2020

Olsker Cupcakes

Gruppe: JUMBOSNEGL

Navn: Emil Alexander Westergård

Email: cph-ew45@cphbusiness.dk

Github: https://github.com/Tenz331

Gruppe: JUMBOSNEGL

Navn: Janus Stivang Rasmussen

Email: cph-jr270@cphbusiness.dk

Github: https://github.com/Janussr

Gruppe: JUMBOSNEGL

Navn: Mathias Toubøl

Email: cph-mt326@cphbusiness.dk

Github: https://github.com/MGDelux/cupcake

Gruppe: JUMBOSNEGL
Udarbejdet af: Janus Rasmussen, Mathias Toubøl og Emil Westergård
DAT2SEM

Indledning

Olsker Cupcakes er en nyopstartet iværksættervirksomhed med base på Bornholm. Olsker Cupcakes sætter økologi i førersædet og har dermed fundet frem til den helt rigtige opskrift. Olsker Cupcakes har ambitioner om at få etableret en hjemmeside, hvor de kan drive online-shopping ved salg af cupcakes, hvorefter deres kunder har mulighed for at komme forbi og hente deres bestilte cupcakes. Selve websitet skal indeholde en log ind funktion, hvor kunden har mulighed for at oprette en bruger samt logge ind. Når den pågældende bruger er logget ind, har vedkommende mulighed for at kunne bestille og betale for de valgfrie cupcakes.
Herudover har kunden mulighed for at se sin bestilling i kurven samt slette fra kurven igen. Når man er logget ind som kunde skal man kunne se sin E-mail i toppen på navigationsbaren, dette gælder også dem med admin rettigheder. Er du logget ind som admin skal du have mulighed for at se alle ordre og registrerede kunder og herudover har de også mulighed for at se bestemte kunders ordre samt kunne slette ugyldige ordre i systemet.

Teknologivalg

Ved udarbejdelsen af Olsker Cupcakes hjemmeside har vi brugt IntelliJ ultimate edition IDEA 2020.2. IntelliJ er brugt til alle vores klasser og metoder til programmet, som er skrevet i programmeringssproget Java. I vores JSP(Java Server Page) filer har vi benyttet os af HTML 5, ud fra Java Development kit version 14. I IntelliJ har vi tilkoblet en database via JDBC som køre i MySQL Workbench 8.0. I Workbench har vi oprettet en skræddersyet database til opbevaring af alle olsker cupcakes informationer om kunden, toppings, bunde samt deres bestillinger.

Hjemmesiden kører fra en Apache Tomcat 9.0.38 server, som sørger for at hjemmesiden kan tilgås via en webbrowser på alle tidspunkter af døgnet, da den ligger på en droplet fra Digital Ocean.

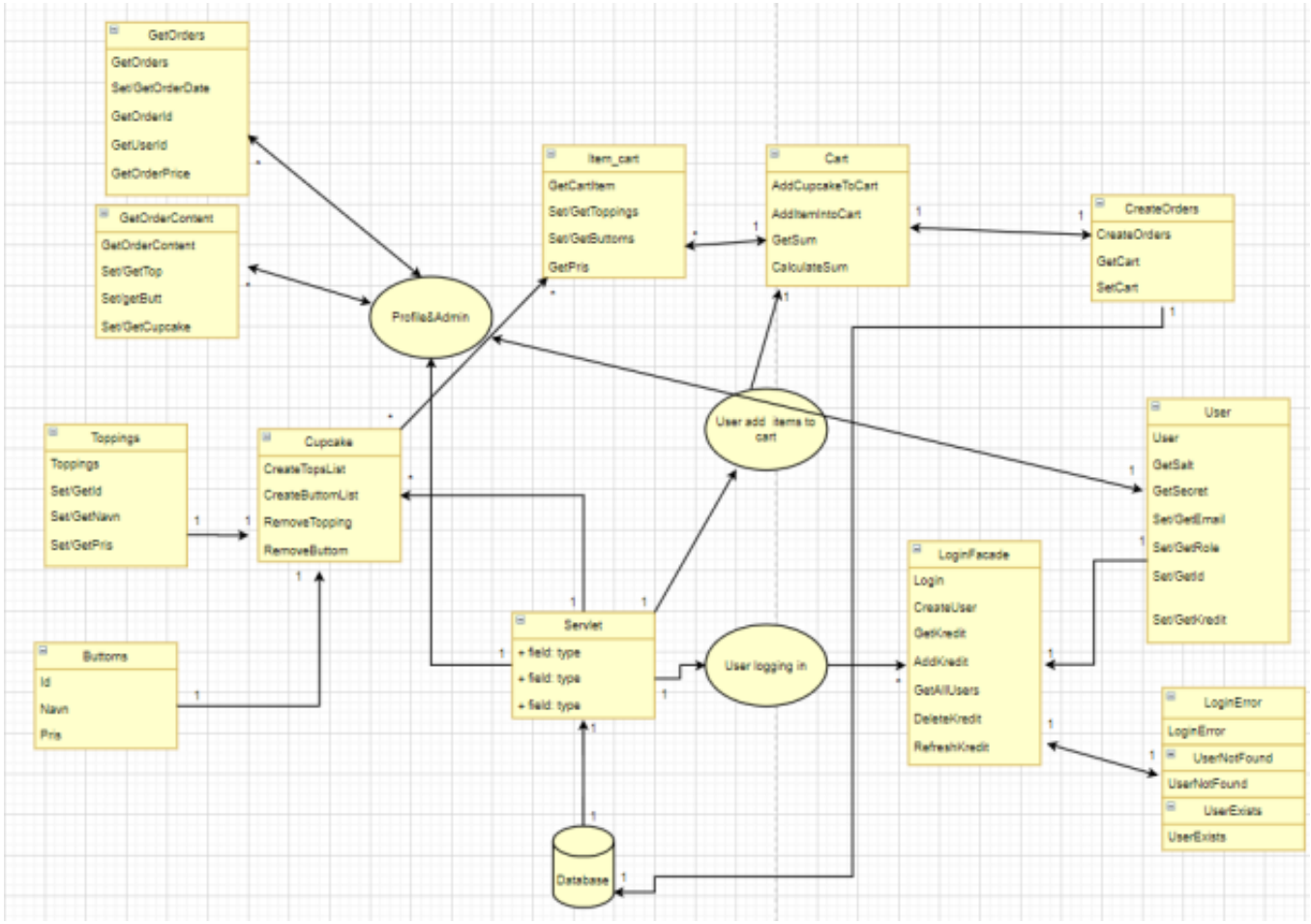
Krav

Olsker cupcakes har valgt at få integreret deres hjemmeside for at opnå en nemmere måde at nå ud til deres kunder på i form af bestillinger fra deres kunder. Via deres hjemmesiden vil Olsker cupcakes prøve at opnå en bredere målgruppe rent geografisk, da deres kunder nu har mulighed for at bestille og betale med det samme online, og har dermed mulighed for afhentning af deres cupcakes i butikken. Olsker cupcakes nye hjemmeside er oprettet med en database som skal sørge

for at holde styr på hvem deres kunder er samt alle deres bestillinger. Herudover er det også et forsøg på at forøge deres nuværende branding, for at nå ud til deres målgruppe.

Udover de krav der er til det som hjemmesiden skal kunne, har Olsker cupcakes også sat krav til selve designet af hjemmesiden. Der er givet et mockup for hvordan de har tiltænkt den skulle se ud med mulighed for modifikationer. Vi har derfor valgt at ændre designet en smule men valgt at beholde deres gennemgående tema med farverne lilla samt det at det skal være en simpel og nem hjemmeside at navigere rundt på.

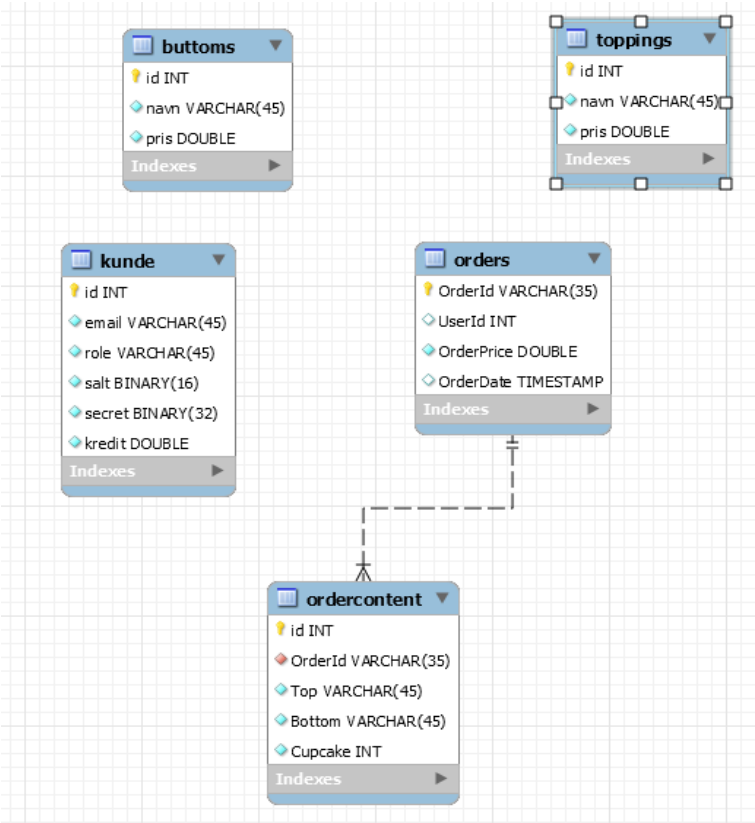
Domænemodel



I vores domænemodel har vi 14 klasser samt en database og en servlet. Eftersom at vores program er lidt dynamisk kan du derfor tilgå forskellige sider på forskellige måder så vi har indikeret en aktion ved hjælp af de gule bobler, som ses foroven. Vi har en servlet som henter vores toppe og bunde (de ligger på selve databasen) og fylder vores dropdown-menuer i cupcakes. Du har også mulighed for at logge ind, som henter user for at tjekke om brugernavn og password er korrekt.

Når du så har fyldt nogle cupcakes i din kurv, går du fra bestillingssiden til din kurv, hvor den henter din kurv og gemmer den på servletten. Herefter trykker du køb og din ordre er gemt i databasen og dermed muligt at se nuværende samt tidligere bestillinger på din profilside.

ER diagram



Som det ses af vores ER tabel foroven, har hver tabel en primary key som bliver indikeret med en gul nøgle (f.eks. har kunde ”id” som er en INT).

For at få vores tabel til at hænge sammen har vi benyttet os af foreign keys, som kun benyttes imellem ”orders” og ”ordercontent”, dette ses ved den røde prik.

Fordelen ved at bruge foreign keys i dette tilfælde er at man kan hente vores orders ned i vores ordercontent.

I vores program bruger vi vores ”toppings” og ”buttoms” til at fylde vores dropdown menuer op med toppe og bunde som vores kunder kvit og frit kan vælge imellem. Vores orders viser kundens ordreoplysninger som vi fylder ind i vores ordercontent til at udprinte kundens køb samt tidligere

køb. Vores tabel Kunder indeholder kundens informationer. På vores website har vi en adminpage, hvor det er muligt at tilføje/fjerne kunder, toppings, bunde og orders via id´er.

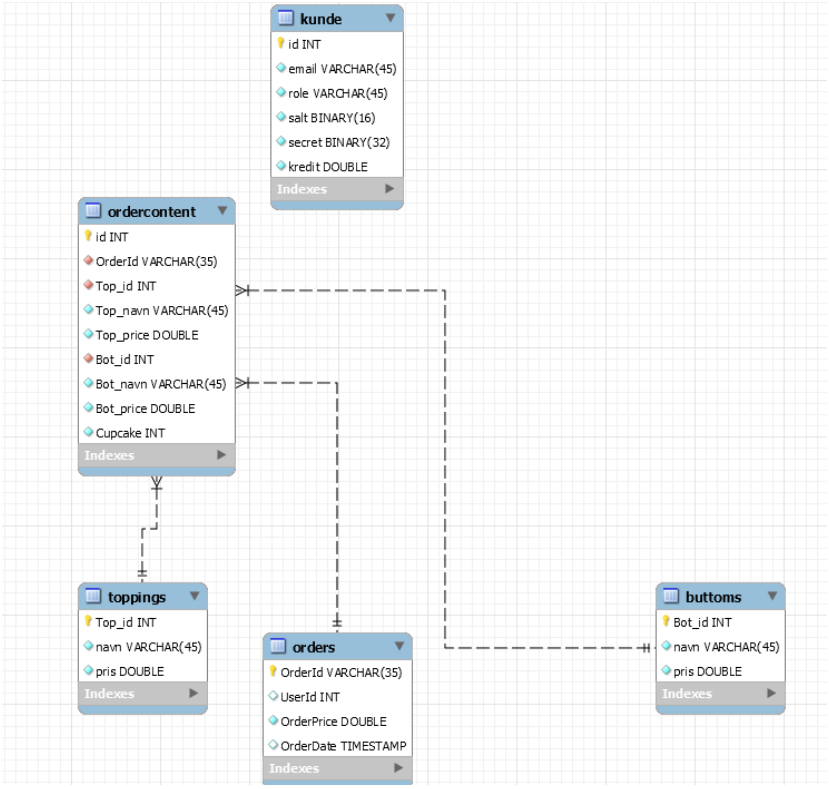
Det er en meget simplificeret database, som ikke er så dynamisk men det er en database der får jobbet gjort. Selve databasen som er connected til programmet er opsat efter CRUD.

Create: Her er det muligt at oprette ny kunde.

Read: Ordren bliver skrevet ud til kunden.

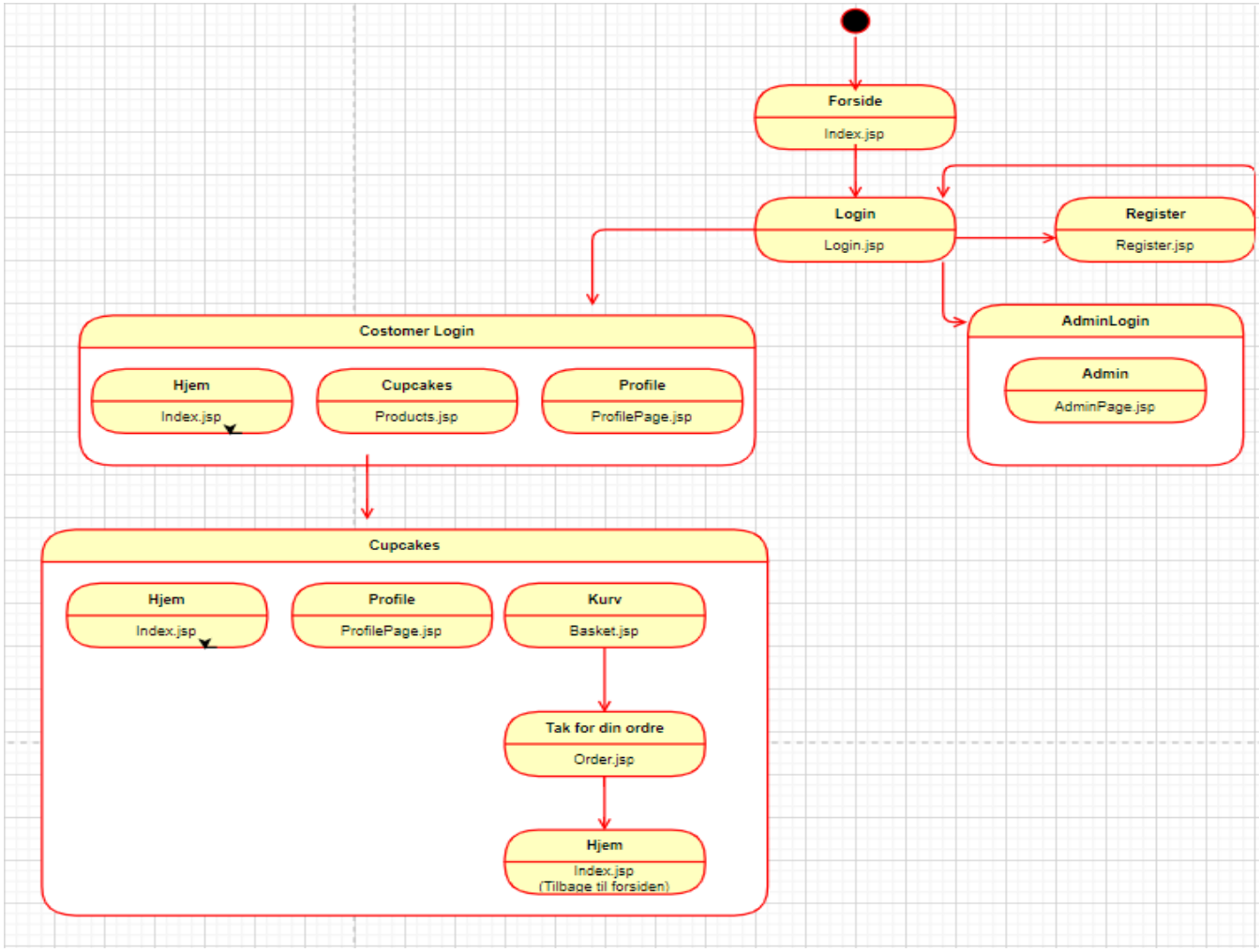
Update: Det er muligt at opdatere kundens kredit

Delete: Det er muligt at slette ordre, kunder, toppings, og bunde.



Det mest optimale ville være hvis flere af vores tabeller ”snakkede sammen” via foreign keys, som illustreret på billedet ovenover. Ovenstående tabel er en tabel vi efterfølgende har prøvet at arbejde os henimod og den virker tildeles, men vi kan ikke få printet ordredetaljerne ud da disse ikke kan hentes korrekt i vores metoder og vi har derfor måtte droppe eller sidesætte opgraderingen af databasen grundet tidspres.

Navigationsdiagram



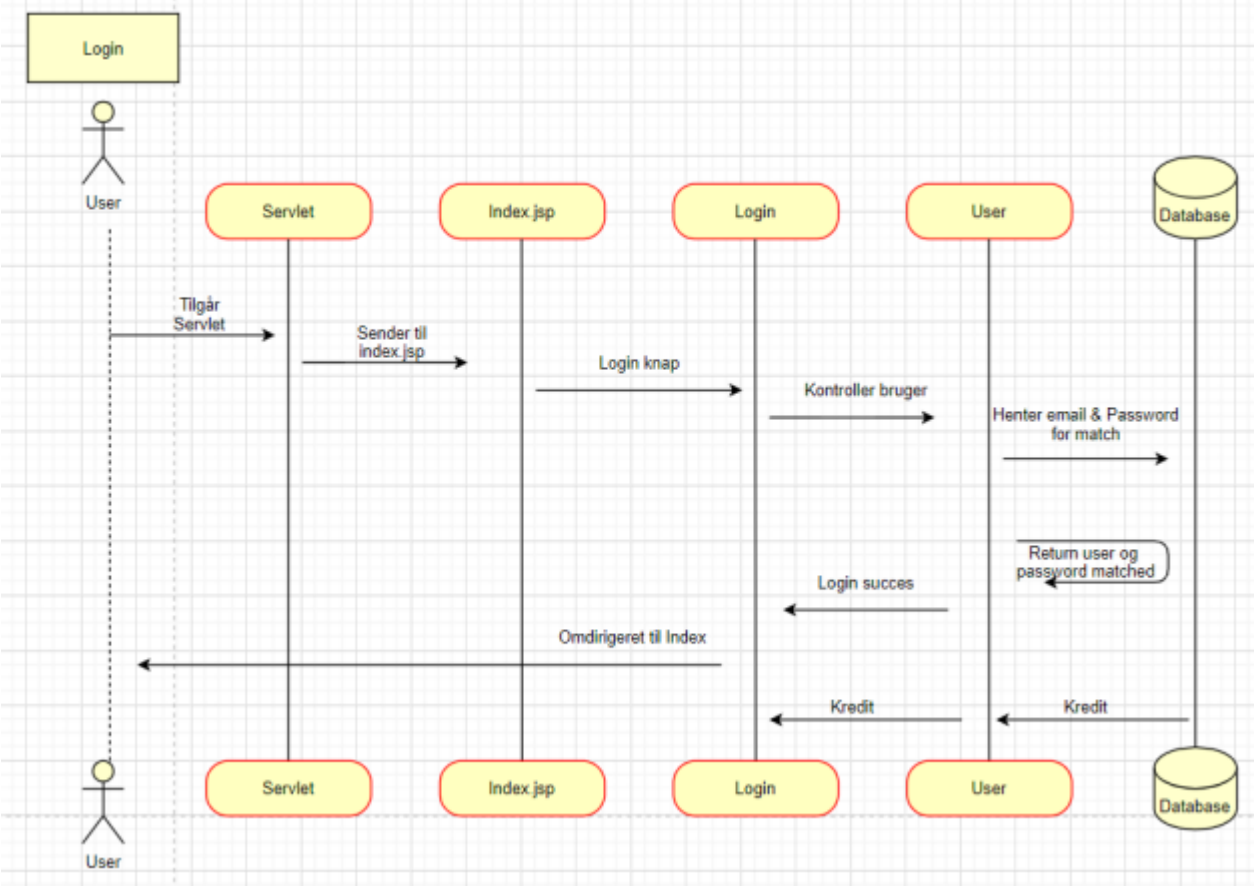
Bilag 1: skitsen illustrere en bruger som er logget ind som inden kunde eller admin.

Det første der sker når man tilgår hjemmesiden er, vores index-side/forsiden, hvor man teknisk set kan tilgå alle options som customer, men har derimod ikke mulighed for at bestille noget. Hvis du som en ikke logget ind bruger klikker på kurven vil du blive sendt til vores log ind side. Her kan man logge ind som enten kunde eller admin (dette defineres ud fra den rolle som den angivende e-mail har i databasen), hvorefter du bliver omdirigeret til vores forside igen. Er du logget ind som en customer, er der en navigationsbar i toppen, hvor du kan navigere dig videre ind på enten hjem(index/forsiden), cupcakes(bestillingssiden) eller profil (Her kan du se dine ordre). Klikker du dig så videre ind på cupcakes, har du her mulighed for at vælge en bund og en top til din cupcake og kan derefter klikke på tilføj til kurv. Når du så har tilføjet de cupcakes du vil købe, kan du klikke på kurv i navigationsbaren, for at komme videre ind til kurven og se dine valgte cupcakes. I bunden under dine cupcakes er der en knap ”køb” hvor du så køber dine valgte vare og bliver omdirigeret

til vores order-page som er en oversigt over det der er bestilt og du har derfor mulighed for at klikke ”tilbage til forside” som dirigere dig tilbage til forsiden.

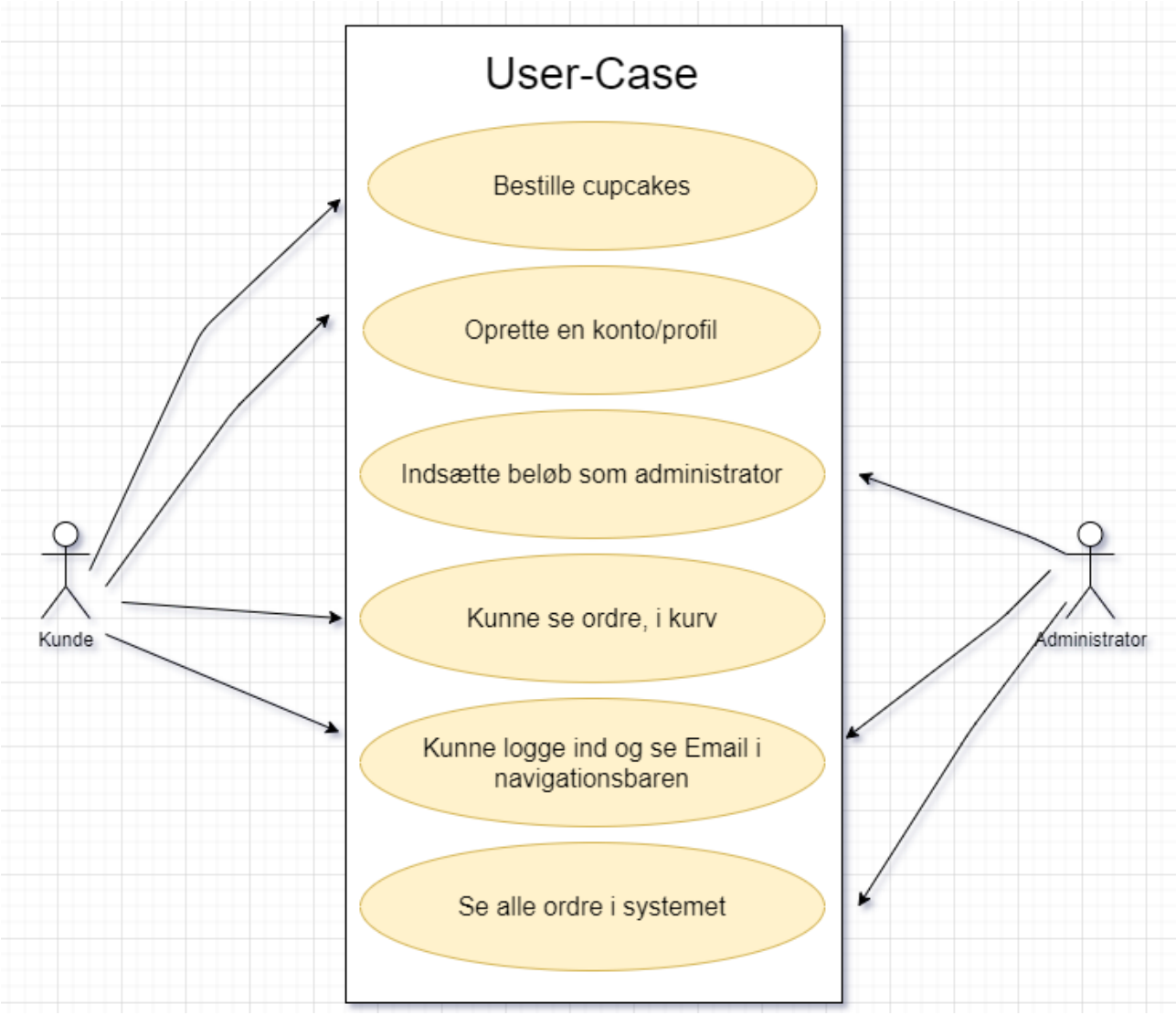
Er man logget ind som administrator er sekvensen præcis det samme som ovenstående men der er dog en enkelt tilføjelse i navigationsbaren som er adminpage. I adminpage er det muligt at indsætte penge på eksisterende kunder, oprette kunder, slette kunder, tilføje/fjern ordre, tilføje/fjern toppe og tilføje/fjern bunde.

Aktivitetsdiagram



I vores sekvensdiagram, har vi lavet en model af vores login funktion, som fungerer således. Vores user tilgår hjemmesiden, hvor han forbinder til servletten som så sender ham videre til vores index.jsp. her er der mulighed for at trykke på en log ind knap, som dirigere videre til vores login.jsp. her taster brugeren sine log ind oplysninger som bliver sendt videre til databasen for et match. Matcher de indtastede oplysninger med en bruger, bliver brugeren hentet fra databasen med det antal kreditter brugeren har og login er en succes hvorefter brugeren bliver videredirigeret til index.jsp.

Use-Case



I vores use-case diagram har vi taget udgangspunkt i de første 6 user-cases, som lyder således.

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 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.

I vores use-case har vi 2 aktører, en kunde og en administrator. Vores kunde skal kunne interagere med user-case 1,2,4 og 5, da kunden skal have mulighed for at kunne bestille cupcakes, oprette en konto/profil, se sin ordre i kurven samt kunne logge ind efter at have oprettet en konto. Vores administrator derimod skal kunne indsætte et beløb/kredit på kundens bruger, så de kan betale for den ønskede varer. Administratoren skal også have adgang til log ind, som administrator for at kunne få en ekstra funktion "adminpage". Via adminpage har vores administrator også mulighed for at se alle ordre og kunder i systemet.

Særlige forhold

Bruger input validering: Vores program er lavet sådan at, man skal logge ind, for at kunne købe cupcakes. Det vil sige at man skal oprette sig i systemet og hvis man er oprettet, indtaster kunden deres log ind-oplysninger, som bliver valideret med de oplysninger der står i databasen. Hvis oplysningerne er forkerte, har kunden ikke lov til at købe cupcakes og vil få en error message.

Herudover er vores password krypteret så selv med adgang til databasen kan man ikke se folks passwords.

Brugertyper: Vi har 2 forskellige typer i vores system den ene er en kunde(customer) og den anden er en admin-user. Som udgangspunkt kan de tilgå det samme men som admin har man adgang til adminpage. Om man er admin eller customer defineres ud fra roller i kunder på databasen.

Session: en del af vores informationer bliver gemt i en session, eksempelvis bruger vi vores sessions til at gemme kunders ordre i kurv eller i vores log ind facade. Ved at gemme i sessions har vi mulighed for at sætte, hver enkelt kunde's attributs ved eksempel req.getSession().setAttributes().

```
user = loginFacade.login(email, password);
req.getSession().setAttribute( s: "user",user);
req.getSession().setAttribute( s: "role",user.getRole());
req.getSession().setAttribute( s: "loggedIn", o: true);
resp.sendRedirect( s: req.getContextPath()+"");
```

eksempel fra loginFacade

Status på implementation

Vi har nået alle user-cases, og generelt det meste af de ting vores kunde har sat af krav. Vi har dog ikke nået at lave nogen former for tests af systemet hverken Junit eller integration tests. Herudover mangler der nogle stumper kode som kunne optimeres og laves bedre hvis tiden var til det, samt refactoring af koden ville også gøre det mere overskueligt. Vi har i de sidste dage ændret nogle af vores arrays til hashmap, da hasmaps er bedre til at strukturere cupcakes. Dette har dog også skabt problemet at noget af koden er lavet til et array og da vi ikke kan nå at lave det om vil det sige at vi har lavet et hashmap om til array igen for at kunne håndtere det.

Herudover at gøre vores navigationsbar mere overskuelig og ens på alle siderne så det er nemmere at navigere på hjemmesiden.