18-03-2018

Cupcake Projekt

David Nielsen, Hjalmar Thulstrup, Martin M. Larsen og Rasmus Lumholdt
CPHBUSINESS, DATAMATIKER KLASSE B

Navn	**David Nielsen**	**Hjalmar Thulstrup**	**Martin M. Larsen**	**Rasmus Lumholdt**
Github usernames	David-M-Nielsen	HjalmarThulstrup	MartinMoller	Ra11e
CPHBusiness email	CPH-DN93@cphbusiness.dk	cph-hh231@cphbusiness.dk	cph-ml511@cphbusiness.dk	cph-rl107@cphbusiness.dk

Indholdsfortegnelse

Indledning

I dette projekt har målet været at udvikle en hjemmeside, til et firma som sælger cupcakes. Hjemmesiden skulle have nogle forskellige funktioner, som skulle hjælpe firmaet med bedre at kunne køre deres forretning.

Baggrund

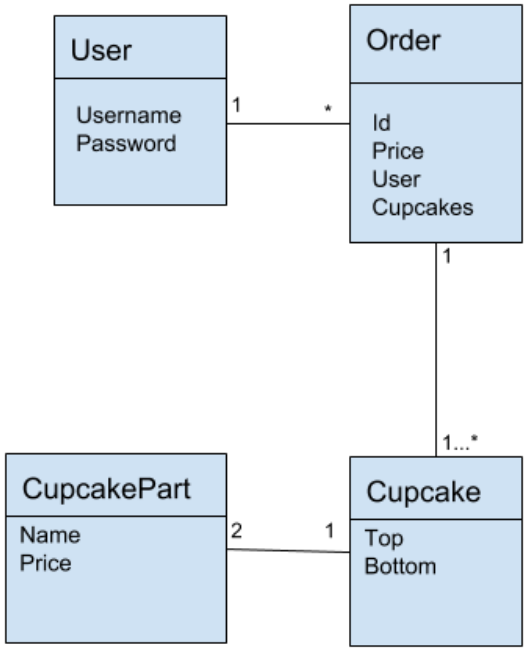
Virksomheden vi arbejder for producere cupcakes, og de har brug for en hjemmeside hvorpå man kan bestille og administrerer deres produkter. Firmaet giver kunden mulighed for at kunne vælge sin egen top og bund af en cupcake, hvilket de vil have afspejlet på hjemmesiden. Firmaet har ønsket nogle forskellige funktioner til hjemmesiden, funktionerne som hjemmesiden skulle inkludere, er mulighed for at oprette en bruger på hjemmesiden, mulighed for at lave en ordrer af cupcakes, for at en bruger kan slette sin ordrer og en admin side hvor en admin skal kunne administrere på kunder og ordrer. Derudover skulle den sættes op med css og bootstrap for at være mere overskuelig og pæn.

Teknologivalg

Teknologi	Version
Netbeans	8.2
JDBC	5.1.44
MySQL	5.7.21
DigitalOcean Ubuntu Server	17.10.1
JavaEE-web-api	7.0
Bootstrap	3.3.7

Domæne model og ER diagram

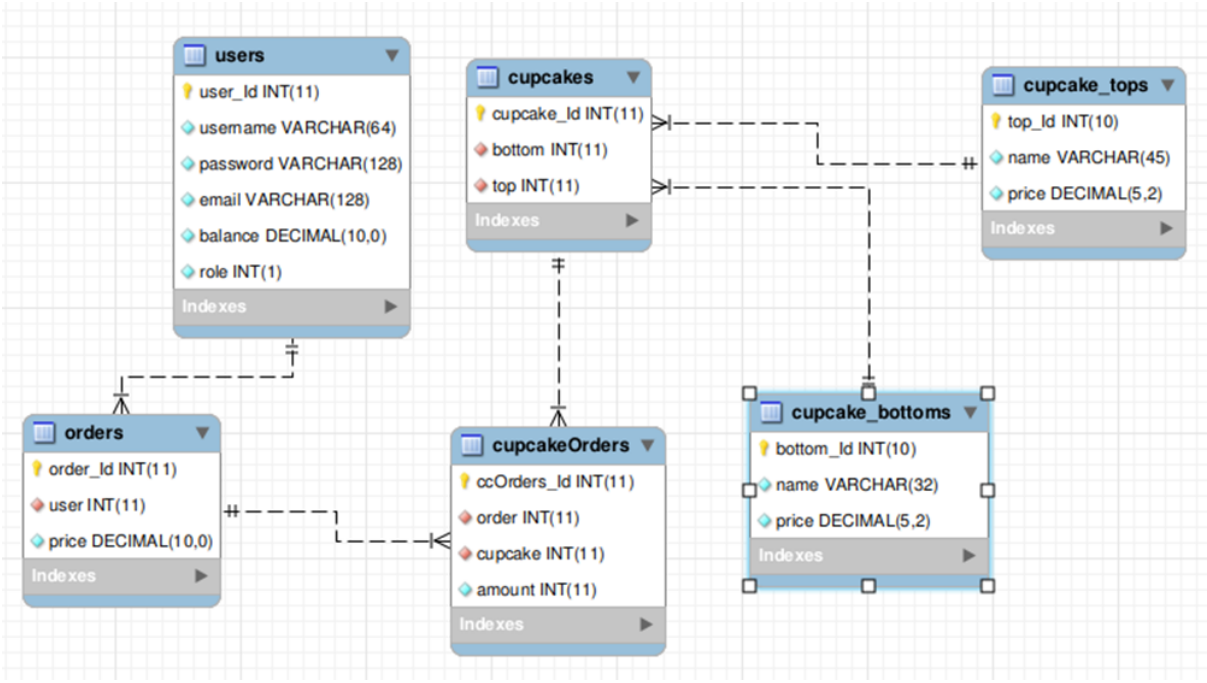
Domæne model



Domænemodellen bliver brugt for kunne få en bedre idé om hvordan produktet skal udfolde sig i udviklingsprocessen. I dette projekt blev domænemodellen lavet meget sent i processen hvor at den normalt er noget af det første man laver. Vores domænemodel er meget simpel og indeholder 4 forskellige navneord. User, order, cupcake og cupcake part.
En users vigtigste attributter er username og password, hvilket brugeren skal bruge til at logge ind, user er forbundet til order, og har et 1 til mange forhold, dette giver mening da en user kan have flere forskellige ordrer.
Orders vigtigste attributter er Id, price, User og cupcakes, grunden til at vi har valgt dem som de vigtigste er at det er de værdier som skal bruges databasen for at lave en ordrer.
Order er forbundet til Cupcake med en ”en til mange” forbindelse da en ordrer kan have mange forskellige cupcakes, men den cupcake kan kun være i en ordrer.
Cupcakes vigtigste attributter er Top og Bottom da en cupcake ifølge opgaven består af en top og en bund. Cupcakes er forbundet til CupcakePart med en ”en til to” forbindelse da, en cupcake kan have to forskellige cupcakeparts, en top og en bund.
CupcakeParts har kun 2 attributter, Name og Price. Dem bruger vi da vi skal vide hvilken top eller bund det er ud fra navn og hvad prisen er, hvilket price indeholder.

Domænemodellen som er blevet udarbejdet meget lille og og meget simpel, hvis man skulle forbedre det, ville man gå mere i detaljer. Man kunne bl.a. tilføje nogle flere kasser for at give et større overblik over produktet. På den anden side er det her et meget småt og simpelt projekt, hvilket gør domænemodellen en smule mindre vigtig. Man ville altså kunne lave dette program helt uden domænemodel. Til gengæld når projekterne bliver en smule større og der skal kommunikeres omkring projektet med andre mennesker i en anden position end os vil det være meget nyttigt.

ER Diagram



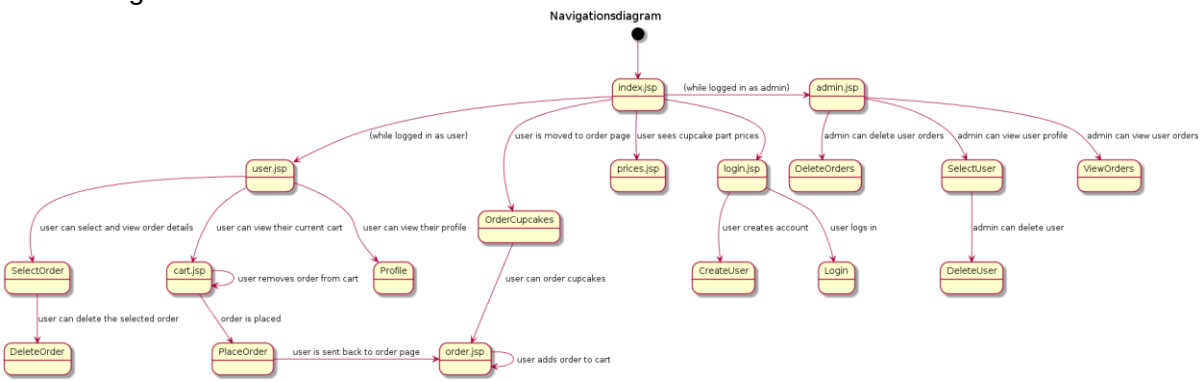
Tabellen her beskriver hele cupcake_factory databasen. Den er rimeligvis overskuelig, men der er dog et par interessante punkter.
Vi kendte ikke til orderline metodologien da vi lavede den, men vi kom alligevel frem til det samme koncept i vores cupcakeOrders tabel.
Denne tabel har 1 til mange relation mellem både cupakes og orders. Det er gjort sådan så en ordre kan have mange forskellige cupcakes.
Vi fandt senere ud af, at selve cupcakes tabellen er unødig, og at man bare ville kunne indsætte cupcake_top og cupcake_bottom direkte ind i cupcakeOrders, og dermed normalisere databasen en smule.

Tabellen cupcakes har 1-1 relation med både cupcake_tops og cupcake_bottoms.
Vi lavede den originalt fordi det virkede som om at det gjorde databasen mere overskuelig, men da der skulle laves nye ordre, blev det hurtigt til et problem.
Måden vi laver ordre på, gør at der laves en ny række i cupcakes med top og bund, selv hvis der allerede er en identisk cupcake. Det kan hurtigt tage meget plads.
Vi har derfor konkluderet at den er unødig. Ved at indsætte tops og bottoms direkte i orderline(cupcakeOrders) undgår vi dette problem helt.

Users tabellen har en 1 til mange relation med orders. Det er fordi en bruger skal kunne lave flere ordre på en gang. Vi så ikke nogen grund til at begrænse det.
Orders tabellen har også et redundant felt, pris. Prisen kan findes igennem cupcake_tops og cupcake_bottoms. Vi lavede originalt pris feltet for at have en nem måde at tilgå indsætning af ny ordre, men igen, så har det kun gjort det mere kompliceret.

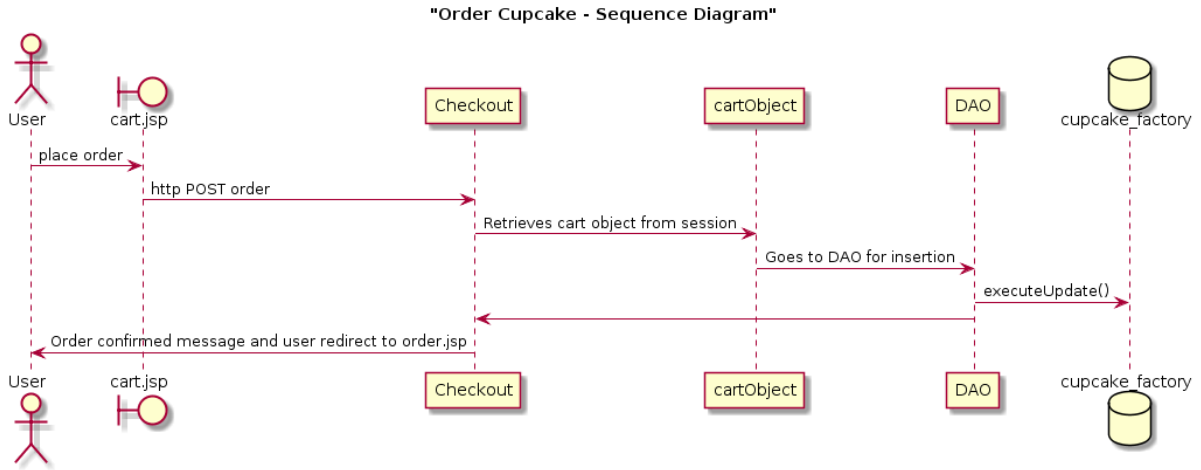
Navigationsdiagram

Målet med navigationsdiagrammet er at gøre det mere overskueligt for dem der skal arbejde på systemet, at få et overblik og gøre det mere overskueligt at skulle sidde med de forskellige dele. For at få en ordentlig version af diagrammet se “navigations_diagram.png” under bilag.



Når en bruger lander på frontpage vil der være adgang til siderne “Order Cupcakes”, “prices”, og “login”, Siderne kan tilgås fra en navbar i toppen af hver eneste side. Hvis en bruger vil have mulighed for at ændre i sine ordrer eller kunne lave dem skal brugeren oprette et login, på siden “login”. Når brugeren bliver oprettet, vil den blive tildelt rollen customer, og vil derfor have adgang til alle siderne under “User”. Når en bruger logger ind, vil han/hun blive sendt til “User” siden. Når en bruger er logget ind, vil der blive generet endnu et link i navbaren, dette link hedder “user” og vil sende en brugeren til sin egen side. På brugeres side vil han/hun have mulighed for at kunne slette eller redigere en ordrer. Når man er logget ind som customer har man også mulighed for at lave en ordrer, som derefter vil kunne ses og redigeres under bruger siden.
Hvis en der bliver logget ind med en admin bruger vil personen have adgang til admin siden, admin siden bliver vist i navbaren, når man er logget ind som admin. Admin siden giver en mulighed for at håndtere brugere og ordrer. En admin vil kunne se en liste over alle brugere og ordrer og vil være i stand til at slette ordrer og eller brugere.
Hvis en bruger skriver sit brugernavn og kode forkert vil de blive redirected/ført til en side der fortæller ham/hende at brugeren ikke findes og får muligheden for enten at lave en ny bruger eller prøve igen.

Sekvensdiagram



I dette sekvensdiagram ses det hvordan en bruger vil bevæge sig igennem de klasser der bliver brugt i processen af at bestille cupcakes.
Det starter med at brugeren placere sin ordre fra indkøbskurven sætter servletten Checkout i gang. Servletten udtager bestillingsinfoen fra brugerens session og sender det videre til vores datamapper DAO's placeOrder metode, som så indsætter ordren i databasen.

Derefter fortæller checkout brugeren at ordren er blevet placeret, og redirecter dem til order.jsp, hvor de nu kan bestille flere cupcakes hvis de har lyst.

Særlige forhold

Session

Projektet er designet sådan at der i sessionen vil blive gemt nogle af brugerens oplysninger. Dette bliver gjort da brugeren så vil kunne bevæge sig rundt på hjemmesiden, og stadig kunne være logget ind på alle de sider han besøger.
I sessionen vil der blive gemt brugerens oplysninger efter han/hun har logget ind på hjemmesiden, disse oplysninger vil blive gemt indtil sessionen udløber eller brugeren logger ud. Brugeren bliver gemt som et "User" objekt i sessionen, og man kan igennem det objekt tilgå oplysninger som, brugernavn, email og id. Vi valgte derimod ikke at have adgang til brugerens kode da dette ville være en sikkerhedsrisiko.
Derudover bliver brugerens kurv også gemt i sessionen. kurven bliver gemt som en arrayliste kaldet "orderList", denne liste indeholder "orderPiece" objekter. Det at brugerens kurv bliver gemt i session, gør at brugeren kan købe flere forskellige cupcakes over længere tid, derudover vil brugeren være i stand til at bevæge sig rundt på hjemmesiden, uden at miste sin kurv.

Validering af brugerinput

I dette projekt har vi ikke lagt så meget vægt på validering af brugerinput, der er dog blevet brugt validering nogle steder i projektet. I projektets videre udvikling skal der gøres noget validering af input på en nogle af siderne, som f.eks. login/create user.

Lige nu er vores projekt sårbart over for angreb, som f.eks. sql injections. Da vi gemmer inputtet fra en bruger og sætter det direkte ind i sql forespørgslen uden at validere inputtet eller behandle det, vil en anden bruger kunne udsætte projektet for en sql injection. For at undgå dette problem ville prepared statements være bedre at bruge. Grunden til at vi ikke har brugt prepared statements var at vi havde problemer med at få det til at fungere, og det var derfor mere effektivt at bruge den anden metode.

Opret bruger

Når man opretter en bruger, bliver man bedt om at skrive et brugernavn, password og en email ind i opret bruger formen.
Brugernavn feltet er bare et normalt tekstfelt, da brugerne må hedde hvad end de vil.
Password feltet ligger under typen “password” hvilket gør at man ser så prikker i stedet for de bogstaver der bliver skrevet ind.
Email feltet er af typen email, og det er derfor krævet af brugeren at de skriver en gyldig mailadresse ind, hvor gyldig betyder det af strengen af tekst skal indeholde et @ og et topdomæne.
Når man laver brugeren, bliver der til gengæld ikke undersøgt om brugeren allerede findes i databasen. Der skal derfor senere indbygges et valideringssystem når man opretter brugeren, da dette vil kunne føre til store problemer med hjemmesiden. Lige nu fungere siden sådan at den crasher hvis brugeren allerede findes, da felterne er sat til unique i databasen. Dette betyder at vi bliver nødt til at implementere en validerings funktion når man laver en ny bruger.

Bestilling af cupcakes

Når man bestiller en cupcake, skal man også vælge antallet af den enkelte cupcake, man vil købe. Vi har valgt at begrænse antallet af cupcakes man kunne købe til at være mellem 1 til 200 cupcakes, da en bruger vil kunne lave ordre alt for stor for cupcakefirmaet at håndtere. Alt efter størrelsen på firmaet kan begrænsningen af antal ændres.

Sikkerhed omkring login

I dette projekt har vi endnu ikke fokuseret på bruger sikkerhed det betyder, at der vil være en del, mangler på sikkerheden. En af de ting man burde gøre, men som vi ikke har gjort kunne være at hashe koder, så de ikke ligger som plaintext i databasen, dette udgør en kæmpe sikkerhedsrisiko. Hvis databasen f.eks. blev hacket, ville alle koderne ligge klar til at andre kan bruge dem, det ville derfor have været en god idé hvis vi hashede koderne før de blev lagt i databasen.
Et andet problem som endnu ikke er løst med hensyn til login, er at hvis man logger ind og skriver den forkerte kode eller det for forkerte brugernavn, bliver dette ikke håndteret i koden så man får en fejlside med exception.
En fejlside vil nemt kunne implementeres, og var ikke set som andet end tid som kunne bruges på nogle mere presserende funktioner.
Login validation er en vigtig sikkerhed på *rigtige* hjemmesider, og vi tog kun denne beslutning netop fordi at det var en projektopgave, og ikke en rigtig produktion.

Brugertyper

Lige nu er der 2 typer brugere. Administrator og normalbruger. Forskellen mellem dem er kæmpe. En normal bruger kan kun tilgå deres egen info og de kan bestille cupcakes. Admin kan tilgå næsten alt info fra de andre brugere, inkluderet brugernavn, email og deres ordre.

Vores database tilgås med en administrator bruger gennem database connectoren, så vi har en enkelt, bruger der udfører alle CRUD funktionerne. For at give vores hjemmeside bedre sikkerhed, kunne man have lavet roller som reader og writer i databasen. Dette ville øge sikkerheden ved at nægte nogle brugere adgang til at redigere, slette og opdatere i tabeller de ikke har noget at gøre med. Det er ikke normalt et problem, men hvis man løber ind i en hacker sådan noget, så kan det godt være en god ide.

Status på implementering

Siden er mere eller mindre fuldt implementeret. Det eneste der lige umiddelbart mangler, er nogle sikkerhedsforanstaltninger i login og CRUD.
Der mangler en valideringsmetode til at finde ud af om en bruger allerede findes i databasen. Derudover mangler der, som nævnt tidligere, input validering fra brugeren. Dette kunne laves med bl.a. prepared statements, hvilket ville være den foretrukne måde at lave det på.

Vi mangler en pæn måde at vise fejlet login og mislykket brugeroprettelse. Det kan gøres ved at vise en alert med en besked på siden ligesom vi gør når der bliver sat en cupcake i cart. På en måde undgår vi en grim error side, og vi gør dermed det hele mere brugervenligt.