Cupcake projekt

Gruppe: 11

Natasja Vitoft, cph-nn194@cpbbusiness.dk, github: NatasjaVitoft

Oliver Ravnkilde, cph-op82@cphbusiness.dk , github: Ravnkilde1995

Aflevering: 17/11/2022

Indledning

Cupcake projektet handler om, at vi har fået en opgave fra en kunde, som ønsker at vi laver en hjemmeside, hvor det er muligt at bestille cupcakes. Ift. backend udviklingen benytter vi os af Java og Java servlets til at implementerer hjemmesiden. Ift. frontend udviklingen benytter vi os af HTML, CSS og bootstrap. Vi har fået implementeret de første 6 user stories, hvilket gør det muligt for en kunde at oprette sig som bruger, og derefter logge ind og bestille cupcakes. Det er også muligt for administratoren at logge ind, se bestillingerne og sætte penge ind på brugerens konto. Alt information bliver gemt, hvor vi bruger MySQL workbench som værktøj.

Baggrund

Vi har landet en vigtig opgave fra Olsker Cupcakes. Det er endnu et dybdeøkologisk iværksættereventyr fra Bornholm, som har ramt den helt rigtige opskrift. Et par hipsters fra København har været forbi bageriet, og indsamlet nogle krav og lavet en halvfærdig mockup af en tænkt forside. En mockup er en meget løs skitse, som viser hvordan det færdige website skal se ud. Det er selvfølgelig ikke alt som er med, og som er tænkt igennem, så det er vores opgave at stille spørgsmål til manglende funktionalitet, komme med forslag osv.

Når man som leverandør skal løse sådan en opgave, er det godt at dele opgaven op i små etaper. Så tænk fra starten over hvor lidt vi kan lave for at få den første prototype i luften. Med andre ord: vi skal ikke lade os forblænde af hipsternes farver og striber og mange funktionaliteter. Vi æder elefanten lidt ad gangen.

Teknologivalg

- IntelliJ IDEA 2021.2.4
- Apache Tomcat 9.0.67
- Java 17
- MySQL
- JDBC

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 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).

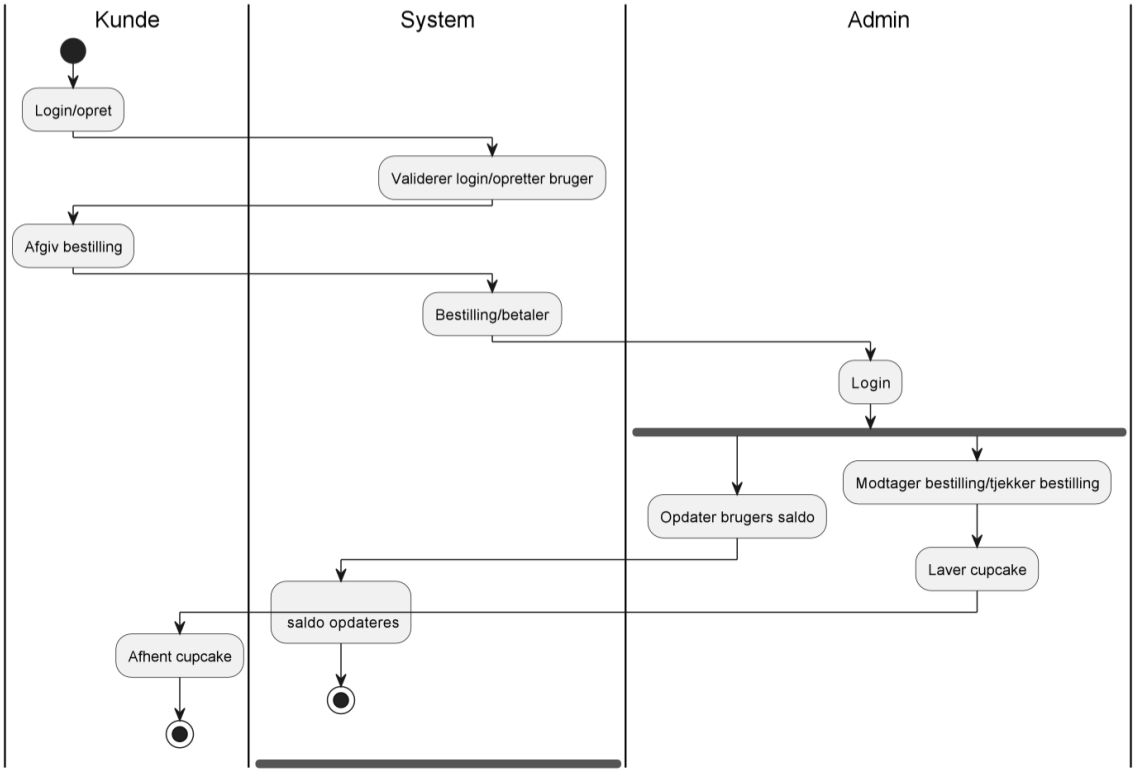
US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

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.

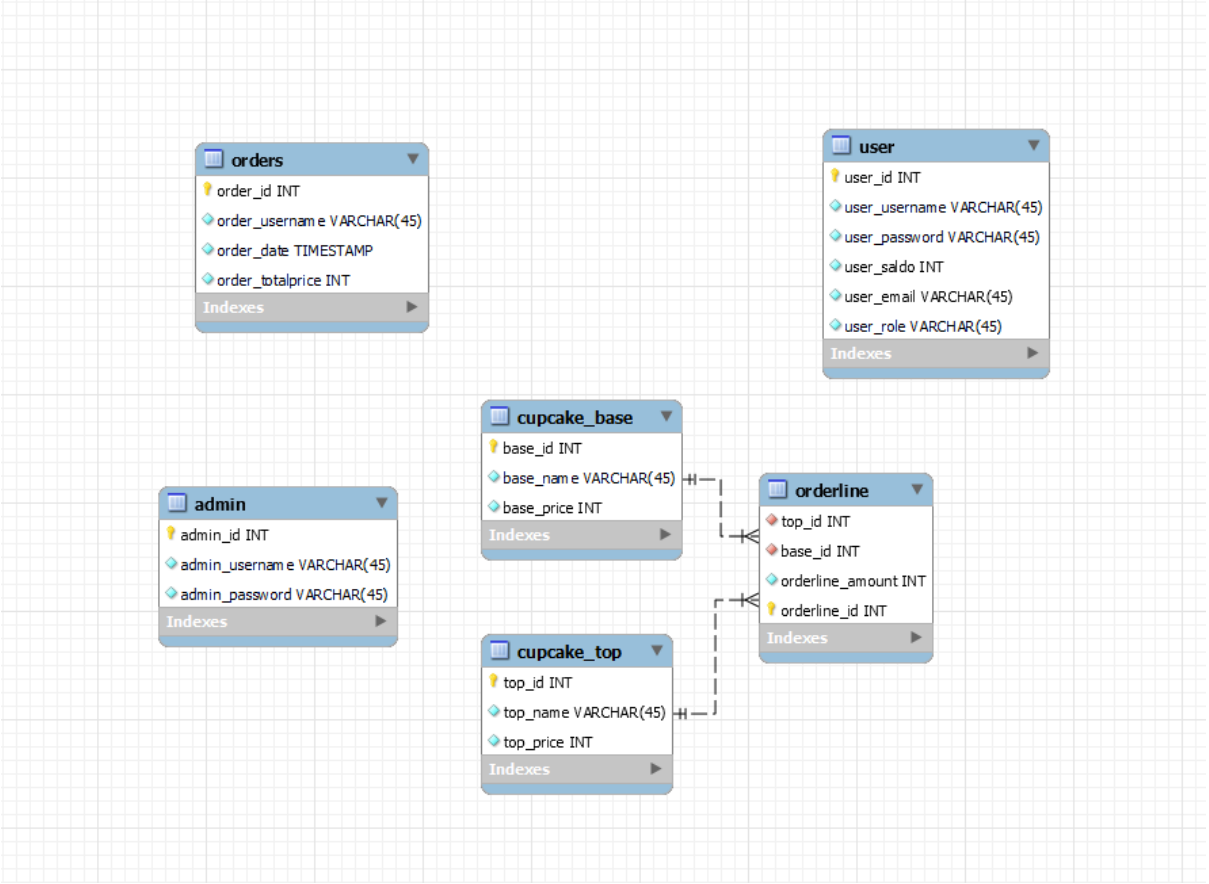
Aktivitetsdiagram



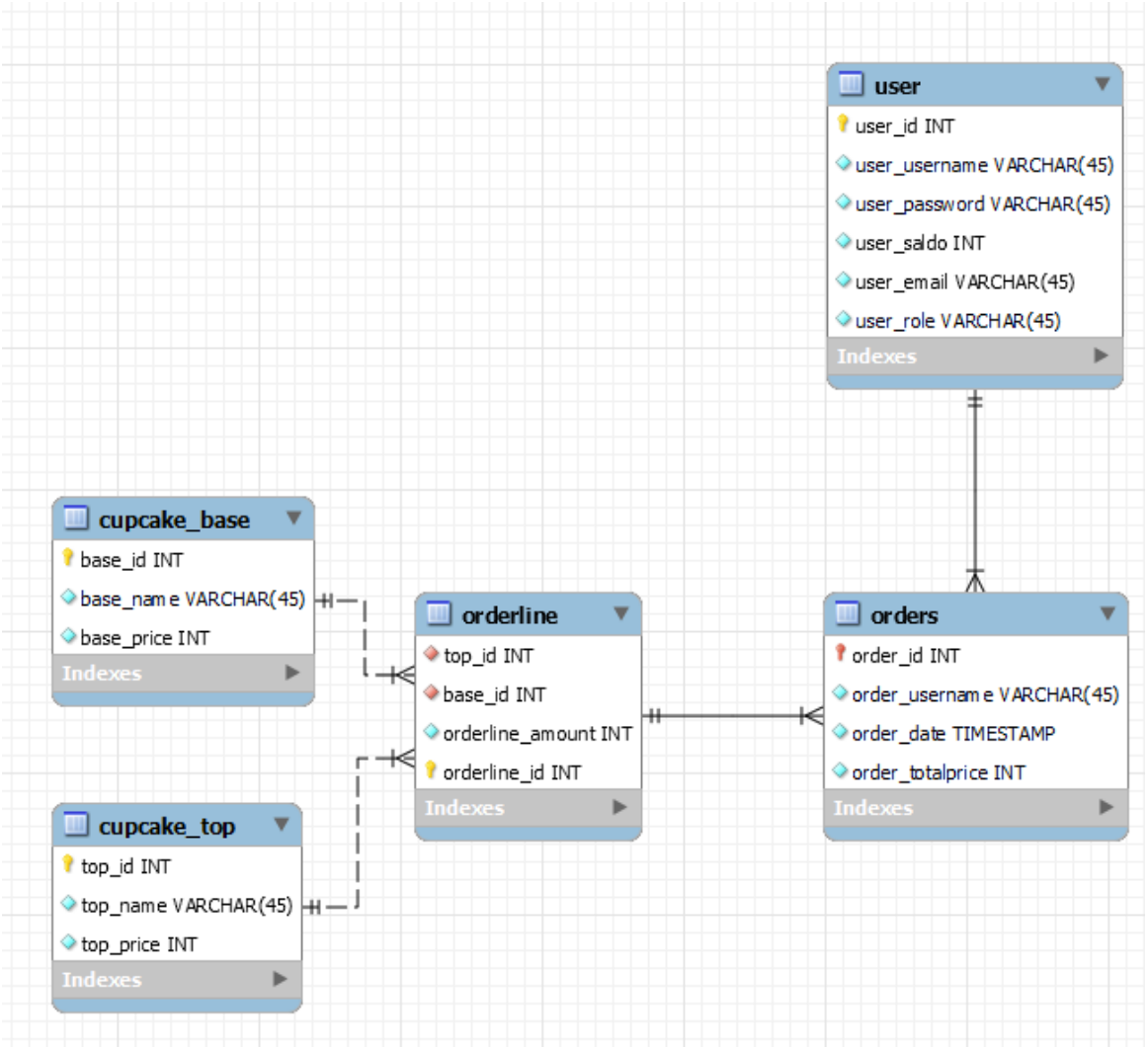
EER diagram

Vi har valgt at lave 5 tabeller i vores database. Det vi i stedet kunne have gjort var at slå admin tabellen og user tabellen sammen, og brugt user_role til at tjekke om hvorvidt det er en admin eller en user som logger ind.

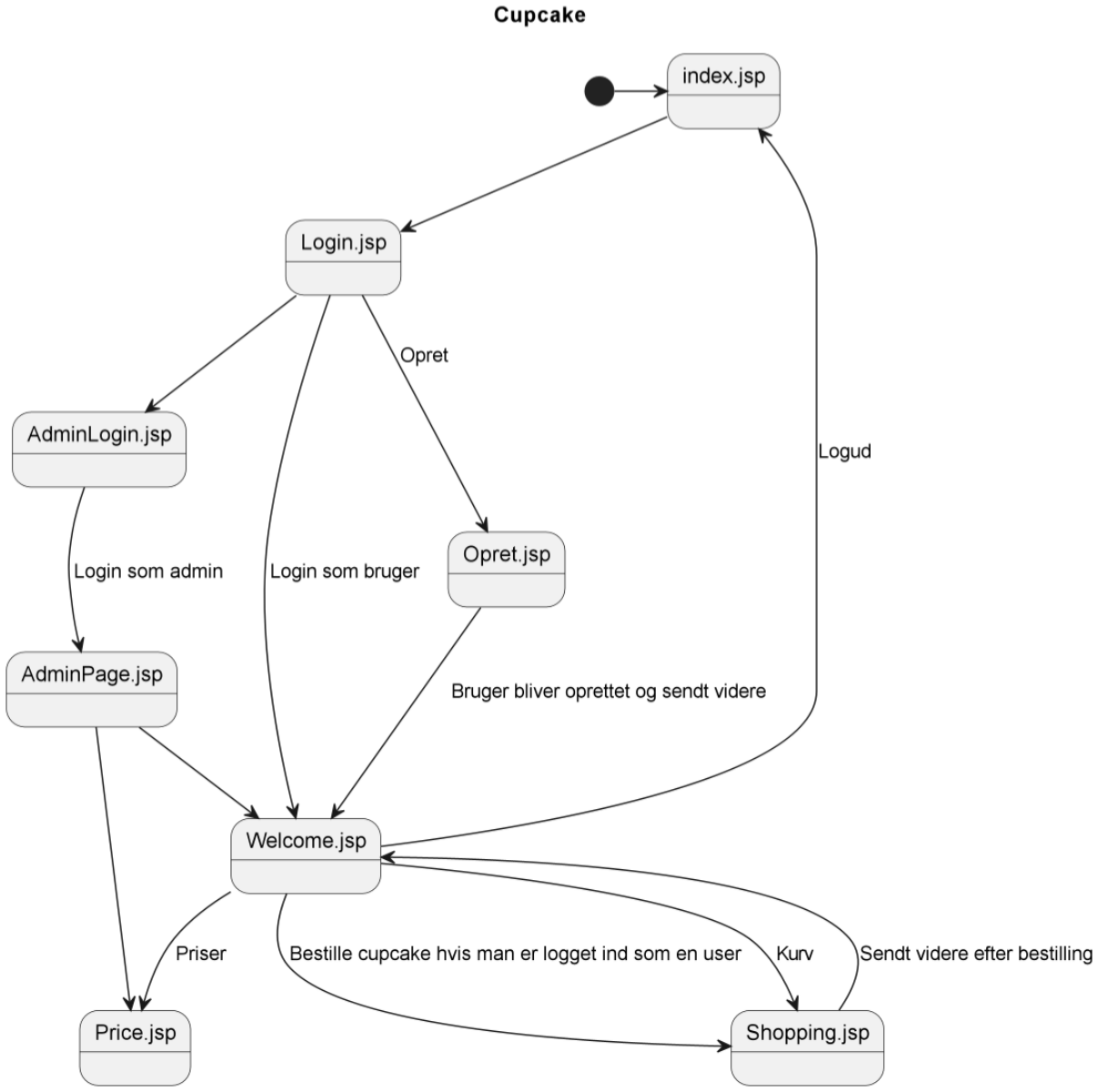
Planen til at starte med var at bruge 3. normalform i databasen, men vi havde nogle problemer med at vi ikke fik lov til at komme data i tabellerne, når vi havde sat en forbindelse. Det lykkedes os kun at lave en forbindelse mellem orderline/cupcake_base og orderline/cupcake_top. Her har vi lavet en 1:n så at ikke kan laves en orderline uden en top og base. Vi ville også gerne have lavet en 1:n relation mellem orderline og orders, så at der ikke kan oprettes en ordre uden en orderline.



Ideelt set ville vi gerne have vores database til at se ud som nedenstående:



Navigationsdiagram



Særlige forhold

- Vi gemmer bl.a. et user object i session når man logger ind, for at vi senere kan få fat i brugerens username, når der skal bestilles og gemmes en ordre - det samme gælder for admin login, hvor admin bliver sendt videre til en anden side, i stedet for at kunne bestille
- Vi gemmer også et user object i session, når der oprettes en bruger
- Vi tjekker brugerinput i login funktionen ved at få fat i parametrene fra input formen, og tjekker om de stemmer overens med databasen. Hvis informationen ikke stemmer overens

med databasen, kastes der en exception med beskeden “forkert brugernavn eller password”. Vi har vores funktion som tjekker i databasen inde i vores usermapper klasse.

- Sikkerhed: Vi benytter os af ukendte inputs fra brugeren ved oprettelse af en ny bruger, og ved login, for at forhindre SQL injection.
- Vi benytter os af hjælpemetoder og en liste, som vi gemmer på cart i session, så vi til sidst kan printe hvad der er i vores indkøbskurv.

Status på implementering

- Det er lykkedes at implementere de første 6 user stories, hvor vi har brugt create, read og update af CRUD operationerne. Vi mangler stadig delete funktionen.
- Vi mangler lidt styling på siderne ift. vores Figma mockup.
- Vi mangler en hjælpefunktion, som eventuelt tester for om et brugernavn er taget, når man opretter en bruger.
- Vi mangler at få helt styr på databasen ift. om sammenhængen mellem vores orderline tabel og orders giver mening og relationen herimellem.
- Vi mangler at få vores “antal” knap til at virke, når man skal bestille en cupcake
- Måske havde det været bedre at admin kan indsætte penge på brugerens konto, i stedet for at opdatere - i så fald havde vi manglet endnu en CRUD operation
- Vi mangler en fejlbesked når en bruger forsøger at betale for en ordre, men ikke har penge nok. Lige nu sendes brugeren videre til forsiden, uden nogen besked om at betalingen ikke er gået igennem.
- Vi mangler en log ud knap når man er logget ind som admin
- Vi mangler en sammenhæng mellem orderline og orders. Lige nu kan man ikke se hvilke orderlines der hører til hvilken ordre.
- Vi mangler også at fokusere mere på Test mappen. Vi har slet ikke lavet nogle test undervejs.

Proces

Hvad var jeres planer for teamets arbejdsform og projektforløbet?

Vi arbejdede ud fra et fælles google docs, hvor vi planlagde hvad vi skulle nå i løbet af dagen, og lavede en oversigt over vores plan med projektet. Vi besluttede os for at planlægge fra dag til dag, fordi det var svært at vide på forhånd, hvor langt vi ville nå. Vi mødtes hver dag enten fysisk eller online, hvor vi uddelegerede opgaverne. Herefter mødtes vi igen, for at gennemgå hvad vi havde lavet og debugge sammen. Vi arbejdede på hver vores branch, og mergerede sammen ved slutningen af dagen.

Hvordan kom det til at forløbe i praksis?

Vores arbejdsstruktur har fungeret godt, og vi har næsten nået det vi skulle hver dag. Der er noget af funktionaliteten som godt kunne forbedres, men da vi kun er to i gruppen, har vi fokuseret på at nå så meget som muligt.

Hvad gik godt og hvad kunne have været bedre?

Vores database struktur kunne godt have været bedre, hvis vi brugte lidt mere tid på det i starten. Vi synes at udfordringen her var, at det var svært at vide fra start hvordan projektet kom til at se ud, så vores database er blevet ændret en del undervejs. Vi kunne bl.a. godt have sat nogle flere krav til normaliseringen og relationen mellem tabellerne.

Det gode ved vores gruppearbejde er at vi begge to har været meget inde over projektet, da vi kun er en to-mands gruppe. Vi har også været gode til hele tiden at samle op på hvad vi hver især har lavet og forklaret det til hinanden. Vi synes at fordelen ved en to-mands gruppe er at man får lært mere, men det kan være lidt svært at nå det hele dog.

Hvad har I lært af processen og hvad vil I evt. gøre anderledes næste gang?

Næste gang vil vi have mere styr på vores database struktur inden vi går i gang med at kode.