Olskers Cupcakes

Dette projekt er lavet af Gruppe 1 bestående af:

Israa El-Haj Moussa - cph-ie51@cphbusiness.dk - github.com/israa1809

Malene Rolsted Christensen - cph-mc452@cphbusiness.dk - github.com/malenec

Mark Pihl Kousgaard Pedersen - cph-mp677@cphbusiness.dk - github.com/Marpeddata

Sebastian Timothy Berry - cph-sb468@cphbusiness.dk - github.com/berry1196

Martin Hougaard Lorentzen - cph-ml724@cphbusiness.dk - github.com/zendary

Link til demo af website på youtube:

https://youtu.be/g8m8WdjbTJw

Link til projekt på github:

https://github.com/Berry1196/cupcake

Dette projekt blev lavet:

Fra den 7. november 2022 til den 17. november 2022

Indholdsfortegnelse

Indledning

Vi har i dette projekt arbejdet med en case for virksomheden Olskers Cupcakes, som bager og sælger cupcakes, virksomheden har en butik på bornholm og sælger til lokalområdet. Vi har fået til opgave at udvikle dem en hjemmesiden med dertilhørende webshop. De formelle krav fra Olskars side var et ønske om en mulighed for deres kunder skal kunne sammensætte cupcakes ud fra deres ønske og smag. Her skal de kunne vælge en top og bund via en dropdown menu og blive præsenteret for prisen, samt tilføje ønsket til kurven, hver kunde har sin egen bruger hvor de kan se en oversigt over ordre og her også blive informeret om hvornår en ordre er klar til afhentning i Olskers forretning.

Teknologivalg

Vores valg af teknologier består af dem vi er blevet introduceret til gennem cphbusiness.

Vi har brugt:

IntelliJ 2021.2.4 som IDE

Java - Amazon Correto 11

MySQL

Maven

Krav

Firmaets håb:
Olsker cupcakes ønsker at forøge deres salg af cupcakes ved at få en hjemmeside hvor man kan bestille cupcakes uden at skulle tage ned i butikken.
En generel streamline af brugerens oplevelse i form at bestilling og afhentning af cupcakes gennem hjemmesiden.

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 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 (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.

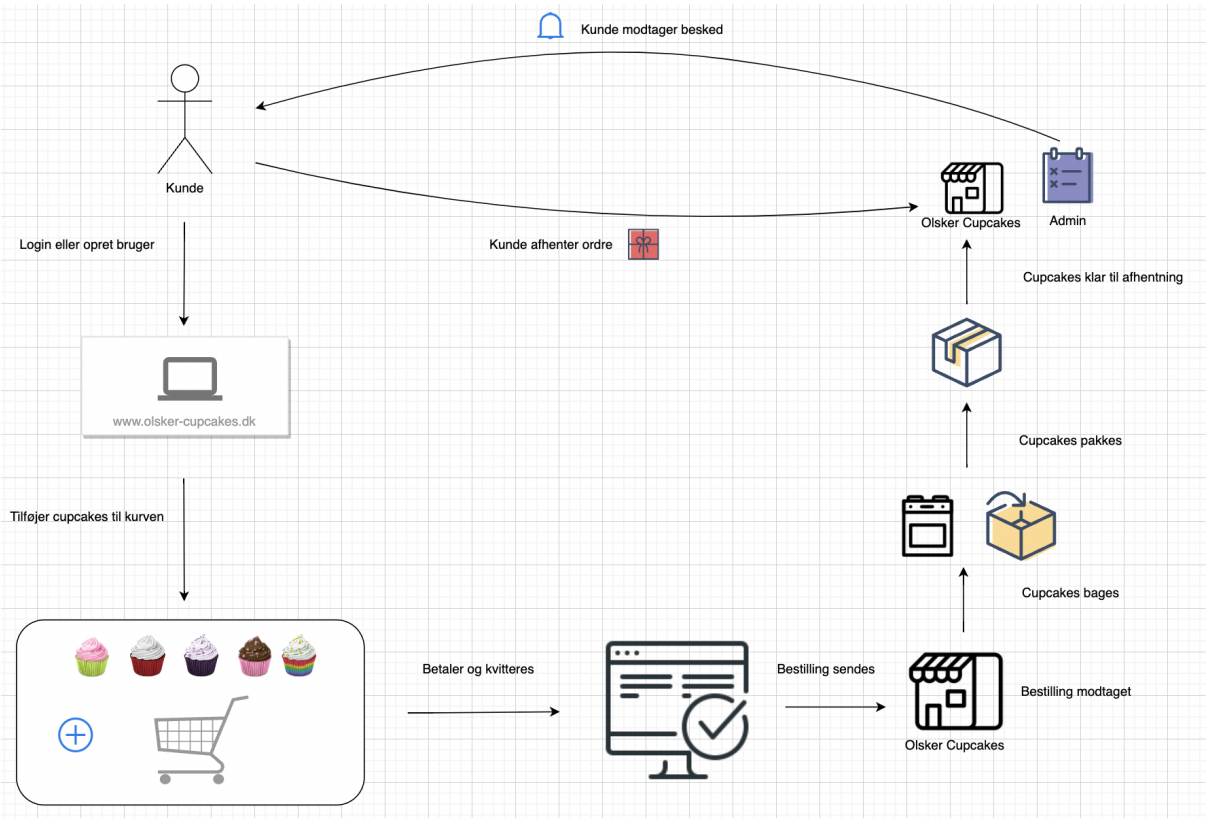
US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

US-8: Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.

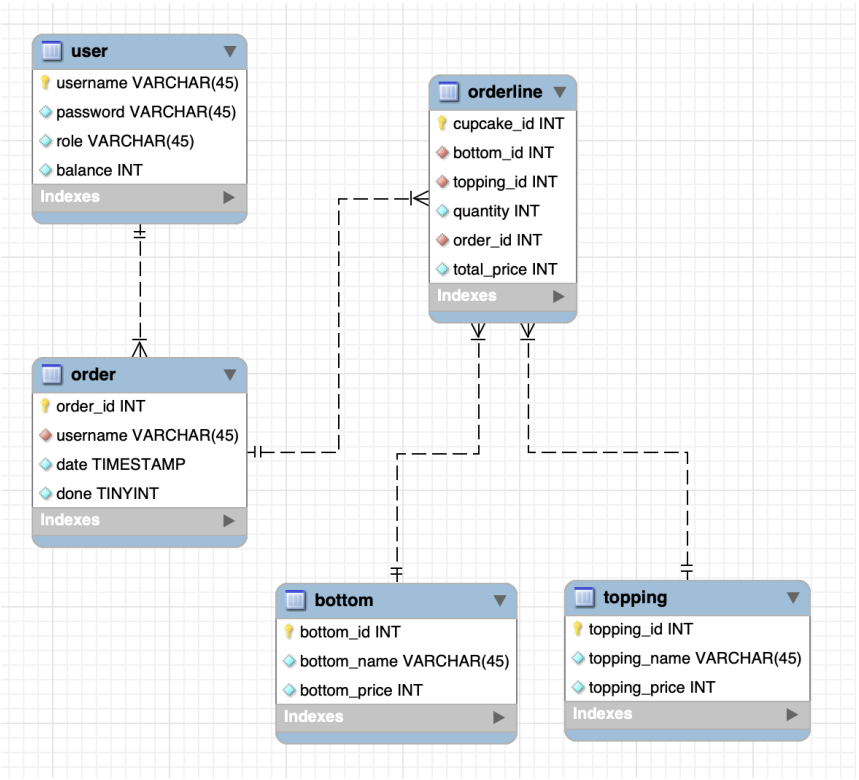
US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Domænemodel og EER diagram

Domænemodel:



EER Diagram:



Særlige forhold

Session

Vi tildeler en given user en shoppingCart ved login og denne hægtes op på sessionen. Listen af kager man lægger i sin indkøbskurv kan dermed bringes med videre gennem både muligheden for at ændre og slette i sine bestillinger samt i sidste ende at afgive ens ordre.

Exceptions

Vi har valgt at ændre drop down menuen på på forsiden, så antallet af kager automatisk sættes til int 1 med mindre man tager et aktivt valg, så vi undgår at få en String med “vælg antal” sendt videre, hvis brugeren glemmer at tage dette valg. Dette er gjort for at kunne samle de 3 drop down menuer med valg af kagebund, kagetop og antal under én nullPointerException.

Validering

Hele systemet kører via drop down options undtagen ved opret ny bruger og login. Ved login sammenlignes der med databasens brugerinfo via equals metoden. Det samme gælder ved opret bruger, hvor equals metoden både bruges til at tjekke at adgangskoden er ens samt at brugeren ikke allerede findes i databasen.

Sikkerhed

Alle jsp sider der bruges efter login såsom velkommen-side, kurv, betaling osv ligger i WEB-INF og kan dermed ikke tilgås uden at være logget ind. Dette samt de tjek der sker ifm equals metoden som beskrevet ovenfor er den mængde sikkerhed der foregår i dette program.

Brugertyper

Der findes på nuværende tidspunkt kun 2 brugertyper der kan tilgå systemet; en ‘user’, hvilket er en kunde og en ‘admin’, som er en fra kagebutikken. Der er et antal servlets og og jsp sider der udelukkende er rettet mod admin og tilsvarende mod user. Rollerne gemmes i databasen som VARCHAR(45), så der er mulighed for udvidelse til flere brugertyper, men vi ville i det tilfælde mangle kravet til 3. normaliseringsform, så der skulle oprettes en tabel til brugertyper med et tilhørende ID.

Status på implementation

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.	Kunden kan vælge en top og bund efter eget valgt, samt hvor mange der ønskes af den specifikke sammensætning. På kundens ordrestatus kan de se status på ordren når den er klar til afhentning.	120%
Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en en ordre.	Brugere på siden vi har udviklet kan oprette sig som bruger og se ordre samt ordrestatus, det er dog ikke muligt at gemme en ordre uden også at lave en bestilling med betaling osv.	80%
Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer.	Administrator har mulighed for at indsætte beløb på en given brugers konto. Det fungere på den måde at administrator skriver brugernavn og beløbet ind og kan derefter klikke for at indsætte på konto i databasen.	100%
Som kunde kan jeg se mine valgte ordrelinier i en indkøbskurv, så jeg kan se den samlede pris.	Kunden kan efter at have valgt topping og bottom af en eller flere cupcakes og puttet det i kurven, se i kurven den samlede pris af af cupcakesne	100%
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	Brugeren kan i løbet af hele login sessionen se sin Email i venstre side af nav baren på alle siderne på websitet.	100%

topmenuen, som vist på mockup'en).		
Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.	Administrator kan se alle ordre på en samlet liste og klikke på "klar til afhentning" for at synliggøre for kunden når deres ordre er lavet og klar til afhentning. *Ordredelen af programmet mangler at blive færdigudviklet, og derfor hentes ordrerne på nuværende tidspunkt op fra databasen via et HashMap, hvilket resulterer i en noget kaotisk visning på ordre jsp-siden. Da der ikke er bygget et ordrelinie objekt hentes ordrerne op fra databasen via den del af systemet der var designet til indkøbskurven, som er et array af kager.	100% *20%
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.	Vores adminside viser alle de ordrer der er igang og alle dem der har været. Der bliver ikke printet en reel liste ud af alle kunde, men så længe en kunde har oprettet en ordrer vil de blive vist på adminsiden. Der kan holdes styr på ordrene fra kunderne gennem muligheden for at have dem enten i "Behandles" eller "Klar til udlevering" En tredje mulighed kunne være at have en "arkiverings" mulighed til de afhentede kager.	70%
Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.	I kurven er der er slet knap til at fjerne et valg af cupcake valgt på bestillingssiden. På denne måde opdateres kurv siden hvor den ikke længere fremgår og den samlede pris opdateres også til at afspejle ændringen.	100%
Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde udgyldige ordrer. F.eks. hvis kunden aldrig har betalt.	Vi har ikke implementeret muligheden for at admin kan fjerne en ordre der ikke er betalt. Dette er fordi vores program er opbygget sådan at en ordre først er oprettet i databasen når kunden har betalt med de penge der er tilknyttet deres account gennem databasen. Vi har i stedet tilføjet en boolean(tinyInt) i vores database der bruges til at fortælle om hvorvidt en ordrer er færdiglavet fra butikkens side.	50%

Proces

På opgavens start dato satte vi os sammen og tænkte højt om hvordan vi skulle opbygge vores MySQL database og ER diagram. Vi oprettede et google docs diagram til at holde overblik over hvilke primære funktioner der skulle udvikles, hvor gruppens medlemmer ville arbejde og hvornår vi kunne arbejde. Vi planlagde at mødes alle hverdage kl 09:00 - 15:00, for at diskutere hvad der skulle laves på dagen og hvem der ville være på hvilken opgave. Gruppen kunne så udover den planlagte arbejdstid, aftale om man ville arbejde videre senere på aften. Vi aftalte at de sidste to dage inden opgavens deadline skulle afsættes til oprydning af kode, CSS og rapportskrivning.

Vi måtte udvide vores database en enkelt gang med en udvidelse af admins mulighed for at tjekke done/undone af på ordrelisten.

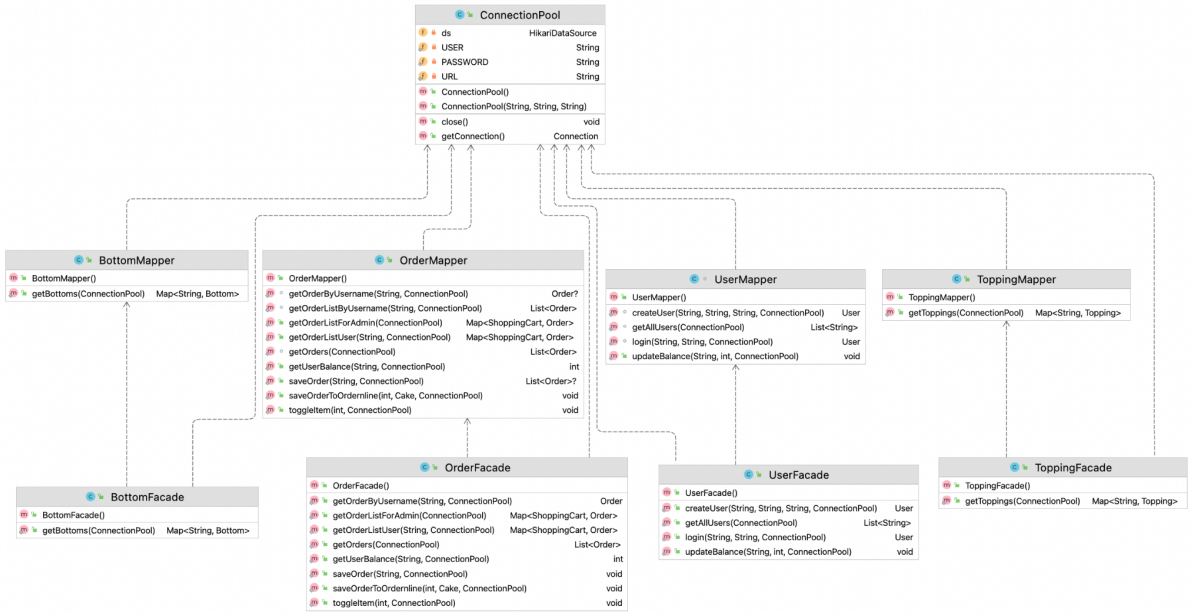
Hverdagene forløb meget som planlagt og var virkelig strukturerede. Vi var gode til ofte at tjekke ind hos hinanden i de forskellige undergrupper, vi fordelte os i, og vi har udelukkende arbejdet på branches, som vi ofte committede til med mere eller mindre meningsfulde navne. Vi merge ofte tilbage til main og sørgede for at det spillede sammen med det de andre havde lavet. Vi er løbet ind i forholdsvis få mergekonflikter, der har været ukomplicerede at løse.
Weekenden havde vi ikke fået drøftet ordentligt, og derfor røg strukturen lidt i disse dage og dermed også lidt af overblikket. Dette resulterede i at mandagen måtte bruges på et forsøg på at "redde trådene ud" og derfor er vi heller ikke kommet i mål med en ideel løsning på håndtering af ordrer, når de bliver trukket op af databasen igen - hverken på admin-siden eller kundesiden.

I fremtidige projekter har vi tænkt at bliver bedre til at projektstyre for at holde den røde tråd i projektet. Dette kunne være gennem bedre brug af kommentarer igennem koden samt mere logiske metodenavn. Derudover kunne code review være en fin metode, her er en kode først godkendt når en anden udvikler på projektet har kigget koden igennem og kan forstå den. Dette vil tvinge os til at rydde op i koden og sikre at det bliver mere forståeligt før det præsenteres for en kollega. Vi kan også her blive bedre til brug af trello til at holde styr på vores process samt hvad der er færdig og mangler at blive review. Vi vil også bruge mere energi i starten på at sikre at alle funktionerne er skitseret som vi ser de skal se ud på siden(mockup samt klassediagram) så vi er enige om fra start hvilke, samt hvordan funktionerne skal vises på de enkelte sider. Dette er i backend såvel som frontend. Dette skal selvfølgelig løbende videreudvikles, så det ikke er én model der laves fra starten og følges slavisk, men noget mere agilt der kan bruges til at holde styr på om vi overordnet har samme idé om hvordan systemet skal fungere, og hvor nogle specifikke metoder/attributter skal eller ikke skal (gen)bruges. Den sidste del vi har talt om er at vi skal arbejde på mindre dele af systemet af gangen, inden nogle andre kigger det igennem, så man kan nå at lave "damage control" - eller alternativt "go for it" - inden det påvirker for stor en del af systemet.

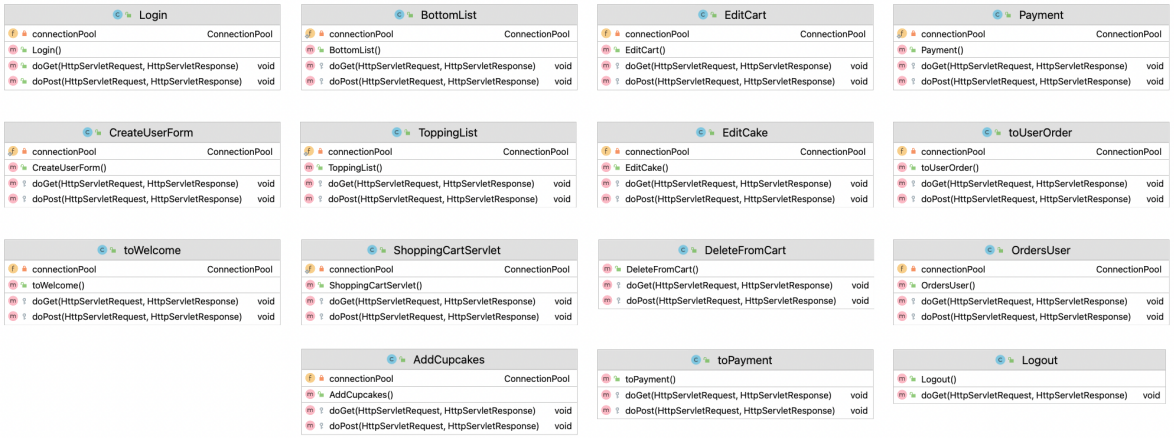
Klassediagrammer

Klassediagram: Entities package

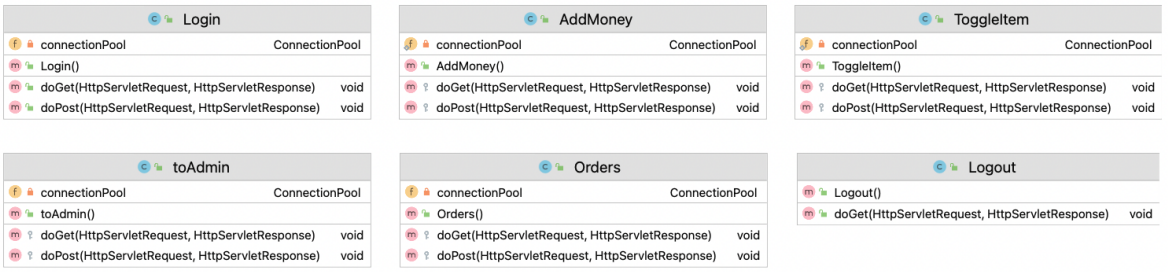
Klassediagram: Persistence package



Klassediagram: Control package - Servlets til User



Klassediagram: Control package - Servlets til Admin



Overblik over forbindelser i større del af klassediagrammet - uden metoder og attributter: