Copenhagen Business Academy

Datamatiker

Campus Bornholm

Minervavej 1.

DK-3700, Rønne

Rapport

Projekttitel:	Cupcake
Projektperiode:	04.april.2022 – 22.april.2022
Lavet af:	Timea Ballok
	Otto Prüssing Kristensen
	Nicki Kielsen
	Oliver Fuhrmann Gudbergsen
Vejleder:	Jon Bertelsen
	Nikolaj Brandt Hemmeshøj
Github repo:	https://github.com/nk11-dat/KopOgKageFinal

Antallet af tegn i rapporten: 18812

Indholdsfortegnelse

Indledning

Dette rapport er skrevet af datamatiker studerende på 2. semester. Projektet handle om at lave en website til en lille cupcake butik. Det er vores første web-stack opgave. Opgaven tager udgangspunkt i en startkode, hvor projektet skal udvides fra. Rapporten er lavet til andre datamatiker studerende og skulle gerne vise hvordan er vi er kommet til vores færdig løsning.

Baggrund

Formålet med opgaven er at lave en hjemmeside til en cupcake butik. Butikken ligger i Olsker og vil gerne tage imod web-bestillinger. Butikken forestiller sig at kunden kan logge ind, kan bestille cupcakes på den måde at kan vælge imellem forskellige bund og top. Bageriet vil gerne se bestillingerne på en ordre side, vil kunne godkende en ordre når den er betalt ved fysisk afhentning i butikken og vil kunne slette den bagefter.

Teknologi valg:

Vi har brugt følgende teknologier til at opbygge vores system:

- Java Amazon coretto 11.0.14
- HTML 5
- CSS 2.1
- JSP (?)
- JavaScript ES2015
- Apache Tomcat® 8.0.27
- Git 2.35.1 / Github
- Twitter Bootstrap 5.1.3
- MySQL Workbench 8.0.27
- IntelliJ IDEA 2021.3.3

JDBC:

- JSTL 1.2
- Maven 3.3.2
- Mysql-connector-java 8.0.28

Krav

Krav til systemet kan opdeles til funktionelle og ikke funktionelle krav. Funktionelle kraverne fik vi udleveret som usecases. Vores system skal som minimum lave op til de første 6 krav.

Funktionelle krav

Her under usecases beskriver vi også hvad vi har implementeret, hvordan vi har implementeret det og hvad vi ikke kunne 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.
    - Efter at en kunde har logget ind (eller oprettet sig som ny bruger), kan de gå til bestillings siden ved at trykke på 'bestilling' i navigationsbaren hvor de så kan fylde cupcakes i deres kruv, for så at gå til kurven go bekræfte deres bestiliings. Derfter kan de så køre forbi butikken i Olsker og hente deres order.

- **US-2** Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en en ordre.
    - Hvis man ikke har en bruger kan man på login siden trykke på linket 'opret bruger' for så at blive sendt til en side som lader en oprette sig i systemet, hvor man angiver: navn, email, password og saldo. Denne bruger kan så benyttes til at oprette/bestille en cupcake order, som bliver gemt i databasen.

- **US-3:** Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer.
    - Hvis man har adgang til databasen som adminstrator, kan man ved hjælp af en simpel SQLquery fx: $UPDATE\ user\ SET\ balance\ =\ ?\ WHERE\ user\_id\ =\ ?;$. Eller ved brug af GUI'en efter man har lavet en $SELECT\ *\ FROM\ user;$ og trykket apply.

- **US-4:** Som kunde kan jeg se mine valgte ordrelinier i en indkøbskurv, så jeg kan se den samlede pris.
    - I vores løsning har vi valgt at kalde det for orderitems, men det er det samme som en orderline. Efter en kunde har tilføjet cupcakes til deres kruv, kan de trykke på kurven og se dens indhold og den samlede pris for hele kurven.

- **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).

    - Det er muligt at logge ind på hjemmesiden som enten bruger eller admin med email og kodeord. Den email som bruges til at logge ind vises i navigationsbaren på alle undersider mens man er logget ind.

- **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.
    - Når man er logget ind som administrator har man adgang til oversigt siden som viser alle kunder og de ordre som de har oprettet. Den viser, hvad for nogle cupcakes, antalet af cupcakes, hvornår orderen er oprettet, totalprisen og om den er betalt eller ej.

- **US-8:** Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.
    - Efter man har kommet en cupcakes i kurven så kan man kigge i kurven hvor der er en mulighed for at slette et orderItem/orderLine.

- **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.
    - Efter man er logget ind som admin kan man på oversigt siden, slette en order, en kunde har lagt. Både før og efter man har godkendt orderen.

Ikke-funktionelle krav

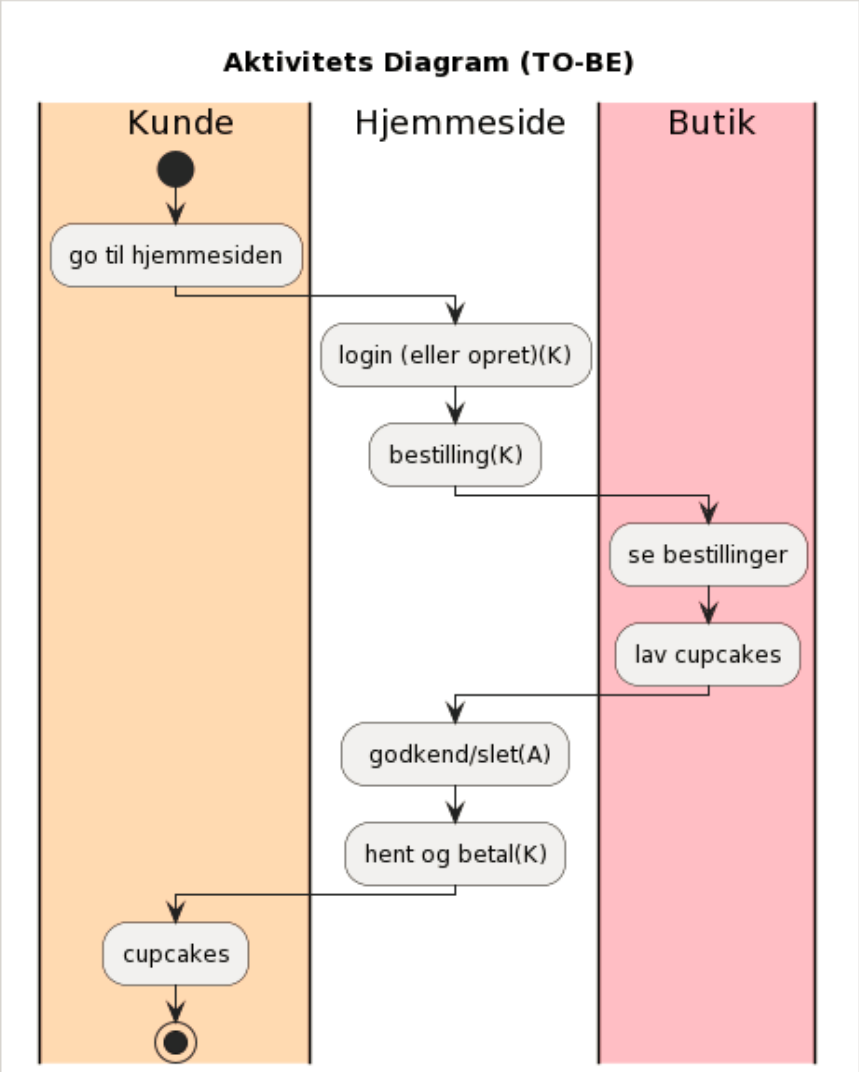
Ikke-funktionelle krav gav en ramme til vores projekt, men sat også en grænse inden for udviklingen. Her under kraverne beskriver vi hvad og hvordan vi har implementeret dem.

- Der laves en mockup i Draw.io eller lignende, som viser de websider den færdige løsning kommer til at bestå af.
    - Vi lavede ikke mockup i Draw.io. Vi tegnede det ønskede side op på tavlen og diskuterede hvordan de forskellige dele skal hænge sammen
- Ordrer, kunder og øvrige data skal gemmes i en database.
- Databasen skal normaliseres på 3. normalform med mindre andet giver bedre mening.
    - Data gemt i 3. normalform i MySQL-Workbench
- Kildekoden skal deles på GitHub.
    - Github repo: https://github.com/nk11-dat/KopOgKageFinal

- Det færdige produkt skal udvikles i Java, MySql, HTML, CSS, Twitter Bootstrap og køre på en Tomcat webcontainer.
    - Vi brugte også lidt Javascript udover overnævnte
- Løsningen skal udvikles med udgangspunkt i vores startkode.
    - Startkoden bygget op efter MVC princippet. Vi udviklede det videre at tage højde for det.

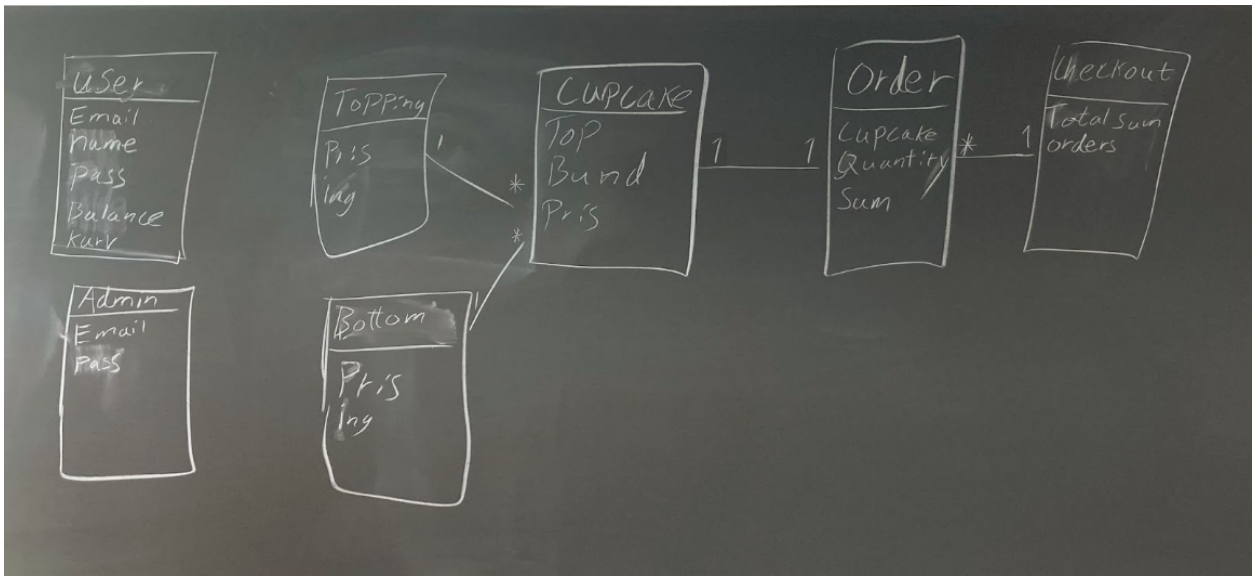
Aktivitetsdiagram

Her kan det ses det ønskede overordnede workflow.



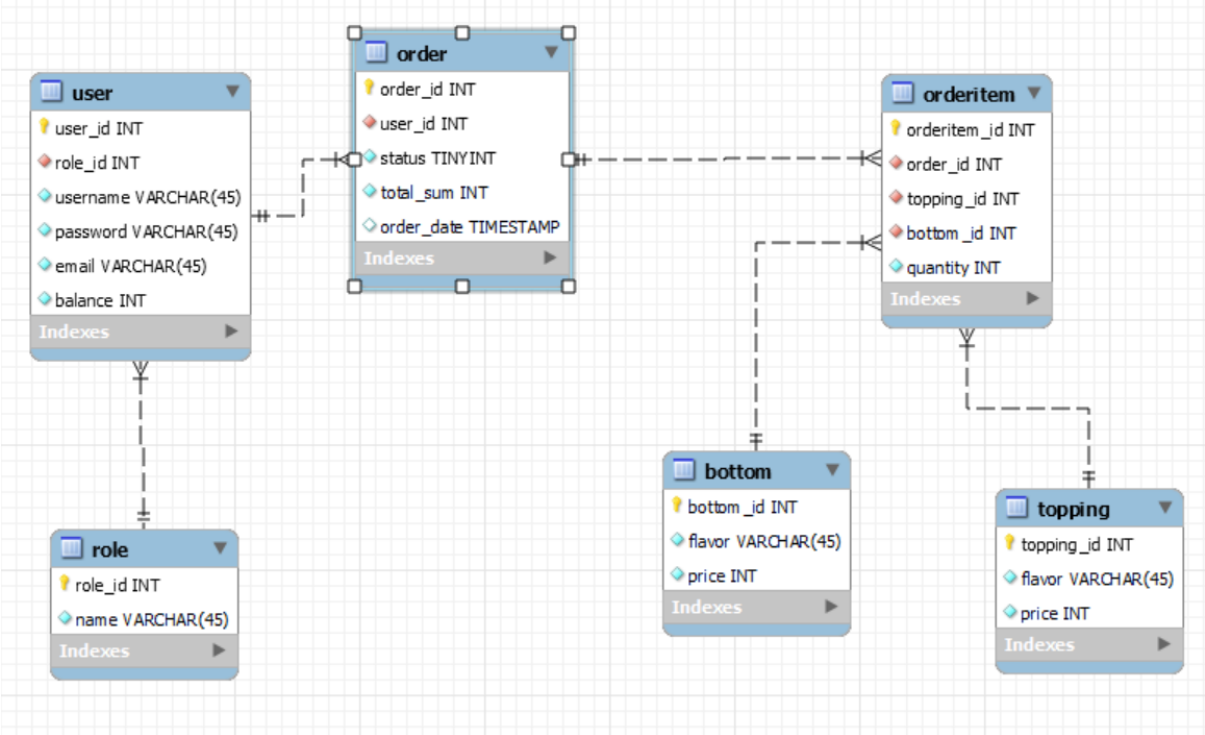
Domæne model og EER diagram

Domænemodel



Vi startede med at tegne cupcake og skrive hvad en cupcake skulle bestå af, som langt hen af vejen var givet ud fra opgavestillingen, for der stod at en cupcake bestod af ‘en valgfri top og bund’, dertil tilføjede vi så en pris. Til dels pga pris attributten fandt vi frem til at Topping(top) og Bottom(bund) skulle komme fra hver deres respektive tabeller, bestående af Pris og ing(ingredienser).
En cupcake kan indgå i en Order(orderitem/line) som består af en cupcake samt et antal af den valgte cupcakes, og så den totalpris for ordren (sum), som er udregnet ud fra cupcakens pris ganget med antallet.
Til sidst fremstår orderene på en checkout(kvittering), som også har en totalsum for alle ordre man har bestilt.
Systemet har 2 bruger typer som benytter sig af websiden, dog på forskellige måder.
User(kunden) som vi der er noget grundliggende information omkring navn som kan bruges til at identificere en kunde når de skal hente deres bestilling, name, password(kode), balance(saldo) og kurv som bruges på hjemmesiden når kundes skal bestille.
Den anden er admin, som også har email og password, der bruges til at logge ind. Men han skal have adgang til en anden version af hjemmesiden end kunden.

EER diagram



Med udgangspunkt i vores domænemodel udarbejde vi et EER diagram, men vi fandt forholdsvis hurtigt ud af at der var mangler i vores domænemodel som der ville være nødvendige for at få en bedre database. Det færdige resultat kan ses på billedet foroven, som er den database vi endte ud med på 3. normalform , den indeholder ingen 1:1 eller mange:mange relationer.

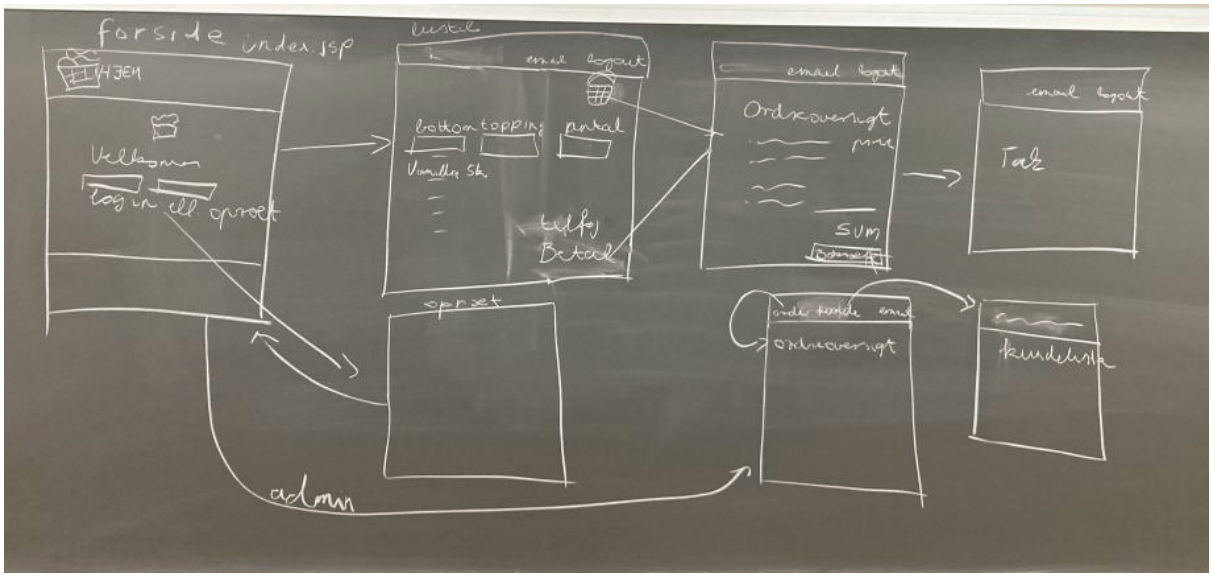
Først of fremmest er alle primærnøgler entydig identifiers og med auto_increment, således at det er let at få fat i en givet entitet eller oprette nye entities.. Vi valgte at lave en user tabel som har et field role_id, som agere foreignkey for role tabelen, der indeholder navnet på rolen, som har typen integer i user tabellen, da vi kun endte med at havde 2 roller i vores database kunne den også havde været en boolean/tinyint. Men vi valgte at holde den som en int hvis man en dag ville udvidede role tabellen med andre roller udover user og admin. Den kan man bruge til at koble role navn på en user ved hjælp af en join operatiion. Dette er en en-til-mange relation, forstået sådan at en role kan havde mange users, der kan fx. være mange kunder.

Order tabellen benytter sig af user_id som en foreginkey, for at kunne identificere hvem der har oprette orderen. Derudover agere Order tabellens primærnøgle som foreginkey i Orderitem(orderline) tabellen. Vi debatterede om vi skulle inkludere order_date eller ej, om der skulle være end eller 2, en til order oprettelse og en til order færdiggjort/betalt. I sidste ende endte vi med at inkludere order_date som en autogenerated timestamp, som så kunne bruges til at sorteres efter. Status fletet var også oppe og vende, om det skulle være en boolean eller en integer, som var tredelt pending, completed, delivered.

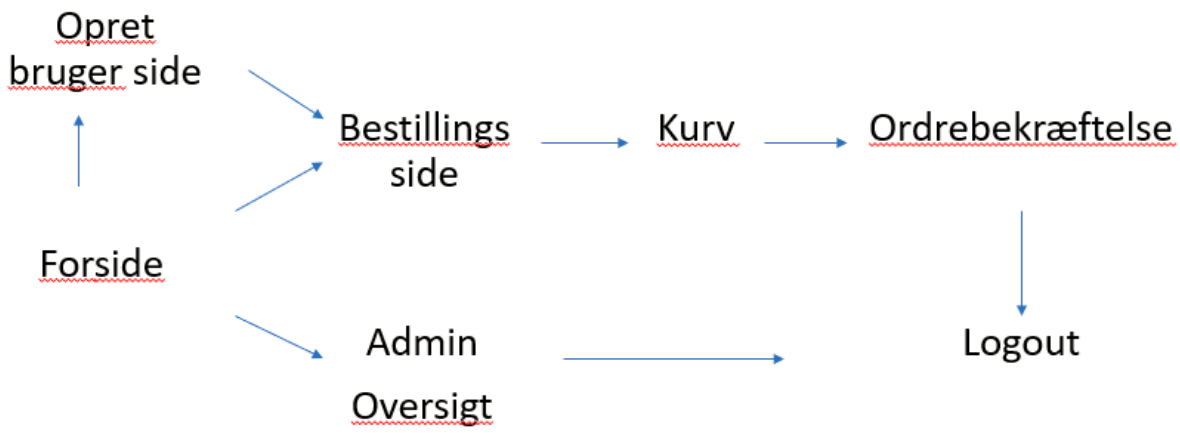
Orderitem tabellen har tre foreginkeys order_id, topping_id og bottom_id som benyttes til at joine de fire tabeller sammen, som tilsammen danner et komplet billede af en bestilling, med hvem den er til (user_id), totalpris(total_sum) og flavor fra bottom og topping tabellerne. Vi gik frem og tilbage mellem navnet til denne tabel, det var enten orderitem, orderline eller cupcake.

Navigationsdiagram

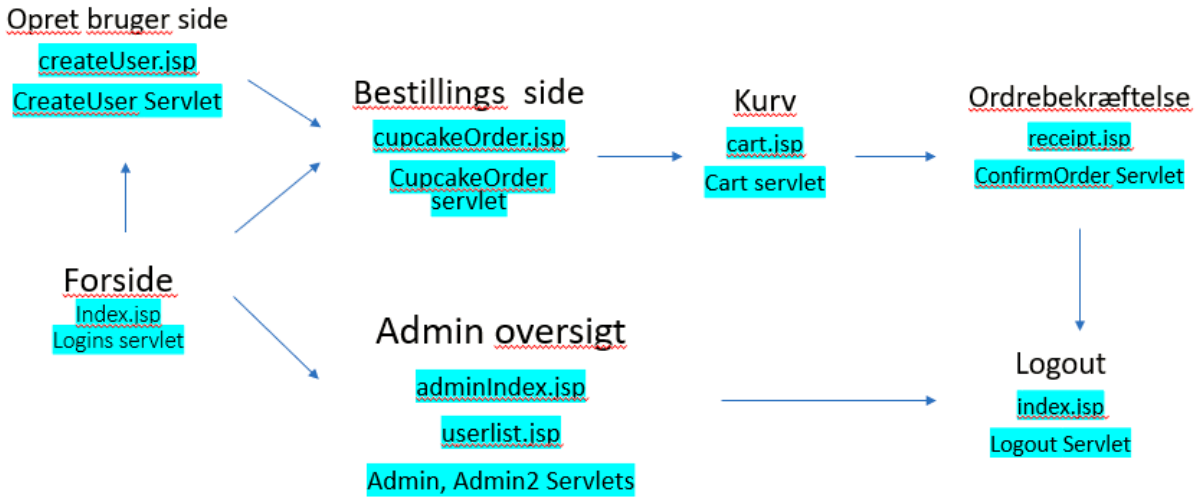
Vi prøvede at få et overblik over siderne. Det er vores første udkast, som tegnet på tavlen. Det gav os en forståelse for flowet gennem siderne.



Flowet gennem siderne efter finpudsning kan ses nedenfor.

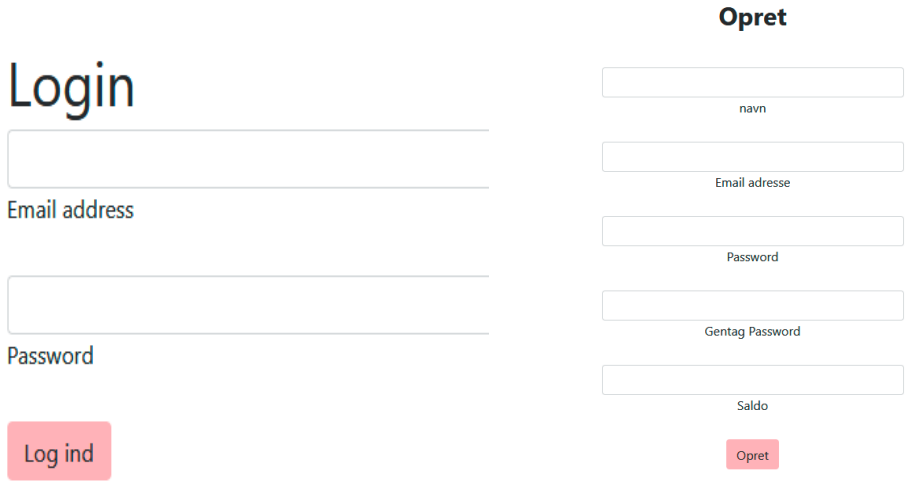


Overordnet diagram nu med servlets og jsp sider:



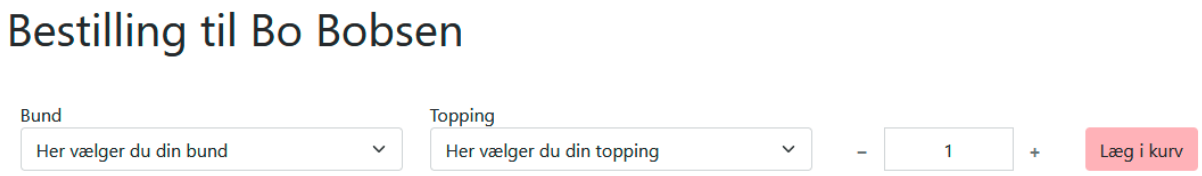
Beskrivelse (navigationsbar beskrives i næste afsnit):

Brugeren/kunden mødes med forsiden. Brugeren skal logge ind for at kunne komme til bestillings siden. Hvis brugeren ikke har et konto, skal dette oprettes først.

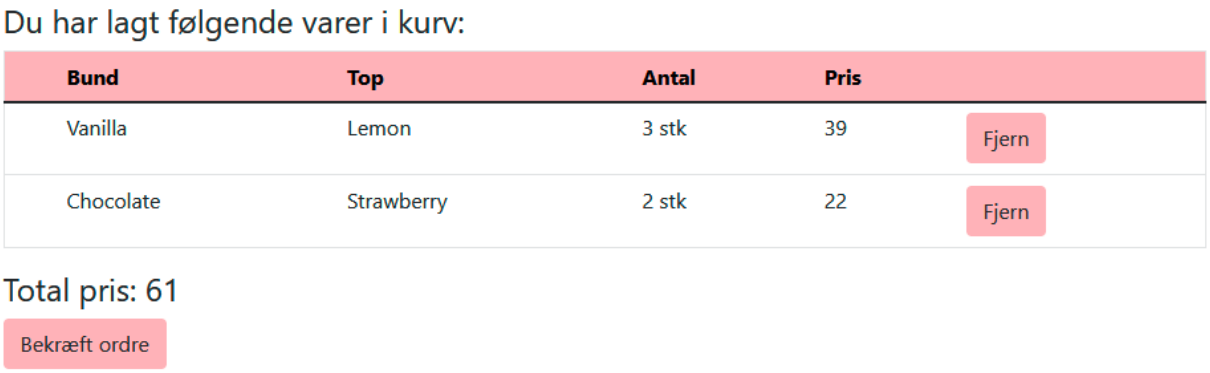


Efter kunden har logget ind, møder han/hun med en velkommen og så skal man trykke på 'bestilling' i navigationsbaren for at kunne komme til bestillings siden. Vi har diskuteret lidt at fjerne den del og sende useren videre med det samme til bestillings siden.

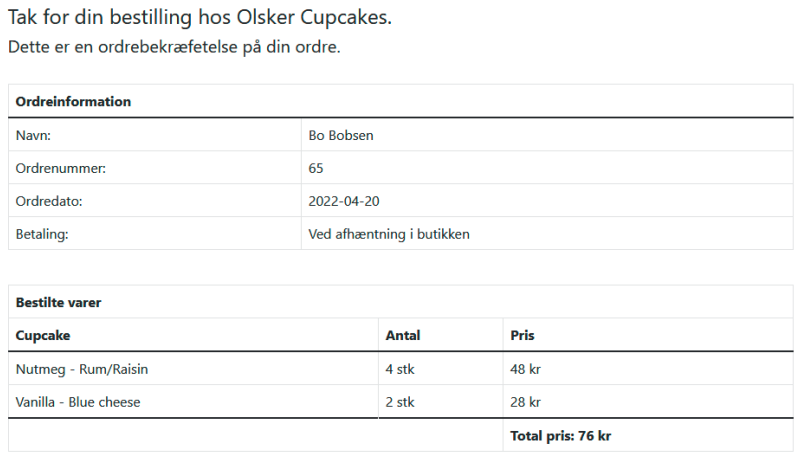
På bestillings siden møder man med drop down menuer, hvor man kan vælge bund og top til sin cupcake, mængde osv.



Efter at kunden har fyldt varer i kurv, skal man trykke på kurv tegn i navigationsbaren. Her kan man se kurvens inhold og her kan man bekræfte købet.



Når man har bekræftet sit køb, sendes man videre til ordrebekræftelses siden. Her kan man se ordreinformation og de bestilte varer.



Hvis brugeren som er logget ind er en admin, skal man trykke på 'oversigt' i navigationsbaren. Her får adminen lov til at se kundelistem, ordrene, godkende ordre eller slette dem.

Navigation bar:

Her ser vi den navigationsbar der møder brugere som ikke er logget ind.

Vi besluttet at en ikke logget ind bruger ikke skulle have nogle muligheder for andet end at logge ind, dog hvis vi havde fået tildelt en "*about us"* side med i opgaven, ville den tilgængelig for ikke ind logget bruger. Huset med hjertet er en hjem knap smider brugeren, ligemeget hvilken slags bruger, hen på den samme side, nemlig startsiden.

Her kigger vi på den almene brugers navigationsbar.
På denne version af navigationsbaren er der blevet tilføjet nogle flere elementer, først er *login* skiftet ud med *log ud*. vi har også implementeret at brugerens email står i navigations baren, denne funktion er også synlig for alle ind logget brugere, den fungere dog ikke som en knap og kan ikke integreres med. Venstre for brugerens email er der en kurv som kan integreres med og den bringer brugeren hen til deres kurv, denne kurv er også dynamisk og et antal puttes på når man kommer et objekt i kurven. Efter den kommer der endnu et ikke integrere-bart element, her viser blot brugerns saldo, som er dynamisk. Til sidst er et link til en bestillings side, hvori man som almen bruger kan bestille sine cupcakes.
Som tidligere så smider huset med hjertet en ind på index siden, som nu ville være ændret da der er en bruger logget ind.

Sidste iteration af navigations baren er for admins.
Som på den foregående navigationsbar, er *login* skiftet ud med *log ud* og brugerens email er synlig.
Det eneste nye på denne version er et link til adminens oversigt side, her kan finder man adminen alle ordre som han kan thekke og godkende eller slette.

Særlige forhold

Dette afsnit bruges til at beskrive særlige forhold der benyttes i programmet. Det kan f.eks. være:

- Hvilke informationer gemmes i session

RequestScope: bliver brugt når adminen skal godkende eller slette en brugers ordre. Bliver også brugt når en bruger skal slette en ordre fra sin kurv.

SessionScope: bliver sat i gang når man logger ind og lagt ned når man logger ud. Kunders ordre bliver også hæglet op på SessionScope'et, før det bliver stemplet ned i databasen(når man bekræfter inde i kurven) .

ContextScope: bliver brugt når man henter topping og bunde.

- Hvordan håndterer man exceptions. Det kommer vi tilbage til senere i semesteret.

Vi benytter DatabaseExecption klassen som vores startkode var født med. Vi har ikke udviklet det videre.

- Hvordan man har valgt at lave validering af brugerinput.

Vi tjekker for at username og password passer sammen, når vi henter det ud fra databasen. Udover det har vi ikke lavet andre form for validering.

- Hvordan man har valgt at lave sikkerhed i forbindelse med login.

Har haft benyttet den tomme Authentication klassen start-koden kom med, som så er blevet bygget op så den virker.

- Hvilke brugertyper, der er valgt i databasen, og hvordan de er brugt i jdbc

Der findes 2 bruger typer: user og admin. User brugeren har adgang til en anden GUI end admin brugeren. Det eneste de to har til fælles er visning af hvem der er logged in og logout knappen, begge i navigations baren. User har adgang til sider har relevance for oprettelses af en order, hvor en admin har adgang sider der har oversigt og redigering/sletning af ordre at gøre.

- … andre elementer

  Vi har ikke nået at lave en ordentlig refaktorering. Har heller ikke nået at køre test.

Vores implementeringer:

JSP-Sider:

- Vi har fået implementeret alle de jsp-sider som vi havde tiltænkt, dog med den ene forskel at vi har kombineret 2 sider. Admin havde somså ikke et behov for at have både en kundeliste side og en ordreoversigt.
- Dog har vi 2 jsp som ikke bliver benyttet i vores endelige version, det er henholdsvis den login-side vi fik med fra start-koden, her valgt vi at login funktion fungeret bedre at have på start/index siden. Man vil stadig kunne URL-hacke sig vej til denne side. Den anden side der ikke benyttes er den JSP-siden som vi har lavet i forbindelse med navigations-diagrammet, det ville ha været en kundeside, men som tidligere nævn er den kombineret med ordreoversigts siden.

CRUD:

Vores nuværende CRUD-metoder dækker de tiltænkte ønsker vi har, fx. Så bruger vi *insert* når man opretter en bruger, og vi bruger *update* når adminen bekræfter ordre (ændre bekræftes bruger saldo). Dog kunne man godt argumenter for at lave CRUD-metoder til at *update/insert* topping og bunde tabellerne, det ville blot være en admin funktion.

Styling:

Vores styling af hjemmesiden er desværre blevet mindre visuel end ønsket. Vores overall styling består i et gennemgående farvevalg, hvori vi godt kunne have haft udtænkt noget mere visuelt, nok nogle flere billeder.

Fejl / mangler:

- Fejl /site-crashes har vi ikke nogle af, dog kunne vores håndtering af nogle af dem godt ha været bedre. Fx. når man glemmer at vælge en topping eller bund til cupcaken så refresher siden blot med en fejlbesked, dette kunne vi godt ha haft at blot highlighte den dropdown menu hvori fejlen består, dette vil benytte sig af JavaScript.
- En mangle ville dog være at når man opretter en bruger på siden, så er der et felt til at gentage sit password, og strengt sagt så har denne form ikke nogle nuværende værdi. Dog er den implimenteret således at man hurtigt ville kunne tage den i brug. Det har vi dog glemt ved flere lejligheder.
- Ved samme sted så burde man nok have ændret det således at når man opretter en bruger så bestemmer man ikke selv sin egen saldo.
- Vi har også bidt mærke i at når man ankommer på siden og ikke er logget ind endnu, så er der to knapper der henviser til den samme side, det ville være en nok lidt ligegyldige *login* knap i navigations baren som henviser til den som man allerede er på, men der er også huset med hjertet, som har nøjagtigt samme funktion.
- En anden mangle vi har i forhold til vores navigations diagram, er at når man har logget ind så burde man blive sendt hen til bestillings siden med det samme, her løb vi dog ind i

det problem at når man logger ind som Admin er det ikke meningen at man skal tilgå den side. Vi fandt desværre ingen løsning, i øjeblikket, så vi har valgt at gå med at når man logger ind så refresher den blot index siden.

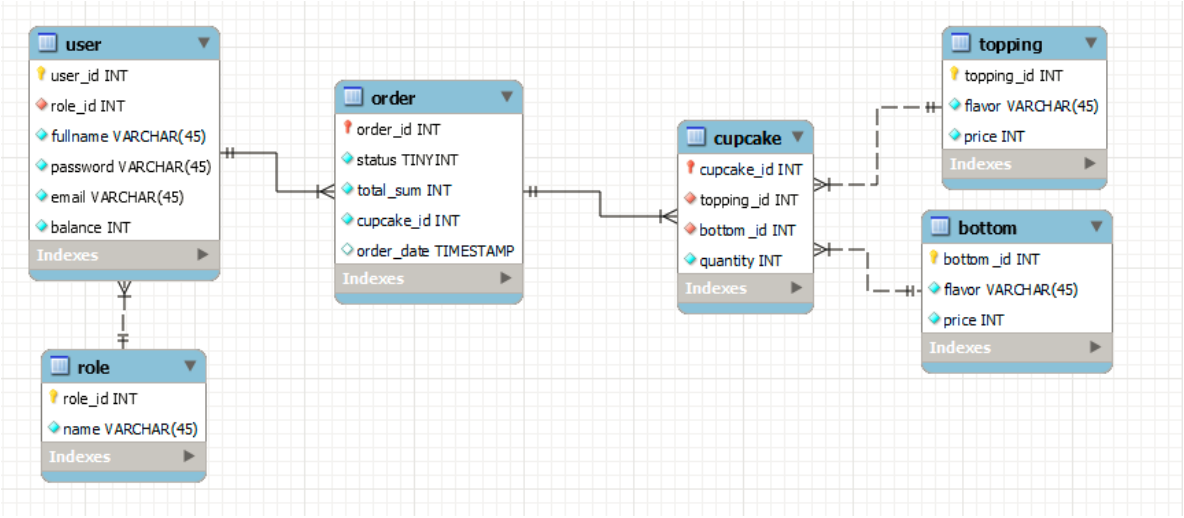
- Et sted vi har bemærket kunne være en mangel er i ordre-oversigten, her tyder det på at selve oversigten er sorteret efter order_id'et. Hvori vi tænkte at det måske ville give bedre mening at sortere det efter tidspunkt, dog er vi kommet lidt uden om det ved at der er en søge-funktion.

Proces

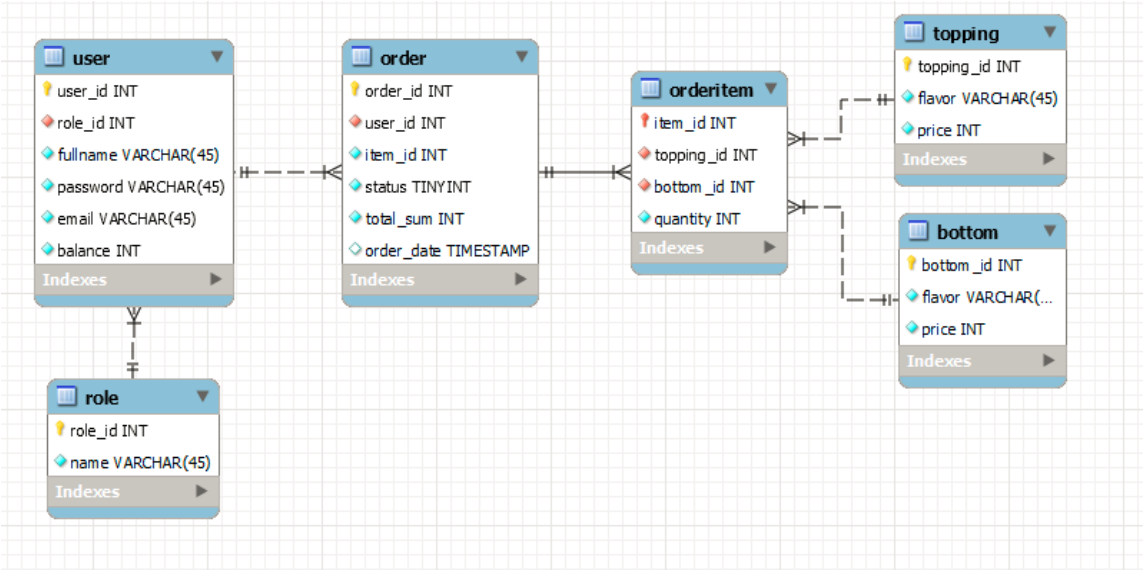
Vi arbejdede i et team. Arbejdet foregik med enten fysisk fremmøde eller online på Discord. Domæne-modellen, aktivitetsdiagram, EER modellen og navigationsdiagram udarbejdede vi i fællesskab. Koden begyndte vi at skrive efter use cases og det delte vi ud imellem os.
Vi har desværre haft noget problemer med GIT i starten, som vi brugte rigtig meget tid på at fixe.
Vi arbejdede relativt "smooth" sammen og ikke har haft store uenigheder. Vores vision var i starten at nå de første 6 use cases, men med at processen glidede stille og roligt frem kunne vi se at vi kunne nå alle use cases. Vi har lagt mere vægt på back-end delen end på front-end. Det kan godt ses på vores userinterface, men overordnet er vi tilfredse med det, og vi synes vi har nået en lidt simpel men nemt overskuelig resultat.
Vi kan se nu at det havde været godt hvis vi havde mere tid på forarbejde og design af vores kode. Det kunne spare os tid under implementation af use cases. Det havde også været rigtig godt hvis vi havde mere erfaringer med GIT og front-end.

Udkast af de forskellige EER diagrammer vi har haft :

Her viser vi de forskellige EER diagrammervi har haft igennem processen, som viser vorses overvejelser gennem det hele.
Første udkast:



Anden udkast: Efter overvejelserne og snak med Jon



Sidste udkast: også vist tidligere i rapporten