CUPCAKE SHOP

PREPARED FOR

Ronnie Dalsgaard

Thomas Hartmann

Kasper Østerbye

Copenhagen Business Academy

PREPARED BY

Dan Tomicic

Esben Dalgaard

Casper Marx

Kim P. Pedersen

DECK Coding Solutions IVS

l18dat2ae

Oktober, 2018

Indholdsfortegnelse

1. Indledning

Projektet omhandler opsætning af en Java drevet side, primært visualiseret i JSP og på Apache Tomcat servlet. Desuden anvender projektet også en Linux droplet server med MySQL kørende til opbevaring af fast data, samt arkivering af ordrer, ordre detaljer og købere.

Cupcake butikken omtales i rapporten som kunden. Cupcake butikkens kunder beskrives som brugere. Eventuelle ansvarlige for projektet, samt nuværende folk på projektet beskrives som programmørerne. Vedligeholdere eller fremtidig styring af det færdige projekt beskrives som administratorer.

2. Baggrund

Projektet er lavet med henblik på at forme en webshop til en bager, specifikt med 'cupcakes' for øje. Kunden er derfor kun interesseret i sammensætning og salg af cupcakes på webshoppen. Andet bagværk er ikke nødvendigt, men kan overvejes implementeret senere.

Cupcakes siden har følgende krav:

- Der skal være mulighed for at logge ind som en bruger. Brugeren skal kunne oprettes og gemmes i databasen så de kan logge ind igen og kædes sammen med deres ordrer.
- Cupcakes produktet skal bestå af en bund samt én type pynt på toppen af kagen. Forskellige bunde og typer af pynt kan også have forskellige priser.
- Køberen skal have en balance de kan købe for, og der skal være et arkiv over deres ordrer og hvad de indeholder.
- Køberen skal have en kurv med varer, og gerne en der knyttes til sessionen, så den ikke forsvinder pludseligt. Den skal kunne checkes ud og omdannes til en ordre.
- Ordrer og produkter skal ligge i en database for sig.

3. Teknologivalg

Til udvikling af Cupcake webshoppen benyttes følgende:

Backend

- MySQL ver. 14.14 for LINUX benyttes til database
- Java 1.8.0 og JSP på backend

Frontend

- HTML og CSS på frontend.

Software

- MySQL Workbench 6.3
- NetBeans IDE 8.2
- Apache Tomcat 8.0.27

4. Krav

Der ønskes udviklet en simpel webshop ved hjælp af MySQL database, java servlets og jsp sider på backend og html, css og javascript på frontend.

Webshoppen sælger cupcakes, men kun som afhentning. Brugere skal kunne bestille en ordre og derefter hente deres cupcakes. Sending af cupcakes i posten har vist sig at være en dårlig ide.

Bageriet har en meget hurtig cupcake maskine, så i det øjeblik hvor ordren er placeret, er Cupcakes klar til afhentning.

Cupcakes har en bund og en topping. Disse kan kombineres på mange måder, men en cupcake skal altid have både bund og en topping. Bottoms og toppings findes i bilag A.

Bilag A.

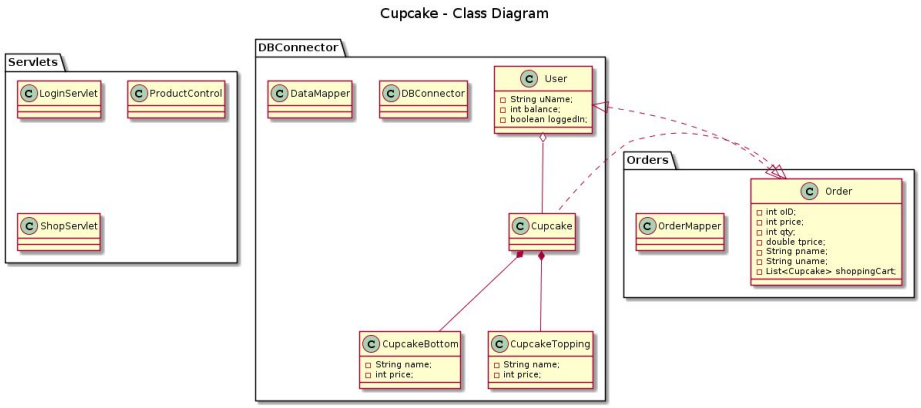
Topping	Price	Bottom	Price
Chocolate	$ 5,00	Chocolate	$ 5,00
Blueberry	$ 5,00	Vanilla	$ 5,00
Rasberry	$ 5,00	Nutmeg	$ 5,00
Crispy	$ 6,00	Pistacio	$ 6,00
Strawberry	$ 6,00	Almond	$ 7,00
Rum/Raisin	$ 7,00		
Orange	$ 8,00		
Lemon	$ 8,00		
Blue cheese	$ 9,00		

Brugerne har hver en konto i webshoppen og ordrer kan kun placeres, hvis kontoen har nok penge til at dække prisen. Betalingen er behandlet ved et andet system, og hvis depositum skal behandles manuelt i databasen, bliver der tilbagebetalt, når cupcakes bliver bestilt.

For at kunne betale med deres konto skal brugerne bruge et brugernavn og et kodeord til at logge ind, før de bestiller.

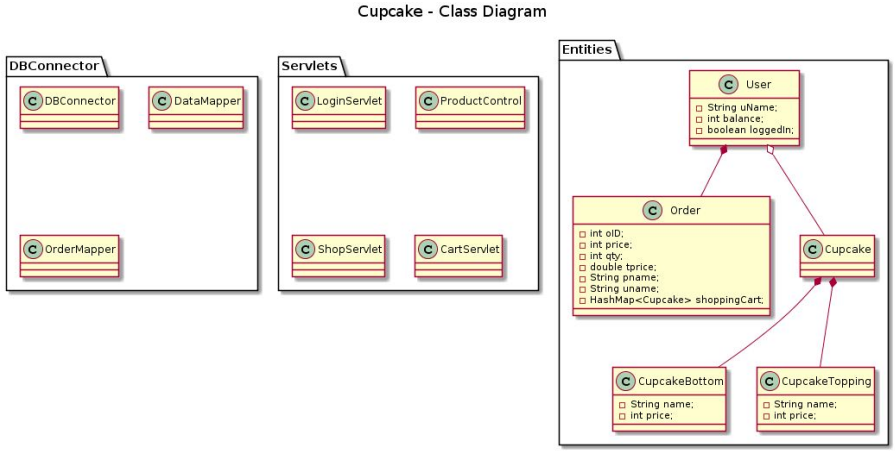
5. Domæne model og ER diagram

Domæne model



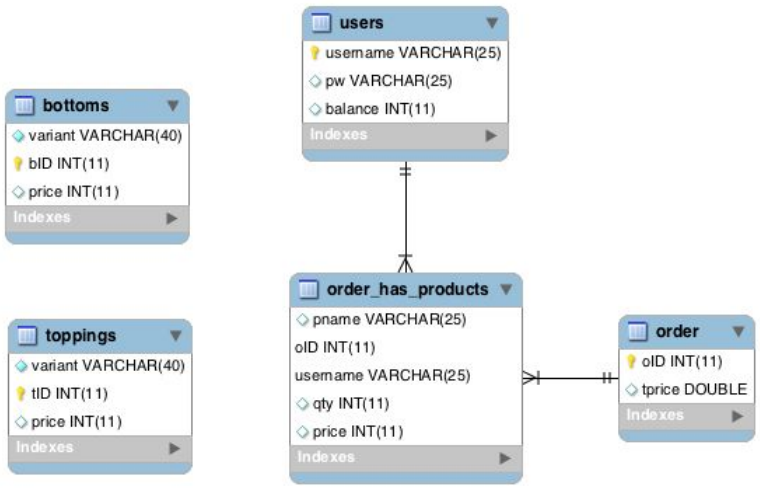
Figur 1: Kan også ses her: http://www.deck-cs.dk/diagrams/domainmodel.png

Cupcake systemet er opbygget af 3 packages; `Servlets`, `DBConnector` og `Orders`. Orders har sin egen separate package, hvilket kun er sket fordi vi ønskede at udvikle på `DataMapper` og `OrderMapper` på samme tid. Optimalt ville vi organisere det mere som det ser ud nedenunder i figur 2.



Figur 2: Kan også ses her: http://www.deck-cs.dk/diagrams/domainmodel2.png

ER Diagram

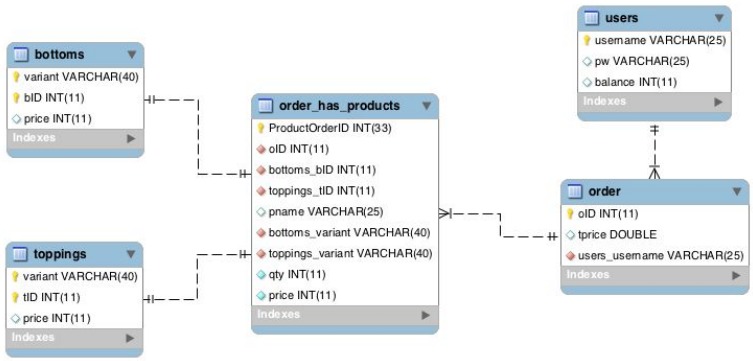


Figur 3: Kan også ses her: http://www.deck-cs.dk/diagrams/DB-ERmodel.png

Databasen er opbygget af 5 tables med flere columns. Der er `order_has_products` som en tabel der er bygget op af info flettet fra `order` og `users`, således flere genstande godt kan være kædet til samme bruger og ordre. Der kan derfor godt være flere cupcakes med samme navn i tabellen, men de burde have forskellige ordrer.

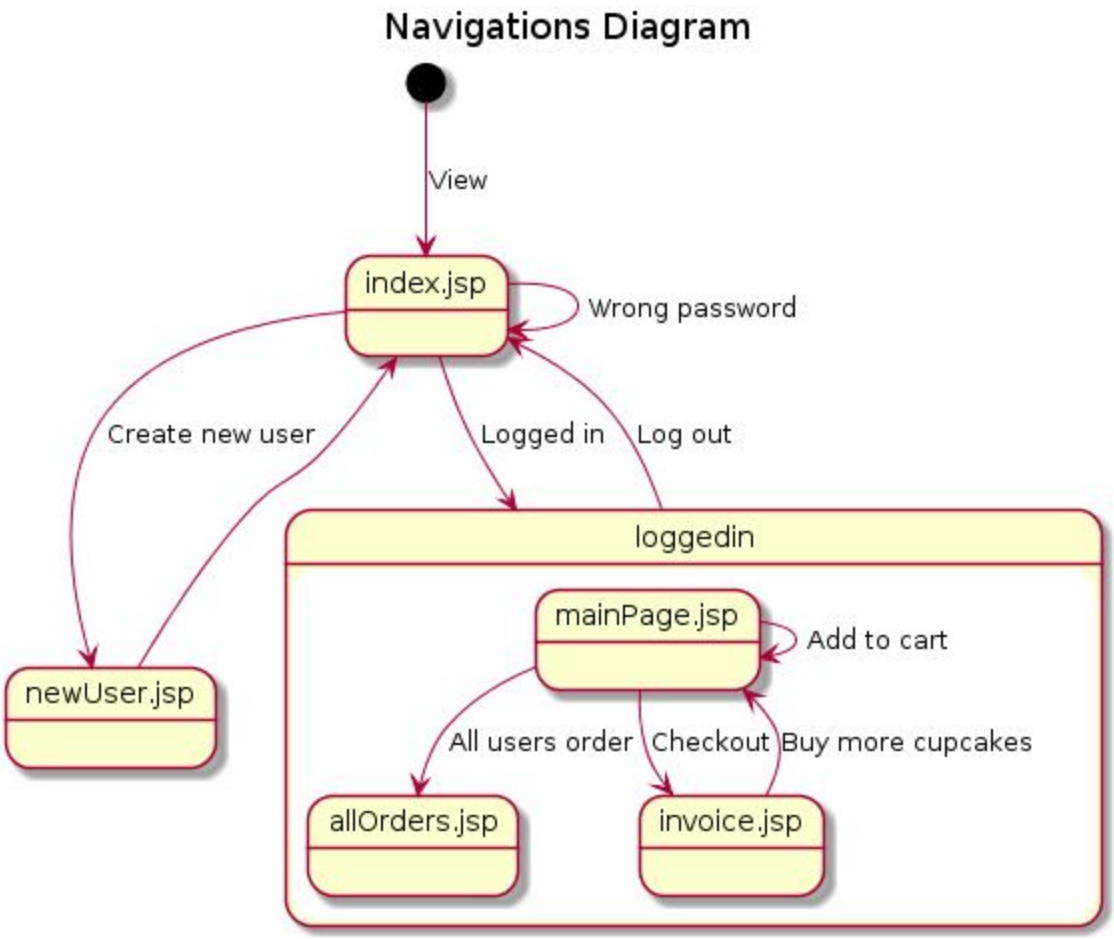
Ideelt kan man ændre `order_has_products` til at lave en primary key, der er en kombination af `toppings tID` og `bottoms bID` samt `order oID`, hvilket ville gøre at databasen holder sin egen integritet.

Der er også valgt at bruge `username` som primary key, da vi så kan anvende det til at undgå redundans i vores brugernavne.



Figur 4: Kan også ses her: http://www.deck-cs.dk/diagrams/CupcakeDBNew.png

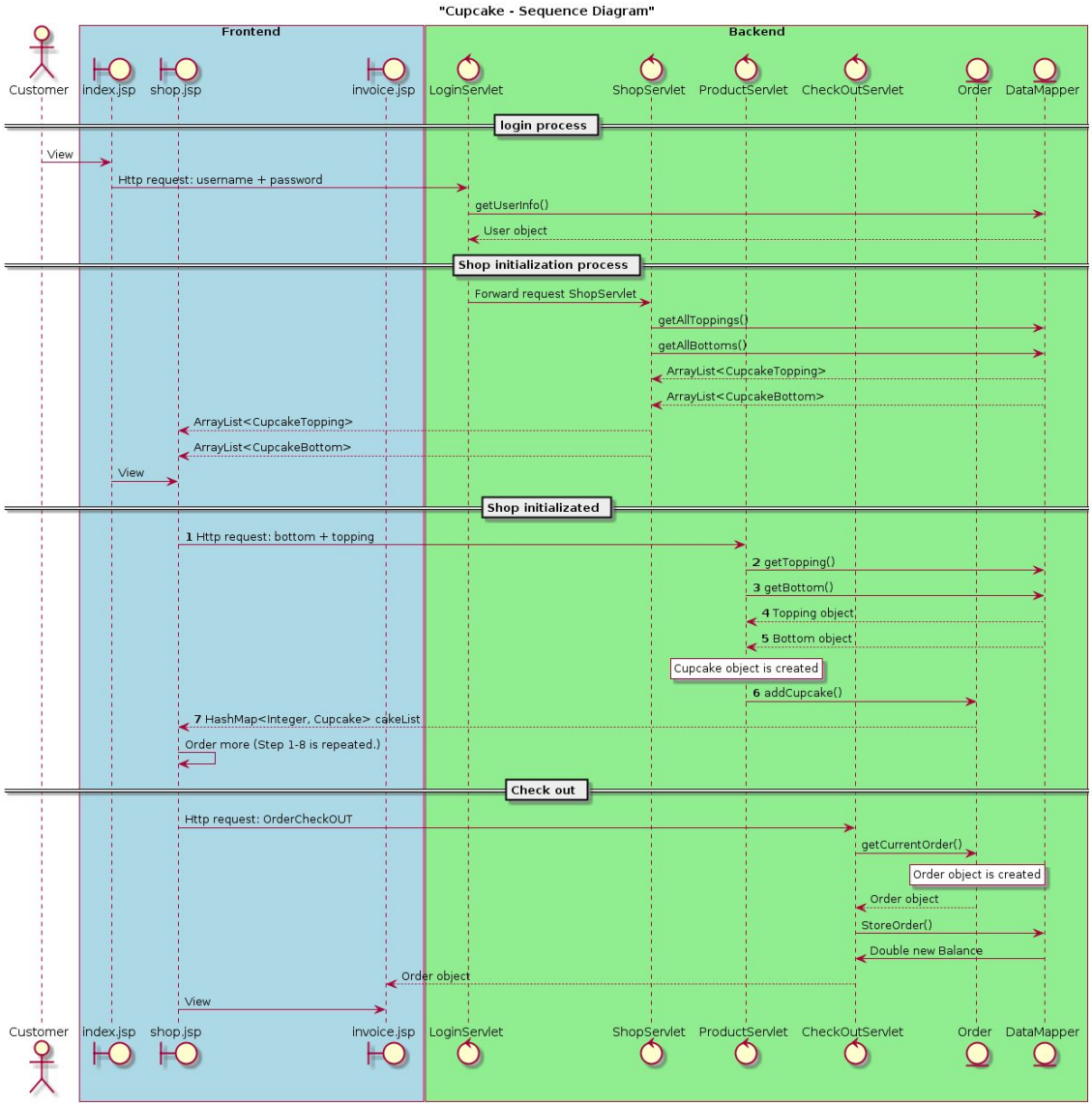
6. Navigationsdiagram



Figur 5: Kan også ses her: http://www.deck-cs.dk/diagrams/navigationsdiagram.png

Siden er delt op i om man er logget ind eller ej. Alt udenfor 'loggedin' feltet arbejder mod at få sat brugerens session til at få en 'loggedin' status, så at de kan tilgå shoppen og handle. Når brugeren har opnået dette, kan man få adgang til `invoice.jsp`. Fremtidigt bør der også implementeres en administrator mulighed, med flere muligheder.

7. Sekvensdiagrammer



Figur 6: Kan også ses her: http://www.deck-cs.dk/diagrams/sequencediagram.png

Startsiden for systemet `index.jsp`, indeholder vores login prompt. Dette er gjort mht. at `shop.jsp` kun vises hvis køberen er logget ind. Login processen er bygget op af et MySQL query der bliver sendt til databasen via `LoginServlet`, hvor der foretages et check på brugernavn og adgangskode. Adgangskoden bliver kun hentet i tilfælde af, der findes et brugernavn der er lig med det indtastede.

`shop.jsp` er baseret på to drop-down menuer hvor køber kan vælge top og bund til deres cupcake, samt et tekstfelt til antal. Drop-down menuerne er dynamisk genereret ud fra de

toppe og bunde der findes i databasen. Hvis flere kager indsættes i databasen, bliver `shop.jsp` automatisk opdateret næste gang man åbner siden.

User objektet indeholder alt information omkring køber. Det indeholder brugernavn, saldo, login status samt et Order objekt som indeholder alt information omkring ordren. Vha denne struktur sørges der for, altid at have kædet ordre og bruger sammen, hvilket også gør dem nemme at tilgå.

Som førhen skrevet genereres Cupcakes fra databasen; dette er også gældende for når et Cupcake objekt bliver oprettet. Således sørges der for, at der kun findes kager, som findes i databasen. Dette er vores sikring mod en korrupt ordre.

Indkøbskurven er valgt at laves som HashMap, så der kan være en nøgle (den enkelte Cupcake) og en værdi (antal). En anden mulighed havde været to lister der arbejdede sammen, men der blev besluttet at HashMap var en mere stabil løsning, da flere sammenkædede lister har tendens til at fejle.

8. Særlige forhold

Køberens navn, konto balance og login-status gemmes i et User objekt som parameter i sessionen fra starten af. Senere lægges også Order objekt til User objektet .

Der er et enkelt sted hvor der håndteres Exception med andet end at vi udskriver stacken. Når brugerens kodeord kontrolleres vil en fejl tilbagelevere et `User` objekt username “Error”. Dette kan bruges på et senere tidspunkt i koden til at behandle fejlen.

Der er ikke udbredt fokus på sikkerheden i det nuværende build. Køberens kodeord vises ikke i URL’en som parameter, men den ligger ukrypteret i databasen. Kodeordet vises dog ikke i html boksen når det tastet.

Der er brugt lidt mere varchar i databasen end de fleste nok ville foretrække, da vi har navnet på køberen som primary key, men ellers er de fleste andre genstande identificeret ved en int.

9. Status på implementation

`Order.java`

Den vigtigste mangel i funktionaliteten findes i `Order` objektet og måden det bliver håndteret og gemt på i databasen. Der findes en `OrderMapper` klasse som har udkommenterede metoder, men disse skal omskrives, før de har funktionalitet.

Som det er nu, findes `Order` objektet som indeholder et `orderID`, en `totalPrice` og et HashMap af Cupcake objekter som key og antal som value. Der mangler at blive skrevet metoder til omdannelse af `Order` objektet til simple datatyper samt lagring af disse i databasen via `DataMapper` objektet.

En anden mangel i shoppingCart, er at den kan lave duplikater på visningen af dine vare. Dette kunne udbedres vha input validering, men dette mangler i implementationen.

`invoice.jsp`

I flere af diagrammerne ses siden `invoice.jsp`. Denne er ikke lavet endnu, da den er tæt knyttet til `Order` objektets færdiggørelse. Optimalt skulle denne side vises efter køber har placeret en ordre. Her kunne en liste over alle andre ordrer, en køber har foretaget, også ligge - disse ordrer skulle være links, som så ville hente den specifikke ordre fra databasen.

Styling

Systemet er bygget med funktionaliteten i første række, og da der har været nogle udfordringer undervejs som har taget længere tid end forventet, er stylingen nedprioriteret.

10. Tidsplan/forbrug

Total estimat af timeforbrug: 181,5

Milestone	Tasks	Reporting	Hrs	Date
1 - Analyse/Planlægning				
1.1	Analyse og design fase, samle data og skabe system mockup	Ingen	20	17/09/18
1.2	Arkitektur design	Ingen	1	17/09/18
1.3	Udvikle arbejdsplan (uddeling af opgaver til udviklingshold)	Ingen	10	17/09/18
2 - Backend				
2.1	Opret database	Ingen	10	24/09/18
2.2	Importer eksisterende klientdata (database)	Ingen	2	24/09/18
2.3	Rens af klientdata (database)	Ingen	0,5	24/09/18
3 - Frontend				
2.4	Skab JSP sider	Ingen	30	24/09/18
2.5	Integration med klientdata (database)	Ingen	10	24/09/18
2.6	Integration af Tomcat på droplet	Ingen	8	24/09/18
4 - Testing				
3.1	Alpha test applikation	Ingen	25	01/10/18
3.2	Test af webshop funktioner	Ingen	25	01/10/18
3.3	Beta test (frivillige medarbejdere Casper, Esben, dan og Kim))	Ingen	20	01/10/18
3.4	Afslut dokumentation (rapport)	Ingen	20	05/10/18

NB. alle timetal er et estimeret og opgaver er fiktive. Fremadrettet vil der blive ført til tabel hver dag for dokumentation.

11. Test

Der er ikke blevet udført, eller implementeret tests i dette projekt.