Cupcake rapport

Navn	CPH-mail	Github Navn
Matias Storck Rottbøll	cph-mr573@cphbusiness.dk	rotten99
Lucas Hellemose Hemmingsen	cph-lh464@cphbusiness.dk	LucasHemm
Marcus Sangill Hindsbøl	cph-mh1054@cphbusiness.dk	hindsMarcus

Kode Projekt

7-11-2022 t.o.m. 14-11-2022

Rapport

14-11-2022 t.o.m. 17-11-2022

Indholdsfortegnelse

Indledning

Dette projekt omhandler en cupcake hjemmeside, der er udarbejdet som et Minimum Viable Product. Hjemmesiden er designet til en virksomhed, med det formål at virke som en webbaseret bestillingsside. På hjemmesiden kan kunder oprette sig og lægge bestillinger på cupcakes, samt få et overblik over tidligere bestillinger. Hjemmesiden er også designet med henblik på, at man som admin kan skabe sig et overblik over alle kunder og ordrer.

Demo video af projekt

Link: https://youtu.be/qykPIT9BH7Q

Baggrund og Kravspecifikationer

Baggrund

Olsker Cupcakes har stillet os en opgave, hvor vi skal udarbejde en brugbar hjemmeside til dem, hvorpå de kan modtage bestillinger til deres butik. De er et nyt firma fra Bornholm, som har fundet den helt rigtige opskrift.
Hos Olsker Cupcakes vil de gerne have leveret en hjemmeside, hvor en ny kunde kan oprette en bruger, hvorefter brugeren skal have diverse muligheder (Se kravspecifikationer).

Kravspecifikationer

I følgende afsnit vises der en oversigt over de krav, som vores kunde har stillet til webapplikationen. Kravene er opdelt ift. hvilken aktør, som den specifikke user story er tilknyttet, samt om den er blevet opfyldt eller ej. Vi har i det her tilfælde løst alle user stories, for både en *Kunde* samt *Admin*, som vises ved, at alle kravene har en grøn afmærkning i de venstre felter.

Kunde

US-1	Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, så 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 ordre.
US-3	Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.
US-4	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-5	Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.

Admin

US-1	Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer.
US-2	Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.
US-3	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-4	Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

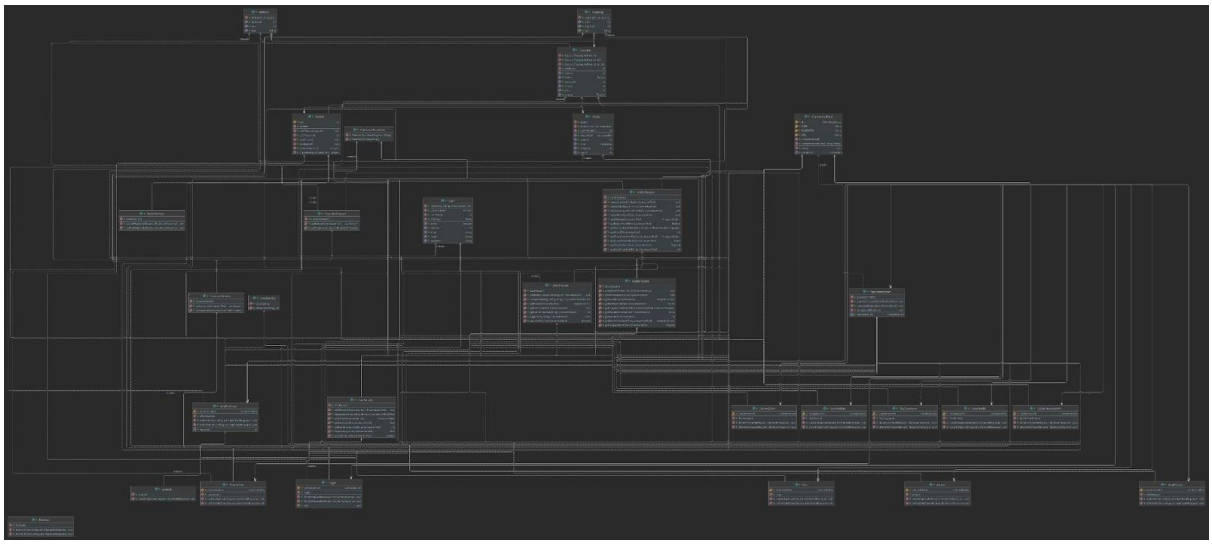
Teknologivalg

I dette projekt har vi benyttet os af følgende teknologier:

- Intellij 2021.2.4
- Apache Tomcat 9.0.69
- MySQL Workbench 8.0 CE
- coretto-11 Java version 11.0.16
- JDBC connector - mysql-connector-java:8.0.30

Klassediagram og ER diagram

Klassediagram



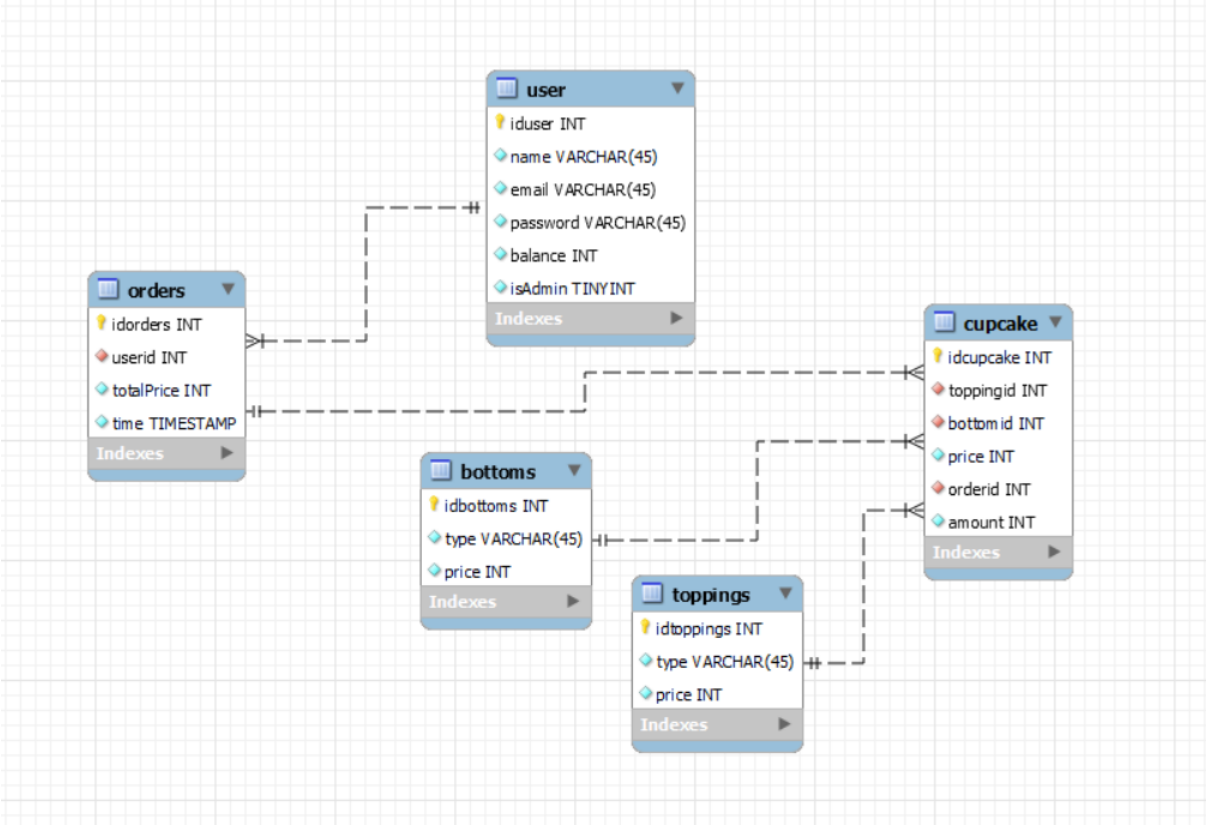
For at se hele diagrammet henvises der til Github projektet hvor der i dokumentation vil ligge et billede af klassediagrammet i fuld opløsning

Som der kan ses i diagrammet, bliver der fulgt en Model-view-controller arkitektur, som skiller projektet ind i 3 dele. Denne arkitektur bliver brugt for bedre at kunne strukturere hvordan projektet skal se ud. I forlængelse af den arkitektur, bliver der brugt et facadelag mellem mapper klasserne, som skaber et ekstra niveau af struktur.

Dette facadelag bliver brugt ift. Model delen af vores kode, som eks. vil være de forskellige entiteter som projektet er bygget op om eller data persistens. Det vil sige at hver gang der skal kaldes en metode, som har at gøre med persistens af vores data, vil man skulle gå igennem en facade for at komme til mapper klassen hvor metoden, som skal persistere dataen, vil blive kørt.

I vores controllag ligger vores servlets, som styrer hvordan brugeren kan bruge hjemmesiden. Det kan f.eks. være servletten 'ViewProfile' som bliver kaldt når brugeren skal se sin profil. Servletten vil så bruge OrderFacade fra Model laget til at hente brugerens tidligere ordre, så de kan blive vist på hans profil side.

EER diagram



Vores overvejelser med databasen startede med hvilke "entity"-klasser, vi ville få brug for for at få vores hjemmeside til at køre. Vi blev enige om, at vi ville få brug for en bruger og en bestilling bestående af cupcakes, som ville bestå af bunde og toppings. Det ville dog havde været hensigtsmæssigt at kalde cupcake-tabellen for "orderlines" i stedet for, da navnet overlapper med Databasens navn.

En bruger skal kunne have mange ordre, og der skal ikke kunne eksistere en ordre uden en bruger tilknyttet. Derfor har tabellen, orders, en foreign key på bruger-ID'et.

Ligeledes skal en ordre kunne have mange cupcakes, og en cupcake skal ikke være i databasen uden at tilhøre end ordre. Derfor er der en foreign key fra cupcake-tabellen mod ordre-ID'et. Da en bestilling af en cupcake vil bestå af en bund og en topping, skal der også være en foreign key forbindelsen fra cupcake til henholdsvis til bundens ID og toppings ID.

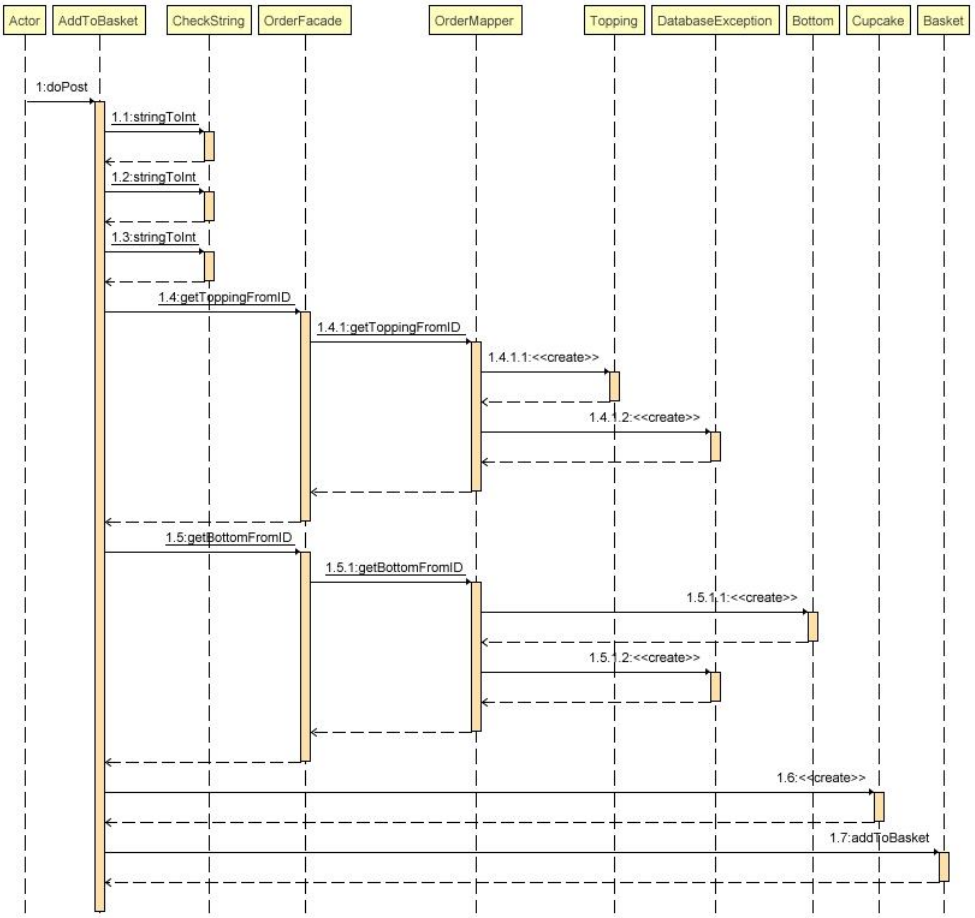
En sidste overvejelse, vi skulle gøre, var på hvilken måde brugerens rolle skulle implementeres på. Da der kun skulle være to roller på hjemmesiden, kunde og admin, men vi en boolean/TINYINT, kaldet isAdmin, var en bedre løsning end en streng, der beskriver rollen

Sekvensdiagrammer

AddToBasket

Dette sekvensdiagram gennemgår vores AddToBasket servlet.

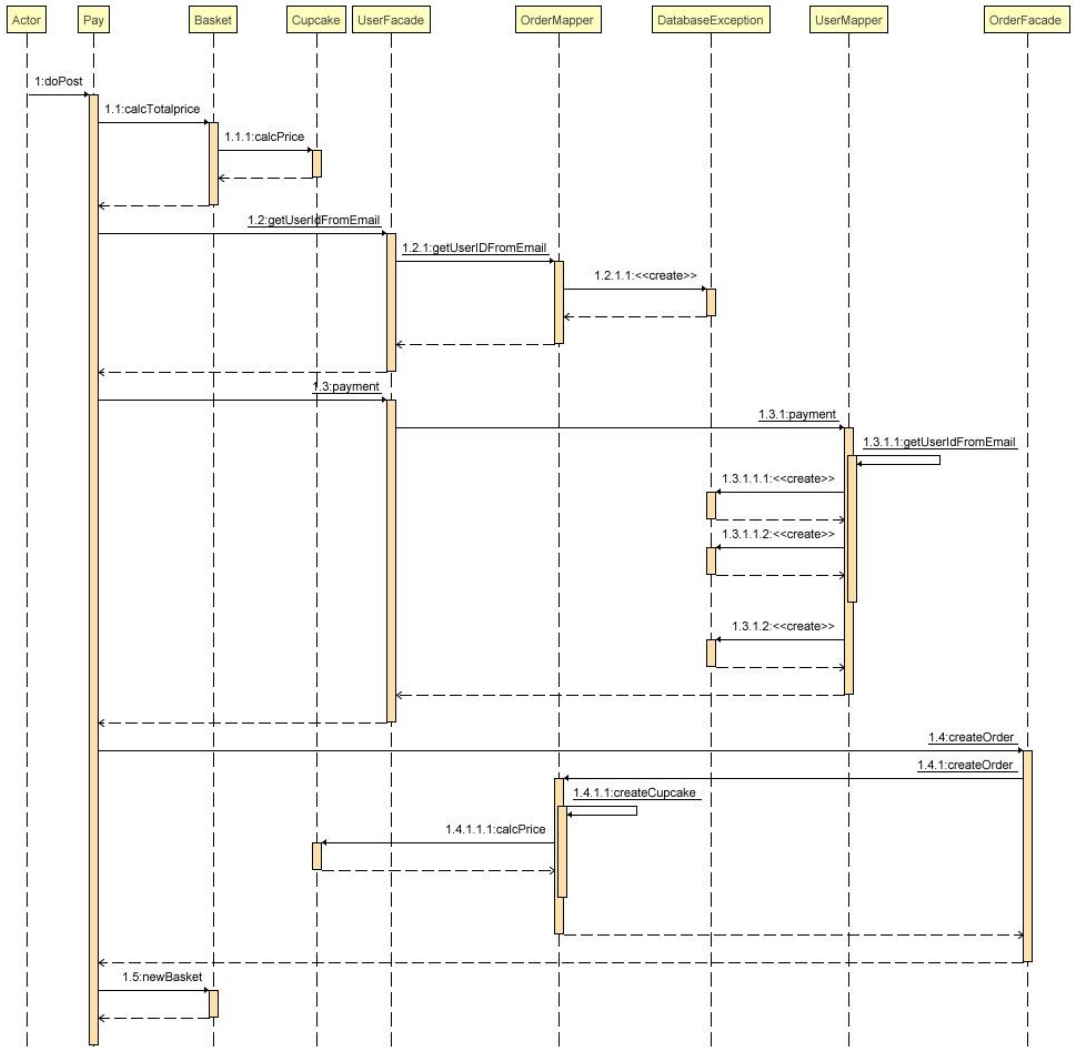
- Som det kan ses, bliver der først tjekket i checkString for brugerinput og valideret, at det er et korrekt input og tager højde for punktum.
- Derefter går det videre i ordefacaden og ordremapperen. Her hentes topping og bottom, som laves til entiteter
- Efter dette bliver der oprettet en cupcake, hvor topping- og bottom-entiteterne bliver brugt til dette. Når det er sket, bliver cupcaken lagt i kundens basket som ligger på Sessionscopet.



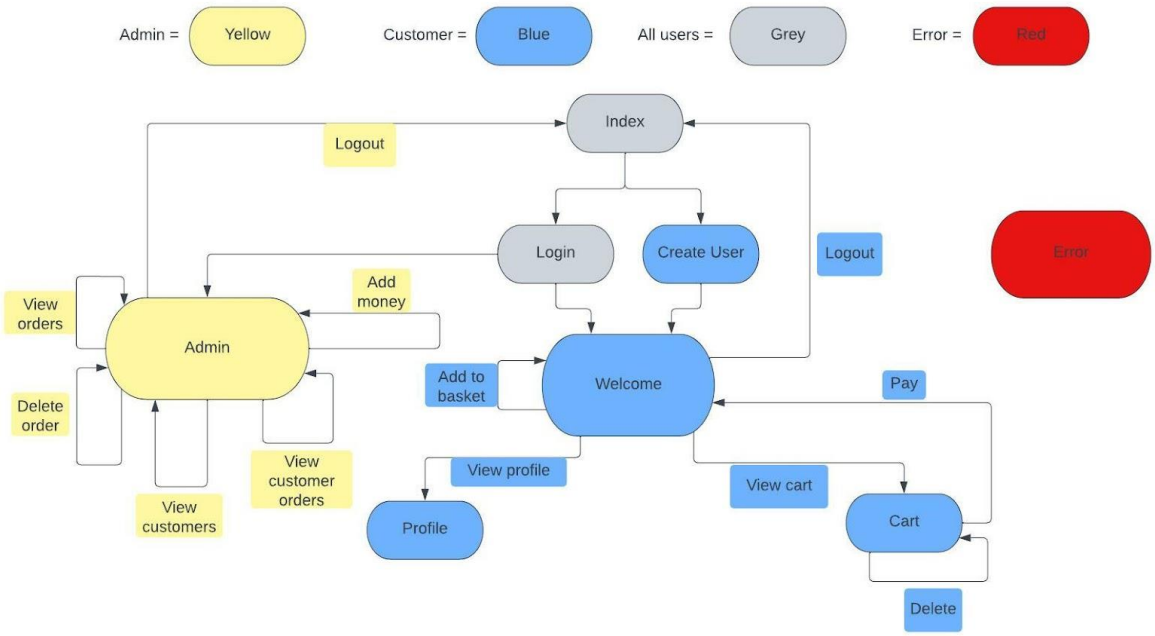
Pay

Dette sekvensdiagram gennemgår vores 'pay' servlet

- Først bliver den fulde pris fastlagt for ordren når brugeren har trykket pay inde i sin basket.
- Derefter bliver bruger ID hentet. Samtidigt bliver der tjekket at brugeren har de nødvendige penge til at lave bestillingen. Hvis brugeren har pengene, vil bestillingens pris blive trukket fra kundens saldo.
- Når ID'et så er hentet, bliver der først lavet en ordre entitet, som indsættes i databasen.
- Når ordren er blevet oprettet i DB, bliver ID'et på ordren returneret og brugt til at oprette de forskellige cupcakes som er bestilt, således at de hører til den ordre.
- Når dette er sket, vil der blive oprettet en ny basket som kunden kan bruge til lave en ny separat ordre hvis kunden ønsker det.



Navigationsdiagram



Navigationsdiagrammet giver et overblik over alle de måder en bruger vil kunne bevæge sig igennem systemet på. Diagrammet deler sig i to ved login, hvor der vil blive checket for om man er logget ind som en administrator eller en kunde. Alle de blå sider kan kun tilgås som kunde, hvor de gule kun kan tilgås hvis man er admin. Den røde Error side ligger for sig, da alle sider vil kunne føre dertil hvis der sker en fejl i systemet.

Særlige forhold

I en session vil der blive gemt to entiteter. Den første entitet er et user objekt, som bliver oprettet når brugeren vil logge ind. Her bliver brugerens data hentet fra databasen og puttet ind i entiteten. Den anden entitet er et basket objekt som også ligger på session scopet. Det valgte vi at gøre da en basket på hjemmesiden kun skulle fungere for brugeren mens de var logget ind for at lægge en bestilling. Og hvis brugeren så ikke betaler for bestillingen, vil sessionen udgå og dermed også deres basket.

Vores håndtering af brugerinput med tal, såsom startsaldo, indsæt penge eller antal cupcakes, bliver valideret med html input typen “number” hvorefter vi også tager højde for punktum i inputtet.

Exceptions bliver håndteret ved at brugeren vil blive sendt til en error page, hvor de vil få skrevet en besked som bedst beskriver problemet de er ramt ind i.

I databasen har vi to forskellige brugertyper, en kunde og en admin. Det bliver styret ved en boolean. Når denne boolean er true vil brugeren efter login i stedet blive sendt til admin siden og ikke den almindelige kunde side.

Status på implementering

Alle user stories er blevet implementeret fuldt ud. Al planlagt funktionalitet fungerer som det skal, og alle bugs er blevet rettet og fixet så programmet virker hensigtsmæssigt. Stylingen af webshoppen er ikke blevet fuldendt til hvad der var planlagt, men den overordnede struktur af hvordan siderne skal se ud er blevet implementeret.

Proces

Vi dannede os et overblik over hvordan vores EER-diagram skulle struktureres samt hvilke entiteter, som skulle indgå i vores kode. Vi fastlagde en plan for gruppens mødetider og fordelte diverse opgaver ud på dagene i ugen.
Vi fulgte planen og pga. en god strukturering samt effektiv arbejdsindsats, fik vi fuldt implementeret alle User Stories. Vi stødte selvfølgelig på nogle problemer, som vi ikke havde regnet med ville opstå, men gennem konstruktive overvejelser i gruppen fandt vi frem til fyldestgørende løsninger.
Vi har erfaret, at det at strukturere projektets forløb har hjulpet os gevaldigt med at fastholde overblikket over processen, men også at man ikke kan planlægge og forudse alle forhindringer på forhånd.