Navn	*Mail*	*GitHub*
Joachim Justesen	*Cph-jj431@cphbusiness.dk*	*jojus1101*
Shpat Jakupi	*Cph-sj389@cphbusiness.dk*	*shpatjohn*
Jacob Jabr	*Cph-jj433@cphbusiness.dk*	*JacobJabs*
Syed Hamad Shah	*Cph-ss451@cphbusiness.dk*	*Hamadcph*

Cupcake

Link til GitHub

https://github.com/shpatjohn/CupCake

Alle medlemer er fra B-klassen

Indholdsfortegnelse

Indledning:

Dette projekt omhandler en opgave, om at ”lave” en Cupcake forretning i form af at oprette en tabel med kunder i Cupcake forretningen, som skal indeholde informationer som Username, Password og Balance. Samt skal der laves tabeller af Bottoms og Toppings af Cupcakes, med pris for at kunne vise produkter og lave bestillinger. Det skal udarbejdes i JSP og have en servlet til at kontrollere alle JSP funktionerne sådan at det fungere optimalt, samt i samarbejde med diverse javaklasser, heri DAO-klasse(r), som står for Data Access object også kendt som mapper.

Baggrund:

- Virksomheden som skal bruge systemet er en Cupcake butik.
  De skal bruge systemet til at kunder kan have oversigt over deres kunder og registrere nye kunder. De kunder og potentielle nye kunder skal have mulighed for at kunne se menuen, samt kunne sammensætte cupcakes sammen til deres ønskede ordre, og betale via systemet. Selve de cupcakes de tilbyder består af top og bunde, og skal derfor have et sted at opbevare oplysninger om Cupcakes og deres kunder.
  De ønsker også at kunderne har mulighed for at kunne indsætte og ændre deres ”balance” inde på system til betaling af deres bestillinger. Alle de informationer om kunder skal kunne gemmes på en database og det gælder også de forskellige cupcakes smage, sådan at hvis virksomheden får udvidet deres smage, nemt kan indsætte dem inde på hjemmesiden, prissætte dem, eller evt. Hvis de føler at de sælger deres varer for billigt, kan højne prisen eller omvendt, dette kan også gøres nemt.

Teknologiske valg:

Det som er brugt under udarbejdelsen af systemet.

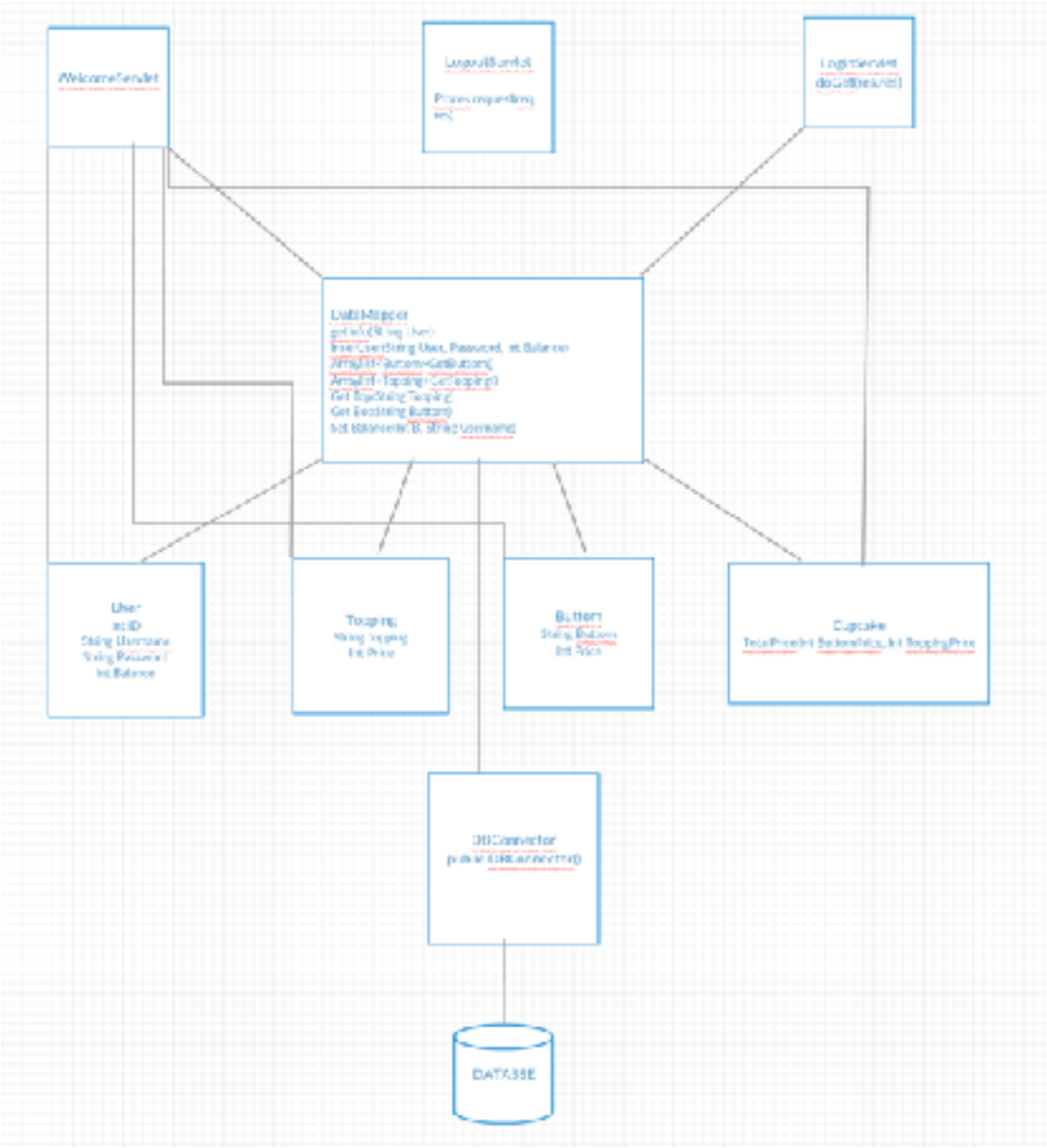
- NetBeans IDE 8.2: Til at skrive Java, HTML & CSS

- MySQL Workbench 8.0.15: Database for User & Cupcake informationer.
- JDBC 8.0.15: I form af vores Connector.
- Apache Tomcat 8.0.27.0: Server vi kører på.

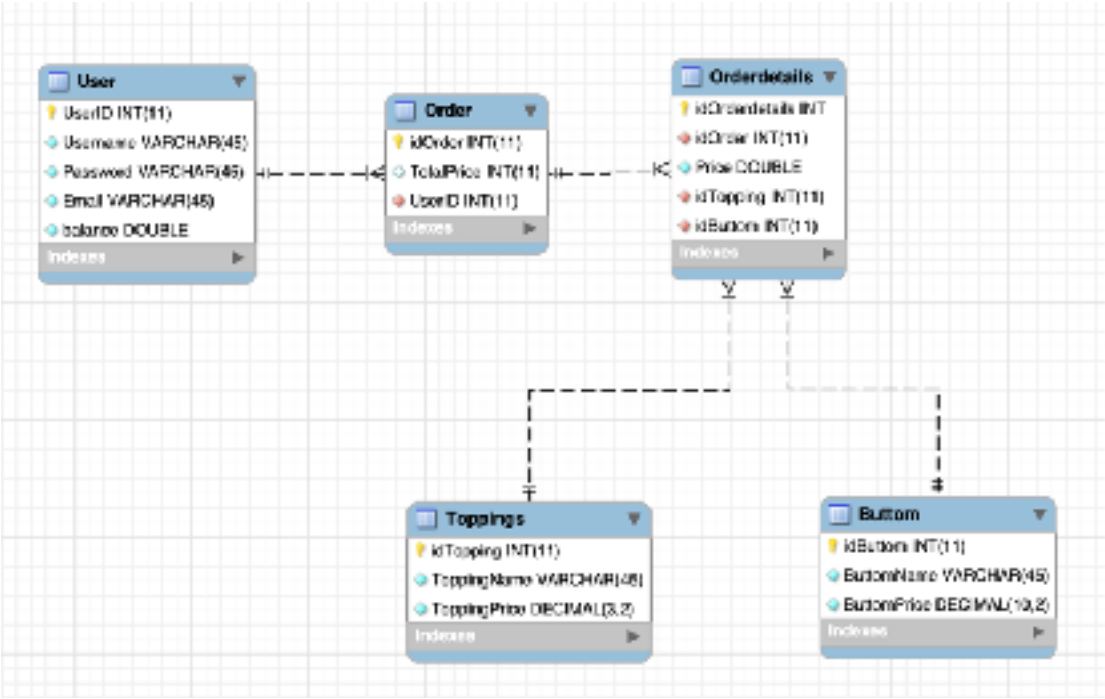
Krav:

De krav virksomheden stiller til systemet, er at de får adgang til et system, hvor kunderne har mulighed for at oprette sig selv, som bruger, og efterfølgende direkte kan logge sig ind, efter man er registreret som bruger i databasen. Derefter at kunne lave bestillinger, samt se hvilke muligheder der er at vælge mellem og prisen på disse muligheder. Desuden skal det gøres muligt for virksomheden at have adgang til at se alle deres kunder, samt se hvilke produkter de har gennem en database, som i dette tilfælde er en mySQL-workbench.

Domænemodel & ER diagram:



I ovenstående domænemodel, ser vi hvordan hele processen og samarbejdet foregår mellem de forskellige klasser og databasen. Vi ser hvordan DataMapper, er vores helt store spiller, som har en finger med i hele spillet. Alt fra servlets og Databaser, fordi man gennem den opretter de forskellige ting, såsom en user, som dernæst bliver kastet videre til vores database og indsat derinde, som man har den stående.



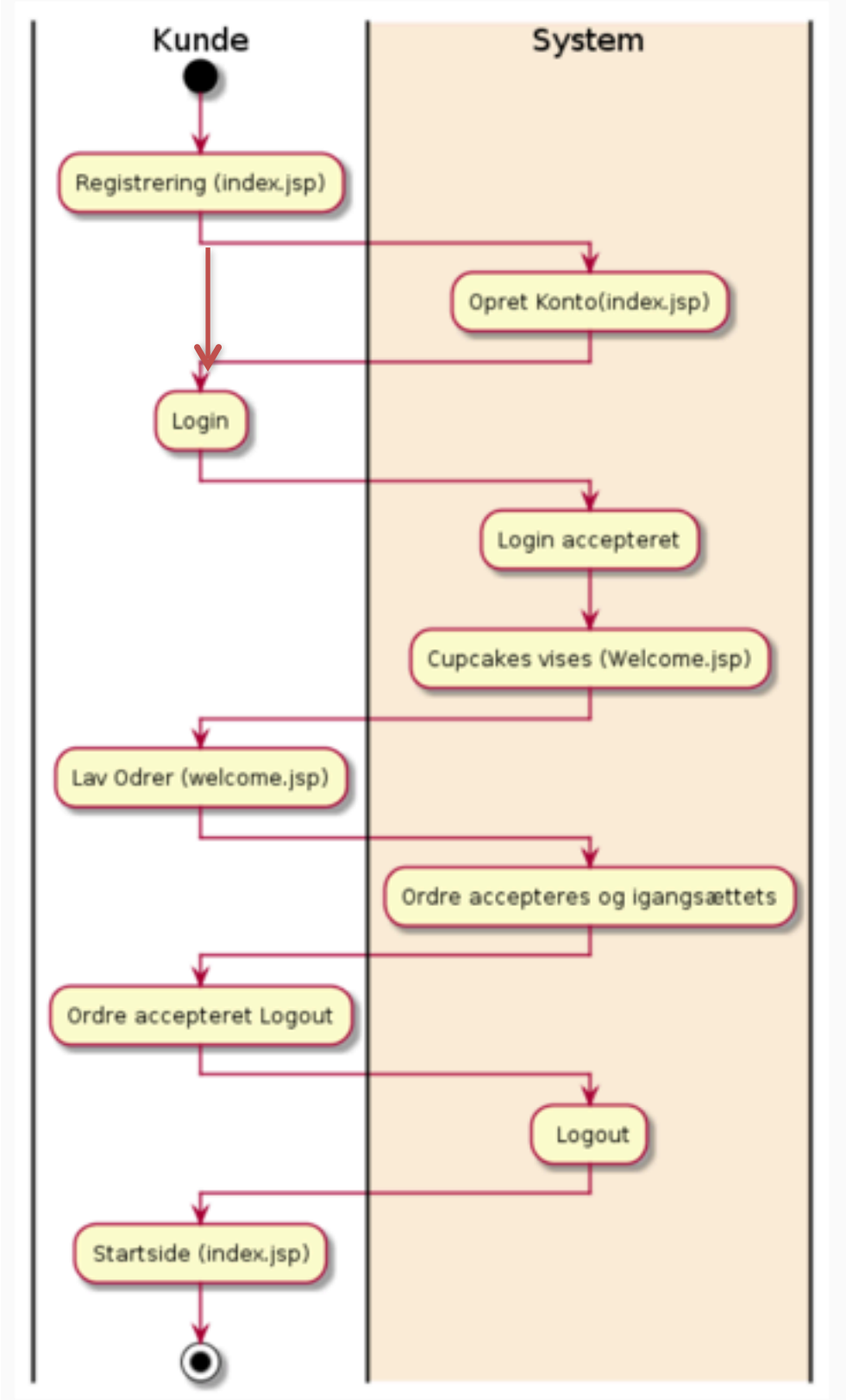
User har en 1 til mange relation til Order i den forstand at den enkelte User godt kan lave flere ordre.
Det er UserID som er Forreign Key mellem de to tabeller.
Order har en 1 til mange relation til Orderdetails.
Forreign Key idOrder til Orderdetails.
Der er mange til 1 relation i mellem Oderdetails og Buttom og Toppings. Det er sådan fordi i Orderdetails kan der være lavet flere forskellige Cupcake kombinationer.
De Forreign Keys der er i tabellerne er lavet, som værende de mest praktiske Keys til at skabe relationer imellem hinanden uden at skabe forvirring og mulige problemer.
I vores system er der ikke blevet implementeret en klasser for tabellerne Order og Orderdetails, men de 2 tabeller er essentielle i form af at det overordnede system, som virksomheden ønsker skal dette være i, og hvis et ER diagram blev lavet over de ting, som vi har implementeret i vores system ville der ikke være nogle relationer mellem tabellerne. Derfor er de blevet indsat i den form vi ville kunne se et færdigt ER diagram over opgaven.

Navigationsdiagram:

Kode til navigationsdiagram. Dette kan bruges på plantuml.com:
@startuml

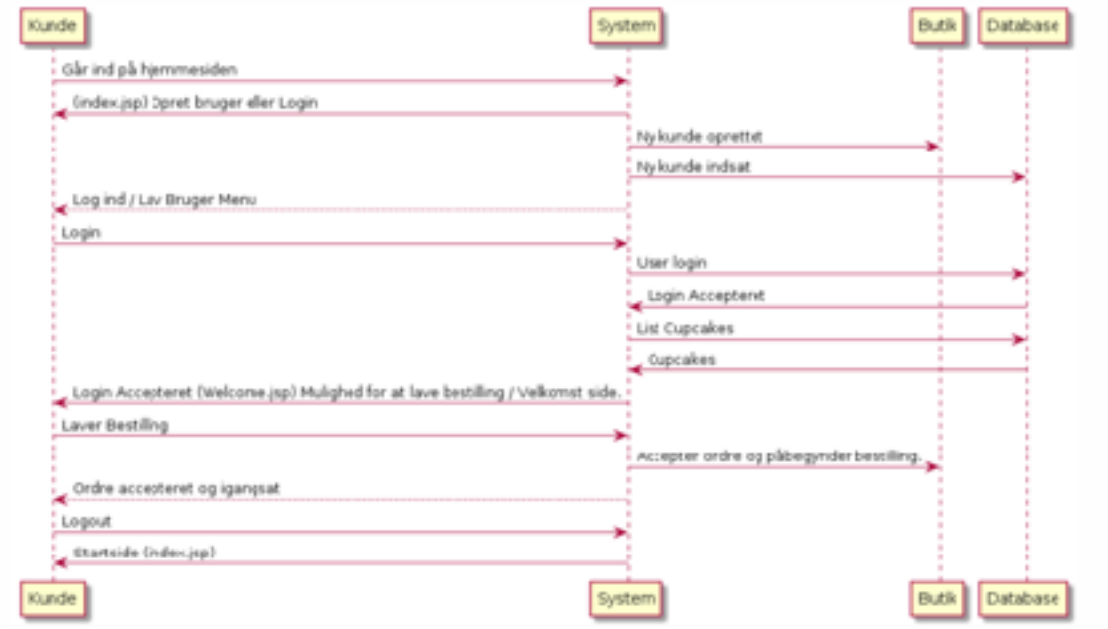
```
|Kunde|
start
:Registrering (index.jsp);
|#AntiqueWhite|System|
:Opret Konto(index.jsp);
|Kunde|
:Login;
|#AntiqueWhite|System|
:Login accepteret;
:Cupcakes vises (Welcome.jsp);
|Kunde|
:Lav Odrer (welcome.jsp);
|System|
:Ordre accepteres og igangsættets;
|Kunde|
:Ordre accepteret Logout;
|System|
: Logout;
|Kunde|
:Startside (index.jsp);
stop
@enduml
```

Alle servlets bliver brugt under navigationsdiagramet. De har hver deres formål i kundes proces om at lave en bestilling af cupcakes.


Kunde
System
Registrering (index.jsp)
Opret Konto(index.jsp)
Login
Login accepteret
Cupcakes vises (Welcome.jsp)
Lav Odrer (welcome.jsp)
Ordre accepteres og igangsættets
Ordre accepteret Logout
Logout
Startside (index.jsp)

Sekvensdiagram:

Kode som kan bruges i på Plantuml.com :
@Startuml
Kunde -> System: Går ind på hjemmesiden
System-> Kunde: (index.jsp) Opret bruger eller Login
System -> Butik : Ny kunde oprettet
System -> Database: Ny kunde indsat
System --> Kunde: Log ind / Lav Bruger Menu
Kunde -> System: Login
System -> Database: User login
Database -> System: Login Accepteret
System -> Database: List Cupcakes
Database -> System: Cupcakes
System -> Kunde: Login Accepteret (Welcome.jsp) Mulighed for at lave bestilling / Velkomst side.
Kunde -> System: Laver Bestilling
System ->Butik: Accepter ordre og påbegynder bestilling.
Kunde <-- System: Ordre accepteret og igangsat
Kunde-> System: Logout
System -> Kunde: Startside (index.jsp)
@Enduml



Særlige forhold:

Validering af brugerinput;

```
String uname = request.getParameter("uname");
String pass = request.getParameter("pass");


if (uname != null && pass != null && uname.equals(data.getInfo(uname).getUsername()) && pass.equals(data.getInfo(uname)
{
    HttpSession session = request.getSession();
    session.setAttribute("balance",data.getInfo(uname).getBalance());
    session.setAttribute("username", uname);

    response.sendRedirect("/CupCake/welcome.jsp");
}
else
{
    response.sendRedirect("/CupCake/index.jsp");
}
```

Som det kan ses ud fra koden handler det om at det input som bruger indtaster matcher med det som er i databasen. Hvis bruger inputtet ikke matcher det som skulle være deres login vil de blive henvist tilbage til start siden og kan derefter selv tage beslutning om de vil lave et nyt login eller logge ind igen.
Det kan samtidig også ses at det er herfra at hvis brugeren indtaster den korrekte information, vil vedkommende blive henvist til velkomst siden.

Brugertyper:

I vores system er der ind til videre kun ind form for brugeren som er kunden eller Useren som ikke har nogle roller eller noget. Det kan senere videre udarbejdes at der kan være admins, som kan have overblik over bestillinger og faktura, men som bruger typer ville de stadig høre under en User, men i stedet få tildelt en rolle, som i deres tilfælde ville blive admin, og samtidig ville de også komme i samme tabel i databasen, under User.
Det er vores DB Connector som finder brugertypen User i Databasen og ved den connection gør at en bruge kan blive oprettet inde i Netbeans og blive brugt som en kunde og igen indsat i Databasen.

Status på implementation:

Opgave beskrivelse:
https://docs.google.com/document/d/1DH8Apv6kSGZJc_hFKnIRcaw96WFEwf7YJZNpnPjV_E8/edit

Ud fra de ting som er det endelig produkt er alle de ting som skule implementeres ikke alt sammen blevet implementeret. Det som fungerer er en succesfuld Login og registrering af bruger, samt et overblik over de Cupcake kombinationer som er mulige at lave, eller bedre sagt menu-kortet. Man kan desuden se, hvad den endelig pris angivet i dollars er efter valget af de forskellige toppings og bottoms.
Ud fra opgavebeskrivelsen er det de grønne opgaver som er blevet implementeret, men ikke endvidere andet som er blevet implementeret.
Ud fra opgavebeskrivelsen er de manglende opgaver 5, 6, 7 og 8 som ville kunne implementeres med mere tid, og bedre koordination.

De mangler der er prioriteret for senere implementation er at få hele ordre systemet op at køre så kunderne ville kunne lave en bestilling og samt have alle de muligheder der høre til, som at indsætte valuta på deres individuelle balance, lave en Shoppingcart funktion, og samt tildele dem en rolle. Som i brede ord at lave alle opgavens skridt.

Test:

Der skal være lavet test. Vi har ikke i dette projekt haft fokus på at teste, så det skal ikke med. Du skal dokumentere tests ved at beskrive i tabel form: