

Cupcake

2. Sem
DAT A
30/3/2023

Bastian Holm

✉ cph-bh213@cphbusiness.dk
👤 Bastian-cph

Sander Roust

✉ cph-sr319@cphbusiness.dk
👤 snadering

Tobias Berg

✉ cph-tb266@cphbusiness.dk
👤 TobiasBergg

Tobias Christiansen

✉ cph-tc184@cphbusiness.dk
👤 TobiasTheDanish



OLSKER
CUPCAKES
MUMS FILIBABA



Indholdsfortegnelse

Indledning	1
Baggrund.....	1
Teknologivalg.....	1
Teknologi	1
Software	1
Krav.....	2
Olsker Cupcakes vision	2
User stories	2
ER-diagram	3
Navigationsdiagram.....	4
Særlige forhold	5
Status på implementationen.....	5
Figma	6
CRUD	7
Fejl på målstregen.....	7
Proces	8
Trello	8
GitHub	9
Konklusion	9
Bilag	11
Bilag 1	11
Bilag 2	11
Bilag 3.....	12
Bilag 4.....	12
Bilag 5.....	13
.....	13
Bilag 6.....	14

Produkt: <https://youtu.be/twVfGxDS-gA>

Indledning

Cupcake-projektet omhandler en nystartet iværksættervirksomhed, Olsker Cupcakes, der gerne vil sælge cupcakes online som take-away. Rapporten tager udgangspunkt i en datamatikerstuderende på 2. semester som fagfælle. Virksomheden indleder med en håndfuld user stories, som de ønsker opfyldt ved det færdige produkt.

Baggrund

Den kunde Olskers Cupcakes prøver at komme ud til er alt fra den enkelte person med en sød tand til den store familie, som har behov for noget lækkert til fredagshyggen, eller den store fest som skal have en stor portion med flere forskellige varianter. De egner sig godt til mange forskellige kunder, da det er et enkelt system at bruge samtidigt med at de forskellige knapper på siden er meget intuitive.

Det skal kunne være en ung som gammel, der kan bestille uden større besvær. På siden vil man kunne skabe cupcakes helt som man selv ønsker det, med lige den bund og top som man ønsker.

Teknologivalg

Teknologi

- Java 19
- JDBC (mysql-connector-java 8.0.30)
- JSTL 1.2
- Bootstrap 5.1.3
- HikariCP 4.0.3
- slf4j 1.7.36

Software

- IntelliJ 2021.3
- MySQL 8.0
- Tomcat 9.0.73

Krav

Olsker Cupcakes' vision

Olsker Cupcakes ønsker en funktionel og flot hjemmeside, som skaber en nem og hurtig kunderejse for deres kunder. De ønsker at bruge mindre tid på at tage imod ordrer på telefonen, men i stedet at bruge deres tid på at lave flere cupcakes og yde en ekstra god service i butikken. Derudover ønsker de data på hvad de sælger mest og mindst af, så de kan justere ud fra kundernes ønsker.

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 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 e-mail 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.

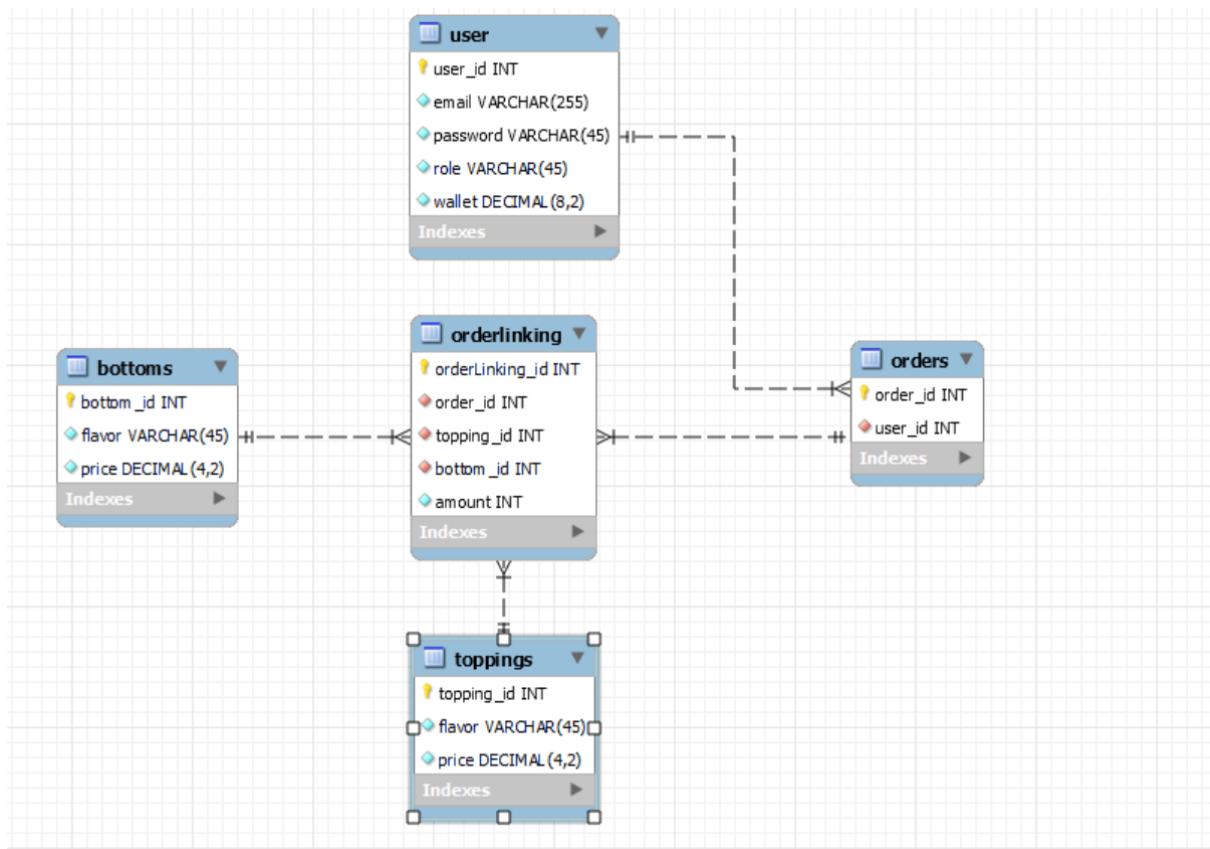
ER-diagram

Vores ER-diagram er struktureret, som vist på figur 1, med fem tabeller. Alle tabeller har fået en-til-mange-relationer. Man kan blandt andet kigge på vores relation mellem *orders* og *user*. Én kunde vil altså kunne have mange ordrer. Alle kolonner i tabellerne er sat til ‘not null’ hvilket betyder, der skal være en værdi for at undgå fejl i vores program. Det er vigtigt at nævne, at vi har valgt at bruge en normaliseringssproces for at undgå redundans i vores database og sikre nøjagtigheden af vores data. Dette har resulteret i en database i 3. normalform, som sikrer at hver tabel kun indeholder unikke og relevante data. Dette er en god praksis for at opnå optimal databaseydelse og for at undgå problemer med inkonsistens i data.

Vi har valgt at lave en separat tabel til *toppings* og *bottoms*, så man senere vil kunne tilføje nye smage lettere. Ydermere har vi lavet en *linktabel* til de to, så hver ordre kan indeholde mere end blot en slags cupcake. At vi har valgt at strukturere det på denne måde gør også at det vil være lettere at udvide databasen hvis Olskers cupcakes pludselig skulle få lyst til at udvide til andre produkter på et tidspunkt.

I *user-tabellen* har vi valgt at *user_id* er auto incremental for at sikre, at hver user der bliver lavet, har en unik identifikator. Det vil gøre det lettere at gemme data om både kunden, deres bestillinger og deres tidligere ordrer. Det vil også gøre det lettere at gå ned i koden og se hvor fejlen opstår. Den samme fremgang har vi lavet på vores *orders* tabel så hver ordrer kommer til at være unik.

Hvis man tager et kig på hvilke datatyper der er blevet brugt i databasen, så kan man se at alle vores auto incrementerede data er blevet tildelt *int*. *Int* er smart i denne sammenhæng da det er en af de mest effektive datatyper til at håndtere store datasæt på en hurtig måde, uden at bruge en masse hukommelse. Ydermere har *int* en rækkevidde der strækker sig over 2 milliarder, hvilket bør være nok i denne her type database. På price kolonnerne under både *toppings* og *bottoms* er der blevet brugt datatypen *decimal* og ikke *float*. Grunden til valget er præcisionen hvor *decimal* har en højere præcision. Derudover behøves der ikke mere end to decimaler når der skal laves en pengetransaktion som i en butik eller i dette tilfælde en webshop. Som man kan se, har vi valgt et højere tal i vores *wallet* da det skal være muligt at købe for et højere beløb.



Figur 1

Navigationsdiagram

I bilag 5 og bilag 6 ses to navigationsdiagrammer, som beskriver brugernes vej rundt på hjemmesiden. Navigationsdiagrammet er todeelt, da der på nuværende tidspunkt er to forskellige user-roller, henholdsvis ‘Admin’ og ‘User’.

Brugerens rolle bestemmer, hvor brugeren kan navigere hen på hjemmesiden. En ‘admin’ bruger har flere funktioner tilgængelig end en ‘user’ bruger har. Dette indebærer f.eks. muligheden for at se alle registrerede brugere, samt ændring af en brugers konto.

Vi benytter en navigationsbar som er fælles for alle sider. Her er der også forskel på hvad brugeren ser, alt efter brugerens rolle. Er brugeren ‘admin’ vil de have et link ‘Admin’, som tager brugeren til en side som kun admin brugere skal have adgang til.

Når en bruger trykker på en knap eller et link, hvor der skal skiftes til en ny JSP-side, sendes der først en *request* til en *servlet*, som derefter forwarder brugeren til den rigtige adresse. Der er dog en undtagelse, som er, hvis brugeren ønsker at registrere sig i systemet. Her sendes brugeren direkte til registreringssiden, da brugeren endnu ikke er logget ind, og vi derfor ikke har behov for at sende en request til en servlet.

Særlige forhold

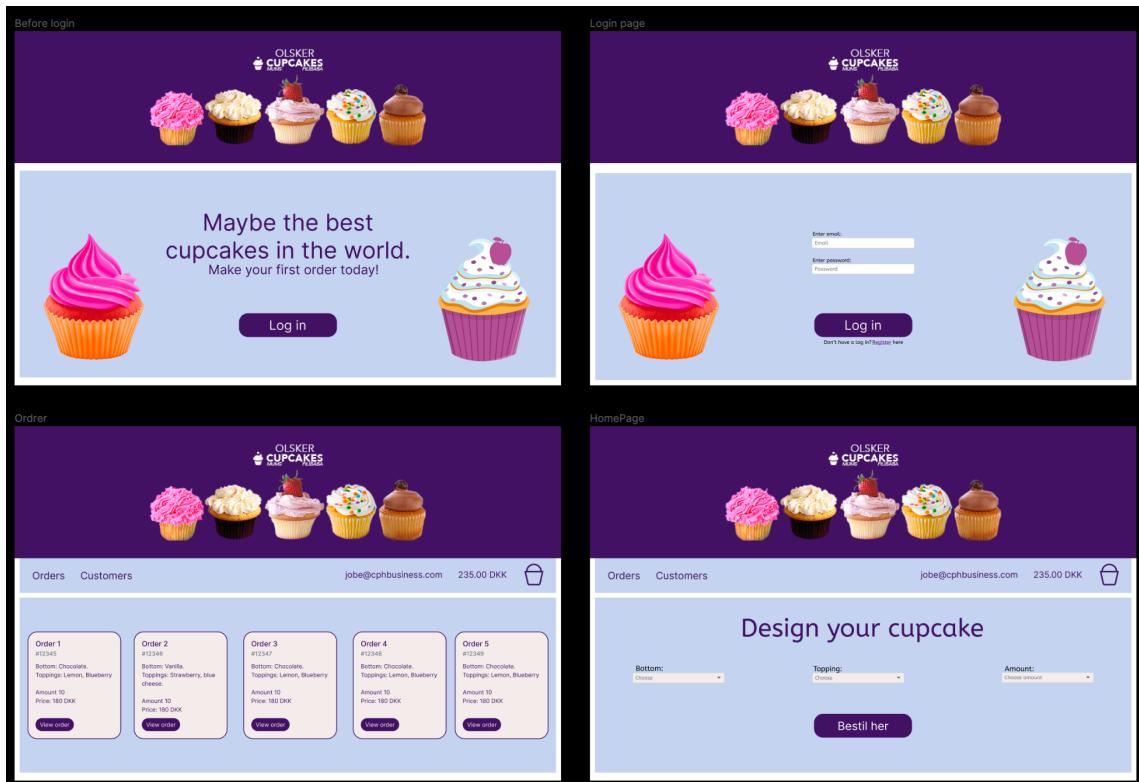
Når en bruger logger på instantieres et User objekt, som der gemmes en reference til i session scopet. Denne reference bruges blandt andet til at vise brugerens e-mail i vores navigations bar, og til når brugeren tjekker ud fra shopping cart siden. Ydermere gemmes der en reference til et nyt Order objekt når brugeren logger på. Dette Order objekt bruges som indkøbskurven, og det er her ordre linjerne som brugeren har tilføjet gemmes.

Status på implementationen

Hjemmesiden er fuld funktionel men langt fra perfekt. Alle de ni User Stories er blevet implementeret. Derudover er der tilføjet indledende *styling* til alle JSP-sider, således gestaltlovene går igen på alle siderne.

Figma

På figur 2 kan man se et udsnit af vores Figma mock-up som vi har forsøgt at ramme med vores endelige produkt, så præcist som muligt ved hjælp af *css styling*.



Figur 2

Vi har gjort brug af en farvetemateknik, der er kendt som (60/30/10).

Man udarbejder altså en palette med sine temafarver. Vores palette ser ud som vist på figur 3

Den udtonede blå farver er altså vores dominante farve. Som kontrastfarve til den blå, har vi valgt en cremehvid, som skal danne et modspil til den blå fyldefarve, som primært skal skabe baggrunden. Til sidst har vi den iøjnefaldende lilla farve, som vi udelukkende bruger til at highlight det vigtige på siden. Det kan være knapper, tekst, overskrifter osv..

Den lilla farve fylder, som undtagelse, lidt mere end 10% som antaget, da vi endte med at gå med et stort header-billede med samme farve.



Figur 3

CRUD

Create, Read, Update & Delete (CRUD), bliver brugt nærmest alle steder på siden. Når en bruger opretter en profil, så bliver der oprettet en *query* som laver et *CREATE*-kald ned til databasen, der så opretter i bruger.

Når en bruger skal logge ind, så benytter vi *SELECT* til at verificere email med kodeord. *UPDATE*-kaldet bliver aktuelt når en bruger vil opdatere sin ordre. Hvis brugeren beslutter sig for kun at bestille fem cupcakes i stedet for ti, så bliver der lavet en *query* til at opdatere en allerede eksisterende ordre.

Hvis en bruger beslutter sig for at slette sin ordre, så bliver der naturligvis lavet et *DELETE*-kald ned til databasen, som sørger for at den pågældende ordre bliver slettet.

Fejl på målstregen

Her er en liste over fejl/bugs som ikke nåede at blive rettet:

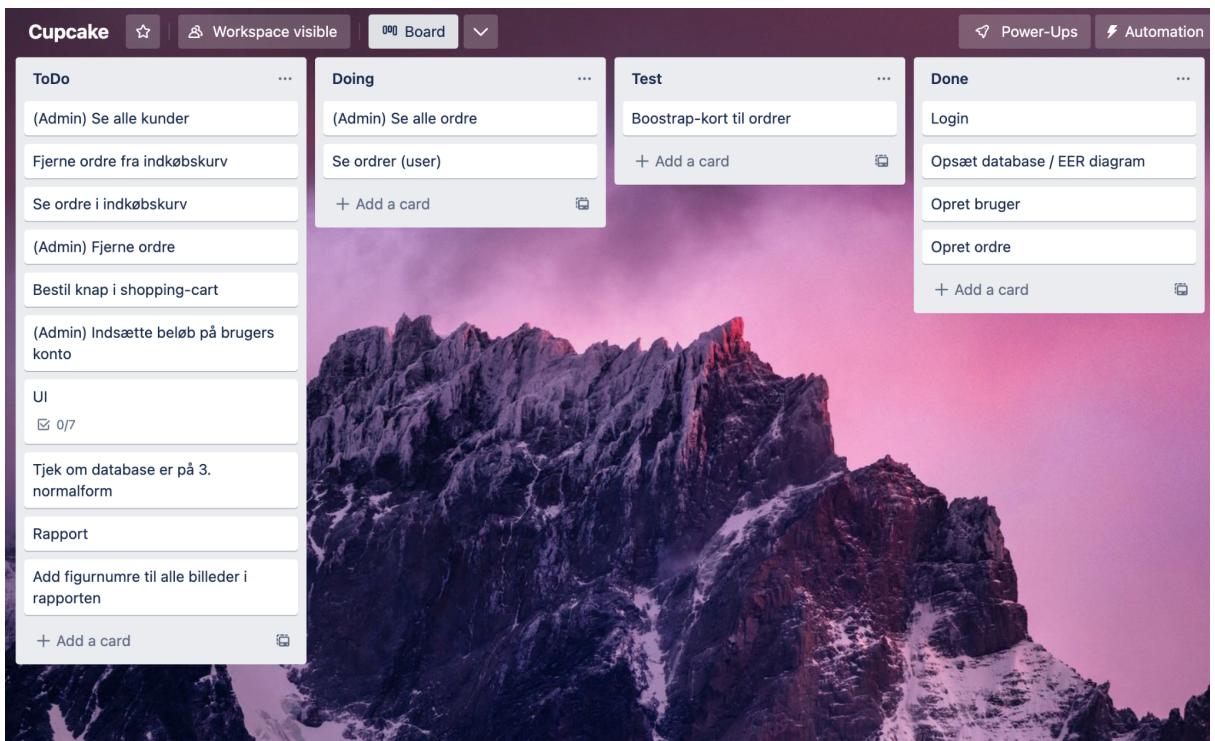
- Man kan ikke sætte decimaltal ind på brugerens *wallet*.
- Error-beskeder er ikke målrettet den endelige bruger. (f.eks. når man smider et bogstav ind som parameter i *amount*, når man bestiller cupcakes).
- Hvis man, som admin, sletter en ordre og derefter trykker på “Go back”-knappen, så bliver man sendt tilbage til siden hvor ordren stadig eksisterer. (Dog er ordren fjernet fra databasen, så hvis man opdaterer siden, så er den væk).

Hvis vi skulle lave videre på projektet, så ville vi tilføje en alternativ checkout-metode, som gør det muligt at betale med andre metoder end kredit, f.eks. dankort, MobilePay, PayPal mm..

Proces

Trello

Vi startede med at lægge en plan for, hvordan projektet skulle forløbe. I gruppen har vi tidligere erfaring med projektstyringsværktøjet, Trello. Det er et værktøj man bruger fra browseren, som nemt og overskueligt giver et overblik over det vilkårlige projekt.



Figur 4

På figur 4 kan man se vores indledende Trello Board. Der er altså oprettet fire kolonner. ‘ToDo’-kolonnen er der hvor alle vores opgaver starter. Efterfølgende kan hvert gruppemedlem så vælge en opgave fra ToDo som de vil udvikle på. Når man vælger en opgave, så flytter man kortet over i ‘Doing’-kolonnen. Når så man har færdiggjort sin opgave, flytter man kortet til ‘Test’-kolonnen. Når ens implementation er blevet testet, og sandsynligvis rettet til, så ender kortet naturligvis i ‘Done’-kolonnen.

Ovenstående er vores ideelle arbejdsmetode, men i praksis foregik det lidt anderledes. Fordelen ved et Trello Board er, at det giver et naturligt overblik over processen hele

vejen igennem forløbet. En af ulempene ved Trello er, at det fungerer bedst, hvis man som gruppe har en fuldendt opgaveliste. Vi sidder første dag med projektet og skal prøve så vidt muligt at forudsige hvilke opgaver, der vil komme og hvilke eventuelle vanskeligheder der kan forekomme. Det er meget usandsynligt at man får alle projektets opgaver noteret på første dagen. Med det sagt, så er Trello skabt til at tilføje opgaver undervejs i forløbet. Det gjorde vi også, når der pludselig dukkede flere opgaver op. Problemet er bare, at man typisk hopper på en opgave med det samme man opdager den. Så de når sjældent at blive skrevet ind på Trello. Når det så er sket gentagne gange, så er der en masse implementationer, som mangler på Trello. Det er ikke hensigtsmæssigt, da Trello gerne skulle give et overblik før, undervejs og efter projektet.

GitHub

I gruppearbejde er det vigtigt at strukturere arbejdsfordelingen fornuftigt. Dette gjorde vi ved hjælp af GitHub, som gør det muligt at arbejde på ét projekt men på flere *branches*. Vi benyttede udelukkende GitHub Desktop, som gør det nemt og overskueligt at oprette *branches*, *push*, *pull* osv.

Vores arbejdsmetode var at oprette en ny *branch* hver gang man tog sig en større opgave fra Trello. Dette var en arbejdsmetode som ingen i gruppen havde prøvet før. I tidligere projekter havde hvert gruppemedlem sin egen ‘branch’ opkaldt efter medlemmet selv, og så arbejder man på den hele forløbet igennem. I dette projekt prøvede vi en ny metode. En *branch* kunne f.eks. hedde *CreateUser*. Når gruppemedlemmet så har *merged* til vores *master branch*, så sletter pågældende *CreateUser branchen* så vi ikke ender med at have for mange *branches* liggende.

Det eneste tidspunkt vi ikke benytter GitHub Desktop er, når vi skal *merge*. Her benytter vi i stedet IntelliJ, da vi er vant til at håndtere *merge conflicts* gennem dét program.

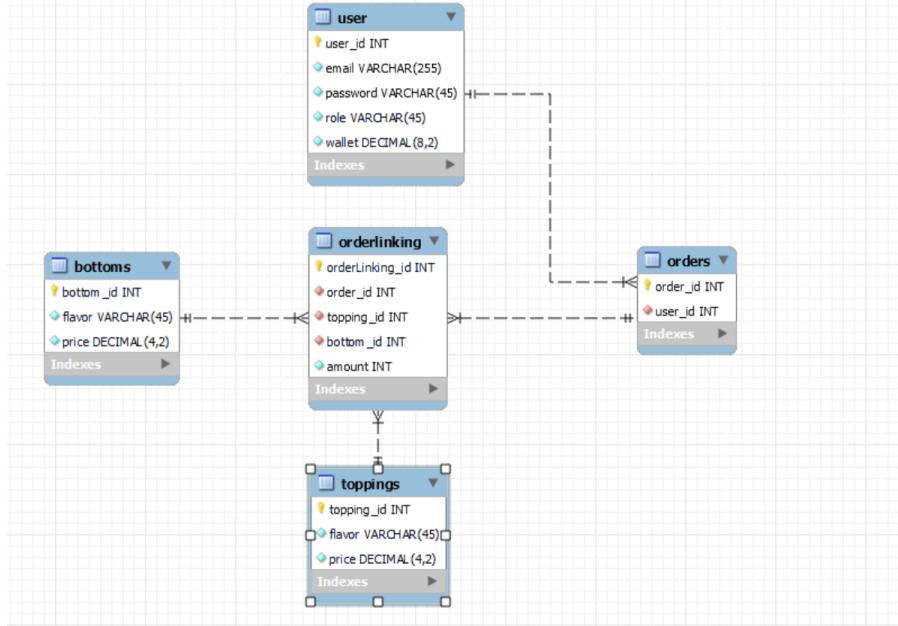
Konklusion

Ved projektets ende kan vi konkludere, at Trello helt sikkert har hjulpet mere end det har gjort skade, og vi vil helt sikkert bruge det igen. Dog må vi også erkende, at vi ikke har brugt det til det fulde. Til næste gang vil vi forsøge at skærpe normerne for hvornår vi opdaterer Trello Boardet, således det kan beskrive det fulde projekt til slut også.

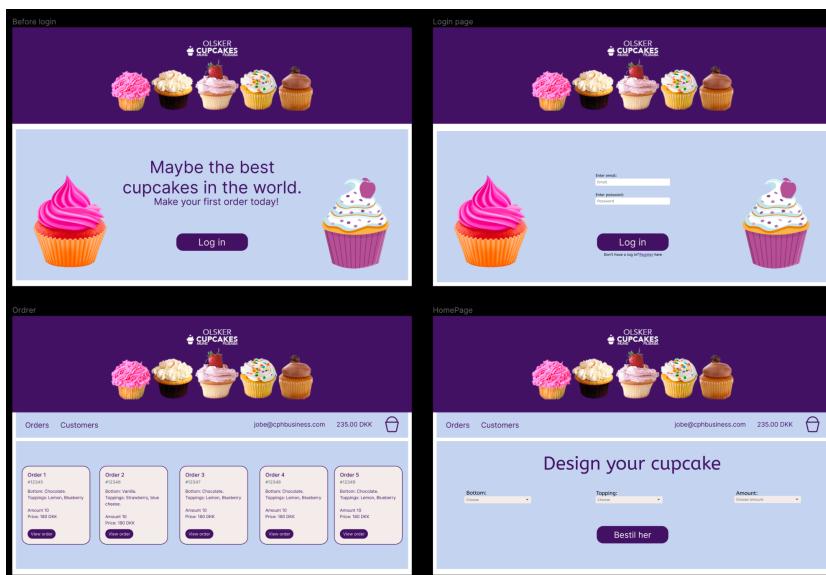
GitHub Desktop har helt sikkert også været til mere gavn end skade, og vi vil med al sandsynlighed benytte det igen. Det er virket upåklageligt hele vejen igennem. Vi lærte at bruge *branches* til hver opgave i stedet for at have én *branch* til hvert gruppemedlem. Vi har bemærket at denne arbejdsmetode, har gjort det mere overskueligt igennem hele forløbet.

Bilag

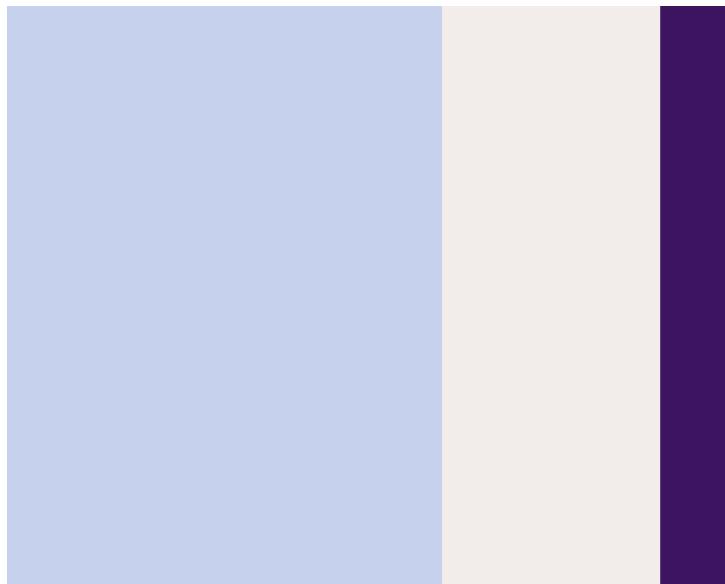
Bilag 1



Bilag 2



Bilag 3



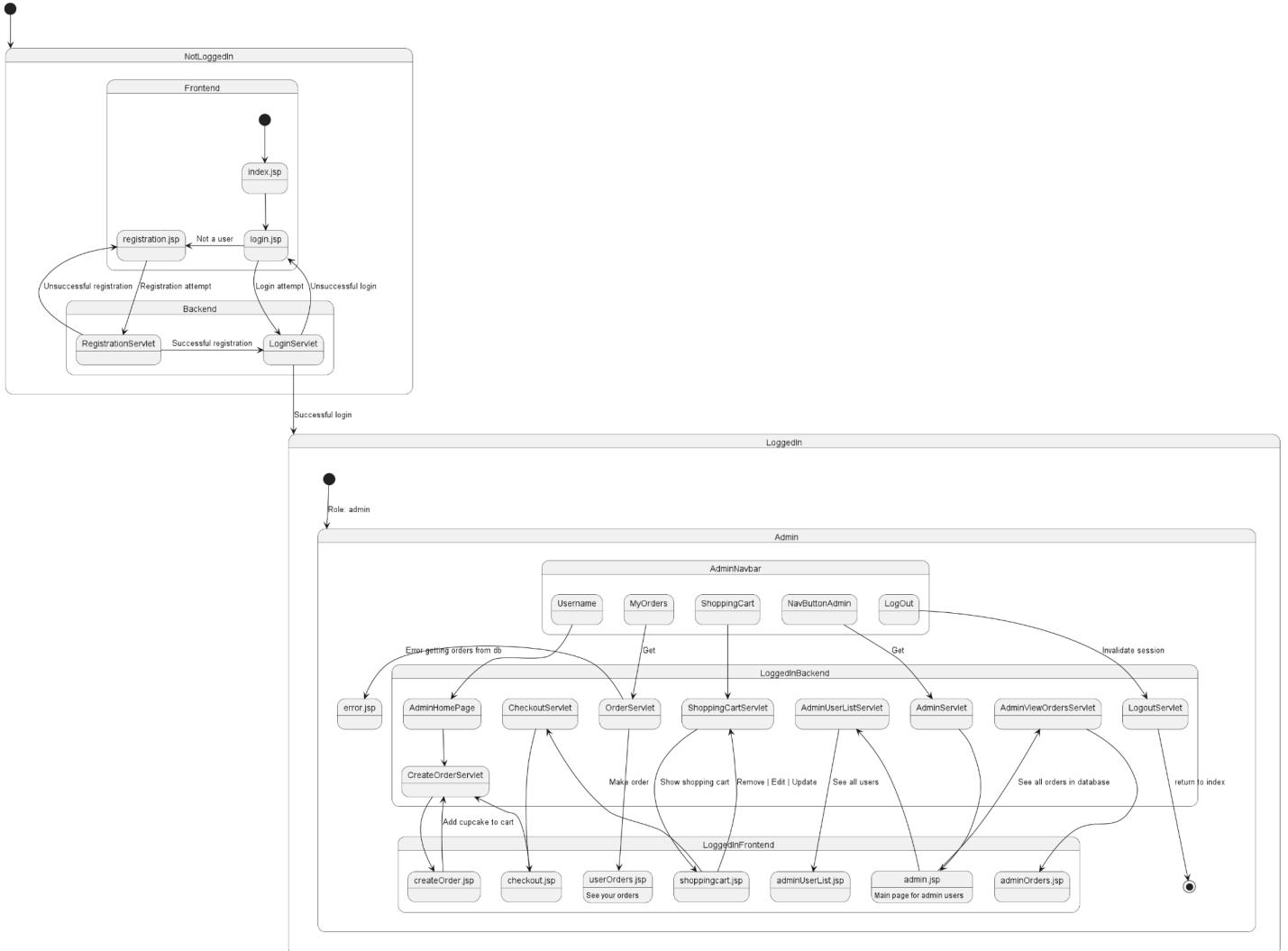
Bilag 4

The screenshot shows a Trello board titled "Cupcake". The board has four columns: "ToDo", "Doing", "Test", and "Done".

- ToDo:**
 - (Admin) Se alle kunder
 - Fjerne ordre fra indkøbskurv
 - Se ordre i indkøbskurv
 - (Admin) Fjerne ordre
 - Bestil knap i shopping-cart
 - (Admin) Indsætte beløb på brugers konto
- Doing:**
 - (Admin) Se alle ordre
 - Se ordrer (user)
- Test:**
 - Bootstrap-kort til ordrer
 - + Add a card
- Done:**
 - Login
 - Opsæt database / EER diagram
 - Opret bruger
 - Opret ordre
 - + Add a card

The background of the board features a scenic image of a mountain range at sunset.

Bilag 5



Bilag 6

