Cupcake

Asger Storgaard Høffner cph-ah323@cphbusiness.dk

Lene Annette Skov cph-ls283@cphbusiness.dk

Oliver Scholz Lønning cph-ol31@cphbusiness.dk

2017-10-08

Indholdsfortegnelse

Indledning

Et bageri vil have et system, hvor man kan købe deres cupcakes online til afhentning. Man skal kunne oprette en bruger, logge på en allerede eksisterende bruger, og til sidst, når man er logget på, skal man kunne bestille sine cupcakes.
Vores system er inddelt i to store dele, en frontend hvor vi gør brug af html, css, jsp, javascript, bootstrap, og en backend, som er programmeret i Java og gør brug af en mySQL database. Hele systemet er så lagt op på en linux server vha. tomcat.

Baggrund

Bageriet ønsker en webshop hvor man kan bestille cupcakes, og betale for dem med penge der allerede står på kundens konto.
Brugeren skal altså have mulighed for at logge ind eller oprette en konto, for at kunne bestille og betale for cupcakes. Butikken ønsker ikke mulighed for forsendelse, da det har vist sig at være upraktisk.
Når man bestiller en cupcake skal man vælge en top og en bund, samt antal af den ønskede kombination, dette tilføjes til den samlede ordre på samme side, indtil brugeren afslutter ordren.
Brugeren kan ikke bestille cupcakes for mere end hvad der står på deres konto.

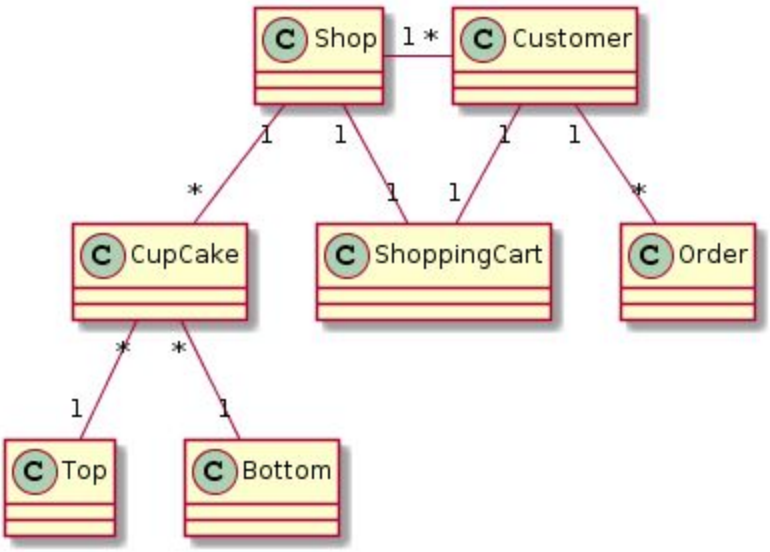
Teknologi valg

Projektet er skrevet i Netbeans 8.2, og databasen er en MySQL database oprettet ved hjælp af MySQL Workbench 6.3 CE.
Backend er skrevet med Java.
Frontend er skrevet med .jsp-sider til at kommunikerer med vores java kode og bruger javascript, css, HTML og Bootstrap. HTML-siderne er css og Bootstrap.

Krav

Firmaets håb for dette system er at man skal få en brugervenlig oplevelse for at kunne bestille en cupcake efter eget ønske online. Visionen er, at brugerne skal kunne se et system som er let overskueligt, men samtidigt yderst funktionelt.
Forhåbentligt kan dette også forbedre firmaet, da det først og fremmest vil øge omsætningen, men samtidig give brugerne en god oplevelse.

Domænemodel

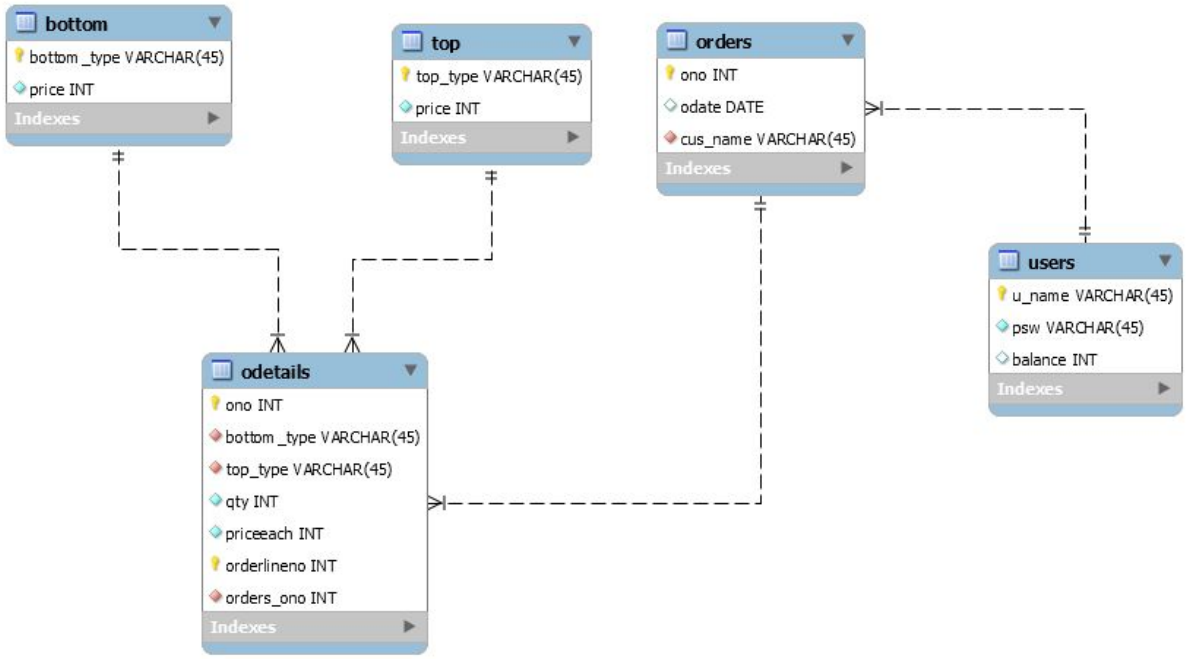


Domænemodellen illustrerer webshoppen sådan som vi forestiller os den ser ud. Butikken har kunder, og cupcakes. Alle cupcakes har en bund og en top. Kunderne kan så lægge cupcakes i shoppingcart, og ved at tjekke ud skabe en ordre. En kunde kan have flere ordre, men en ordre kan kun tilhøre en kunde.

Vi har valgt at lave 1 - * relation mellem shop og kunde, da shoppen kan have mange kunder, og fra vores udgangspunkt er der kun denne ene cupcake-shop.

Vi har også valgt en * - 1 relation mellem cupcake til top og bund, da en cupcake kun kan have en top og en bund, men de forskellige typer toppe og bunde, kan sidde på mange forskellige cupcakes.

ER diagram



Vores ER-diagram viser tabellerne sådan som de er sat op i databasen.

Vores database overholder 1. Normalform, i det alle tabeller har en primær nøgle, og ingen kolonner indeholder mere end en information.

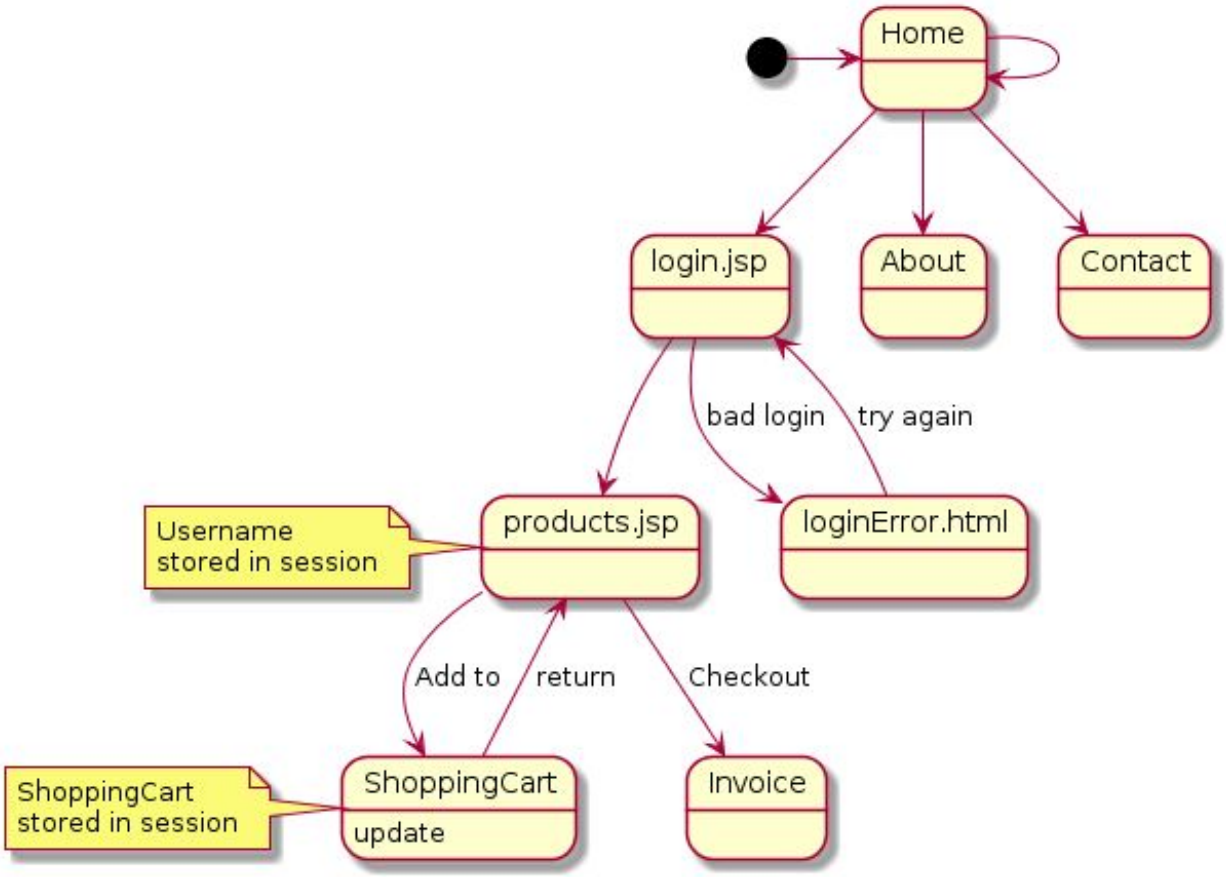
Vi overholder 2. Normalform da vi i vores tabel *odetails* har en sammensat nøgle af ordrenummer og ordrelinje, hvor alle kolonner er afhængig af begge nøgler.

3. Normalform overholdes, da vi ingen steder i databasen har værdier der er afhængig af andet end den angivne primære nøgle.

Diagrammet viser desuden hvordan tabellerne har 1 til mange relationer. Fx kan en user have mange ordre, men en ordre kan kun have en user.

Bemærk at ShoppingCart ikke er en del af databasen, i modsætning til domænet.

Navigationsdiagram



For at få et godt overblik over selve hjemmesiden, har vi lavet et navigationsdiagram, som fortæller os hvordan systemet hænger sammen og hvilke krav der skal opfyldes for at kunne købe en eller flere cupcakes.

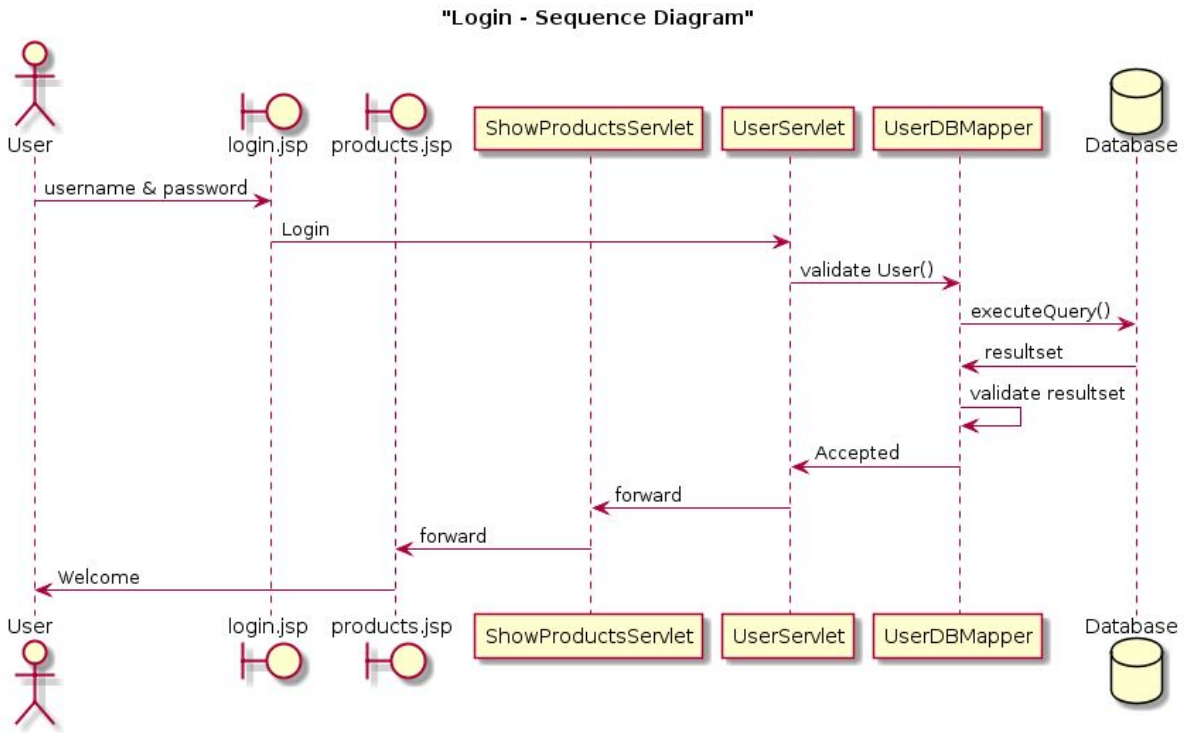
Home er vores "index" side, forside og hvor hele projektet begynder. Home introducerer projektet, og har referencer til tre andre sider(About, Contact og Login) vha. en universal navbar på alle fire sider.
About og *Contact* er begge to HTML-sider, som kun viser information på hjemmesiden, vha HTML, css, og bootstrap.
login er en JSP-side som giver muligheden for enten at logge ind eller registrere sig på samme side. Hvis du logger ind med en vellykket bruger, bliver du sendt over til vores products.jsp side.

Hvis du har fejl i dine login oplysninger bliver du sendt over til en *loginError*, som giver dig mulighed for at komme tilbage til login og prøve igen.
products.jsp er siden, hvor du kan købe dine cupcakes. Du vælger en bottom, topping, og et antal af den type cupcake, hvorefter du kan smide det ned i din *ShoppingCart*, som bliver gemt vha. en session. Hver gang man trykker "add to cart" på denne side, så hopper vi over i *ShoppingCart*, opdatere, og derefter returnerer til products.jsp, hvor vi får vist en liste af vores produkter.
Til sidste når man er tilfreds med sit køb, trykker man "checkout" og bliver sendt over til en *Invoice* side, som bekræfter ens ordre.

Sekvensdiagrammer

Her illustrerer vi tre forskellige sekvenser der sker igennem vores projekt. Et *Login* til at logge ind med, et *Register* til at lave et Login med, og til sidst vores *Add cupcake* der giver muligheden for at lave cupcakes. Læg mærke til at *Login* og *Register* finder sted på samme side.

Login - Backend



I vores Login sekvensdiagram, starter brugeren med at indtaste sine oplysninger på login.jsp siden vha. en form der laver en action. Formen kræver to parametre, *name* og *password*.

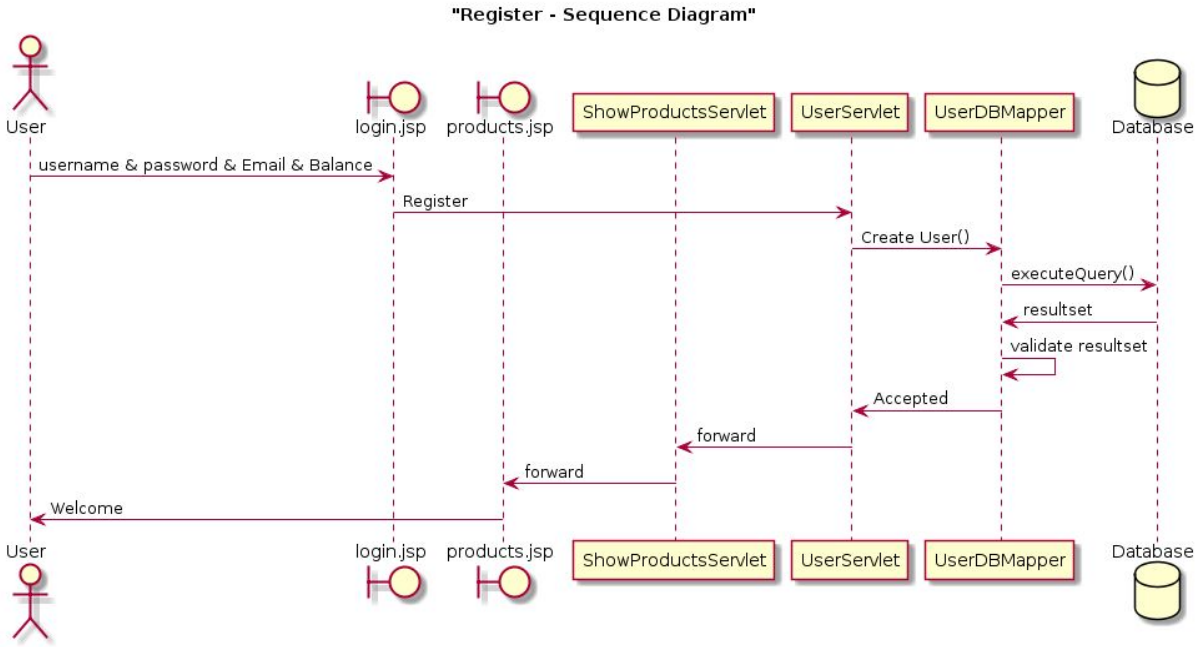
Når du har trykket Login med de valgte informationer, vil formen blive sendt videre over til vores UserServlet, som vil lave et user objekt vha. De to parametre du har indtastet på login.jsp siden. Derefter når *User* er lavet, vil UserServlet sende en request, som skal validere om *User*

objektets parametre eksisterer i vores database. Her bruger den UserDBMapperens *getUser* metode til at kommunikere med databasen.

UserDBMapperen starter med at lave en validering imellem User objektet fra UserServlet og databasens informations parametre i form af en *String SQL = "SELECT - FROM - WHERE"*, derefter laver den en *createStatement().executeQuery(SQL)* statement til at hente et resultset der matcher med vores 2 parametre fra *User* objektet op fra databasen(name og password).

Nu går processen så tilbage til vores UserServlet, som bruger et *if-statement*, der bedømmer ud fra vores resultater, hvad den skal gøre nu. Hvis brugeren eksisterer vil vi komme over til ShowProductsServletten, som går videre over til products.jsp siden vha. en *request.getRequestDispatcher(nextURL).forward(request,response)* ellers vil den gå til vores loginError.html page, som navigere os tilbage til login.jsp, og så må du prøve at logge ind igen.

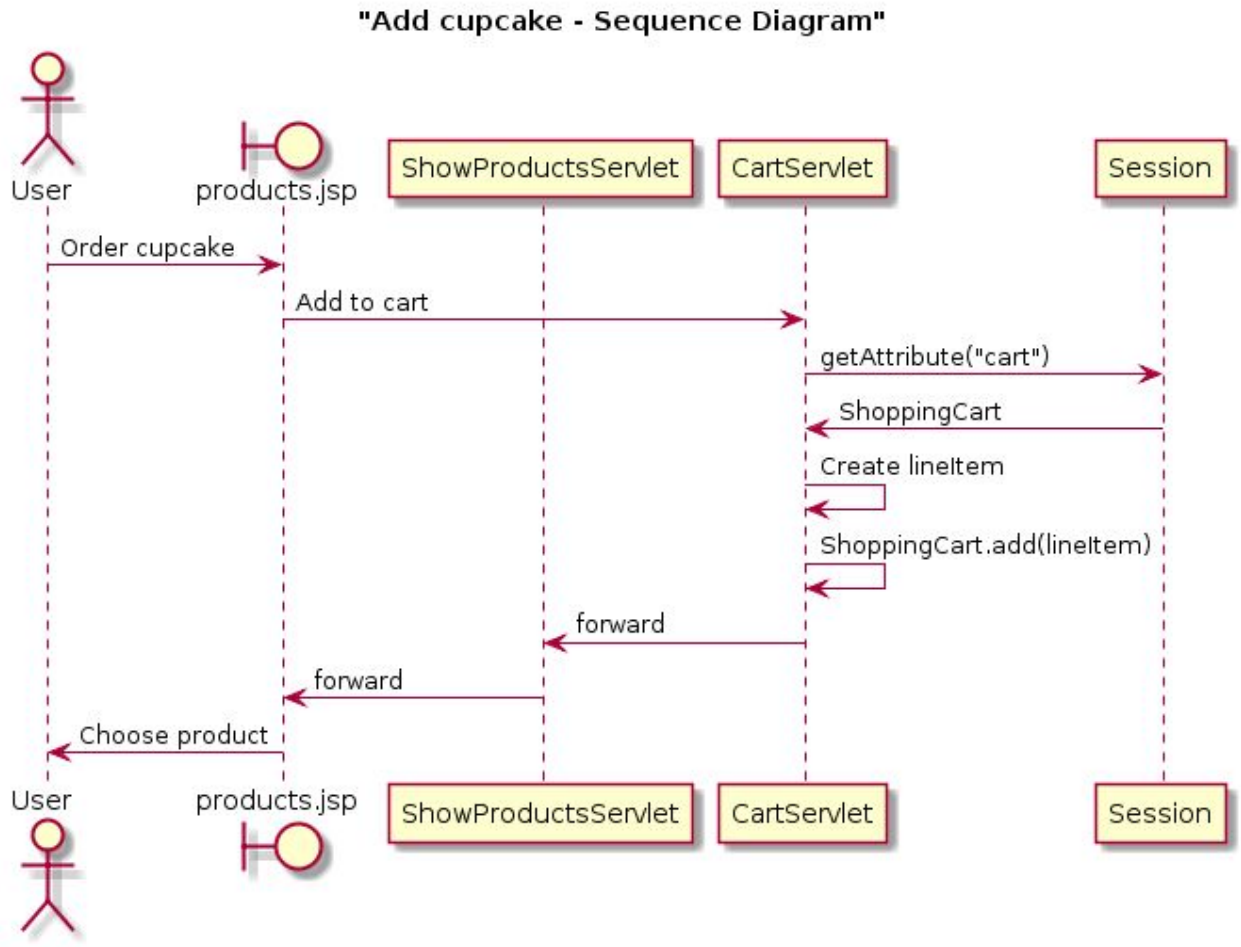
Register - Backend



Register fungere nogenlunde på samme måde, som login gør. Du indtaster 4 typer parametre og kalder *createUser* i UserServlet, men i stedet for at lede efter en allerede eksisterende bruger først, laver projektet en ny i databasen vha. UserDBMapperens *createUser* og derefter bruger værdierne til at logge på.

Den laver en SQL String med de fire parametre du har indtastet med *"INSERT INTO - VALUES"*, og bruger *createStatement().executeUpdate(sql)* til at opdatere databasen med en ny kolonne af information. Når databasen så er blevet opdateret går den op til vores getUser metode igen og validere om User() eksistere vha. validering igen. Derefter vil den blive sendt tilbage til product.jsp på samme måde som login.

Add cupcake - Backend



Vores sidste sekvensdiagram, er hvordan vi laver en cupcake i systemet.
Når brugeren er logget ind, kan kunden bestille deres egen type cupcake vha. En liste af topping, bottom og quantity på products.jsp siden. Listen af bottom og topping er vist vha. Et for-loop og det hele er sat ind i en form, som laver en action der bliver sendt over til CartServlet.

I CartServlet sker der 4 ting, først laver vi en *session.getAttribute("cart")***,** som returnerer en ShoppingCart til vores CartServlet. Så laver den 2 objekter og en integer udfra de parametre vi har indtastet på products.jsp siden(*bot, top og quantity vha Integer.parseInt()*)**.**
Nu bruger den så de tre typer information og skaber et *lineItem* af bot, top og quantity, som bliver smidt ned i ShoppingCart vha. *shoppingCart.addItem(lineItem).*
Når processen er færdig hopper den videre over til ShowProductsServlet, som til sidst viser en liste af LineItems på products.jsp siden.

Særlige forhold

I dette system har vi nogle forskellige særlige forhold der gør sig gældende.
Vi gør brug af sessions hvori vi gemmer Shoppingcart og User, så disse oplysninger kan gå igen på flere sider uden at gå tabt.
Eftersom det er et system med bl.a. Kontakt til en SQL-database, indeholder systemet mulighed for Exceptions, disse håndterer vi de fleste steder i koden med try-catch, men har dog også enkelte steder hvor at der bliver kastet exceptions, der så bliver catchet andre steder i koden. Dette gør er koden er inkonsistent i forhold til håndtering af exceptions, hvilket tages til efterretning til fremtidige projekter.
Systemet tager også imod bruger-input, fx i forbindelse med login og bestilling af cupcakes. Kun et sted i applikationen, sådan som det står nu, kan brugeren indtaste "forkert" input. Dette er når man skal registrere en bruger, og skal indtaste sin balance, så hvis man indtaster andet end en integer, sker der en fejl, og man bliver ledt over til fejlsiden. Sådan som applikationen står i skrivende stund, kan fejlsiden ikke fortælle hvilken fejl der er skyld i fejlsiden.
Brugeren kan dog også desværre indtaste ondsindet input der påvirker databasen, så som delete og drop table.

Status på implementation

Vi har oprettet en DBConnector, samt en UserMapper og en CupcakeMapper. Forbindelsen virker, og vi kan bruge vores mappere, men som nævnt tidligere er databasen ikke sikret imod ondsindet brugerinput som fx “drop table” da det ikke er lykkedes os at bruge Prepared Statement.
Det er muligt at oprette eller registrere en bruger. Forsøger man at logge ind med en ikke-eksisterende bruger eller med forkert password, kommer man til en login fejlside. Forsøger man at oprette en allerede eksisterende bruger, kommer man til en standard fejlside.
Ved login og registrering, tjekkes handlingen ved hjælp af en userservlet, samme servlet til begge handlinger, hvis det lykkedes sendes brugeren videre til produkt-siden via en productservlet.
På produktsiden vises brugernavn og balance for den bruger der er logget ind. Derudover kan man vælge en bund og en top til sin cupcake, vælge antal af valgte kombination, og lægge denne i kurven.
Designmæssigt kunne man sikkert vælge en mere tydelig måde at vise priser på de forskellige bunde og toppe i forhold til brugervenlighed, da det ikke fremgår tydeligt af siden at tallene ud for de forskellige muligheder er prisen for valgte mulighed.
ShoppingCart indeholder en liste af LineItems, og bliver gemt i session. ShoppingCart bliver vist på produktsiden, hvis denne ikke er tom. ShoppingCart viser også totalpris for den samlede ordre.
Vælger man at checke ud, bliver man sendt videre til en confirmation side. Lige pt er dette den eneste handling der udføres, når man trykker check ud. Brugerens ordre bliver altså ikke gemt, og brugerens balance bliver ikke opdateret, sådan som siden står nu. Der bliver heller ikke tjekket om brugeren køber for mere end hvad der står på dennes balance.
Databasen er dog klargjort med tabeller for at kunne modtage og gemme data om ordre, og dennes detaljer.
Der er endnu ikke oprettet en bruger-side, hvor kunden kan se sin ordrehistorik, og slå detaljer om den enkelte ordre op.

Der er heller endnu ikke oprettet en admin side, hvor en admin-bruger kan se alle ordre, og se disses detaljer. Databasen er i den forbindelse heller ikke gjort klar til forskellige bruger-roller såsom kunde og administrator.
Alle sider følger et grundlæggende design, med brug af en css- og bootstrap template hentet fra https://startbootstrap.com/