CupCake Shop

af Adam Lass, Nikolai Perlt Andersen & Stefan Hjalholt Gregersen

Marts 2018

B-klassen

Kontaktoplysninger

Adam Lass - @adamlass - cph-al262@cphbuiness.dk

Nikolai Perlt Andersen - @Perlten - cph-na130@cphbuiness.dk

Stefan Hjalholt Gregersen - @Tallark - cph-sg119@cphbusiness.dk

Kodens kilde: https://github.com/adamlass/CupCakeShop

Indledning

Dette projekt er en online cupcake shop der vil kunne underbygge det digitale fundamentet for en lille forretning der sælger cupcakes. Som bruger på siden er det muligt at designe sin egen cupcake, tilføje penge til sin konto, ligge ordre og se en oversigt over dem senere hen. Derudover findes der en administrator-side hvor ejerne af shoppen kan administrere brugere, ordre og tilføje nye cupcake-dele. Webshoppen er en full-stack der bruger jsp sider på en Tomcat server i frontend, java servlets i det logiske lag og en MySQL server som database. For at undgå forskellige injections på vores side har vi valgt at tage brug af prepared statements som et lag mellem bruger inputtet og databasen. Hele projektet ligger på en ubuntu server der bliver hostet af DigitalOcean.

Baggrund

Kunden er en mindre cupcake forretning der ønsker at udvide deres forretning med en webshop.
De ønsker en hjemmeside hvor deres kunder kan oprette en profil, og bestille cupcakes med selvvalgt top og bund.

Teknologi valg

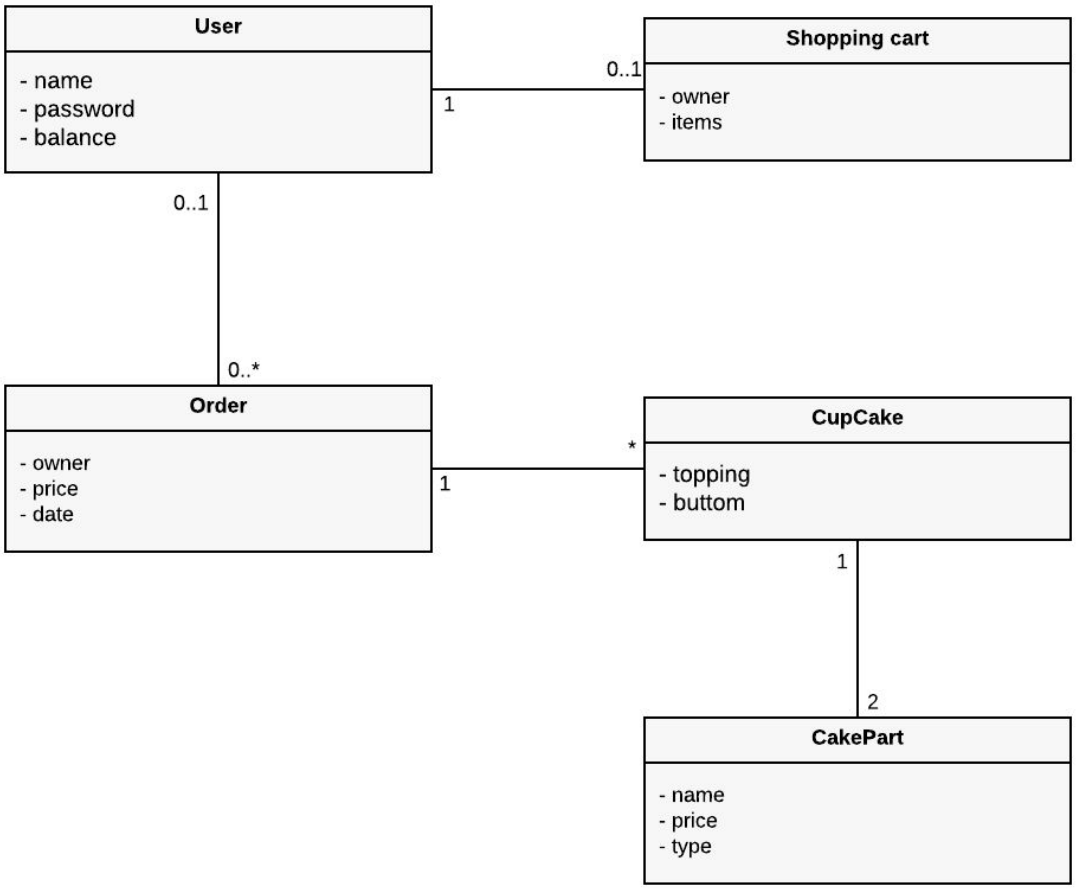
NetBeans 8.2
mysql-connector-java 5.1.44
JDK 1.8
MySQL
jsp-sider
java ee web api
Git
TomCat8 server

Krav

Hjemmesiden skal have en global navigation bar / menu der er tilgængelig fra alle sider.
En kunde skal være oprettet, og logget ind for at kunne se og bestille cupcakes.

Det skal være muligt for en bruger at tilføje penge til sin konto, samt se sin indkøbskurv og ordre.
En administrator skal kunne tilføje nye cupcake dele til databasen, og have mulighed for at gøre andre konti til administratorer eller slette dem. Endvidere skal det være muligt for en administrator at se all ordre, samt at opdatere prisen på, eller slette ordren.
Hjemmesiden skal kunne ligge på en ubuntu server, og være tilgængelig fra samtlige browsere.

Domæne model og ER diagram

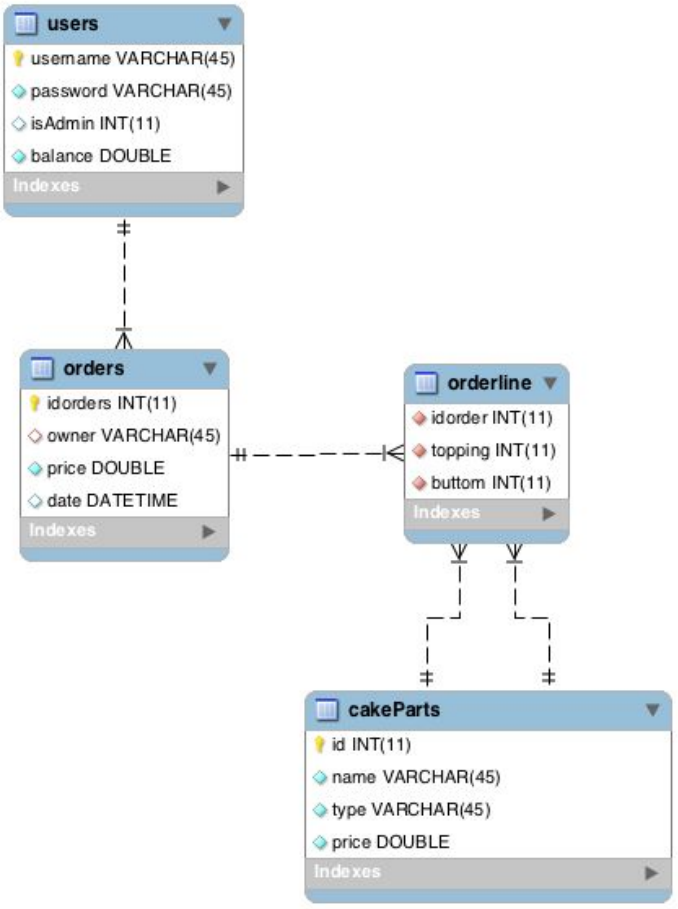


Domæne model

Domænet her viser hvordan forholdene mellem de forskellige entiteter og hvordan de arbejder med hinanden. Det er muligt for brugeren at have op til en indkøbskurv og en indkøbskurv har kun en bruger tilknyttet som værende ejer af kurven. En bruger kan til gengæld have flere ordre, og en ordre kan enten have én eller ingen ejere. Dette skyldes at vi ikke ønsker at slette ordre dataen hvis brugeren lige pludselig ønsker at slette sin konto. En ordres indhold består af cupcakes, men dens

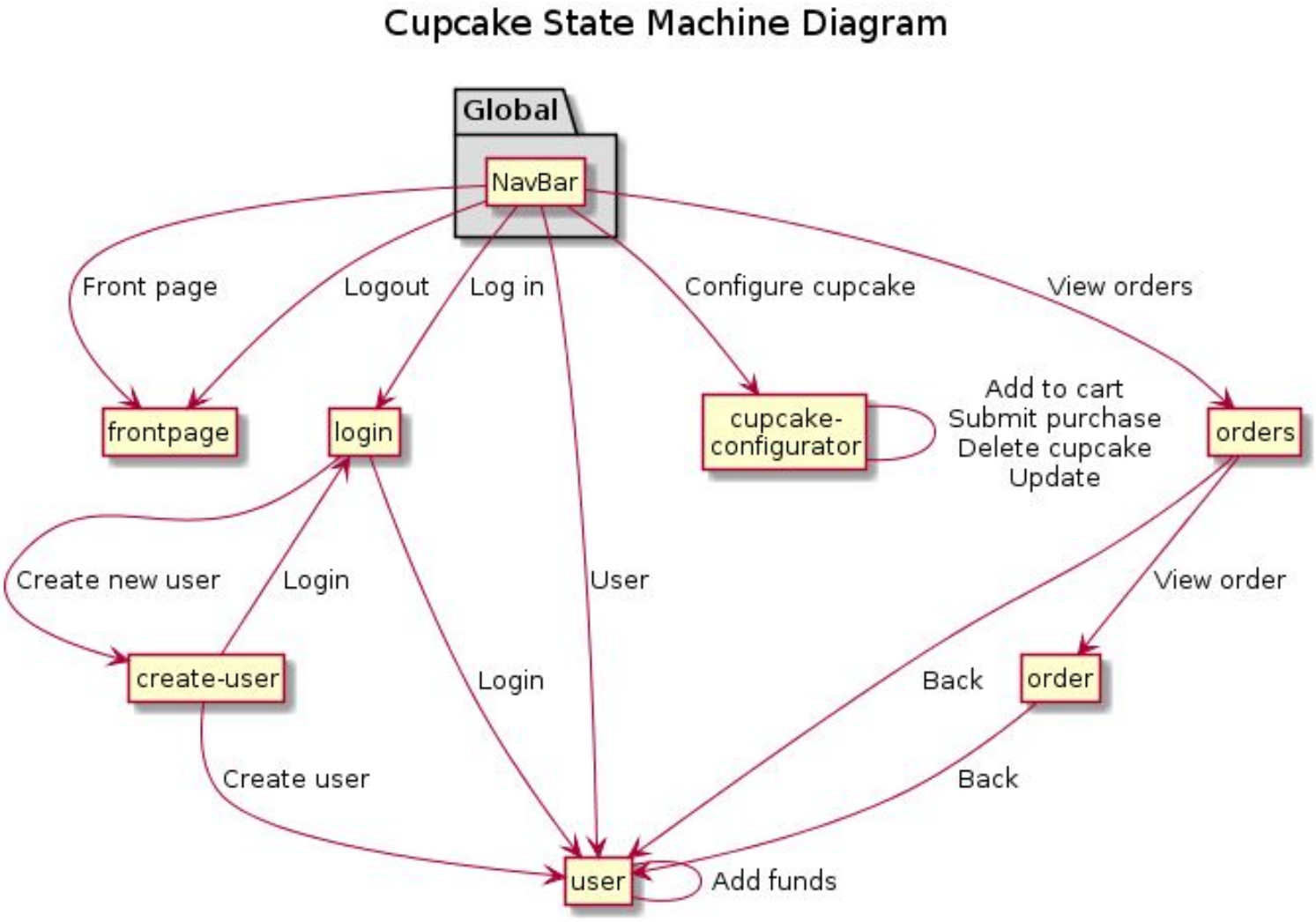
pris er fastsat under bestillingen og afhænger ikke af selve cupcakens pris. En cupcake består af 2 cakeparts som enten er af typen *topping* eller *buttom*.

ER diagram



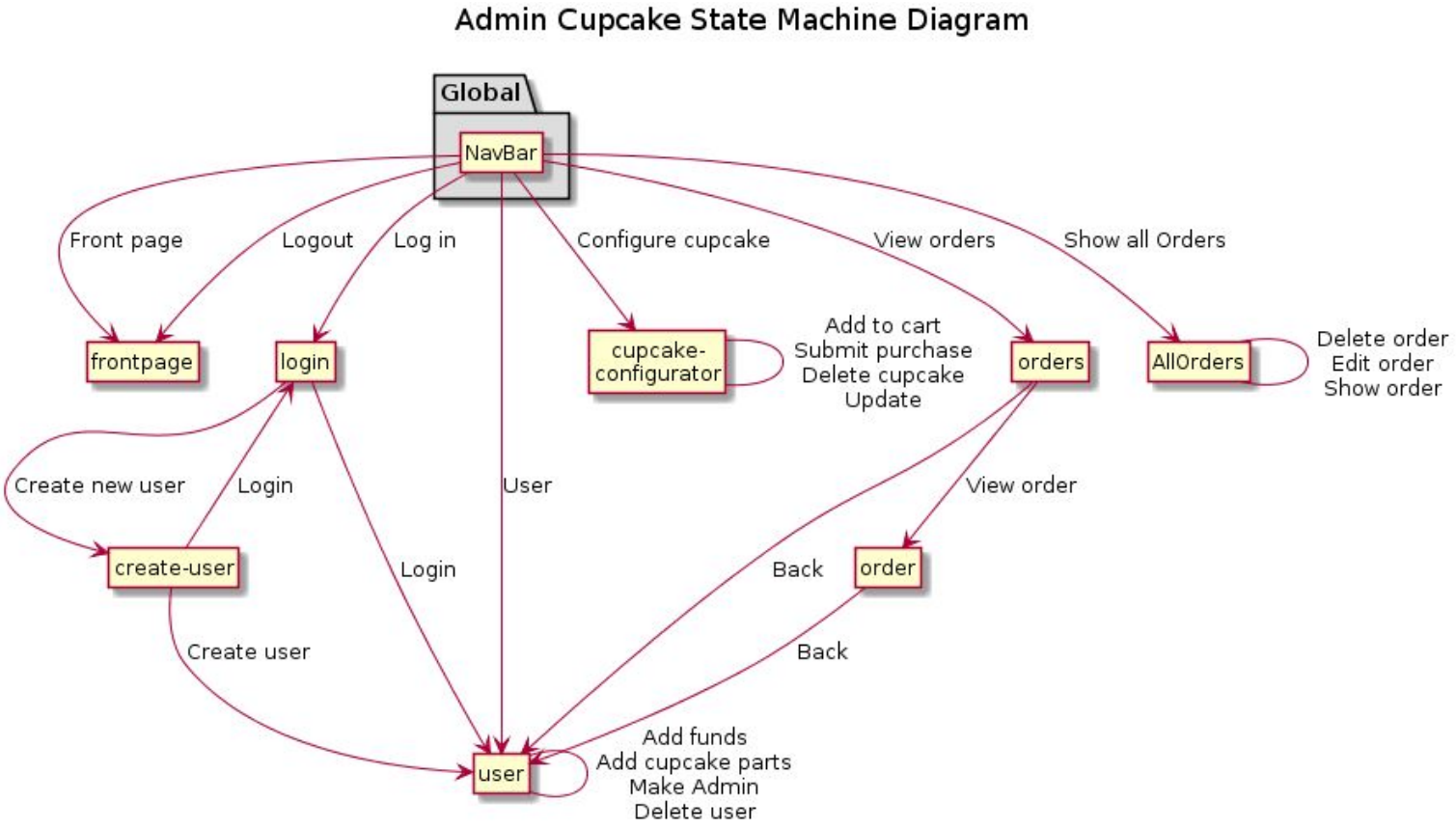
En bruger har sit brugernavn som primær nøgle, og orders har her en fremmednøgle hertil. Bliver en bruger slettet sættes ejerens ordre sat til null. Sletter man derimod en ordre, slettes dens tilhørende ordrelinjer da der også her ligger en fremmednøgle til ordre id'et. Tabellen med ordrelinjer fungerer som en mellem-tabel mellem ordren og de to cakeparts der udgør selve cupcaken i domænet.

Navigationsdiagram



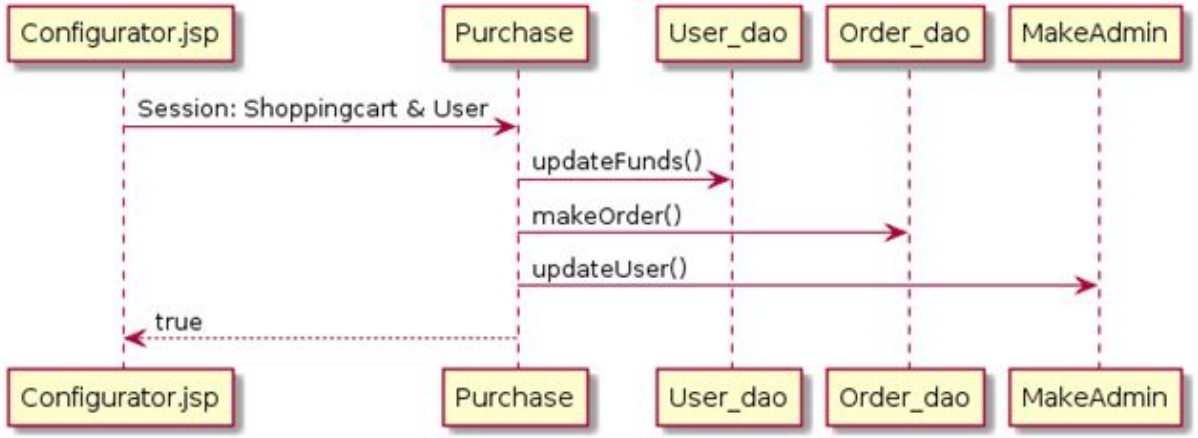
Hjemmesiden udnytter en navigationsbar der giver en kunde mulighed for at logge ind.
Når en kunde er logget ind vil de have mulighed for at komme rundt på siden ved hjælp af navigationsbaren.
Fra navigationsbaren er det muligt at navigere ind på følgende sider: User profile, Cupcakes, order og logout. Bruger du en administrativ konto vil det også være muligt at komme ind på: All Orders.

Når en bruger logger ind eller opretter en konto vil brugeren blive taget direkte videre til brugerens profil side.



Administratorer har også mulighed for at slette ordre og bruger, samt redigerer prisen på ordre lavet af en kunde, og se alle ordre.

Sekvens diagram



Når brugeren har valgt sine cupcakes bliver han sendt ind på en servlet som tester for at se om han har penge nok til at gennemføre købet. Hvis dette ikke er tilfældet bliver du sendt tilbage til Configurator.jsp med beskeden ”Not enough funds!”. Hvis brugeren derimod har nok penge til at købe sin cupcake, bliver både databasen og bruger objektet opdateret. Hvis alle kriterierne bliver opfyldt bliver true returneret.

Særlige forhold

Når der bliver skabt en ny bruger bliver brugeren gemt i session, og gemt der indtil brugeren enten logger ud eller der logger en ny bruger ind. Derudover når brugeren opretter sin første cupcake bliver der oprettet en liste som bliver lagt op i session som shopping cart. Når brugeren så enten vælger at købe sine cupcakes eller logger ud, bliver shopping cart igen sat til null i session.

Måden exceptions bliver behandlet ved brugerinput, er ved at få fat i request.getParameter som returnere det beløb brugeren gerne vil have indsat. Denne værdi bliver så castet til en double, hvis brugeren har indtastet andet end et tal vil der blive kastet en numberFormatException som bliver fanget og servleten sender brugeren tilbage til user.jsp med false som parameter. Taster brugeren en tal ind, som bliver korrekt castet men som er mindre end 1, bliver der kastet en IllegalArgumentException. Den bliver fanget og brugeren bliver sendt tilbage til user.jsp med invalid som parameter.

Når brugeren taster noget input som har noget at gøre med vores database(r) så kommer det altid igennem en prepared statement. Derudover som nævnt i kolonnen ovenover så bliver enhver anden form for brugerinput test for at se om det er gyldigt input, så vi undgår bogstaver hvor vi forventer tal.

Når en bruger gerne vil logge ind taster han et brugernavn og password som kommer igennem en prepared statement og derefter køre vi igennem vores database af bruger og ser om vi kan finde en bruger der matcher de to input. Hvis dette er tilfældet sender vi brugeren over til user.jsp og gemmer hans bruger i session. Hvis der ikke er noget match sender vi ham tilbage til login.jsp med false som parameter. En af måderne vi sikre især hans password er ved at bruge post i stedet for get. Dette sørger for at følsomt input som f.eks. vores password ikke kommer med i url.
Hvis en bruger konto balance kommer under 0 vil denne blive skrevet med rødt for at gøre ekstra opmærksom på problemet. Så snart brugeren er tilbage på et positivt tal vil farven skifte tilbage til grøn.

Hvis en bruger bestiller og køber en cupcake, vil den samlet pris selvfølig blive trukket fra hans konto. Hvis en administrator skulle vælge at ændre eller slette prisen vil differencen blive henholdsvis trukket eller tilbagebetalt til brugeren. Dette gør det muligt at efterlade brugeren med en negativ balance, hvis der ikke er nok dækning på kontoen, når prisen bliver forhøjet.

Der findes to typer brugere på serveren: standard brugeren og administrator brugeren. Forskellen mellem de to bliver defineret i databasen ved brug af en isAdmin boolean, og dens tilstand kan ændres af andre admins via ‘user’-siden. En bruger linje på databasen består af et brugernavn, en adgangskode, en isAdmin switch og en balance i datatypen double.

Da en brugers balance, rettigheder og andre artefakter kan ændre sig undervejs i en brugers session på siden, sørger vi altid for at opdatere sessionens bruger objektet

når det kan have en indflydelse på den enkelte sides tilstand. Dette gøres via en statisk updateUser metode der fungerer ved at hente samme bruger op fra databasen og gemme den i sessionen på ny.

Når en ordre gemmes bliver antallet af de enkelte cupcakes behandlet som individuelle ordrelinjer. Dvs. at der kan komme mange rækker på en tabel når man læser ordren. Alternativet var at gemme antallet som et felt hvilket ikke er blevet implementeret.

Status på implementation

Når man først kommer ind på siden bliver man præsenteret af en tom frontside, denne bør inkluderer mulighed for at en bruger kan logge på eller oprette sig direkte.

Er en bruger allerede logget ind, og indtaster URL'en til hjemmesidens logind side, bliver brugeren ikke redirected til en anden side, men vil tillade at man logger ind igen.

Derudover er personfølsomme information, som f.eks. brugerens password, ikke blevet gemt krypteret, men direkte. Dette kan give nogle problemer hvis nogen får adgang til databasen.

Den brugte datatype i databasen for at vise om de er normale brugere eller administratorer er integer og ikke tiny integer.

Når der bliver lavet et request på ordre lister bliver der hentet mere data fra databasen end der er behov for, dette resulterer i en højere runtime end nødvendigt.

Hvis en bestilling fejler af en anden årsag end manglende kapital på kontoen, vil brugeren ikke modtage en fejlbesked, men kan se oppe i URL'en at den fejlede.

Ekstra indhold

Class diagram