30.03.2023

Deltagere:

- **Caner Turan**
- Github: Turan10
- Mail: cph-ct193@cphbusiness.dk
- Klasse: B

- **Orhan Secilmis**
- Github: Ghostfear1713
- Mail: cph-os104@cphbusiness.dk
- Klasse: B

- **Martin Thuren Christensen**
- Github: Martinthuren
- Mail: cph-mc489@cphbusiness.dk
- Klasse: B

- **Philip Hatley**
- Github: PhilipHatley
- Mail: cph-ph140@cphbusiness.dk
- Klasse: B

Copenhagen Business Academy
Datamatiker 2. Semester.

Cupcake

Indholdsfortegnelse

Cupcake

1.1 Indledning

Denne opgave henvender sig til en datamatiker studerende på 2. semester.
Projektet handler om at tage en fysisk cupcake butik beliggende i Olsker 3770 på Bornholm ”online” dvs. lave den webbaseret som man som kunde kan kigge, bestille og købe cupcakes via deres hjemmeside.

Opgave består i at udarbejde et projekt, som kan håndtere et bestillingssystem for vores kunde, Olsker Cupcakes. Det er et lille bageri i Bornholm, som har ønsket et system som kan bestille cupcakes, med topping og bund til. Kunden skal bl.a. kunne bestille en ordre for en cupcake, hvor en valgfri bund/top kan tilvælges, indsætte ordren i en indkøbskurv, og få en ordrebekræftelse på det man har bestilt.

Hertil har vi udarbejdet en domain model, som viser hvordan de forskellige entities er relateret til hinanden og et EER-diagram som viser forholdet af entities på databasen.

Cupcake

1.2 Baggrund

Virksomheden Olskers Cupcake er en lille butik beliggende i byen Olsker, på Bornholm. Butikken rummer 3 ansatte, og har indtil d.28.03.2023 kun været en fysisk butik. Olsker Cupcakes fik lavet en webbaseret butik.
Kundens ønske var en hjemmeside som var modelleret efter aftale, hvor på der skulle være mulighed for at bestille og betale cupcakes til afhentning.

1.3 Teknologi valg

Der er brugt følgende programmer til udviklingen af ”Cupcake”.

- IntelliJ, version: 2021.3
- MySQLWorkbench, version: 8.0.31
- Tomcat 9.0.73
- Figma
- Trello

Cupcake

1.4 Krav

Funktionelle krav

Det første kundemøde mundede ud i en række såkaldte ”user-stories”. De beskriver på kort form hvilke brugere, som har hvilke behov og hvad de ønsker at opnå.

- 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.

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

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

- Som kunde kan jeg se mine valgte ordrelinier i en indkøbskurv, så jeg kan se den samlede pris.

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

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

- 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.

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

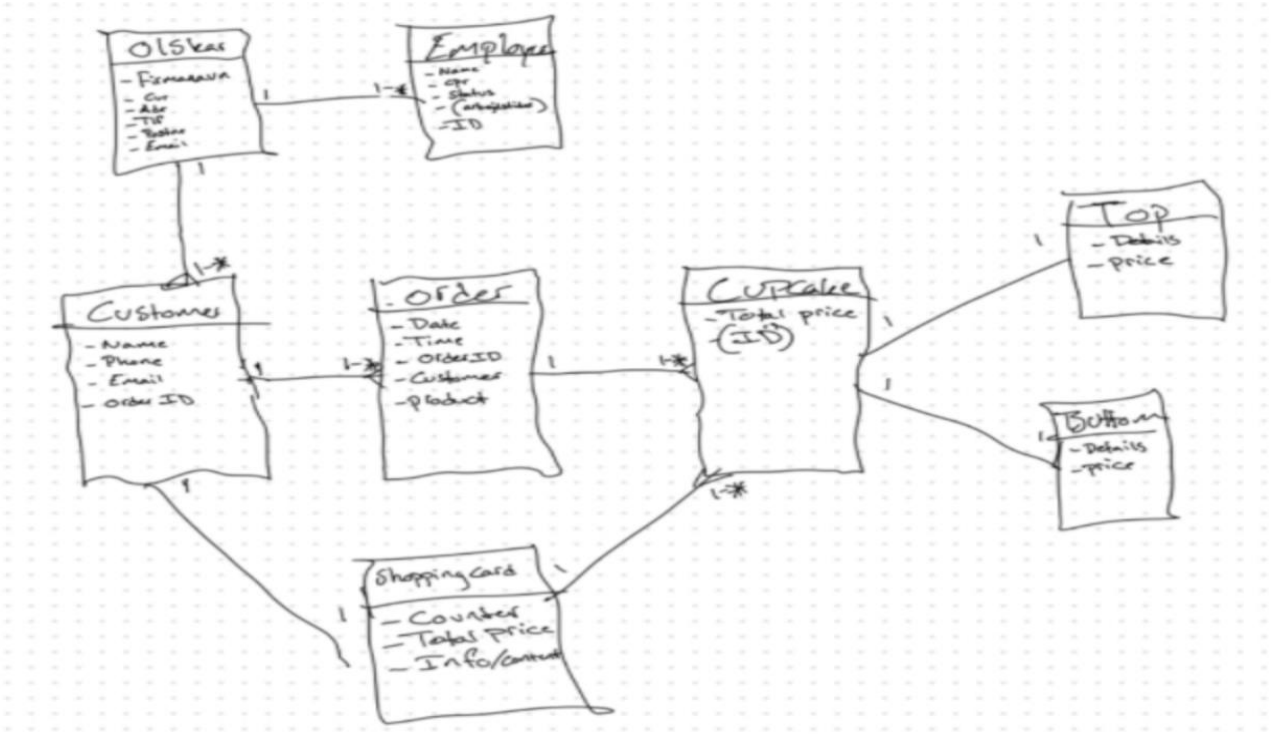
- Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Cupcake

1.5 Domæne model

Vores domæne model beskriver hvordan vi havde udtænkt vores opgave til at starte med, og hvad systemet arbejder med. Vi havde tanke om, at vi ville lave både en customer, en employee og en administrator. Hver customer har en 1 til 1* relation til en order, da en customer kan have mange order, men der kun må være en customer. Vi havde tænkt os, at vi ville samle et cupcake ID ud fra et ID fra top og bund, som samlet skulle give et ID videre til order, og på den måde kunne man altid hente en order fra systemet - Dette nåede vi også i mål med.

Vores shoppingcart skulle på en måde snakke både med customeren med også med cupcaken. Vi blev klogere senere hen, og vurderede at shoppingcarten blot skulle være en del af customerens session scope og ikke have et direkte forhold til cupcaken. Customer, admin og employee synes vi blev for rodet at dele op. Vi havnede vi med en user, som havde en ”role” der kunne være enten customer, admin eller employee. Cupcaken snakkede vi lidt om, hvordan vi ville lave relationerne her. En fra gruppen snakkede om, om vi skulle gøre det muligt at få lavet en cupcake med mere end 1 bund (eller en cupcake med 2 toppe), men det det besluttede vi os om ikke skulle være tilfældet. Vores domæne diagram har ændret sig siden, men sammenspillet mellem dem "in its core" er stadig den samme - Valgfri bund/top med ID's som sendes til en cupcake (1-1 relation) - lav en cupcake ud fra dem og giv den et ID (sørge for at kunne oprette flere cupcakes til en ordre) - sende den til en ordre som så kan sende det til en shoppingcart.

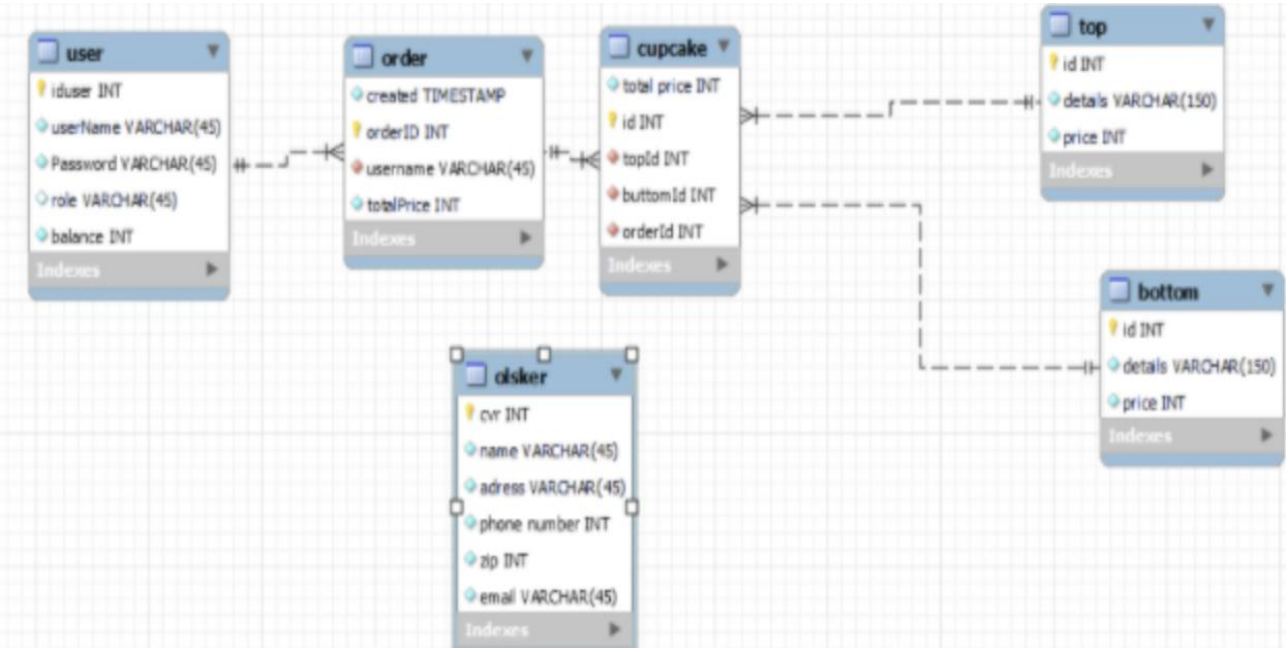


Cupcake

1.6 ER diagram

Vores EER diagram er blevet ændret en del gange henover opgaven - Vi havde flere tabeller til at starte med som vi har kortet ned til 6. Vores top og bottom er tilknyttet til cupcake tabellen med FK's, således at vi kan få info fra de tabeller ved cupcake tabellen. Ved vores order tabel, har vi valgt at bruge username som en sekundær PK som er tilkoblet attributten username i user tabellen. Det har vi gjort da vi i vores kode har username som det unikke - Der er ingen bruger som kommer til at have det samme brugernavn. Som udgangspunkt har vi sørget for at overholde normaliseringskravet om, Alle non-prime attributter skal være fuldt funktionelt afhængige af primær nøglen. (dvs. ingen partiel funktionel afhængighed). Ellers er relationerne

1 user kan have mange ordre, men en ordre kan ikke blive skabt uden at være tilknyttet 1 user
En ordre kan have mange cupcakes, men cupcakes kan kun tilknyttes til 1 ordre
En cupcake kan kun have 1 top og 1 bund, men der kan laves adskillige cupcakes med 1 bund og 1 top.



1.7 Særlige forhold

Sessionscope

Det skal påpeges at shoppingcart kører i sessionscope. Dette gøres da der ikke tale om indhold der skal gemmes i en database, men blot et sted for brugeren at kunne bekræfte sine valg.
Når brugeren vælger at fortsætte fra shoppingcart til checkout, gemmes indholdet i databasen i form af 2 tabeller. Den ene for ordren, mens den anden er for de cupcakes der er bestilt. Dette gøres for at kunne holde styr på hvilke cupcakes der er bestilt, samt hvilken ordre de hører til. Dette gøres ved at bruge ordreID som foreign key i cupcake tabellen.

Exceptions

Exceptions håndteres i servlets, og i mapperne. Mapperne sender database exceptions videre til servlets, som så håndterer dem. Dette gøres for at sikre at exceptions ikke bliver sendt videre til jsp siderne, da dette kan give en sikkerhedshul. Vi har i vores kode en klasse for DatabaseExceptions som vi gør brug af. Dette gøres for at sikre at exceptions bliver håndteret på samme måde i hele vores kode. Dette gøres ved at lave en metode der håndterer exceptions, og som så bliver kaldt i alle mapperne.

Validering

Validering af brugerinput gøres under oprettelsen af en bruger, og login.
For oprettelsen tjekkes der om brugeren skriver den samme password i begge felter (password og repeat password). Dette gøres ved at sammenligne de to felter, og hvis de ikke er ens, bliver brugeren sendt tilbage til signup siden med en fejlbesked.
For login tjekkes der om inputtet i usernamme og password eksisterer i databasen. Dette gøres ved at kalde en metode i UserMapper som tjekker om brugeren eksistere i databasen. Hvis brugeren ikke eksistere, bliver brugeren sendt tilbage til login siden med en fejlbesked.

Brugertype

Vi har valgt at lave et felt i vores user tabel kaldet role. Denne ”role” skal differentiere brugerens rolle i vores system. Vi har valgt at lave 2 brugertyper, nemlig user/kunde og admin. Dette har vi gjort for at kunne sikre at brugere ikke kan tilgå sider de ikke har adgang til. Brugertyperne bruges i jdbc, således at der tjekkes om brugeren har den korrekte rolle, før de kan tilgå admin relaterede sider.

Sikkerhed

Udover vores tjek på login og opret brugere, samt ikke at kunne ligge en ordre uden at have nok midler, har vi ikke implementeret sikkerhed.

Cupcake

1.8 Status på implementation

Vi har i vores gruppe haft udfordringer med at få kodet funktionel kode.
Dette har gjort at vi ikke har nået at implementere alle de krav der er stillet i opgaven, og har derfor valgt at håndplukke de essentielle elementer, og krav for at kunne have en kode der virker.
Dermed kan man med vores program, logge ind, eller oprette sig, sammensætte en cupcake, og bestille den.

Vi mangler at implementere en side hvor brugeren har adgang til alle sine ordre, og det der er bestilt på hver ordre. Vi mangler samtidig også at style vores sider, så de er mere brugervenlige. Admin mangler en jsp side, hvor admin kan se alle ordre, og hvad der er bestilt på hver ordre i vores shoppingcart har vi ikke implementeret en funktion for afhentningstidspunkt. Vi mangler at implementere test i vores program.

1.9 Process

Vores planer i starten for projektet gik rigtig godt. Vi lagde alle sammen en fælles plan, havde en fælles afstemning om hvordan det vi ville gribe projektet an, hvor morgenmøder, tidsrum vi arbejdede, spørge hvis der var behov for hjælp og vigtigst af alt, give det projektet alt det vi har. Som projektet forløb sig, kom 3 fra gruppen mere og mere bagud. Dette blev ikke kommunikeret ud, hvilket resulterede sig i, at ingen vidste hvordan det rigtig gik. Morgenmøderne blev skippet. Man skrev hurtigt hvordan det gik, men der var ingen reelle resultater der hverken blev lavet eller fremvist så man havde en idé om hvor henne man var med opgaven. Efter flere dage med dette, kom der en ret sent udmelding om, at man faktisk havde det rigtig svært. Folk havde svært ved at sige det højt, trods aftalen der blev lavet for start. Det har derfor resulteret i, at Turan har skulle step op og som følge af dette har lavet langt største delen af opgaven. Vi har slet ikke været gode til være et hold og der har været et enormt stort pres på vores gruppemedlem

2.0 Hvad har vi lært?

Udbyttet af projektet har været mest synlig da vi sad med Jon d. 28 april, hvor de som har været bagud har kunne få noget hjælp og få forklaret diverse syntax og sammenspillet mellem de enkelte klasse med JSP'en. Turan har været i stand til at få udbyttet ved at kode for sig selv, fejle og spurgt om eksternt hjælp når behovet har været der. Vi er sikre på, at hvis planen var fulgt op på og man havde kommet til læren meget tidligere i forløbet, at mange af vores udfordringer ikke havde været nært det her niveau.

Cupcake

2.1 ”Til næste gang”

Vi skal have lagt en god plan. Hvis planen ikke afholdes, så reagerer man som gruppemedlem ASAP. Man får snakket med ens gruppe og får vendt tingene med dem, således at alle ved hvad der foregår og hvor der er udfordringer. Muligheden for at hjælpe når intet siges eller gøres, bliver i den grad udfordret hvis ingen i gruppen handler på en udfordringer man har. Vi har også forstået vigtigheden i at læse godt op på det stof man er udfordret med inden man går ind i et stort projekt. Den første tid bør ikke bruges på at undersøge hvad teorien går ud på, men i stedet en plan på hvordan man vil takle opgaven.