Cupcake-Rapport

Gruppemedlemmer:

Alexander: cph-as789@cphbusiness.dk, tlirtie

Nicolai: cph-nk267@cphbusiness.dk, NicolaiKristiansen

Sofus: cph-sl512@cphbusiness.dk, sofusk7cdk

Gruppenavn: Team 17

Hold: Datamatiker B

Tidspunkt for udarbejdelse af projekt og rapport: 24-03-2025 til 04-04-2025

Indholdsfortegnelse

Indledning

Dette projekt omhandler udviklingen af en hjemmeside som kan bruges af en cupcake-virksomhed på Bornholm. Programmet skal have nogle bestemte funktionaliteter som er blevet beskrevet ved hjælp af såkaldte "User Stories", hvilket er et begreb der beskriver nogle forskellige eksempler på handlinger som en bruger skal kunne tage inde på hjemmesiden. Vi har på forhånd fået udleveret en skabelon på en forside som vi kan bruge som inspiration når det kommer til designet på selve hjemmesiden, hvilket giver os en ide om hvordan layoutet skal se ud, inden vi går i gang med prototyper og et eventuelt endeligt design.

Den virksomhed som hjemmesiden skal udvikles til (kunden), er et cupcake-bageri på Bornholm. Denne virksomhed har bedt os om at producere en hjemmeside, hvor deres kunder har mulighed for at bestille cupcakes, som derefter vil blive produceret af bageriet.

Kunden har på forhånd givet os et par krav, som skal overholdes når programmet udvikles for at sikre at programmet fungerer som kunden ønsker. Disse krav kan ses under overskriften "Krav" senere i rapporten.

Teknologivalg:

Under udviklingen af hjemmesiden har vi gjort brug af en del forskellige teknologier for at få tingene til at virke. For det første, er der blevet brugt IntelliJ v. 2024.2.3 til selve koden og til at gemme alt media som f.eks. billeder osv. Der er blevet brugt Azul Zulu 17.0.14 som JDK i projektet. Derudover er der blevet brugt Thymeleaf og Javalin som web-frameworks, der gør det lettere for os at udvikle vores hjemmeside. Til at opbevare data vedrørende brugere, cupcakes og ordrer, gøres der brug af PostgreSQL, som giver os mulighed for at oprette tabeller og opbevare data inde i de tabeller. Programmet ved navn Docker bruges til at oprette en lokal server på vores computere, hvor disse databaser kan ligge og tage imod data, og sende data tilbage til hjemmesiden hvis der er brug for dette. JDBC (Java Database Connectivity) bruges til at forbinde vores Java-kode med vores PostgreSQL-database. I starten af projektet gjorde vi også brug af programmet Figma til hurtigt at lave nogle prototyper af, hvordan vores hjemmeside skulle ende ud med at se ud.

Selve programmet / hjemmesiden er udviklet i 3 forskellige sprog: Java, HTML og CSS. Da hoveddelen af logikken i programmet foregik i Java, giver det selvfølgelig mening af gøre brug af et IDE (Integrated Development Environment) som IntelliJ, der primært er specialiseret til Java. Dog er der stadig mulighed for at bruge andre sprog som ofte bruges sammen med Java, for eksempel HTML og CSS, hvilket vi gør brug af i løbet af

vores projekt. HTML bruges til at lave vores tekst, knapper, tekst input-felter osv. Derefter inddrages der noget CSS til at få teksten og knapperne osv. Til at se pænere ud, og passe vores ønskede design som vi lavede i Figma-prototyperne.

Krav

I vores projektbeskrivelse, fik vi nogle krav givet af kunden i form af "User Stories" som vores hjemmeside skal overholde. Kravene kan ses forneden.

User stories (funktionelle krav)

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 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.

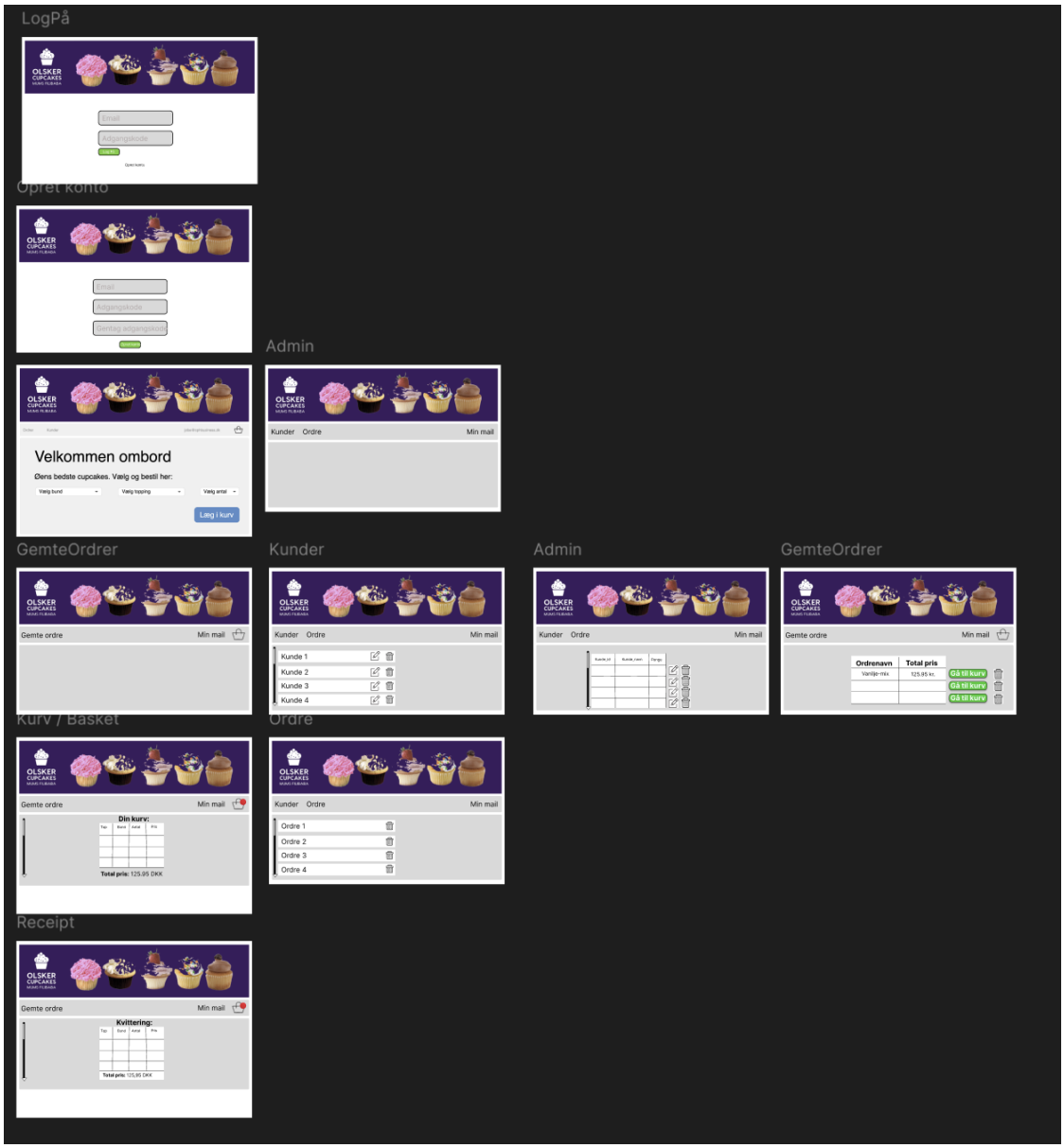
US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde udgyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Ikke-funktionelle krav

Der laves en mockup i Figma eller lignende, som viser de websider den færdige løsning kommer til at bestå af.
Ordrer, kunder og øvrige data skal gemmes i en database.
Databasen skal normaliseres på 3. normalform med mindre andet giver bedre mening.
Kildekoden skal deles på GitHub.
Det færdige produkt skal udvikles i Java 17, Javalin, Thymeleaf template engine, Postgres Database, HTML og CSS.
Websitet skal helst kunne fungere tilfredsstillende både på en almindelig skærm og på en mobiltelefon (iPhone 12 og lignende). Hvis det volder problemer, så lav kun jeres løsning til en laptop.

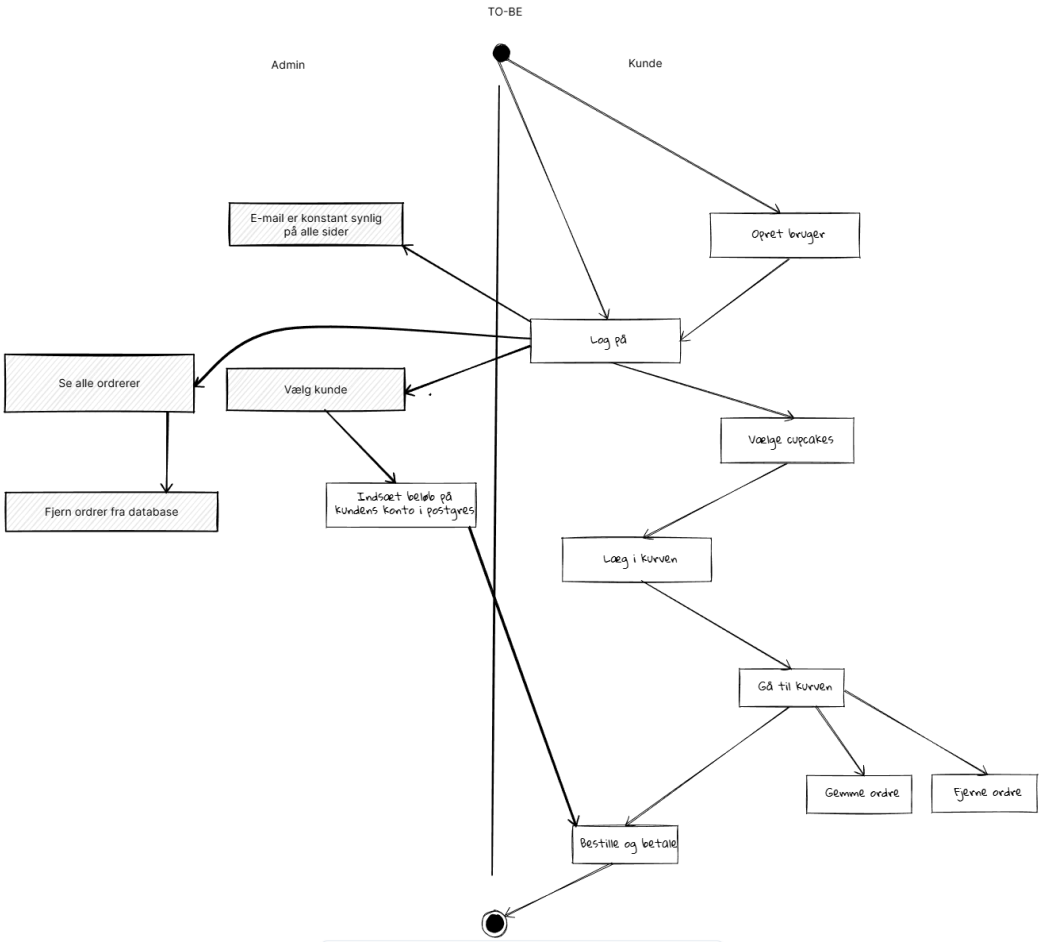
Figma Mockups

Forneden kan der ses et skærmbillede af alle vores prototyper fra Figma. Disse prototyper blev brugt under udviklingen af programmet i HTML og CSS, hvor vi prøvede at efterligne udseendet på prototyperne så meget som muligt. Under billedet er der også et link til prototyperne, hvis der er brug for at se nærmere på dem



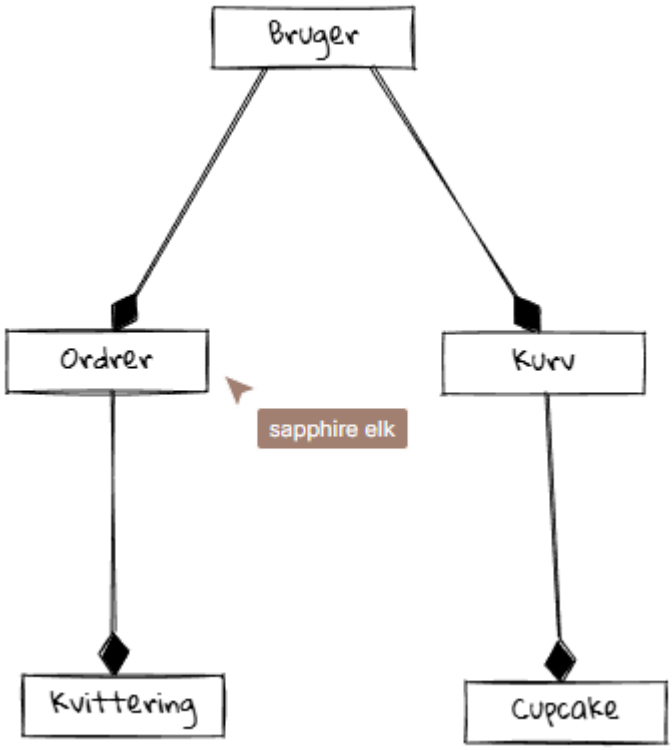
Link til Figma mockups:
https://www.figma.com/design/u70Caf9znBaZnj5ovMKL4K/Untitled?node-id=0-1&t=5SBj3ntYehYd65bW-1

Aktivitetsdiagram

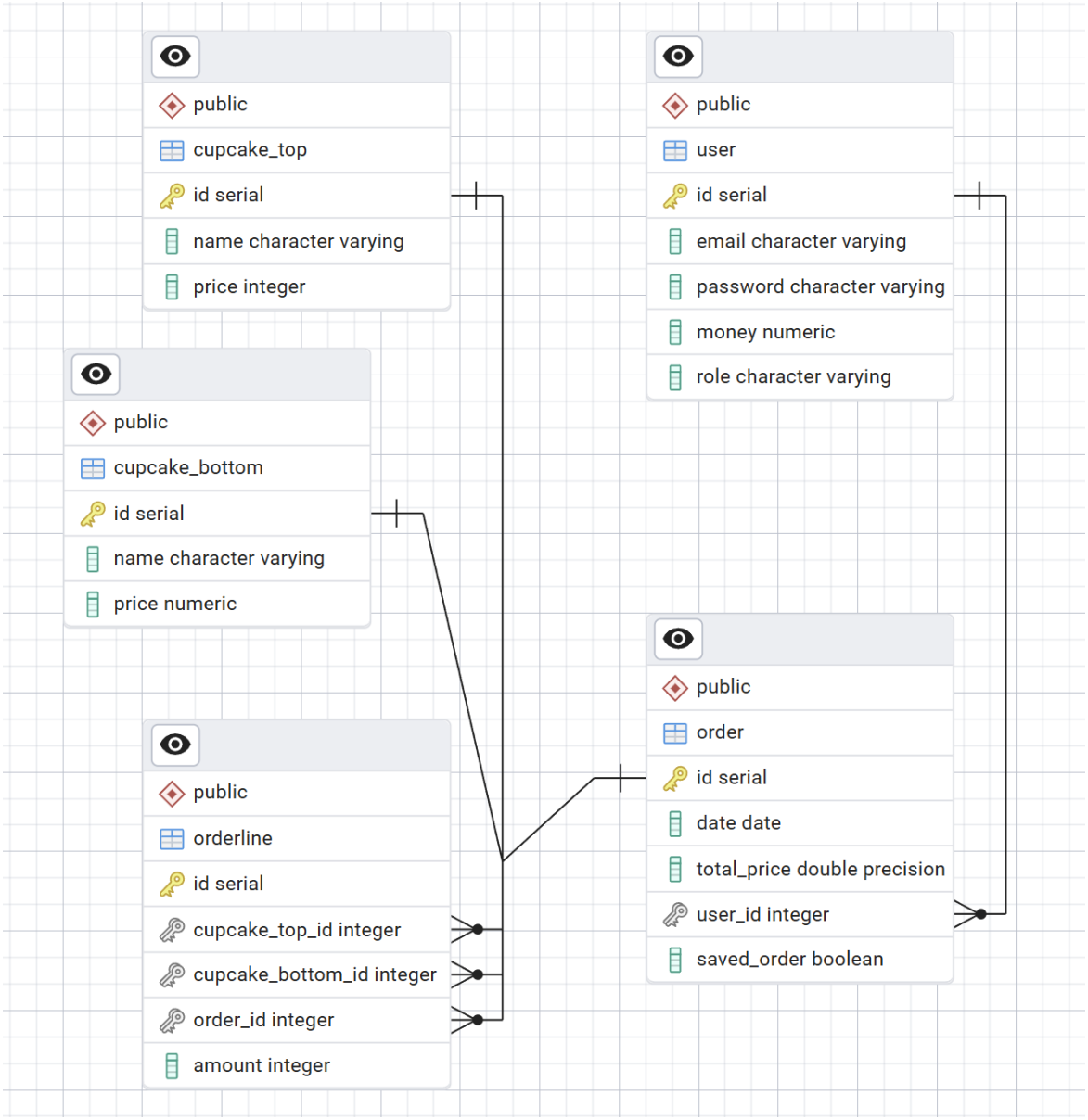


Domæne model og ER diagram

I starten af projektet lavede vi en domænemodel og et ER-diagram til at få et overblik over hvordan vi ville overordnet strukturere vores projekt. Både domænemodellen og ER-diagrammet kan ses her forneden.

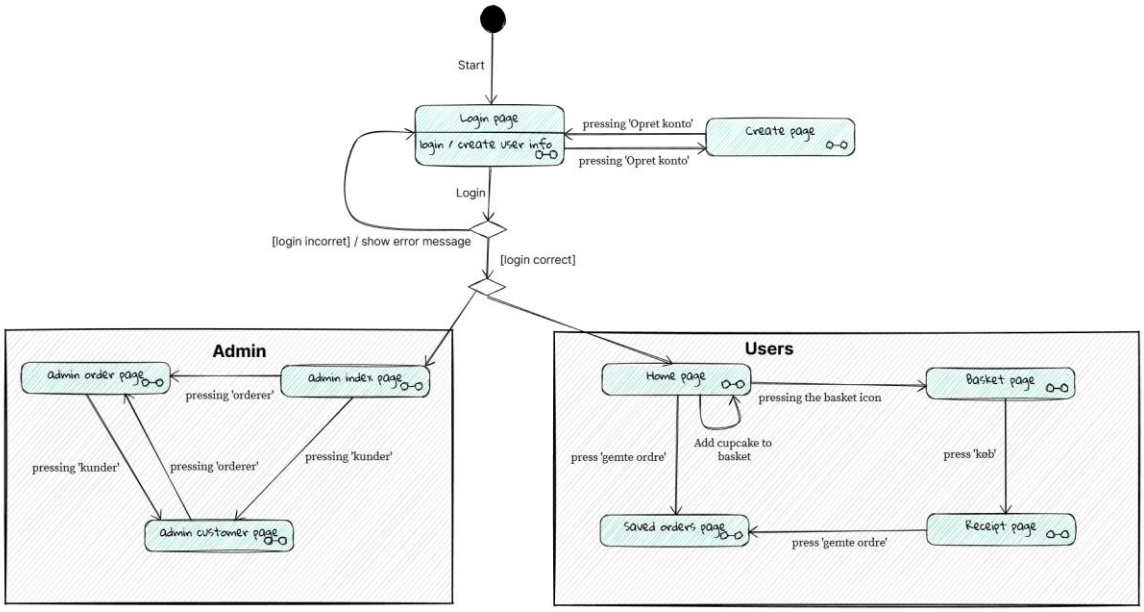


Entity relation diagram (ER-Diagram):



Dette er vores ER diagram. Vi har lavet 5 forskellige tabeller med relationer. Vores 'user' har 1 til mange relation til 'order' da en bruger kan have flere ordre. En ordre kan have flere 'orderline' da en ordre kan bestå af flere cupcake kombinationer. Derudover er der en cupcake_top og cupcake_bottom med en 1 til mange relation til orderline da der kan være man forskellige smage.

Navigationsdiagram



Særlige forhold

- Når man logger ind tjekker vi om brugernavnet og passworded eksistere i vores user table. Og vi bruger et PreparedStatment for at undgå at man ikke kan ændre den String vi indsætter ind i databasens query tool.
- Vores nuværende user er gemt i session. Hvilket bruges i mange funktion til at returnerer user'ens gemte information fra databasen
- Vi sørge for at vores kunder og admin ikke kan ødelægge vores database ved at gøre at de kun kan indtaste de muligheder vi har allerede valgt er okay og så gør det om til en PreparedStatment.
- Hver gang vi skal hente, slette eller modificere noget på databasen bruger vi SQLExecption.
- Vores bruger kan enten være 'user' eller 'admin'. Admin giver dem adgang til at se alle ordre og slette dem, eller de kan se alle kunder og ændre mængden af penge de har på deres konto. Users kan dog kun skabe order og orderline objekter som vil blive gemt på vores database når de køber cupcakes. Alle nye brugere der oprettes, er som default en "User". Den eneste måde at oprette en admin-bruger på, er ved at manuelt lave en i postgres. Der er dog umiddelbart kun brug for en enkelt admin bruger, så det burde ikke være noget problem.

Status på implementation

Vi har overordnet set fået udviklet et program som virker som det skal. Dog er der visse "quality of life" ting der kan tilføjes for at få programmet til at virke bedre for brugeren. Dette er for eksempel en knap der giver brugeren mulighed til at gå tilbage og bestille flere cupcakes efter de allerede har købt, og modtaget kvitteringen. Lige nu, ville man være nødt til at lukke hjemmesiden og åbne den igen for at komme tilbage til starten, eller bruge de browser-indbyggede pile til at gå tilbage. Som bruger af programmet, vil man dog højst sandsynligvis ikke bestille to omgange af cupcakes, det ville selvfølgelig give mere mening at samle dem i en enkelt ordre, så måske er det ikke et alt for stort problem. Dog selvfølgelig noget der burde ændres hvis der var tid til det.

Ikke så lang tid inden det var tid til at aflevere rapporten, fandt vi desuden ud af at vores saved_orders.html ikke indlæser det layout som det er meningen den skal bruge, som ellers virkede før. Derfor ser denne side ikke særlig pænt ud, men hvis man går lidt tilbage på vores GitHub-repository historik, er det muligt at se en version hvor det virker, hvis der skulle være brug for det.

Vi har fået lavet alle CRUD metoderne som skulle virker til vores tabeller. I forhold til admin siderne implementeret vi en update og delete metode. Vi har brugt UPDATE-metoden til at opdatere kundernes mængde af penge og DELETE-metoden til at slette en ordre hvis den er udgået. I forhold til login system bruger vi INSERT-metode til at oprette en ny kunde i databasen og tabellen user.

Vi har stylet siderne så det ligner vores mockups i Figma så meget som muligt. Men hvis programmet skulle sendes til en kunde skulle der højst sandsynligvis en UX designer på, som kunne sørge for at indsamle mere data vedrørende hvad helt præcist det er at vores kunde vil have på hver enkelte side, og hvilke funktionaliteter der burde inkorporeres.

Proces

Da vi startede projektet og havde udviklet nogle Figma prototyper i fællesskab, uddelte gruppen roller baseret på de forskellige gruppemedlemmers styrker og svagheder. Nogle af os fik som opgave at arbejde på selve logikken bag programmet, altså Java-delen, andre fik som opgave at udvikle den grafiske interface, altså ved hjælp af sprogene HTML og CSS. Gruppen valgte at lave møder enten hver dag, eller i hvert fald hver anden dag, hvor der blev samlet op på hvor langt hvert gruppemedlem var nået med deres tildelte opgaver, og hvis et gruppemedlem havde fuldført sin opgave, blev der tildelt en ny.

Vi mener at vores arbejdsproces i teamet gik relativt godt, og vi løb ikke ind i nogen problemer vedrørende fordelingen af tjanser. I fremtiden kunne man overveje at uddele nogle forskellige roller indenfor forskellige sprog til de forskellige teammedlemmer, så alle får arbejdet med forskellige ting, i stedet for at alle bare arbejder med den samme ting igennem hele projektet. Det gik dog meget godt med denne arbejdsfordeling, så det er ikke ligefrem garanteret at vi har tænkt os at ændre arbejdsprocessen i fremtidige projekter.