Deltagere

Tobias Zimmermann - cph-tz11@cphbusiness.dk - Github: CphTobias

Jonas Soldath - cph-js438@cphbusiness.dk - Github: Jonasmik

Alexander Hüsig Petersen - cph-ap322@cphbusiness.dk - Github: AlexanderGitHusig

Gruppe: De Resterende

klasse: Datamatiker 2. semester - efterår 2020

Tidspunkt: søndag, 8. november 2020

Indholdsfortegnelse

Indledning

Den nyopstartet virksomhed Olsker Cupcakes har hyret et team af unge programmører til at udarbejde et website, som de kan bruge til at lancere deres Cupcake sortiment.

Denne rapport er udarbejdet, som en dokumentation af projektet olsker cupcakes, det er formuleret sådan, så folk fra andet semester i datamatiker uddannelsen kan forstå hvad der foregår.

Cupcake projektet har til formål at oprette et website, der gør det muligt at bestille og kombinere cupcake toppe og bunde. Cupcake-toppene/bundene m.m. er lagret i en database, hvilket gør det muligt at gemme og redigere data via SQL.

Baggrund

Virksomhedsbeskrivelse

Olsker Cupcakes er en nyopstartet enkeltmandsvirksomhed som har ben på Bornholm, i byen Olsker. Olsker Cupcakes er ene leverandør til Bornholms cupcake forbrug, og de skal bruge en hjemmeside, som gør det muligt for dem, at kategorisere deres cupcake toppe og bunde, samt holde styr på deres kunder og ordreliste.

Kravs beskrivelse

Hvad er vigtigt for Olsker cupcakes? Hvad vil de gerne have vi fokuserer på? Hvad er mindre vigtigt?

Vi har haft en møde med direktionen i Olsker cupcakes, og har sammen specificeret nogle funktionelle krav, hvordan websitet skal opføre sig.

Vi blev enige om nogle obligatoriske krav for udleveringen af websitet.

Til at starte med skal der være nogle grundlæggende funktioner, såsom at kunden skal have mulighed for, at tilgå sortimentet af cupcake toppe og bunde, og derefter kombinere dem efter eget ønske. Udover dette skal kunden have mulighed for at gemme deres ordre og senere betale eller redigere deres ordre før betaling.

For at kunne gemme og betale deres ordre skal det være muligt for kunden at oprette en profil til websitet. For at mindske tvivl omkring hvorvidt man er logget ind eller ej, vil ens email være synlig på hver eneste side af websitet, så vidt man er logget ind.

Olsker cupcakes vil også gerne have mulighed for at kunne tilgå websitet som administrator. Administratoren vil have mulighed for at se alle betalte ordre, samt rettigheder til at indsætte kredit på en kundes konto.

Ud over disse obligatoriske krav, kom vi også frem til nogle ekstra funktioner som ville forbedre oplevelsen, samt gøre det nemmere at operere websitet. Disse ekstra funktioner ville koste flere penge, og ses som en luksus løsning.

Disse funktioner ville inkludere følgende:

En administrator vil have adgang til, at se alle kunder i systemet, samt deres ordrer, derudover ville det også være muligt at fjerne ugyldige ordre fx hvis kunden aldrig

har betalt. Yderligere ville kunderne have mulighed for at fjerne en cupcake fra deres ordre, og dermed forbedre oplevelsen og gøre systemet mere manøvredygtigt.

Teknologivalg

- IntelliJ IDEA 2020.2.3
- JDBC
- APACHE TOMCAT
- Mysql 8.0
- Mysql workbench 8.0
- Adobe XD
- Github
- Git bash 2.26
- Digital ocean
- Online tool til at lave sequence diagrams:
- https://plantuml.com

Krav

Håb med system

Olsker cupcakes drømmer om, at have en flot og overskuelig hjemmeside, der gør det nemt for kunder, at mixe og matche Cupcake toppe og bunde, i håb om at finde, deres drømme cupcake.

User Stories

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 ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.

US-5 : Som kunde eller administrator kan jeg logge på systemet med e-mail og kodeord. Når jeg er logget på, skal det være klart på hver side.

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

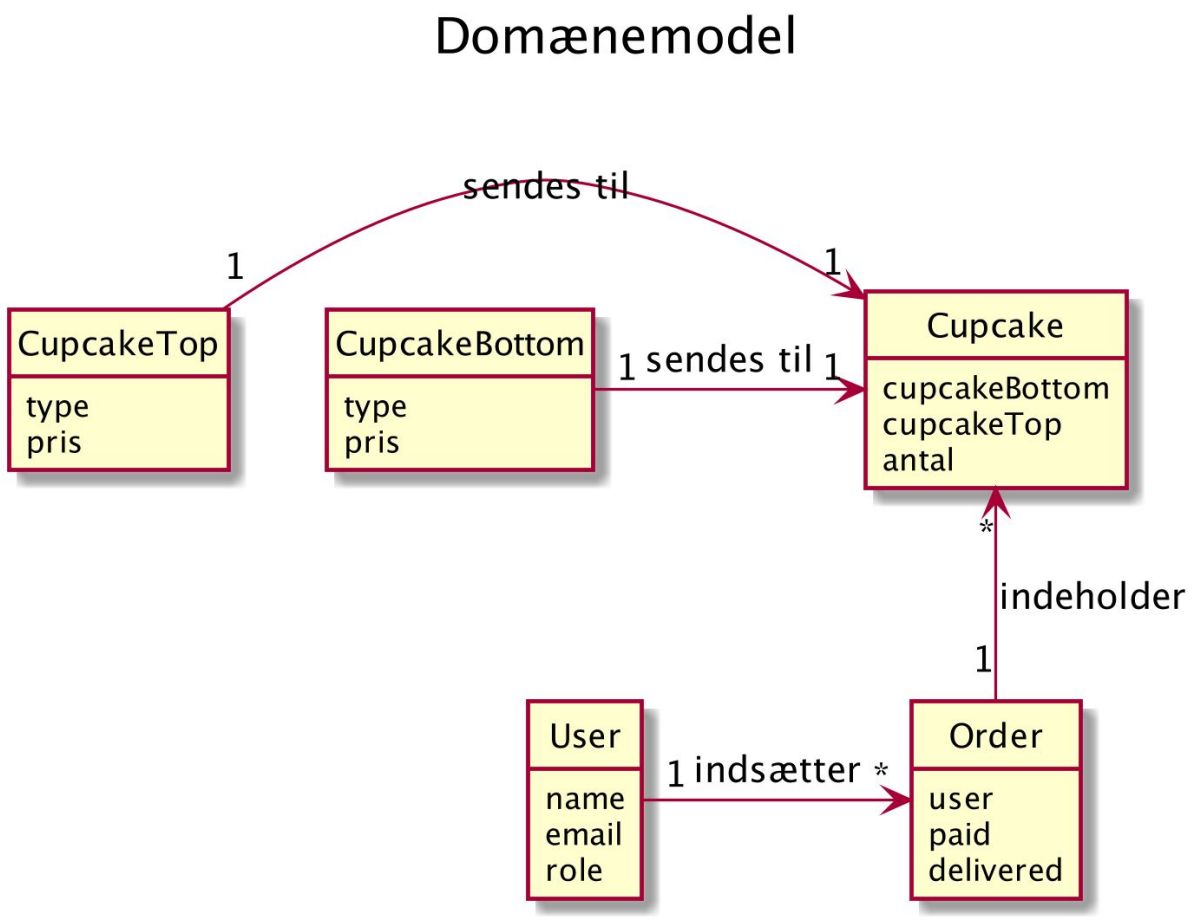
US-7 : Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

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.

Domæne model og ER diagram

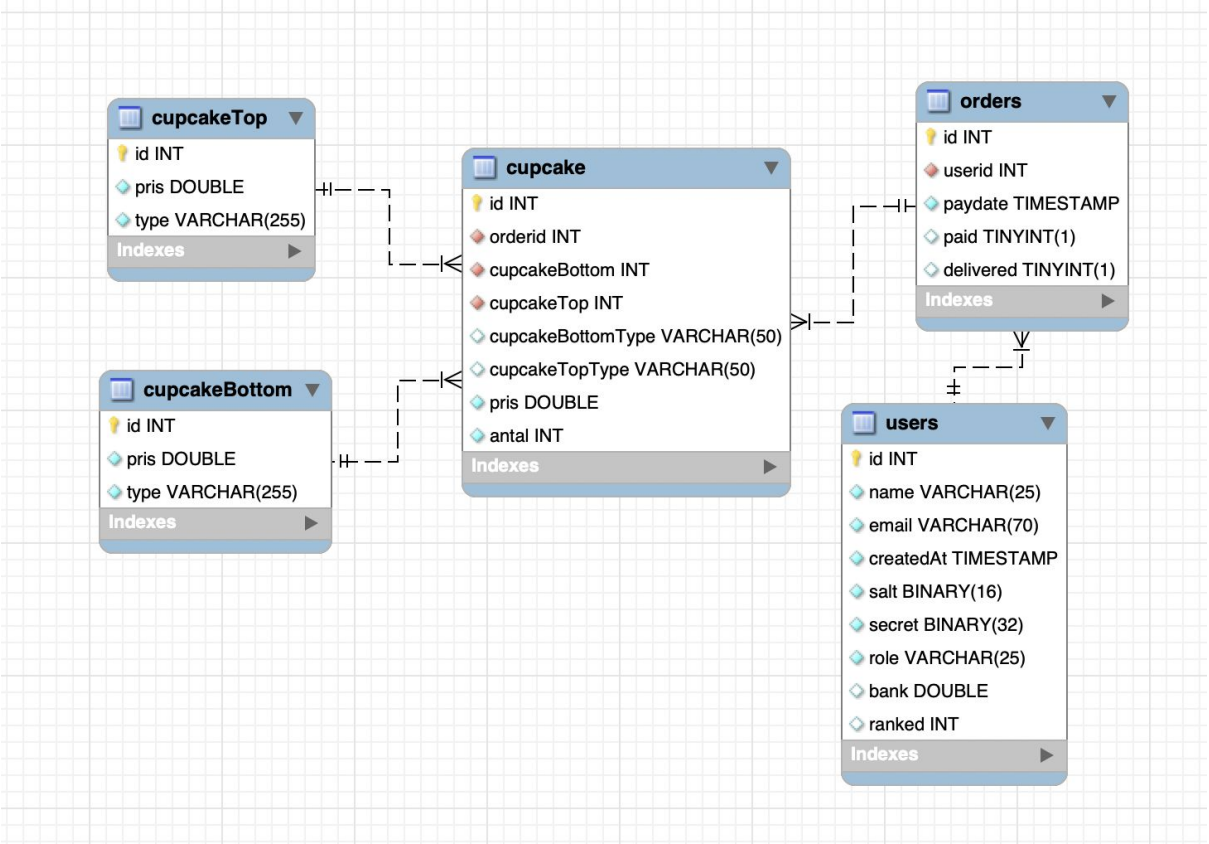
Domæne model



Ovenfor ses et billede af vores domæne model, som illustrerer relationerne mellem vores klasser.

Vores system benytter sig både af 1-til-mange og 1-til-1 relationer. Vi starter med at kigge på vores User, som har en 1-til-mange relation med vores Order, og på samme vis benytter vores Order sig også af en 1-til-mange relation med vores Cupcake. Vores Cupcake er derefter bygget op af to 1-til-1 relationer med hhv. Cupcaketop-bottom.

ER diagram



Forklaring

Vores system fungere på den måde at når kunden laver en cupcake, bliver toppingen og bunden lagt ind i en cupcake. Denne cupcake bliver sat i en ordre, med brugerens id, når kunden betaler kan vi derfor checke hvor mange cupcakes vi har med brugerens id, og derefter sætte “paid” til true.

Vores passwords er umiddelbart ikke mulige at se i databasen, fordi de bliver lavet om til binary code.

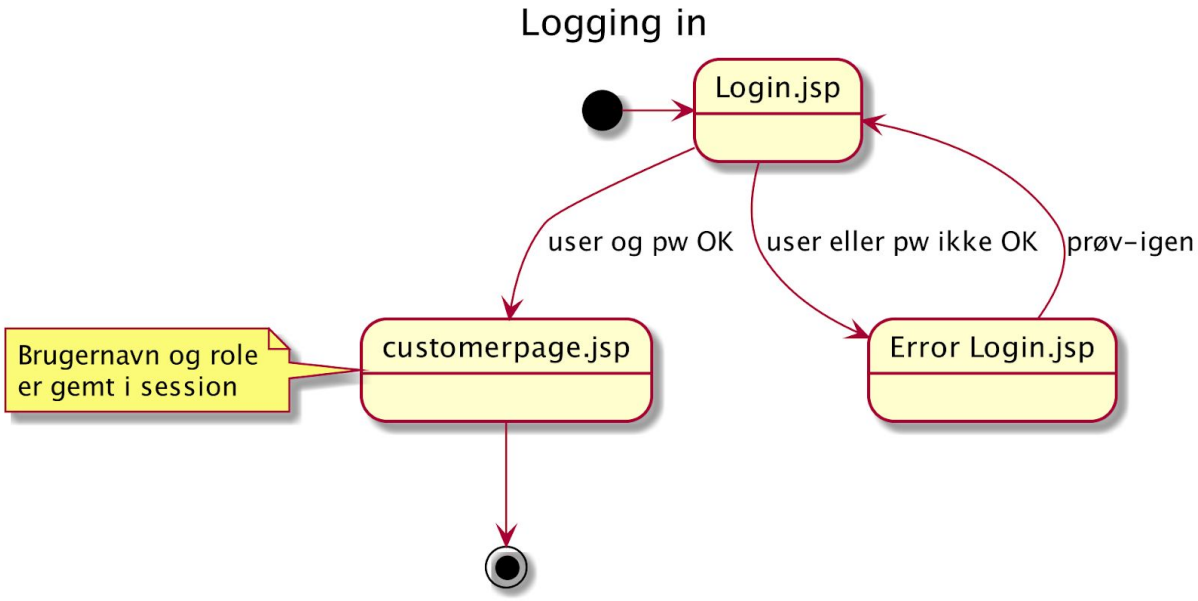
Når det kommer til 1-til-1 relationen mellem cupcake bund/top og cupcake, er det et bevidst designvalg, for at gøre databasen mere overskuelig. Alternativt ville det være nødvendigt, at have gemt hver eneste mulige kombination af cupcake bund og top, som valgmulighed under cupcake. I stedet er det smartere at vi har delt dem op, hvilket også tillader os, at give hver enkelt bund/top en individuel pris..

Overvejelser

Der er nogle steder, hvor vi overvejede hvilke løsninger der ville være bedst, et af dem var da vi skulle have en måde at vise, at en ordre var blevet betalt. Vi endte med, at gå med løsningen, hvor vi har en boolean som hedder “paid”, som vi bare sætter til true hvis ordren er betalt. Vi kunne også have lavet en helt ny tabel, som vi kunne have kaldt kurv, når ordren var betalt kunne vi have ført ordren fra kurv over i ordre. Det ville have gjort flere ting en del simplere og nemmere at arbejde med, såsom at finde ordre som ikke er betalt endnu.

Navigationsdiagram

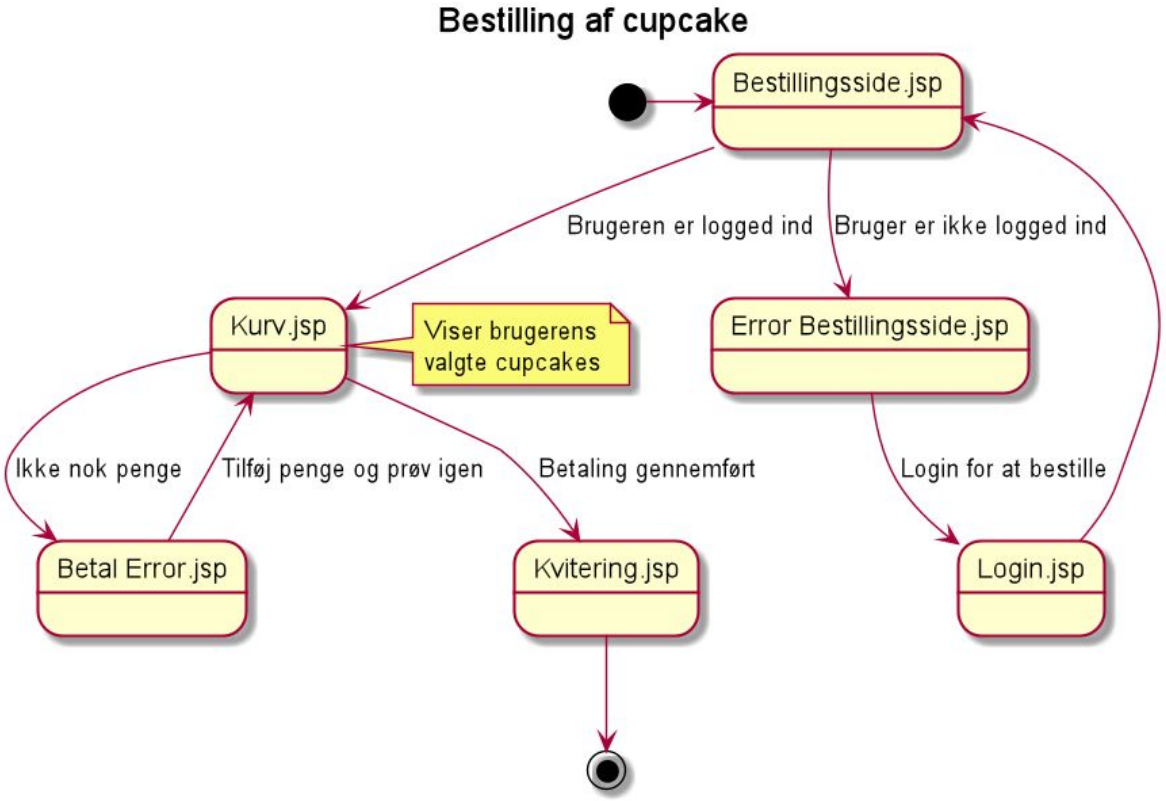
1. Login



Figur 1, illustrerer vores login portal. Ved login bedes bruger indtaste brugernavn og kodeord. Er brugernavn eller kodeord forkert, bedes brugeren indtaste korrekte oplysninger igen.

Der vil blive tjekket for rolle så snart brugeren logger ind, hvis brugeren er en admin vil der være flere funktioner tilgængelige end for en kundeprofil, fx admin page.

2. Bestilling

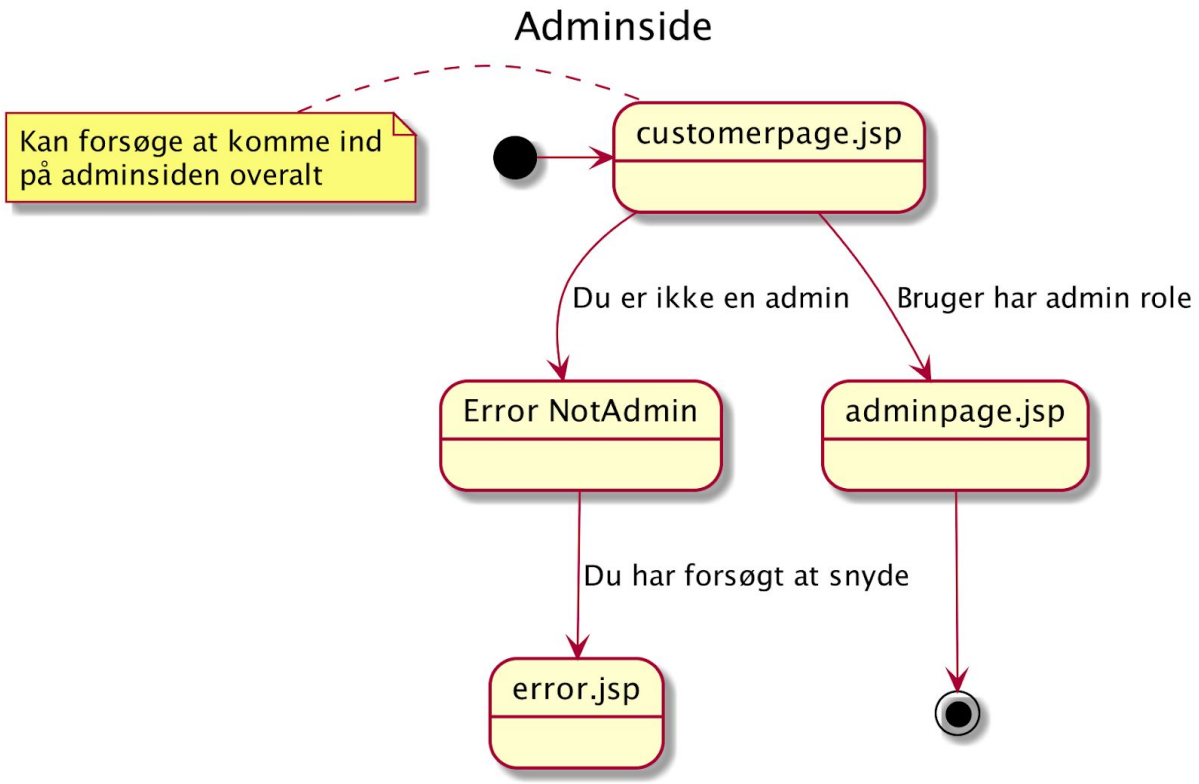


Ved bestilling af cupcakes vil brugeren blive bedt om at logge ind hvis de ikke allerede er det, da bestilling af cupcakes ikke er mulig uden et gyldigt brugerlogin. Bestilling foregår ved at en bruger, som er logget ind, trykker på “kurv” ikonet, hvor de derefter kan se deres indkøbskurv og vil have mulighed for at bestille. Ved bestilling vil der blive tjekket for hvor mange penge, der er

på brugerens konti. Er beløbet ikke tilstrækkeligt i forhold til ordren, vil brugeren blive sendt til en errorpage. Errorpagen vil oplyse om manglende penge på konti og brugeren vil have en funktion, til at tilføje rigtige penge til deres konti. (Vi fik ikke tilføjet denne funktion eftersom det er en skoleopgave og ikke en officiel hjemmeside, så derfor er det kun admin der kan tilføje fiktive penge).

Ved succesfuld køb af ordre, vil brugeren blive sendt videre til en kvitteringsside, hvor ordre samt betalt beløb vil blive vist.

3. Admin Side

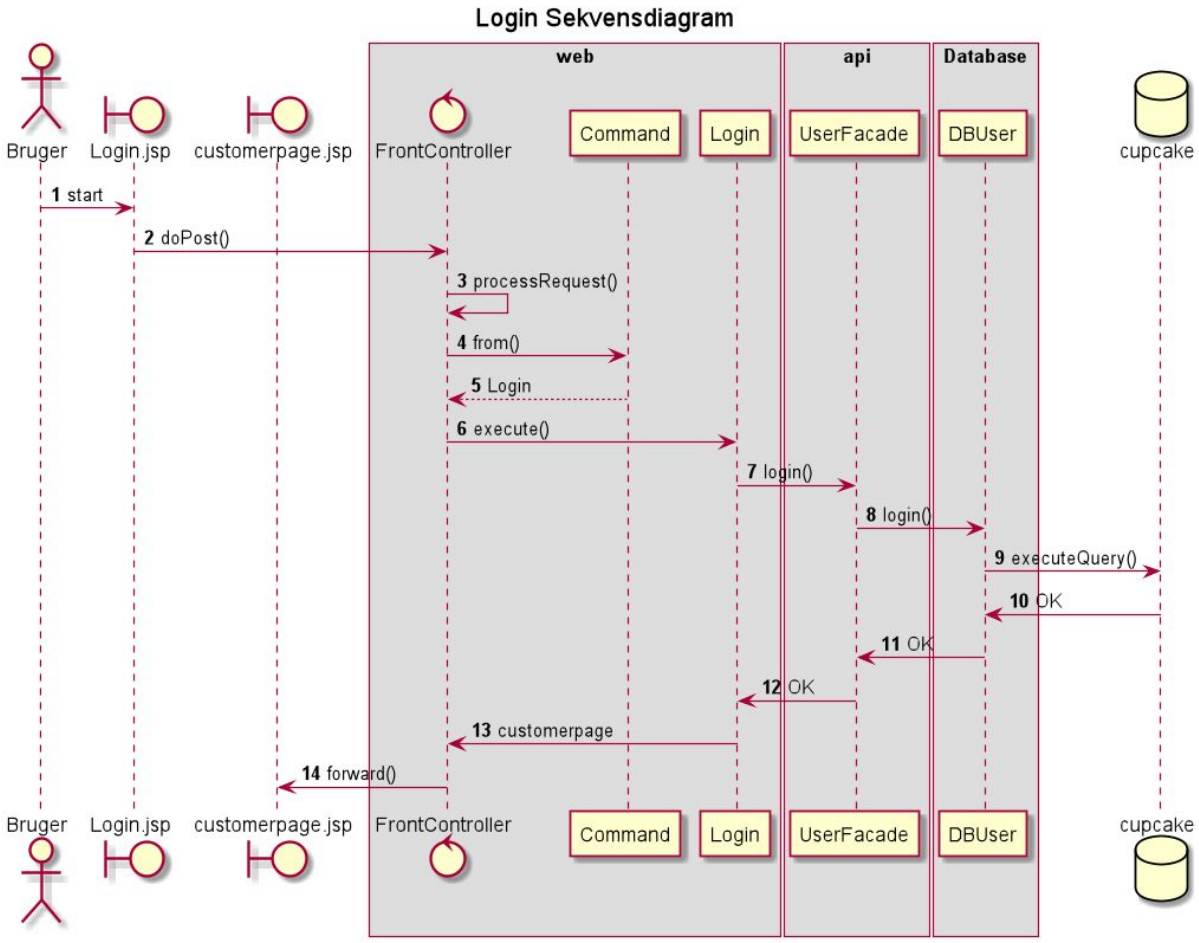


Som administrator efter du har logget ind, vil du kunne se en “Adminpage” i navigationsbaren. Denne vil tage dig til admin siden, hvor der vil være diverse admin funktioner, såsom at vise ordre i systemet.

Hvis du derimod ikke er en admin og du forsøger, at komme til admin siden f.eks. via URL’en vil du blive sendt til error siden.

Sekvensdiagrammer

1. Login



Brugeren er inde på login siden, og skriver sin email og password. Disse argumenter bliver derefter videresendt til Front Controlleren, som søger i Command om der

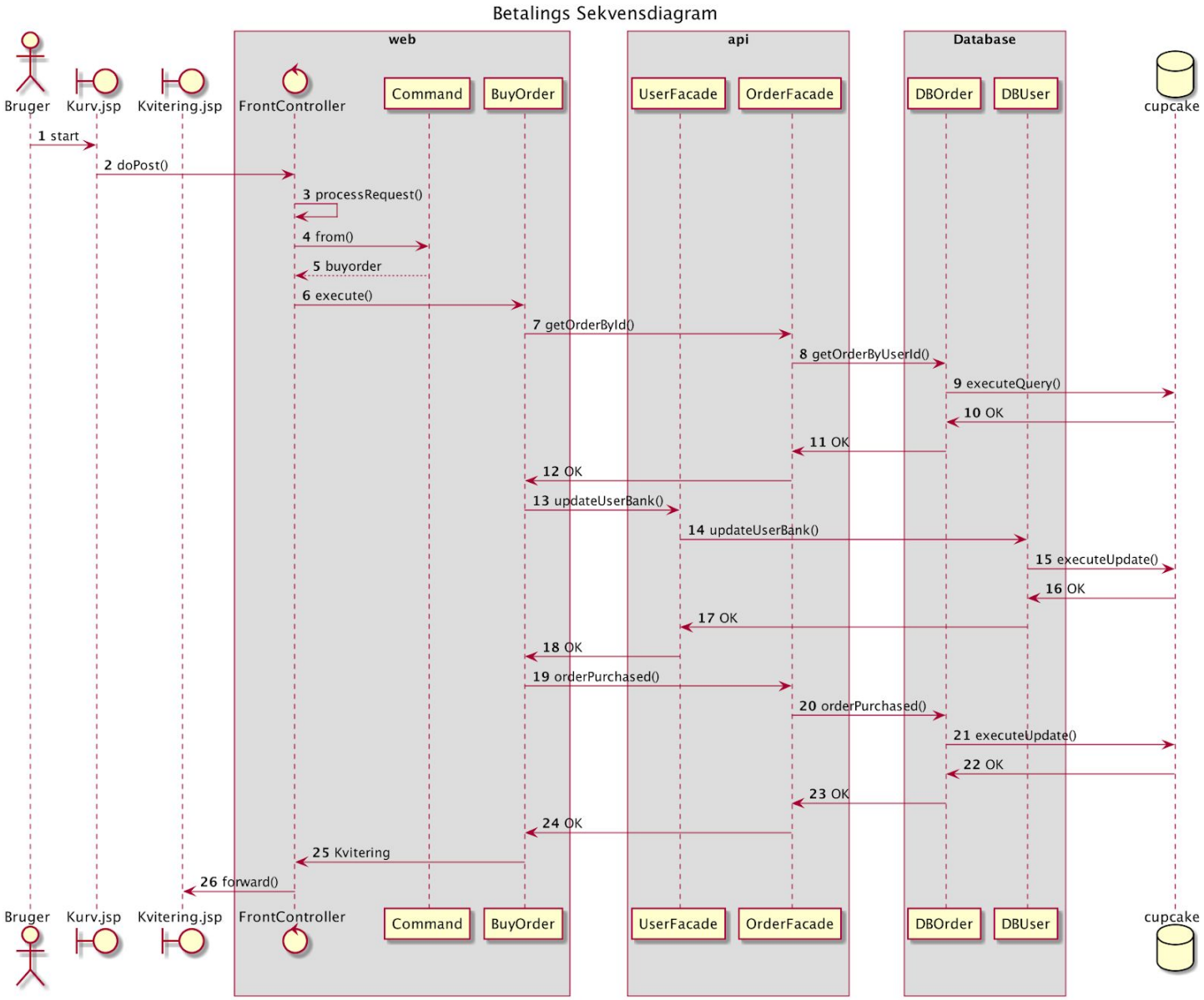
findes en værdi med navnet “target”. “Target”’s value bliver derefter returneret til FrontControlleren, som så i dette tilfælde er “login”.

Nu kalder FrontControlleren login.execute(), som kalder login i api’ets UserFacade. User Facaden kalder DBUser.login() som så kalder databasen, med en executeQuery().

Vi har nu fundet en bruger der matcher den indtastede email. Nu bliver denne bruger sat i “sent session”, som kan bliver kaldt overalt i programmet.

Hvis vores database kald ikke finder en user eller at det er forkert password så vil den returnere null. Hvilket vil resultere i at vores if statement inde i Login.java returner “Login” i stedet for customer page og så vil der også blive printet en error.

2. Betaling



Brugeren er inde i kurven, og trykker på betal ordre. Denne knap sender nogle gemte værdier med til FrontControlleren (target, brugerens bank, brugerens id og ordrens pris). Disse værdier bliver derefter videresendt til FrontControlleren, som

søger i Command om der findes en værdi med navnet “target” og dette targets value bliver derefter returneret til FrontControlleren, som så i dette tilfælde er “buy order”.

Nu kalder FrontControlleren buyorder.execute(). Vi er nu i BuyOrder klassen, som starter med at undersøge om der rent faktisk er noget i ordren. Hvis der ikke er det bliver brugeren returneret til kurven.

Den kalder nu en metode som beregner hvad kundens bank ville være efter at have købt disse cupcakes. Hvis denne beregning ender med at være under 0 kr, bliver kunden sent tilbage til kurven med en besked om, at de ikke har nok penge på deres konti.

Vi prøver nu på at finde brugerens ordre, som bliver gjort med metoden getOrderById inde i api’ets OrderFacade. Facaden kalder nu DBOrder, som eksekverer en query i databasen for at finde kundens ordre.

BuyOrder kalder nu på metoden updateUserBank i UserFacaden som bruges til, at opdatere brugerens pung, med den forhenværende udregnede nye bank. UserFacaden kalder nu updateUserBank i DBUser, som derefter kalder databasen med en .executeUpdate.

BuyOrder kalder nu OrderFacaden med orderPurchased, som derefter kalder DBOrder med orderPurchased. DBOrder eksekvere nu en .executeUpdate i databasen og sætter paid til true.

Nu returnerer BuyOrder “Kvittering” til FrontControlleren, som derefter kalder .forward() til Kvittering.jsp med den nye betalte ordre.

Særlige forhold

Vores Cupcake projekt har været lidt af en lærings process, når det kommer til sessions. Vi startede ud med kun at gemme useren i vores session, og derefter viste vi ting, som vores ordre i et request. Vi har senere hen fundet ud af, at vi nok skulle have brugt sessions noget mere til f.eks. vores ordre.
Vores exception handling er ikke helt optimal, vi skulle have brugt mere tid på at lave individuelle exceptions til vores domæner, og kalde dem de rigtige steder.
I vores projekt bruger vi dog factories, til at validere inputtet af nylavede objekter såsom en ny cupcake. Den vil gå ind og checke om valideringen af f.eks. en string til en integer er gået godt.

Når en ny bruger bliver lavet har vi lavet noget sikkerhed for brugerens skyld, med at checke om han/hun har stavet deres password korrekt. Dette bliver gjort ved, at få brugeren til at skrive sit password to gange.
Når passwordet bliver gemt i databasen, genererer vi nogle random bytes, som i kombination med vores password bliver krypteret til binær-kode for at beskytte sensitiv data.
I vores program gemmer vi ikke bare passwordet i databasen, vi pakker det ind i bytes, og derefter lægger det ind i databasen.
For at finde ud af om en bruger taster et password korrekt ved login, har vi lavet en isPasswordCorrect(), som undersøger ved hjælp af et givet password om de matcher i bytes.

For at undersøge hvilken brugertype der er logget ind, bruger vi en række i databasen som hedder “role”. Denne “role” tager bare imod en string. Hvis rollen er lig med “admin”, bliver dette sat som en attribut i systemet. Vi kunne også have valgt at bruge en enum til at gøre det samme.

Status på implementation

Vi nåede egentlig alt, som var vigtig for os at få klaret, der er selvfølgelig nogle små mangler vi gerne ville have nået også. Der er vores FAQ side, som vi ikke fik sat op, vi vurderede den som mindre vigtig og det endte med, at blive skubbet tilbage indtil vi ikke rigtig havde tid til det mere. Vi ville gerne have lavet en “settings” til alle users, sådan at de havde mulighed for at ændre email, password og navn. Det var bare for stort at lave eftersom, at det ikke var vigtigt for selve opgaven, selvom at vi syntes at det var en relevant funktion.

Til sidst havde vi så nogle mindre ting, såsom admin page ville vi gerne have stylet lidt bedre og det samme med vores footer. Det sidste ville nok være at få hjemmesiden til at virke lidt bedre på andre platforme end computer, sådan at når hjemmesiden blev åbnet på en mindre skærm ville det se bedre ud. Så hvis vi havde haft lidt mere tid, ville vi gerne have arbejdet noget mere med det, så det ville se godt ud på alle platforme.

Test

Vi har i denne opgave ikke gået så meget op i tests, og derfor har vi kun testet CupcakeFactoryen. I disse tests bliver der lavet en test på “isValid”, som er vores validation af at alle de validations, som skulle have gået godt, rent faktisk gik godt.

Vi har også lavet en test for de exceptions, som skal kastes rent faktisk bliver kastet, hvis der går noget galt.

Denne test af Factories er vigtig fordi, det er rigtig godt at vide om vi rent faktisk får det rigtige input ud til vores database.