30-03-2020

Projekt Cupcake

Gruppe 45

Klasse E

Navn	Github	Mail
Cecilie Langsø Nielsen	CecilieLNielsen	cph-cn274@cphbusiness.dk
Gustav Wernegreen	gustavw10	cph-gw30@cphbusiness.dk
David Josefsen	Josefsendavid	cph-dj154@cphbusiness.dk
Mathias Noe Clausen	MathiasNoeClausen	cph-mc366@cphbusiness.dk
Abdelhamid Hariri	Abed01-lab	cph-ah482@cphbusiness.dk

Indholdsfortegnelse

Indledning

Denne rapport er udarbejdet af fem studerende fra Cphbusiness Lyngby, David, Mathias, Gustav, Abed og Cecilie. I denne rapport vil vi uddybe den pågældende virksomheds krav, teknologivalg og opstille relevante diagrammer som dokumentation for projektet.

Baggrund

Kunden er Olskers Cupcakes, som ligger på Bornholm, og deres projekt til os består af en online webshop til salg af deres varer, som er cupcakes. Olsker Cupcakes har stillet krav til, at der både skal være et internt og eksternt delsystem i denne webshop nemlig, at der både er en administrator- og kundedel.

Olsker Cupcakes vision er, at deres kunder skal kunne logge ind, eller lave en ny bruger, hvis det er første besøg. Desuden skal kunden kunne vælge mellem bunde og topping samt antal og lægge det i en kurv, hvor der både kan fjernes fra ordren, og hvor man endeligt kan bestille, hvorefter de kan komme og hente bestillingen i den fysiske butik.
Administratoren skal endvidere kunne administrere kundernes saldo, se en liste over kunderne og de foretaget ordrer.

Teknologivalg

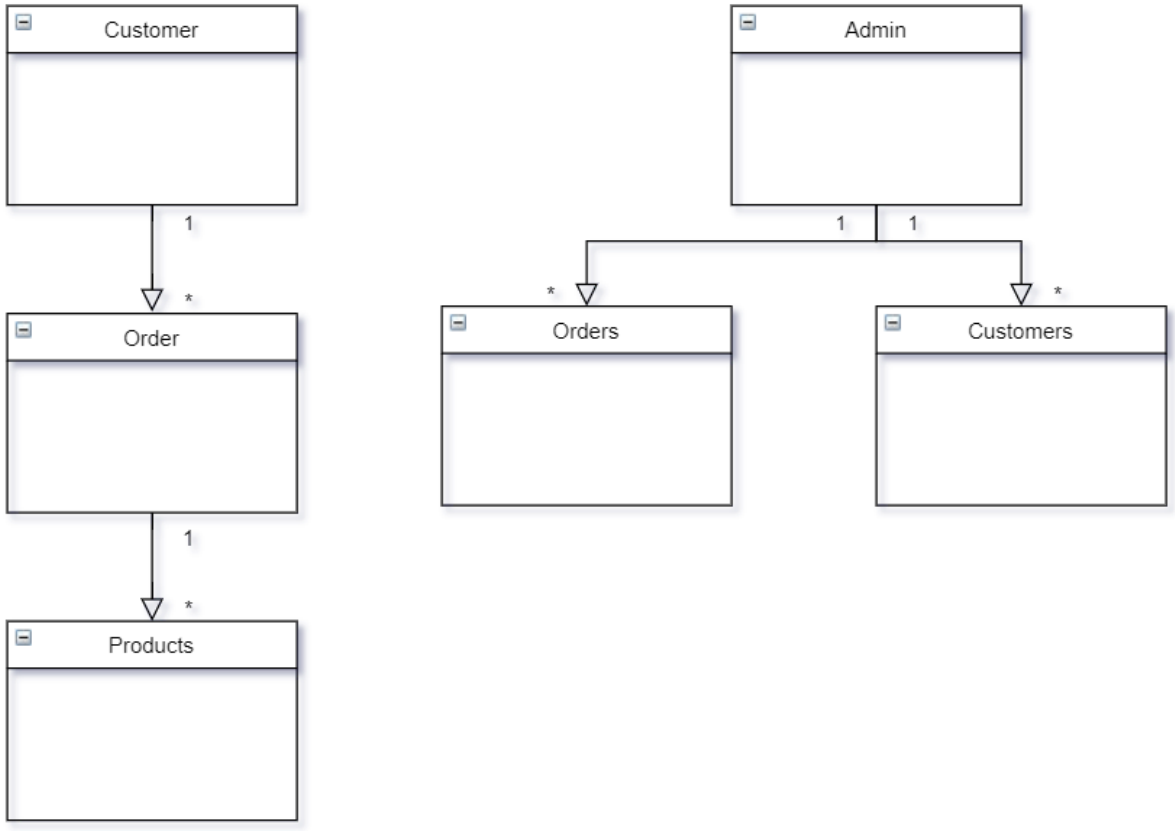
Vi har kodet projektet i IntelliJ IDEA ULTIMATE version 2019.3.
Vi har valgt dette, da det er dette program vi er blevet introduceret for på dette semester i skolen i den forberedende undervisning til dette projekt.

Vi har Mysql Workbench version 8.0.19 til oprettelse af database.
Vi har valgt dette fordi, at det er det program, som vi også tidligere har brugt til databaser under uddannelsen.

Vi har brugt Apache Tomcat version 9.0.31 i forbindelse med testkørsler af webapplikationen under kodningen af projektet. Dette har vi valgt på baggrund af anbefalinger fra undervisere samt den simple konfiguration af Apache Tomcat gennem IntelliJ.

Vi har lavet en droplet på Digitalocean til at få vores program op på vores virtuelle maskine, så man kan tilgå hjemmesiden udefra.

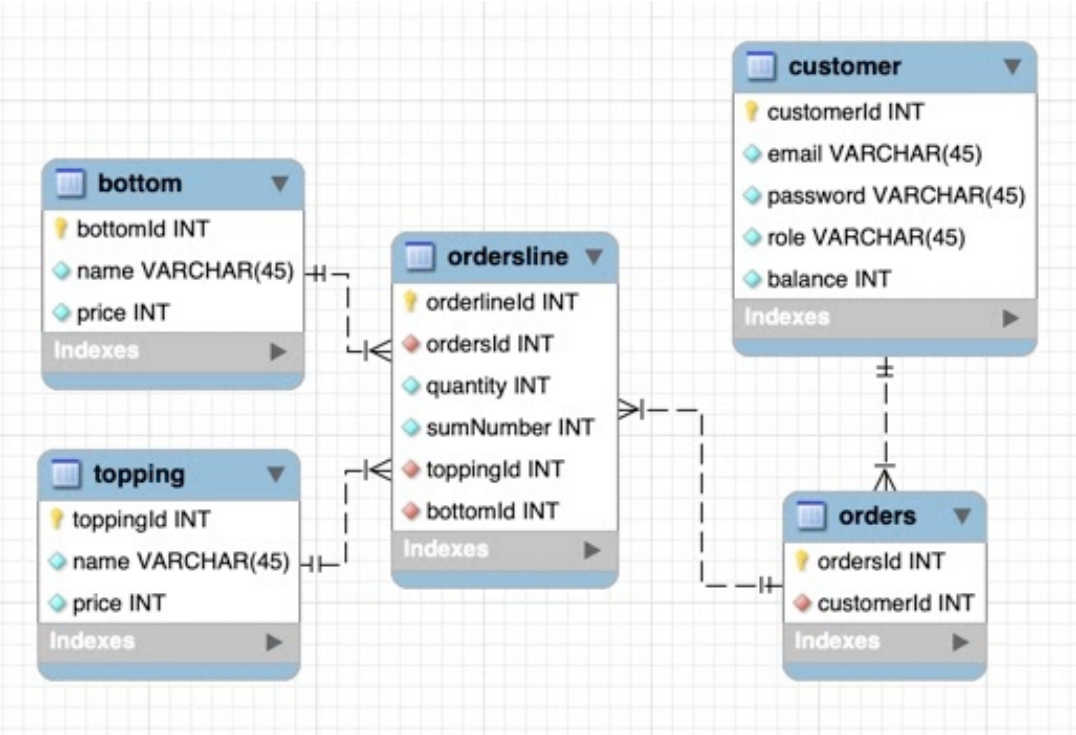
Domænemodel



Domænemodellen repræsenterer en oversigt over det domæne, som man som udvikler har fået til opgave at lave. I dette tilfælde har vi udarbejdet domænemodellen, efter vi er blevet færdige med at kode, hvilket giver en mere præcis repræsentation af, hvad Olskers Cupcakes domæne indeholder. Modellen bruges i forhold til dialog med kunden, med henblik på at få et samlet overblik over domænet, samt at udviklerens såvel som kundens forventninger til produktet stemmer overens.

Domænemodellen for Olsker Cupcakes, viser at man starter på en log ind side, hvorefter man kan logge ind som enten kunde eller administrator. Som administrator kan du se alle de forskellige kunder og ordrer i systemet, samt indsætte et bestemt beløb på en given kundes bruger. Når man er logget ind som kunde, kan man se alle de forskellige produkter Olsker Cupcakes har at tilbyde. Ud fra produkterne kan man rimelig intuitivt lave flere forskellige variationer af ordrer, som man lægger i kurven enkeltvis. Hvorefter man til sidst betaler for sin ordre og bliver sendt videre til domænets bestillingsbekræftelse side.

ER diagram



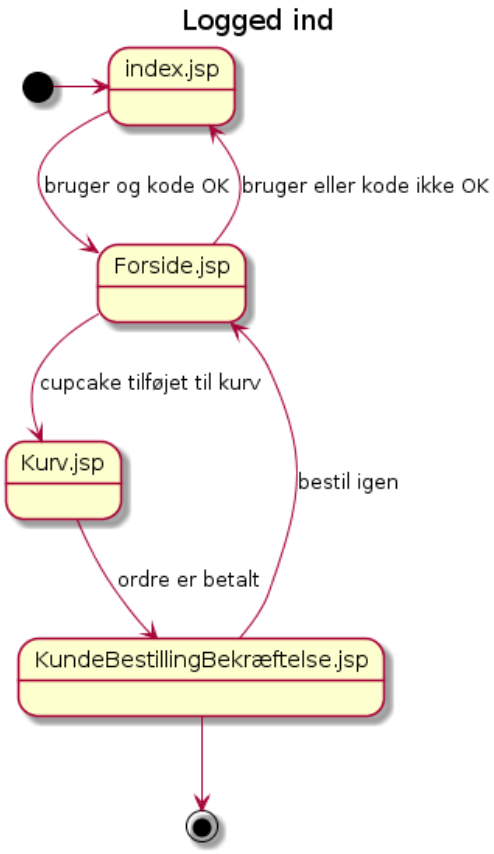
Vores database blev opbygget ud fra den foreslåede skabelon. Ved at følge normalform reglerne elimineres redundans i databaserne, og vi opnår en mere effektiv og let tilgængelig struktur. Hver af vores tabeller har en primærnøgle der alene identificerer sin tabel og bruges i data relationer med andre entiteter; alle kolonner i en gældende tabel indeholder kun data om sin egen, enkelte entitet; og ingen af kolonnerne er direkte afhængige af andre attributter end primærnøglen. Den klare opdeling af vores data i relevante og separate entiteter betød at det var nemt at hente informationer fra tabellerne igennem vores mappere samt at få dem til at samarbejde via fremmednøgler, når det var nødvendigt at samle information fra forskellige tabeller. Vi undgik f.eks. en masse tunge WHERE statements, når vi skulle omdanne toppings tabellen til et ArrayList af toppings objekter ved at fra starten af at have separeret de to typer, toppings og bottoms, i sine egne tabeller.

Navigationsdiagram / Tilstandsdiagram

Navigationsdiagram er et hjælpediagram, der skaber overblik i hvordan applikationen navigeres. Diagrammet viser "States" som er jsp-siderne. Teksten og pilen er den aktivitet, der sker, når man brugeren bevæger sig gennem applikationen.

Vores navigationsdiagram viser den del hvor brugeren logger på til at bestille en ordre af cupcakes.

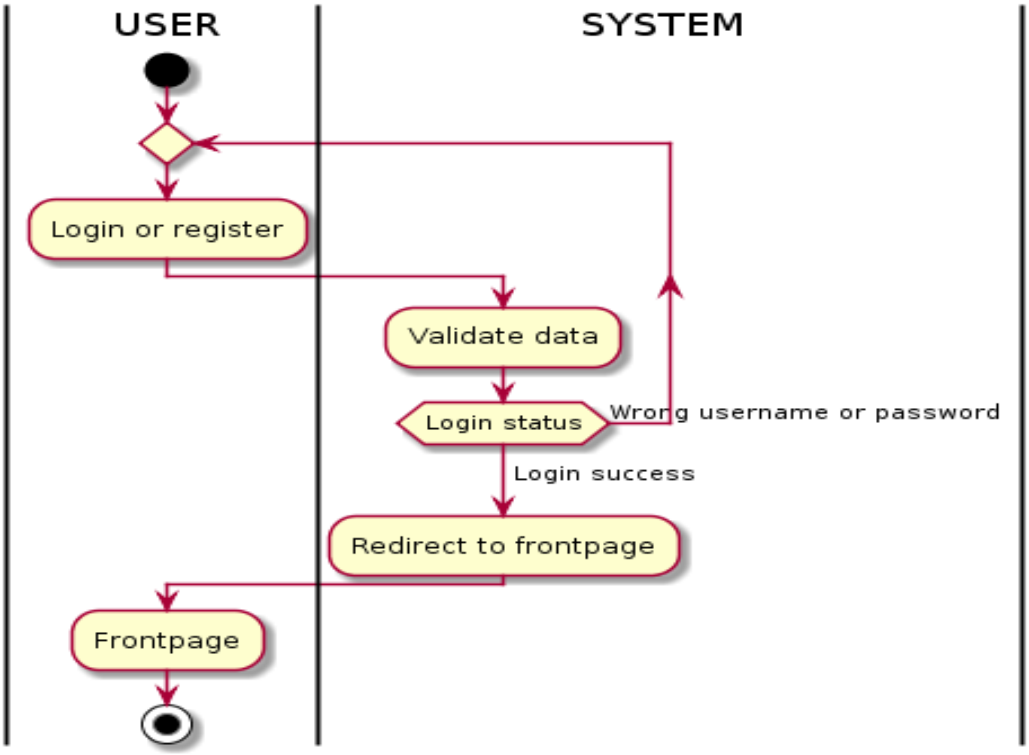
1. Brugeren lander på vores index side, som også er loginsiden.
2. Hvis de tastede oplysninger er korrekte, vil brugeren sendes videre til forsiden.
3. Hvis oplysningerne er forkerte, vil brugeren tilbagesendes til index siden med en "forkert kode" besked.
4. Ved at tilføje en cupcake til kurven lander brugeren på kurv siden.
5. Når ordren er betalt, bliver brugeren sendt til KundeBestillingBekræftelse siden.

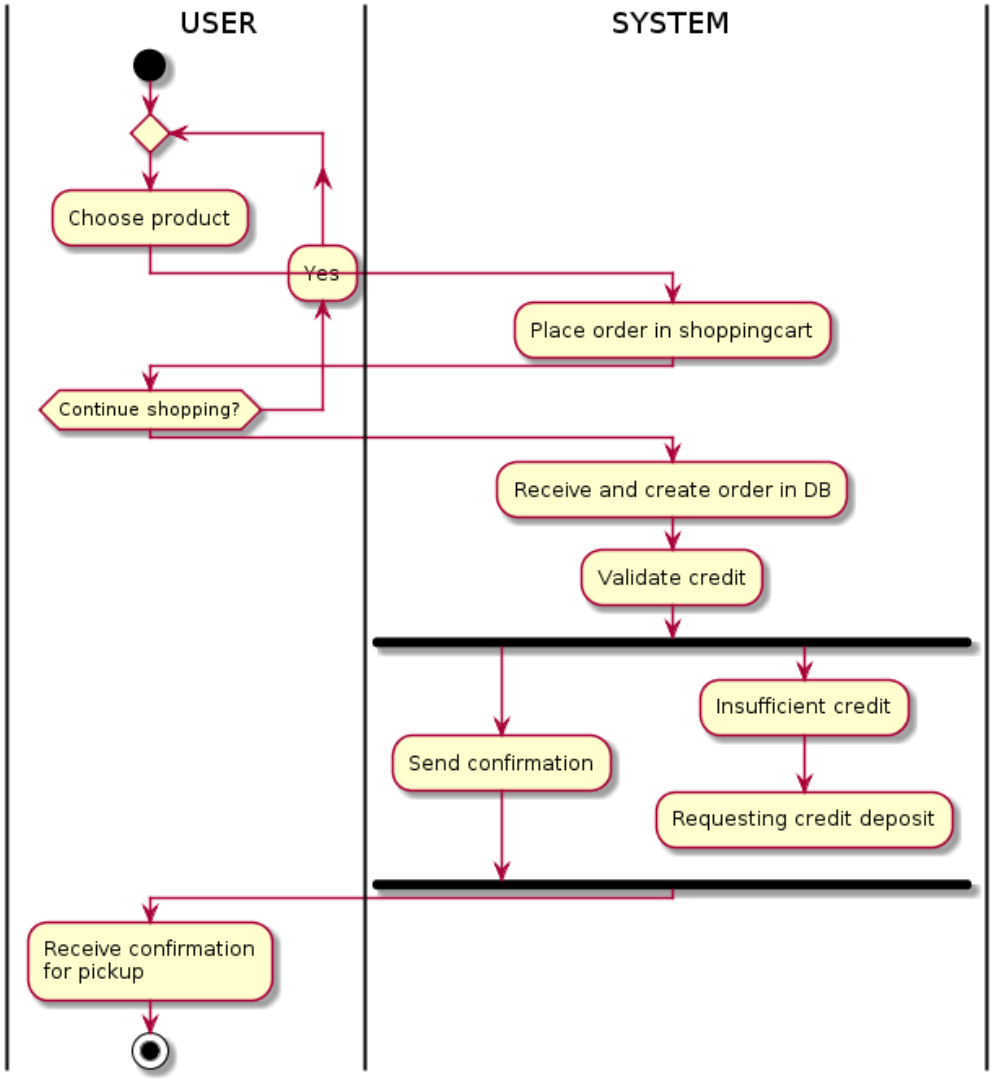


Fordelene ved dette diagram er, at det skaber et mere klart billede af, hvad der skal programmeres og hvilke funktioner der skal implementeres i applikationen. Diagrammet kan også tegnes med kunden, da det skaber en mere klarhed omkring kundens formål med applikationen og hjælpe programmøren til at vide, hvad der skal kodes.

Aktivitetsdiagram

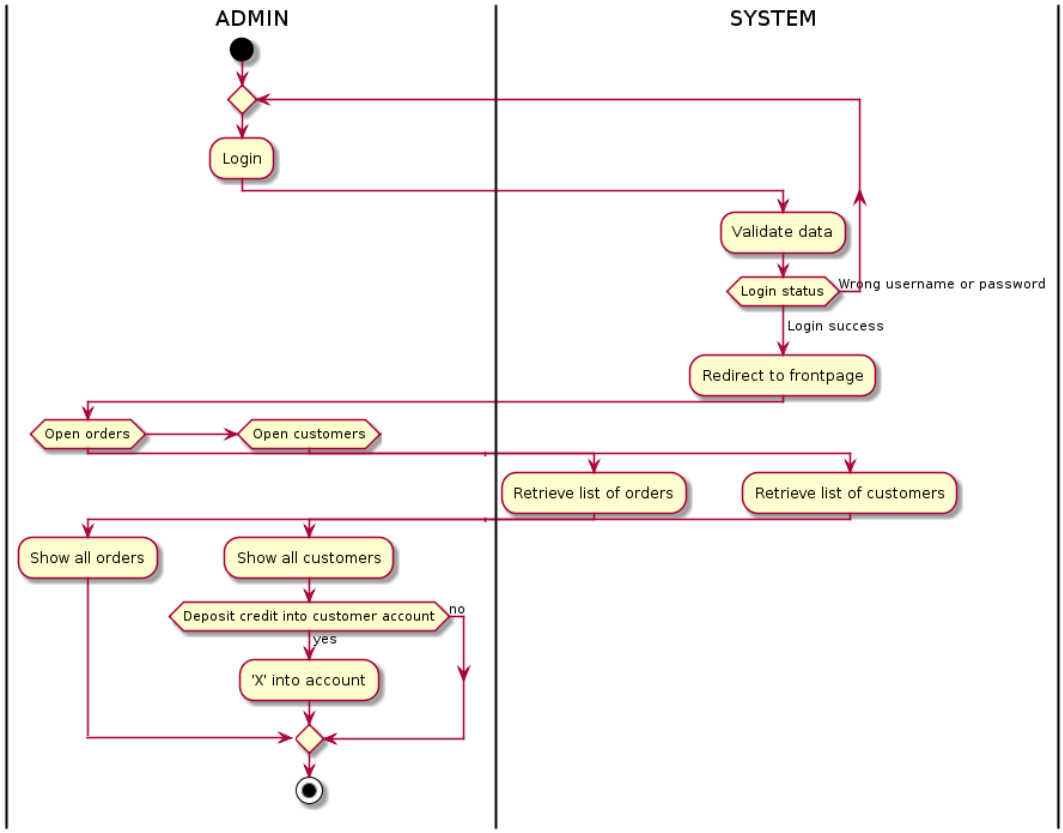
Følgende aktivitetsdiagrammer er bygget med swim-lanes vha. plantUML.





Formålet med et aktivitetsdiagram kan være at illustrere aktivitetsflowet af et system eller beskrive de forskellige sekvenser, der forekommer mellem 2 aktiviteter som i vores eksempel er, User og System.

I det ovenstående diagram tager vi udgangspunkt i rollen user, hvor useren bliver bedt om et brugernavn og kodeord, hvorefter dataen bliver sammenlignet med, hvad der står i databasen og valideres af systemet. Opretter brugeren sig som ny kunde, vil systemet udskrive en besked om, at brugeren er oprettet, og kan derefter logge sig ind. Når dataen er valideret bliver brugeren henvist til forsiden, hvor der kan vælges et produkt. Hvis dataen ikke stemmer overens med oplysningerne i databasen, vil systemet smide en fejlmeddelelse. Den samme funktion forekommer, når brugeren køber sin ordre, hvor ens kredit vil blive valideret og til sidst give en order confirmation eller smide smide en fejlmeddelelse.



I dette aktivitetsdiagram tager vi udgangspunkt i rollen admin. Når admin er logget ind viderestilles der til forsiden, som viser en liste over kunderne og deres saldo. Her kan administratoren vælge at se alle ordrer eller kunder. Vælger man at se kunder, vil systemet hente dataen ned, og vise detaljer for hver kunde, samt kunne tilføje kredit til en kunde, hvor systemet går ind og behandler anmodningen. Hvis administrator åbner alle ordre, vil systemet vise antal af cupcakes, hvilken type bund, topping og pris for hver ordre id, samt det tilknyttede kunde id.
Et aktivitetsdiagram kan bruges til flere forskellige behov:

1. Lave en model af workflow i aktiviteter.
2. Lave en model af forretningskrav til systemet.
3. Få en højere forståelse af systemets funktioner.
4. Undersøge forretningens krav på et senere tidspunkt.

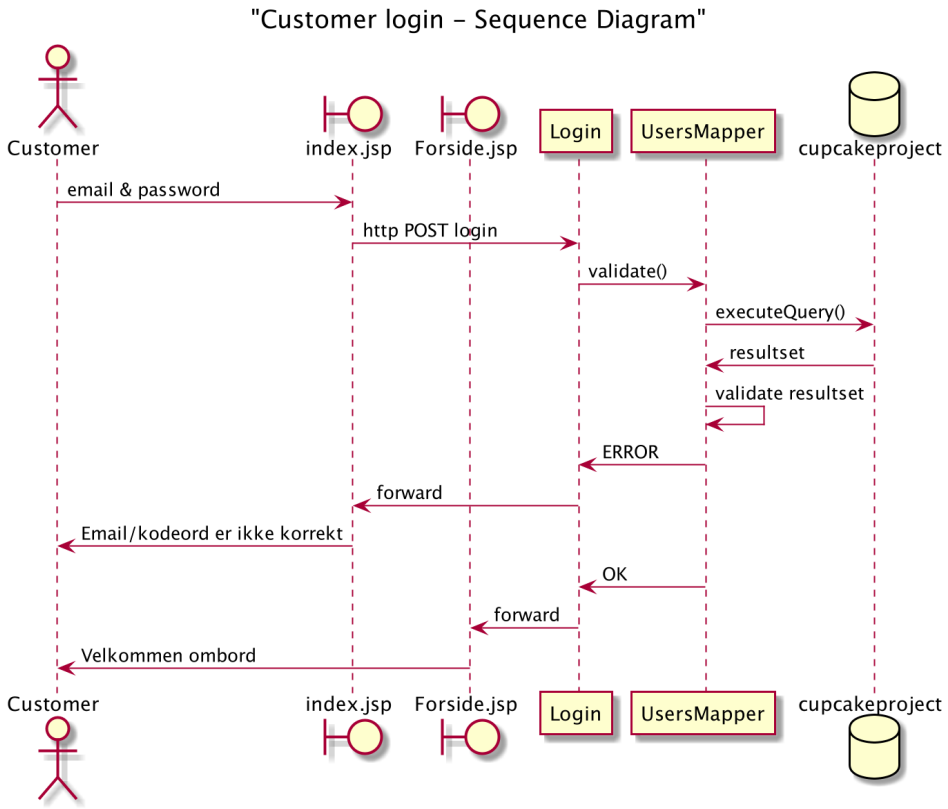
Sekvensdiagram

Sekvensdiagrammer giver et overblik over hvilke forespørgsler samt hvilken respons, der bliver sendt frem og tilbage i systemet i forbindelse med en brugers aktivitet, som på dette diagram er indlogning.

På diagrammet ses scenariet, hvor en kunde logger ind på index siden med sin email og kodeord. Hvorfra der sendes en POST forespørgsel til systemet, hvor kundens email og kodeord bliver valideret, hvis de indtastede oplysninger ikke stemmer overens med dataen i databasen, vil brugeren bliver viderestillet tilbage til index siden, hvor kunden vil få en fejlmeddelelse.
Hvis oplysningerne findes i databasen vil kunden derimod blive viderestillet til forsiden, hvor det er muligt at bestille cupcakes.

Diagrammet gælder principielt både for kundens kunder og administratoren den eneste forskel ved processen er, at forsiden de bliver viderestillet til vil være forskellig fra hinanden afhængig af deres rolle, da de skal have adgang til hver sine funktionaliteter.

Diagrammet er kodet med PlantUML plugin i IntelliJ.



Status på implementation

Tidsperioden har omspændt cirka to uger. I starten af projektet havde vi estimeret at aflevere et færdigt projekt til deadlinen, med dette menes, at alle user stories ville være implementeret, hvilket vi har formået at realisere inden for tidsrammen.

Vi har dog nogle fejl og mangler på projektet, som vi ikke har fået rettet op på grund af tidspres:

1. Hvis kurven har en ordre i en session, og siden genindlæses, tilføjes der en ekstra til cupcake.
2. Selvom kurven er tom, er det stadig muligt at trykke på køb knappen og få en bestillingsbekræftelse.
3. Databasen overholder ikke tredje normalform i feltet sum, idet attributten sum afhænger af andre attributter i tabellen, der ikke er nøgler. Det gjorde det nemmere at hente ned men fra et databasedesigns synspunkt, er det en fejl.