Projekt: Cupcake

Deltagere (Navn – Mail – Github):

Alle deltagere er fra hold b.

- Kristofer Pedersen – cph-kp278@cphbusiness.dk – ”Krisdanoda”
- Michael Frederiksen – cph-mf315@cphbusiness.dk – ”Magical Glove”
- Sofia Mathilde – cph-sm436@cphbusiness.dk – ”sofiadoingcode”
- Søren Merved– cph-sm428@cphbusiness.dk – ”SMerved”

Projektet blev udarbejdet fra d. 04/04-2022 til d. 08/04-2022.

Indholdsfortegnelse

Indledning

Vi fik til opgave at bygge en funktionel hjemmeside til et cupcakebageri ved navn Olsker Cupcakes. Opgaven er en øvelse for os datamatiker studerende på 2. semester, så vi yderligere kan forstå, hvordan man kan tage en klients behov og formidle det til en funktionel hjemmeside. Firmaet er derfor fiktivt. Denne rapport er skrevet til en "fagfælle", hvilket vil sige en datamatiker studerende på 2. semester. Denne studerende kender ikke til cupcakeopgaven.

Baggrund

Olsker Cupcakes er et cupcakebageri på Bornholm med et simpelt koncept. De tilbyder deres kunder at sætte en hvilken som helst cupcake bund, sammen med en hvilken som helst cupcake topping. Dette ønsker de, at kunderne kan bestille fra en hjemmeside, som de har bedt os om at bygge.

På hjemmesiden skal man kunne logge ind som enten administrator eller kunde. Kunden skal på hjemmesiden kunne bestille én eller flere cupcakes med en vilkårlig top og bund. Ordren skal lægges i en indkøbskurv, der er tilgængelig og kan redigeres af kunden, indtil ordren er betalt. Administratoren skal have adgang til en liste over alle kunder samt deres ordrer. Administratoren skal derudover kunne fjerne kunder og ordrer.

Teknologi valg

Vi har brugt disse teknologier i vores projekt:

- IntelliJ – 2021.2.2
- JDBC – 4.3
- MySQL – 8.0
- Java -- Version 17
- CSS – 4.15
- HTML – 5
- Bootstrap -- 5
- TomCat -- 9.0.60
- Maven – 3.6.3

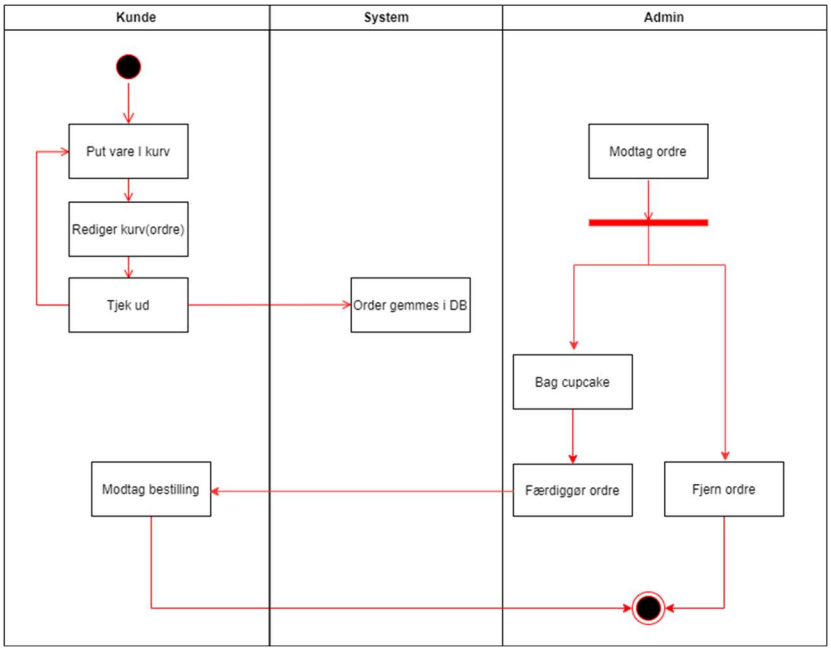
Krav

Vi modtog en række specifikke krav til hvad hjemmesiden, som minimum, skulle kunne håndtere:

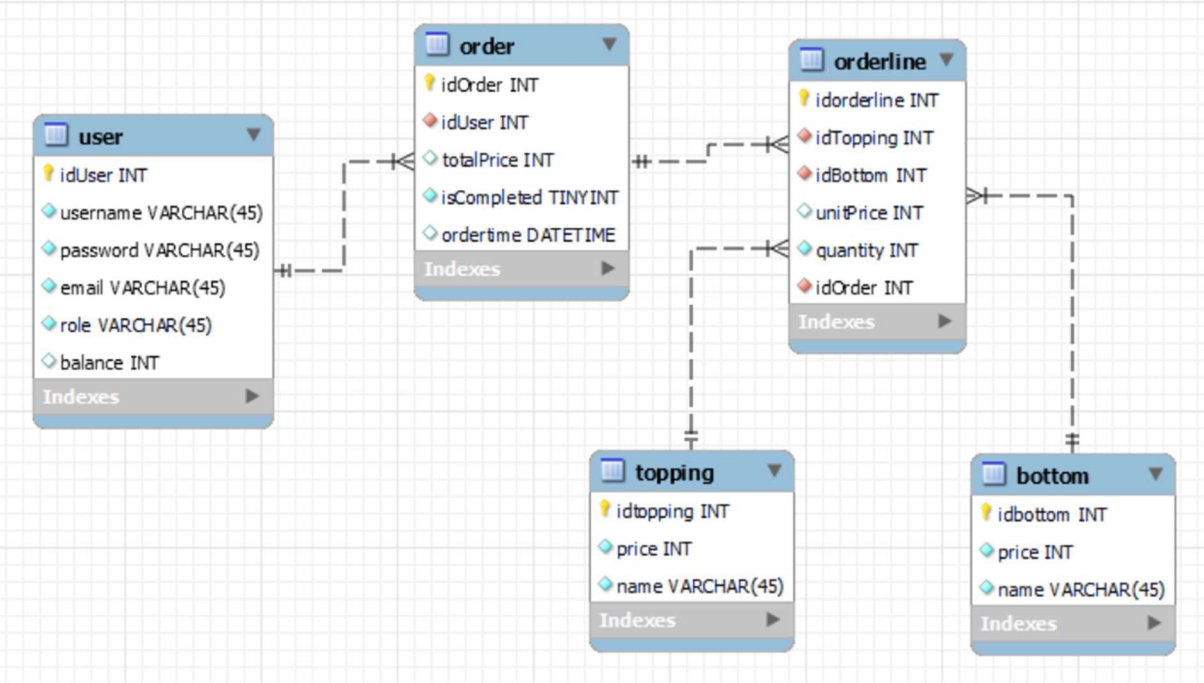
- Kunden skal kunne bestille og betale cupcakes med en valgfri bund og top, som derefter kan hentes i bageriet.
- Kunden skal kunne oprette en konto for at kunne betale og gemme en ordre.
- Administratorer skal kunne indsætte beløb på en kundes konto direkte i MySQL, så kunden kan betale for sin ordre.
- Kunden skal kunne se sine aktive ordrelinjer i en indkøbskurv, samt den samlede pris for ordren.
- Man skal kunne logge på systemet, som kunde eller administrator, med brugernavn og kodeord. Derudover skal brugerens e-mail være synlig på hver side.
- Administratorer skal kunne se alle ordrer i systemet.
- Administratorer skal kunne se alle kunder i systemet og deres ordrer, for at have et overblik over bageriets kunder og kunne følge op på deres ordrer.
- Kunden skal kunne fjerne en ordre fra deres indkøbskurv.
- Administratorer skal kunne fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer.

Aktivitetsdiagram

Nedenfor ses vores TO-BE-diagram. Det giver et overblik over, hvordan hele bestillingsprocessen vil finde sted, efter de har implementeret vores program. Det viser, hvordan en kunde ville vælge de ønskede cupcakes og herefter lægge dem i kurven. Når kunden så er tilfreds med bestillingen, tjekker vedkommende ud, hvorefter ordren gemmes i databasen. Herefter kan en ansat (hvis konto har rollen "admin") se ordren, og ud fra den, lave de ønskede cupcakes og derefter give dem til kunden. Hvis de laver cupcakesne og sælger dem som planlagt, kan den ansatte færdiggøre ordren inde i ordrelisten. Hvis der gik noget galt under udførelsen af ordren, kan den ansatte i stedet vælge at fjerne ordren fra listen.



EER-diagram



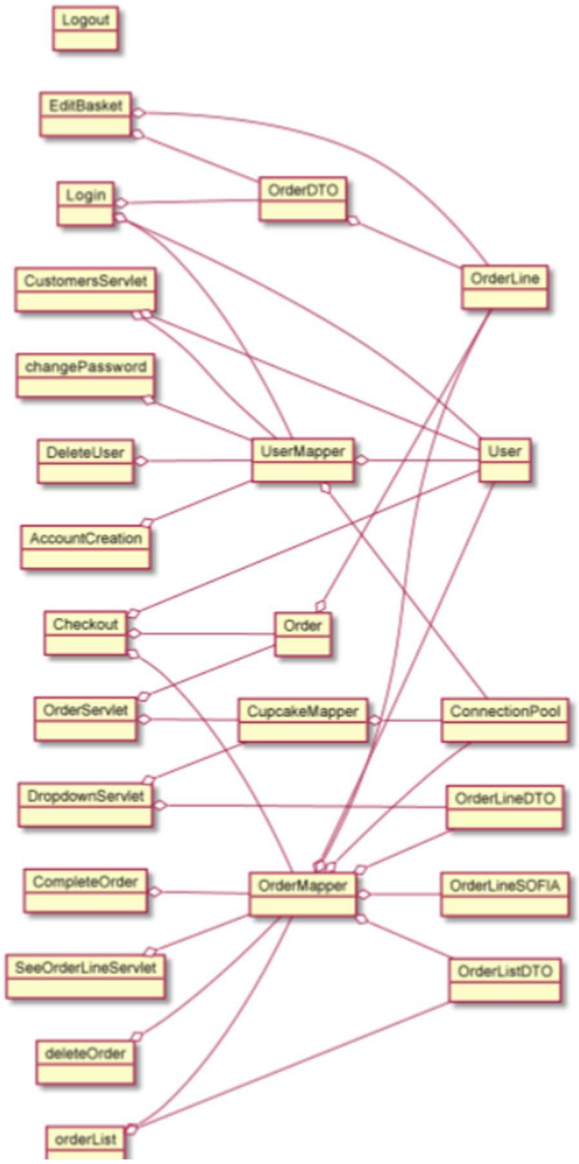
I vores EER-diagram repræsenterer "topping" og "bottom" de enkelte toppe og bunde som kunden kan vælge imellem til deres cupcake. "Orderline" er en enkelt ordrelinje, som består af den valgte top og bund, deres samlede pris, og antallet der bestilles. "Order" er en hel ordre, som kan have mange ordrelinjer. Ordren har også en samlet pris, et tidsstempel og en boolesk udtryk, der fortæller om bageren er færdig med den givne ordre. Til sidst er der "user", som skal repræsentere den enkelte bruger eller kunde på systemet. Brugeren har et brugernavn, kodeord, email, role (da man både kan være kunde og administrator i systemet) og en saldo, som de kan bruge på hjemmesiden. En kunde kan selvfølgelig have mange ordrer, hvis de ønsker det.

EER-diagrammet er opbygget således at en enkelt ordrelinje har en fremmednøgle fra den givne top og bund som kunden har valgt. Ordrelinjen har derudover også en fremmednøgle fra ordren, så det er til at finde ud af hvilken given ordre, den enkelte ordrelinje er en del af. Ordren har en fremmednøgle fra brugeren, så man kan knytte den enkelte ordre til den rigtige kunde.

Vores EER-diagram opfylder 3. normalform.

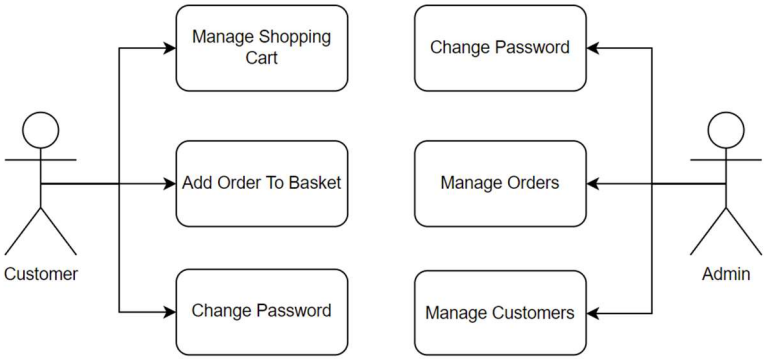
Domænemodel

Den lodrette række som ses til venstre i figuren, viser programmets servlets. Det ses at størstedelen af servletsne har en slags mapper, som for eksempel CupcakeMapperen og UserMapperen. Nogle af servletsne har en entitet, såsom orderDTO eller orderLine. Alle mapperne har en connectionpool, som står for at lave forbindelsen til databasen.



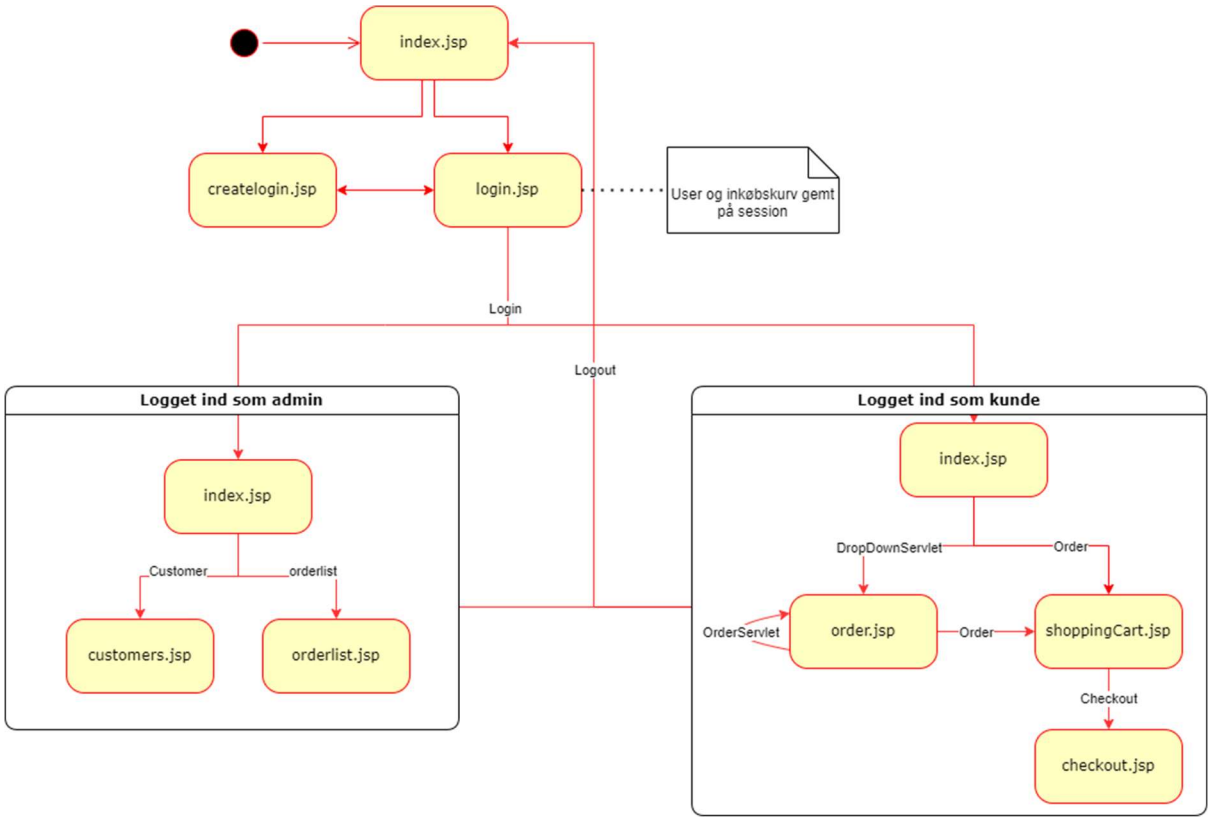
Usecase-diagram

Dette usecase-diagram viser, hvilke funktioner og muligheder man har, som enten ”admin” eller ”customer”. Diagrammet fokuserer udelukkende på de vigtigste funktioner, mens de lidt mere underordnede, ikke bliver vist her. For eksempel viser diagrammet ikke, at man som begge roller har mulighed for at se sin ”profile page”, hvorpå der står informationer omkring den konto, man er logget ind som.



Diagrammet viser, at man som ”customer” i grove træk, kun kan udføre handlinger, som fører vedkommende i retning af at bestille cupcakes, da det er formålet med hjemmesiden.
Som ”admin” handler funktionerne om at håndtere bestillinger og kunder, således at de kan danne sig et overblik over de ordrer, som skal udføres.

Navigationsdiagram



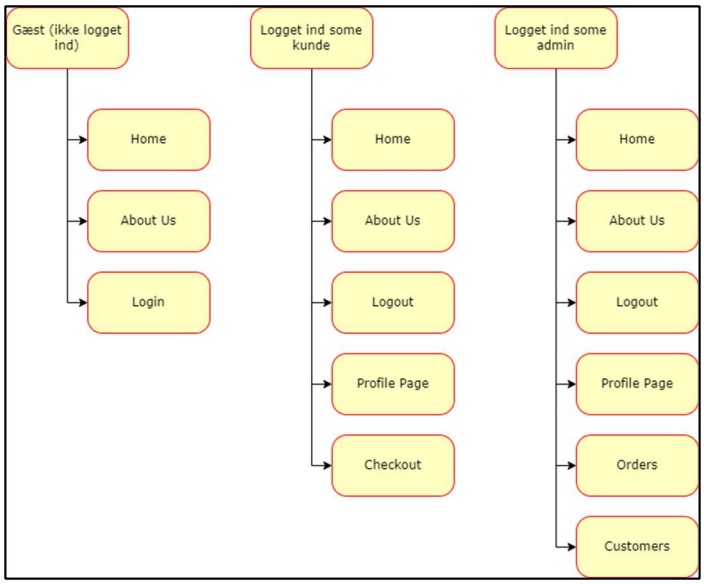
Navigationsdiagram: teksten på pilene er de forskellige servlets som redirecter siderne

Navigationsdiagrammet viser det generelle navigationsflow for hjemmesiden. Index-siden fungerer som forsiden. De eneste muligheder en guest har, er at lave en konto, se ”about us” og home. For at kunne få adgang til flere funktioner på hjemmesiden skal man logge ind, enten som admin eller customer.

De funktioner, der er specifikke for admin, er at se customers og orderlist. Orderlist viser en liste af alle ordrer og deres information, og hvor admin kan kontrollere ordrer. Customers-siden viser en liste af kunder, hvor de også kan blive slettet fra databasen.

Efter man har logget ind som kunde, har man muligheden for at komme på bestillingssiden. Når en kunde logger ind får de givet en indkøbskurv på deres session. Kunden tilføjer cupcakes ved at vælge en topping, en bottom og et antal. Efter en cupcake er tilføjet til indkøbskurven, bliver de ført tilbage til bestillingssiden, hvor de enten kan tilføje flere cupcakes eller gå videre til ”checkout”. Under ”checkout” vises der en liste over alle cupcakes, som er i sessions indkøbskurv. På denne side har kunden stadig muligheden for at gå tilbage, eller fortsætte med checkout og færdiggøre deres ordre. Efter kunden har trykket på checkout er deres ordre gemt i databasen og kurven på sessionen er nulstillet.

På hver side vises der en navigations bar som ligger øverst på skærmen. Den har forskellige states som varierer, afhængigt af om brugeren er logget ind og hvilken rolle de er givet. Alle brugere, som er logget ind, kan se deres e-mail, som er et link til deres profilside. Som gæst vil der være en loginknap i navigationsbaren, denne knap bliver erstattet med en logud knap, når man er logget ind.



Navigationsdiagram for navigationsbar

Særlige forhold

Når programmet er oppe at køre, bliver nogle informationer gemt i sessionen. Session gemmer den user man er logget ind som, samt den kurv som tilhører den specifikke user.

Når det kommer til validering af brugerinput, er vores program rimeligt veludviklet, da den kan tage imod størstedelen af input, altså både bogstaver, tal, diverse tegn osv. Den tager herefter inputtet, og tjekker databasen for users igennem, for at se om inputtet matcher en bruger i systemet. Hvis en bruger matcher det input programmet får, bliver man logget ind, mens man får en fejlbesked, hvis brugeren ikke findes.

Vedrørende login har vi valgt ikke at kryptere kodeord i databasen, da det ikke ligger inde for vores nuværende pensum, og vi derfor hellere ville bruge tid på noget, som gav bedre mening for vores læringskrav.

I databasen har alle brugere en rolle. Den kan enten være sat som ”admin” eller ”customer”. Rollen ”customer” repræsenterer en almindelig kunde og bliver sat til alle nye brugere, når de laver en konto via hjemmesidens ”Create Account” funktion. Når en bruger har rollen ”customer”, har de ikke adgang til andre brugeres informationer eller ordrer. En brugers hovedsagelige formål er at være i stand til at kunne lave bestillinger, og deres muligheder er derfor begrænset til præcis dette, med undtagelsen af også at kunne ændre kodeord.

Hvis en bruger har rollen ”admin”, bliver vedkommende anset som værende ansat i virksomheden. De har derfor mulighed for at se information om deres kunder, såvel som kundernes ordrer. En bruger med rollen ”admin” har også mulighed for at slette ordrer og selve kunderne i systemet.

Når man er logget ind som ”admin”, kan man, som tidligere nævnt, se en liste over ordrerne i systemet. Via denne liste har den ansatte mulighed for enten at erklære en ordre som færdiggjort eller slette den fra databasen. Hvis den bliver mærkeret som færdiggjort, bliver den sat ned til en ny liste nedenfor, kaldet ”Completed Orders”. Her har man mulighed for at gendanne ordren, hvis ønsket. Begge lister indeholder muligheden for at se ordrelinjerne i ordren, så virksomheden kan se, hvilke slags cupcakes der ønskes.

Status på implementation

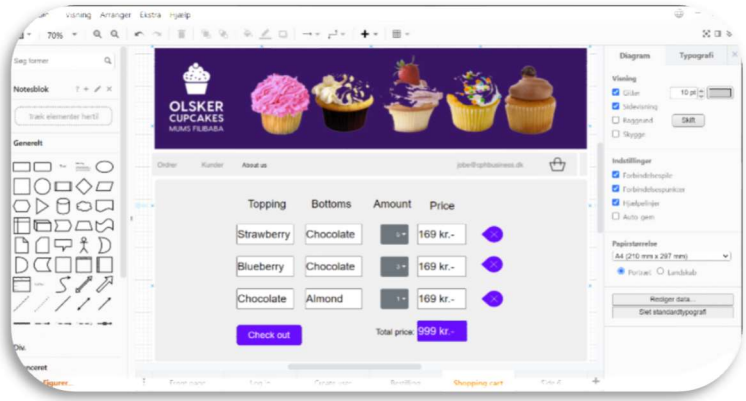
- Vores hjemmeside opfylder alle kundens krav, samt yderligere funktioner.
- De fleste af vores sider er nogenlunde præsentable, men kunne godt bruge noget mere styling/indhold, da nogle af siderne er lidt “tomme”.
- Når man er logget ind som kunde, mangler der et link til bestillingssiden i navigationsbaren. I den nuværende version af systemet, kan man kun komme ind på bestillingssiden via forsiden.
- Hvis en kunde genindlæser siden efter, at de har lavet en bestilling ved at trykke på “add to cart”, bliver formen sendt igen, dvs. den samme bestilling vil blive tilføjet til indkøbskurven igen.
- Der er manglende tjek i forhold til om man er logget ind som admin eller kunde, hvis man skriver navnet på en jsp-side/servlet direkte i url’en.
- Når man opretter en bruger på hjemmesiden, bliver man ikke herefter automatisk logget ind.

Proces

Plandannelse og udførelse

Vores planer gik ud på at vi først ville lave en mockup af vores hjemmeside, så vi kunne danne os et overblik over, hvad den skulle indeholde og være i stand til. Herefter ville vi oprette databasen, da den er en vigtig del af fundamentet til en velfungerende hjemmeside. Først efter databasen var sat op, ville vi begynde at lave JSP-sider og derefter back-end.

Vores planer blev udført som de var tiltænkt. Vi startede ud med at lave en mockup over hjemmesiden i fællesskab. Her blev vi enige om, hvordan de forskellige sider på hjemmesiden skulle se ud, og hvilke knapper der skulle være på hver side. Et eksempel på vores mockup ses til højre. Det var vores udkast til, hvordan vi havde forestillet os, at vores indkøbskurv skulle se ud.



Mockup af indkøbskurv

Som planlagt begyndte vi herefter at lave databasen, hvilket forløb uden problemer. Først da vi både havde lavet en mockup og en database, begyndte vi at lave JSP-sider. Vi havde planlagt først at lave alle JSP-siderne og derefter back-enden, men i praksis valgte vi at lave en JSP-side efterfulgt af dens tilsvarende back-end. På den måde kunne vi se, at de sider vi havde udarbejdet både så godt ud, men også rent faktisk virkede, inden vi gik videre til andre dele af hjemmesiden. Dette var lidt en afvigelse fra planen, men vi tænkte det gav bedre mening på den måde, og det var derfor intet problem at improvisere og forbedre vores fremgangsmåde.

Refleksion over processen og eventuelle forbedringer

Da vi begyndte gruppearbejdet, var CSS temmelig nyt for mange af os. Vi startede ud med kun at benytte én CSS-fil, som indeholdt alle specifikationerne, til alt på vores hjemmeside. Alt fra vores knapper til vores billeder blev formet og pyntet ud fra den ene CSS-fil. Der gik ikke længe, før vi begyndte at tabe overblikket og evnen til at separere andre medlemmers kode, fra ens egen. At deles om én CSS-fil gav også komplikationer, hver eneste gang vi skulle pushe eller pulle vores kode til og fra vores fælles repository. Forskellige gruppemedlemmer havde nemlig lavet tilføjelser og ændringer på de samme linjer, hvilket skulle løses, hver eneste gang der blev pullet eller pushet**.** Dette problem løste vi, ved at lave én CSS-fil per person. På den måde kunne man pushe og pulle, uden at skulle tage stilling til, om det ville påvirke andre individer i gruppen. Selv om dette var en klar forbedring, har vi besluttet os for, at vi næste gang laver én CSS-fil til hver afdeling af hjemmesiden, samt én CSS-fil til de elementer, som går igen flere forskellige steder (for eksempel layoutet til knappen, som findes på hver eneste side). På den måde passer én CSS-fil, til én af siderne på hjemmesiden. Dette ville både løse vores problem omkring at kunne pulle uden problemer, mens det samtidigt også ville gøre koden mere overskuelig og nemmere at finde rundt i.

Vi havde som tidligere nævnt, en idé om at lave en JSP-side, før vi lavede den tilsvarende back-end. Dette var en meget vellykket idé, som vi også vil tage gavn af næste gang. Det gjorde vores proces langt nemmere, da det gav os overblik over hvilke knapper, tekstfelter, links osv., som skulle kodes.
Ved næste projekt, vil vi dog gøre mere ud af at gruppere vores JSP-sider og servlets i mapper/directories, således de er mere organiseret, da det vil spare os tid i løbet af skriveperioden.

I løbet af denne periode benyttede vi os af Github Desktop, således vi nemt kunne dele koden med andre medlemmer af gruppen. Dette fungerede enormt godt, men vi vil dog nok næste gang gøre mere ud af at bruge hver vores branch, så vi kan bruge mindre tid på at løse konflikter ved push og pull.

Sidst, men ikke mindst, kunne nogle af vores klasser også have godt af en mere nøjagtig navngivning, for ikke at skabe forvirring for nye læsere af koden.