**Indledning**

Systemet er udviklet til brug i en online cupcake shop. Der kan ved oprettelse af en bruger bestilles cupcakes, som efterfølgende kan afhentes på firmaets adresse.

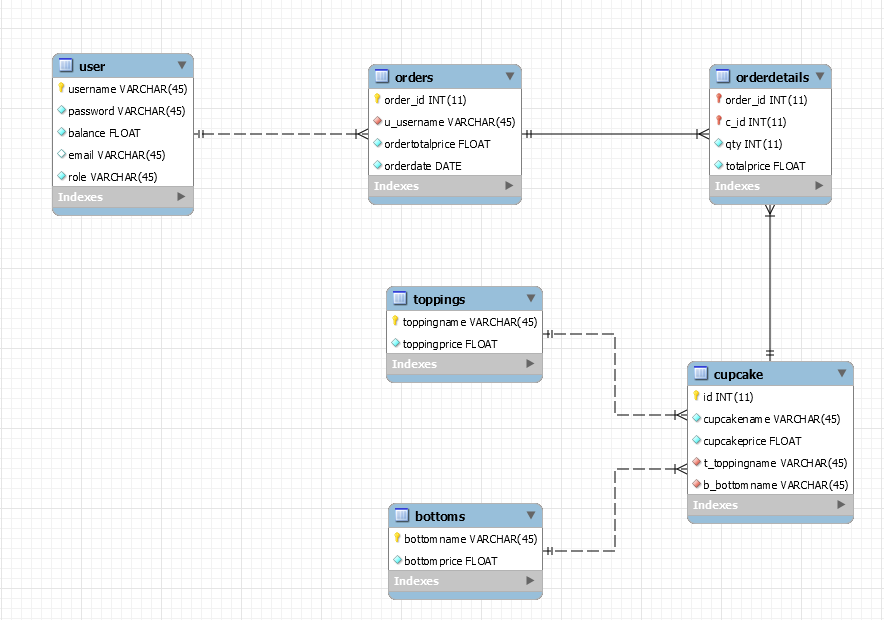
**Kundes krav:**

* Brugeren skal kunne oprette en bruger med tilknyttet saldo til køb af cupcakes.
* Brugeren skal logge ind og have nok kredit på sin saldo for at kunne gennemføre en ordre.
* En cupcake består af en bund, en top og et antal.
* Brugeren skal kunne se sine fakturerer fra tidligere bestillinger.
* Admin login, hvor alle brugernes fakturerer er listet, men også kan ses i detaljer.

**Teknologi valg:**

* Netbeans IDE 8.2
* MySQL Workbench 6.3 CE
* Visual Paradigm 15.1
* Microsoft Visio 2016 Pro

ER diagram



ER diagrammet illustrerer måden hvorpå systemet er sammensat af entiteter og relationer. Når en bruger afgiver en ordre, vil valgte toppe og bunde blive omdannet til cupcake objekter, som indsættes i databasen, til brug i tabellen orderdetails, som repræsentere brugerens faktura.

Cupcake tabellen er indsat således, at tidligere fakturas angivne priser ikke påvirkes ved en evt. prisændring senere.

**Beskrivelse af primærnøgler:**

Attributterne ***bottomname*** og ***toppingname*** er primærnøgler af datatypen VARCHAR. Ift. at virksomheden selv skal kunne vælge deres salgsvarer, og brugeren skal kunne forstå indholdet af valgt bund eller top, er disse primærnøgler ikke autogenereret. Dette forhindre oprettelse af identiske navne.

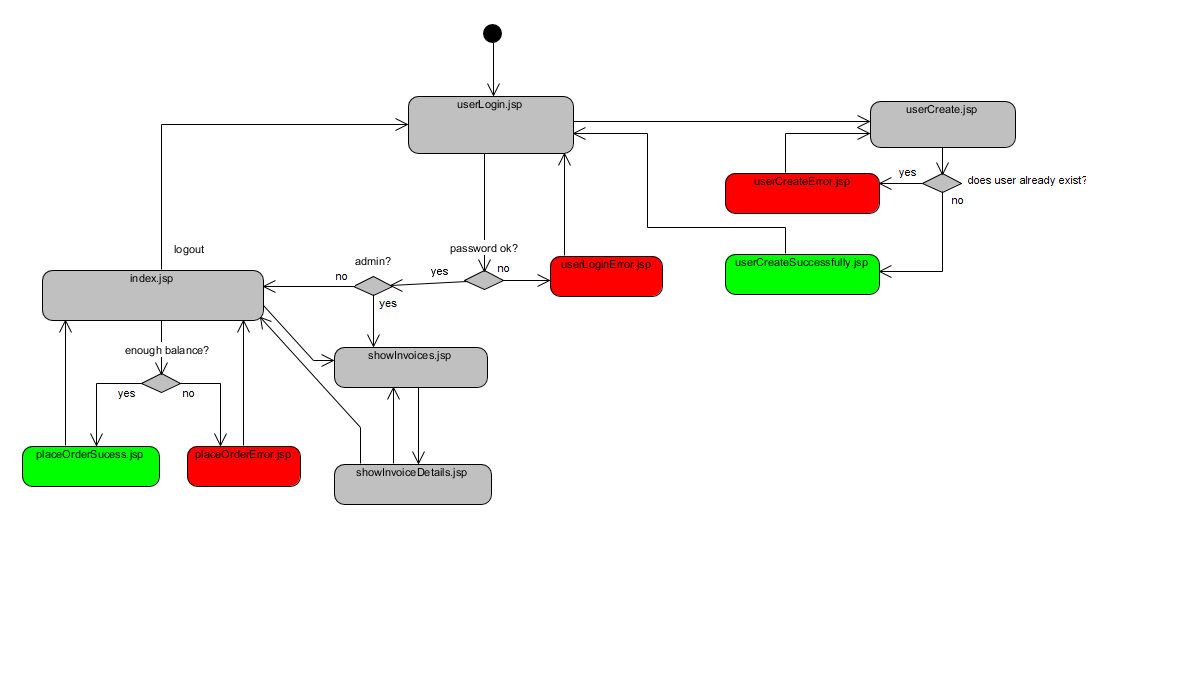
Attributten ***username*** er af hensyn til brugeren også af datatypen VARCHAR og autogenreres ikke, da det kan være til gene for brugeren ikke selv at kunne vælge sit brugernavn. Dette forhindre oprettelse af identiske brugernavne.

Attributterne ***order\_id*** og ***c\_id*** er begge fremmednøgler til hhv. ***orders*** og ***cupcake*** tabellen. De to fremmednøgler anvendes som en sammensat primærnøgle for ***orderdetails*** tabellen.

Attributten ***t\_toppingname*** i tabellen ***cupcake*** er fremmednøgle til tabellen ***toppings***, dette er gjort således, at der ikke er risiko for oprettelse af en cupcake med en top, som ikke eksisterer ***toppings*** tabellen.

Attributten ***b\_bottomname*** i tabellen ***cupcake*** er fremmednøgle til tabellen ***bottoms***, dette er gjort således, at der ikke er risiko for oprettelse af en cupcake med en top, som ikke eksisterer ***bottoms*** tabellen.

Navigationsdiagram



Navigations diagrammet illustrerer brugerens

**Implementeret:**

Som det fremgår af diagrammet har både admin og bruger adgang til ***showInvoices.jsp***. Siden bliver dog generet ift. om det er hhv. en admin eller en bruger der et logget ind. Brugeren får kun adgang til sine egne fakturaer, mens adminen har adgang til at se alle brugeres fakturaer.

**Login:**

MEDDELELSE:

**User:**Det er bygget op således at en User er i kontakt med *’login.jsp’* siden.   
Hvis en User skal logge på, skal der angives et ’*username*’ og ’*password*’.

**Login.jsp:**Trykkes der på login – sendes der et http POST request til FrontControlleren med origin parameter = loginUser.  
  
**FrontController:**For at mindske kald til databasen fremadrettet, så kaldes metoderne *getAllToppings(); & getAllBottoms();* . Det resultere i at toppings & bottoms til cupcakes er tilgængelig i sessionen, når der bestilles cupcakes.   
Ydermere sørger FrontControlleren også for at kalde metoden *checkPassword();* som sikre at user’s oplysninger stemmer overens med det som ligger i databasen.

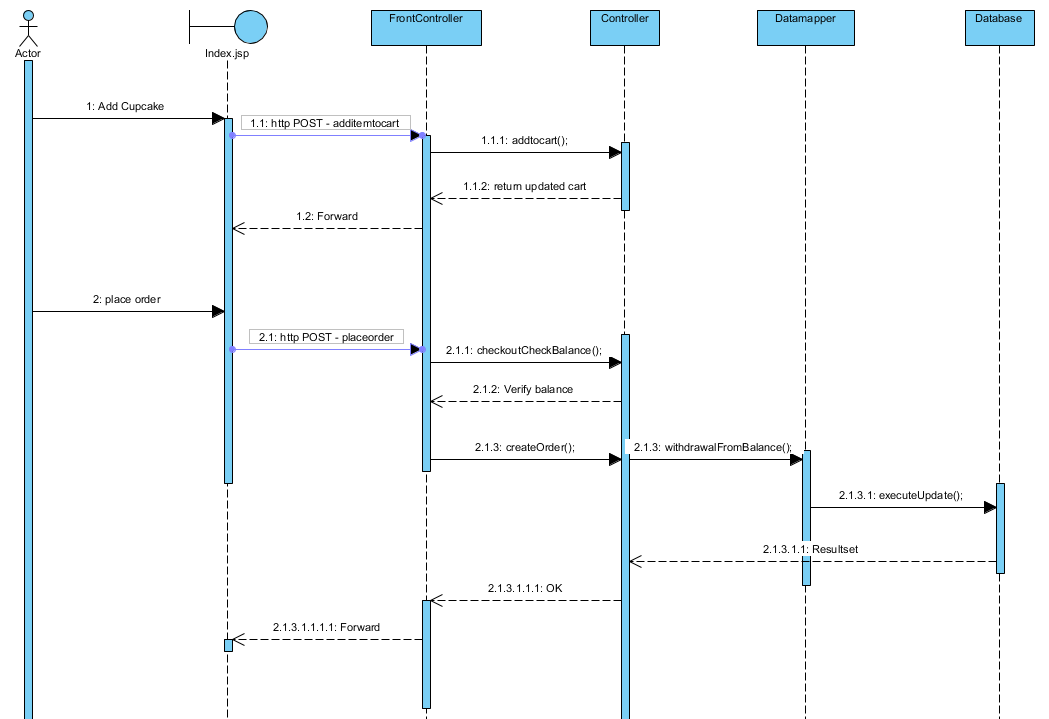
**Controller:**Controlleren kalder de nødvendige metoder i datamapperen – i dette tilfælde skal der bruges et userObject samt lister af toppings og bottoms.  
  
**Datamapper:**Sørger for at oprette forbindelse til databasen og køre de queries som blev forespurgt.

RETUR MEDDELELSE:

**Database:**Sender to ResultSet tilbage  
  
**Controller:** Controlleren modtager to ResultSet

* ResultSet – User  
  Der tjekkes hvorvidt password stemmer overens med det som vi har liggende i databasen på den specifikke user.   
  Når valideringen er godkendt sendes det videre til FrontControlleren
* ResultSet – Toppings & Bottoms  
  Dette ResultSet bliver lavet til to lister, som gives videre til FrontController

**FrontController:**Forwarder begge dele til JSP siden



MEDDELELSE:

**Bruger:**

Brugeren har valgt en bund og en top, samt antal.

**index.jsp:**Brugeren trykker på ”Add to cart” – sendes der et http POST request til FrontControlleren med origin parameter = ”addItemToCart”.  
  
**FrontController:**FrontControlleren videregiver de valgte oplysninger på cupcaken til Controller.

**Controller:**

Controlleren kører metoden addToCart();, og returnere herefter den opdaterede kurv, bestående af lineItems og totalpris .  
  
**FrontController:**

Frontcontrolleren forwarder den opdaterede kurv til index.jsp.

**Bruger:**

Brugeren trykker på place order.

**Index:**

Der sendes et http POST request til frontcontrolleren med origin parameter ”placeOrder”.

**FrontController:**

Frontcontrolleren videregiver brugerens saldo til controlleren.

**Controller:**

Controlleren kører metoden checkoutCheckBalance(); som verificer brugerens saldo.

**FrontController:**

Frontcontrolleren videregiver om saldoen er verificeret og får controlleren til at kører metoden createOrder() og withdrawalFromBalance();

**Controller:**

RETUR MEDDELELSE:

**Database:**Sender to ResultSet tilbage  
  
**Controller:** Controlleren modtager to ResultSet

* ResultSet – User  
  Der tjekkes hvorvidt password stemmer overens med det som vi har liggende i databasen på den specifikke user.   
  Når valideringen er godkendt sendes det videre til FrontControlleren
* ResultSet – Toppings & Bottoms  
  Dette ResultSet bliver lavet til to lister, som gives videre til FrontController

**FrontController:**Forwarder begge dele til JSP siden

De fleste programmører kan læse de enkelte metoder i et program, mens det kan være svært at skabe sig et overblik over hvordan programmet virker på overordnet plan. Et sekvens diagram bruges til at vise hvordan et typisk forløb foregår, eller til

*med sekvensdiagrammer, man vælger altid nogle interessante eksempler*.

Et eksempel på et typisk forløb kunne være at brugeren præsenteres for indkøbssiden. Her skal der vises følgende:

* Selve diagrammet, startende med jsp-siden eller servlet.
* Brug de rigtige klassenavne og metode navne
* Undlad argumenter til metoderne

I forklaringen til diagrammet skal du særligt lægge vægt at beskrive hvilke grene af if-sætninger der er brugt i de enkelte metoder.

Særlige forhold

Dette afsnit bruges til at beskrive særlige forhold der benyttes i programmet. Det kan f.eks. være:

* Hvilke informationer gemmes i session
* Hvordan håndterer man exceptions. Det kommer vi tilbage til senere i semesteret.
* Hvordan man har valgt at lave validering af brugerinput.
* Hvordan man har valgt at lave sikkerhed i forbindelse med login.
* Hvilke brugertyper, der er valgt i databasen, og hvordan de er brugt i jdbc
* … andre elementer

Husk: det er bedre med 2 linjers dokumentation end ingen.

Status på implementation

Dette afsnit skal liste hvor langt man er nået med implementationen. Typiske ting man kan have sprunget over er:

* Man har ikke nået at lave alle de jsp sider man har med i navigationsdiagrammet.
* Man har ikke nået at lave alle CRUD metoderne til alle tabellerne
* Man har ikke fået stylet sine sider
* Man har fundet en fejl “i sidste øjeblik”, men har ikke haft tid til at rette det. - F.eks. at man har brugt session forkert, sådan at man på en af siderne kan komme ind uden at være logget ind.
* tests der fejler på afleveringstidspunkt
* … andre mangler