Cupcake Projekt

(I-dat-da 0222a) Datamatiker 2. semester E22

Projekt og rapport udarbejdet

i perioden d. 7/11-2022 til d. 17/11-2022.

Afleveringsfrist: 17/11-2022 kl. 12.00

Github link: https://github.com/thejamiegc/Cupcake
Video link: https://www.youtube.com/watch?v=JcDrueGC2pc

Udarbejdet af Cupcake-gruppe-2 - Deltagere:

Navn: **Felicia Favrholdt**
Cph-email: cph-ff62@cphbusiness.dk
Github navn: FeliciaFavrholdt
Github link: https://github.com/FeliciaFavrholdt

Navn: **Helena Botn Lykstoft**
Cph-email: cph-hl237@cphbusiness.dk
Github navn: HelenaLykstoft
Github link: https://github.com/HelenaLykstoft

Navn: **Jamie Grønbæk Callan**
Cph-email: cph-jc266@cphbusiness.dk
Github navn: thejamiegc
Github link: https://github.com/thejamiegc

Navn: **Markus Isak Møgelvang**
Cph-email: cph-mm712@cphbusiness.dk
Github navn: solskinIsak
Github link: https://github.com/solskinIsak

Navn: **Andreas Grand**
Cph-email: cph-ag198@cphbusiness.dk
Github navn: ElGrand
Github link: https://github.com/ElGrand

Indholdsfortegnelse

1. Indledning

I vores 2. semester prøve-eksamens projekt fik vi til opgave at kode en digital løsning til den fiktive virksomhed Olsker Cupcakes. Vores funktion som firma er at levere en digital løsning i form af en hjemmeside, der løser virksomhedens ønske om at deres kunder kan bestille cupcakes, aflægge ordrer og dernæst afhente disse ordre i Olsker Cupcakes' fysiske butik.

Målgruppen for dette er projekt er en fagfælle[1]. Projektet er kodet med udgangspunkt i noget startkode[2], vi har fået udleveret af vores undervisere på 2. semester. Både backend[3] og frontend[4] er kodet på baggrund af vores egne implementationer samt en række user stories, som vi ligeledes har fået udleveret til projektets udførelse.

2. Baggrund

Olsker Cupcakes ønsker en webshop (format: hjemmeside) til deres fysiske butik på Bornholm. De ønsker, at man som besøgende på webshoppen skal kunne oprette en bruger og/eller logge på med en eksisterende bruger. Som kunde skal man kunne bestille cupcakes ved at vælge en valgfri cupcake-top og -bund (disse prædefineret af virksomheden selv). Når kundens ordre er oprettet og bestilt, skal kunden selv afhente deres ordre i Olsker Cupcakes' fysiske butik. Herudover ønsker Olsker Cupcakes også, at der er en admin funktion. Som ansat i virksomheden skal man have et admin login og derefter kunne logge ind på webshoppen og have mulighed for at se alle de oprettede brugere, og deres bestilte ordre. Det er også et ønske, at man skal kunne slette ordrer, der ikke er blevet betalt.

3. Videreudvikling

- Fokus på design af hjemmeside - UX/UI.
- Implementation af CRUD på alle tabeller i databasen.
- Tilføje en edit order knap.
- Mulighed for at kunne vælge tidsrum at hente en ordre på.
- Flytte topping- og bottom-list på application scope.
- At automatisk have ordrerne liggende inde på update-siden i stedet for at skulle trykke update hver gang.
- Grafisk live-demonstration der viser ens valgte Cupcake kombination.

[1] Fagfælle: En anden datamatiker studerende på 2. semester der er på samme niveau, men som ikke kender cupcake opgaven.
[2] Startkode 2. sem: https://github.com/jonbertelsen/startcode_2sem_2022
[3] Backend: Den grundlæggende aktivitet - på serverniveau
[4] Frontend: Brugergrænseflade orienteret aktivitet

4. Teknologi valg

I forbindelse med udarbejdelsen af vores projekt har vi benyttet os af nogle forskellige programmer og teknologi valg. Disse er listet her.

4.1. Programmer

- Intellij IDEA - Version: Intellij Ultimate 2021.2.4
- Module SDK: corretto-11 Amazon Corretto version 11.0.16.
- Java database connector - Version: mysql-connector 8.0.30
- Database (localhost) - Version: Mysql workbench 8.0 CE
- Tomcat Webcontainer (server) - Version: Tomcat 9.0.67

4.2. Programmeringssprog

- HTML 5
- CSS
- Twitter Bootstrap
- Java
- HTTP Protocol
- GIT
- Javascript (JST)

4.3. Planlægning og rapportskrivning

- Google Docs - Anvendt til at skrive dokumenter, tage noter mm.
- Trello - Agil scrum værktøj der holder styr på hvor langt vi er nået i processen
- Draw.io - Anvendt til udarbejdelse af forskellige diagrammer.
- Github - projektet tilgås via https://github.com/thejamiegc/Cupcake

4.4. Mockup

- Vi har anvendt Figma til at brainstorme en tidlig mockup af webshoppen - link: https://www.figma.com/file/jXMf3n7oQ6yzCmuFlmfVEt/Cupcake-mockup?t=90lbqpBf8dW07QWZ-0

4.5. Video presentation

- Vi har anvendt Screencast-O-Matic til at optage en gennemgang af vores færdige projekt - denne viser løsningen af kundens brugeroplevelse - link: https://www.youtube.com/watch?v=JcDrueGC2pc

5. Krav til systemet

5.1. Firmaets vision og værdi til virksomheden

Vores vision som firma er at løse Olsker Cupcakes' ønske om en webshop - igennem en brugervenlig hjemmeside. Kunderne skal føle, at det er nemt at logge ind, kunne oprette en bruger og bestille cupcakes med en valgfri top og bund. Herudover skal kunden kunne se deres bestilling i en indkøbskurv, og hvor de skal afhente deres ordre. Den værdi vi tilføjer virksomheden er en udvidelse af en funktion i deres fysiske butik (salg af cupcakes) til en online butik, der er med til at drive større kundeklientel, kundetilfredshed, synlighed og fleksibilitet. Vi vil have fokus på funktionalitet frem for design.

5.2. User stories (funktionelle krav)

Efter et kundemøde med virksomheden har vi fået udarbejdet en række user stories, der kort beskriver hvilke brugere, der har hvilke behov, samt hvad de ønsker at opnå:

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 ordrelinier 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.

6. Diagrammer/modeller

Diagrammer og modeller er indsat som bilag og forklaret kort her under hver titel.

6.1. Aktivitetsdiagram

Aktivitetsdiagrammet er en oversigt over det overordnede workflow i forretningen. Vi har udarbejdet to diagrammer; AS-IS[5] og TO-BE[6]. Diagrammerne *(se bilag 10.1 og 10.2 s. 12-13)* følger en kunde der bestiller en cupcake via Olsker Cupcakes' fysiske butik og på hjemmesiden.

6.2. Model-View-Controller (MVC)

MVC modellen *(se bilag 10.3, s. 14)* giver os et overblik over hele serveren og klienten;

1. Vi starter fra *Client (browser***)**, i vores *"localhost:8080/"*.
2. Herfra sender vi et *http request* ud til en korrekt servlet. Denne servlet fungerer som en *controller*, der håndtere modtagelsen af klienten samt metodekald og data fra vores modeller.
3. Dernæst opretter vi et *metodekald* til *persistence*, som går ned og henter data fra DB (databasen) i MySQL.
4. Data bliver ført tilbage og håndteret i vores persistence lag (mapper, facade).
5. Data bliver håndteret i de forskellige klassers metoder i *entities* og *exceptions* m.m.
6. Når dataen er hentet fra databasen, bliver dataen håndteret i den korrekte servlet *(Controller)* fra tidligere, som så bliver *forward requested* til den ønskede jsp-side man vil refererer til (*View*).
7. Yderligere kan en jsp-side håndtere selve de visuelle komponenter til hjemmesiden (billeder, layout, farver m.m.). Det henter vi så fra et *statisk sammenhæng*. F.eks. i en css fil: "file.css".
8. Til sidst sender jsp-siden en *http response* med data tilbage til klienten.

6.3. Domænemodel

Vores domæne model *(se bilag 10.4, s. 15)* identificerer de konceptuelle klasser og skaber et overblik over kodens opbygning. Som det kan ses i modellen har vi bestemt nogle attributter

[5] AS-IS diagram: Diagram over forretningens nuværende workflow
[6] TO-BE diagram: Diagram over forretningens workflow efter vi har leveret vores system

samt defineret nogle associationer og fastlagt nogle multipliciteter imellem klasserne. Diagrammet læses fra venstre mod højre og nedad.

6.4. EER diagram

Vi har udarbejdet vores database *(se bilag 10.5, s. 16)* med udgangspunkt i udleveret materiale.

Tabeller hvor der anvendes 1-mange relationer:

- User → Order
- Order → Orderline
- Bottom → Orderline
- Topping → Orderline

Vi har gjort brug af nogle primære nøgler; userID (int), orderID (int), orderlineID (int), bottomID (int) og toppingID (int). Disse er surrogat nøgler som er genererede med en automatisk nummerering og fremstår som data skjult for brugeren.

Vi har gjort brug af følgende fremmednøgler; Order - customerID (int), Orderline - orderID (int), Orderline - toppingID (int) og Orderline - bottomID (int). Disse udgør en kombination af kolonner i en tabel og har en sammenhæng med en primærnøgle i en anden tabel.

6.5. Navigationsdiagram (state-diagram)

Vores navigationsdiagram er delt op i 3 dele: admin, user og generel *(se bilag 10.6,10.7,10.8 s. 17-19)*.
Alle diagrammerne starter med index siden og derefter logger man enten ind eller laver en bruger. Både admin- og user-diagrammet viser, at man kan trykke på forskellige sider i navigationsbaren, efter man er logget ind;

- Admin navigations bar:
  - Users (hvor admin kan se alle brugerne, der er oprettet på siden)
  - Orders (hvor admin kan se alle ordrerne, der er oprettet på siden)

- User navigations bar:
  - Make cupcake (hvor brugeren kan lave en cupcake)
  - Shopping cart (hvor brugeren kan se sin shopping cart)

6.6 Sekvensdiagram og Klassediagram

Klassediagram for hele projektet samt sekvens diagram over order confirmation kan ses i projektet under mappen 'documentation'.

7. Særlige forhold

- **Applications scope**: Vi bruger application scope[7] til at gemme data over hele applikationen, i dette tilfælde en Connection Pool.
- **Session scope**: Vi bruger session scope[8] til at gemme data på en session, når en user eller admin logger ind. Det gør, at vi kan gemme dataen til den specifikke bruger. I vores projekt er det username (e-mailen), password, en tom shopping cart samt top- og bottom lists.
- **Request scope**: Vi bruger request scope til at gemme data på en nuværende jsp side og denne data kan videreføres til den servlet der refereres til i jsp siden.
- **Exception handling**: Vi har valgt at håndtere fejl mellem databasen og klienten i vores kode med exception handling. Det gør vi, for at nemmere kunne forstå, hvornår, hvor og hvilke fejl der opstår i koden. På denne måde kan vi nemt tilgå specifikke fejlmeddelelser og derved nemt løse opståede problemer.
- **Login (roller)**: Vi har i vores database valgt at en user skal have en rolle: admin eller user. Når man logger ind, så ved systemet om man er en user eller admin, og ved så hvilke .jsp sider brugeren skal kunne tilgå.
- **Validering af brugerinput**: Et eksempel på dette i vores projekt er vores implementation af topping og bottoms. Disse fik vi udleveret af underviserne, og vi har omdannet dem til tabeller i vores database. Brugeren kan således via *makeacupcake.jsp* siden kun vælge de allerede eksisterende typer toppings og bottoms. Ydermere har vi sat en begrænsning på antallet af cupcakes en kunde må bestille.

8. Status på implementation

- Vi fik implementeret 6 ud af 9 user stories, som var krav.
- User story 7 gik vi igang med og fik nået at implementere, at man som administrator kan se kunderne i systemet. Her mangler vi dog at få implementeret, at man også kan se deres ordrer (på samme side), så admin kan holde styr på disse ordrer.

[7] Her gemmes data som skal bruges over hele applikationens levetid.
[8] Her gemmes data som skal bruges for en specifik bruger/kunde.

- Vi har dog valgt i user story 6 at implementere to sider; en der viser kundernes ordrer, og en der viser kunderne.
- Vi har ikke valgt at oprette en cupcake_test.sql fil og oprette test i vores database, idet det er ikke var krav fra undervisernes side.
- I vores kode har vi ikke valgt at implementere alle CRUD metoder for alle tabeller i databasen, men udelukkende fokuseret på dem vi har skulle bruge for at løse user stories.
- Vi har forsøgt at style .jsp siderne med HTML, så de er funktionelle, men vi har ikke lagt vægt på UI/UX og CSS.

9. Proces/Plan

9.1 Brainstorm - ER diagram

Det første vi gjorde var at sætte og ned os lave en brainstorm *(se bilag 10.9, s. 19)* over hvilke tabeller/schemas, der skulle være i databasen, og hvordan relationerne imellem skulle være. Vi snakkede en del om de forskellige typer af data, og måder vi kunne indlæse, opdatere samt gemme denne data (CRUD).

9.2. Mockup

For at skabe en visuel oversigt over hvordan vi skulle gribe projektet an, besluttede vi os for at lave en mockup i Figma af webshoppens layout. Her anvendte vi den udleverede mockup/header *(se bilag 10.11, s. 20)* og vi fik opsat layout til forskellige sider. Vi snakkede meget om design i denne fase, idet vi gerne ville have en ide om, hvordan de forskellige JSP-templates og -sider skulle opsættes visuelt. Vores mockup er en brainstorm.

9.3. Diagram på tavlen

Vi fik udarbejdet et diagram på tavlen *(se bilag 10.10, s. 20)*, der viste alle JSP siderne, og hvordan deres indbyrdes relation skulle være, herunder enighed om navngivning heraf. Vi blev også enige om at lave 3 page templates (normal, user, admin). Herefter forsøgte vi, at notere de udleveret user stories ned, og hvordan deres tilhørsforhold til siderne var. Dette gjorde vi for, at imødekomme en guideline for, hvordan vi skulle komme i gang med at kode.

9.4. Trello

Vi har anvendt Trello *(se bilag 10.12, s. 21)* som et agilt scrum værktøj. Vi startede med at opdele hver user story og herunder tildele forskellige todo's til hver. På denne måde kunne vi

sætte os selv nogle delmål og interne deadlines. Ved hver user story fik vi pålagt specifikke krav som f.eks. at en DropDown menu, knapper m.m. På denne måde skabte vi en oversigt, der var nem at tilgå ift. hvad der skulle kodes, og hvad vi var nået i mål med.

9.5. GIT & Github

Vi har brugt git via vores lokale terminaler til at branche ud, committe og push/pull vores kode til/fra github. Vi har vedlagt en oversigt over commits for projektet *(se bilag 10.13, s. 21)*. Til gengæld brugte vi værktøjslinjens git, når vi skulle merge branches sammen.

9.6. Refleksion over projektets forløb

Vi fik løst opgaven med godt samarbejde, hvor alle havde mulighed for at komme med inputs og ideer til den endelige løsning. Vi har mødtes fysisk og også arbejdet online via Discord. Vi har haft gavn af at planlægge vores process og iterere i den før- under og efter hver user story og deres implementationer.

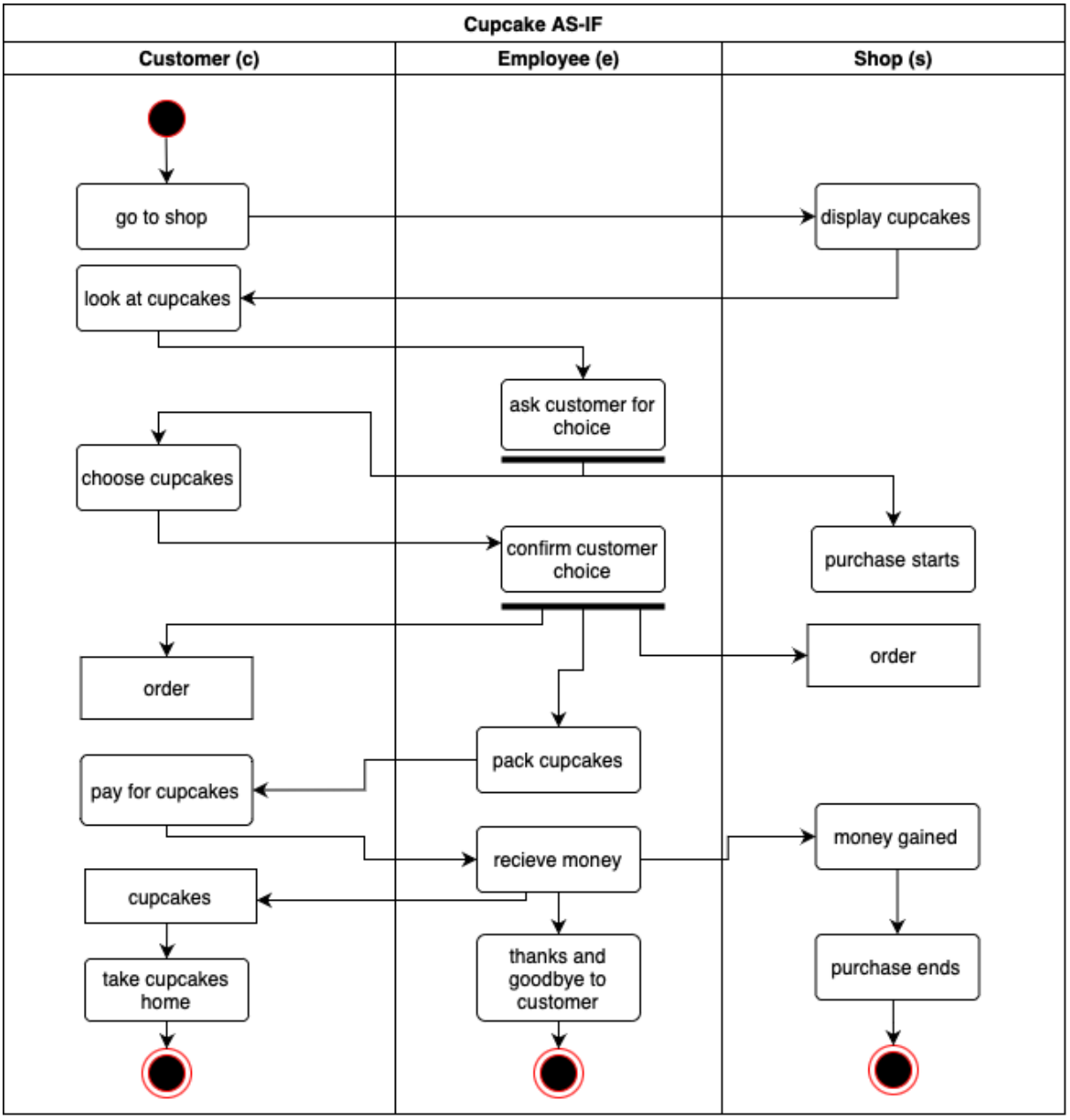
Som gruppe valgte vi ikke at dele user stories op imellem os, idet vi følte, at de hang for meget sammen til, at vi kunne adskille dem komplet. Vi var bekymret for, at adskillige gruppemedlemmer ville sidde og kode den samme kode. Og da vi først forsøgte os med det, fandt vi ud af, at det var dobbelt så meget arbejde og ikke produktivt. Derfor har vi som udgangspunkt kodet sammen men skiftes til at være den bag tasterne så alle har fået lov at kode noget hver især. Vi har formået at kode to versioner af samme kode, idet vores første version var en misforståelse af projektets opbygning. Men det lykkedes, at komme i mål, vi skrottede version 1 og vi fik videreudviklet version 2.

De sidste to dage valgte vi at fordele rapportskrivningen ud til to medlemmer af gruppen, da deres interne styrker var at være struktureret og agere projektledere på projektet - derudover følte vi os også lidt tidspresset ift. at nå at kode de sidste user-stories 6-9, samt opdatere vores visuelle mockup. Derfor måtte vi prioritere vores interne styrker for at nå vores deadline. Dog har alle gruppemedlemmer kommet med inputs til rapportens indhold og læst korrektur herpå - så der er enighed omkring, hvad der afleveres.

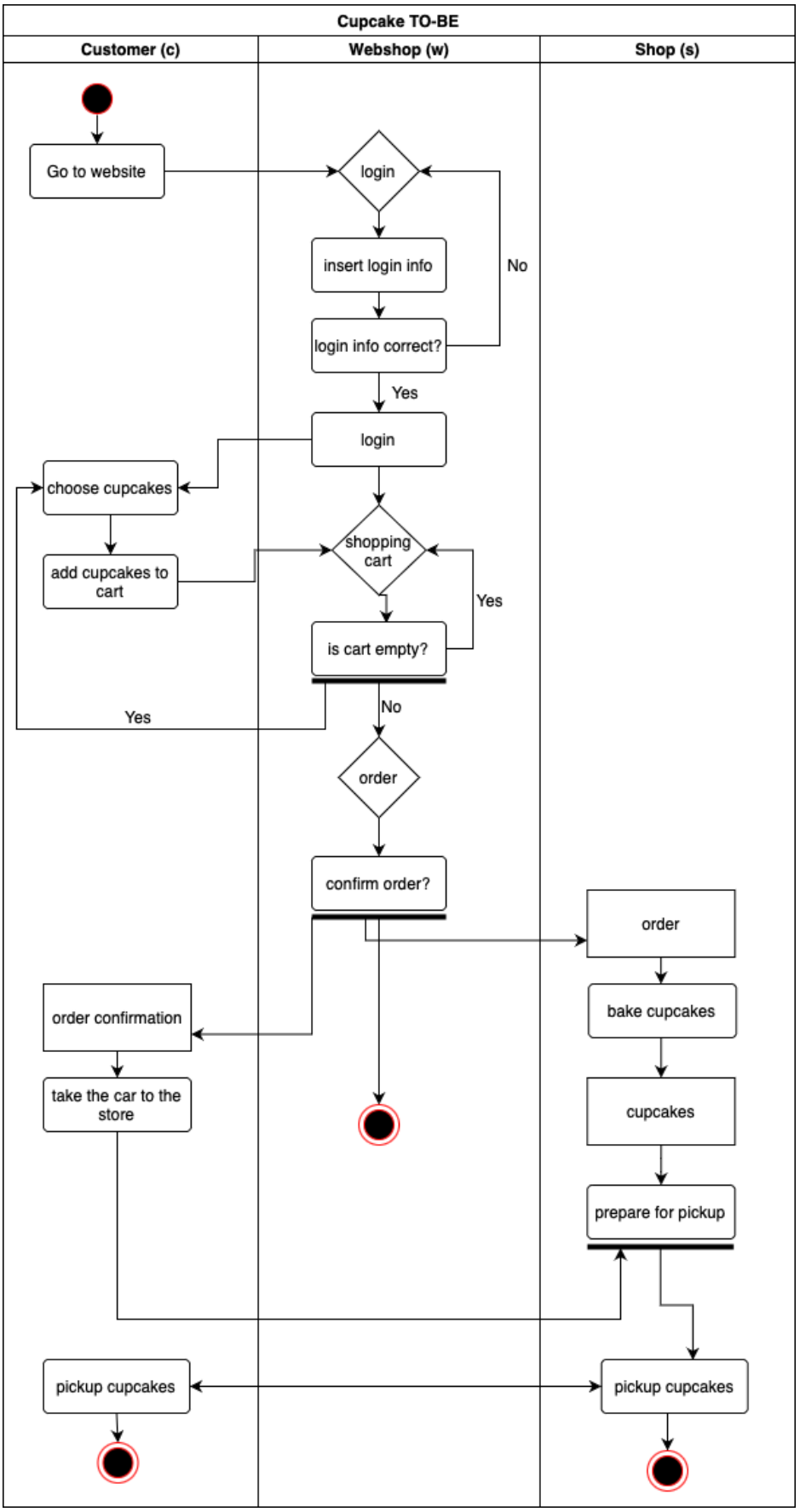
Til næste projekt vil vi nok vælge, at strukturer vores GIT forløb lidt bedre og eventuelt uddele forskellige user stories til gruppemedlemmer - dvs arbejde i par to og to. Herudover vil vi nok skulle forberede os lidt bedre ift. opsætning af de forskellige diagrammer, da meget af vores spildtid har ligget her og i forståelsen af den udleveret startkode - det er ikke altid nemt at forstå kode man ikke selv har været inde over fra start, men det er en god øvelse, som vi klart tager med os i fremtiden. Vi skal også anvende tests fremadrettet, idet vi ikke har valgt at implementere test database i dette projekt.

10. Bilag/Appendiks

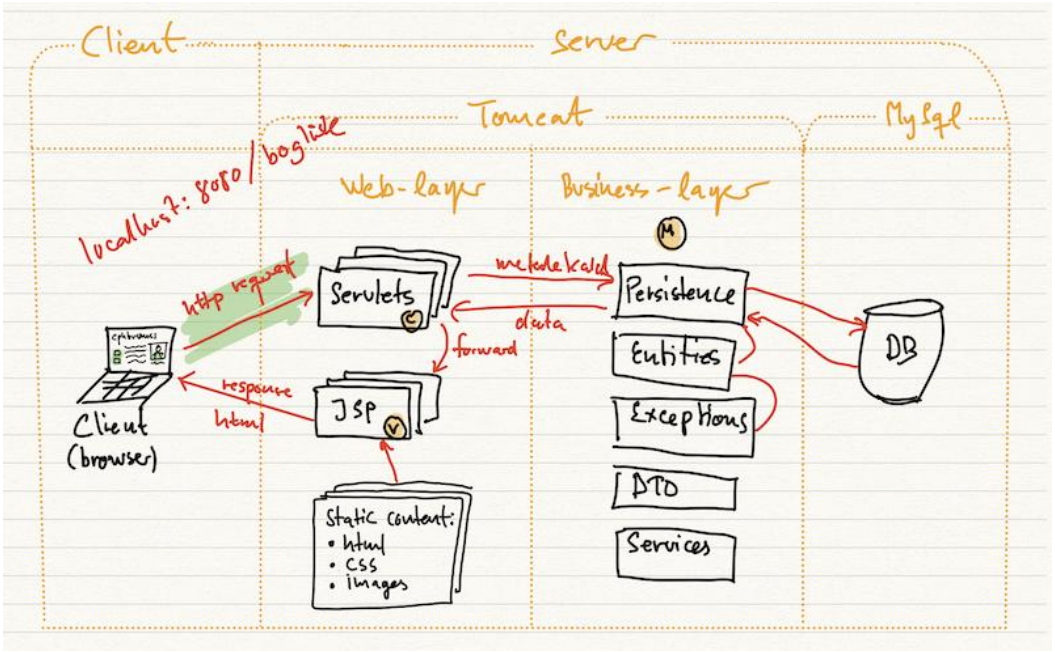
10.1. Bilag - Aktivitetsdiagram (AS-IF)



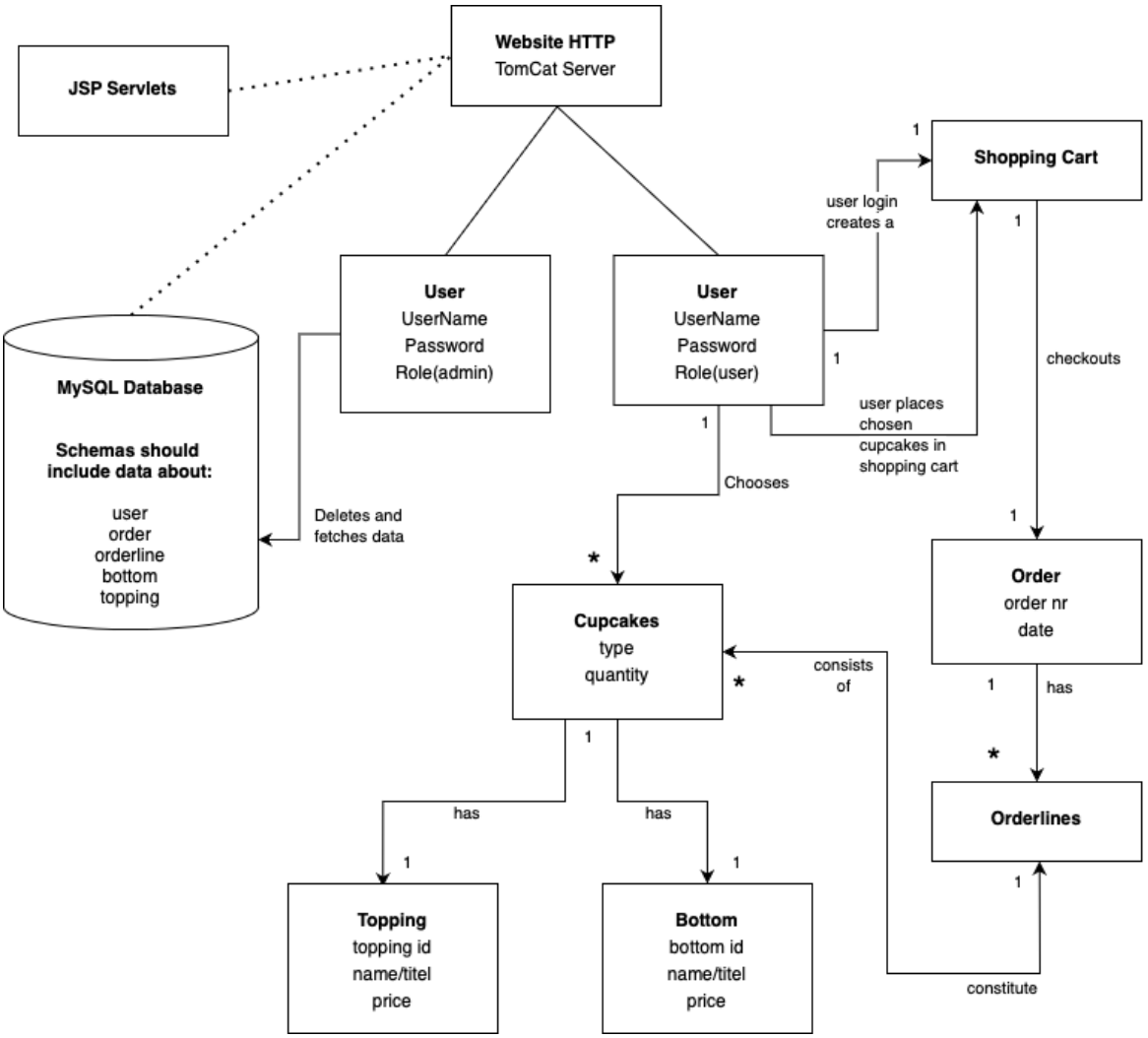
10.2. Bilag - Aktivitetsdiagram (TO-BE)



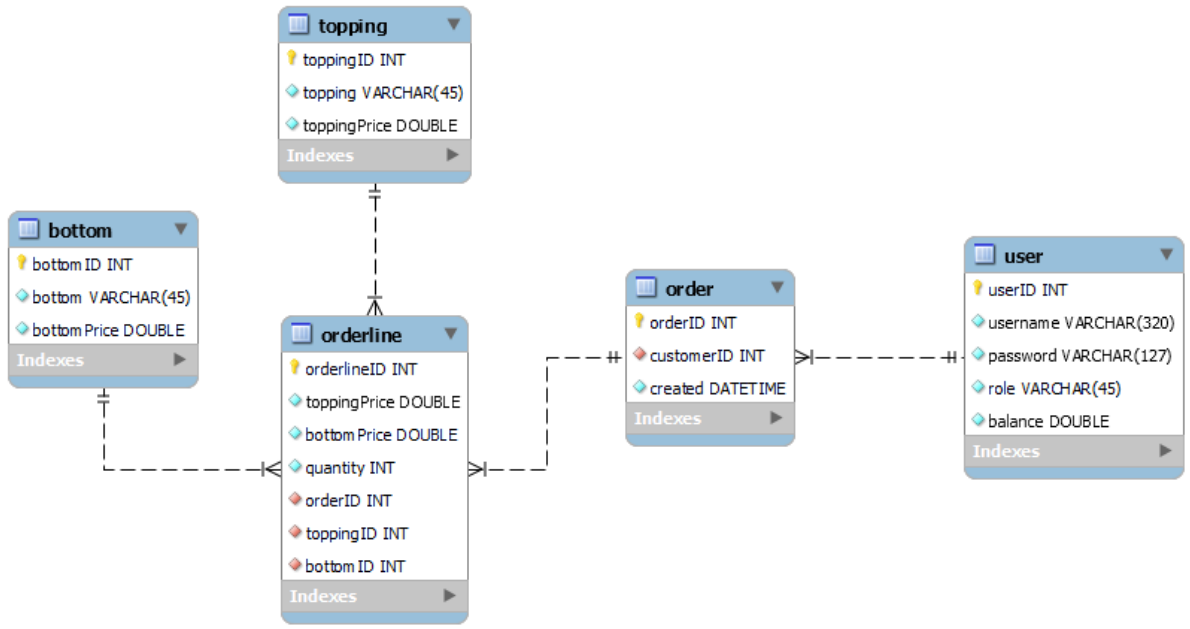
10.3. Bilag - MVC



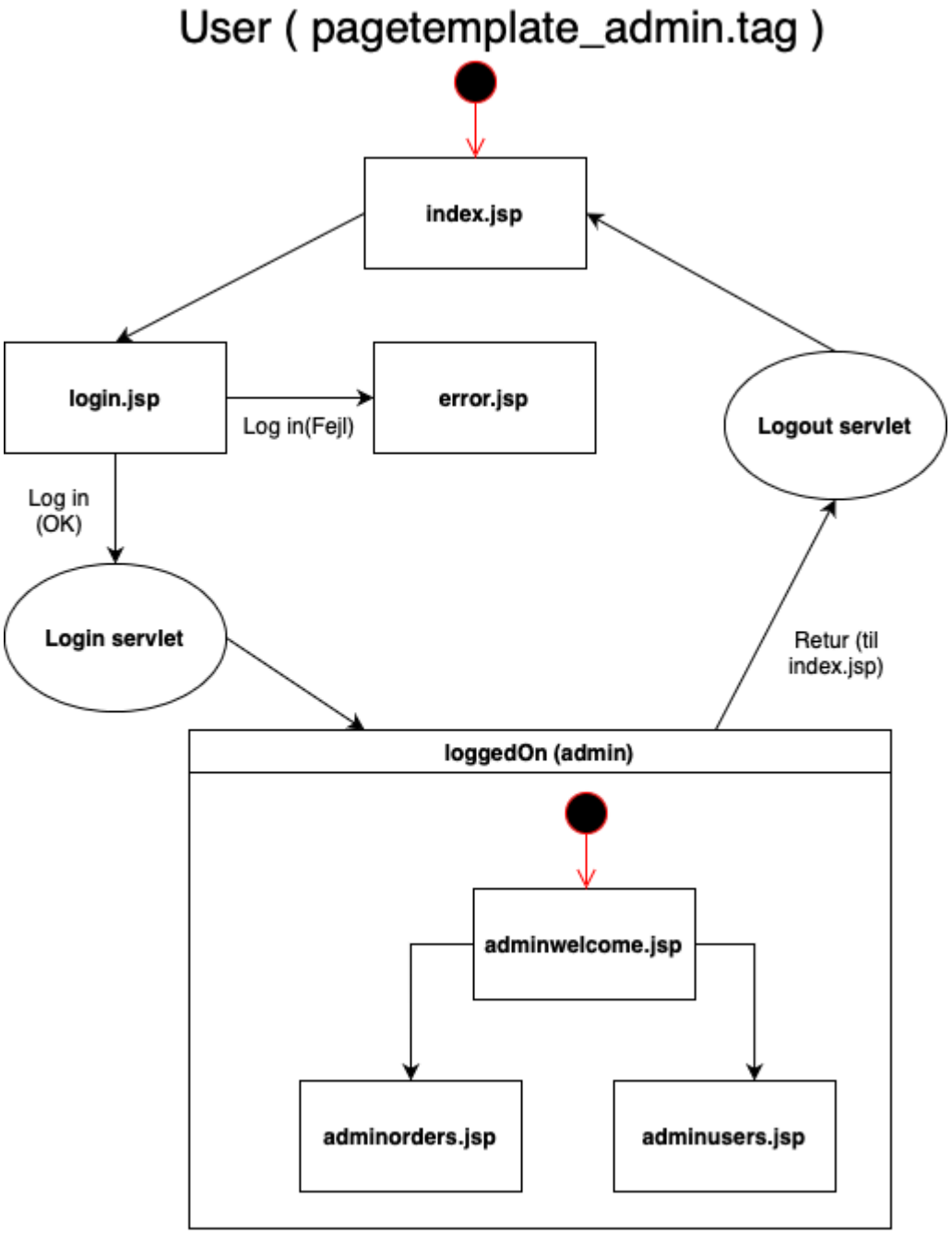
10.4. Bilag - Domæne Model



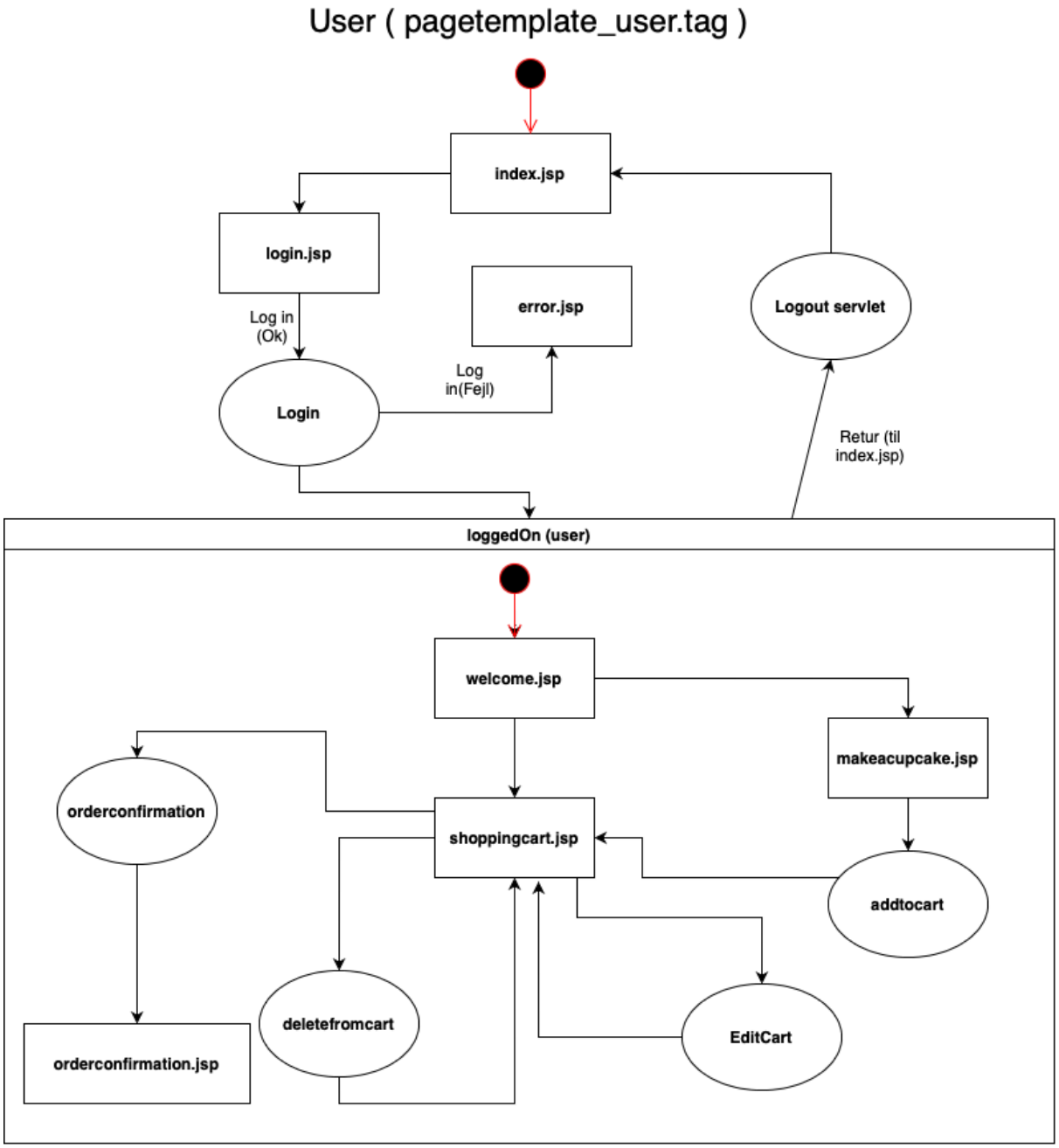
10.5. Bilag - EER Diagram



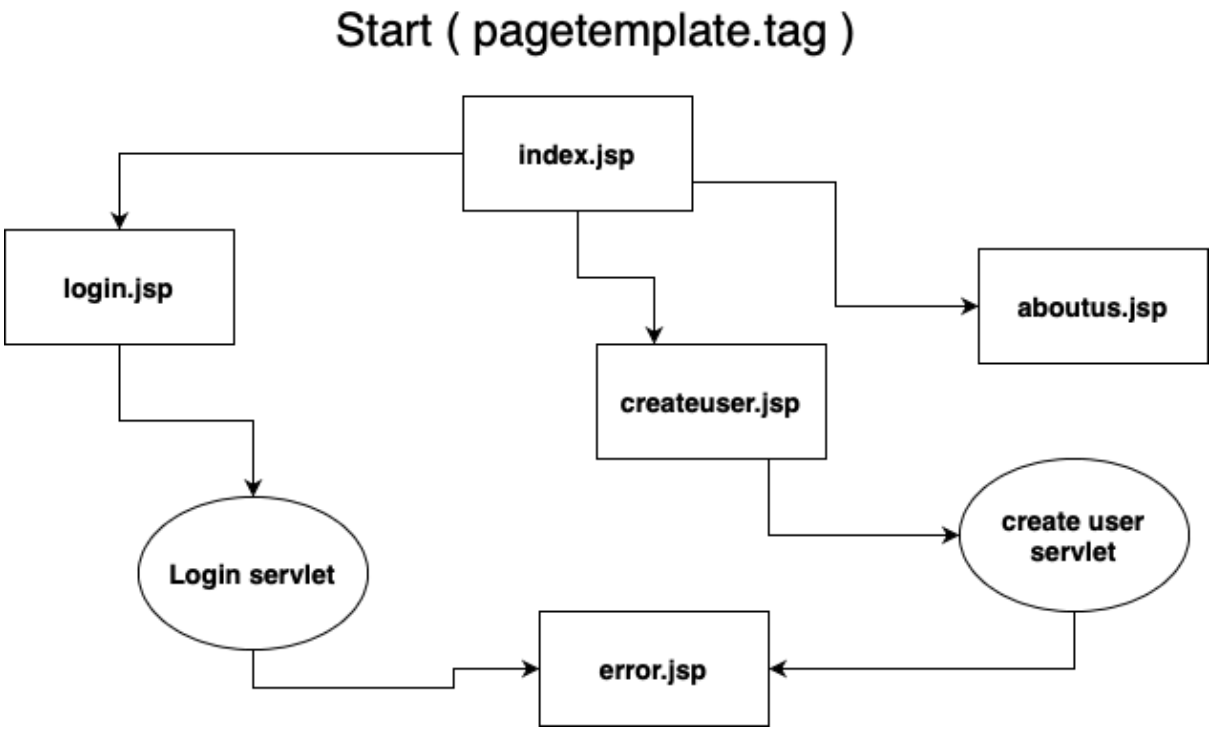
10.6. Bilag - Navigationsdiagram (admin)



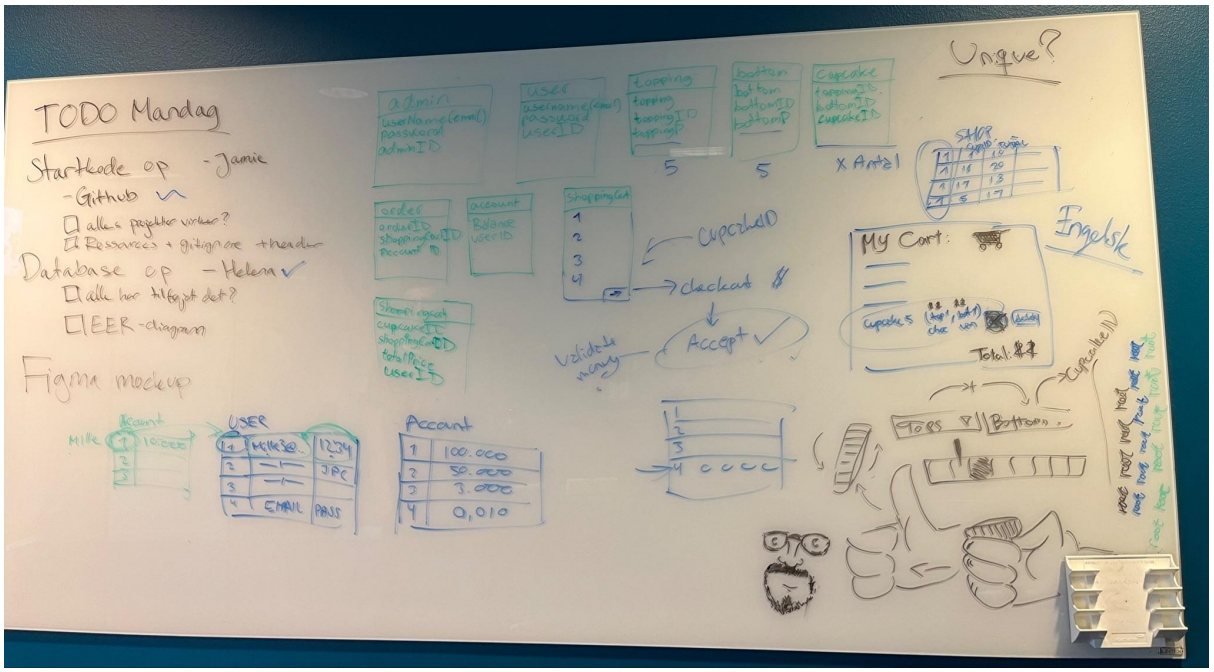
10.7. Bilag - Navigationsdiagram (user)



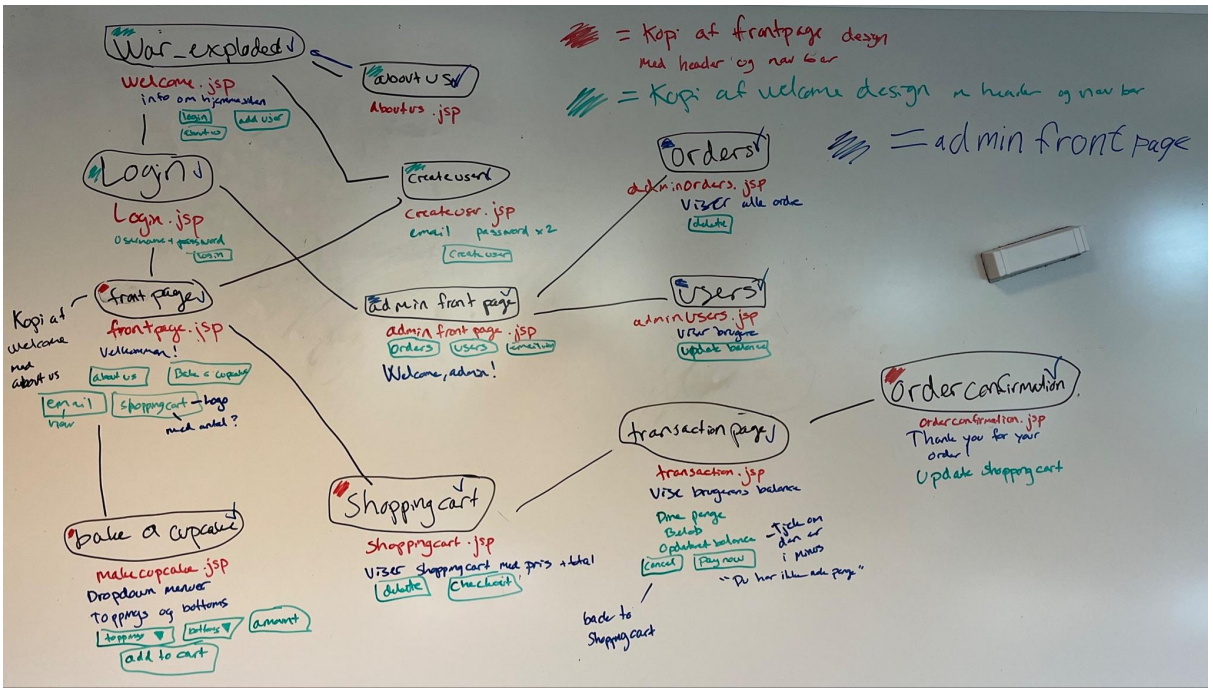
10.8. Bilag - Navigationsdiagram (general)



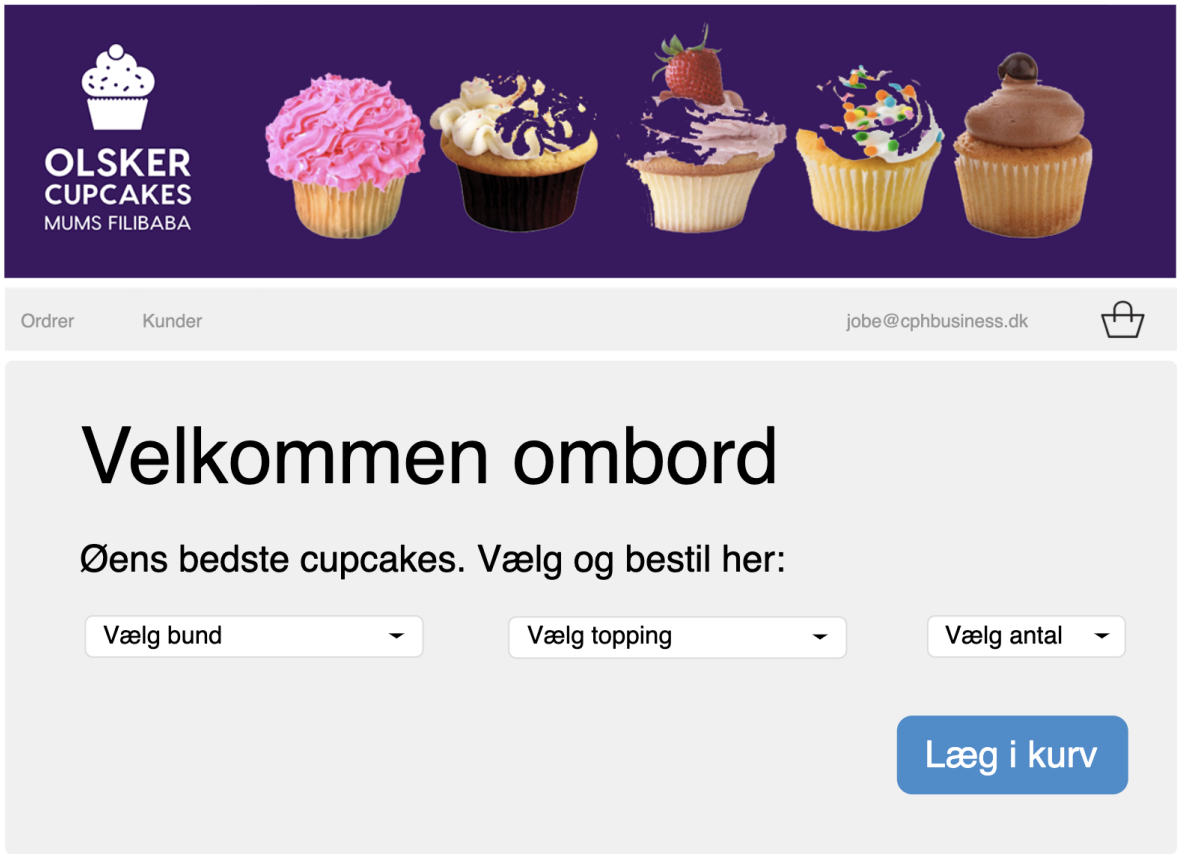
10.9. Bilag - Brainstorm ER



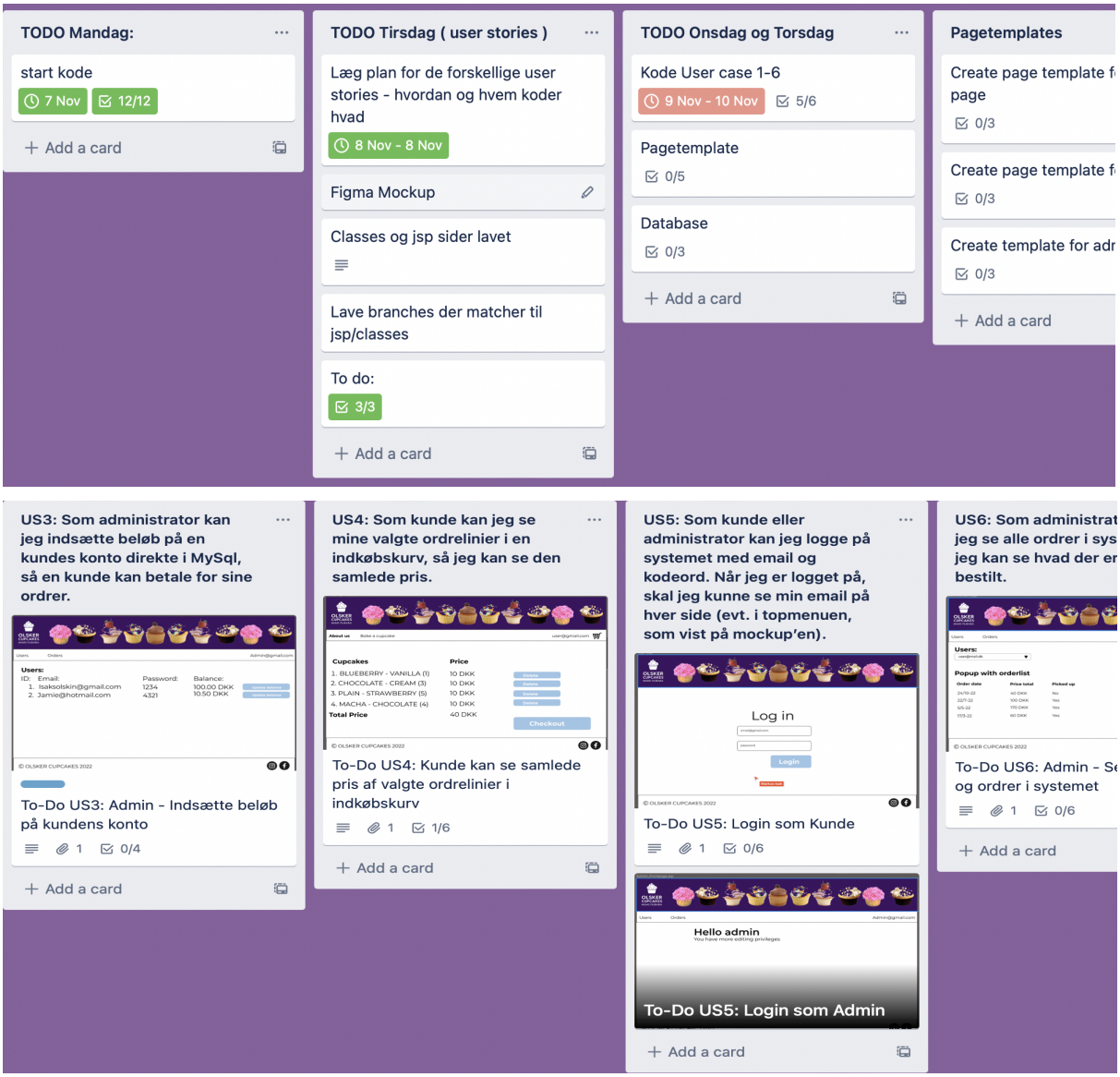
10.10. Bilag - Diagram på tavlen



10.11. Bilag - Udleveret mockup og header



10.12. Bilag - Trello



10.13. Bilag - GIT & Github