Cupcake

Simon Nielsen, cph-sn286@cphbusiness.dk
https://github.com/cph-sn286

Nicklas Hansen, cph-nh192@cphbusiness.dk
github.com/Kofoedsan

Nicklas Bergmann, cph-nb191@cphbusiness.dk
https://github.com/m4cfly

Thorbjørn Jensen, cph-tj308@cphbusiness.dk
https://github.com/ThorbjoernJensen

René Stafaniuk Andersen, cph-ra147@cphbusiness.dk
https://github.com/Rene061089

Link til droplet: http://165.22.27.222:8080/cupcake%20demo/

Github link til projekt: https://github.com/cph-sn286/cupcake-projekt

Klasse C

Gruppe 7

Dato: 28-04-2021

Indholdsfortegnelse:

Indledning

Et nyt iværksætter eventyr er startet på Bornholm, nærmere Olsker nemlig en Cupcake forretning. De har netop udset sig Bornholm som den perfekte lokation for deres forretning. Men de mangler nogle til at bygge et website til dem, da de kun er eksperter i cupcakes og ikke programmering. De har dog mange krav til hvordan den skal opbygges.

Det helt primære krav var at det skal være en webshop, derfor skal *kunderne kunne bestille en cupcake hvor der er valgt både en top og bund,* Det skal bestilles via en profil derfor skal *kunderne kunne oprette en konto for at betale og gemme deres ordre.* For at de kan betale skal de også have penge på den konto, derfor var det et vigtigt krav at man *skal kunne indbetale på sin konto.* Det betyder at *kunderne skal kunne se deres ordrelinjer i en indkøbskurv, hvor de har mulighed for at slette en ordrelinje*. Men iværksætterne vil også have mulighed for at manuelt gå ind og slette en ordre. Derfor skal man som *"employee" kunne se alle kunder og deres ordre og have mulighed for at slette og holde styr på ordrerne.*

Teknologivalg:

For at kunne udføre dette for kunden, kræver det nogle brugbare redskaber. Selve Java delen har vi skrevet i IntelliJ 2021.1. Vi har brugt MySQL Workbench 8.0.24. til at holde styr på kunder og ordre. For at det kan køre sammen, kræver det en JDBC (Java database connectivity) Her har vi brugt version 4.3. Vi har valgt at bruge en tomcat 10.0.5 webcontainer til at ligge vores website på.

Krav

- **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 kunde kan jeg indsætte penge på min egen konto, så jeg kan betale for cupcakes.
- **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 email og kodeord. Når jeg er logget på, skal jeg kunne se min email på hver side
- **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.

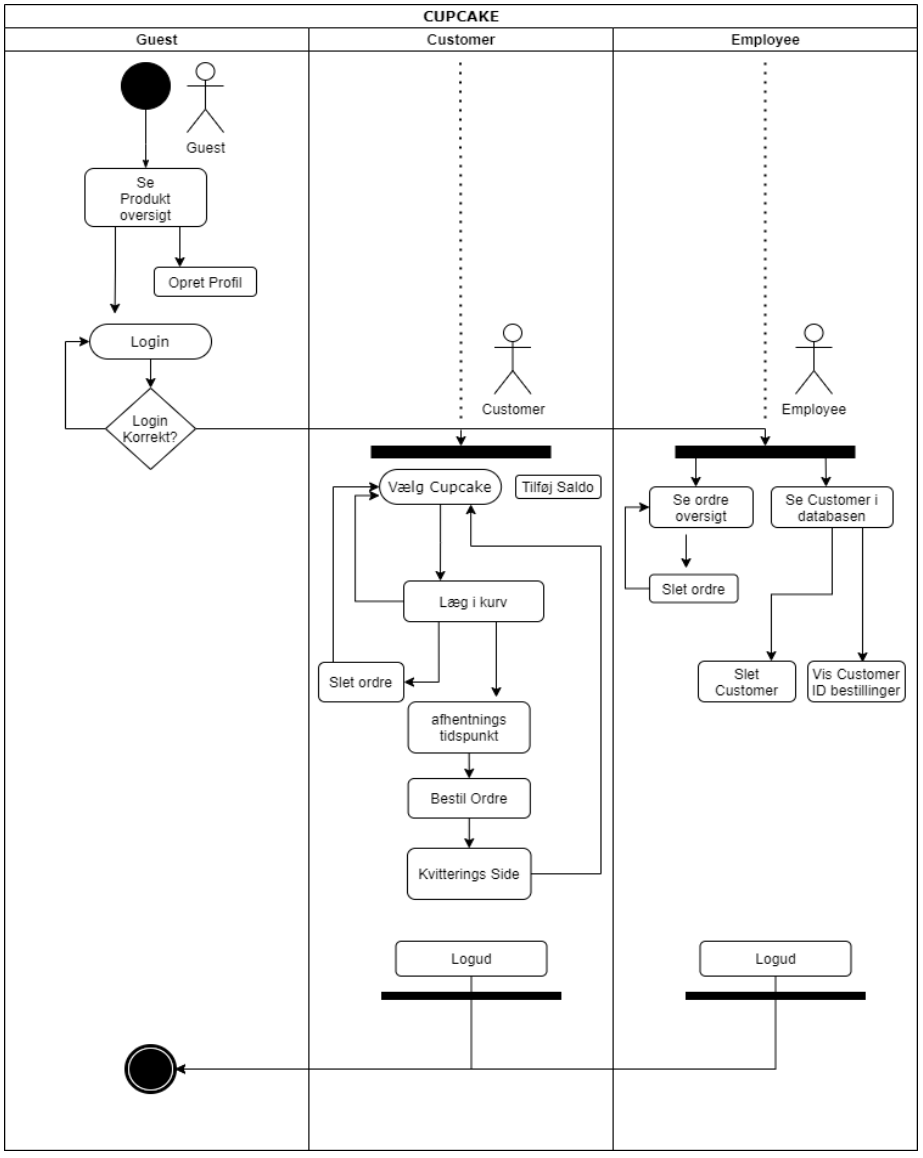
Aktivitetsdiagram

I aktivitets diagrammet illustreres hvordan vi kan tilgå de følgende funktionaliteter indenfor de 3 roller på websitet: **Guest, Customer, Employee**.

Guest: Man kan se Cupcake Oversigt og oprette en profil.

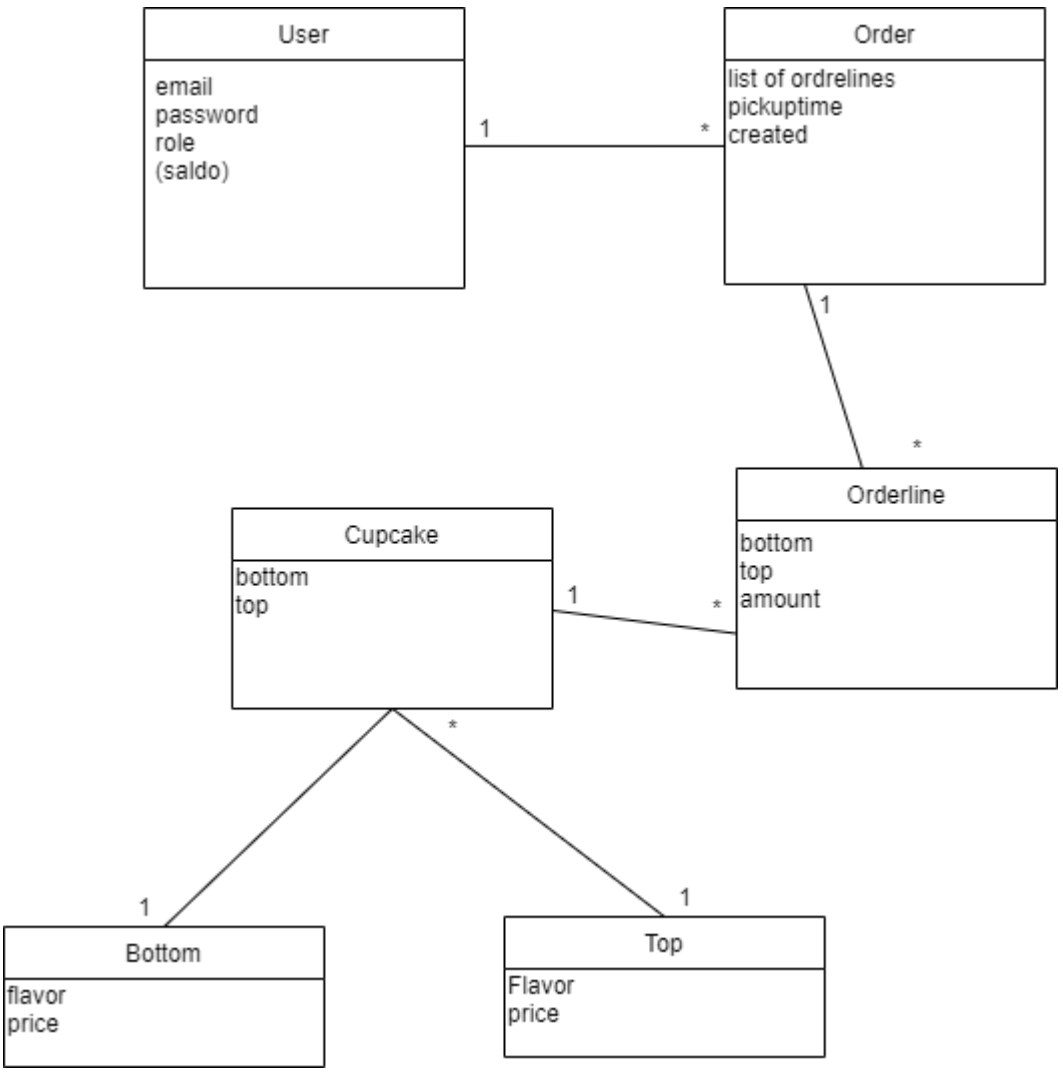
Customer: Login, Vælge Cupcakes, Læg i kurv, Bestil, Kvittering, Logud.

Employee: Login, Se Ordrer Oversigt + Slet , Se Customer i Databasen + Slet, Vis Customer ID + Bestillinger, Logud.



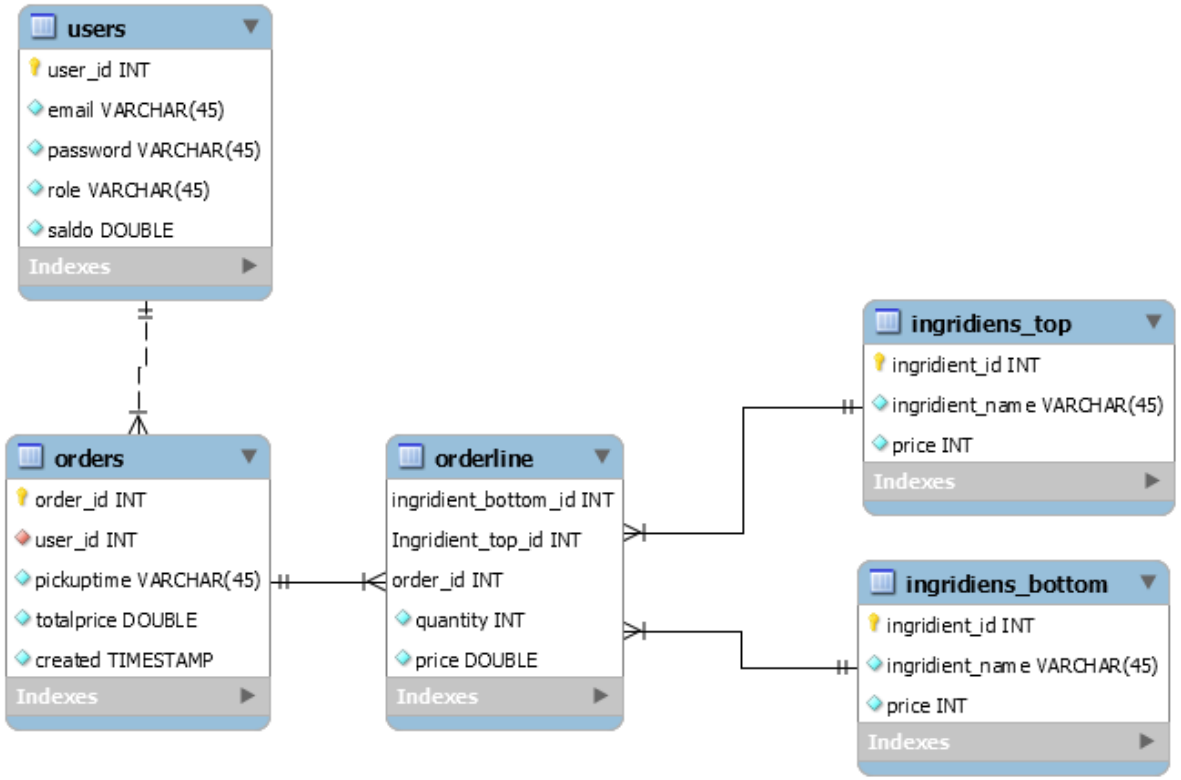
Domæne model og EER diagram

Herunder ses vores domænemodel for ordrehåndteringen i Olsker cupcakes. Titlerne på boksene svarer til de begreber, eller konceptuelle klasser, som kunden (Olsker cupcakes) selv ville beskrive deres arbejdssituation med - det er vores bestræbelse. Trods det at Olsker Cupcakes kan siges at være et bageri, lægger navngivningen af de konceptuelle klasser sig op af en forretningsforståelse af domænet, med ordrelinjer og ordre. Tanken er, at når iværksætteren går til ordrehåndtering er det som forretningsmand.



Den konceptuelle klasse “Order” svarer til en indkøbskurv, dvs. en samling varer (ordrelinjer). Det er ordren der samler al data der vedrører bestillingen, og knytter den til en kunde.

Selve cupcaken indgår her som en sammensætning af bund og top. og en ordrelinje består af et antal af en given cupcake. Her afviger domænemodellen fra vores EER-diagram (og også et evt. klassediagram som vi ikke har med her), da cupcaken ikke er en integreret del af hverken databasen eller java-koden. Her indgår klassen cupcake kun implicit som en sammensætning af en bund og en top.



Bortset fra at cupcake-klassen udgår, indeholder EER-diagrammet tabeller der svarer til domæne-diagrammet. Ud fra forbindelseslinjerne i diagrammet kan vi se retningen på en-til-mange relationerne: én kunde kan have mange ordre, én ordre kan have mange ordrelinjer. og hver bund og top kan indgå i mange ordrelinjer. Modsat kan en ordrelinje kun tilhøre én ordre, kun have én bund, og én top.

For at sikre dataintegritet er tabellerne bundet sammen ved at primærnøgle i én tabel indgår som fremmednøgle i den tabel den står i relation til.

For at opfylde 1. normalform, kræves at alle tabeller har en entydig nøgle. For alle tabellerne, undtagen orderline, imødegås dette ved en et at vi sætter primærnøglen til et autogenereret heltal, der tildeles ved hver tilføjelse i tabellen. Eneste undtagelse er "orderline" hvis nøgle består af de tre felter ingridient_bootom_id, ingridient_top_id og order_id.

Ordrelinjer persisteres først i det øjeblik at de bliver lukket i en ordrebestilling. Når vi koder på java-siden har vi derfor på nuværende tidspunkt ikke brug for en ordrelinje-id der er bundet i databasen. Vi bruger en id-nøgle på ordrelinjer på java siden (int orderlineId), men den er bundet op på session-scope, som i vores kode er afgrænset af brugerens login og logout.

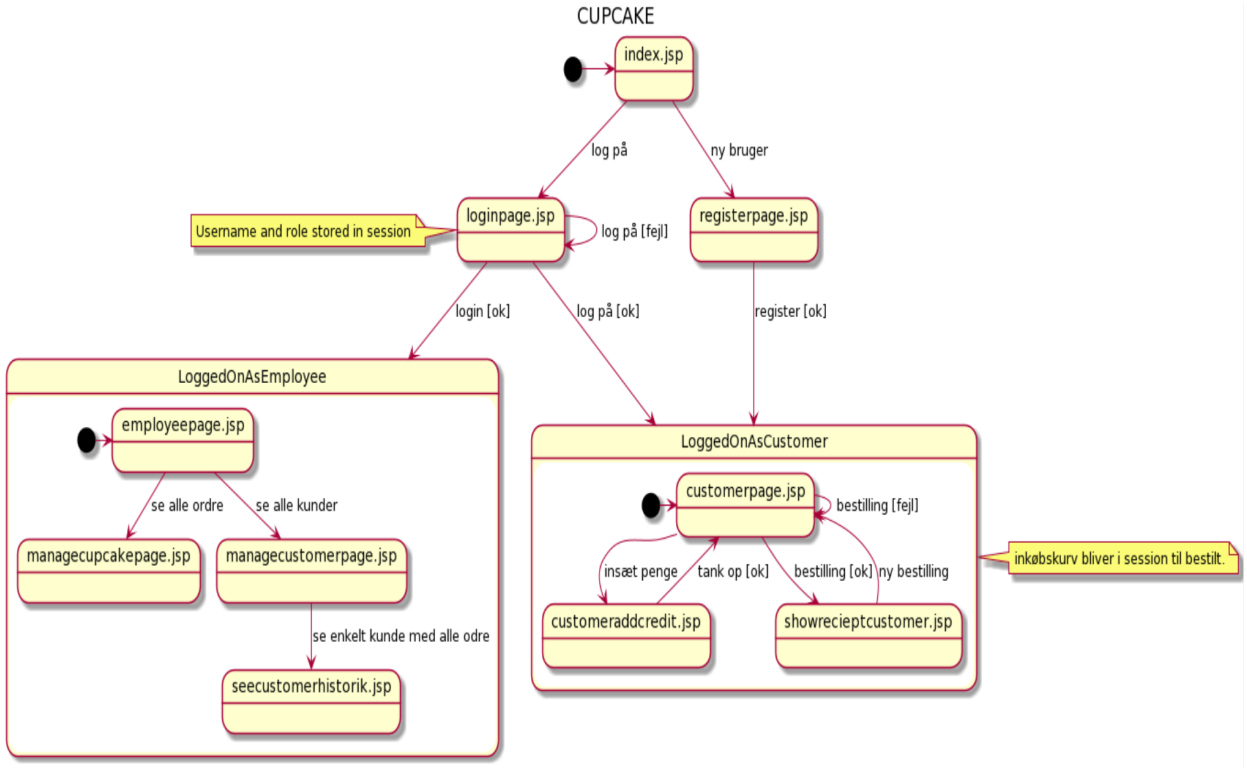
Sådan som vi har implementeret tilføjelsen af cupcakes i java, undgår vi at der kan oprettes flere ordrelinjer med samme bund/top-kombination indenfor samme session scope, og dermed ordre. Hvis der bliver lagt en allerede eksisterende bund/top-kombination i kurven, vil de to tilføjelser blive samlet i én ordrelinje.

At vi ikke har en orderline-id i databasen sparer os for at gemme noget data, en særskilt orderline_id, som ikke umiddelbart har nogen anvendelse. Dette er i tråd med den bestræbelse der driver normalisering; at man så vidt muligt undgår unødig data.

Hvis systemet skulle give mulighed for at ændre i enkeltvis i de persisterede ordrelinjer, ville det give bedre performance at kunne tilgå den enkelte linje med et query på formen "SELECT * FROM orderline WHERE orderline_id = ?". Som vores løsning er nu med tre primærnøgler, ville vi have brug for et, for serveren, mere krævende udsagn: " SELECT * FROM orderline WHERE bottom_id=? AND top_id=? AND order_id =? ".

Opdelingen af data i ordrer og ordrelinjer giver mulighed for at opfylde 2. normalform, ved at felter der vedrører ordren - ordre tidspunkt, og kunde-id ikke skal gentages for hver ordrelinje der tilføjes.

Navigationsdiagram



Vi har valgt at benytte os af en “fælles navigations bar” i toppen af alle siderne. Den vil variere afhængig af om man er logget ind som customer, employee eller slet ikke er logget ind

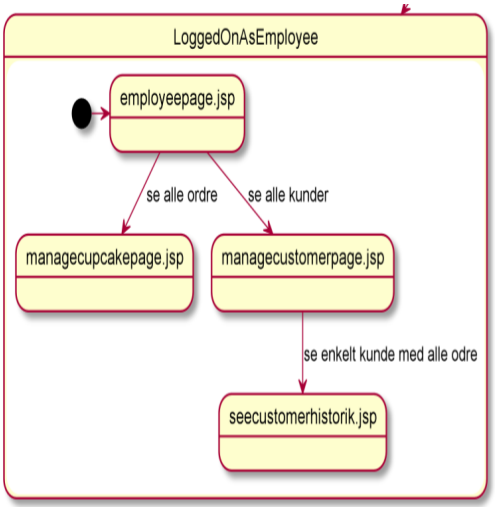
Når man ikke er logget ind, indeholder den “HOME”, “INDKØBSKURV”, “LOGIN”, “SIGN UP”. “HOME” følger navigationsbaren hele tiden lige meget om man er logget ind som customer, employee eller man slet ikke er logget ind. “HOME” vil altid tage dig til index siden. “LOGIN” og “SIGN UP” Gør præcis det de beskriver. OPS: “INDKØBSKURV” Virker kun hvis man er logget ind, hvis man bruger den før, bliver man bedt om at logge ind.

Home Indkøbskurv rene@123 Saldo: 4835.0 Logout Tilføj Beløb

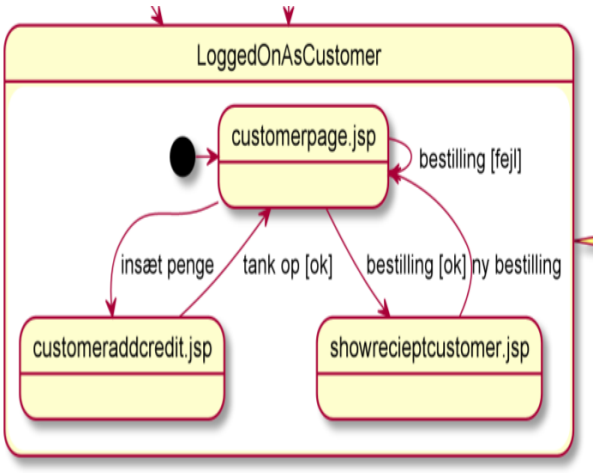
Når man logger ind som customer bliver man ført til custormerpage som er startsiden for brugerne. "LOGIN" og "SIGN UP" bliver udskiftet med "LOGOUT" og "TILFØJ BELØB". "LOGOUT" vil logge dig ud og sende dig til index siden. "TILFØJ BELØB" sender dig til tank-op siden hvor man her kan indsætte penge på sin konto. Yderliger er der tilføjet 'brugernens' email/brugernavn til navigationsbaren, plus 'brugerens' nuværende saldo. Benytter brugeren sig af "INDKØBSKURV" bliver brugeren sendt til customer page som indeholder muligheden for at bestille cupcakes.

Home Indkøbskurv Role: employee Logout Tilføj Beløb

Når man logger ind som employee bliver man ført til employee page som er startsiden for employees. I navigationsbaren for employee er det kun "HOME" og "LOGOUT" som er relevante da employee ikke kan nå "INDKØBSKURV" og "TILFØJ BELØB".



Når man er logget ind som employee kan man nå indholdet der er er på: employeepage.jsp, managecupcakepage.jsp, seecustomerhistorik.jsp, seecustomerhistorik.jsp.



Når man er logget ind som customer kan man nå indholdet der er på: customerpage.jsp, showrecieptcustomer.jsp, customeraddcredit.jsp.

Alle kan nå indholdet der er på: index.jsp, loginpage.jsp, registerpage.jsp.

Særlige forhold

- **Scopes:**

  Hver bruger session gemmes i sessionScope..
  Dvs, brugerId, email, role, saldo,password, kvittering.
  indkøbskurven, som består af en liste af ordrelinjer, gemmes også i sessionscope, og nulstilles når der afgives bestilling.
  I application scope gemmes lister over cupcake top/bund (dette gør vi i Frontcontrolleren).

- **Exception handling:**

  Der er lavet en UserException class i exceptions packages.
  Denne java klasse nedarver fra klassen Exception, og returnere fejlbeskeden til brugeren.
  Den er ikke fuldt implementeret på alle metoder, de metoder hvor UserException ikke er implementeret, vil den udskrive en fuld systemtrace til brugeren. Hvilket ikke er ideélt som forbruger.
  Metoderne skal derfor kunne 'throw(s)' UserException. Ligesom diverse catch/try skal redigeres til den brugervenlige fejlbesked.

- **Validering af brugerinput:**

  Bootstraps inputfelter kan sættes til typen input af numbers, samt text.
  På den måde sikres at der ikke tastes en ugyldig commando.
  Derudover, er der anvendt sliders og drop down menu's, som gør at brugeren ikke kan indtaste forkerte oplysninger.
  //sidste mulighed for brugerinput er tryk på knapper, som også er et styret input af data.

- **Sikkerhed:**

  Hver bruger har et unikt brugerID samt email, password og role.
  Dog er der ikke sat nogle krav til passwordet længde, karaktere eller lign.

Når en forbruger opretter en profil, eller logger på, tjekkes deres login info op imod MySQL databasen, hvis brugeren har rollen “Customer”, vil admin funktionerne ikke være tilgængelig, da det kræver at brugerrollen er sat til employee.

Programmet kører lokalt på en droplet fra DigitalOcean på localhost.

Forbindelsen til tomcat webserveren er ikke krypteret, derfor er sikkerheden lav.

Der er oprettet en specifik MySQL burger, som kun har basale CRUD rettigheder.

- **Hvilke brugertyper, der er valgt i databasen, og hvordan de er brugt i jdbc:**

  Der er som nævnt ovenstående, 2 brugertyper. Employee & customer.

  I jdbc bliver alle kommandoer kørt igennem en commando klasse, her findes to typer, protected & unprotected.

  Protected kræver at den bruger der er logged in, har samme rolle som der er prædefineret. På den måde sikres det, at en bruger af typen customer ikke kan tilgå employee funktioner.

Status på implementation

- **Krav:**
  Alle usercases er implementeret helt eller delvist.

- **Mangler i funktionalitet:**
  Når man opretter en bruger bliver brugerdata ikke gemt i sessionscope, så for at sidens funktioner kan virke skal man logge ind igen.
  Når en employee sletter en ordre, skal saldoen på ordren føres tilbage til brugeren.
  Employee kan se OrdreId sorteret efter UserId, men mangler funktionalitet til at slette/ændre bruger oplysninger.
  F.eks, e-mail, password, saldo.
  Employee kan kun slette en ordre fra en customer, ikke foretage ændringer.
  Metode for en bruger til at slette/annullerer en ordre. Det er Pt. Kun employee der kan fjerne en ordre.
  Refakturering af koden mangler.
  Vi er i tvivl om databasen opfylder 3. normalform da, price er afhængig af quantity i orderline og ingen af dem er en primær nøgle.

- **Styling:**
  Mangler overskrift i headeren, skal fjernes/rettes fra at være “Demo project for DAT 2.semester” til noget mere passende.

  Styling efter Adobe XD mockup mangler i et omfang. Programmets interface er brugbart, men ligner endnu ikke helt det mockup der foreligger.

- **Fejl:**
  Nogle gange bliver ens orderlinje ikke tilføjet til kurven. Dette virker til at forekomme tilfældigt.

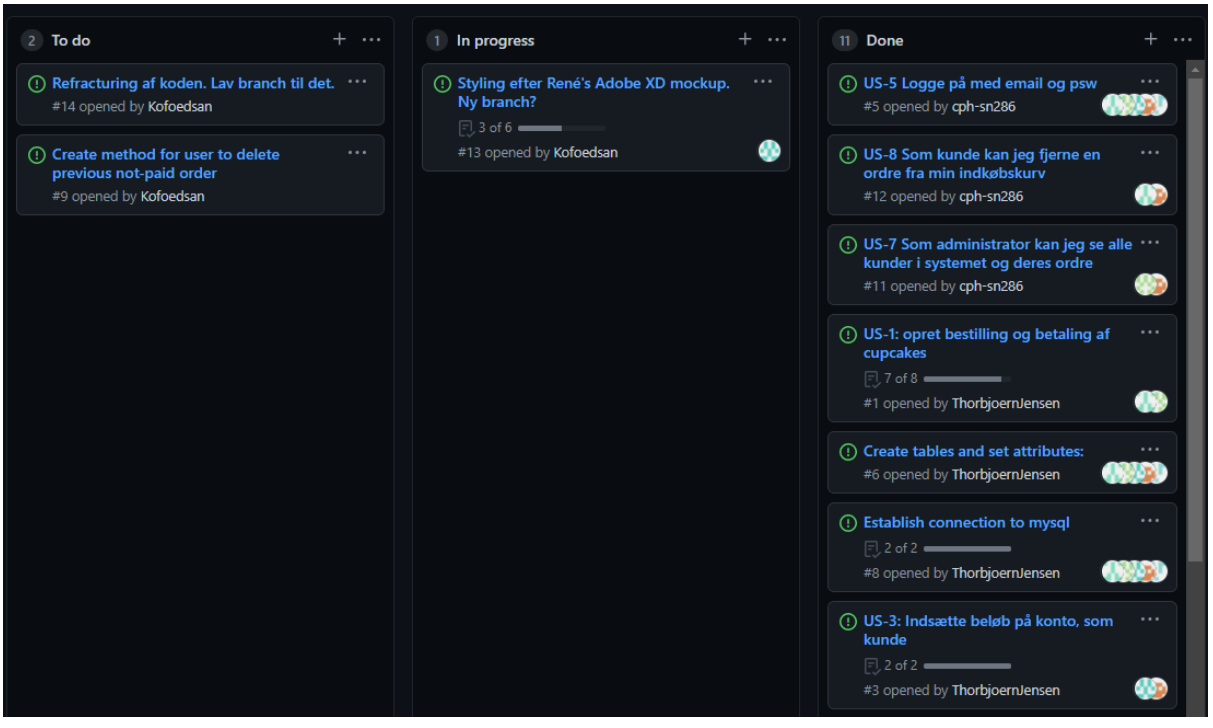
Ikke alle metoder har fået implementeret UserException klassen. F.eks. vil database metoder smide en SQLException.

- **Database:**
  Alle CRUD operationer er lavet til metoderne, til alle tabeller. Mangler UserException.

- **Test:**
  Vi har ikke lavet unittest og integrationstest, men vi har afprøvet programmet i forhold til hvad vi mener er normal brugeradfærd. Vi har haft hjælp af en anden gruppe, via en droplet, til at finde fejl.

Proces

Vi vidste på forhånd at med 5 mand i gruppen, ville det være oplagt at dele opgaverne op og ud, så alle havde noget at lave hele tiden, de større opgaver delte vi op 2 og 3. Før start havde vi snakket om nogle løsninger og ideer til hvordan vi kunne



få det til at fungere. Vi valgte at lave en Kanban med issues, så vi altid havde et overblik over hvad der var lavet og hvad der skulle laves.

Det fungerede ret godt, med at tjekke ind hver morgen, og vise hvad der var blevet lavet. Her fik vi også snakket om, hvad der manglede før en user story var (færdig). Vi havde en rigtigt god kommunikation, og alle var gode til at møde, til de aftalte tidspunkter.

Vores plan om at bruge Kanban fungerede rigtigt godt. Det har hjulpet os med at have et overblik og holde styr på projektet og opgaverne dertil. Vi har lært at det er rigtigt smart at dele opgaverne ud, hvor man sidder i par og løser dem. Vi havde måske lidt et problem i at vi var 5 personer i gruppen, derfor krævede det at vi havde styr på hvad der skulle laves.

På github har arbejdet med en branch for hver usecase, hvilket har lettet overblikket og gjort det let at arbejde på flere usecases samtidigt.

Bilag

Bilag 1:

Eksisterende logins

user_id	email	password	role	saldo
1	bruger1	1	employee	500
3	bruger3	3	customer	70
9	bruger9	9	customer	444