CupcakeProjekt

Webshop Olsker Cupcakes

Klasse D – Uge 11-12-13 - 2020

Thor Christensen – Cph-tc147@cphbusiness.dk – thorchris

Frederik Dahl – Cph-fd76@cphbusiness.dk – dahlfrederik

Josef Marc – Cph-jp325@cphbusiness.dk – josefmarcc

Hallur Simonsen – cph-hs228@cphbusiness.dk – halllursimonsen

Indholdsfortegnelse

Cupcake projekt – Klasse D – Josef, Hallur, Frederik, Thor
GIT – thorchris, dahlfrederik, josefmarcc, hallursimonsen

Introduktion

Til dette projekt er der blevet udviklet en webshop for ”Olsker Cupcakes”, hvor en kunde kan bestille cupcakes til afhentning. Man kan på siden bestille en cupcake med valgfri bund og top, man kan logge ind/oprette sig som bruger, man kan som administrator administrere siden med forskellige funktioner som indsættelse af credit, se ordre og brugeroplysninger osv. Derudover er der kurv siden hvor en kunde kan betale for sin cupcake med sin credit. Credit bliver tjekket om du har ”råd” og derefter er du klar til at afhente dine cupcakes.

Baggrund

Hjemmesiden er lavet til Olsker Cupcakes fra Bornholm. De ønskede en hjemmeside som kunne benyttes som webshop til køb af deres cupcakes. Olsker Cupcakes vil ikke kunne levere cupcakes men derimod vil der på siden kunne betales og derefter kan Olskers kundes bestilling afhentes.

Olsker Cupcakes har flere forskellige krav til deres nye webshop. Man skal på webshoppen kunne:

- Kunne bestille en cupcake med valgfri bund og top
- Kunne oprette sig som bruger
- Kunne sætte penge ind på en kundes bruger som admin
- Kunne se mine ordre på køb siden som kunde og kunne fjerne min ordre fra kurven igen
- Kunne logge ind som admin, med admin rettigheder
- Admin skal kunne se alle ordre og kunder i systemet
- Admin skal kunne slette ordre fra systemet

Teknologivalg

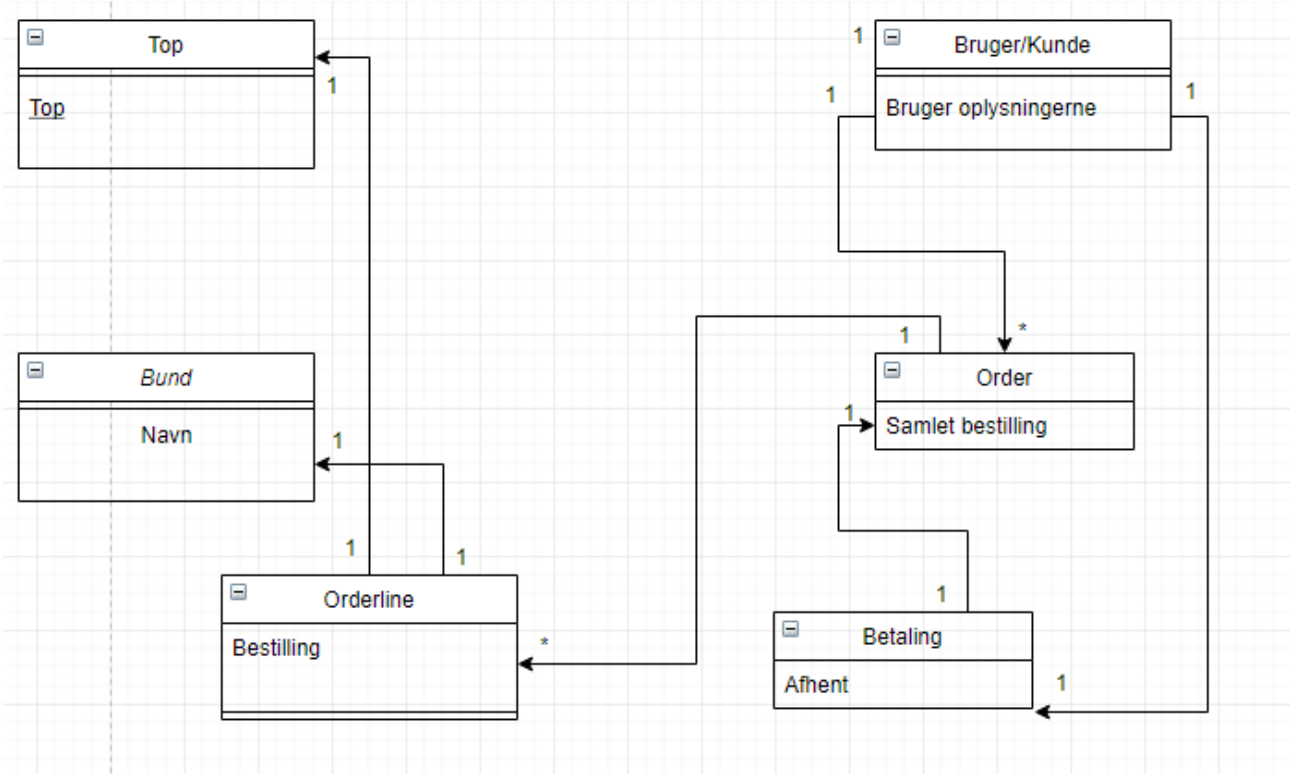
Software valg til udvikling af projektet:

- IntelliJ IDEA 2019.3.3 - Hereunder dependencies: BootStrap, Junit, MySQL connector. Samt regulært brug som IDE til Java, HTML, CSS, og Javascript.
- MySQL Workbench 8.0 CE - Database til brugere, ordre, og cupcakes.
- DigitalOcean Droplet – Host af serveren.
- Apache Tomcat 8.5.51 - Server
- JDK 1.8
- Git Bash 2.23.0
- Filezilla 3.47.2.1

Krav

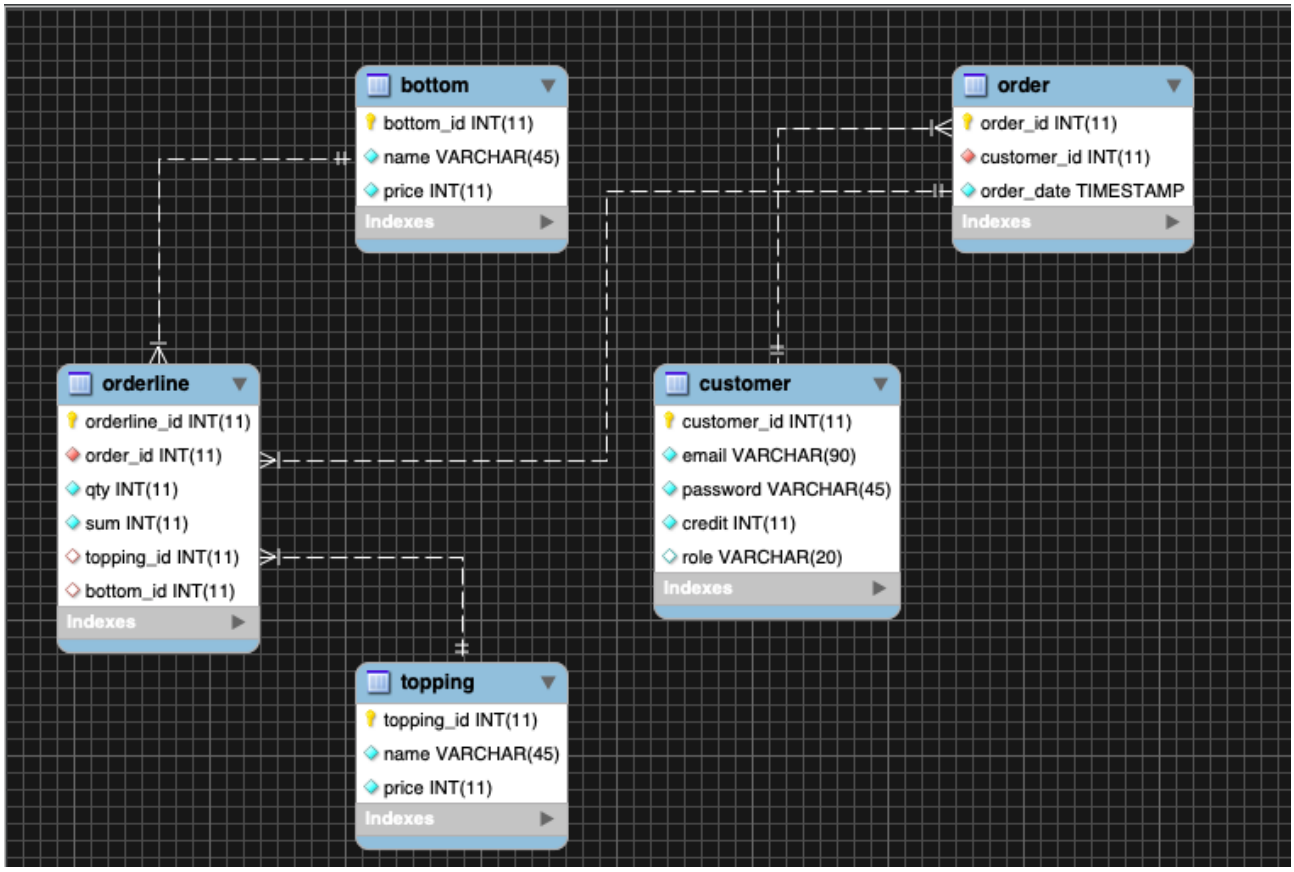
Kunden kan i højere grad koncentrerer sig om at producere cupcakes og øge salget ved at overlade bestillinger og statistik til systemet. Systemet håndterer ordrer, samt lægger dem i en database, hvilke gør det nemt og overskueligt efterfølgende at følge op på, hvilke cupcakes der er mest eftertragtet. Dette vil også gavne kundens kunder, da det er nemmere og hurtigere at bestille, eftersom alt er automatiseret og der ikke er et behov for menneskelig kontakt for at gennemføre en bestilling. Altså skaber dette system effektivitet og dermed økonomiske besparelser.

Domænemodel



I vores domænemodel fremgår problemdomænet, en bruger vil skulle kunne vælge en cupcake top og bund som gemmes i bestilling. Brugeren vil have en bruger hvor deres oplysninger gemmes. Brugerens oplysninger gemmes sammen med ordre detaljerne i ordre og denne kan en kunde så betale for.

EER-diagram



Et ER-diagram benyttes til at illustrere hvordan en database er struktureret.

Som det illustreres ER-diagrammet oppeover, er *orderline* en tabel der indeholder værdier fra de 4 andre tabeller. Den medtager altså *fremmednøglerne* order_id, topping_id og bottom_id. Der er en til mange relation for fornævnte felter og det betyder at en orderline kan indeholde flere forskellige order_id, topping_id og bottom_id.

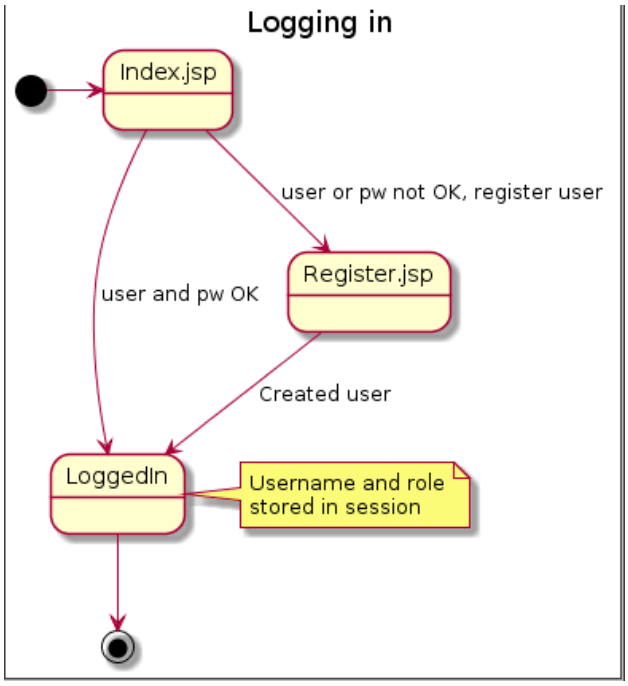
Udover dette indeholder *orderline* et *orderline_id* som er et autogeneret id der benyttes som *primærnøgle*. *Orderline* indeholder ligeledes to felter med værdier af typen *heltal*, *qty* og *sum*. Disse værdier repræsenterer antal af en cupcake i en bestilling samt den samlede pris for bestillingen.

De to tabeller topping og bottom er nærmest identiske idet de begge indeholder en *primærnøgle topping_id/bottom_id,* de indeholder ligeledes begge et felt med en navnestreng og et felt med et heltal som er prisen for henholdsvis topping eller bunden.

Tabellen customer indeholder en *primærnøgle customer_id* af typen heltal som bliver generet automatisk, tre felter af typen *strenge*, en e-mail, et password og role. Customer tabellen har ligeledes et felt med heltal som indeholder kundens kredit.

Order tabellen indeholder et autogeneret *order_id* af typen heltal, en *fremmednøgle customer_id* som referer til *customer_id* fra customer tabellen samt et felt med dato og tidspunkt for ordre oprettelse. Der er en til mange relation imellem customer og order, hvilket betyder at en kunde godt kan have flere ordre.

Navigationsdiagram



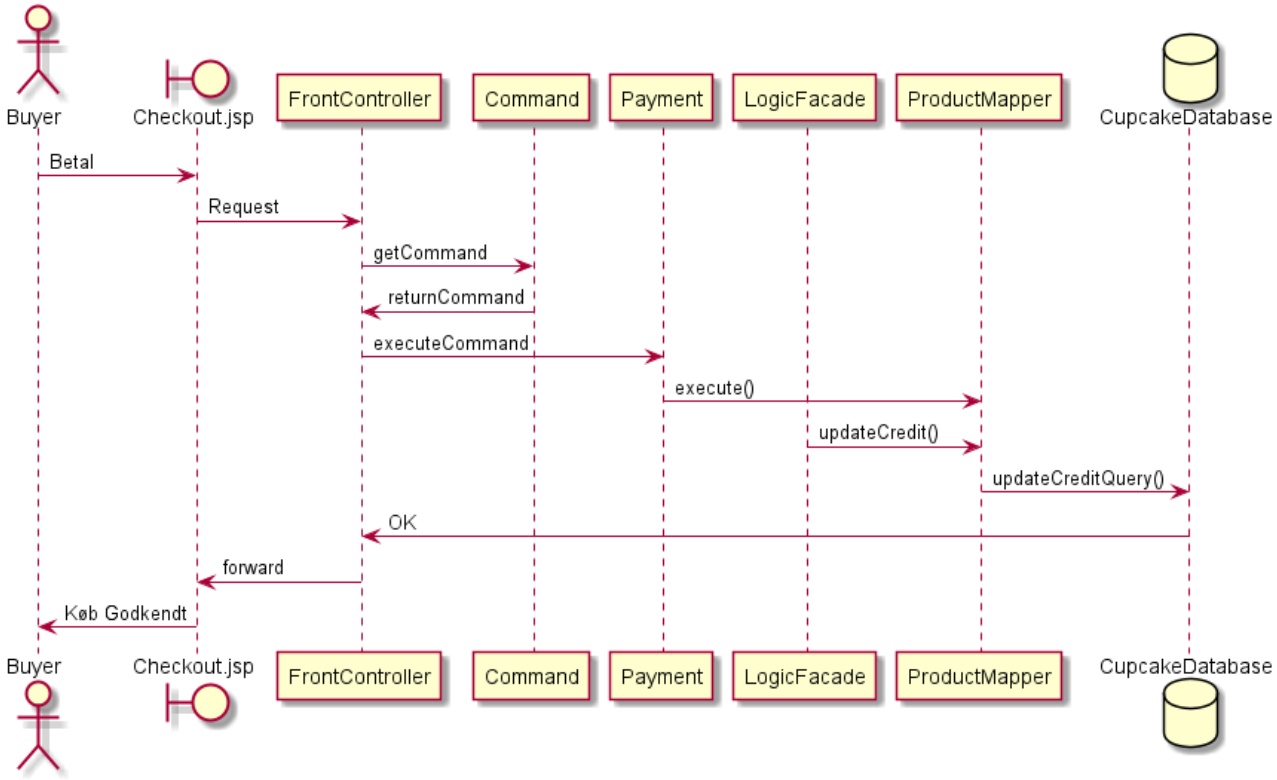
Navigationsdiagrammet vi har valgt, afbilder hvad der sker når en bruger skpå al logge ind vores webside.

Den fyldte sorte cirkel viser hvor vi begynder(ikke logget ind), og den halv fyldte cirkel viser hvor vi slutter(logget ind).

Først får brugeren adgand til index siden hvor brugeren, via navigations barren, kan logge ind. Om email og kodeord er korrekt bliver brugeren logget ind. Derimod om email eller kodeord er forkert, bliver brugeren automatisk omdirigeret til register siden, hvor brugeren så kan oprette en ny bruger. Efter opretning af en ny bruger, bliver den nye bruger logget ind.

Brugeren får også tilgivet en rolle. Vi har valgt at bruge 2 forskellige roller i databasen, Customer og Admin. Det ovenfor viste diagram viser hvad der sker for en Customer. Log ind for admin er lidt anderledes, da vi får adgang til en admin side så snart vi logger ind.

Sekvens diagram



Ovenover er indsat et sekvensdiagram. Et sekvensdiagram bruges til at vise hvordan et typisk forløb foregår i programmet.

Vores sekvensdiagram viser en køber, som har lagt cupcakes i sin kurv og er klar til at betale. Kunden er klar til at trykke på betal på checkout.jsp siden. ”Buyer” klikker betal og der sendes en request fra checkout.jsp til FrontControlleren som sender en command til Command som bliver udført og returnere commanden til FrontControlleren. Payment klassen extender command klassen og derfra bruges execute metoden til at sende betalings queryen til databasen. Kundes credit opdateres og du sendes tilbage gennem FrontControlleren til et ”Godkendt køb”.

Særlige forhold

- Der er ikke decideret validering af brugerinput idet det meste input stammer fra databasen og vælges i drop-down menuer.
- Alt der gemmes i session, er relevant information for brugeren, dvs. e-mail, password og brugerens bestilling

Status på implementationen

Der er nogle implementeringer i navigationsbarren der kunne udvides, fx når en bruger logger ind, skulle *login*knappen ændre tekst til at være en *log ud* knap i stedet for to separate knapper.

Dette har ikke været muligt med vores teknologivalg, da det til vores forståelse kræver JavaScript at ændre dette.

En anden ting i navigationsbarren, er at når man logger ind som admin, videresendes brugeren direkte til admin siden, men det er ikke muligt at navigere sig tilbage til admin siden når brugeren forlader den, f.eks. når man navigerer ind på webshoppens andre sider.

Der kunne i navigationsbarren implementeres en admin knap som først bliver synlig når en admin logger ind. Dette ville vi have lavet med to forskellige navigationsbarrer, en til c*ustomer* og *admin*.

Vi har også implementeret SVG billeder på index siden for at den cupcake på siden kunne skifte farve i toppen og bunden, alt efter brugerens valg af topping eller bund. Dette blev dog ikke gjort færdig udviklet, igen grundet teknologivalget og at dette krævede JavaScript til at ændre variablerne.

Når en kunde trykker betal uden at være logget ind kommer der en fejl besked op, som forklarer at man ikke kan betale uden at være logget ind. Den lyser grøn og er ikke den mest åbenlyse besked at finde. Denne kunne f.eks. være rød og mere åbenlys.

Vi har ligeledes forberedt to funktioner, muligheden for at ændre en brugers kodeord og slette en bruger. Disse er forbundet til databasen, men er ikke implementeret i webshoppens admin interface.

Der er en bug når der slettes fra kundens indkøbskurv, hvis kundens to cupcakes indeholder samme antal, samme top og bund – så slettes de begge fra indkøbskurven. Dette kunne udbedres ved at søge efter ordren på id og ikke på en toString().

Test

I dette projekt stammer størstedelen af dataene fra en database og det har derved været svært at teste dele af projektet idet datamappere vil kaste en exception i tilfælde af fejl hor forbindelsen til databasen ikke bliver oprettet.

Alle vores test er baseret på klasser i util pakken, som er metoder der udfører matematiske udregninger. Disse er kun testet med positive tal, idet vi ikke behøver tage højde for inputvalidering da parametrene metoderne medtager hentes fra databasen og brugeren har derved ikke mulighed for at påvirke disse.

Der er således kun udført test-coverage på metoderne der udregner cupcake priser.

Undervejs i hele projektet er der løbende udført test af metoder med kald fra static metoder, for at undersøge om disse fungerer. Dette benyttede vi fx ved createOrder() hvor der var opstået en

logisk fejl, idet der manglede ”ps.execute();”, men koden compilede og kørte, dog uden at gøre noget.

Vi har ligeledes benyttet IntelliJ’s debugger samt Google Chromes udviklerværktøj til at undersøge hvilke parametre og værdier diverse metoder har medtaget fra JSP siderne og omvendt fra Java til JSP:

Til fremtidige projekter vil vi i stedet for test i static metoder teste nye metoder direkte i unit test (hvis muligt), samt have planlagt flere unittests.