var mere Wiki lignende med skriveadgang til alle. Derudover kan man gøre søgninger i opskrifter mere præcise ved at tildele sigende tags. Igen skal det være almindelige brugere som skal tildele tags til opskrifter, således at kvaliteten af tildelte tags øges med antallet af brugere som har tildelt tagget. Endelig kan det være en fordel at kunne tildele tags til andre tags, således at også tags kan blive beskrevet nærmere. I en fremtidig udgave ville man derved kunne finde opskrifter, som hverken indeholder et ord, eller et specifikt tag, men derimod tags som beskriver de tags som er tilknyttet opskriften. Eller at man kan søge i tags, og finde relaterede tags eller synonymer.

Afgrænsning

Siden sitet er helt åbent for alle brugere at rette i opskrifter, og potentielt oprette en masse nye opskrifter med tvivlsomt indhold, er det vigtigt at man let kan rydde op i indholdet, og at man i høj grad kan filtrere uønsket materiale fra, før det kommer ind i systemet. Oprydningen i indhold kan løses ved at gemme alle versioner af alt indhold, og lade brugerne fravælge upassende versioner. Spamfiltre er noget mere kompliceret at implementere. I dette projekt er der hverken tid til versionering eller spamfiltre.

I alle systemer hvor det tillades brugere at indtaste oplysninger er det vigtigt at man er opmærksom på at sitet før eller siden får opmærksomhed af brugere med urene hensigter. For eksempel er det ret simpelt at helt slette tabeller i databasen igennem et et søgefelt på en side[1], hvis ikke man eksplisit behandler de input, som man modtager fra brugere. Desuden bør et site have politikker for længde af brugernavne og kompleksitet af passwords mm. men alt dette afgrænser jeg mig fra i dette system.

[1] SQL injection

Jeg afgrænser mig også fra at lave et afpudset design. Designet skal være simpelt og funktionelt, men jeg mener at leg med farver, ikoner, baggrunde er mindre interessant i forhold til dette projekt end selve funktionaliteten.

Krav

Systemet skal opfylde følgende krav:

Opskrifter

- Skal kunne oprette opskrifter
- Skal kunne ændre i opskrifter
- Skal kunne søge i opskrifter
    - Søgninger skal kunne udføres i opskrifters titel, beskrivelse og tekst.
    - Søgninger skal kunne udføres i tags, som er tilknyttet opskrifter.
    - Søgninger skal kunne udføres på ingredienser
- Skal kunne give opskrifter en rating

Tags

- Skal kunne sætte tags på opskrifter
- Skal kunne sætte tags på andre tags

Brugere

- Man skal kunne oprette sig som bruger på sitet ved at angive et brugernavn og adgangskode.
- Man skal kunne logge ind på sistet med brugernavn og adgangskode.
- Indloggede brugere skal kunne vælge at gemme opskrifter som favoritter

Analyse

Da der er krav om at brugere skal kunne indtaste forskellige typer information er det nærliggende at dele sitet op i en mængde sider, hvor hver side udfører sin egen lille del af det samlede system. Siderne skal så på en eller anden vis linkes sammen i en så overskuelig sitemap som muligt.

Jeg har dog valgt i dette projekt at eksperimentere lidt, og vælger istedet at kun have én side hvor det istedet er sidens enkeltdele, som tilpasses situationen. Dette kan naturligvis have den ulempe at der bliver for mange valgmuligheder samtidig, som kan gøre at brugeren mister overblikket. Derudover skal opsætningen være simpel med menu og søgning øverst og indholdet i et traditionelt 3-kolonne design.

Systemet bliver lavet i en kombination af php og mySql, som er velegnet til webudvikling. Netop webbens udbredelse og tilgængelighed er en grundlæggende forudsætning for sådan et system,

hvor det er afgørende at enhver kan deltage i at opbygge en fælles informationsmængde til glæde for alle.

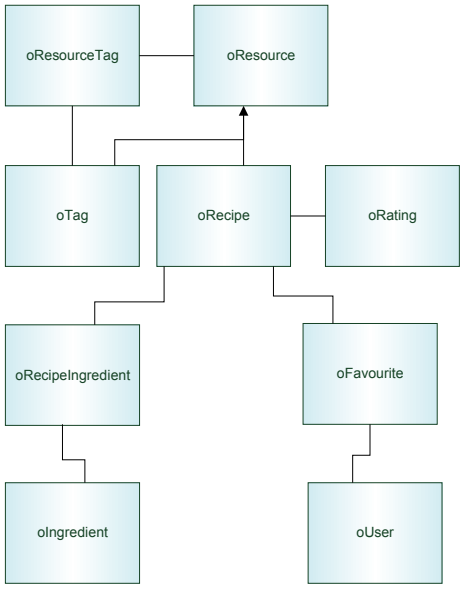
Datamodel

Datamodellen på Figur 1 ligner entitetsmodellen, som blev sat op i den oprindelige systembeskrivelse. Tags skal kunne sættes på både opskrifter og på andre tags. Jeg har derfor valgt at lave en fælles tabel Resource, som Tag og Recipe nedarver fra[2]. Der skal derfor kun laves et sæt funktioner for at tagge resourcer istedet for at tagge hendholdsvis tags og opskrifter. En Resource kan have flere Tags og et Tag kan være tilknyttet flere Resourcer. Derfor er der en tabel mellem Tag og Resource som bryder denne mange-til-mange forbindelse op i to en-til-mange forbindelser.

I forhold til entitetsmodellen i den oprindelige systembeskrivelse, er der nu kun en tabel til tags, men oTag tabellen har istedet fået en såkaldt diskriminator – et type felt, som siger hvad slags tag der er tale om.

Til en opskrift kan der tilknyttes en rating. Ingredienser skal ligesom Tags kun oprettes engang, og derefter tilknyttes opskrifter. Derfor er der en RecipeIngredient tabel mellem Recipe og Ingredient.

Brugere registreres i User tabellen og kan gemme referencer til opskrifter i Favourite tabellen sådan at de nemt kan findes frem igen. Her er det igen gældende at en bruger kan vælge flere opskrifter og at en opskrift kan være valgt af flere brugere, deraf Favourite tabellen mellem de to.



Figur 1 Datamodel

Der er lavet foreign key constraints alle steder hvor "data hænger sammen"[3]. Dette er ikke strengt nødvendig, men sikrer data integriteten ved at man ikke kommer til at f.eks. oprette en Recipe uden først at have en Resource.

De database scripts som blev brugt til at generere tabellerne ligger i er vedlagt i en zip fil sammen med php filerne. Jeg har præfikset alle tabelnavne med o (så de hedder oIngredient, oRecipe osv.), alene for at lettere kunne adskille tabellerne i dette projekt fra tabellerne fra andre projekter.

Arkitektur

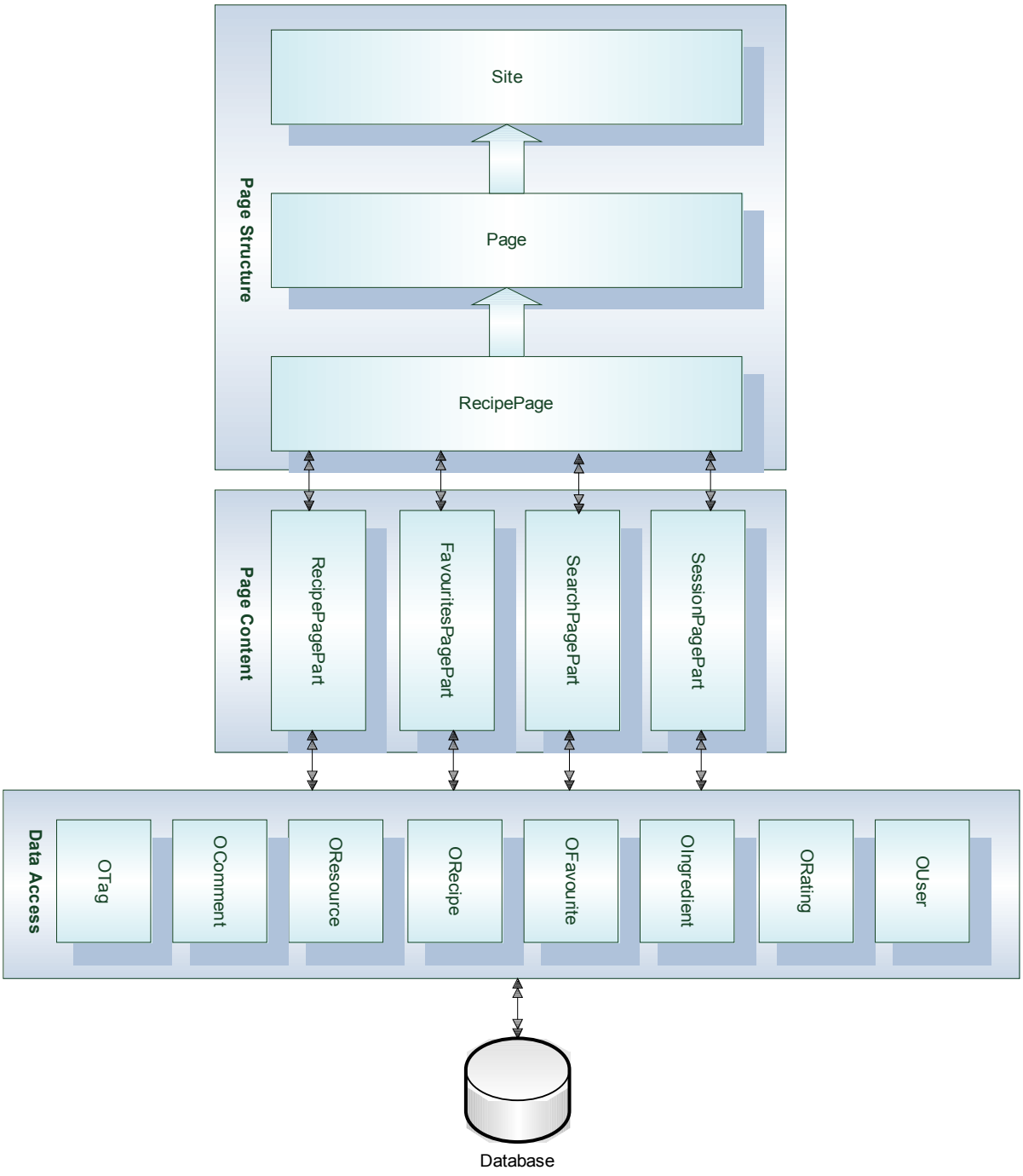
Jeg har valgt at opdele systemet i tre hoved lag:

- Page Structure laget indeholder for det meste statiske oplysninger såsom sitets header, footer og hovedstrukturen, i dette tilfælde en side med en menu øverst og nedenunder den er der tre kolonner til at vise materiale.

[2] Arv findes ikke direkte i mySql, men kan simuleres. I mit tilfælde sker det ved at lade primærnøglen i Tag og recipe være foreign key til Resource's primærnøgle. Men man skal selv lave joins og indsætte i "basis" tabellen før de nedarvede tabeller. Andre databaser, f.eks. PostgreSQL understøtter arv direkte.
[3] For at kunne bruge foreign key constraints er INNODB valgt som database engine.

- Page Content laget behandler alt dynamisk indhold fra Data Acces laget, kombinerer det med valg fra brugeren, og sørger for at rendere oplsningerne til passende html.

- Data Access laget er mellem Page Content laget og databasen og alle informationer, som bliver hentet af basen eller skrevet til basen skal gennem dette lag.



Figur 2 Systemets arkitektur

Indenfor hvert lag er systemet igen opdelt i mindre enheder.

Page Structure

Inspireret af Hudsons "A basic OOP site" [Hudson, kap 6.19] har jeg lavet en Site basisklasse i Page Structure laget som kun indeholder header, footer[4] og så en variabel til indholdet derimellem. Hudson lader sin Site klasse kende til Page klassen, som nedarver fra Site (feks. i setPage metoden). Jeg synes ikke dette er godt design, så jeg har valgt at min Site klasse ikke skal have noget som helst

[4] Header og footer indeholder HTML deklarationen, og header indeholder derudover sidens titel og en reference til et stylestheet.

kendskab til de klasser som måtte nedarve fra den. Site klassen indeholder en render metode som bør være det eneste sted i hele koden hvor der bliver udskrevet HTML (med echo eller print funktionerne). Det giver den fordel at man har bedre kontrol over det output, som skal sendes til browseren, man kan f.eks. zippe indholdet først. Det medfører også at man kan sætte cookies og bruge andre funktioner, som kræver at der ikke er sendt noget til browseren før de bliver kaldt, helt indtil render metoden er kaldt. Dette svarer lidt til at sætte en output buffer i Site klasen, som opfanger alt output indtil bufferen bliver flushet.

Page klassen nedarver fra Site, og meningen er at alle konkrete sider i systemet skal nedarve fra Page. Udover at sætte sidens titel ligger der i øjeblikket ikke noget funktionalitet i Site klassen, men man kunne overveje om det grundlæggende 3-kolonne design skulle ligge her, fremfor at ligge i de klasser som nedarver fra Page.

I øjeblikket er Recipe Page den eneste klasse som nedarver fra Page. Eftersom indholdet på siden er rimelig ensartet (opskrifter, søgning og brugerlogin), synes jeg at det fungerer at have alt på en side, men i takt med at sådan et site vokser, vil det blive nødvendig at have forskellige sider. Naturligvis er det først når man har flere sider at man virkelig får udnyttet mulighederne i arv, således at man ikke skal implementere f.eks. render metoden i hver side klasse. Som nævnt før kunne det her også være praktisk hvis selve 3-kolonne strukturen lå i Page klassen, såfremt alle sider skal have 3 kolonner.

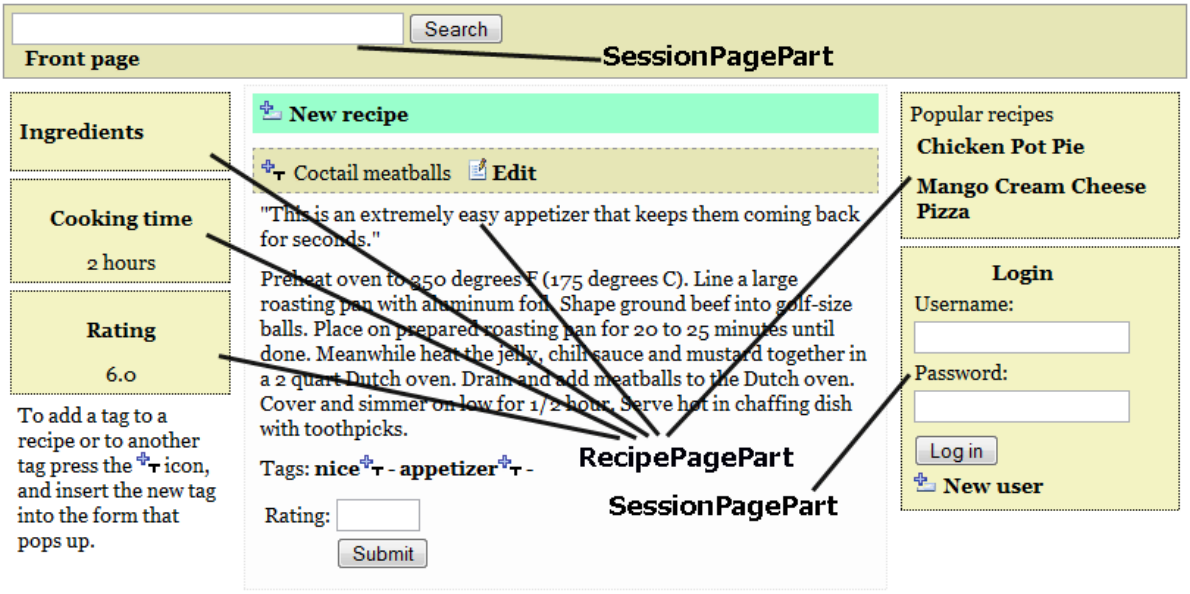
Recipe Page initialiserer alle de Page Parts (beskrevet under kapitlet Page Content nedenunder) som siden skal indeholde. Det er altså Recipe Page som samler alle trådene fra de underliggende klasser.

Page Content

Page Content laget i systemet består af PageParts som har til opgave at indkapsle dele af siden i mindre sammenhængende enheder, som kan have en livssyklus uafhængig af hinanden. Der er altså ingen kommunikation imellem Page Parts.

For at gøre brugen af Page Parts så ensartet som muligt er der lavet et interface for alle PageParts, som specificerer tre funktioner som alle PageParts skal implementere:

- initialize – Initierer Page Partens værdier
- getPagePartKey – Returnerer en nøgle som er unik for hver Page Part
- handlePostBack – Funktion, som lader hver Page Part håndtere de forms som er genereret fra Page Parten. Siden, som Page Parts ligger på kalder den relevante Page Parts handlePostBack metode ud fra den PagePartKey, som getPagePartKey funktionen returnerer.



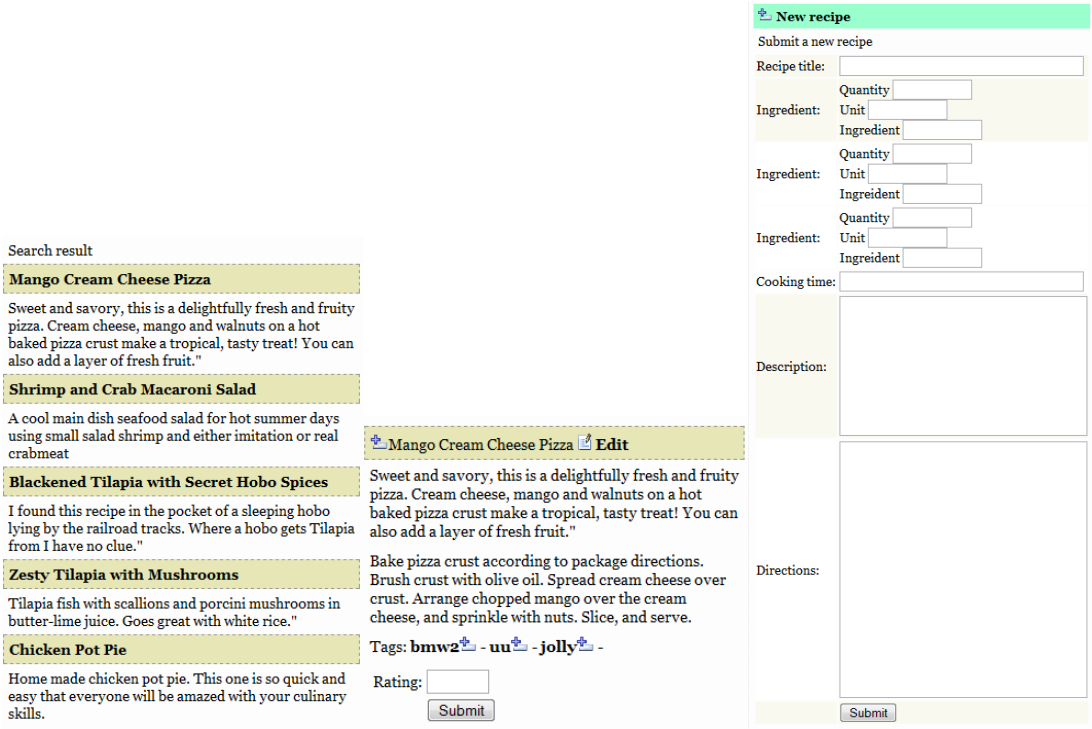
Figur 3 Opskriftssamlingens PageParts

Udover de tre funktioner er det naturligvis frit op til hver Page Part at udstille passende funktioner og attributter, og det er dermed potentielt stor forskel imellem forskellige Page Parts. For eksempel er min SearchPagePart ret simpel, den har en metode til at rendere sig selv til passende HTML, og en metode til at modtage PostBack fra samme form som den selv genererer. Deriomd har min RecipePagePart 9 forskellige render metoder, som hver renderer deres lille del af PageParten, som uafhængig af hinanden kan placeres på siden.

Dermed er RecipePagePart noget mere omfattende end de andre og skal den derfor beskrives lidt nærmere her, ligesom SessionPagePart også skal berøres.

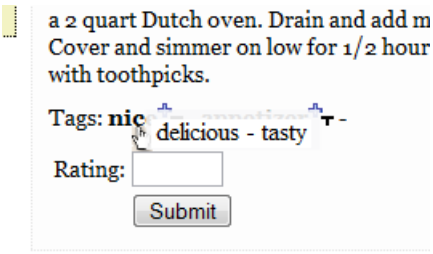
RecipePagePart

Største del af det dynamiske indhold på sitet bliver genereret af RecipePagePart. RecipePagePart kan være i forskellige *modes*, således at den både kan vise en liste af opskrifter eller en specifik opskrift i *view mode*, kan vise en opskrift i *edit mode* så brugeren kan ændre i opskriftens indhold eller *new mode*, som giver mulighed for at oprette en ny opskrift.



Figur 4 Tre forskellige modes. Fra venstre List mode, Item mode og New mode, til hhv. at vise en liste af opskrifter, en bestemt opskrift, og at oprette en ny opskrift.

Når en opskrift er valgt og den bliver vist fuldt ud, bliver dens tags også vist på siden, og de tags som er sat på andre tags, vises når brugeren holder musemarkøren over tagget som Figur 5 Tag på tags indikerer.



Figur 5 Tag på tags

Herudover har RecipePagePart funktionerne

- `renderTagForm()`
- `renderRatingForm()`
- `renderRecipeTags()`
- `renderRecipeIngredients()`

som leverer mindre enheder, som hører til en opskrift, men som kan placeres frit på siden.

SessionPagePart

SessionPagePart giver brugere mulighed for at logge på systemet. Hvis en bruger angiver korrekt brugernavn/password kombination bliver han logget ind, og hans brugerid bliver sendt i en cookie til brugerens browser.

Når brugeren logger af bliver cookien slettet[5].

Data Access

Alt som skal hentes fra eller skrives til databasen skal gennem Data Access laget. Data Access klasserne blev som udgangspunkt genereret ved hjælp af DB_DataObject. Dermed blev der lavet en klasse pr. tabel i databasen. Klasserne har samme navn som databasetabellerne, bortset fra klassernes store begyndelsesbogstav. Da det ofte er mere hensigtsmæssig at arbejde med begreber løst af de underliggende tabeller har jeg i flere tilfælde valgt at lade visse klasser i Data Access laget styre andre klasser. For eksempel ligger metoden til at hente en opskrifts ingredienser i ORecipe klassen, selv om der findes en OIngredient klasse, da jeg mener at logisk er ingredienserne en del af selve opskriften.

I et web system kan der være mange brugere, og derfor skal man være opmærksom på at brugerne kan tilgå de samme resourcer samtidig. Derved kan der opstå uventede konsekvenser.

Hvis to brugere samtidig begynder at redigere i en opskrift, og den ene først indsætter sine ændringer, og den anden derefter gør det samme, opstår der et concurrency problem, fordi den bruger som var sidst til at gemme sine ændringer, i virkeligheden ikke har editeret i seneste version af opskriften, og derved går den første brugers ændringer tabt. I desktopsystemer kan dette nemt løses med locking, hvor der kun er en bruger ad gangen som får skriveadgang til en resource. Men at sætte låse på resourcer i et websystem kan give uheldige bivirkninger, da det er svært at specifisere hvornår låsene skal frigives igen. I mange systemer vælger man derfor at poste en forms gamle værdier tilbage til serveren, sammen med de nye værdier, og så skal der checkes om de gamle værdier som blev sendt tilbage er lig de værdier som ligger i databasen på det tidspunkt. En anden mere simpel løsning er at man på hver tabel har et ModifiedTimestamp og henter et timestamp fra databasen hver gang oplysninger bliver læst, og så kun tillader indsættelse hvis ModifiedTimestamp ikke er efter det timestamp som man hentede ved læsningen. Nogle databaser har også direkte understøttelse for dette, f.eks. har Oracle en ora_rowscn som databasen sørger for er unikt for hver række i hver tabel.

I et flerbruger system og når man arbejder på flere resourcer samtidig, kan det ske at kun en del af operationerne bliver udført, eller at forudsætningerne ændrer sig undervejs. Transaktioner er til for at afhjælpe dette problem, og sørger for at der enten bliver udført hele sekvensen af operationer eller ingenting.

Der har i dette system ikke været tid til at implementere concurrency control og transaktionsstyring.

[5] Man kan ikke direkte slette cookien, men dens værdi bliver sat til tom streng og expire date så langt tilbage som muligt.

Evaluering

Det har været ret interessant at afprøve desiget af programstrukturen i praksis. Jeg mener at ideen med at arve fra Site, og at man lader Site være det eneste sted som skriver output, har været frugtbar. Man får naturligvis ikke meget forærende når man som i mit tilfælde kun har en side som nedarver fra Page og Site, men det giver alligevel en pæn struktur, og er også en god basis til når der skal laves nye sider udover RecipePagePart.

Jeg mener også at opdelingen i PageParts og regelen om at PageParts har deres egen livscyklus uafhængig af hinanden, gør koden mere overskuelig og pæn. Men én side med forskellige PageParts har også gjort det komplekst at holde styr på status i de forskellige PageParts, da der skal slæbes rundt på information om en PagePart når en anden PagePart laver post-back. Hvad skal der f.eks. ske ved RecipePagePart, når en bruger har logget sig på? Og hvilken vej skal kommandoen gå om at brugeren er logget på? Her ville det helt klart være simplere at implementere hvis systemet var mere fragmenteret i flere sider som havde hver deres specifikke opgave.

På grund af at man kan tagge både opskrifter og andre tags, valgte jeg i datamodellen at eksperimentere med noget der ligner arv, således at Tag og Recipe begge arver titel og id fra Resource. Dermed er det muligt at tildele et tag til en resource. Dette har delvist fungeret godt. Der er kun en funktion til at tildele et tag til en resource, i modsætning til to hvis man skulle tildele tags til hhv. opskrifter og tags direkte. Men – og det er svagheden ved den simulerede arv – funktionerne til at hente tags og opskrifter skal manuelt skrives til at joine op til resource, og når resource desuden i nogle tilfælde indgår to gange i den samme query kan det blive lidt uoverskueligt. Her må jeg nok indrømme at det ville have været simplere hvis jeg ikke havde lavet en "basis-tabellen" Resource, men istedet havde implementeret Tag klassens addToResource metode to gange.

Der er i disse dage meget hype omkring intelligent homes, og køleskabe med store displays der kan både det ene og det andet. Flere af de store køleskabeproducenter har i dag systemer til at vise opskrifter. Men et hurtigt rundkig viste mig at producenterne har valgt flotte men lukkede systemer, hvor der findes et meget begrænset antal opskrifter at vælge mellem. Her synes jeg helt klart man kan se en parallel til de lukkede systemer som eksisterede præ-web, som havde en masse finesser, men alligevel blev udkonkurreret af det meget simplere WWW. Derfor mener jeg at et websystem er perfekt til dette system, for kun derved kan man udnytte synergieffekten af alle verdens brugere.

For at evaluere systemet ville jeg have foretrukket at have en lille gruppe på 3-4 almindelige internetbrugere, som så kunne lave en heuristisk evaluering af systemet, og registrere usability-problems. Usability-problems kan i denne kontekst være alt fra inkonsistent sprogbrug, til at finde kringlede sætninger og til mere tekniske ting som manglende eller fejfyldt navigation på siden eller direkte fejl. Undersøgelser har vist at en sådan test med et forholdsvis begrænset antal utrænede testdeltagere[6] kan finde en relativt stor del af de fejl, mangler og uhensigtsmæssigheder, som sådan et system kan indeholde [Nielsen & Molich 1990]. Men da jeg ikke har adgang til det fornødne antal testbrugere, vælger jeg denne testform fra.

Istedet vælger jeg at lave en kort "cognitiv walkthrough" af en lille del af systemet. Jeg forventer at brugerne af mit system har en ret heterogen baggrund, helt fra hjemmegående husmødre til

[6] Dermed er det en billig test at udføre

madglade IT eksperter. Jeg vælger derfor laveste fællesnævner og siger at min bruger ikke har andet erfaring med it, end at vedkommende er habil internetbruger.

Opgaven som jeg kigger på er oprettelse af en ny opskrift. Opgaven består af 3 simple trin.

- Tryk på Add recipe linket
- Indtast opskriftens titel, beskrivelse, udførelsestid og vejledning.
- Tryk på submit knappen.

For hvert trin stiller jeg 4 simple spørgsmål og sætter svarene ind i en tabel.

	Tryk på Add recipe	**Indtast opskriftens oplysninger**	**Tryk Submit**
Vil brugeren prøve at opnå den ønskede effekt?	Ja	Ja	Ja
Vil brugeren opdage at den korrekte valgmulighed er tilstede?	Ja	Ja	Ja
Vil brugeren associere korrekt handling med den ønskede effekt?	Ja	Ja	Ja
Bliver brugeren gjort opmærksom på at der gøres fremskridt hvis korrekt handling udføres?	Ja, siden ændrer status, og får indtastningsfelter til ny opskrift.	Ja	Nej, efter at den nye opskrift er indsendt bliver sidens midte tom.

*spørgsmålene i tabellens venstre side er lånt fra [Lewis og Wharton]

Tabellen viser at der er flere success historier, men at der også er et punkt hvor brugeren ikke ved om det han har prøvet at udføre er blevet udført eller ej.

Jeg har suppleret den cognitive walkthrough med en Think Aloud test [Tognazzini] hvor en bruger uden kendskab til systemet blev bedt om at udføre samme trin som i den kognitive test, og samtidig skulle fortælle hvad hun tænkte imens. Her blev det bemærket at det umiddelbart var lidt svært at finde Add recipe knappen og at det ikke fremgik klart hvad forskellen mellem opskriftens Description og Directions er.

Disse to "failure stories" giver anledning til at redesigne indsætningen af nye opskrifter. Forskellen mellem Description og Directions kunne udpensles lidt nærmere på siden, og det er helt klart nødvendigt at have et statusfelt et sted på siden som siger om det gik godt eller ej, at indsætte opskriften, og desuden helst også vise den nyindsatte opskrift på brugerens skærm.

Så alt i alt har disse to simple brugergrænsefladetest vist nogle forhold, som kræver optimering, og som skal forbedres i næste iteration.

Indsættelsen af tags, både på opskrifter og på andre tags, kunne med fordel suppleres af en AJAX auto-complete funktion, således at systemet kunne hjælpe brugeren med forslag til tags som allerede er brugt. Dette er specielt en hjælp til at få brugere til at bruge samme bøjning af ord til tags.

Appendix A Brugsvejledning

En kørende udgave af systemet findes på linket:

http://www.daimi.au.dk/~marjus/opskrifter/index.php

Søgning

Der kan søges efter opskrifter i sidens øverste venstre hjørne. Fundne opskrifter rangeres først efter hvor tit tags er tilknyttet opskriften. Et tag skal her være tilknyttet en opskrift mindst tre gange for at tælle med.

Siden

I venstre side er tre "kasser". Den første kasse viser ingredienser, den næste hvor lang tid opskriften tager at udføre og den nederste kasse viser gennemsnittet af den rating, som den valgte opskrift har fået.

I højre side vises øverst de mest populære opskrifter, og derunder kan brugere logge på, og skrive sig op som nye brugere.

Opskrifter

I midten vises selve opskrifterne. Efter en søgning vises en liste af opskrifter. Man kan så vælge en opskrift, ved at klikke på dens titel. Opskriften vises så fuldt ud, og ved venstre side af opskriftens titel kan man klikke på Add tag ikonet (). Herefter popper et lille vindue op ovenover opskriften hvor man kan tildele opskriften et tag. Ved titelens højre side findes et Edit link som giver mulighed for at ændre i opskriften.

Under opskriftsteksten vises de tags som allerede er tildelt opskriften. Hvert tag har et Add tag ikon, som giver mulighed for at tildele et tag til et andet tag.

Derunder er et indtastningsfelt til rating. Her kan man give opskriften en karakter.

Appendix B Den oprindelige systembeskrivelse

Synopsis

Der skal alves et website til brugerdefineret semi-struktureret materiale, hvor man har en blanding af klart afgrænsede entiteter og fritekst. Som case vælger jeg at lave en madopskriftssamling, hvor ingredienser og mængder er afgrænsede entiteter og fremgangsmåden bliver beskrevet i fritekst, men casec kunne ligeså godt tage udgangspunkt i en reparationsmanual til biler eller forskrifter for hvordan man anlægger en have.

I prinsippet bliver sitet en blanding af en wiki, med mulighed for at lave kommentarer ligesom i en blog, og med mulighed for at sætte tags på opskrifter, og derudover skal tags kunne sættes på andre tags for at linke relaterede tags sammen. Dette skal alt kunne indsættes uden at brugere er logget ind. Brugere som er logget ind kan lave en liste over deres favorit opskrifter.

Websites

Der skal udvikles et website som i sagens natur består af en webside som skal vises for en bruger i en webbrowser. Websiden bliver afsendt fra en webserver og der bliver sendt forholdsvis simple beskeder tilbage til webserveren om de valg brugeren foretager sig.

I beskrivelsen af systemet vælger jeg at bibeholde opdeleingen af systemet i en front-end og en back-end. Front-end er slutbrugerens kontaktflade til systemet og Back-end er systemet som kører på webserveren. Med andre ord er beskrivelsen delt op i en beskrivelse fra en slutbrugers synsvinkel og en applikationsudvikler synsvinkel hvor der mere fokuseres på teknikken bag systemet.

Front end

Forsiden

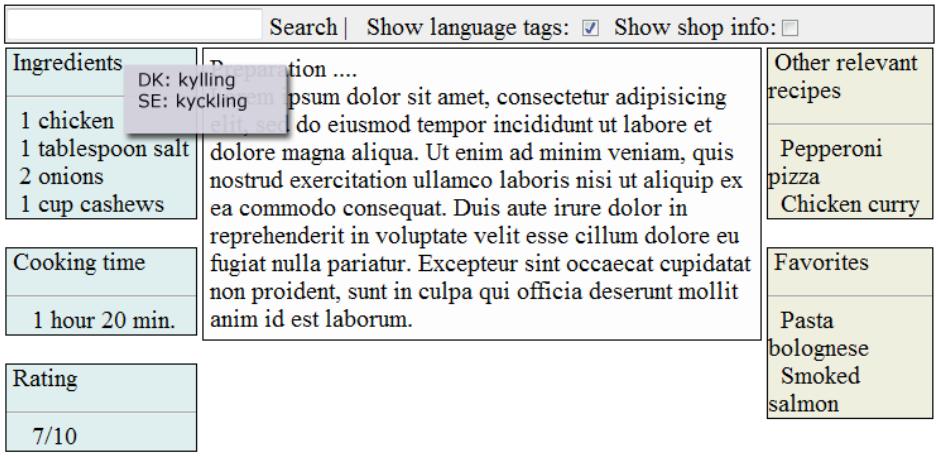
En meget simpel mock-up af systemets forside kan ses forneden. Billedet afspejler ikke det endelige design, men skal udelukkende betragtes som en måde at vise sitets funktionalitet.

På forsiden kan brugere søge efter opskrifter. Søgningen viser en rankeret liste med en kort beskrivelse af hver opskrift. Brugeren kan derefter vælge en af opskrifterne som så bliver vist fuldt ud.

Øverst i venstre hjørne vises en liste af opskriftens ingredienser og hvor meget der skal bruges af hver ingrediens. Ved at holde musen over en af ingredienserne vises de informationer som systemet har vedrørende den pågældende ingrediens. I første udgave af systemet kan det være oplysninger om hvad varen hedder i andre sprog eller hvor man kan købe varen. Hvis der klikkes på en af ingredienserne resulterer det i en søgning på opskrifter som indeholder den valgte ingrediens.

Under ingredienserne vises en vurdering af hvor lang tid det tager at lave retten. Og nederst i venstre hjørne kan brugeren se hvor populær opskriften er blandt andre brugere.

Øverst i højre hjørne vises andre opskrifter som systemet vurderer er relevante i forhold til den valgte opskrift. Derunder finder brugeren de opskrifter som han har valgt som favoritter. Denne funktionalitet kræver at brugeren er logget ind. Hvis brugeren ikke er logget ind kan systemet vise en top-ti liste med de opskrifter som har opnået den højeste rating.



Oprette opskrift

Brugere skal kunne oprette opskrifter hvor de angiver hvilke ingredienser skal bruges i opskriften og i hvilke mængder og hvor lang tid det tager at lave den pågældende ret. Desuden skal en fremgangsmåde skrives i fritekst.

Efter at opskriften er skrevet gemmes den i systemet og bliver tilgængelig for alle andre brugere.

Editere opskrift

Ligesom i en wiki skal det være helt åbent for alle brugere at editere i eksisterende opskrifter, både i ingredienser og i fremgangsmåden.

Kommentarer

Det skal være muligt for brugere at give kommentarer til en opskrift uden at ændre i selve opskriften.

Tags

Ingredienser bliver i systemet opfattet som tags, som tilhører en opskrift. Tags bliver genbrugt, således at hvis en bruger skriver "chi" i et ingrediens-felt skal han kunne vælge "chicken" og "chili" fra en liste, såfremt disse to ingredienser er blevet brugt i forvejen.

Udover ingredienser kan man knytte andre tags til en opskrift og det skal også være muligt at knytte tags til andre tags. På den måde bliver det muligt at beskrive med tags at pizza kommer fra Italien uden at det nødvendigvis skal stå i opskriftsteksten. Og det skal være muligt at knytte et tag til f.eks. en ingrediens, som siger at "chicken" er det samme som "kylling" på dansk eller "kyckling" på svensk. Eller at en vis soya kan købes i Bilka.

Back-end

Systemet implementeres i php og alt data gemmes i en mySQL database. Php er velegnet til dynamiske websider, og mySQL har i mange år været et naturlig datalagringsprodukt at bruge sammen med php, både fordi det er gratis at bruge og ikke mindst fordi der er ret god understøttelse i php for netop mySQL.

Rating

Slutbrugerne kan give opskrifterne karakter fra 1 til 10. Gennemsnittet af de givne karakterer bliver udregnet for hver opskrift.

Tags

For at forbedre datakvaliteten af tags og for at slippe for at brugere – bevidst eller ubevidst – indsætter forkerte eller misvisende tags, er det et krav at der skal være mere end en bruger som har knyttet det pågældende tag til resourcen. For at holde det simpelt vælger jeg i første omgang at sige at tags bliver kun talt med hvis de er knyttet til en resource mindst tre gange – ingredienser dog undtaget, de kommer med i opskriften selv om de kun er indsat en gang. I søgninger vægtes søgeresultatet efter hvor ofte relevante tags er brugt sammen med hver opskrift.

Indtastningsfelter til tags skal have auto-complete funktionalitet således at brugere får hints til andre tags som allerede er oprettet. Auto-complete funktionaliteten skal køre over AJAX, for at give den bedste brugsoplevelse.

Relevante opskrifter

Ud fra hver opskrifts ingredienser og tags skal der hver gang en opskrift bliver vist, også vises en liste af op til 10 andre relevante opskrifter. Relevansen rankeres i denne sammenhæng efter:

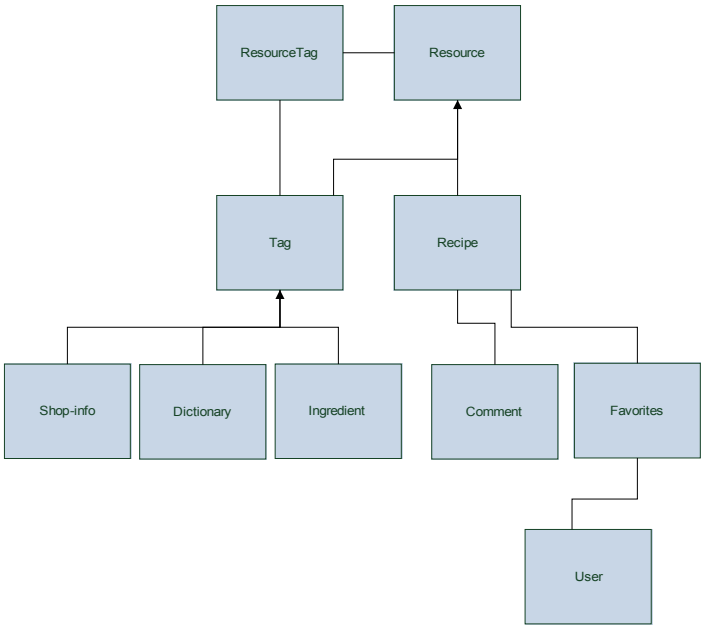
1. Tags som er knyttet til hver opskrift. Opskrifter som deler de samme tags er relevante for hver andre.

2. Ingredienser. Hvis der ikke findes nok opskrifter efter det første kriterie vises opskrifter som indeholder nogen af de samme ingredienser.

Entiteter

I nedenstående diagram er systemets entiteter skitseret. Det skal understreges at entiteterne i diagrammet ikke nødvendigvis mapper direkte til klasser eller tabeller i implementeringen.

Tags kan både tilknyttes opskrifter og andre tags. Det giver anledning til at overveje en fælles type for disse. I diagrammet kaldes de samlet for Resourcer.



En ingrediens er en special udgave af et tag. For at give den bedste oplevelse for slutbrugeren bliver det nok nødvendigt at have flere specialiserede tag typer. Som udgangspunkt vælger jeg udover ingredienser og almindelige tags at lave Dictionary tags og Shop-info tags, men det skal være forholdsvis simpelt at udvide systemet med nye tagtyper. Dictionary tags bruges til at sige hvad en resourcer hedder på andre sprog og Shop-info tags bruges til at fortælle noget om hvor ingredienser eller redskaber kan købes.

I forhold til almindelige tags som normalt er enkeltord, som er tilknyttet en ressource, bliver Dictionary tags enten oprettet som sprog-ord par, eller skal et dictionary tag knyttes til både et sprog og en ressource. Der tages stilling til hvilken løsning skal vælges i implementeringsfasen.

Brugere

Sitet skal kunne bruges af alle brugere uden at de er logget på systemet. Men ved at logge på skal brugere få mulighed for at administrere favoritlister over opskrifter, som de vil gemme og dermed lettere kunne finde frem.

<< Appendix A slut >>

Referencer

Hudson, Paul: *Practical PHP programming*, http://hudzilla.org/phpwiki/index.php?title=Main_Page

Lewis, C og Wharton, C: *Cognitive walkthroughs*, 1997 Handbook of Human-Computer interaction

Nielsen, J og Molich, R: *Heuristic evaluation of user interfaces*, ACM 1990

Tognazzini, B: User testing on the cheap. TOG on interface.