Olsker Cupcakes

Dokumentation

Udarbejdet af gruppe 555 fra klasse-e:

- **Navn: Henrik Thomasen**
  Email: cph-ht92@cphbusiness.dk
  Github: https://github.com/HenrikCPH

- **Navn: Mark Sørensen**
  Email: cph-ms845@cphbusiness.dk
  Github: https://github.com/markmps

- **Navn: Nicolas Allesøe**
  Email: cph-na157@cphbusiness.dk
  Github: https://github.com/NicoCPH

- **Jean-Poul Leth-Møller**
  Email: cph-jl360@cphbusiness.dk
  Github: https://github.com/Jean-Poul

Indholdsfortegnelse

Indledning

Vi har fået til opgave at oprette en webshop, for et bornholmsk økologibageri, kaldet Olsker cupcakes. Inden man skal i gang med at kode, skal der laves et mockup, ved hjælp af Adobe xd, som er til for at kunne vise en skitse af ens website og hvordan man skal kunne navigere rundt på sitet.

Websitet skal være dynamisk og vil blive kodet ved hjælp af bl.a. HTML, CSS, Bootstrap, Java og SQL for til sidst at kunne køre det igennem en Tomcat webcontainer. Mere om dette i afsnittet *teknologi valg*. Når der siges dynamisk, menes der, at man kan gemme og hente data fra en database. Databasen normaliseres på 3. normalform og vi forholder os ikke til 4 og 5 normalform i dette forløb.

Vi skal udvikle det endelige produkt ud fra en skabelon, som vi har fået tildelt. Denne skabelon bruger et design mønster kaldet MVC, som er et framework, der står for Model, View og Controller. Det bruges i programmering til at adskille ens kode, for at gøre det mere overskueligt og nemmere at vedligeholde. Det endelige produkt skal køre på en Droplet hos Digital Ocean. Kildekoden skal være tilgængelig på GitHub.

Opgaven er opdelt i to dele med en administrator del og en kunde/bruger del. Olsker cupcakes ønsker en administrator side, hvor man skal kunne se en oversigt over alle nuværende kunder og deres køb, samt kunne slette ordre i systemet. Vi har valgt at tilføje endnu en egenskab. Forklaring følger i afsnittet *baggrund*.

Som kunde, på siden, skal man kunne oprette en profil for at kunne betale for sine ordre. Ordrerne består i at man kan vælge imellem forskellige toppe, bunde og antal af cupcakes for herefter at kunne betale. I denne opgave skal kunden ikke selv kunne indsætte kredit, men derimod, er det noget, som en administrator vil stå for igennem en administrator siden. En kunde skal også kunne se en indkøbskurv med en ordrelinje samt en samlede pris. Kunden skal derudover også kunne slette en ordre fra sin ordrelinje.

Til sidst skal websitet vise hvem man er logget ind som.

Baggrund

Det økologiske bageri, Olsker Cupcakes, har valgt at transformere sin virksomhed til det digitale marked. Vi har fået til opgave at udarbejde en webshop, som tilfredsstiller deres vision og som kan anskaffe dem flere kunder. Brugerfladen skal være brugervenlig og nem at tilgå, da det ved første øjekast er første gang kunden har bevæget sig ind i den digitale verden.

Efter et møde med kunden fandt vi frem til følgende user-stories, som fortæller hvilke bruger har hvilke behov og hvad der ønskes at opnås ved denne løsning.

US-1: Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, sådan at jeg senere kan køre forbi butikken i Olsker og hente min ordre.

US-2 Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en ordre.

US-3: Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySQL, så en kunde kan betale for sine ordrer.

US-4: Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.

US-5: Som kunde eller administrator kan jeg logge på systemet med e-mail og kodeord. Når jeg er logget på, skal jeg kunne min e-mail på hver side (evt. i topmenuen, som vist på mockup'en).

US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

US-8: Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.

US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

US-10: Som administrator kan jeg søge efter en kunde og se de tilhørende ordrer. Denne user-story har vi valgt at tilføje, da vi mener det er til stor gavn for overblikket, som administrator.

Teknologi valg

Følgende teknologier er blevet anvendt for at kunne udarbejde projektet og holde kommunikationen intakt i vores gruppe.

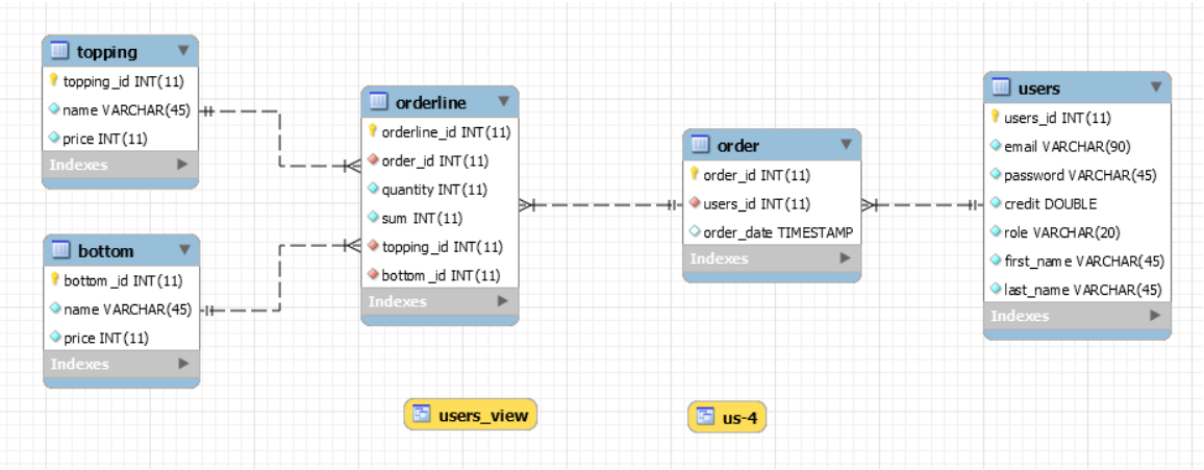
Adobe Creative Cloud	- Version 5.1.0.407
Adobe XD	- Version 28.2.12
Draw.io	- Version 11.3.0
Discord	- Version 0.0.306
IntelliJ	- Version 193.6494.35
Java 8	- Version 8.0.2410.7
FileZilla Client	- Version 3.46.3
Git bash	- Version 2.23.0.windows.1
Google Chrome	- Version 80.0.3987.149
Linux Ubuntu OS	- Version 5.3.0.29-generic
MySQL Connector/Workbench	- Version 8.0.18
Photoshop	- Version 20.0.7
PuTTY	- Version 0.73.0.0
Tomcat	- Version 8.5.5.1
WinRAR	- Version 5.80.0
Word	- Version 18.2002.1101.0
Windows	- Version 10 18362.720

Mockup

Vi har lavet og added et Mock-Up til vores projekt. Mock-Up'en er lavet som en af de første ting i vores projekt og vi har taget udgangspunkt i den. Med det sagt er der lavet ændringer i Mock-Up'en efterfølgende, som ikke er blevet updateret og vi har derfor ikke en updateret version af Mock-Up'en. Med mere tid kunne man havde lavet en updateret version man kunne have sammenlignet med den oprindelige version.

Diagrammer

EER- Diagram:



Der er brugt 3. normal form/(3NF) i databasen i vores cupcake projekt. Dette betyder at bordet skal opfylde de krav der stilles til 2. normal form og at databasen kun indeholder kolonner som er ikke-transitivt afhængig af en primærnøgle.

Vi har også brugt CASCADE i databasen for at kunne slette brugere, ordre og ordrelinje korrekt.

CASCADE: CASCADE er brugt på fremmednøgler, til at fjerne tilhørerne informationer. Hvilket skal forstås på den måde, at hvis man sletter en bruger som har lavet en ordre, som også indeholder div. Ordrelinjer, så kan man kun slette denne bruger hvis også man sletter de ordre og ordrelinjer som tilhører brugeren. Dette kan lade sig gøre med CASCADE.

Vores ER-diagram består af 5 borde: users, order, orderline , bottom og topping. Udover de 5 borde så er der 2 visninger, users-view og US-4.

US- 4: Den første visning ved navn US-4, viser en oversigt over alle order_id, som er et id der bliver tildelt en ny ordre hver gang en ny ordre bliver lavet. orderline_id som er et id der bliver tildelt en ny ordrelinje hver gang en ny ordrelinje bliver lavet. quantity, er antal af cupcakes kunden har bestilt. sum, er den samlede pris kunde skal bestale for sin ordre. Topping_name og bottom_name er henholdsvis navnene på cupcake bundende og toppene. Topping_price og bottom_price er henholdsvis priserne på de cupcakebunde- og toppe som kan bestilles. Meningen med US-4 visningen, var at kunne vise kunden en oversigt over

hvad kunde havde bestilt. Vi brugte ikke US-4 visningen i vores projekt, men det havde været en god fremgangsmåde at bruge US-4 i stedet for en SQL-String af SELECT for at løse opgaven af samme navn, US-4.

users-view: Den anden visning er lavet, for at vise alt tilhørende brugerne på vores cupcake hjemmeside. Visning bruges kun til at kunne se disse informationer i databasen, for hurtigt at kunne for et overblik over brugerne. Udover de nævnte informationer fra US-4 visningen, indeholder users-view også: first_name, hviket er fornavnet på brugeren. last_name, hvilket er efternavnet på brugeren. users_id, er et id alle brugere for tildelt, når de bliver lavet. email, brugerens email. password, brugerens password. order_date, dette er datoen på hvornår ordren er bestilt. topping_id og bottom_id, er henholdsvis id'er på de toppings og bunde cupcake hjemmesiden tilbyder.

Primær nøgle/PRIMARY: En primær nøgle definerer et bestemt felt i et bord. Et bord må kun have en primær nøgle og ikke være NULL, NULL betyder at den SKAL indeholde noget. De andre felter i bordet er afhængige af primærnøglen.

Fremmed nøgle/fk: En fremmed nøgle er et eller flere felter i et bord som identificerer en række i et andet bord. bordet som fremmednøglen er defineret i, kalder man for børneborderet og refererer til en primær nøgle i et andet bord.

Unik/UNIQUE: Unik er et eller flere set af koloner i et bord, som identificerer en registrering i et bord. En unik nøgle gør at værdien i en kolonne er unik i hele bordet.

users: Indeholder, user_id (PRIMARY), email (UNIQUE), password, credit, role, first_name og last_name. Der er en til mange relation, imellem users bordet og order bordet.

order: Inderholder, order_id (PRIMARY), users_id (fk_user_order_id) og order_date.

Der er en til mange relation, imellem order bordet og orderline bordet.

orderline: Inderholder, orderline_id (PRIMARY), order_id (fk_orderline_order_id), quantity, sum,

topping_id (fk_orderline_topping_id), bottom_id (fk_orderline_bottom_id).

Der er en til mange relation, imellem orderline bordet og topping bordet.

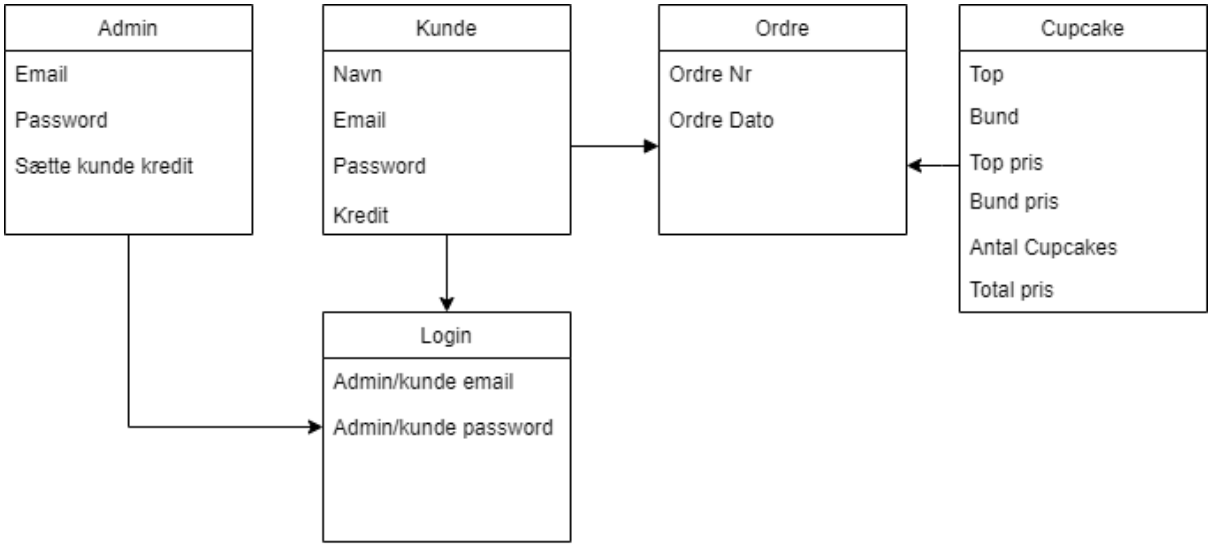
Der er en til mange relation, imellem orderline bordet og bottom bordet.

topping: Inderholder, topping_id (PRIMARY), name og price.

bottom: Inderholder, bottom_id (PRIMARY), name og price.

Domæne model

Klienten ønsker at få en hjemmeside hvor man kan bestille en cupcake, lav en ordre, oprette en bruger med e-mail og password samt få fornavn og efternavn på brugeren, kunne login med e-mail og password og derudover skal der være en admin som kan administrere systemet, hvor de kan udføre diverse metoder til kunden m.m.



Domæne modellen består derfor således af tabellerne: Admin, Kunde, Login, Ordre og Cupcake.

Relationen mellem disse tabeller bliver skrevet således:

Kunde til ordre er en til mange relation, da en kunde kan afgive flere ordre.

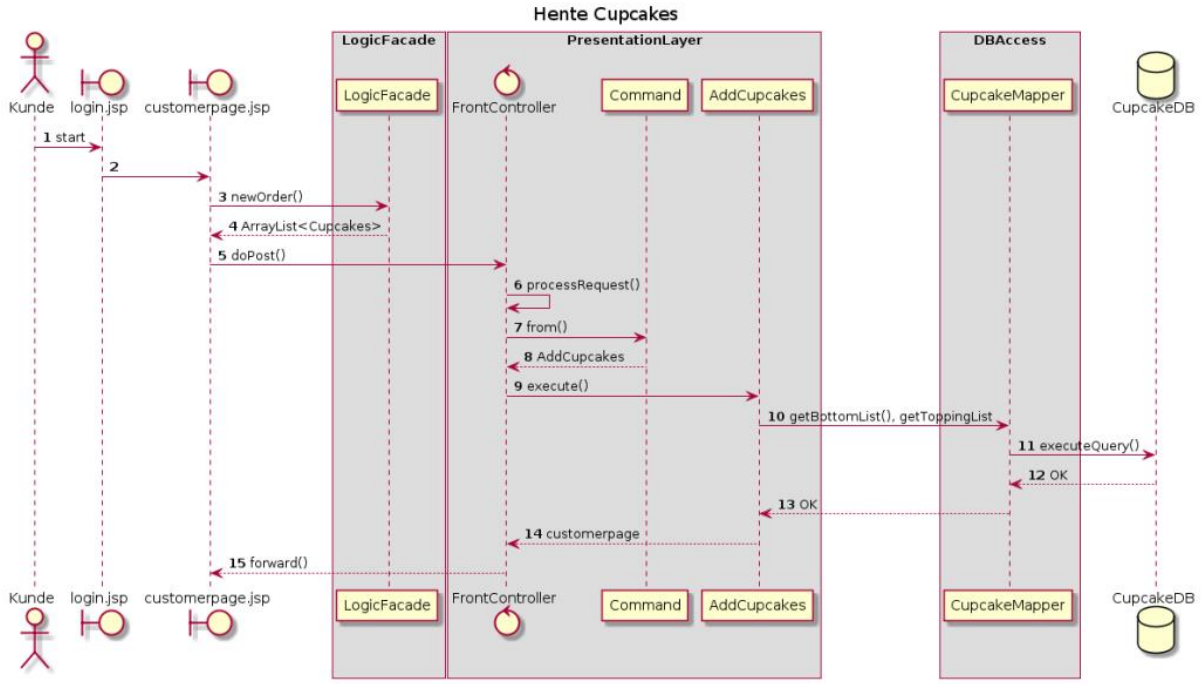
Ordre til Cupcakes har en mange til mange relation, da en ordre kan have indeholde mange cupcakes og cupcakes kan afgive til flere ordre.

Admin og kunde har en til en relation med login.

Sekvensdiagram

Sekvensdiagrammet har fokus på at hente cupcakes.

Sekvensdiagrammet viser hvilke metoder der bliver kaldt i de forskellige klasser og hvad der returneres tilbage når en kunde ønsker at bestille en cupcake.



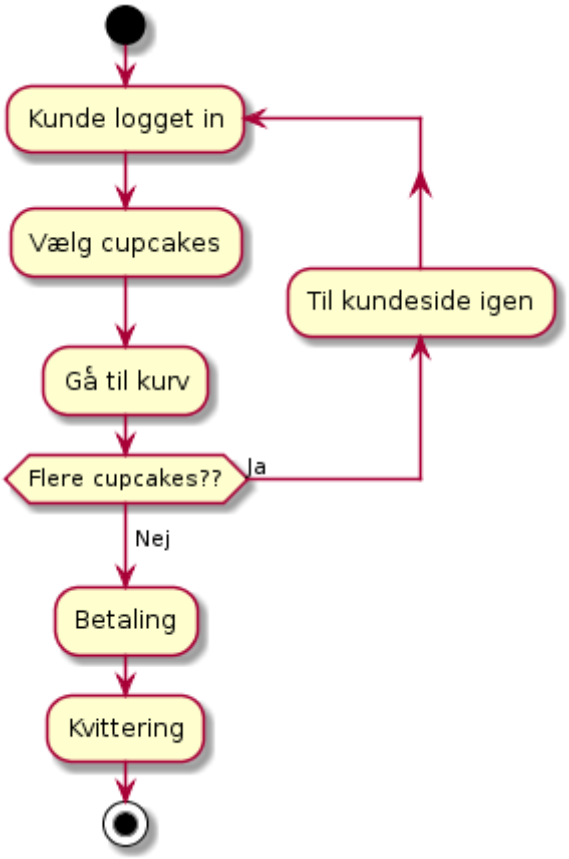
Scenariet for, at hente cupcakes sker i 15 trin og beskrives således forneden:

1. Kunden logger ind på siden med sin e-mail og password.
2. Efter loginoplysningerne er ok, bliver kunden sendt over til customerpage.jsp.
3. DropDown Java klassen, som er tilsvarende til customerpage.jsp fra Command, så kalder den en metode i LogicFacade som hedder newOrder() som initialisere et ArrayList af cupcakes for, at kunne bestille cupcakes.
4. Metoden returnerer et ArrayList af cupcakes.
5. customerpage.jsp kalder metoden doPost() fra FrontController.
6. FrontController kalder sin egen metode processRequest().
7. FrontController kalder metoden from() fra Command, som skal hente AddCupcakes klassen fra dens HashMap
8. AddCupcakes klassen bliver returneret
9. execute() bliver så kaldt fra AddCupcakes
10. DropDown kalder så metoderne getBottomList() og getToppingList() fra CupcakeMapper, som skal hente et topping id og bottom id fra databasen.
11. CapcakeMapper kalder preparedStatement som laver to executeQuery() og henter bottom id og topping id fra databasen, så kunden kan finde navnene tilsvarende til deres respektive id's.
12. Query bliver udført og information sendes tilbage til CupcakeMapper.
13. Den information sendes tilbage til AddCupcakes.
14. DropDown returnerer customerpage til FrontController.

15. FrontController returnerer så metoden forward() som indeholder de request.getAttribute() værdier som er i klassen.

Aktivitetsdiagram

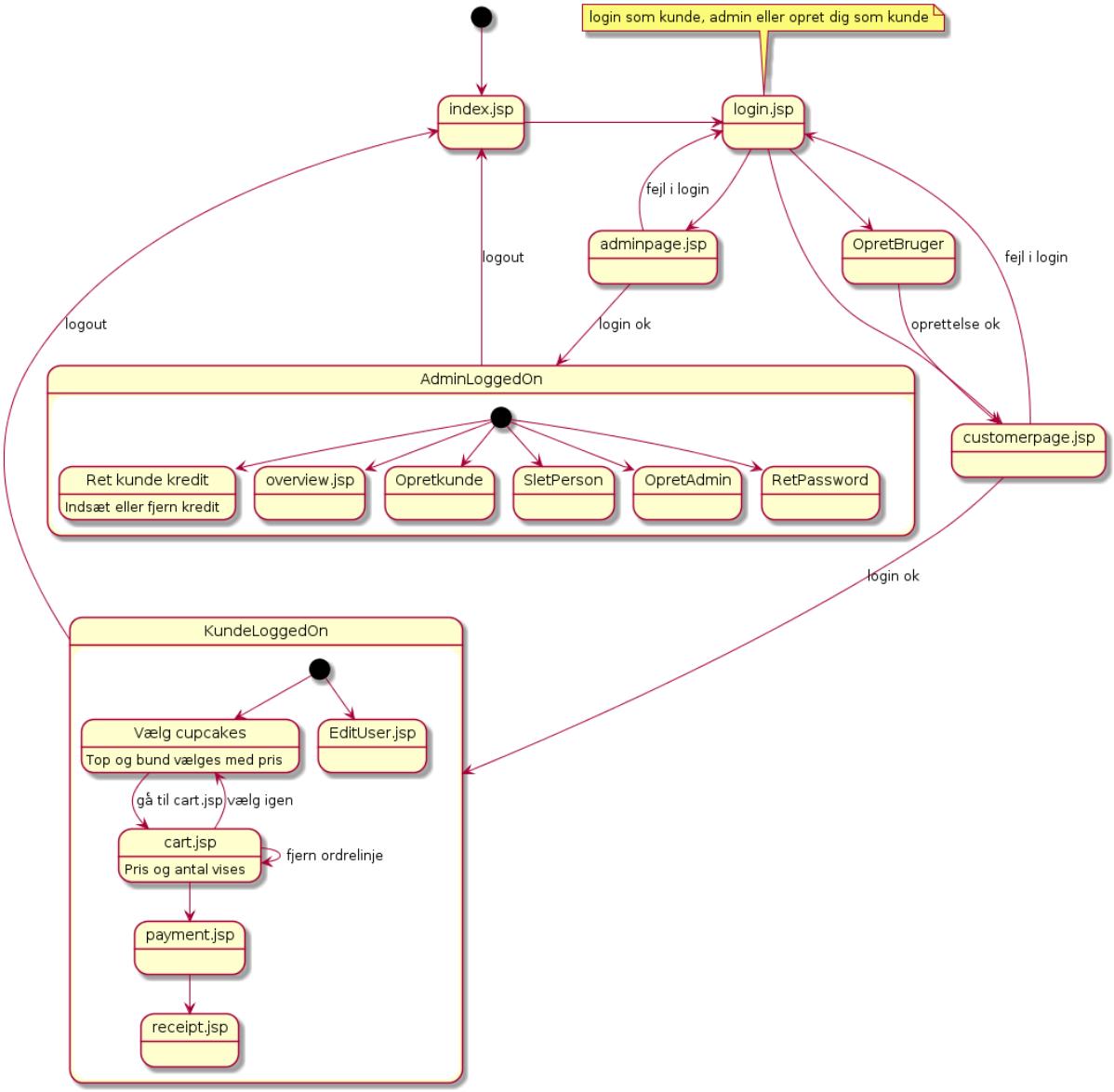
Aktivitetsdiagrammet viser kort hvordan begivenheden foregår når en kunde vil bestille



cupcakes fra siden og dette sker således:

Kunden starter med at være logget ind. Kunden kan så derefter vælge cupcakes med forskellige bunde og toppe, samt se prisen på det. Kunden kan vælge at gå til kurven. Hvis kunden ikke er færdig med at shoppe, så kan kunden bestille flere cupcakes. Hvis kunden derimod er færdig, så sendes de videre til betaling hvor de derefter får en kvittering på bestilling.

Navigations/tilstandsdiagram



Navigations/tilstandsdiagrammet viser hvordan man navigerer rundt i systemet. Den sorte ''knap'' indikere, at det er her vi starter når programmet køre. Man starter i index.jsp hvor man derefter går videre til login.jsp. Dernæst bliver man videresendt til henholdsvis adminpage.jsp eller customerpage.jsp hvis loginoplysningerne er ok. Hvis loginoplysningerne ikke stemmer overens, bliver man sendt tilbage til login.jsp. Når oprettelsen er ok, sendes kunden videre til customerpage.jsp.

Admins starter i adminpage.jsp, hvor de kan gøre diverse ting så som, at rette en oprette en bruger, redigere en kundes kredit m.m. Hvis admins vil se oversigten over kunde, så sendes man videre til overview.jsp. Man kan også logge ud med logout knappen, hvor man så vender tilbage til index.jsp.

Som kunde kan man redigere sin bruger, hvor man så bliver viderestillet til edituser.jsp. Kunden kan vigtigst af alt bestille cupcakes og når kunden har valgt et antal de er tilfredse med, så kan de trykke på kurven som så sender en videre til cart.jsp. Når kunden er tilfreds, kan de trykke på betaling som sender en videre til payment.jsp og til sidst får de en kvittering som sender en til receipt.jsp. Kunden kan også logge ud og når de har logget ud, ender de tilbage i index.jsp.

Særlige forhold

Informationerne som gemmes i session, er en brugers/administrators:

email(brugeren/administratoren e-mail).

user(hvem er logget ind).

userid(hvilket id har brugeren/administratoren).

role(hvilken rolle er du: bruger/administrator).

credit(hvor meget kredit en bruger/administrator, har på sin konto).

credit-total(credit-total er den nuværende kredit brugeren/administratoren har, minus det totale beløb som brugeren/administratoren skal betale for sin ordre).

totalcredit(hvor meget kredit en bruger/administrator har efter et køb).

Invalidate()(logger brugeren/administratoren ud).

neworder(laver en ny ordre).

sum(prisen på toppings og bunde gange antallet).

Vi håndterer exceptions på følgende måde:

I en java klasse har vi en throw new LoginSampleException, som håndteres i en catch, herefter genereres en besked gennem LoginSampleException java klassen, som så grippes i FrontControlleren for så til sidst at smide brugeren til error.jsp siden som fremviser fejlen.

Vi har valgt at lave validering af brugerinput på følgende måde:

Vi har lavet en if – else i de forskellige klasser, hvor man skal logge ind og oprette sig.

Vi har valgt at lave sikkerhed i forbindelse med login på følgende måde:

Man opretter en bruger og for at kunne logge ind skal man angive sit kodeord og en rigtig e-mail. E-mail påkræves med et @, som vi har gjort ved syntaksen type=”email”. En bruger rolle bliver tildelt når man bliver oprettet. Enten er man customer eller admin og har dertil forskellige muligheder.

Hvilke brugertyper, er der valgt i databasen, og hvordan bruges de i JDBC:

Der er 2 brugertyper, admin og customer. De bruges ens.

Status på implementation

Model:

Denne del fylder mest i vores projekt og har derfor også fået det meste af vores opmærksomhed. Det er her vi har vores data access objekt klasser. Vi mener at vi har været igennem kravene til denne opgave, men hvis tiden var der, ville vi have navngivet flere metoder anderledes og et par variabler. Opsætningsmæssigt kunne det også have være lavet mere strømlinede, men det har været en god lærings proces for os omkring håndtering af tiden og planlægning.

View:

Vi har, i dette projekt, fået konstrueret den webshop, som vi først havde lavede et mockup til, dog med et par ændringer, som blev rettet, da vi stødte på dem imens vi fik kodet. Bl.a. blev der lavet en ekstra side til den specifikke kundes ordre. Navigationen har været til diskussion flere gange. Det omhandlede brugervenlighed og optimering. Det blev i sidste ende et spørgsmål om tid. Pga. dette blev administrator og bruger navigationen stort set ens. Vi fik dog implementeret det sådan, at en bruger ikke kan komme ind på administrator linket, da man skal have en rolle som administrator for at kunne tilgå det link.

Vi har fået stylet vores side, men igen pga. tidsmanglen, tog vi den beslutning at fokusere mere på back end programmering, da det vigtigste, i vores øjne, ville være at få alt funktionaliteten op at stå først. Der var planer om at style siden mere og gøre hele oplevelsen mere brugervenlig.

Javadoc er blevet lavet på vores execute metode i java klassen DropDown. Hvis vi havde mere tid, ville der blevet lavet javadoc til alle klasser og deres metoder.

Vi har derudover valgt at dokumentere alle kommentere på engelsk, med et par undtagelser. Vi tog denne beslutningen fra start. Alt skulle foregå på engelsk, men alt output til brugeren foregår på dansk, da det er en dansk virksomhed for danske brugere.

Controller:

Frontcontroller tager imod en forespørgsel (request) og sender herefter et svar (response) tilbage. Da dette var en udleveret skabelon, er der kun blevet tilføjet en try/catch med følgende kode, for at kunne håndtere æøå fejl til output på jsp siderne:
request.setCharacterEncoding("UTF-8");

response.setCharacterEncoding("UTF-8");.

Derudover har vi tilføjet request.setAttribute("error", ex.getMessage()); til vores catch med LoginSampleException, for at kunne sende brugeren videre til en error.jsp side, hvis en fejl skulle opstå.

Session og request scope har vi efter alt at dømme anvendt rigtigt, men da det er første gang vi arbejder med det, kan der have været begået et par fejl med dette. For en god ordensskyld, så ved vi at der er application, session, request og page scope, men har i vores projekt kun taget brug af session og request scope.

Vi har i dette projekt ikke haft fokus på unit test, men kunne have været til stor gavn for os til validering af flere af vores metoder.

Til sidst kan der nævnes, at Github kun er blevet brugt på den måde at en person skubber (git push) alt op, hvorefter de andre kunne hente (git pull). En større fokus på dette, samt unit test ville have været til stor hjælp for os, men vi kan godt se at tiden ikke har været der til det.

Konklusion

Forløbet har været svært, da vi har været hæmmet, af at vi ikke har haft mulighed for at kunne sidde ved siden af hinanden. Det krævede tilvænning og har bestemt gjort forløbet langsommere end det burde. Man har ikke haft tid til at kunne sidde med alt sammen, så der har været uddelegeret opgaver. Ved denne metode er der nogen som står stærkere på visse punkter i opgaven end andre. Vi har dog været igennem alle krævende til opgaven og noget i mål, med de løsninger vi havde sat os for øje. Det har været svært at få hjælp fra vores undervisere og vi kan forestille os, at det også har været en tilvænnings fase for dem, men mere tilgængelighed ville have været til stor hjælp.

Vi har fået større forståelse for MVC-frameworket, database og sammensætningen med Java kode og gruppe arbejde, samt tidsfordeling.

Med mere tid eller en større forståelse på hvad vi skulle fokusere på, ville vi kunne have gjort koden mere brugervenlig og mere strømlinet, samt fået anvendt UX/UI mere på vores website.