Cupcake

Christian Madsen, Jacob Simonsen & Jonas Jørgensen
D. Klassen

	Email		Github
Christian:	cph-cm298@cphbusiness.dk	//	CMadsen99
Jacob:	cph-js441@cphbusiness.dk	//	jacmac2812
Jonas:	cph-jj467@cphbusiness.dk	//	worldjones

30/3/2020

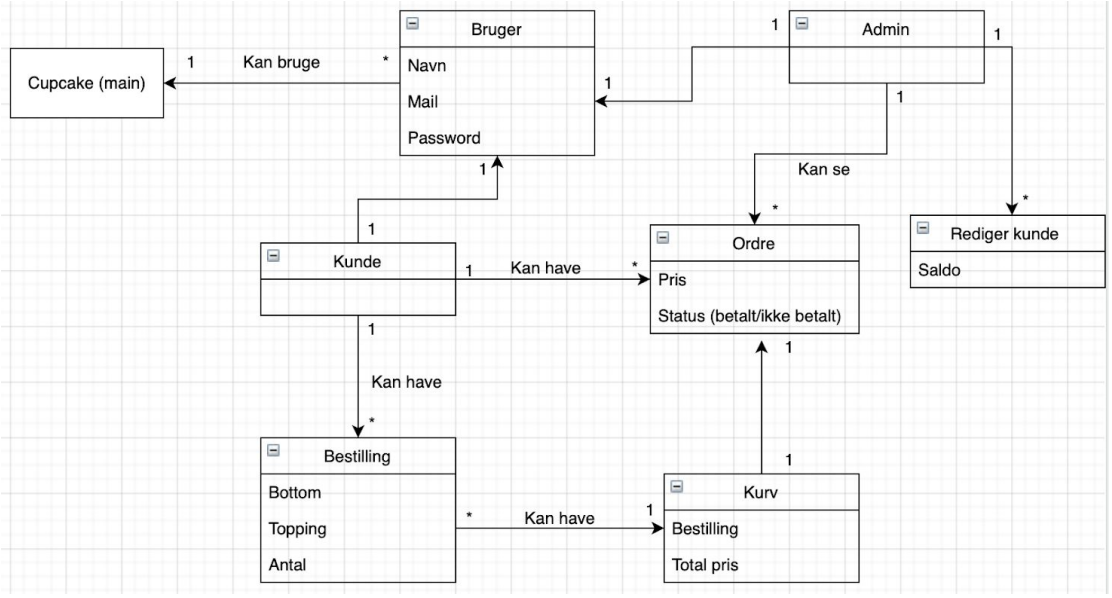
Indledning

Vi har fået til opgave at lave et projekt for virksomheden “Olsker Cupcakes”. Vores projekt er en hjemmeside, hvorledes en kunde kan bestille, og derefter afhente cupcakes hos virksomheden.
Kunden skal kunne bestille en eller flere cupcakes med valgfri bund og top. Kunden skal kunne oprette en konto, som giver dem mulighed for at betale og gemme deres ordre. Kunden skal derfor også have adgang til en indkøbskurv, hvor de kan se den samlede pris, eller redigere deres køb. Systemet skal også indeholde administrator brugere, som kan indsætte et beløb på en kundes konto, så kunden kan betale. Administratoren skal kunne se alle ordrer i systemet, samt redigerer i dem, hvis en kunde f.eks. ikke har betalt. Administratoren skal også kunne se alle kunder i systemet samt deres ordrer, så de kan holde styr på hver enkelt kunde. Enhver bruger skal kunne logge på med email og kodeord. Denne email er vist i navigationsbaren på toppen af siden, så det kan ses at man er logget ind.

Til projektet har vi brugt:

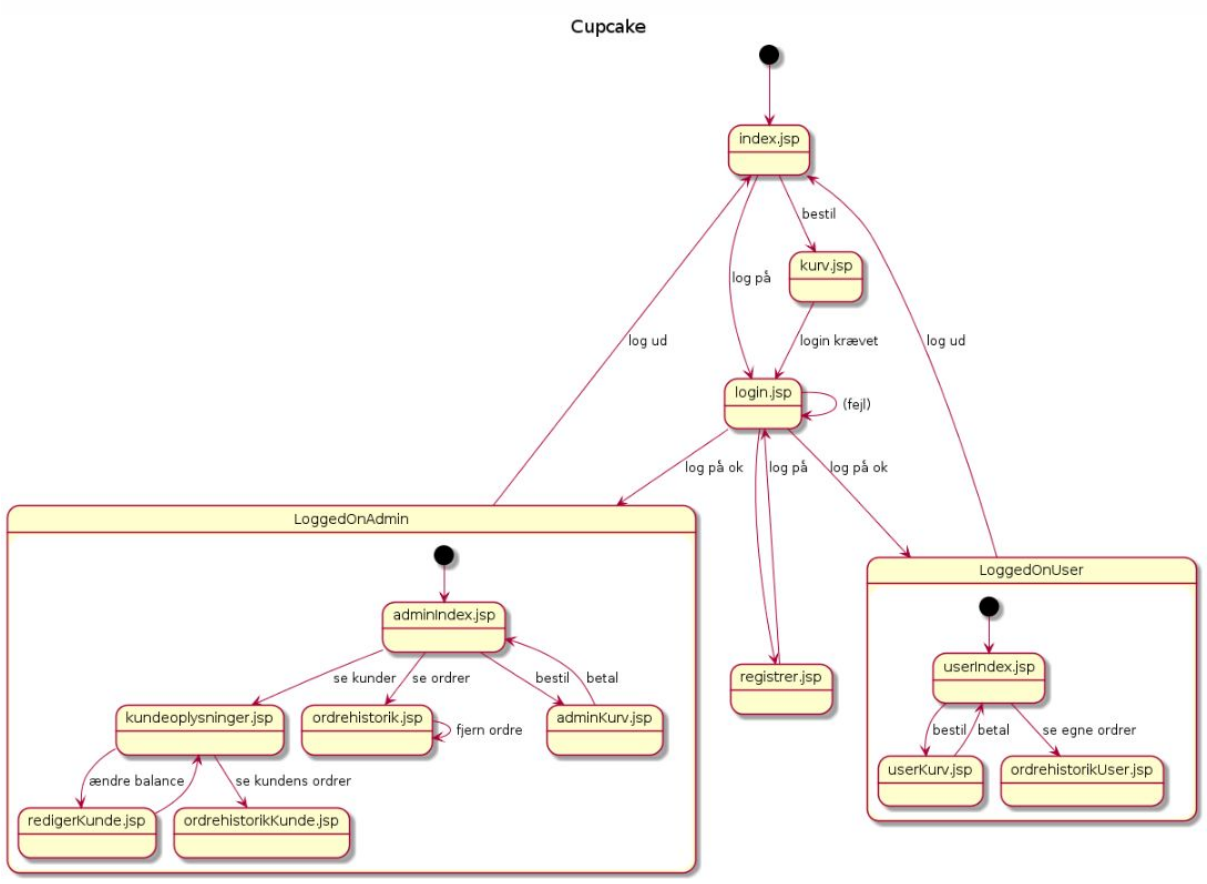
IntelliJ - 2019:3.3 x64(Ultimate)
MySQL Workbench - 8.0 CE
Tomcat - 8.5

Domæne model



For at lave en bruger skal man udfylde, navn, mail og password. Som man kan se på domænemodellen, kan brugeren enten være kunde eller admin, man kan dog ikke oprette sig som admin inde på siden, ellers kunne alle være admins på hjemmesiden.
Hvis man kigger på kunden kan man oprette flere bestillinger i samme kurv, hvor bestillingen indeholder Bottoms, Toppings, samt antal. Når bestillingen kommer ind i kurven bliver den gemt som en ordre med en total pris, som både kan ses af kunden selv, men også af adminen. Desuden kan man også kan se status på, om ordren er blevet betalt. Udover at adminen kan se ordrerne, som kunderne opretter, kan adminen også redigere i kundernes saldoer, så de kan betale for fremtidige ordre inde på hjemmesiden.

Tilstandsdiagram / Navigationsdiagram



Når brugeren åbner index siden, vil de altid skulle logge ind for at bruge funktionerne. Det er dog muligt at tilgå kurven med det samme, men man vil blive sendt tilbage til login siden alligevel, hvis man prøver at betale, for det man har lagt i kurven. Login siden giver også mulighed for at gå til registrer siden, hvis man ikke allerede er oprettet i systemet som bruger. Angiver man de forkerte login-oplysninger, bliver man sendt tilbage til login-siden og kan prøve igen.

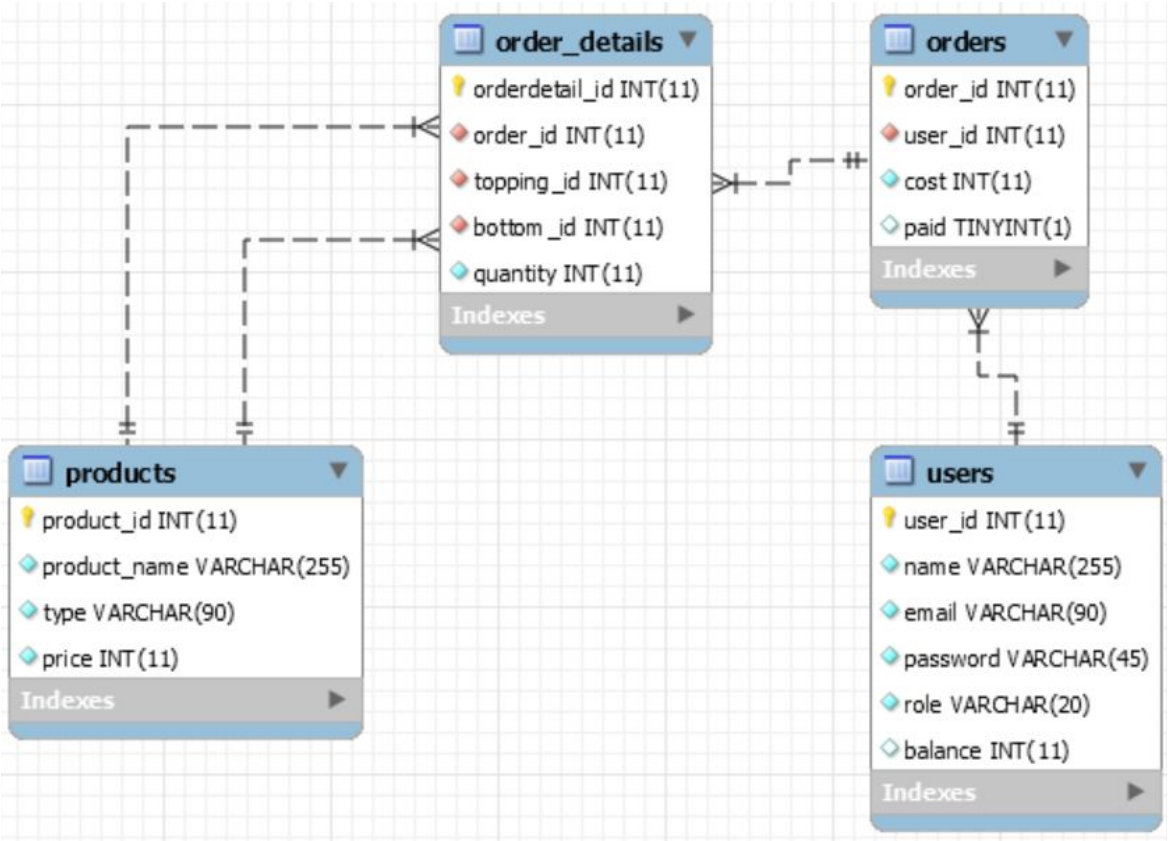
Projektet har forskellige sider afhængig af brugerens status (admin eller kunde). Alt efter om man er admin eller kunde, vil der være to forskellige navigationsbarer, som er tilpasset til den status man har som bruger.

Som admin kan man fra sin index-side tilgå en ordrehistorik af alle kundernes ordre. Man kan tilgå en side med oplysninger om de forskellige kunder i systemet, hvor man desuden også kan se de enkelte kunders ordrehistorik samt ændre i kundernes balance. Admin kan også selv bestille cupcakes og se kurven, for at kunne teste om hjemmesiden og databasen virker korrekt.

Som kunde kan man fra sin index-side også tilgå en ordrehistorik af sine egne tidligere ordre. Denne side er tom, hvis man ikke har bestilt før. Hovedformålet for kunderne er at kunne bestille og betale cupcakes, som kan gøres henholdsvis på index-siden og inde i kurven.

Begge typer af brugere vil blive sendt tilbage til index så snart de logger ud.

EER-diagram



Databasen indeholder fire tabeller:
En til brugernes informationer, en til ordrerne, en til detaljer om ordrerne og til produkterne.

Bruger-tabellen indeholder et autogenereret bruger id, som bliver tildelt brugeren når de opretter sig inde på hjemmesiden. Brugeren bliver oprettet med en email som er unik, et brugernavn, og et password. Brugeren bliver som default sat til at være en “customer” i “role” kolonnen, og starter også som default med at have en balance på 0 kr.

Ordre-tabellen indeholder et autogenereret ordre id, som bliver tildelt når ordren oprettes. Derudover viser tabellen hvilken bruger, der har bestilt den pågældende ordre. Bruger id'et er derfor en fremmednøgle med constraint til Bruger-tabellen, hvilket gør at der ikke kan oprettes en ordre, hvis ikke brugeren allerede findes i Bruger-tabellen.
Tabellen indeholder ordrens samlede pris, som bliver udregnet direkte inde på hjemmesiden, men som ellers kunne have været udregnet med SQL-queries.
Den sidste kolonne i tabellen er for at vise om den pågældende ordre er blevet betalt eller ej, da ordren blev oprettet. Kolonnen er oprettet som en boolean, men inde på MySQL bliver boolean lavet om til tinyInt, og så tjekker man for true/false, ved at se om tallet er 0 eller 1.

Orderdetails-tabellen indeholder et autogenereret id, for at de forskellige linjer kan skelnes fra hinanden, hvis man skal bruge tabellens informationer. Derudover indeholder den tre fremmednøgler med constraints, og til sidst et antal, som viser hvor mange af den pågældende cupcake, der er blevet bestilt. Den første fremmednøgle går hen til ordre-tabellens ordre id, da der er nødt til at være oprettet en ordre før man kan få nogle detaljer om den. De to andre fremmednøgler går til produkt tabellen, da den indeholder både toppings og bottoms, og en cupcake består af begge to, så derfor kan ordren ikke bestilles uden produkterne findes i produkt-tabellen.

Den sidste tabel, produkt-tabellen, indeholder al information om de produkter, der bliver solgt på hjemmesiden. Hvert produkt har et autogenereret produkt id, navnet på produktet, prisen på produktet og til sidst hvilket type produkt det er. Om det er en topping eller bottom. Hvilket er grunden til at der går to constraints til orderdetails-tabellen. Det kunne måske være undgået ved at lave en tabel mere, som hed "Product_type", som bare indeholdte topping og bottom, og som produkterne så kunne refereres til.
Vi har gjort det på den måde som vi har, fordi vi synes det var smartest til at starte med, men kan godt se nu, at den anden løsning måske havde været bedre.

Usecases:

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 mine ordre.

US-2: Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme 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 mune valgte ordrelinjer i en indkøbskurv, så jeg 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).

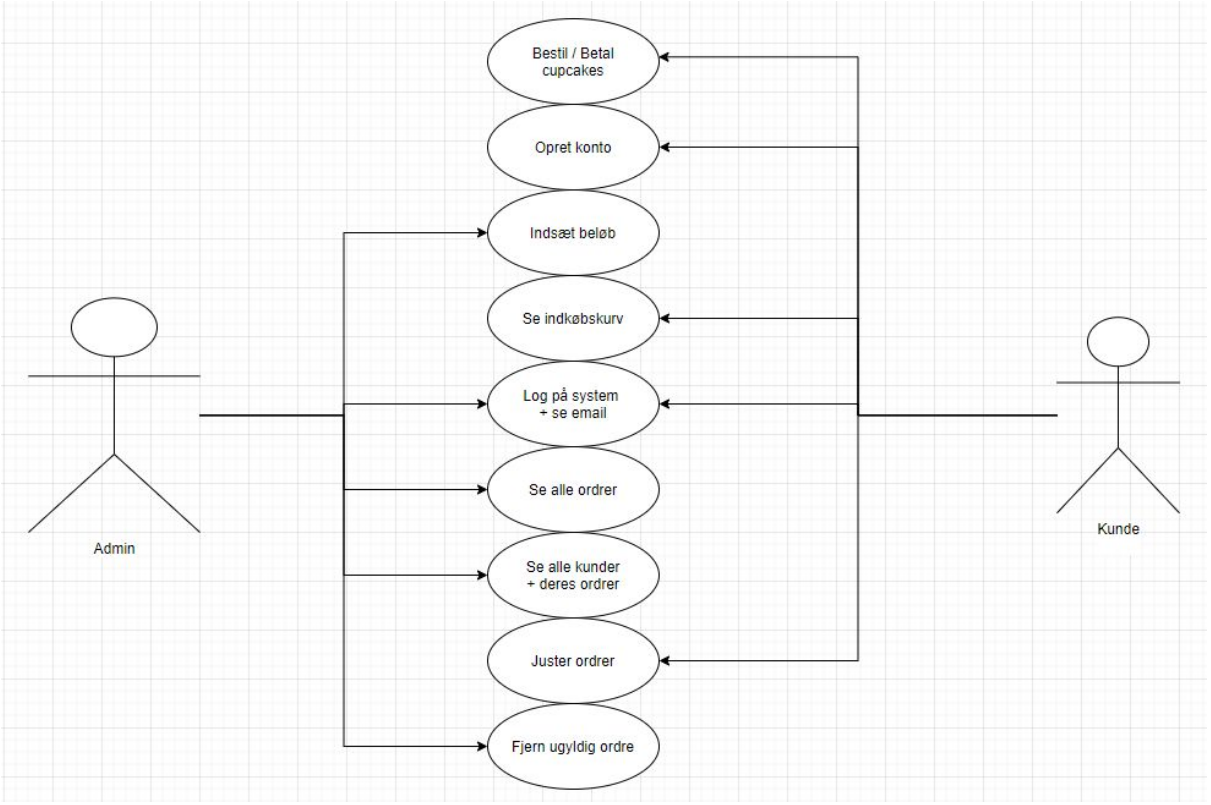
US-6:Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

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 mine ordre.

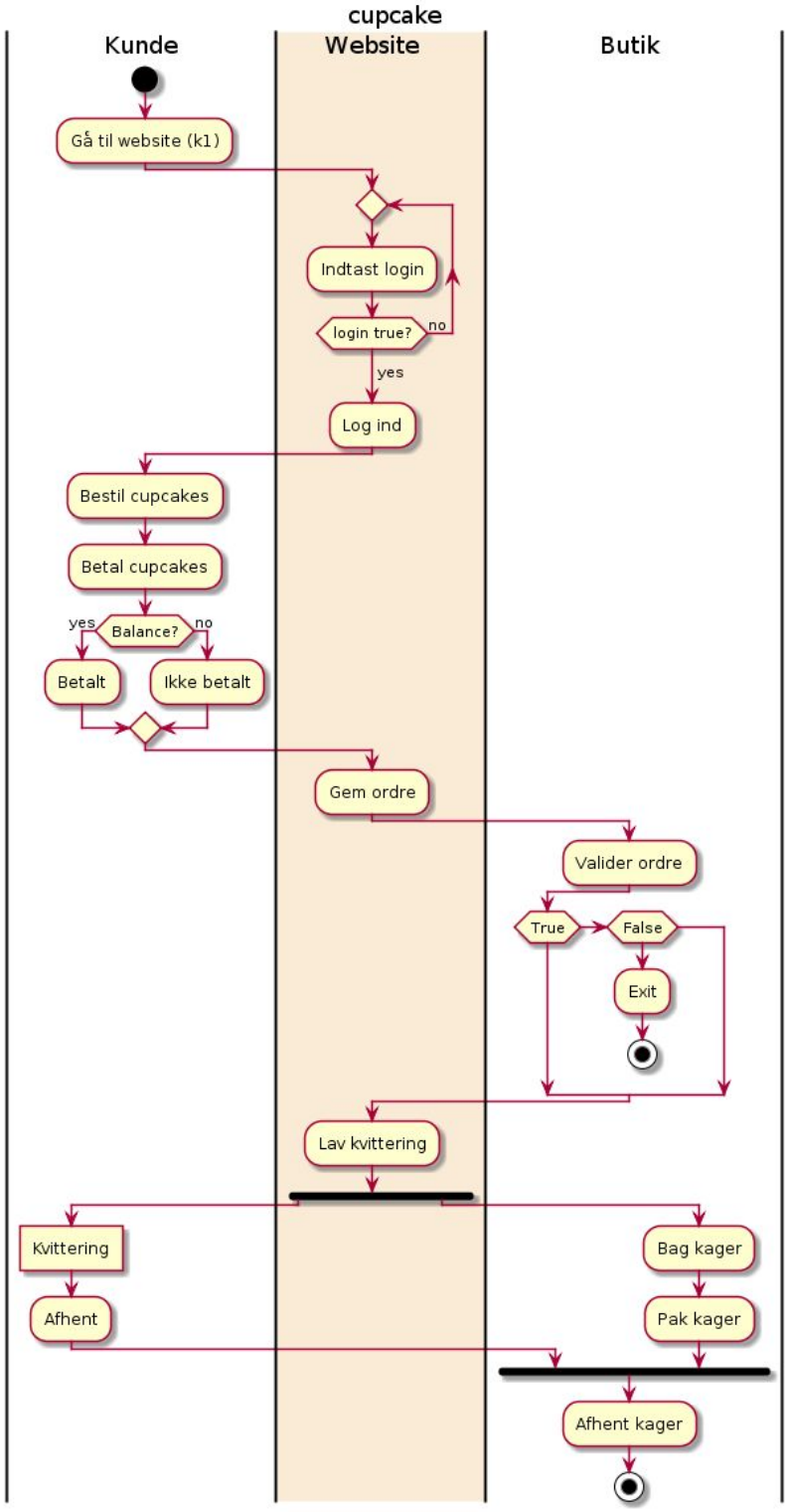
US-9: Som administrator kan jeg fjerne en ordre så systemet ikke kommer til at indeholde ugyldige ordrer f.eks. hvis kunden aldrig har betalt.

UseCase-diagram



Ud fra de givne use cases kan vi se at en normal bruger skal kunne lave en konto, bestille og oprette cupcakes og se deres indkøbskurv. Her skal de kunne redigere i deres bestilling. Både admin og bruger kan logge på og se at de er logget ind med e-mail på navigationsbaren.
En admin skal ud over de normale funktioner også kunne se og justerer alle kunder og deres ordre. De skal også have muligheden for at indsætte et beløb.
Adminen kan teknisk set også bestille cupcakes og se/justerer dem i deres indkøbskurv, men dette er ikke super relevant i praksis, da administratorerne ikke har brug for disse funktioner.

Aktivitetsdiagram

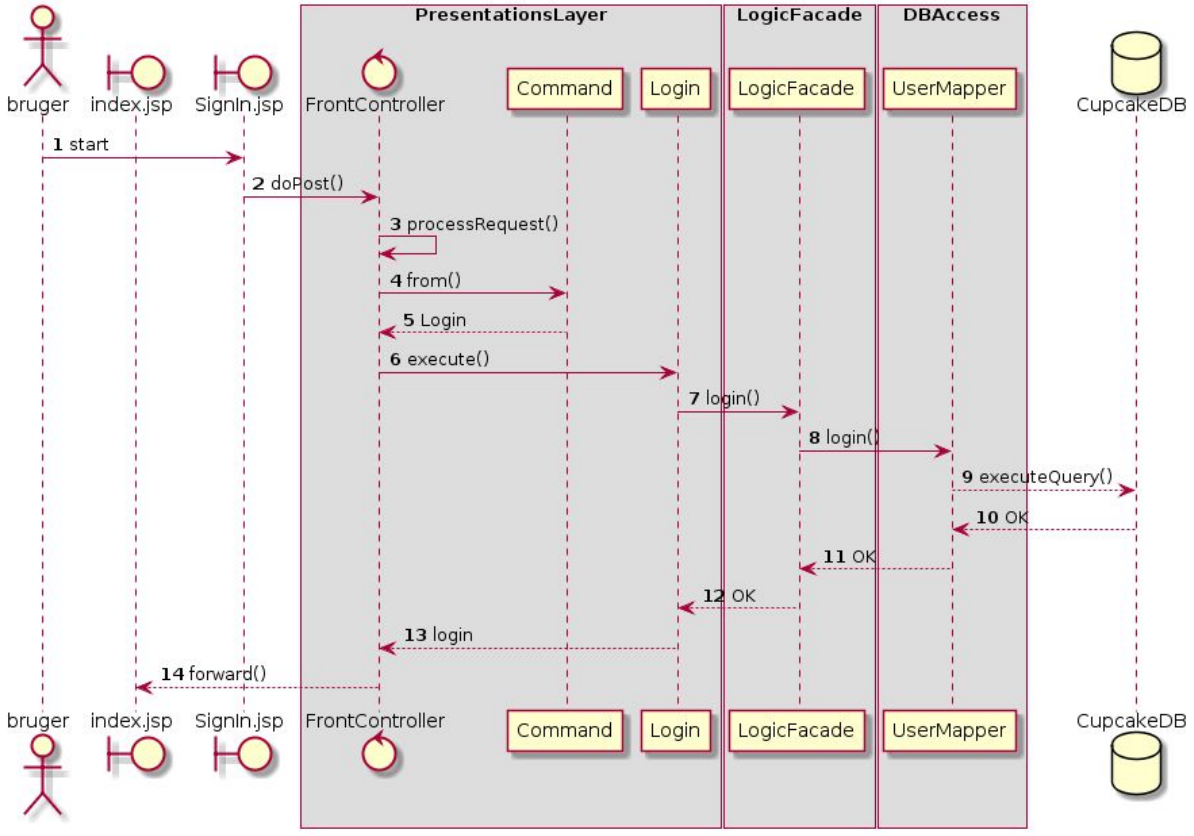


Som vi kan se kan man som bruger først bestille en cupcake efter man er logget ind. Hvis man ikke skriver et korrekt login, bliver man sendt tilbage og får lov til at prøve igen. Bliver det godkendt, kan man bestille sine cupcakes.

Kunden kan efterfølgende betale for deres cupcakes, hvis de har penge nok. Kundens balance bliver tjekket når de prøver at betale, og hvis balancen er højere end ordrens pris, bliver bestillingen godkendt og gemt som at være betalt. Er balancen lavere end ordrens pris, bliver bestillingen gemt som en kladde, og kan eventuelt betales senere. Ordrerne bliver gemt ligemeget hvad, men hvis bestillingen er gemt som kladde, kan en administrator bestemme om den givne bestilling skal valideres eller ej. Bliver den valideret, laves der en kvittering, som kunden får. Bestillingen bliver efterfølgende sat i produktion og pakket, hvor kunden derefter kan afhente den i butikken.

Sekvensdiagrammer

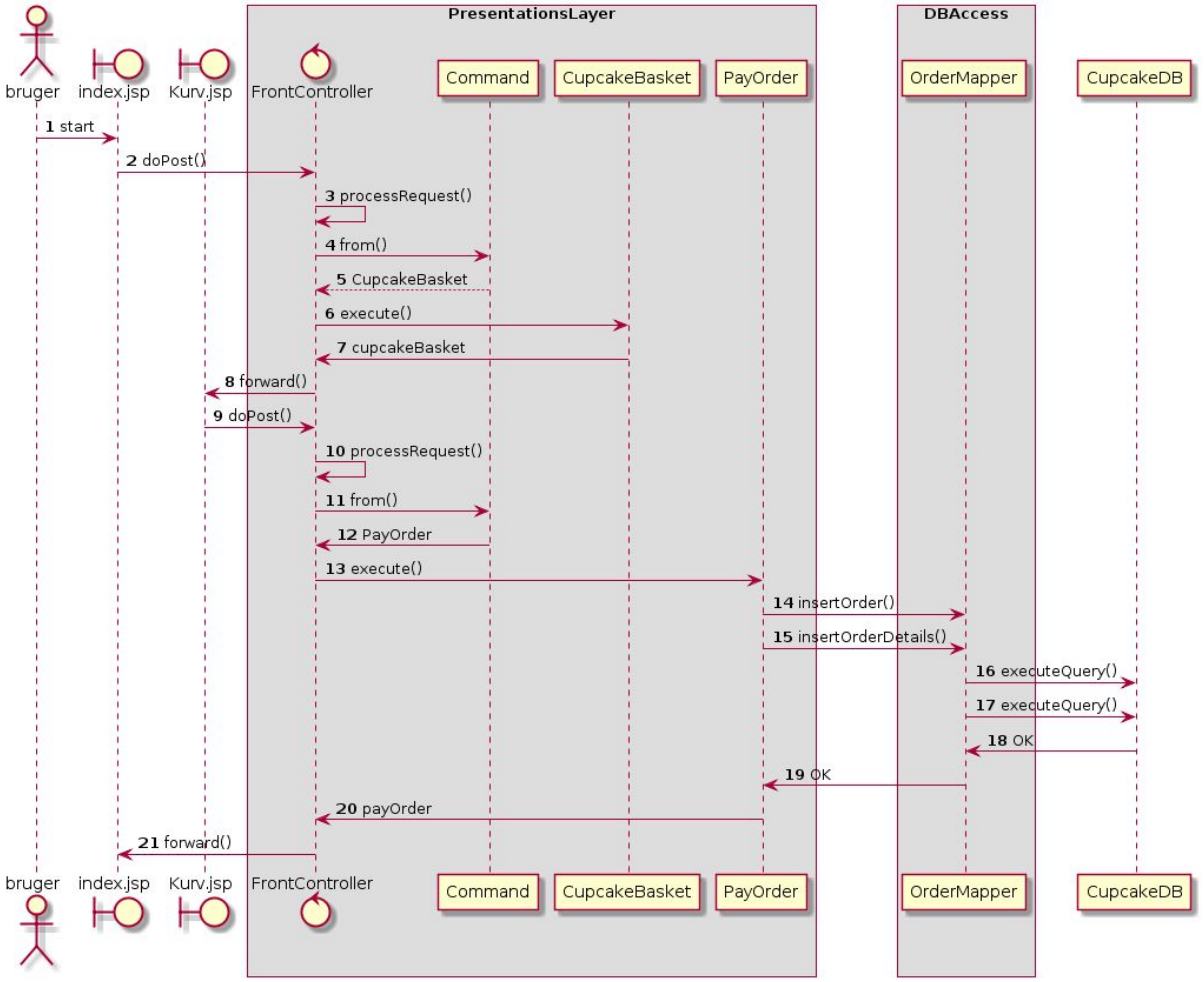
SekvensLogin



I det første step kommer brugeren ind på signin.jsp og indtaster sine login oplysninger. Oplysningerne bliver sendt til FrontControlleren, som processRequest metoden tjekker. Derefter kommer det igennem Command, som finder ud af hvilken metode der passer til den efterspurgte handling. Det bliver sendt tilbage til FrontControlleren, som sender den javakode Command har fundet frem til videre til LoginFacade. Derfra hopper den videre til UserMapperen, som tjekker informationerne nede i CupcakeDB.

Hvis det hele passer går den tilbage gennem alle steppene, for at brugeren til sidst er logget ind på hjemmesiden og føres hen til index.jsp via forward metoden.

SekvensKurv



I det første step kommer brugeren ind på index.jsp hvor kunden kan vælge sine cupcakes. De valgte oplysningerne bliver sendt til FrontControlleren, som processRequest metoden tjekker. Derefter kommer det igennem Command, som finder ud af hvilken metode der passer til den efterspurgte handling. Det bliver sendt tilbage til FrontControlleren, som sender den javakode Command har fundet frem til videre til CupcakeBasket, som sender det tilbage igen, hvorefter de valgte cupcakes bliver vist på kurv.jsp via forward metoden. Her kan brugeren vælge at betale for bestillingen i kurven. Det bliver sendt til FrontControlleren, som processRequest metoden tjekker. Derefter kommer det igennem Command, som finder ud af hvilken metode der passer til den efterspurgte handling. Det bliver sendt tilbage til FrontControlleren, som sender den javakode Command har fundet frem til videre til PayOrder, som i step 14 og 15 sender informationerne til OrderMapperen. Den udfører SQL-querierne og indsætter ordren i CupcakeDB.

Hvis alt går som planlagt, bliver man ført tilbage til index.jsp, hvor PayOrder skriver en meddelelse om at ordren er betalt.

Særlige forhold

Brugerens login oplysninger er gemt i session. Kurven bliver også gemt her, men når man bruger log ud knappen bliver ens session invalideret. Ud over det bliver kurven tømt når man har betalt.

Vi har lavet en exception, som giver en fejlmeddelelse hvis man prøver at bestille en cupcake og glemmer enten topping, bottom eller antal.

I forhold til login sikkerhed, så bliver ens password-input lavet til dots, så andre ikke kan se hvad man skriver.

Status på implementation

Vi har gennemført alle usecases, dog er det stadig muligt at lave nogle forbedringer forskellige steder i projektet.

Hvis man laver en blank bestilling skriver programmet en fejl, som vi ikke har lavet exception handling på.

Man kan ikke se hvilke typer cupcakes der er bestilt i en given ordre, når man kigger på ordre-historikken. Vi kunne godt have tænkt os at lave en form for dropdown, på hver ordre, som ville vise den ordres detaljer.

På afleveringstidspunktet har vi ikke fået projektet til at virke på vores Digital Ocean server, men localhost virker fint. Vi håber at kunne nå det inden eksamenstidspunktet.

Vi mangler at style den side hvor man kan ændre kundens balance.