CUPCAKE VIRKSOMHED FRA BOLHOLM

Implementeringen af en hjemmeside

-

til dette formål

Udarbejdet fra: d. 04/04/2022

til og med 20/04/2022

Claes Christian Heise - Hold A

cph-ch568@cphbusiness.dk

Github: ClaesHeise

Indhold

Indledning

Til dette projekt, har det været til formål at udvikle og implementerer en hjemmeside, for en cupcake forretning lokaliseret på Bornholm. Både de visuelle og funktionelle implementeringer, har derved skulle reflektere dette.

I denne rapport gennemgås de forberedende processer, værktøjer, såvel som aktuelle processer der blev gjort brugt af, i udviklingen af dette. Ligeledes bliver det også beskrevet, hvordan de forskellige implementeringer skal bruges, er tænkt ind og hvilke mangler, der endte ud med at være efter processen.

Link til youtube video - gennemgang af program:
https://www.youtube.com/watch?v=YX0mkwztkSk

Baggrund

Virksomheden som dette produkt er udviklet til, er kaldet Olsker Cupcakes. Virksomheden er lokaliseret på Bornholm og vil gerne have implementeret en hjemmeside.

På denne hjemmeside skal kunderne blandt andet kunne sammensætte - i form af selv at vælge bund og top, på de(n) givne kage -, bestille og til sidst kunne betale for Olsker Cupcakes - kager.

Ligeledes skal ledelsen af forretningen have adgang til funktioner kun de har adgang til, via administrations rettigheder. Disse funktionaliteter skal eksempelvis gøre dem i stand til, nemt at organiserer deres bestillinger og kunder, såvel som at ændre i disse data, herunder den enkeltes kundes beløb på kontoen, fjerne ordre m.m.

Teknologi valg

Til udformningen af dette projekt er der blevet brugt diverse både organisatoriske- og implementerings programmer. Disse programmer og deres formål, er som følger:

Draw-io (browser version):

Denne software er blevet brugt til at udarbejde et *Mockup* af hjemmesiden. På denne *Mockup* planlægges og implementeres hvordan hjemmesiden skal præsenteres visuelt, såvel som hvilke og hvor mange sider, hjemmesiden skal bestå af, for at løse de ønskede funktionaliteter som virksomheden har opstillet. Denne *Mockup* er som nævnt en del af planlægningsprocessen, så den repræsenterer ikke det færdige produkt, men kan være et godt værktøj til, også i løbet af arbejdsprocessen, at referere tilbage til, når man som udvikler er i tvivl om hvordan de enkelte ting skulle implementeres.

MySQL WorkBench og MySQL version 8.0.27:

Til at gemme og hente data, såsom brugere og kagerne, til brug på hjemmesiden er der gjort brug af databaseværktøjet MySQL. Til oprettelse af de nødvendige tabeller og koloner, hvori dataene er placeret, er der blevet gjort brug af MySQL WorkBench. MySQL WorkBench er ligeledes blevet brugt til at udarbejde EER diagrammet over den implementerede database, som vises senere i rapporten.

Intellij IDEA 2021.3.3:

Som editor til at skrive både front- og backend koden, er der blevet gjort brug af Intellij. Herunder er der blevet brugt en smule CSS og Bootstrap 5.1.3 til at implementerer det visuelle/frontend delen af hjemmesiden. Der er brugt Java klasser og Servlets til at implementerer de funktionelle/backend aspekterne af hjemmesiden. Til sidst er disse kombineret ved brug af .jsp filer, skrevet ved brug af HTML, til at interagerer med brugeren.

Tomcat 9.0.60:

I løbet af udviklingsprocessen, for konkret at se hvordan de forskellige implementeringer, både front- og backend, så ud og om hvorvidt de virkede som ønsket, blev der gjort brug af Tomcat igennem Intellij, som Plugin. Tomcat oprettede en lokal server, når man igangsatte det, således at udvikleren kunne se hvordan hjemmesiden så ud og virkede, på det givne tidspunkt.

Krav:

De krav og funktioner der blev opstillet af virksomheden, er blevet beskrevet i 9 forskellige user-cases, de lyder således:

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.

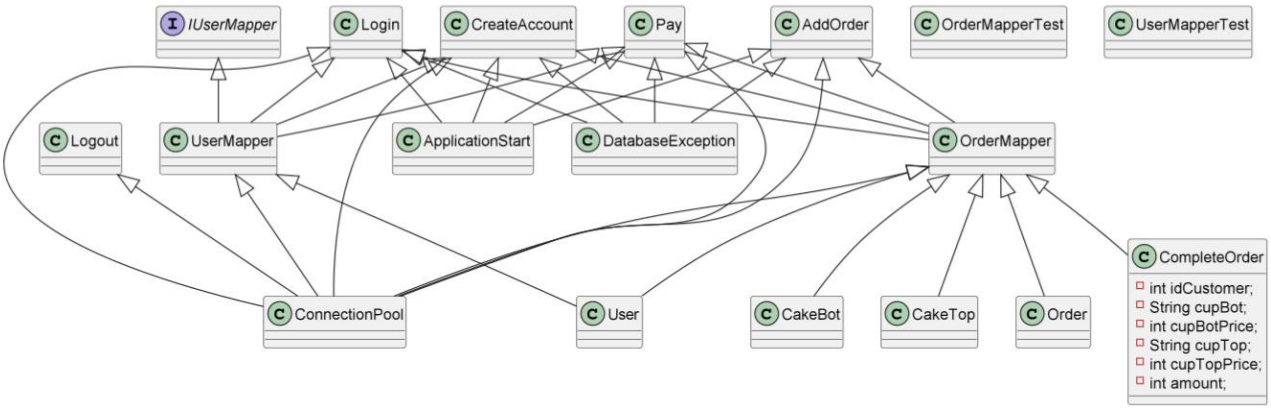
US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

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

Domænemodel og ER-diagram

Domænemodelen, dog lidt rodet, ser således ud:



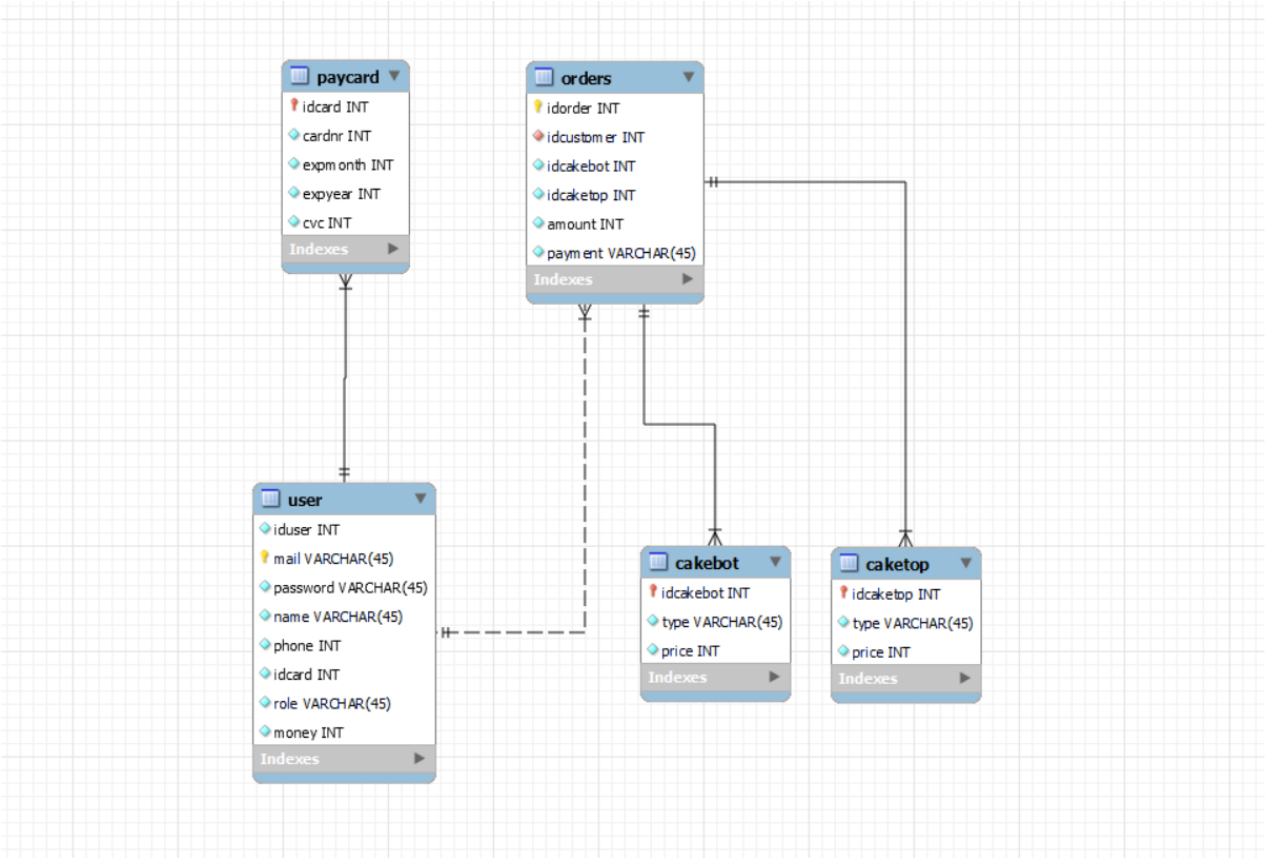
Som lidt bagvedlæggende forklaring, på denne mindre rodede repræsentation, så kan klasserne ses således, der er 3 lag.

1. Lag: På det nederste lag, er der *entities* altså de objekter som bliver hentet fra Databasen. De fleste af disse, værende klasserne: *User*, *CakeBot*, *CakeTop* og *Order* har alle de samme attributter, som tabellerne i databasen ligeledes har. Den eneste der skiller sig lidt ud, og derved også årsagen, til at dens attributter vises herpå Domænemodellen, er *CompleteOrder*. Dette objekt er en sammensætning af *Order*, *CakeTop* og *CakeTop*, således at man får et objekt der, i stedet for at indeholde en masse id'er, har de skriftlige informationer omkring ordren, som derved kan præsenteres til brugeren.

2. Lag: På det mellemste lag, er der *Mapperne* hvilket er de klasser der indeholder metoder til at hente, uploade og ændre data i databasen, og præsentere det for brugeren ved at omdanne dataene fra databasen, til objekterne fra 1./nederste lag. På dette 2./mellemste lag, ligger også klasser såsom *ApplicationStart* og *DatabaseException*, da disse, ligeledes som *Mapper* klasserne, går op til *Servlet* klasserne som står beskrevet nedenunder ved 3. Lag.

3. Lag: Dette øverste og sidste lag, er som kort beskrevet alle *Servlet* klasserne, hvilket vil sige de klasser der behandler alle de funktioner og data der ligger på backend, og giver det videre til HTML dokumenterne, som dernæst kan præsentere det for brugeren, via hjemmesiden.

Dernæst er der Er-diagrammet, som ser således ud:



Her vises de forskellige tabeller og deres koloner, med de forskellige data de lagre. Her kan ses, som også nævnt ovenover, at de er næsten ens med klasserne ved 1. Lag på overstående domænemodel.

Hvad der ellers kan beskrives ved disse, er at de er delt ud i flere forskellige tabeller, med id'er der referer til hinanden, for at normalisere databasen, så vidt gav mening. Ved brug af *Mapperne* bliver dataene fra de forskellige tabeller dernæst sammensat, til hvad end der giver mening at præsenterer for brugeren.

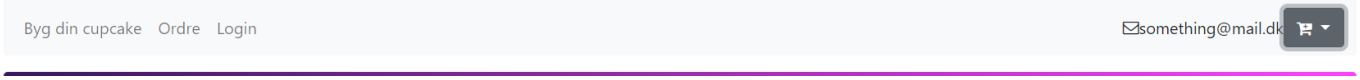
Navigationsdiagram

Der er desværre ikke helt lavet et ordentligt navigationsdiagram her, af tidsrelaterede årsager, men der indsættes billeder af navigationsbaren for *Før login*, *Efter kunde login* og *Efter admin login*, og beskrives herfra, hvilke .jsp sider og servlets, der fører brugeren hen til de forskellige.

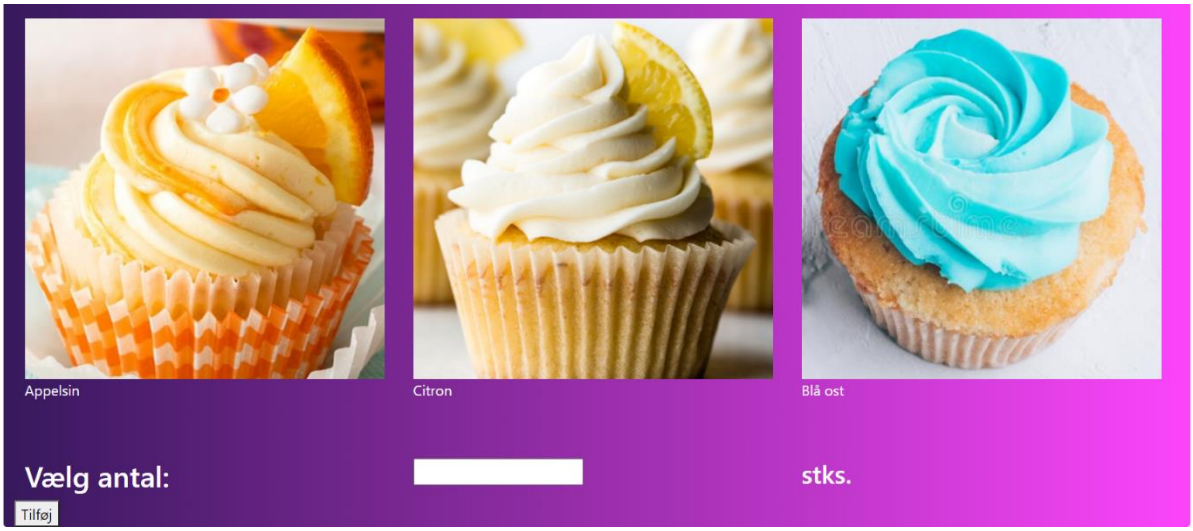
For at se de forskellige sider visuelt, refereres der her til cupcakeMockUp.drawio fil der ligger i samme mappe som denne rapport, eller ved selvfølgelig at køre programmet.

Før login:

Når brugeren først kommer ind på hjemmesiden, vil navigationsbaren se således ud:



Byg din cupcake: De vil befinde sig på index/startsiden (index.jsp), hvoraf de kan bygge deres egen cupcake, ved at trykke på billederne for den bund og top de ønsker (radioknapper) og angive et antal. Dernæst kan de lægge den i kurven, ved at trykke på den (nuværende) lille og lidt malplacerede "*Tilføj*" knap, som også ser herunder:



Når "*Tilføj*" knappen trykkes, sendes dataene som brugeren har angivet, via en form, videre til AddOrder.java klassen/servletten, som tilføjer ordren til en bruger (i tilfælde af ikke at være logget ind, oprettes en anonym bruger - dog hvis der logges ind efter at en ordre er tilføjet, vil ordren automatisk blive tilføjet, til den bruger der er logget ind). Når denne er tilføjet, sender AddOrder.java brugeren tilbage til index.jsp (startsiden).

Ved på et vilkårligt tidspunkt at trykke på "*Byg din cupcake*" kommer brugeren tilbage til index siden.

Ordre: Når "*Ordre*" knappen klikkes, føres brugeren videre ind til Ordre siden (orders.jsp). Her kan brugeren se alle sine ordre, hvis nogle er lagt i kurv, ellers vil der bare stå "Ingen varer er lagt i kurv". Det skal også påpeges, hvis brugeren har ordrer fra tidligere besøg på hjemmesiden, vil disse også blive vist under ordrer, såvel som nyligt tilføjede. Om end dette er en god applikation til

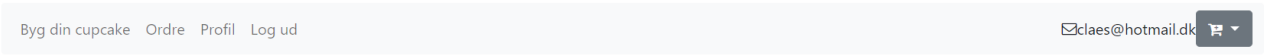
produktet eller ikke, er noget der ville kunne bedømmes hvis der skulle arbejdes videre med produktet.

Nederst på ”*Ordre*” siden, kan der ligeledes findes en, mindre malplaceret, ”*Betal*” knap, som kan ses herunder:

Denne knap sender brugeren videre til Pay.java klassen/servletten som beregner hvorvidt brugeren har nok penge på deres konto (fra databasen), til at betale for den nuværende ordre. Her skal det siges, at som programmet er lige nu, skal brugeren være logget ind, for at kunne gennemfører denne betaling, da administrationen (virksomheden) ikke har kunnet sætte penge ind på en anonym/midlertidig bruger. Når hjemmesiden har været igennem Pay.java sendes brugeren videre til betalt.jsp siden. Hvis købet har været succesfuldt (bruger havde nok penge på konto), får brugeren her en bekræftelse på dette, og hvis ikke det var succesfuldt, får brugeren dette at vide. Lige meget om det var en succesfuldt eller ej, er der en knap nederst på siden som sender brugeren tilbage til index.jsp.

Login: Knappen ”*Login*” sender brugeren videre til login.jsp. På denne side kan brugeren både logge ind på en eksisterende bruger, ved brug af deres E-mail og Password, og dernæst trykke på knappen ”*Log på*”. Ved tryk på ”*Log på*” sendes brugeren videre til Login.java servletten, her ses der på, om hvorvidt brugeren har angivet en korrekt E-mail og Password. Hvis dette er tilfældet, vil brugeren enten blive logget ind som ”*admin*” eller ”*user*” (kunde), hvoraf navigationsbaren tilpasser sig til den nuværende bruger, hvilket bliver beskrevet snarligt. Ligeledes vil relevante ordrer, komplette ordre, kunder (hvis admin) og flere objekter, blive hentet og tilføjet til et session scope.

Efter kunde login

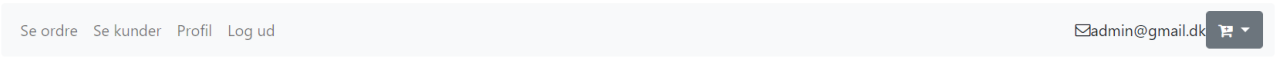


De fleste punkter/knapper som kan ses ved denne navigationsbar, når en bruger er logget ind, af typen kunde (user), så både ”*Byg din cupcake*” og ”*Ordre*” og de tilhørende knapper herunder, vil opføre sig på samme måde, med undtagelse af at brugeren nu kan betale for deres ordre.

De nye knapper her, vil være ”*Profil*”, denne er dog ikke implementeret endnu, men var tiltænkt som en side hvor brugeren kunne se og ændre (via form) i sine informationer.

Den sidste knap er ”*Log ud*” denne knap fører brugeren hen til Logout.java som, ved brug af session.invalidate(), tømmer session scope for objekter, hvilket ligeledes vil sige bruger objektet, og brugeren vil derved være logget ud, dernæst sendes brugeren tilbage til index.jsp.

Efter admin login



Efter en bruger er logget ind som typen admin, vil navigationsbaren se ud som overstående. ”*Profil*” og ”*Log ud*” er det samme som ved kunde login. Men der er ikke længere en ”*Byg din cupcake*” eller ”*Ordre*” disse er i stedet erstattet med ”*Se ordre*” og ”*Se kunder*”. Da siderne bag disse knapper er meget ens, kan de forklares samlet. Ved ”*Se ordre*” kommer brugeren ind på adminOrders.jsp og ”*Se kunder*” er adminCustomers.jsp. På begge sider fremgår der en tabel med forskellige data om henholdsvis *ordre* og *kunder*, således at administrator kan følge med i hvilke ordre der er, til hvilke kunder og alt andet eventuelt brugbart for virksomheden.

Særlig forhold

Under særlige forhold, uddybes der på de forskellige sider, der indeholder lidt mere backend end visse andre. De sider der ikke har så meget backend eller ikke er påbegyndt, medtages derved ikke, da beskrivelsen fra overstående kapitel, burde være tilfredsstillende nok på nuværende tidspunkt.

Nu da de forskellige sider og dels også servlets er beskrevet ovenover, skrives disse op ved .jsp navn og uddybes lidt yderligere på, herunder:

index.jsp: Som nævnt, ved tryk på knappen ”*Tilføj*” kaldes servletten AddOrder.java. Denne laver et Order objekt, via kundens id (hvis de er logget ind, ellers bliver kunde id = 0 - altså anonym/midlertidigt login), et id for både bund og top af cupcaken, og til sidst mængden. Alt dette tages fra den <form> som ligger på index.jsp siden, hvilket vil sige bruger inputtet. Med dette Order objekt, og informationen herfra, bliver der ligeledes også skabt et CompleteOrder objekt, som ligeledes indeholder navnet på både bund og top af kagen, såvel som den samlede pris for hele kagen og antallet af disse. Begge disse objekter bliver tilføjet til et session scope, så de kan bruges på de andre sider.

orders.jsp: Under denne side, er der ikke noget data fra bruger, men når brugeren trykker på ”*Betal*” knappen, køres Pay.java servletten. Denne opretter ikke nogen objekter, men hvis købet var succesfuldt, vil den opdatere den variabel under kundens Order objekter, der hedder ”*Paid*” som beskriver om hvorvidt ordren er betalt eller ikke, den opdaterer derved også de pågældende ordrer i databasen, og så opdaterer den Order og CompleteOrder objekterne i session scope.

login.jsp: Under denne side kan brugeren enten logge ind eller lave en ny bruger. Ved at logge ind, kaldes Login.java servletten, denne er allerede nogenlunde beskrevet. Hvis det var et succesfuldt log in, så vil den hente den pågældende ”*User*” fra databasen og inkludere det i et session scope, ligeledes tages denne brugers Order og CompleteOrder, og placeres i session scope. Hvis en det er en admin der logger ind, vil den hente alle Order og User objekter ned fra databasen, og placerer dem i session scopes.

Hvis i stedet brugeren opretter en konto, vil de skulle udfylde en form med alle de nødvendige informationer og dernæst vil disse sendes videre til CreateAccount.java servletten, som opretter et User objekt, både i databasen, såvel som programmet, som den så også tilføjer til session scope. Ligeledes, hvis brugeren havde tilføjet nogle ordrer til kurven, inden oprettelse af konto, vil disse automatisk blive tilføjet til denne nye brugers id, ligeledes i databasen og session scopet.

adminOrders.jsp & adminCustomers.jsp: Når en admin logger ind, via login.jsp siden, beskrives der ovenover, at alle User- (af typen kunder/”*user*”) og Order objekter bliver tilføjet til hver deres session scope, derved gør adminOrders.jsp og adminCustomers.jsp siderne, brug af disse objekter, til at oprette tabeller med disse.

Status på implementation

Nu da de ting som er blevet implementeret ved programmet, er gennemgået, vil dette kapitel omhandle de punkter som mangler, eller kunne implementeres bedre. Som tidligere nævnt, har dette projekt været dels under tidspres, af personlige årsager og der er en dels mangler der kan nævnes, og ligeledes formodentligt også nogle, der er blevet overset i processen. Her kommer dog de, som er fundet og gennemtænkt.

Mangel på exceptions/errors: Som programmet ser ud lige nu, er der kun 1 error.jsp side, og fejl beskeder er ikke tilpasset de forskellige fejl, som en bruger kan komme ud for. For at løse dette, ville der som minimum skulle bruges noget tid, til at tilpasse teksten af de fejlbeskeder brugeren kan komme ud for, for at give dem et klart billede af hvad der er gået galt. Hvis dette tages et step yderligere, kunne der også laves mere individuelle .jsp sider, når en fejl opstår. Et eksempel på dette kunne være lidt ligesom betalt.jsp der indeholder en *if-statement*, der sørger for at, hvis betalingen ikke gik igennem, får brugeren en bestemt fejlbesked der fortæller dette og en knap tilbage til forsiden.

Mangel på restriktioner: Dette punkt kommer lidt i forlængelse af forrige, men er mere fokuseret på, hvis brugeren skriver for få eller for mange tal til, eksempelvis deres kort nummer, cvc osv. Altså at begrænse hvad brugeren kan angive af data, eksempelvis også for at undgå at der bliver oprettet 2 brugere med samme E-mail eller lignende.

Manglende visuelle detaljer: Dette er et mere generelt punkt for det visuelle, frontend delen. Hjemmesiden er, i store penselstrøg, blevet visualiseret som det var tiltænkt (af producent), men mange af finesser mangler. Dette indebærer både malplacerede knapper, besynderlige skriftstørrelser, positioner af billeder, objekter der skal være centrerede og listen fortsætter. Givet mere tid, kunne disse forholdsvist nemt udbedres, men mere påtrængende punkter har taget prioritering for nu.

Manglende sider: Som det mest konkrete eksempel mangles der, at implementerer profil siden. Men som også allerede lidt hentydet til, kunne der muligvis også være plads til flere fejl sider. Yderligere, hvis der blev arbejdet videre med både de designmæssige, såvel som funktionelle aspekter af dette produkt, ville der formodentligt også findes flere relevante sider der kunne laves.

Manglende tests: De første par implementerede klasser, værende UserMapper og OrderMapper, har fået lavet medfølgende tests, men de resterende klasser, hvoraf tests ville være relevante, mangles.

Manglende user-stories: user-stories nummer 8 og 9, som begge omhandler at kunne fjerne en ordre, fra henholdsvis en kundes kurv, eller fra hele systemet, hvis det var en admin. Disse som de eneste er ikke blevet implementeret. Måden hvorpå de ville være tænkt implementeret, hvis der havde været tid, ville være ved brug af en ”*slet*” knap ud fra de(n) pågældende varer. Mere konkret, både når en kunde går ind og ser sine ordre, eller når en administrator ser alle ordre, fra deres adminOrders.jsp side, kørers der igennem et for-each loop der printer de forskellige ordre ud. Når disse for-each loops kørers igennem, kunne der implementeres denne ”*slet*” knap, der pegede på den enkelte ordre, og hvis blev klikket, både slettede den ordre fra session scope, såvel som database.

Mindre fejl: I løbet af kodningsprocessen er der stødt på forskellige fejl, ikke af den slags der stopper systemet, men som giver et forkert output til brugeren. Nogle af disse fejl er blevet ordnet

løbende, og på nuværende tidspunkt vides ikke hvor mange konkret var ordnet og hvilke ikke var, og der er igen ikke lige tid til at finde frem til alle. Selv ved nedskrivning af dette underkapitel, blev det eneste huskede eksempel hurtigt lige løst, så desværre ingen konkrete eksempler her, men det huskes som der var nogle af denne slags fejl.

Finpudses: Dette er også et ret generelt punkt, og flere af de overnævnte kunne teknisk set også hører under dette. Men her tænkes der mere konkret på punkter såsom interaktionsdesignet, eksempelvis at der eventuelt skulle være 2 knapper ved betalt.jsp, en, som nu, der fører tilbage til index.jsp siden og en der førte tilbage til den forrige side brugeren lige befandt sig på. Ligeledes kunne der være mange andre eksempler, på hvordan dette stadig kunne forbedres gevaldigt.

Proces

Grundet de personlige årsager/problematikker som nævnt, er alt ved denne opgave blevet udarbejdet på egen hånd og på mindre tid, end håbet på. Derved er det ikke muligt at beskrive et gruppearbejde og processen herved, da der intet var. Men der kan stadig beskrives mine egne arbejdsprocesser og hvad jeg kan tage med, selv fra en ualmindelig og forhåbentligt ikke gentagne arbejdsforhold/arbejdsproces.

Når jeg reflekterer tilbage på arbejdsprocessen, havde jeg ønsket at jeg havde gjort flere af de forberedende processer (domænemodel og lignende), inden jeg begyndte at kode. I dette tilfælde blev der først udarbejdet en fuldkommen *Mockup*, efterfulgt af implementering af de tabeller og koloner der skulle være i databasen, hvorefter kodningen straks begyndte. I forhold til selve arbejdet, har dette ikke været et problem, det har ikke forsinket kodningen, men fra et skolerelateret perspektiv har det ikke været det bedste. Som resultat, er jeg endt ud med at sidde og skulle lave mange af disse ikke koderelaterede opgaver, samme dag som denne rapport skrives, hvilket vil sige dagen forinden/på dagen for aflevering. Hvoraf det virker til, at der muligvis ligger dels mere prioritering på at der er taget hånd om alle processer, fremfor et helt færdigt kodet produkt. Så den tidsmæssige prioritering af disse, har været malplaceret og tages med i overvejningerne til fremover.

Ellers føler jeg, at den tid jeg har kunnet bruge på udvikling af projektet, er blevet gjort ret godt, eller så godt som nu muligt. Hvis der skulle være en ekstra lille mente, skulle det være at bruge Bootstrap mindre fremover, så vidt studiet tillader det i hvert fald. For mit had, for denne, er kun vokset i løbet af projektet, og har ofte vist sig, at være mere tidskrævende end alternativer. Så enten at skære ned på dette, hvis muligt, eller se frem til det efterfølgende semester, hvis ikke.

Til sidst ser jeg reelt frem til at prøve at arbejde en af disse projekter igennem, i en gruppe og se hvilke forskelle der er der, på godt og ondt. Så den, dels, optimistiske tilgang til gruppearbejde, er også noget jeg vil tage videre med mig, og se frem til.