Cupcake projekt

Galler, Frederik
cph-fg56@cphbusiness.dk
github.com/Freddiiy

Jakobsen, Christian Nybye
cph-cj449@cphbusiness.dk
github.com/Tsukani

Weinell, Nikolaj
cph-nw76@cphbusiness.dk
github.com/Weinell

19. november 2021

Indhold

1 Indledning

1.1 Baggrund

Olsker Cupcakes havde anmodet om en digital platform, hvor kunder har mulighed for at bestille cupcakes og komme forbi butikken og hente dem. Cupcakes skal kunne laves med en brugerdefineret top og bund, og der skal være mulighed for at bestille flere cupcakes. Flere af disse krav vil blive nævnt senere.

1.2 Teknologi valg

Der er blevet benyttet:

- Java til backend, med Maven som build tool
- HTML og Bootstrap 5 til frontend
- MySQL til database, med JDBC som database connector (DBeaver til at lave databasen)
- Tomcat 9.0.54 til webserver som kører på Ubuntu 20.04
- Trello (agile) til projektstyring

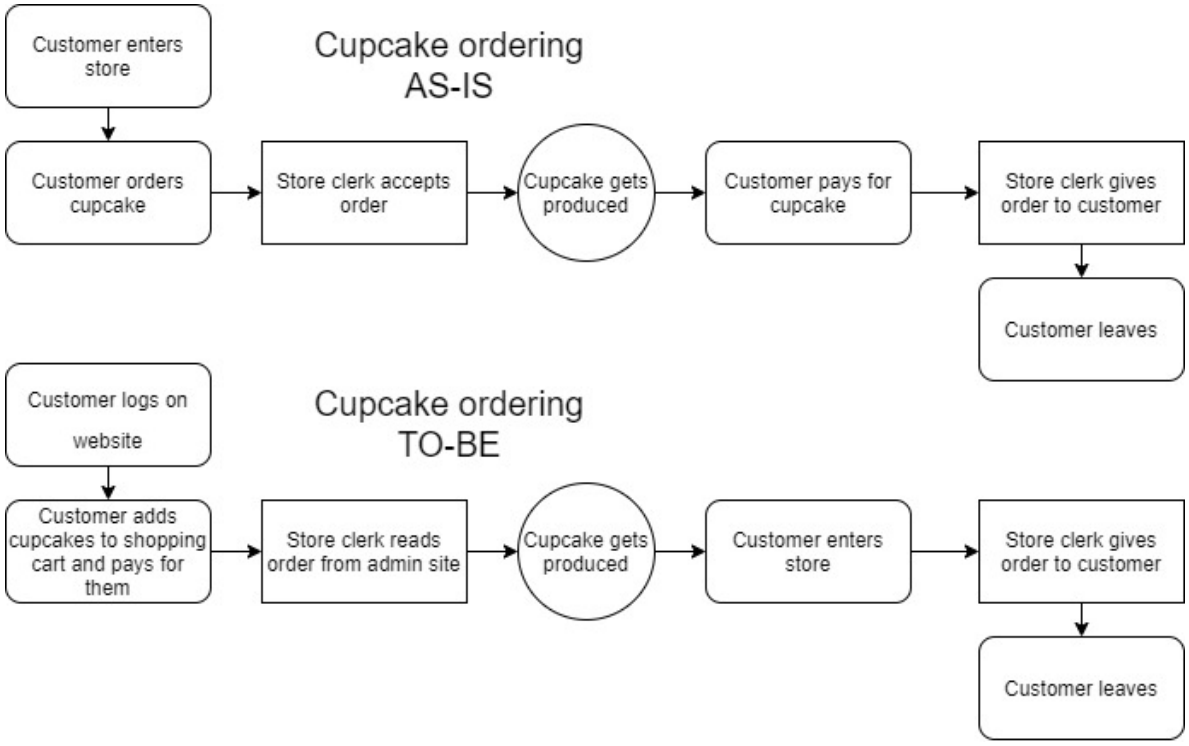
2 Krav

Som nævnt i sektion 1.1 havde Olsker Cupcakes specifikke krav til projektet:

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.
2. Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en en ordre.
3. Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer.
4. Som kunde kan jeg se mine valgte ordrelinier i en indkøbskurv, så jeg kan se den samlede pris.
5. Som kunde eller administrator kan jeg logge på systemet med email og kodeord. Når jeg er logget på, skal jeg kunne min email på hver side (evt. i topmenuen, som vist på mockup'en).
6. Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.
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.
8. Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.
9. Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde udgyldige ordrer. F.eks. hvis kunden aldrig har betalt.

3 Aktivitetsdiagram

Følgende aktivitetsdiagram viser et flow over hvordan en kunde ville bestille en cupcake før i tiden (AS-IS), og efter vores digitale løsning (AS-IS).



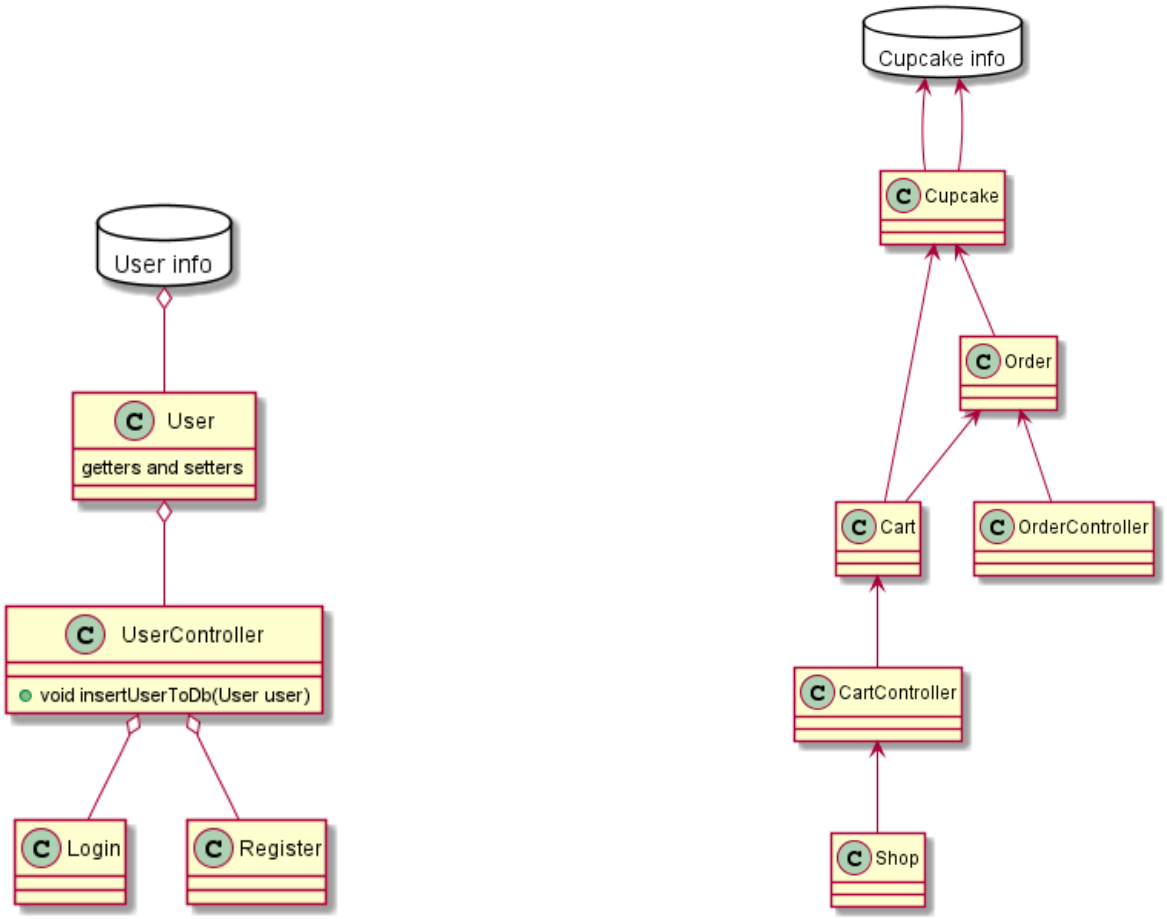
Figur 1: Aktivitetsdiagram

Diagrammet viser hvordan der ikke bliver færre led i orderprocessen, men stadigvæk gør processen nemmere for både kunder og butiksejer. Hele processen er frontloaded, hvilket gør det nemme for butikken, da de kan fokusere på at lave cupcakes. Processen er også nemmere for kunden, da de har mulighed for at bestille deres cupcakes hvorend de er, og har intet stress da de kan sende ordren når de føler sig klar.

4 Domænemodel og ER diagram

4.1 Domænemodel

Vi har valgt at sætte fokus på to forskellige domænemodeller (klassediagram), da det gør det mere overskueligt og nemmere at forklare.



(a) User UML

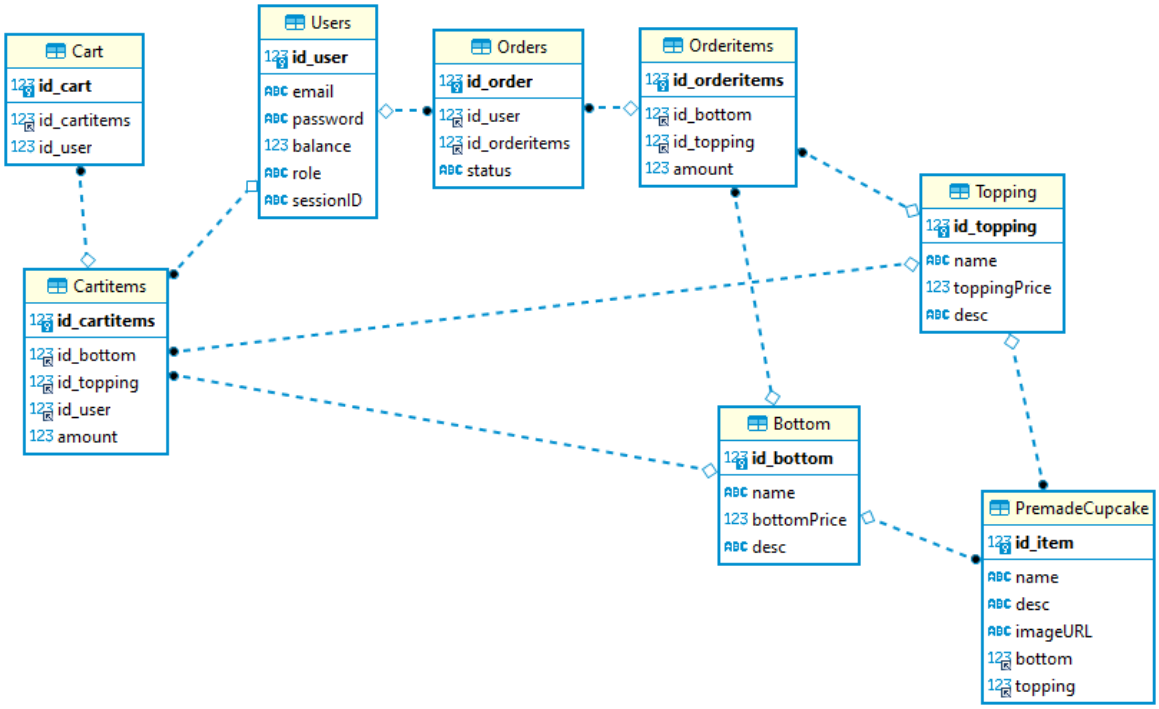
(b) Ordering UML

Figur 2: To forskellige UML diagrammer brugt i programmet

I figur (a) "User UML"ses flowet således at en UserController objekt kommunikerer med et User objekt som bliver hentet fra databasen. Hvis databasen ikke indeholder den User (email adresse), så vil Loginklassen vide dette, og forwarde brugeren til Register siden.

I figur (b) "Ordering UML"ses flowet således at hvordan en bruger der allerede er logget ind interagerer med shoppingsystemet. Cupcake information bliver hentet fra databasen og via "getters"ved Order og Cart, hvilket infomerer om hvilke cupcakes der er på lager og deres pris. Shop er der hvor brugeren vælger en cupcake som CartController derefter tilføjer til Cart - for at betale for en order skal kunden trykke Køb, hvorefter Cart sender ordren til Order.

4.2 ER Diagram



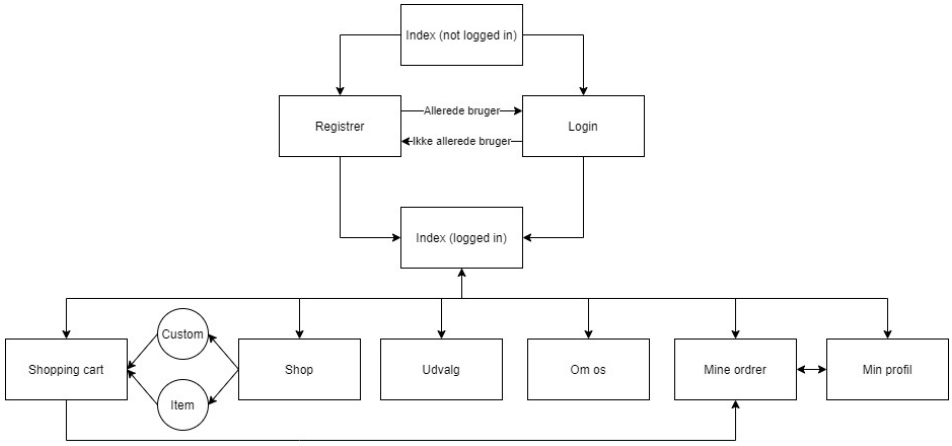
Figur 3: ER diagram

- Cart er vigtig for at validere hvilke items der er i en users shopping cart.
- CartItems er vigtig da den definerer hvilke items og hvor mange der er i den users cart.
- Users indeholder alt information vi har om en kunde eller admin. Obs. session ID på backend til validering.
- Orders gemmer hvilke ordrer en kunde har haft.
- OrderItems indeholder informationen om en kundes ordrer.
- Topping og Bottom indeholder cupcake navn og pris for diverse cupcakesmage - en sjov ting at tilføje i fremtiden ville være lagerbeholdning.
- PremadeCupcake indeholder vores egne personlavede cupcakes, som en kunde har mulighed for at bestille, istedet for at lave deres egen.

Som det ses fra ER diagrammet, figur 3, har vi valgt at lave en tungt relationsbaseret database. Hver gang en relation er oprettet er det gjort på 3. normalform, hvor der bliver sendt primary keys videre. Vi har gjort det på denne måde, da det gør det nemmere at skalere vores database, og gør at man nemt kan tilføje en ny smag på Topping eller Bottom uden at påvirke de andre klasser såsom Cart.

5 Navigationsdiagram

Navigationsdiagrammet hjælper en stakeholder i at navigere hjemmesiden uden, at være på hjemmesiden. Alle sider kan tilgås ved hjælp af vores navigationsbar (så længe man er logget ind).



Figur 4: Navigationsdiagram

6 Særlige forhold

I sessionen bliver en User gemt - da vores database er relationsbaseret, kan man tilgå den ønskede information på diverse sider.

Alt validering foregår på backenden. Dette er meget vigtigt da validering på frontend kan manipuleres og derfor ikke er sikkert. Når en bruger registrer sig som ny kunde tjekkes der om den brugte email-adresse er gyldig, både på frontend for at gøre brugeroplevelsen simpel samt på backend for at fange folk der muligvis har kommet udenom vores frontend tjek. Derudover sikrer vi at email-adressen ikke eksisterer i databasen i forvejen, samt at de to givende adgangskoder er ens. Det samme gælder når man logger ind. Når en bruger logger ind tjekkes om email-adressen eksisterer i vores database samt om kodeordet er korrekt. Et vigtigt sikkerheds fokuspunkt er sikker opbevarelse af kodeord i databasen. For at sikre mod uopfordret databaseadgang sikres alle adgangskoder med et hashingsalgoritme som gør det umuligt at læse adgangskoder. Hashingalgoritmen er nem at kryptere med men er utroligt svær at dekryptere. Kodeord valideres ved at kryptere koder der bruges i login og tjekke om den krypterede kode er ens med den krypterede kode gemt i databasen.

I projektet benyttes to forskellige brugertyper i databasen. Disse roller er customer eller admin, hvilket definerer hvilke sider en bruger har mulighed for at se.

Vores exceptionhåndtering er primært brugerorienteret. Hvis brugeren ikke er logget ind bliver de automatisk redirected til loginsiden. Ved fejl ved login eller registrering infomeres brugeren om fejlen.

7 Status på implementation

Implementation gik over forventning - vi har opnået alle kundens krav (user stories), og vi har endda tilføjet mange ekstra funtionaliteter som vi kunne forvente kunden ville ønske i fremtiden.

8 Proces

Projektplanlægning skete via et Trello board, hvor både kundens krav og egne oversigtsliste blev skrevet ind som individuelle "cards". Det resulterede i et nemt og overskueligt oversigt over alle ting der skulle arbejdes på og hvilke der allerede var færdiggjort.

Hvisse cards valgte vi at lave individuelt, andre cards valgte vi at lave fælles som en gruppe - men hver gang vi arbejdede sad vi i samme Discord-kanal og snakkede.

Det gode ved vores arbejdsproces var, at vi pressede os selv til at lave et produkt som var bedre og mere sikkert end det som kunden havde bedt om. Det gjorde vi for at øve os i at lave et realistisk projekt, som en kunde ville spørge om i virkeligheden.
En mindre god ting ved vores arbejdsproces var et mindre fokus på dagsplanlægning. I fremtiden vil vi sikre større overblik og specificere hvad der vil lave de enkelte dage.