Cupcake - Webprojekt

Gruppe: Krasnolud

Alessandro Accaroli, cph-aa264@cphbusiness.dk, Git: Alek2750

Martin Rasmussen, cph-mr373@cphbusiness.dk, Git: Swoop86dk

Nicolai Bagge Johansen, cph-nj146@cphbusiness.dk, Git: Jacktheman1993

Simon Kepp Stennicke, cph-ss409@cphbusiness.dk, Git: fb05

Klasse: B

Dato: 19-02-2018 til 18-03-2018

Indholdsfortegnelse

Indledning

I dette projekt er der stillet til opgave, at udvikle en webapplikation kaldet “CupCake” shop. Det skal være muligt, at kunne logge ind som ny bruger, og vælge hvilken type cupcake man har lyst til at købe. Ydermere, skal der være valgmuligheder i kombinationen på toppings og bottoms af cupcakes.

For at kunne udarbejde og stille dette projekt op bedst muligt skal der benyttes forskellige områder af ekspertise indenfor følgende: MySQL Database, HTML, CSS, Bootstrap, jdbc, servlets, jsp, server og java klasser.

Webshoppen ligger på en server, men kan tilgås lokalt ved at køre programmet. Databasen er forbundet på samme server så hjemmesiden henter og lagrer data på samme tid. For mest optimeret oplevelse henviser vi dog til at benytte et link til en IP-adresse hvor webapplikationen ligger.

Baggrund:

Dette system der bliver udviklet er til en konditor, som har brug for en nemmere og mere udbredt måde hvorpå der kan sælges cupcakes, samtidig med der kan holdes styr på hvilke varianter som er populære blandt kunderne til konditoren.

De krav der er stillet i kontrakten fra konditoren er:

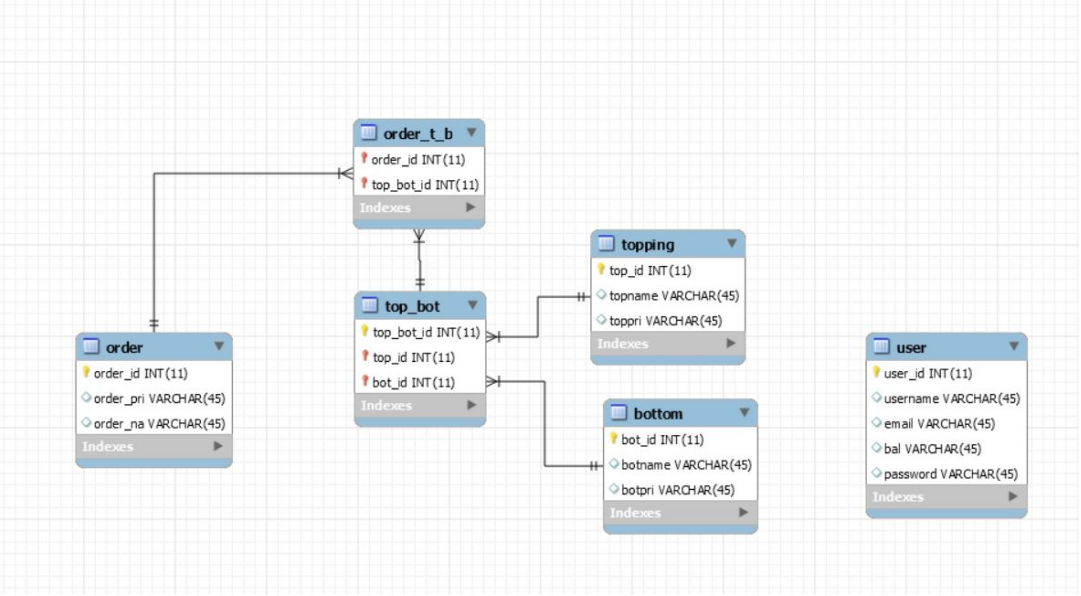
- Kunden skal kunne logge in.
- Kunden skal kunne oprette en konto.
- Kunden skal kunne ændre på sin konto.
- Kunden skal kunne bestille forskellige varianter af toppings og bottoms.
- Det skal være nemt og overskueligt at navigere rundt på online butikken for kunden.
- Der skal være en personlig indkøbskurv.
- Kunden skal kunne betale for sin bestilling via en sikker session.

Teknologiske valg:

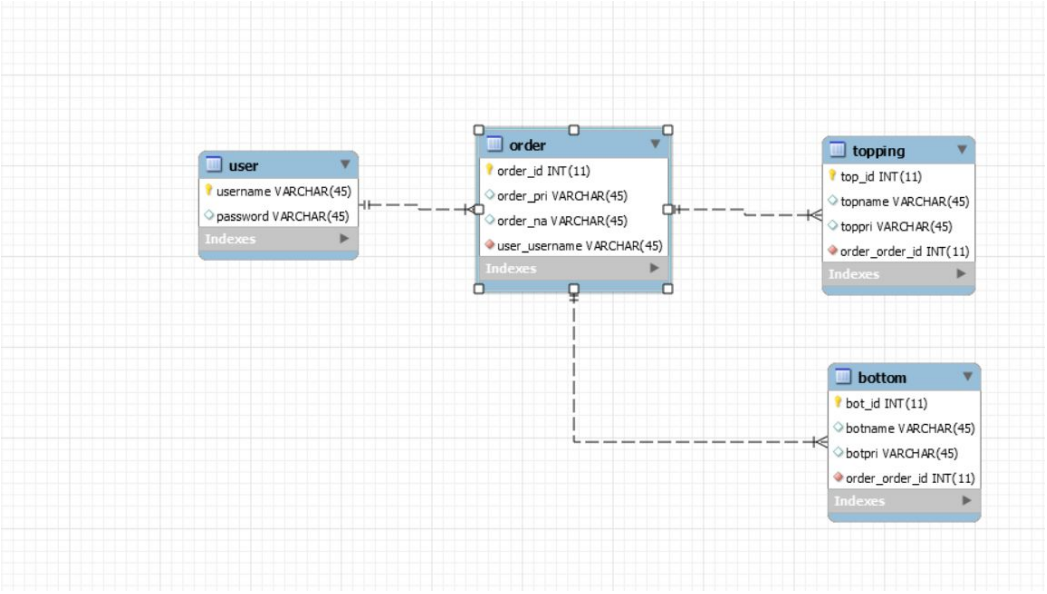
Dette er en oversigt over de teknologiske materialer vi har anvendt i dette projekt.

Teknologi	**Version**
MySQL workbench	**6,3 CE**
NetBeans	**8.2**
PuTTYgen	**0.70**
FileZilla	**3.31.0**
Git Gui	**2.16.2**
Git Bash	**2.16.2**
Notepad++	**v7.5.5**
Droplets	**DigitalOcean**
Github Desktop	**1.1.0**

Domæne model & ER diagram



1.1 Udkast til en tidligere model af ER-diagram.



1.2 Nuværende version af ER-diagram.

Udkastet for dette ER-diagram hænger således sammen; at en *user* kan lave en order på en cupcake. En cupcake består så af to dele; *topping* og *bottom* hvori disse har en såkaldt “mange til én”-relation.

Tabellen *order* er “én til mange” fordi man skal kunne vælge flere toppings og bottoms med forskellige kombinationer. Dette giver brugeren mulighed for at lave flere forskellige ordrer.

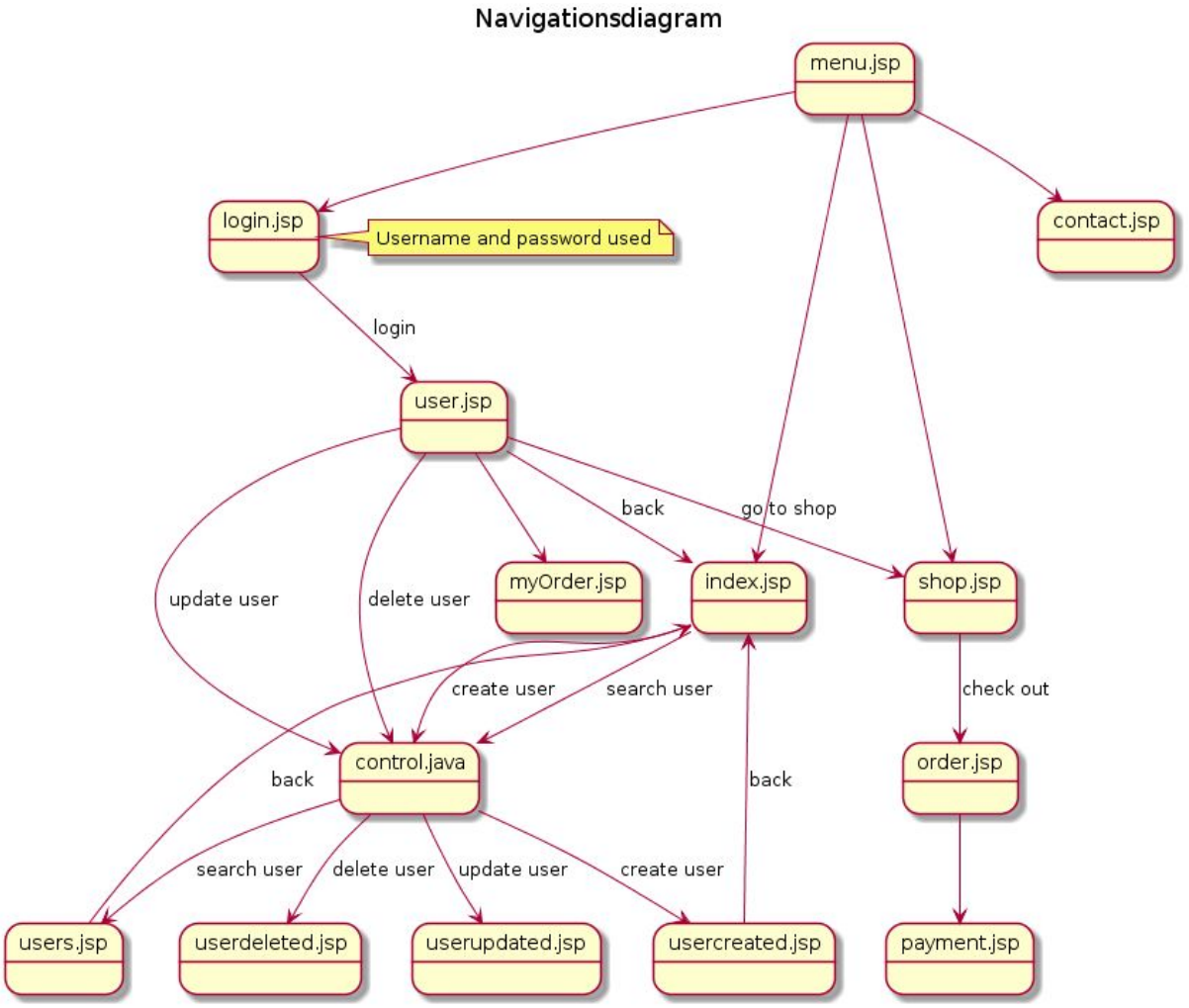
Topping og bottom består hver af deres egne *foreign keys* ”*order_order_id*” som relateres til *order,* der har sin primary key; “*order_id”*. Herefter forbindes *user* med *order* via følgende foreign key; “*user_username*”.

De triggers der bliver brugt er: *insert*, *select* og *delete*.

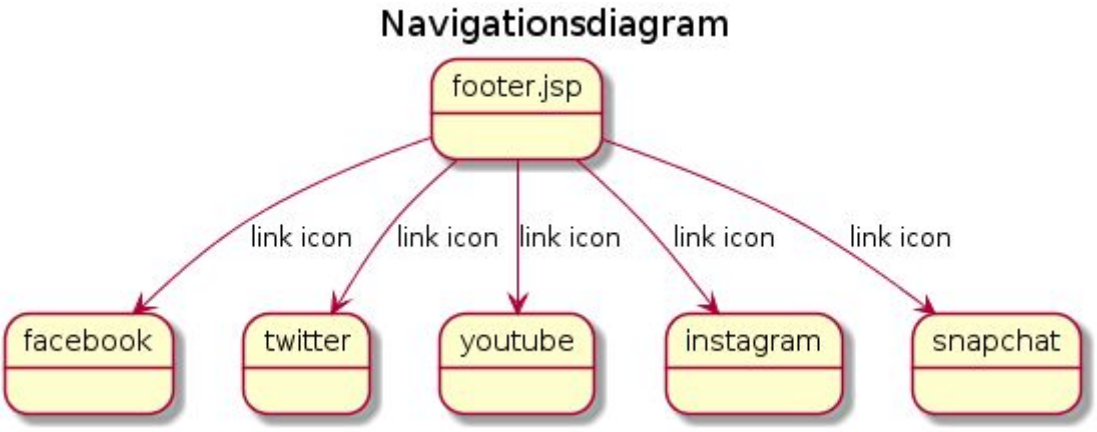
Insert er når der indsættes data i databasen. Select bruges til sql statements når der skal eksekveres en session eller test. Delete skulle kunne fjerne data (username og password) fra databasen.

Vi bruger Anden Normalform ”2NF” da alle tabeller har en primærnøgle, men vi har også *kandidatnøgle* siden vi har “én til mange” relation i henholdsvis order, topping og bottom.

Navigationsdiagram



2.1 Oversigt over det fulde Navigations Diagram for projektet

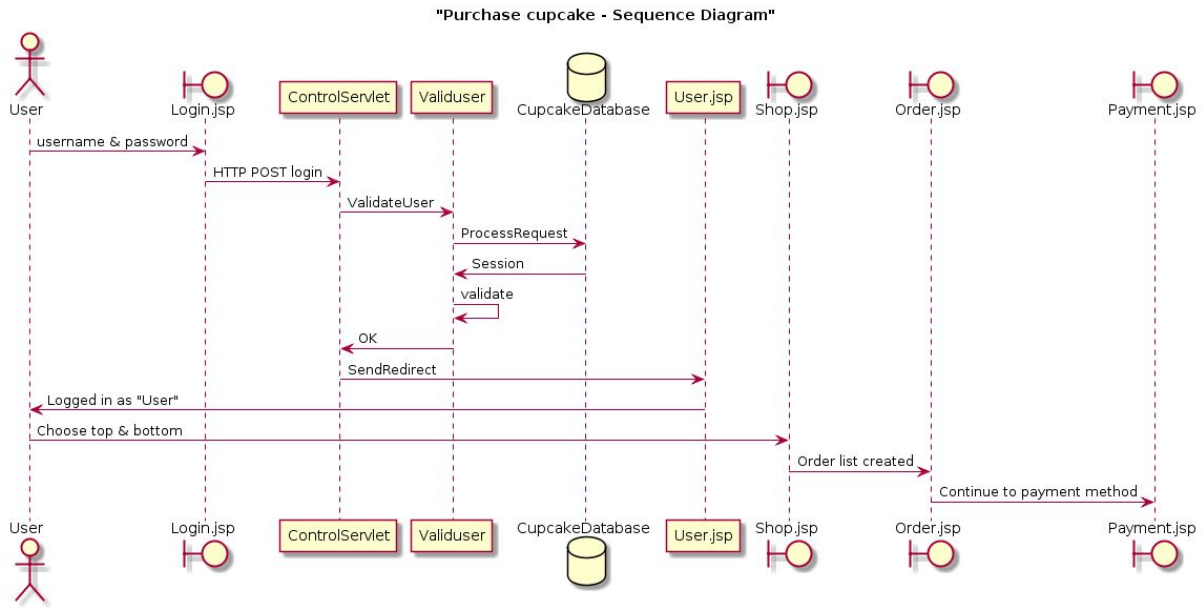


2.2 Navigations Diagram over footer.jsp som er separeret fra det overordnede Navigations Diagram ovenfor.

Menu.jsp er en fælles navigations bar i toppen af hjemmesiden, så det er brugeren bedre kan komme rundt på hjemmesiden. Derudover er der en fælles footer med links til de forskellige sociale medier.

Control.java er vores servlet der håndterer alt der har med user-input af gøre, som create, search, update og delete. De sider hvorpå en bruger kommer gennem en servlet er på index.jsp og user.jsp når der bliver trykket på en knap, før de bliver omstillet til næste jsp side.

Sekvensdiagram



3.1 Sequence Diagram som viser hvordan man som bruger logger ind og gennemfører et køb på cupcake webshoppen.

En bruger angiver inputs (username & password) på login.jsp siden. Her valideres brugerens informationer i ValidUser.java klassen som anvender Control servletten til at lave en request via SQL databasen. En såkaldt session holder styr på de to parametrer gemt i databasen og Control servletten får så de parametrer tilbage fra to tabeller og “godkender” brugerens input. Der er altså kun én servlet som vha. metoden processRequest anvender en switch til at vide hvor den skal sende én videre til.

En if-sætning i User.jsp informerer om brugeren logges ind med success eller om login fejles. Hvis login fejles angives det stadig at brugeren har adgang, men som anonym bruger.

Brugeren videresendes nu til sin personlige User.jsp side hvor der er adgang til webshoppen.

Særlige forhold

En bruger har et username og password som skal angives for at kunne logge ind på brugerens personlige side. Det vil sige, at denne session kun bruger to typer information som gemmes i databasen i hver deres tabel. For at sikre at der ikke er flere som har samme username, er denne parameter blevet gjort unikt. Hvilket betyder, at der kun er én som kan have det specifikke username der bliver brugt.

Til at validere en User der prøver at logge ind bruges en PreparedStatement som sætter username og password som String værdier. Ved at gøre dette, kan der skrives en sql statement (executeQuery) som kalder alle usernames og passwords fra sql databasens tabeller, som er identiske med det som brugeren prøver at logge ind med. Hvis der ikke kan findes en korrekt kombination af username og password som er identisk med det fra databasen, meldes en fejl med meddelelsen “login failed” og brugeren bliver konverteret til anonym bruger. Det vil sige at der ikke er nogen sikkerhed i at nægte brugere adgang til siden, selvom de ikke logger på via en bruger som er gemt i databasen.

Status på implementation

Databasen:

I starten valgte vi at lave flere tabeller til inddelingen af users med en kombination af username, password, UserIDs, balance (valuta) og en email. Parametrer som skulle gøre det nemmere at differenciere imellem diverse brugere i databasen. Vi fandt dog hurtigt ud af at dette gjorde tests sværere at udføre, samt kreationen af en ny bruger krævede alle disse oplysninger før det blev en success. Vi valgte så at droppe denne fremgangsmåde, og fokusere på det funktionelle og simplistiske, frem for det kreative og komplicerede.

En brugers username er unikt og der behøves ikke tildeles et UserID for at identificere diverse brugere. Det betyder, at der ikke kan være to brugere med samme username.

En bruger som ikke bliver valideret under login konverteres til anonym bruger. Dette skal laves om så brugeren bliver nægtet adgang og skal prøve et nyt login forsøg. Det ideelle ville være at alle havde adgang til webshoppen, men så til sidst skulle verificeres som bruger for at kunne gennemføre et køb.

Deleteuser.jsp og updateuser.jsp får p.t. ikke svar fra servletten og kan derfor ikke eksekveres som forventet, men switch-“delete” virker når testen bliver kørt fra DataMapperTest.java.

Derudover mangler der sessions mellem siderne så man forbliver logget ind og det er ikke muligt at få gennemført en update eller delete user via siden. Dette kan kun udføres via tests i DataMapperTest.java.

Navigations Diagram:

Navigations Diagrammet fra tidligere er den visuelle idé til, hvordan hjemmesiden skulle struktureres. Herunder er status over hvilke sider der blev oprettet.

JSP sider:

Krav for siden	**Status**	**Kommentar**
Index	√	Forsiden
MyOrder	-	En side hvor brugeren kan se sin personlige ordre
Contact	-	En kontaktinformations side til brugeren så der er let adgang til oplysninger om firmaet
Order	-	En ordreside som viser den nuværende ordre som den er sammensat
Payment	-	En side hvor brugeren vælger betalingsform
Login	√	En side hvor brugeren kan logge ind.
Shop	√	En side hvor brugeren kan vælge topping og bottom af cupcakes.
Menu	√	En header med navigations bar
footer	√	En footer med links til de sociale medier

user	√	En side hvor brugeren kan se sin konto og ændre den.
users	√	En side hvor man kan se users på siden
Usercreated	√	En side der giver feedback på at en user er lavet.
Userdeleted	(√)	En side der giver feedback på at en user er slettet.
Userupdated	(√)	En side der giver feedback på at en user er blevet opdateret.

Webshoppen:

Selve shoppen er visuelt hardkodet med CSS og bruger derfor ikke data (topping & bottom) fra databasen. Det vil sige at lige så snart man skal vælge topping og bottom får brugeren ikke mulighed for, at lave et køb og checkout. Grundet tidspres blev vi nødt til at benytte denne midlertidige løsning for, at udarbejde et layout og komme på noget funktionelt til hele brugerfladen.

Fejl:

Når der laves en bruger bliver brugeren gemt i databasen på korrekt måde, og giver outputtet "*user created*". Hvis man så laver en bruger med tilsvarende username (uanset hvilket password der angives) får man det samme outputtet, men brugeren bliver ikke tilføjet til databasen. Den bliver bare kasseret / ignoreret.

Test

I dette projekt er DataMapper.java klassen blevet testet. Da det er den klasse som har med users at gøre følte vi at det var mest relevant at fokusere på disse tests.

Inde i DataMapper.java klassen er der metoder for getUsers, getUserByName, createUser, deleteUser, updateUser og validateUser.

- *getUser* metoden henter users fra en ArrayList og gengiver alle users der findes.
- *getUserByName* metoden returnerer en user, hvis det username man søger findes i sql databasen.
- *createUser* metoden laver en ny user med det username og password man angiver, dog fejler det hvis username (unikt) allerede findes.
- *deleteUser* metoden fjerner en user fra databasen, ved at angive username da det er unikt.
- *validateUser* metoden tjekker om username og password til brugerens konto stemmer overens med det som står i databasen.

Hvilke klasser er blevet testet?

De metoder der er brugt ligger allesammen i DataMapper.java som er implementeret fra interfacet DataMapperI.java. Derfor er det kun denne klasse som bruges til tests.

Hvilke metoder er testet?

Metoderne fra DataMapper.java findes herunder:

GetUser: Finder en bruger ud fra et givent username, som er unik. Derfor, kan der ikke findes to brugere med samme username.

CreateUser: Laver en bruger med parametrene: username og password.

Redegørelse for valget af de anvendte tests

I de angivne tests er der blevet fokuseret på at bruge specifikke inputs som har til formål at tjekke om data bliver håndteret og eksekveret korrekt. Specifikke parametrer såsom *username* og *password* har været mest hyppigt brugt. Jo flere parametrer der skal håndteres, desto sværerer bliver testene at udføre. Derfor har vi valgt denne enkle metode.

Kildefortegnelse

iBøger:

Damhus, Martin & al. 2013:"Informationsteknologi"

https://it.systime.dk/

Systime A/S forlag

Hjemmesider:

Stackoverflow.com

Dat 2Sem 2018 Forår: "Rapportskrivning og dokumentation"

datsoftlyngby.github.io/dat2sem2018Spring, marts

https://datsoftlyngby.github.io/dat2sem2018Spring/Modul2/Week4-Report/

(sidst opdateret 15-03-2018 09:48)

PlantUML in a nutshell

plantuml.com

http://plantuml.com/

(lokaliseret 12-03-2018)

PlantText:"-The expert's design tool"

planttext.com

https://www.planttext.com/

(lokaliseret 12-03-2018)