Cupcake Shop

7. september 2018

Gruppe J M R

Navn, mail, github

Mads Fløistrup, cph-mf226@cphbusiness.dk, mf226

Jonatan Mejer Hjelm, cph-jh409@cphbusiness.dk, JonatanHjelm95

Rasmus Porse, cph-rp127@cphbusiness.dk, Porselicious

Indledning

Virksomheden er en bager der udelukkende sælger cupcakes til sine kunder. Kunden har stillet som krav, at der blive lavet en funktionel online butik. Her skal det være muligt for shoppens kunder at kunne oprette brugere, bestille og betale cupcakes. En kunde af butikken skal derfor have sit eget brugernavn og password samt en mulighed for at kunne indbetale penge, som skal kunne trækkes når der afgives en ordre. Der forekommer ingen leveringer fra butikkens side, kunderne skal selv pick-ups af ordrerne. Kagerne varierer i forskellige kombinationer af toppe og bunde.

Teknologivalg

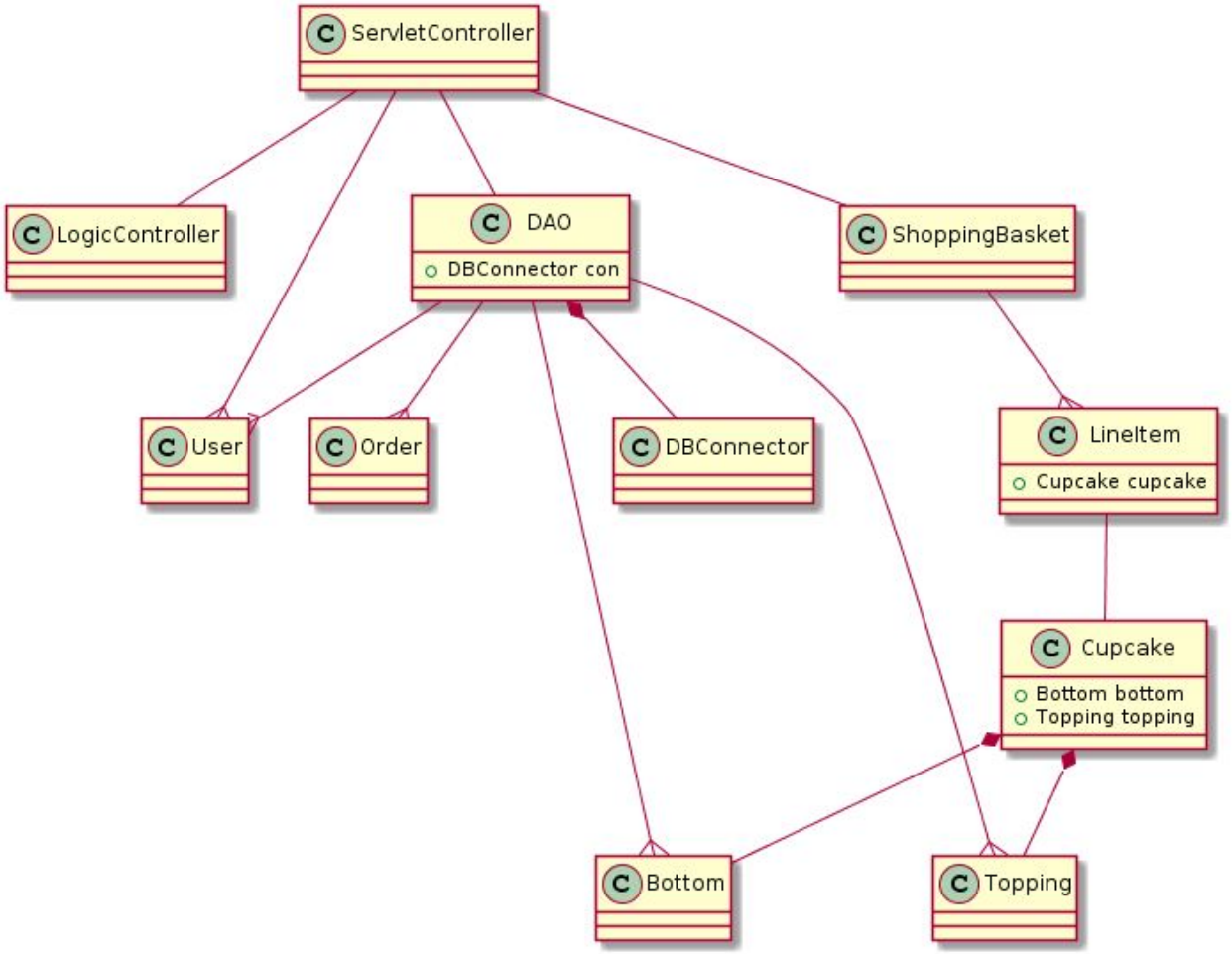
I dette projekt er følgende programmer brugt:

- Netbeans 8.2
  - Java
  - mysql-connector-java 5.1.47
  - javaee-web-api 7.0
- MySQL Workbench 6.3 CE
  - Database

Diagrammer

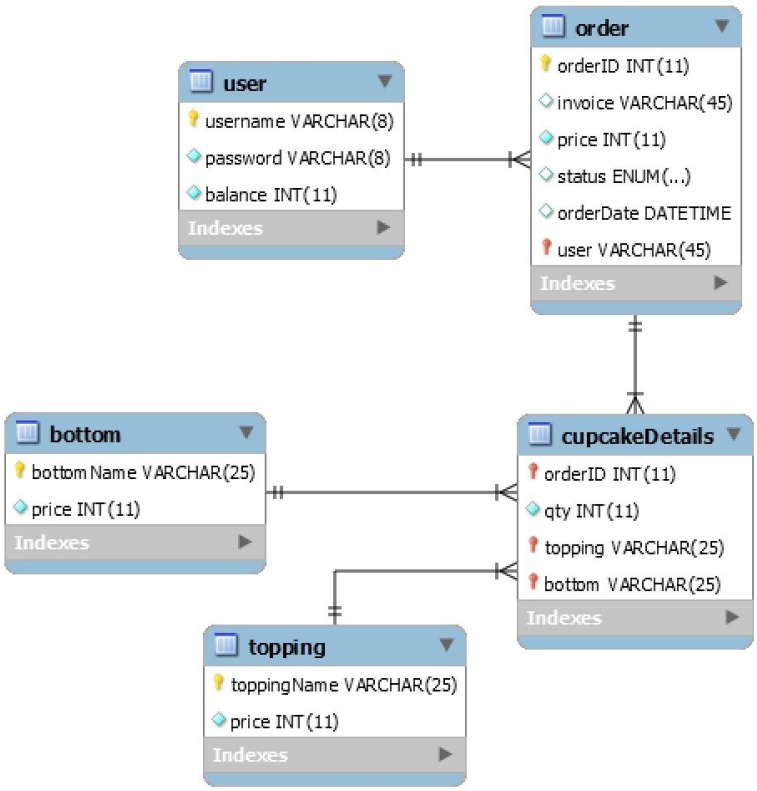
De følgende diagrammer er lavet ved hjælp af plantuml.com og MySQL Workbench.

Domæne model og ER diagram



- **ServletController** holder forbindelser til alle de underordnede klasser i en 1-til-1-relation. Klassen har til ansvar at få videreført input fra JSP-siderne til DTO'erne, og samtidig at modtage HTML-strings fra logicControlleren, som bliver sendt ud til JSP-siderne.

- **LogicController** tager imod *User*, *Topping og Bottom*, og genererer HTML-strings ud fra deres attributter. Disse strings kommer derefter ud til JSP-siderne via *ServletControlleren.* Formålet med LogicControlleren er at fjerne så meget java-kode fra JSP-siderne, som muligt. Samtidig medvirker LogicControlleren til en mere dynamisk html.
- **ShoppingBasket** indeholder en ArrayList af LineItems. Disse LineItems indeholder bl.a. Cupcakes, som er en komposition af *Topping* og *Bottom.*
- **Topping/Bottom** har ud over relationen til Cupcake, også en direkte relation til *DAO*. *DAO* får via *DBConnector* hentet ovenstående objekter fra databasen i en én-til-mange-relation.
- **DAO** har også til ansvar at tilgå og oprette *User* og *Order* i databasen i en én-til-mange-relation.

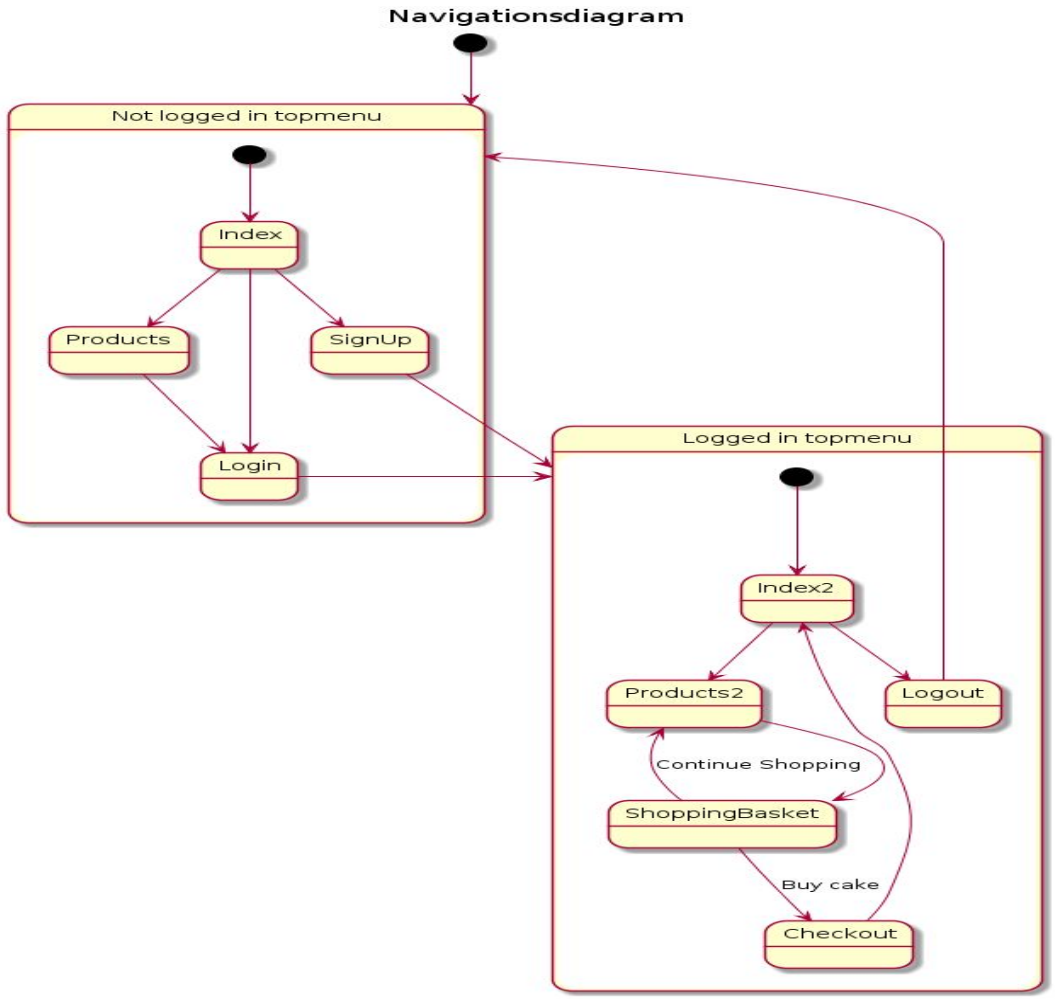


I det ovenstående ER diagram vises relationerne i Cupcake databasen.

- **user -** Tabellen holder på en generisk bruger med en konto.

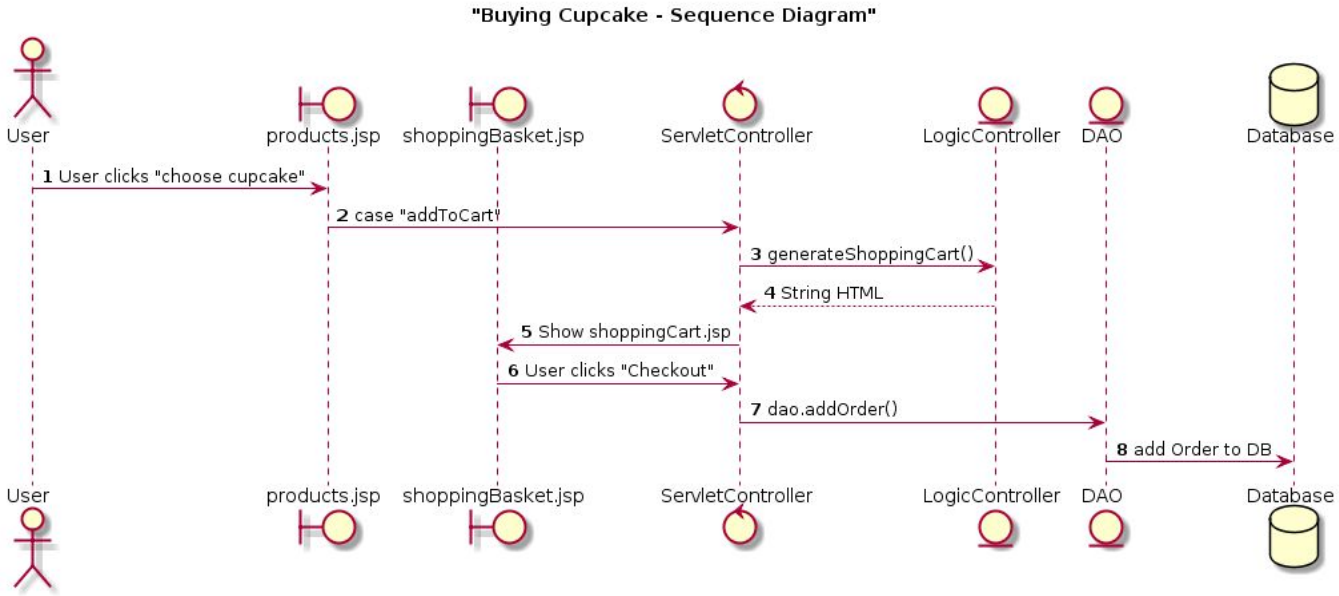
- **order -** Denne tabel indeholder selve ordre oplysningerne. Kolonnen ‘price’ betyder her den samlede pris for hele ordren. Fremmednøglen ‘user’ refererrer tilbage til tabellen ‘user’ for at kunne binde ordren op på en bruger og sammenholde ‘price’ og ‘balance’ med hinanden.
- **cupcakeDetails -** ‘topping’ og ’bottom’ nøglerne er til så der kan hives navne og priser fra de to tabeller, så der kan samles en cupcake i ‘cupcakeDetails’. Fremmednøglen ‘orderID’ er til så når der er lavet en cupcake af en bund og top, så kan den og antallet af dem, samt price hæftes på ordreID’et. ‘OrderID’ er sat til ‘On Delete CASCADE’, så data om de samlede cupcakes forsvinder hvis den tilhørende ordre slettes.
- **topping og bottom -** er simple tabeller der er til at konstruere cupcakes. Vi valgte at have dem i hver deres tabel, da det ville blive nemmere at hive de individuelle lister ud.
- Det blev besluttet ikke at have en CupCake tabel, da den blot ville indeholde redundant information.

Navigationsdiagram



Navigationen på brugergrænsefladen foregår ved hjælp af en topmenu der har to "states", enten er brugeren logget ind eller også er han ikke. Når brugeren logger ind eller laver en ny bruger bliver der skiftet til den nye state og en "shoppingbasket" kan nu ses, hvortil der kan tilføjes cupcakes og endelig kan der klikkes på "checkout". Hvis brugerens "balance" er høj nok tilføjes ordren til databasen.

Sekvensdiagrammer



Det ovenstående sekvensdiagram beskriver den “use-case” hvor brugeren har besluttet sig for én cupcake der skal bestilles. Brugeren vælger en cupcake og der trykkes add to cart, som giver en tilsvarende case “addToCart” i servlet Control. Her checkes i en if statement først om brugeren er logget ind. Hvis brugeren ikke er logget ind bliver varen ikke tilføjet og der bliver redirected til en errorpage. Herefter hentes den valgte cupcake som parametrene “topping” og “bottom” samt et antal, “qty”. Den valgte cupcake tilføjes til brugerens “ShoppingBasket” på sessionen, og der genereres en HTML String med indholdet som ligeledes tilføjes til sessionen. Herefter bliver der “redirected” til ShoppingBasket.jsp hvor brugeren kan se indholdet af kurven og herefter vælge “continue shopping” hvorefter skridt 2, 3 og 4 ville blive gentaget. I ovenstående diagram vælger brugeren dog “Checkout”. Dette ender i Servlet Control switch case “CheckOut” hvor der checkes om brugeren har penge nok inden der via “DAO’en” tilføjes en ordre til databasen.

Særlige forhold

En session bliver kun oprettet ved succesfuldt login eller ved at oprette en bruger. Derfor kan vi bruge (request.getSession(false) != null) for at checke om en bruger allerede er logget ind.

På sessionen bliver der gemt en User (den der er logget ind/oprettet sig) og til at starte med en tom ShoppingBasket.

Status på implementation

In development

Website:

- Products:
    - Når man ikke er logget ind er det stadig muligt at vælge produkter, men det bliver ikke gemt i 'Shopping cart' og der stadig mulighed for at vælge 'Checkout' hvilket fører til en blank side. Her skulle brugeren flyttes til loginsiden, for at blive ført tilbage efter et succesfuldt login.
    - Der mangler generelt validering af brugerindtastning.

- Styling:
    - På 'Home' mangler der et billede af en cupcake.

DAO:

- En metode til at udtrække alle ordre for en given bruger mangler. Det ville specielt være aktuelt for en bruger at se tidligere ordrer.
- Det er kun muligt for brugeren at indsætte penge på sin konto ved oprettelse af den. En mulighed for at kunne indsætte efterfølgende er en væsentlig funktion der mangler.

Admin:

- Implementationen af en admin-funktion mangler på nuværende tidspunkt. Her ville det være oplagt at admin havde mulighed for at kontrollere indkommende ordrer, status på samme og kunne få overblik over indkomst af bestilte cupcakes.

Fejl

- Database:
    - I servlet “presentation/Control.java”, linje 114, trækkes der penge fra User objektet i java, og derefter forsøges useren i databasen at blive opdateret. Man kunne forestille sig en situation hvor user objektet får fratrukket sin balance, men dao.updateUser fejler.
- Shoppingcart:
    - På nuværende tidspunkt vil den html-tabel, som bliver genereret i LogicControlleren, oprette duplikater af cupcakes i shoppingcart-tabellen. Metoden generateShoppingCart() opretter html-tabellen ud fra LineItems. LineItems, som består af *Topping, Bottom* og *Quantity*, bliver oprettet når brugeren trykker ‘addToCart’ i products.jsp. Her ville det være optimalt at kontrollere, om der i forvejen findes en cupcake med brugerens valg af *Topping* og *Bottom*. Hvis tilfældet er, at brugerens valg af *Topping* og *Bottom* er identisk med et LineItem, som i forvejen ligger i brugerens shoppingCart, skal *Quantity* blot lægges sammen.