Cupcake Rapport

Forfattere:

Malik Sharfo, cph-ms706@cphbusiness.dk, Github: MalikSh96, B

Jacob Petersen, cph-jp266@cphbusiness.dk,Github: UnknownQuantity , B

Dennis Holm Hansen, cph-dh143@cphbusiness.dk, Github:Joklin92, B

Projekt start: 19-02/2018

Dato for skrivning: 15-03/2018

Contents

Indledning

Formålet med dette projekt har været, at lave en cupcake shop hvor det har skullet være muligt, at sammensætte en cupcake bestående af en topping og en bund, for derved at få en samlet cupcake.

Et af hovedformålene med projektet, har været at teste vores færdigheder, da vi har skullet kombinere vores viden om backend og frontend sammen, så vi i sidste ende har en samlet cupcake shop hvor det er muligt at bestille cupcakes.

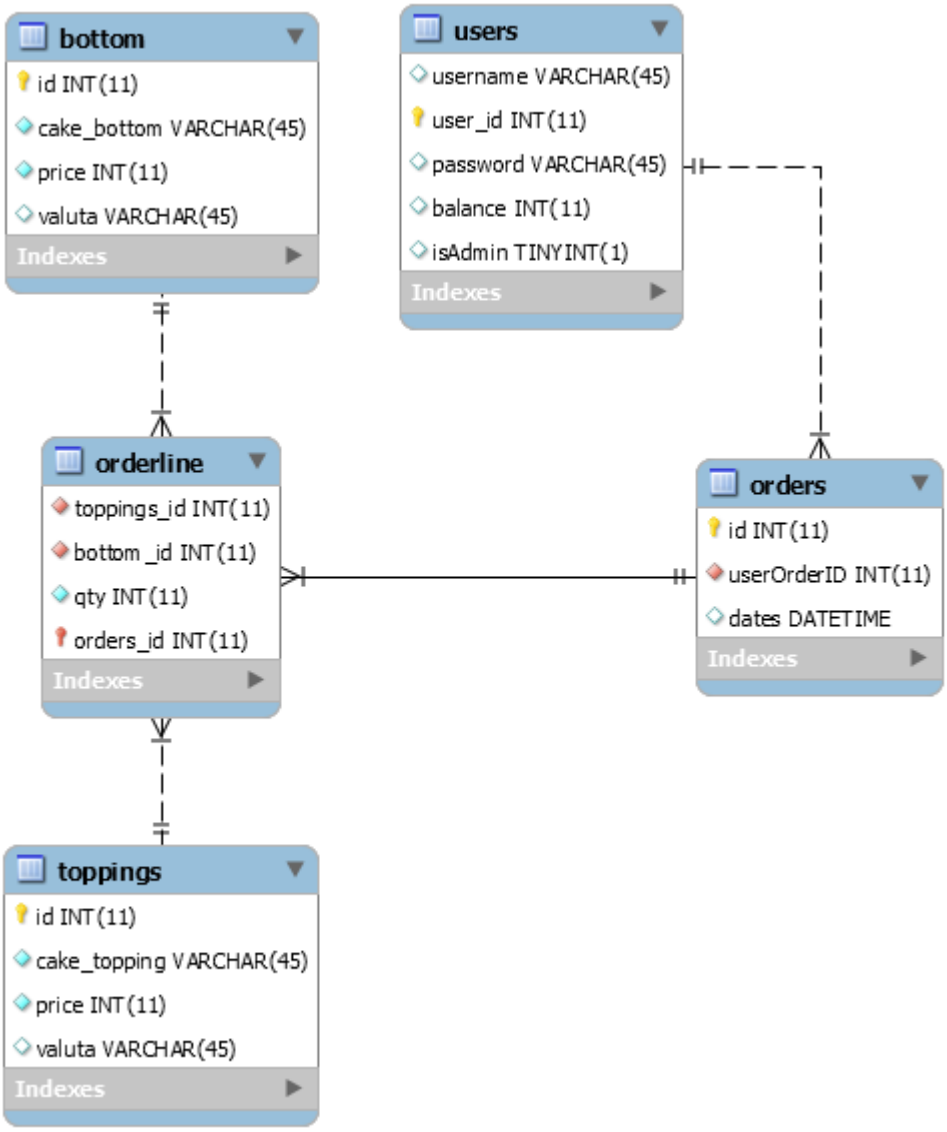
Baggrund

Hjemmesiden skulle laves på baggrund af en cupcake shop som ønskede at sælge cupcakes via internettet. Cupcake siden skulle give besøgende mulighed for at oprette sig som bruger og tilføje penge til sin konto. Det skulle også være muligt for brugeren at kunne bestille cupcakes og hjemmesiden skulle, ud fra brugerens input, hente cupcakes i en database. Brugeren skulle til sidst kunne bekræfte dennes ordre.

Teknologivalg

Projektet er lavet i programmeringssproget Java (Java SE 9.0.4) og er skrevet i programmet NetBeans (NetBeans IDE 8.2 (Build 201609300101)). Samtidig er GitHub.com brugt flittigt for nemt at dele projektet i mellem udviklerne, enten via NetBeans integrerede løsning, eller via Git-Bash.
Til projektets database er der brugt MySQL Workbench (MySQL Workbench Community (GPL) for Windows version 6.3.9 CE build 10690321 (64 bit))
For at få projektet op på cloud'en er der blevet oprettet en "droplet" på DigitalOcean.com som er en amerikansk udbyder af cloudservices. En Droplet er i vores tilfælde en Linux (Ubuntu 17.10) server.
For at kommunikere med vores "droplet" er programmet Git-Bash (Git-Bash 2.16.2 for 64-bit Windows) blevet brugt. Programmet tillader at man kan skrive linux kommandoer til serveren når man har forbundet sig til den.
Af sikkerhedsmæssige årsager er der blevet oprettet SSH nøgler vha. Putty (Putty 0.70 for 64-bit Windows) så udefrakommende med eventuelt onde hensigter ikke kan få forbindelse til dropletten.
For at få lagt SSH nøglen op på dropletten er FileZilla (FileZilla 3.31.0) brugt, som er et FTP (File-Transfer-Protocol) redskab som nemt tillader at flytte filer i mellem to systemer.

Domæne model + EER diagram



I forhold til hvordan databasen skulle bygges op, kiggede vi på hvad vores database burde indeholde.

Vi har oprettet fem tabeller, hvor hver tabel skulle dække det nødvendige indhold.

Toppings og bottom tabellen er ens, og begge tabeller har fået tildelt en id. Denne id har vi fået databasen til at genere for os. Herefter benyttede vi auto_increment primary key, så vi ikke selv skulle styre hvilken id vores toppings og bottoms fik.

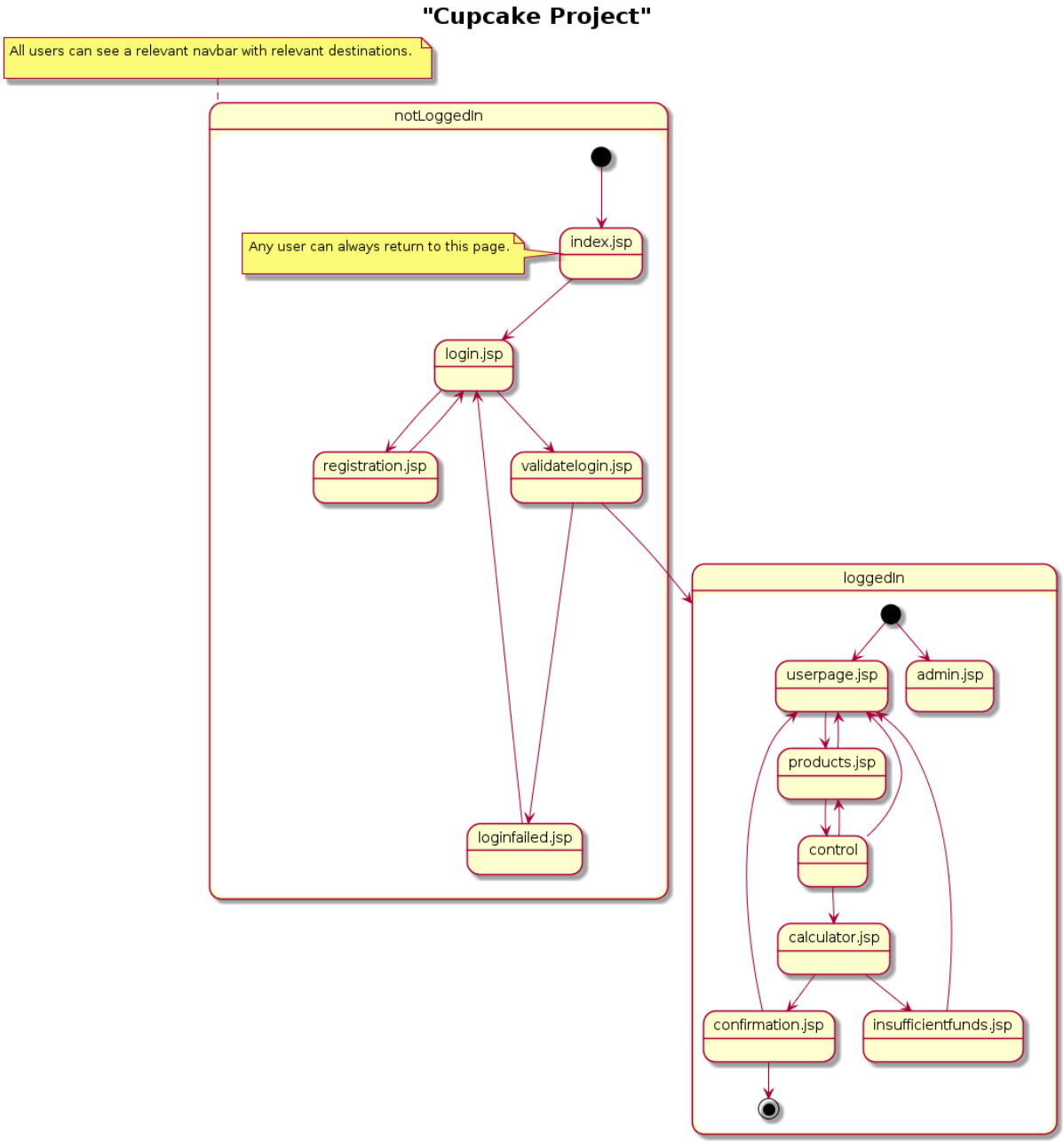
Udover det har de fået tildelt et navn samt en pris som derefter er blevet indsat i databasen ved hjælp af "insert".

Vi har lavet en tabel som skal holde vores brugere, kunder såvel som admin, en user får tildelt et brugernavn, et password, en balance, og der tjekkes om vedkommende bruger er en admin eller ej.
Det vigtigste i bruger tabellen er user_id, denne gør også brug af auto_increment primary key, således at en bruger også bliver tildelt en computer genereret id.
Slutteligt er de sidste to tabeller en orderline og orders tabel.
Order tabellen har en id samt en userOrderID, ideen her var, at når en ordre blev sat, så ville ordren få tildelt en id, igen genereret ved hjælp af auto_increment primary key.
Orderline tabellen har til opgave, at indeholde id'en på den topping og bottom som er valgt til at danne cupcaken som kunden ønsker.
Denne indeholder quantity, således så man kan se antallet af de cupcakes som er valgt, her tildeles ordren også en id.

Brugen af foreign key constraints:

Tabellen er forbundet på den måde, at en user kender til en order, en 1 til 1 forbindelse, denne order kan flere orderlines, en 1 til flere forbindelse.
En user kan ikke kende til orderline, da brugeren kun kan sammensætte en order.
Til orderline tabellen er der en forbindelse fra orders, toppings og bottom, hvor hver har til opgave, at matche hinanden på baggrund af id.
Det vil sige, at når en topping og bottom bliver valgt, så skal id'en som genereres fra toppings og bottom tabel, matche den id der tildeles i orderline tabellen.
Orders_id i orderline tabellen, skal kende til id'en i orders tabellen.
Toppings_id i orderline tabellen, skal kende til id'en i toppings tabellen.
Bottom_id i orderline tabellen, skal kende til id'en i bottom tabellen.
Den primære nøgle i orderline tabellen er orders_id, da det i sidste ende er den id, der skal henvises til når en ordre bliver sammensat.

Navigations diagram



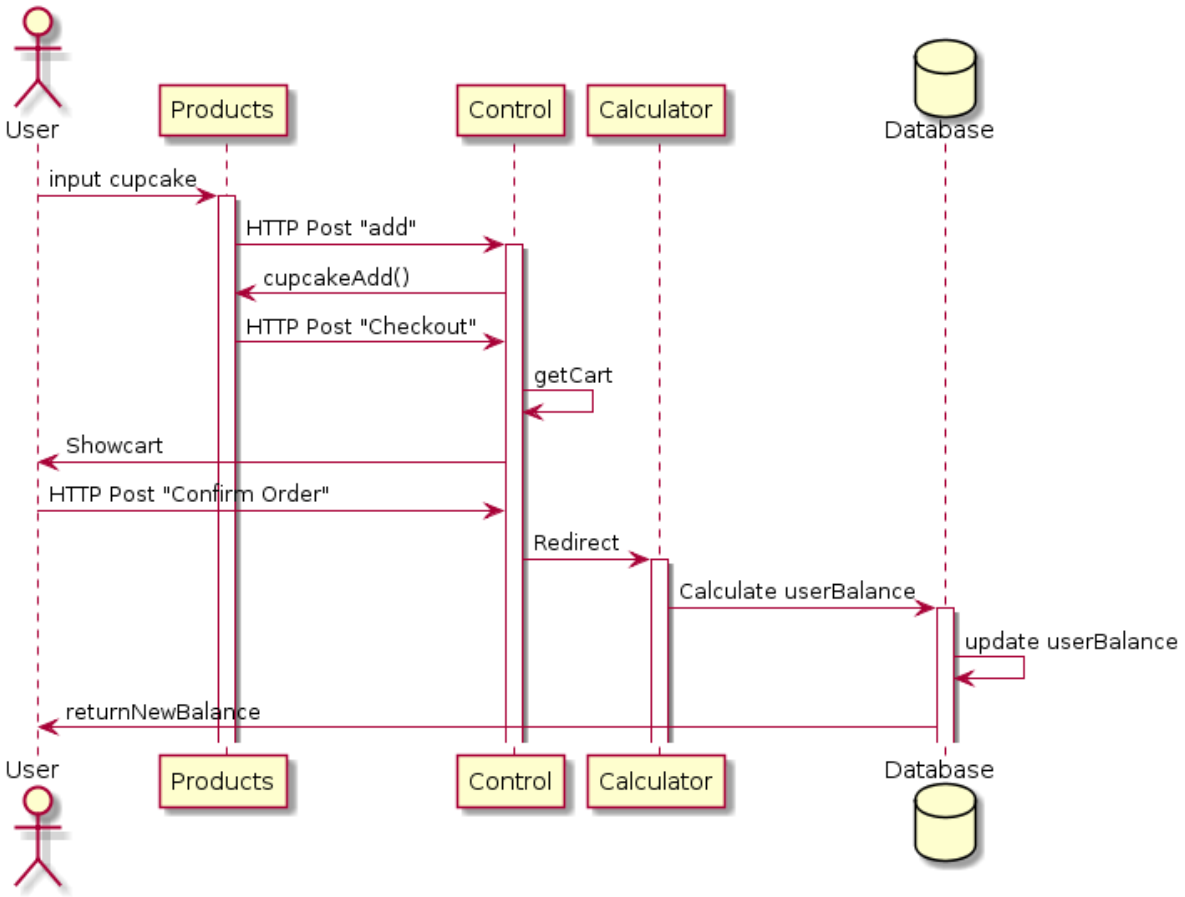
Vores hjemmeside er lavet med en navigations bar i toppen som tillader for nem navigering i mellem de forskellige sider. Alt efter om man er logget ind, eller ej, har man adgang til to forskellige navigationsbarer.

Er man ikke logget ind, har man via navigationsbaren adgang til Homepage, Login og Registration.

Er man logget ind, har man via navigationsbaren adgang til Homepage, User Page og Products.
Er man admin, har man adgang til en speciel admin side, samtidig med at dennes navigations bar understøtter alle ovenstående muligheder, logget ind eller ej.

Siden benytter kun én Servlet, som hedder *Control*.
Control bruges til at hjælpe navigationen i mellem .jsp siderne, og holder også en stor del af deres logik, samtidig med at den også har en række println's som udgør en side som brugeren også kommer igennem.
.jsp sider der får logik fra *Control* er: *login.jsp, registration.jsp, products.jsp* og *admin.jsp.*
For at lægge logikken ind på .jsp siderne benyttes en switch med tilhørende cases samtidig med at der er et hidden inputfield på .jsp siderne som switchen bruger til at lægge den rigtige logik fra *Control* ind på den korrekte .jsp side.

Sekvens diagram



Når en bruger skal bestille en cupcake, gøres dette ved at bruger laver input der fortæller hvilken cupcake brugeren ønsker ud fra det udvalg der findes i databasen. Idet bruger har valgt sin cupcake tilføjes denne til brugers shoppingcart. Dette sker i controlleren som modtager input fra brugeren, laver en cupcake og smider den i shoppingcart sammen med antallet brugeren har oplyst. Vi bruger her to if-statements for at tilføje cupcake til shoppingcart. Først tjekker vi om "Add to cart" knappen er trykket og hvis dette er tilfældet tjekker vi om der er mere en 0 cupcakes tilføjet. hvis dette er tilfældet tilføjes cupcake til shoppingcart.

Efter brugeren har tilføjet en cupcake sender controlleren en redirect til produktsiden hvor brugeren så kan tilføje flere cupcakes til sin shoppingcart, eller vælge checkout.

Når brugeren vælger checkout sendes denne videre til controlleren hvor brugeren så kan se

sin nuværende shoppingcart og bekræfte ordren hvis denne er korrekt.
Controlleren sender derefter jobbet videre til en mellemliggende jsp side "Calculator.jsp" som brugeren aldrig ser. "Calculator.jsp" henter så brugerens nuværende balance fra databasen og checker, ved hjælp af en if-statement, om brugeren har penge nok til den igangværende handel. Er dette tilfældet opdateres brugerens balance i databasen og brugeren sendes til en ordrebekræftelse hvor denne kan se sin nye balance og returnere til sin profile side. Hvis "Calculator.jsp" finder frem til at pengene ikke er tilstrækkelige i forhold til brugerens balance, bliver brugeren videresendt til en side der fortæller at der ikke er penge nok og brugeren kan herfra returnere til sin profil.

Særlige forhold

Når brugeren logger ind, og bliver godkendt i "validatelogin.jsp", bliver der samtidig startet en session for brugeren som den bærer med sig videre så siden kan genkende hvilken bruger der er logged ind på det givne tidspunkt.

I forbindelse med login bruges prikker til at skjule password som brugeren inputter, men yderligere sikkerhed er der ikke i forhold til login.

Brugerinput valideres på forskellig vis. I login fasen bruges en usynlig jsp side: "validatelogin.jsp", som tager informationen brugeren har givet og leder databasen igennem efter en match. Findes der en match bliver brugeren logget ind og ellers bliver brugeren sendt videre til en ny jsp side: "loginfailed.jsp".
Ved bestilling af cupcakes oplyser brugeren hvilken top, bund og antal cupcakes brugeren ønsker. Her bruges en if-statement som validerer at antal inputtet ikke er under 1, hvis dette er tilfældet tilføjes intet til shoppingcart.

I databasen findes to brugertyper: "Admin" og "Ikke Admin". Vi benytter dog ikke bruger typerne i jdbc. vi benytter brugertyperne til at bestemme hvilke sider de skal have adgang til.

Status på implementation

De udleverede krav er opfyldt, dog fandt vi små fejl.
Ved at benytte adressebaren kan enhver bruger tilgå sider som ikke skulle kunne tilgås før senere i programmet, eller overhovedet. En almindelig bruger kan f.eks. tilgå admin siden ved simpelthen bare at skrive "admin.jsp" til sidst i adresselinjen, i stedet for den nuværende aktive jsp. Den almindelig bruger kan derfor få adgang til handlinger kun admins skal have adgang til, i dette tilfælde: at slette brugere i databasen.
Brugere kan ligeledes, igennem adresse baren, tilgå siderne: "loginfailed.jsp", "Insufficient Funds.jsp" samt "confirmation.jsp" uden at fejle et login eller gå igennem en købsprocess.
Når man logger ind som admin bliver man sendt til en speciel admin side, hvor man har adgang til "admin only" handlinger. Denne side er dog ikke mulig at komme tilbage til efter at admin har forladt den.

Kan ikke slette bruger, uden at slette deres ordre. Dette kan dog rettes ved at lave en metode som, ud fra bruger id, sletter samtlige ordrer den givne bruger har lavet og efterfølgende slettes brugeren. Men det burde være bedst at gemme alt data.