Olsker Cupcake

Deltagere

- Christian Kepp Stennicke - cph-cs519@cphbusiness.dk
- Metin Akbas - cph-ma733@cphbusiness.dk
- Sitthichai Phuanphimai Seest - cph-sp242@cphbusiness.dk
- Vivek Singh Nagra - cph-vn56@cphbusiness.dk

Copenhagen Business Academy

Datamatiker

2.Semester

Marts 2023

Indholdsfortegnelse

Indledning

Cupcake opgaven går ud på at udvikle en webapplikation for Olsker Cupcakes, der tillader kunder at bestille og betale for cupcakes med valgfrie bunde og toppe. Kunderne kan også oprette en konto for at gemme og betale deres ordrer. Adminstratoren har adgang til systemet for at se alle ordrer og kunder, og de kan redigere eller fjerne ordrer fra systemet og indsætte midler direkte på en kundes konto.

Baggrund

Olsker Cupcakes er virksomhed på Bornholm der sælger cupcakes.

Olsker Cupcakes vil gerne have at deres kunder skal have mulighed for at oprette sig som bruger for derefter at kunne bestille og betale cupcakes gennem webappilikationen. Som kunde skal man også kunne se sin valgte ordrer i en indkøbskurv så man kan se den samlede pris på ordren derudover skal man have muligheden for at fjerne sin ordre. Olsker Cupcakes vil også gerne have at der er en adminstrator rolle på webapplikationen så man kan logge ind som adminstrator. Adminstratoren skal have nogle specifikke muligheder i forhold til en kundekonto, Adminstratoren skal kunne indsætte midler direkte på en kundens konto, se alle kunder og deres ordrer i systemet og adminstratoren skal kunne fjerne ordre, så systemet ikke bliver fyldt med ubetalte ordre.

Teknologikrav

Webapplikationen udvikles ved hjælp af IntelliJ IDEA 2021.2.4, Java , Mysql Workbench 8.0 CE, jdbc, HTML, CSS og Twitter BootStrap og skal køre på en Tomcat 9.0.73 webcontainer.

Java: Bruges til at lave klasser, som indeholder metoder og attributter.

Myqsl Workbench: Vores MySQL workbench bruges til at oprette og redigerer i vores database. Via reverse engineer, arbejder vi med et EER-Diagram(Enhanced Entity Relationship), som skal sikre en god database integritet, og sammenhæng mellem vores forskellige entiter og klasser.

Jdbc: Vores Java Data Base Connecter bruger vi til at oprette forbindelse til vores MySQL database, så vi kan indsætte samt hente informationer ud af databasen direkte i intelliJ.

HTML: Bruger vi til at bygge fundamentet af webapplikationen.

CSS: Bruger vi til at style vores webapplikation

Twitter BootStrap: Bruger vi til at gøre designet brugervenligt og responsivt.

Tomcat: kreerer en webserver til webapplikationen

Krav

Olsker Cupcakes håb med systemet er at gøre dem kyndige online, så de både kan sælge gennem deres fysiske butik og online. Vores system ville kunne tilføre deres forretning mere effektivitet, i forhold til at kunderne nu kan bestille og betale online. Det vil gøre så kunderne blot skal svinge forbi og afhente deres cupcakes, som vil gøre at kunderne ikke skal stå i en eventuel lang kø. Butikken får også muligheden for at få et bedre indblik i deres ordre historik og dagens ordre.

User stories

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 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 ordrelinier i en indkøbskurv, så jeg kan se den samlede pris.

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

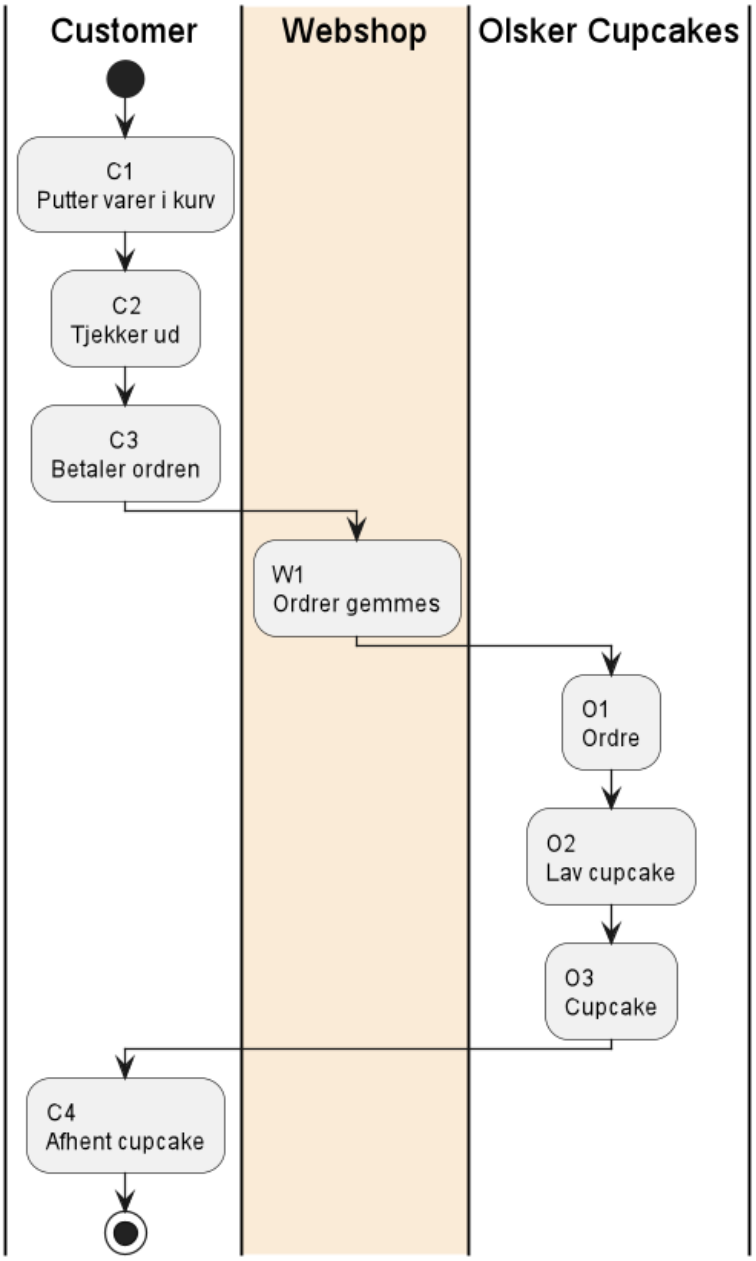
US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

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 udgyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Aktivitetsdiagram



Aktivitetsdiagrammet viser processen for en kunde ,der bestiller og afhenter en cupcake hos Olsker Cupcakes.

Vi har tre swimminglanes, som er repræsenteret af Customer, Webshop og Olsker Cupcakes.

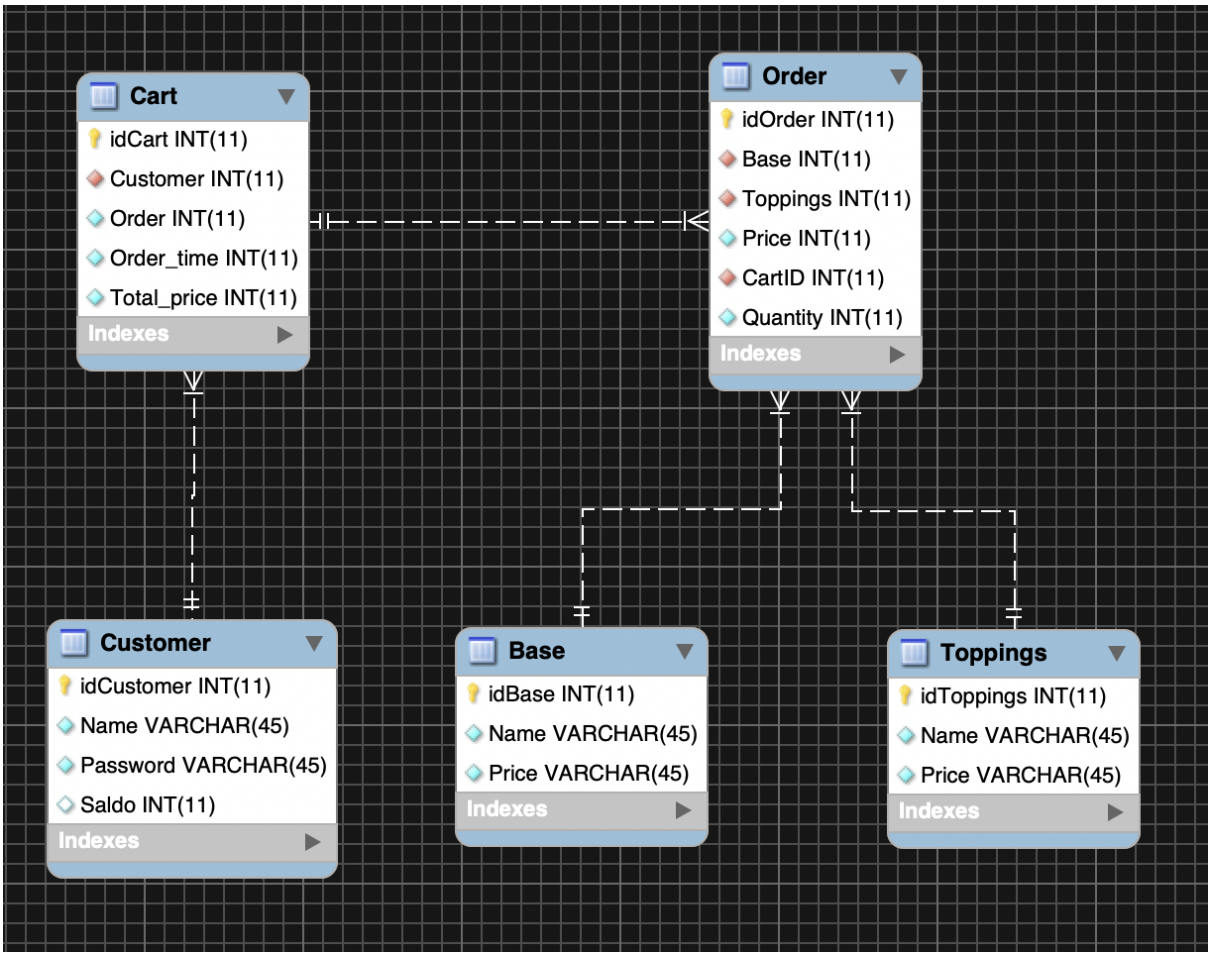
Processen begynder med kunden ved at putte varer i deres kurv, markeret som C1. Herefter går kunden videre til at tjekke ud af kurven C2. Efter kunden har tjekket ud, betaler de ordren i C3.

Webshoppen gemmer ordren (W1), når kunden har betalt.

Efter betalingen er godkendt, fortsætter processen til Olsker Cupcakes, som har 3 aktivteter: O1, hvor der bliver lavet en ordre, O2, hvor en cupcake bliver lavet, og O3, hvor cupcaken bliver klar til afhentning.

Kunden afslutter prrocessen ved at afhente sin ordre (C4).

EER diagram(Enhanced Entity Relationship)



Vores EER-Diagram består af 5 entiteter - **Customer, Base, Toppings, Cart** og **Order.**

Customer: Denne enhed repræsenterer en kunde, der kan placere ordrer. Kundeentiteten indeholder oplysninger som navn, password og har et ID som er auto-genereret og er unik til den kunde.

Base: Base repræsenterer bunden for en cupcake. Der kan være forskellige typer af kagebunde, f.eks. chokolade eller vanilje, og hver type har deres egen pris.

Toppings: Toppings repræsenterer forskellige toppings, som en kunde kan vælge til deres cupcake. Hver topping har også deres egen pris.

Cart: Cart repræsenterer kundens kurv, som kan indeholde en eller flere cupcakes. Kunden kan tilføje eller fjerne varer fra kurven, inden de fuldfører deres ordre.

Order: Order repræsenterer en ordre, der er placeret af en kunde. Vi har valgt at se en ordre som én cupcake som ligges i kurven.

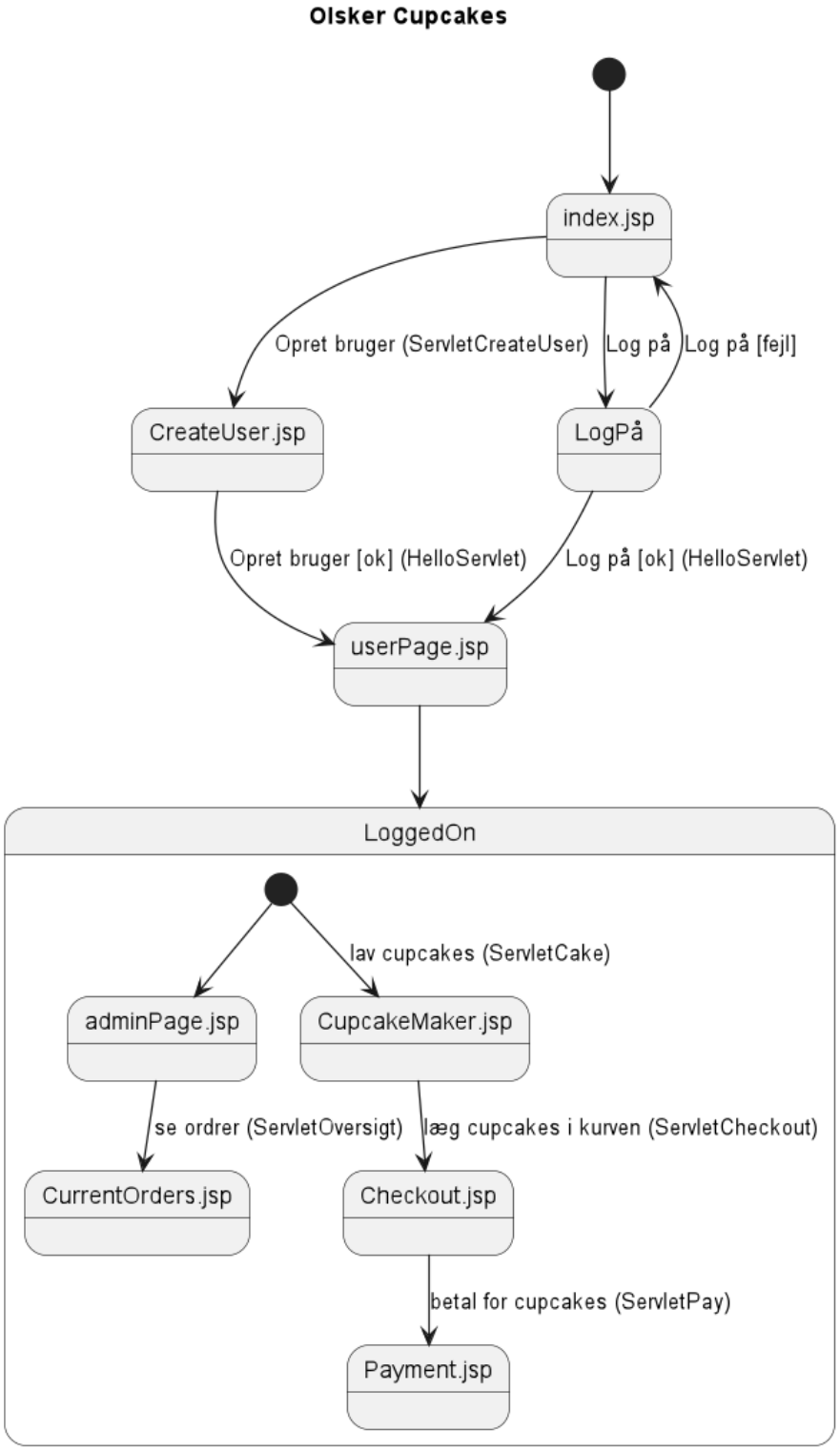
Entiternes relationer

En kurv kan have mange ordre, som er en sammensætning af topping og bund(en hel cupcake). Derfor har Cart og Order en *én til mange* relation.

En kunde kan have mange kurve. F.eks. når en kunde bestiller noget, og derefter bestiller noget igen efter noget tid, så får de en ny tom kurv. Derfor har Customer og Cart en *én til mange* relation.

En base kan bruges til mange forskellige ordre, og det samme gælder for toppings. Derfor har de en *én til mange* relation med Order

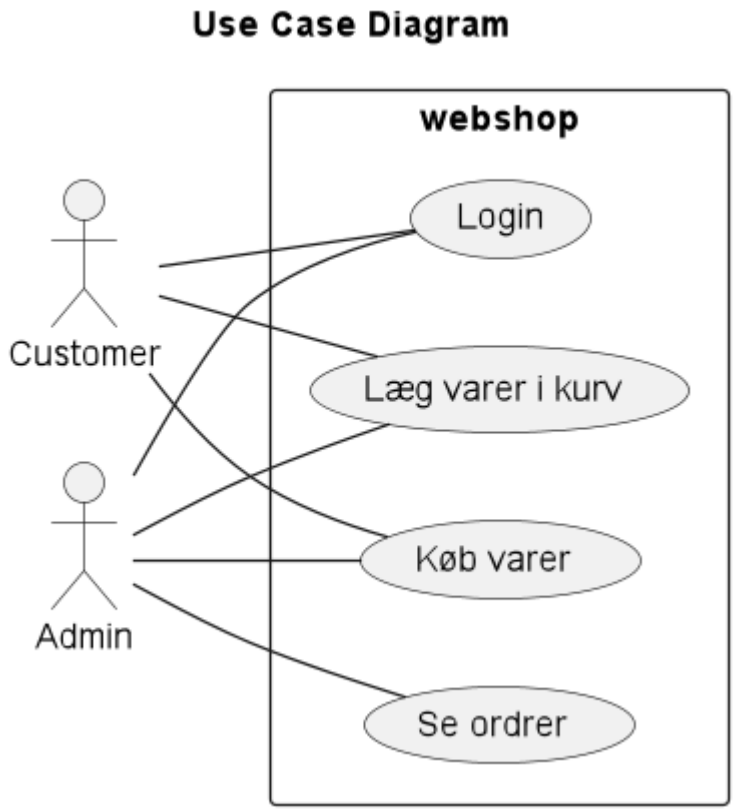
Navigationsdiagram



Navigationsdiagrammet viser et overblik over hvordan systemet er bygget op, og hvordan brugeren navigere mellem de forskellige sider.

Applikationen starter ved index.jsp, hvor brugeren kan se en login-formular. Når brugeren indtaster deres login-oplysninger og trykker på knappen "Log på", vil applikationen validere disse oplysninger. Hvis logindetaljerne er korrekte, vil brugeren blive omdirigeret til brugerens side (userPage.jsp), og hvis logindetaljerne er forkerte, vil brugeren blive omdirigeret tilbage til index.jsp med en fejlbesked. På brugerens side (LoggedOn) har brugeren flere muligheder: Hvis brugeren har administratorrettigheder, kan de også se en oversigt over alle ordrer på adminPage.jsp. oprette nye cupcakes på CupcakeMaker.jsp, lægge cupcakes i kurven på Checkout.jsp, og betale for deres ordrer på Payment.jsp. Brugeren kan også logge ud ved at trykke på "Log af" -knappen, hvilket vil tage dem tilbage til index.jsp.

Use case diagram



Use case diagrammet Diagrammet repræsenterer en webshop, hvor der er to aktører: "Kunde" og "Admin".
Kunden har tre Use Cases: "Login", "Læg varer i kurv" og "Køb varer". Kunden kan logge ind i webshoppen, tilføje varer til deres indkøbskurv og købe de valgte varer. Admin har også tre Use Cases: "Login", "Læg varer i kurv" og "Køb varer", men derudover har Admin også adgang til en ekstra Use Case: "Se ordrer". Admin kan også logge ind i webshoppen, tilføje varer til deres indkøbskurv og købe varer ligesom kunden, men de kan også se ordrer, som er blevet placeret i webshoppen. Webshoppen er repræsenteret som en rektangel i midten af diagrammet og indeholder alle de Use Cases, som både Kunden og Admin har adgang til.

Særlige forhold

Fra index siden bliver brugernavn og kode gemt og sat til brugernes input, disse bliver så valideret med værdier i vores database, hvis valideringen godkender brugerens input, får de lov til at komme videre ind til det vi kalder userPage siden.

Ved brugerns userPage gemmer vi det inputtet til deres brugernavn og sætter det på “session scopet”, det vil sige at inputtet med brugerens navn gemmes indtil deres Session udløber, enten ved inaktivitet eller ved at bruger selv afslutter ved at klikke log af eller lukker deres browser ned. Ved at gemme deres “brugernavn” kan vi igennem hver ny .jsp side vise og tilgå brugerens navn, og derved opfylde user story kravet der ber om at brugerens navn kan ses på hver side de besøger. Vi vælger, at udnytte session scopet fremfor et request scope, da det ved et request scope kun vil være tilgængeligt under request sammenhænget og derefter vil blive slettet fra hukommelsen. Yderligere er det brugerens valg af top og bund til deres cupcake, som igen gemmes ved “session scopet”.

Under brugerens rejse kan de støde på forskellige “exceptions”, hvis deres input under login ikke kan valideres støder de på et block som sender dem tilbage til vores indexside(velkomst siden), på den måde afbryder vi brugerens oplevelse, ved at de har muligheden for at prøve og logge ind igen. Det er også på den måde vi håndterer “fejl” ved brugerens cupcake valg.

Vi validerer brugerens input ved, sammenligne det med værdierne i vores database, hvis inputtet ikke “matcher” med nogen værdi fra databasen returnerers det som en “null” værdi, hvorefter der på vores servlet side så kontrolleres modtagelsen fra database siden. Hvis inputet har en værdi der matcher med brugerens input og dermed ikke er “null”, så er processen godkendt/valideret.

Sikkerheden i vores program forekommer kun ved valideringen af brugerns “brugernavn” og “kode”, uden denne validering kan programmet ikke komme videre ind til de reserende sider. Vi er i gruppen dog enige i, at der stadig vil være mulighed for, at komme rundt om dette ved f.eks. et “SQL injection” angreb, hvor der kan indsættes “falsk” data.

Status på implementation

Vi tog valget om at starte fra “Scratch”, og ikke ud fra startkoden som var gjort tilgængeligt. Dette resulterede i at vi stødte på et par udfordringer under de første dage af forløbet, som satte os tilbage ifht. arbejdstid.

Vi har delvist nået at implementere 6 ud af 9 User Stories i vores webapplikation. Man kan logge ind, samt oprette sig som bruger. Når man er kommet igennem, kan man som administrator tilgå en side hvor man kan se alle ordrer der er lavet i systemet. Som bruger kan man bestille en cupcake ved at vælge base, topping samt antallet af denne sammensætning af cupcake som man ønsker. Derefter bliver man videresendt til en kurv-side og betalingsside. Vi har heller ikke nået at implementere det således at man kan tilføje forskellige typer sammensætninger af cupcakes til sin kurv. Det er således kun muligt at tilføje én sammensætning, men der kan tilføjes flere antal af denne til kurven.

Vi har ikke nået at implementere yderligere end dette. Derudover mangler vi at styke vores sidste to JSP sider som er Kurv-siden samt betalingssiden.

Ifht. vores database kan vi indsætte en ny bruger i vores “CUSTOMER” tabel, når der oprettes en ny bruger på hjemmesiden. En ny bruger oprettes med et navn samt et password, hvorefter der auto-genereres et ID til brugeren. Vi kan tilgå vores database og hive brugeren ud når der logges ind på siden. Vi har ikke nået at få implementeret det således at ordrene der laves, indsættes i databasen. Derfor indeholder admin side heller ikke nogle ordre.

Process

Teamets fremgang til projektet har været varierende, vi uddelegerede opgaverne imellem os. I vores team har vi meget forskellige kompetencer, nogle er mere skarpe til blandt andet HTML, MySQL, JSP, Java, samt integrationen mellem Java og MySQL. Undervejs i forløbet samlede vi op og gennemgik de forskellige barrierer der forekom. Arbejdsmetoden resulterede selvfølgelig i at vi lavede forskellige ting i løbet af projektet, samt at alle ikke var 100% inde over de forskellige koder og metoder der blev implementeret.

Vi kom hurtigt igennem de basale ting, og fik stablet en køreklar prototype op. Vi havde en nogenlunde god ide om hvordan systemet skulle opbygges og integreres i vores løsning. Men undervejs dukkede mange komplikationer op, og vi brugte lang tid på at løse dem. Det kunne gøres bedre, hvis vi havde lavet en mere detaljeret oversigt over vores implementering af funktioner samt kommunikationen mellem de forskellige elementer og tabeller i MySQL databasen. Vi har lært en hel masse under forløbet, blandt andet integration af Java-kode i HTML og MySQL. Samt fået en ide om hvordan man stabler en lille webshop op, hvor der er mulighed for at customize produktet.