CUPCAKE

Cupcake Bornholm Gruppe 2

Nicklas Bergmann	David Alexander Vig	Jenny J. Lenebjer Nielsen
cph-nb191@cphbusiness.dk	cph-dv48@cphbusiness.dk	cph-jn466@cphbusiness.dk
GitHub: m4cfly	GitHub: Davud05	GitHub: Jenjolen

11. APRIL 2024

CPHBUSINESS DATAMATIKER

2. semester

Indholdsfortegnelse

Indledning

Dette dokument omhandler et web udviklingsprojekt for Olsker Cupcakes, et iværksættereventyr fra Bornholm, der har fundet en succesfuld opskrift inden for bageribranchen. Projektet er indledt med en initial opgave, hvor vi skal skabe et nyt website for virksomheden baseret på en mockup - en foreløbig visuel guide - skabt af et par hipsters fra København. Denne mockup tjener som et første skridt mod visualiseringen af det endelige website, men den dækker ikke alle aspekter eller funktioner, som websitet til sidst skal indeholde. Derfor er det vigtigt for os som udviklere til hjemmesiden at stille kritiske spørgsmål, identificere manglende funktionaliteter, og foreslå forbedringer for at fuldende projektet.

Baggrund

Virksomheden: Olsker Cupcakes er en nystartet virksomhed baseret på Bornholm, der har specialiseret sig i produktion og salg af cupcakes. Med en dybdeøkologisk tilgang har de skabt en unik forretningsmodel, der appellerer til en bred kundebase, herunder hipstere fra storbyområder som København. Virksomheden står nu over for behovet for at udvide deres rækkevidde og effektivisere deres salgsproces gennem udviklingen af et nyt website.

Kundens krav: Kunden har udtrykt et behov for et webbaseret system, der muliggør en række specifikke interaktioner for både kunder og administratorer. I brede termer inkluderer disse krav muligheden for at bestille og tilpasse cupcakes (valg af bund og topping), oprette en brugerprofil for at gemme og betale for ordrer, administrere ordrer og kundekonti, samt en visning af ordrelinjer i en indkøbskurv for at se den samlede pris. Det understreges, at systemet skal være brugervenligt og tilgængeligt for både kunder og administratorer, med fokus på enkelthed og funktionalitet frem for et overvældende antal funktioner fra starten.

Dette projekt er således ikke kun en teknisk opgave, men også en mulighed for at bidrage til væksten af en spirende virksomhed ved at levere en digital løsning, der matcher deres unikke brand og kundetilgang.

Teknologivalg

Vi har skrevet vores Java kode med disse specifikationer: Maven 4.0.0, xml 1.0, encoding UTF=8, html, css, jdbc, Thymeleaf 3.1.2 RELEASE, Thymeleaf extras version 3.0.4.RELEASE Javalin 6.1.3, Javalin-rendering.version 6.1.3, Maven compiler 17, junit.version 5.10.2, hamcrest.version 2.2, postgresql.version 42.7.2, hikariCP.version 5.1.0, jackson.version 2.17.0-rc1, slf4j 2.0.12, 1.0 - SNAPSHOT, Jetty i IntelliJ IDEA 2023.2.5 (Ultimate edition) Build #IU-232.10227.8.
Runtime version 17.0.9+7-b1000.46 amd64

Vi har brugt pgAdmin 4 version 8.3 til at skrive vores SQL og til at holde på vores database
Vi har desuden brugt Figma version 116.17.11

Krav

For Olsker Cupcakes skal det nye website repræsentere en teknisk forbedring og et bedre værktøj der kan udvide og hjælpe virksomhedens potentiale.
Virksomhedens håb med dette projekt kan opdeles i nøgleord der kan forklare hvad systemet skal tilføje til deres forretning.

Forbedre brugervenlighed og kunde tilgængelighed
Olsker Cupcakes vil gerne gøre det nemmere for deres kunder at bestille og tilpasse de forskellige cupcake toppings og bunde efter kundens individuelle præferencer.
Dette burde gerne øge kundetilfredsheden, tiltrække flere kunder, skabe mere brugervenlighed og gøre det mindre tidskrævende for kunden at bestille deres cupcake.

Administrator og salg
Olsker ville gerne have at vi digitaliserede deres ordrer og administrationsprocessorer, og grunden til at de gerne ville have at vi gjorde det, var at de gerne ville have en hjemmeside med færre fejl og

mere præcis ordrehåndtering, så de kunne holde styr på alle deres ordrer, og gøre det nemt at administrere deres kundekonti, hvilket ville hjælpe personalet og produktiviteten.

Branding

Ligesom alle firmaer, håber Olsker Cupcakes selvfølgelig på at dette system er en mulighed for dem at styrke deres brand og gøre dem mere konkurrencedygtige på deres markedsposition, ved at lave en platform der viser deres værdier og deres kvalitetsstandarder. Sørge for at folket ved at de står inden for Dybdeøkologi.

Overblik

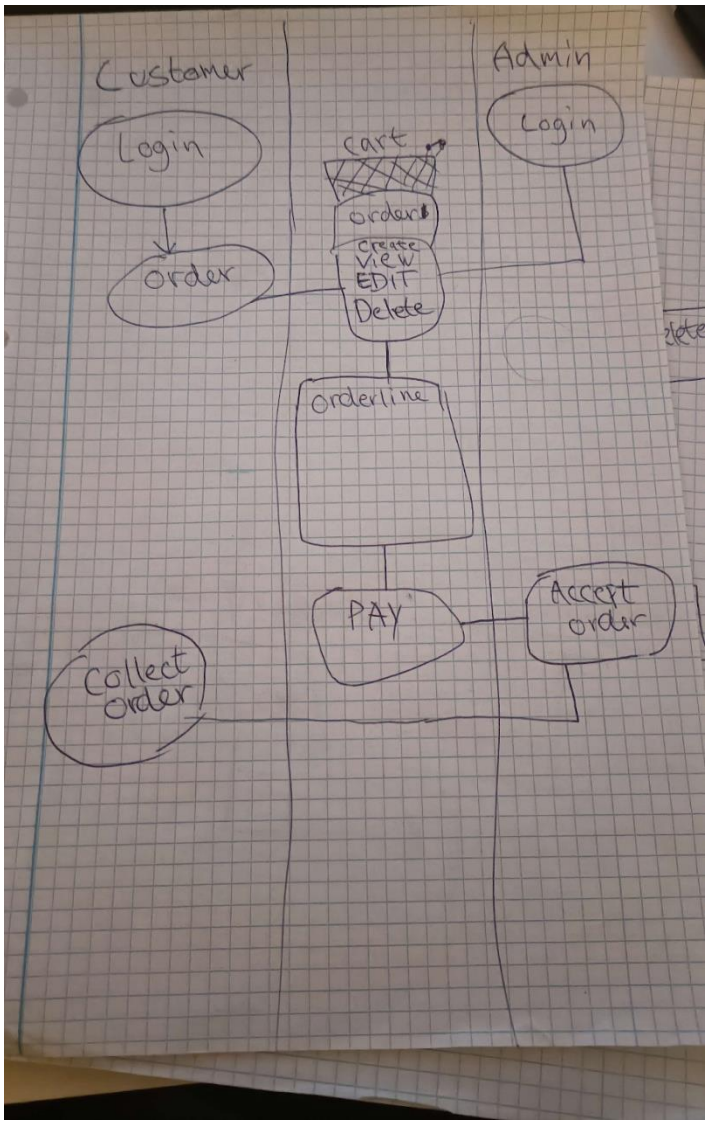
De ønsker også at systemet kan give dem et bedre overblik over det hele, så de ville kunne håndtere flere og større kunder uden at det hele falder sammen pga. deres system ikke kan følge med.

Alt i alt skal systemet repræsentere Olsker Cupcakes ved at integrere teknologiske løsninger for at skabe en kundeorienteret og effektiv virksomhed og mest af alt være dynamisk.

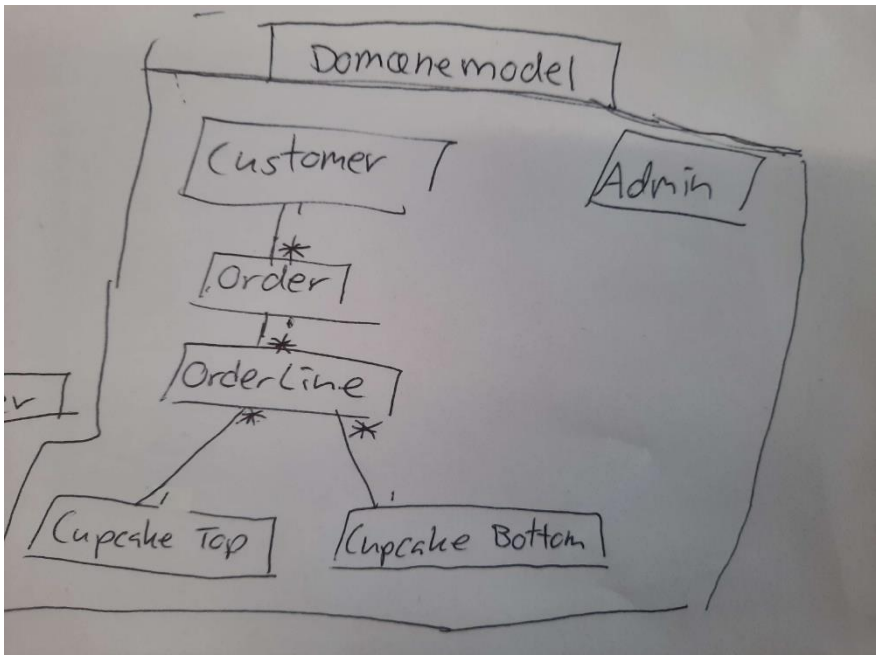
User stories:

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 Postgres, 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 ordrelinie 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.

Aktivitetsdiagram



Domænemodel:



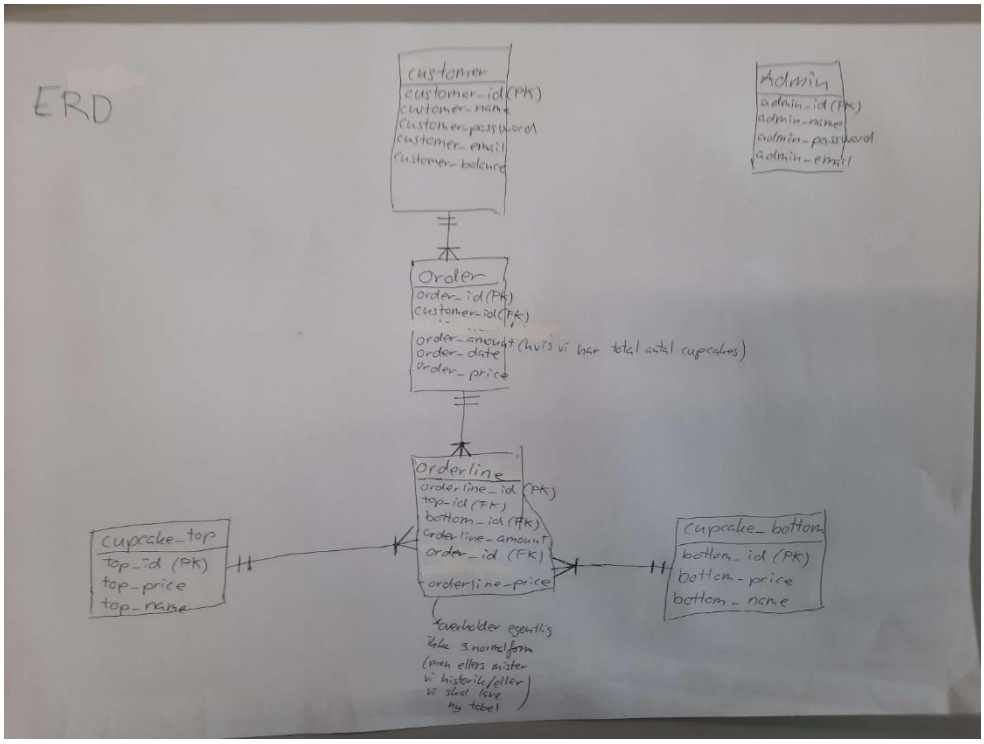
Til at starte med lavede vi en domænemodel, så vi kunne få overblik over de entiteter/klasser vi gerne ville implementere i Java. Vi havde mange overvejelser omkring hvorvidt vi skulle separere Customer og Admin eller ej. I starten havde vi for eksempel tænkt, at vi kunne lave en User entitet der havde attributen "user_role", med typen String der ville have to mulige roller - Customer og Admin. Vi besluttede os dog endeligt for at separere dem helt i hver deres entitet, hvor Admin ikke har nogen direkte forbindelse til de andre entiteter, da kravene reflekterede at Admin skulle være mere overvågende og have overblik over kundernes handlinger, frem for at Admin også selv skulle kunne bestille cupcakes. Customer skulle dog være den entitet, en kunde kunne logge på siden via, og kunne bestille cupcakes igennem. Vi vurderede derfor at de ville kræve forskellige metoder, og at de heller ikke rigtig delte nogle metoder, så vi tænkte at det ville være nemmest at separere dem.

Customer har her en en-til-mange relation til Order, fordi en kunde skal kunne lave mange ordrer, mens en ordre kun kan høre under én specifik kunde. Order repræsenterer her den overordnede ordre, der indeholder ordrens nr, et kundenr og en dato samt ordrelinjer, som OrderLine tager sig af. OrderLine snakkede vi om skulle være et mellemled mellem Order og de to Cupcake-entiteter - Cupcake Top og Cupcake Bottom. OrderLine er hvor cupcaken bliver "samlet", fordi at OrderLine skal indeholde numre der henviser til fx hvilket specifikt topping vi har med at gøre. Fx hvis en kunde vil have

en mandel-cupcake med blåbær-topping på, skal vi kunne samle disse to informationer i OrderLine, som så kan samle de to entiteters individuelle pris, og antallet af bestilte cupcakes med denne kombination. Derefter kan vi skrive disse informationer i ordrelinjer, som kan videregives til Order, når vi vil samle vores cupcake-kombinationer i en ordre. Således vil vi kunne lave en ordre med underliggende ordrelinjer. Order og OrderLine's relation er derfor også en-til-mange, da en ordre kan have mange ordrelinjer, mens en ordrelinje er forbundet specifikt til en ordre.

Selve Cupcaken har vi valgt at dele op i to entiteter - Top og Bottom. Cupcake Top repræsenterer cupcakens topping, som kunden vælger, mens Cupcake Bottom så er cupcakens bund. Vi har skilt dem ad fordi at kunden vælger bund og topping separat, derfor giver det også mening at de kan oprettes separat, før at vi kan sætte oplysningerne sammen via OrderLine. Deres relationer til OrderLine er for dem begge Cupcake-entiteter en en-til-mange-relation, da en OrderLine kun kan have en Cupcake Top og en Cupcake Bottom, mens både Cupcake Top og Cupcake Bottom kan indgå i mange OrderLines.

ERD



Efter vi havde lavet vores domænemodel, lavede vi et Entity Relationship Diagram (ERD), som vi ville bruge til modellering af hvordan vores database skulle se ud. Vi arbejdede videre fra de entiteter vi havde lavet i vores domænemodel - customer, admin, order, orderline, cupcake_top og cupcake_bottom. Alle disse entiteter laver vi til hver deres tabel i databasen. De har alle et unikt autogenereret id som deres eneste primære nøgle.

Admin og customer minder meget om hinanden, da de begge har til fælles at de skal bruge en email, et navn og et password for at de kan oprette sig som konto på siden. Customer har dog udover disse også en "customer_balance", som er hvor mange penge kunden har stående til at handle for - fx 1000 kr. - som er hvad kunden betaler med, når der købes en cupcake.
Order har udover primærnøglen "order_id" også "order_amount", "order_date" og "order_price", som er hvor mange cupcakes der samlet ligger i ordren (dvs. antallet af cupcakes i alle ordrens ordrelinjer), datoen for ordrens oprettelse og ordrens pris (som også er samlet fra alle ordrelinjerne). Desuden har order-tabellen også en fremmednøgle, nemlig "customer_id", som vi bruger til at forbinde til customer-tabellen med, fordi vi gerne vil kæde hver ordre på den kunde, som har oprettet den.

Orderline/Productline - orderline-tabellen er den meste komplekse af alle tabellerne, da denne tabel er det helt store knudepunkt for det hele, da cupcake_top og cupcake_bottom samles her. Orderline giver desuden mange informationer videre til order, og har derfor også "order_id" som fremmednøgle,

der forbinder orderline til order-tabellen. Man kunne evt. også godt have lavet en fremmednøgle for orderline i order-tabellen, men det gjorde vi ikke, fordi orderline er afhængig af order, mens order også skal kunne fungere i en bredere sammenhæng, så order er ikke lige så afhængig af orderline, som orderline er af order. Desuden har orderline også to andre fremmednøgler - "top_id" fra cupcake_top og "bottom_id" fra cupcake_bottom, som sørger for at en orderline kan indeholde kundens valgte topping og cupcake-bund, når der bestilles en cupcake. Vi har også "orderline_amount" og "orderline_price", hvor vi tæller mængden af cupcakes med den valgte smagskombination, der er bestilt, og selve ordrelinjens samlede pris. Men orderline_price afhænger dog af prisen på den valgte topping og den valgte bunds priser, der så bliver lagt sammen til en samlet pris. Derfor er orderline_price ikke 3. normalform, fordi den afhænger af top_id og bottom_id, og ikke bare primærnøglen. Vi har valgt alligevel at gøre dette, da vi ellers ville miste historik, hvis vi ikke gemmer priserne andre steder end i cupcake_top og cupcake_bottom, fordi at gamle ordrers priser vil komme til at ændre sig, hvis vi ændrer cupcake-priserne senere. Det giver jo ikke mening, så vi kan ved at have prisen gemt i orderline, undgå det. Samtidig kan vi også slippe for at lave en ny tabel kun til orderline_price, som ville blive overflødig ift. hvor lille funktion den ville have.

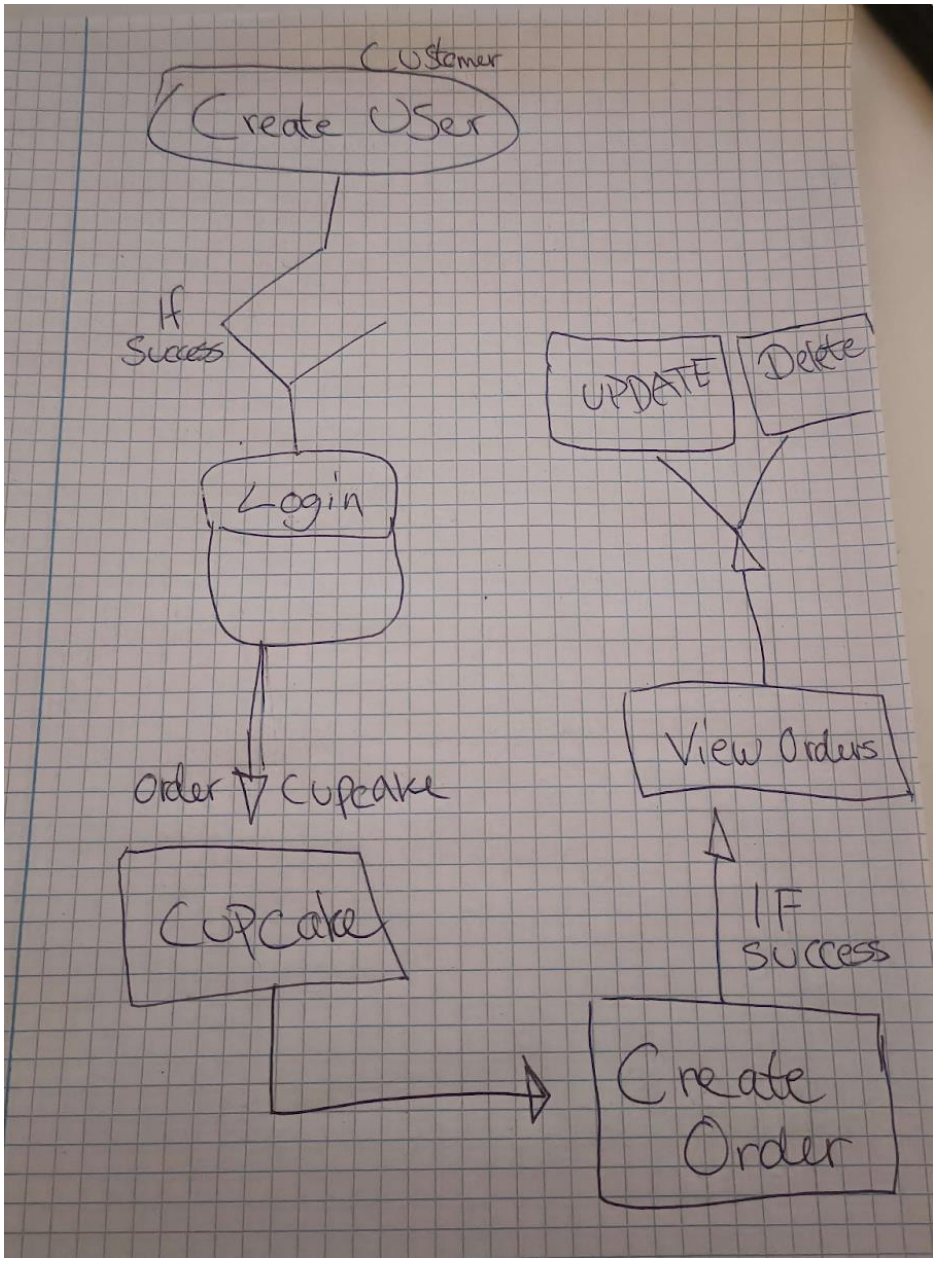
En ting vi kæmpede med senere i projektet var at orderline og order-tabellerne navnene mindede meget om hinanden, hvilket endte med at skabe meget forvirring. Vi endte derfor med at ændre databasen og Java-entiteten Orderline om til Productline for at mindske forvirring, derfor er ERD-billedet fra databasen også med productline i stedet for den orderline som vi havde brugt på vores papirtegning, da vi lavede den originale version af ERD-diagrammet.

Cupcake_top og cupcake_bottom er ligesom i domænemodellen separerede, men de minder om hinanden, da de begge har autogenerede primærnøgler, en pris og et navn som kunne være fx Chocolate. Så de to tabeller kan det samme, de repræsenterer bare henholdsvist cupcakens topping og bund.

Ift. tabellernes indbyrdes relationer med hinanden er de nærmest det samme som domænemodellen, altså at admin ikke har nogen relationer, da admin er mere overvågende frem for at den deltager i cupcake-bestillinger. Alle andre tabeller som i domænemodellen er forbundet med en-til-mange relationer, beholdes også her, bare som "one-and-only-one to many"- relationer, som er en mere udvidet udgave. Dvs. at customer kan have mange orders, men order kan kun have en og kun denne ene kunde. Sådan forløber det også med de andre tabeller: order kan have mange orderlines, en orderline kan kun

forekomme i en order. Cupcake_top og cupcake_bottom kan ligge i mange orderlines, men en order-line kan kun have en cupcake_top og en cupcake_bottom. Således forløbes vores ERD-diagram.

Navigationsdiagram



Særlige forhold

- Vi gemmer UserID i session for at holde styr på brugeroplevelsen
- Hvis vores system fejler, så kan exceptions forhindre problemet før det opstår
- Vi validerer på vores create Customer om de har skrevet det rigtige password to gange.
- Vi bruger prepared statements
- Customer og Admin er separerede fra hinanden.

Status på implementation

Vi nåede ikke helt så langt med implementeringen som vi havde sat for. Vi nåede kun at få 3 hele user stories til at virke - US-2, US-3, US-5. US-2 ift. at vi kan oprette en konto/profil, dog fik vi ikke selve betalingsdelen med. US-3 fik vi med, da man jo kan gå ind i databasen og manuelt indtaste en kundes saldo via kolonnen "customer_balance". Til sidst har vi også helt styr på US-5, fordi både Admin og Customer kan logge ind via email og kodeord, og vi kan se mailen på siden efter vi er logget ind. Dog kræver det at vi vælger hvilket logind side vi vil bruge, da admin's login side kræver at vi trykker på [ADMIN] Login.
Men vi er ikke i tvivl om at vi ikke har været langt fra ift. US-1, US-4 og US-6. Vi havde lavet en hel del af kodearbejdet dertil, men vi lykkedes ikke med at få det hele til at køre. Vi kan derfor ikke bestille og betale cupcakes, samt se vores valgte ordrelinjer/Productlines i indkøbskurven, for at se den samlede pris.

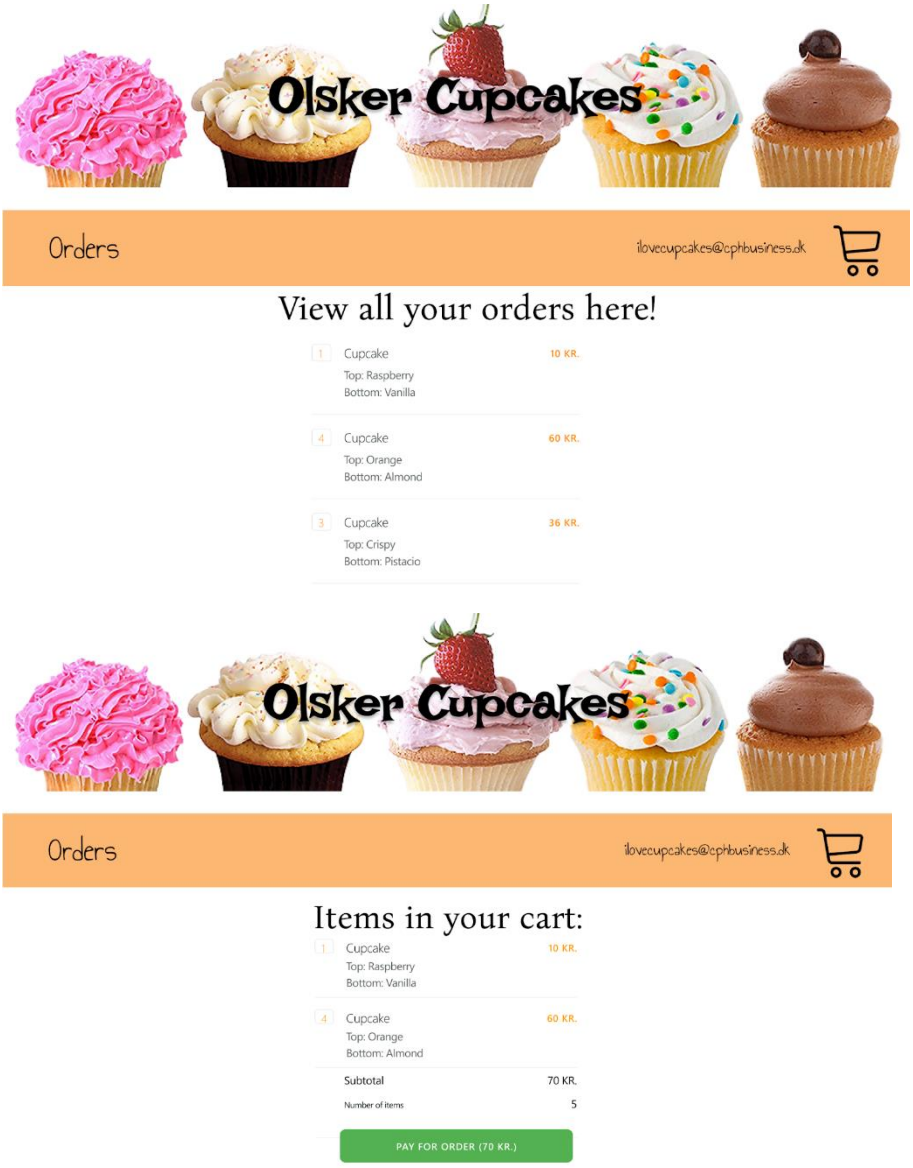
Dvs. at vores login-system fungerer som det skal. Udover det, får vi lov til at komme ind på siden hvor kunden kan vælge en cupcake-kombination, som kunden så kan vælge at bestille. Det er når vi prøver at trykke "bestil" at det går galt, fordi siden bliver smidt tilbage til login-siden, og der kommer fejl-beskeden "There are no cupcakes". Dette kommer fra ProductlineControllerens search-metode. Denne besked kommer fordi vi ikke har lykkedes med at tilføje en ProductlineList fra Productline-Mapperens search-metode ind i Controlleren. Vi lykkedes dog til gengæld at hente værdierne top, bottom og quantity fra html-siden, ift. hvad kunden har valgt af disse muligheder, så de burde have forbundet sig til Controlleren. Derfor går vi ud fra at fejlen må ligge i ProductlineMapperens search-metode.

Vi sad et godt stykke tid og prøvede at ændre search-metoden i ProductlineMapper, for at prøve at lægge en insert query ind i vores sql-statement. Vi mistænker nemlig at da vi ikke har indsat noget i databasen i denne metode, at det måske ville være det, der ikke fungerer, for vi brugte nemlig kun et select query for at kunne hente databasens værdier. Vi har endnu ikke implementeret noget, der løser dette.

Vi har også reflekteret over at vi kunne lave en ny metode i ProductlineMapper, den kunne fx hedde addToDatabase, som kunne tilføje den information, vi har brugt search-metoden til at lede i databasen efter. Vi har dog ikke indsat cupcaken ind i databasen endnu, men her kunne vi så påkalde addToDatabase metoden indeni search-metoden. Så kunne vi lave en insert-query i addToDatabase metoden og få sat vores valgte cupcake ind i databasen. Det er noget vi måske kunne bruge til at finde en løsning, hvis vi ikke havde været i tidsmangel.
Således havde vi mange forskellige mulige løsninger, vi prøvede at implementere, og bagefter så vi koden igennem for at se hvor den ville fejle. Fx fik vi også database-fejl ved implementeringer af andre muligheder. Til sidst måtte vi dog sande at vi ikke kunne nå at få den til at køre.

Udover dette har vi også haft andre refleksioner omkring hvordan vi måske kunne lave de andre resterende user stories. Vi kunne fx have samlet Customer og Admin i en User entitet, hvilket vi havde tænkt på mens vi lavede vores domænemodel. Således kunne vi bruge if-statements omkring user_role i fx de Mappers, hvor vi kunne kræve at kun en Customer eller kun en Admin skal bruge en bestemt metode (fx kun Customer skal bestille cupcakes). Således kunne vi nok havde lavet US-6, det var ihvertfald noget der kunne have gjort processen nemmere. Sådan havde vi også nemmere kunne lave så at Admin også kan logge ind, da den så har samme muligheder som Customer i US-5, fremfor at vi måtte lave en separat loginside til Admin. Hvis vi også havde fået vores Lists til at blive skabt, som vi havde forventet at de ville, kunne vi havde printet OrderLists og ProductlineLists ud på hjemmesiden og således kunne vi lave ordrelister. US-7 og US-9 ville også kræve noget lignende, ift. at en administrator skal kunne få en oversigt over kunderne og deres ordrer og kunne fjerne dem. US-8 ville også kræve at vi kunne fjerne en Productline fra vores ProductlineList. Derved har vi heller ikke nået at implementere alle CRUD-metoderne, da vi ikke har brugt en delete-query. Vi har dog brugt en update-query i payToOrder-metoden i OrderMapper.
Vi kunne også have lavet en paymentController, som vi kunne have brugt ift. bestillinger som vi vil betale for, som så også kunne have lavet en bestillingsliste.

Ift. layout og styling havde vi lavet en mockup i Figma, inden vi gik i gang med kodningen, for at få overblik over hvordan vi ville have at vores hjemmeside ser ud. Mockuppet for de vigtigste sider ser således ud - resten ligger som bilag:

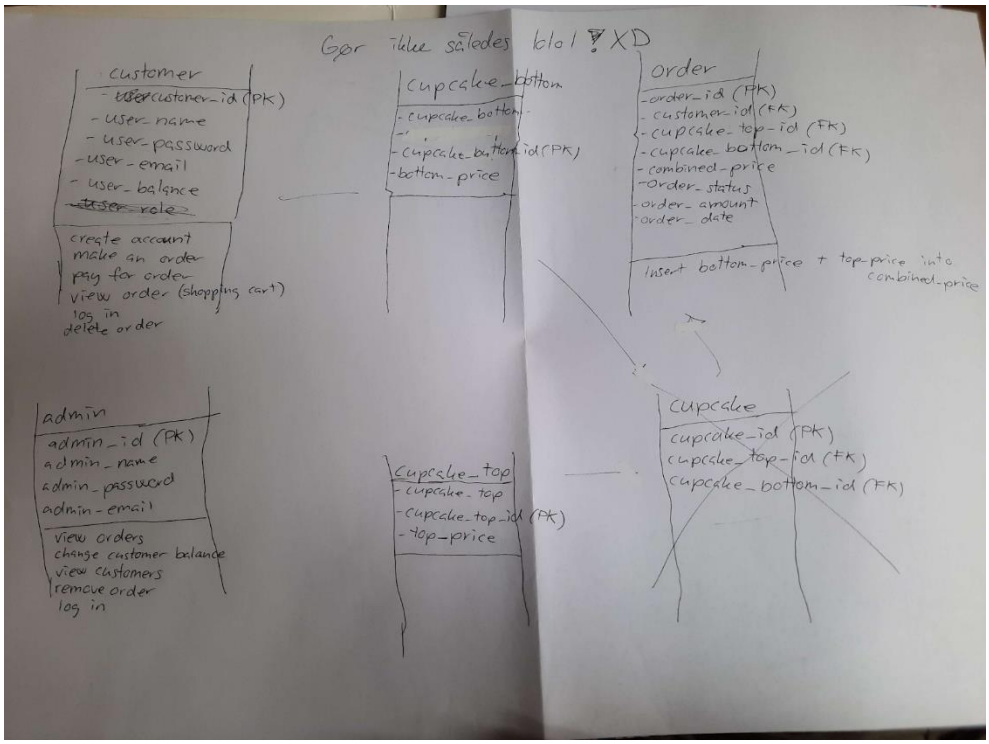


Det var sådan vi helst ville have siden at se ud. Vi har her en login-side, en bestillingsside, en ordre-liste-side og en indkøbskurv-side, samt 2 sider forbeholdt admin, der skal kunne se alle kunderne og deres ordrer og give overblik over salget af cupcakes. Dog måtte vi nedprioritere stylingen på vores sider, da vi ikke havde fået løst vores øvrige problemer ift. user stories, som vi vurderede var vigtigere at fokusere på. Vi fik stadig sat lidt styling på, men det var ikke med det helt samme udseende som Figma-mockuppet. Udover det nåede vi heller ikke at få implementeret alle siderne. Vi fik implementeret login-siden (index.html), en create customer-side (createcustomer.html), bestillingssiden (cupcake.html), og vi fik påbegyndt indkøbskurvs-siden (cart.html). Så der var en del ting mere vi gerne ville have nået, men som vi desværre ikke kunne nå.

Proces

Hvad var vores planer for teamets arbejdsform og projektforløb? Vi startede faktisk lidt med at sætte os ned og snakke om vores user stories og om hvordan vi ville løse dem. Vi snakkede lidt om hvilken kode vi skal skrive for at kunne klare de forskellige stories.
Derefter snakkede vi om at vi skulle lave en domænemodel og et ER Diagram.

Vi startede med at lave vores domænemodel, som vi faktisk lavede forkert, da vi lavede et ER diagram og domænemodel sammen i stedet for domænemodel for sig selv og ER diagram for sig selv (som vist på billedet nedenunder). Men det blev hurtigt rettet op på, og så fik vi lavet vores domænemodel som beskrevet i Domænemodel afsnittet.



Bagefter vi havde lavet domænemodellen, var det første vi snakkede om, at vi ville få vores database op og køre og begynde at skrive vores kode så at vores hjemmeside kunne komme til livs. Og hjemmesiden skulle være i udgangspunkt af vores user stories.
Men da vi først gik i gang med at kode, endte vi lidt med at være lidt all over the place, altså vi fik fordelt vores kode så at vi ikke ødelagde noget eller skrev noget i hinandens ting osv. f.eks. havde vi en der stod for Top og Bottom, og en der stod for både controller, mapper osv.

Men vores problem var lidt at vi ikke helt fulgte vores user stories som vi snakkede om at vi ville gøre og det gjorde lidt at vi løste tingene hist og pist og vi tog rigtig lang tid før vi havde sat en prototype op, så vi havde næsten ingenting at vise, før vi havde fået næsten det hele sat op.

Hvad gik godt og hvad kunne have været bedre?

Altså det der gik godt, var helt klart at vi alle kom med vores input, vi kom alle med vores meninger, vi har alle lagt meget tid i opgaven, og vi har alle været gode til at række ud og hjælpe hinanden.

Vores fordeling af arbejdet var også rigtig god og vi fik virkelig udnyttet at kunne bruge Git til vores opgave så vi kunne alle lave vores egen kode.
Vi fik også fordelt tingene forskelligt fra sidste opgave vi havde, så vi alle sammen fik lov til at prøve at arbejde i de forskellige ting, så det f.eks. ikke var den samme der arbejdede på SQL begge gange.

Det der helt klart kunne have været bedre, var at vi startede ud med at lave nogle fejl med domæne-model og ER diagram osv. og det kostede os en del tid og vi har også tit gjort mange af tingene alt for komplicerede, og vi ender tit med at bruge rigtig meget tid på at snakke i dybden om tingene, hvilket er et dobbeltsidet sværd, da vi har brugt rigtig meget tid på at snakke om tingene. Det har gjort at når vi så sætter os ned og koder igen, har vi lige pludselig lidt travlt, samtidig med at vi har over-kompliceret mange af vores ting. Men vi har taget rigtig meget ved lære af det og ved hvordan vi skal undgå de samme fejl til næste projekt.

Hvad burde vi ændre til næste gang?

Helt klart mængden af tid vi bruger på at snakke om ting for meget i dybden hvor vi slet ikke behøver det, og at undgå at komme i for meget snik snak og ikke altid fokusere på opgaven. Vi vil også fokusere mere på at gå efter user stories og målene med vores opgave, lidt ligesom en opskrift, hvor vi tager den en ad gangen fra toppen af, så vi ikke f.eks. hopper rundt fra user story 1 til story 9. Så vi har et mere decideret mål i vores projekter og ved hvad vi hver især skal sætte os ned og gøre.

Bilag


Olsker Cupcakes
ORDERS VIEW
SHOW ALL ORDERS
CREATE ORDER
VIEW ORDER
UPDATE ORDER
DELETE ORDER


Olsker Cupcakes
CUPCAKES VIEW
SHOW ALL CUPCAKES!
CREATE CUPCAKE
SHOW CUPCAKE
UPDATE CUPCAKE
DELETE CUPCAKE