Cupcake

Lavet af:
Frederik Elbo Thorup - cph-ft36@cphbusiness.dk - github: frederiket1912
Ahmad Talha Noory - cph-an148@cphbusiness.dk - github: TalhaNoory
Sebastian Bernth Jepsen - cph-sj383@cphbusiness.dk - github: SebastianBernth
Frederik Juel Andersen Hurup - cph-fh132@cphbusiness.dk - github: Fhurup

Klasse: B

Rapporten dokumentere et projekt der fandt sted fra d. 25/2 2019 til d. 8/3 2019.
Rapporten blev afleveret d. 15/3 2019.

Link til vores cupcake shop: http://167.99.222.181/Cupcake/

(I kan selv oprette en bruger, eller logge ind med admin@admin.dk og koden “1234”)

Indledning

“Cupcake” er en opgave der handler om at udvikle et system til en cupcake virksomhed, der vil sælge sine produkter via en hjemmeside. Systemet skal tage sig af, at brugerne skal kunne oprette sig eller logge ind, hvis de allerede er oprettet, og selve bestillingen af cupcakes. Brugerne skal selv hente de cupcakes de bestiller og cupcakesne er færdige lige så snart ordren er blevet lavet. Det betyder at systemet ikke behøver gemme på brugernes adresser og det er ikke nødvendigt at udvikle funktionalitet til opbevaring, afsending og levering af cupcakes.

Cupcakes består af en top og en bund som begge skal vælges fra en prædefineret liste. Toppene og bundene har hver især en pris associeret med sig og den totale pris for cupcaken er toppens og bundens pris lagt sammen. En cupcake skal bestå af både en top og en bund, altså kan man ikke bestille en cupcake uden topping. De cupcakes brugeren bestiller gemmes løbende i en shoppingcart og først når brugeren er færdig med at vælge cupcakes og trykker “buy” bliver ordren gemt i databasen.

Når en bruger bliver oprettet, beder vi om følgende information: username, password, email, balance og om de er regulære brugere eller admins. Vi har lavet det så når brugere bliver oprettet bliver de automatisk logget ind også. Når returnerede brugere skal logge ind, beder vi om brugerens email og hans password. Grunden til at vi valgte email i stedet for username er at vi har valgt emails til at være primary key i vores tabel med brugere, og vi bliver derfor nødt til at checke deres password op imod deres email, da det ellers kunne forekomme at to brugere havde samme username og password. Hver bruger har en balance som de bruger til at betale for deres cupcakes med. Selve betalingen, og hvad det indebærer med sikkerhed og dankort osv., tager vores system sig ikke af. Det eneste vores system gør er at tjekke prisen på brugerens endelige køb imod deres balance, hvis balancen er større, trækkes det endelige beløb fra brugerens balance når de trykker “buy”.

Websiderne skal styles med css og vi er blevet opfordret til at bruge en bootstrap template.

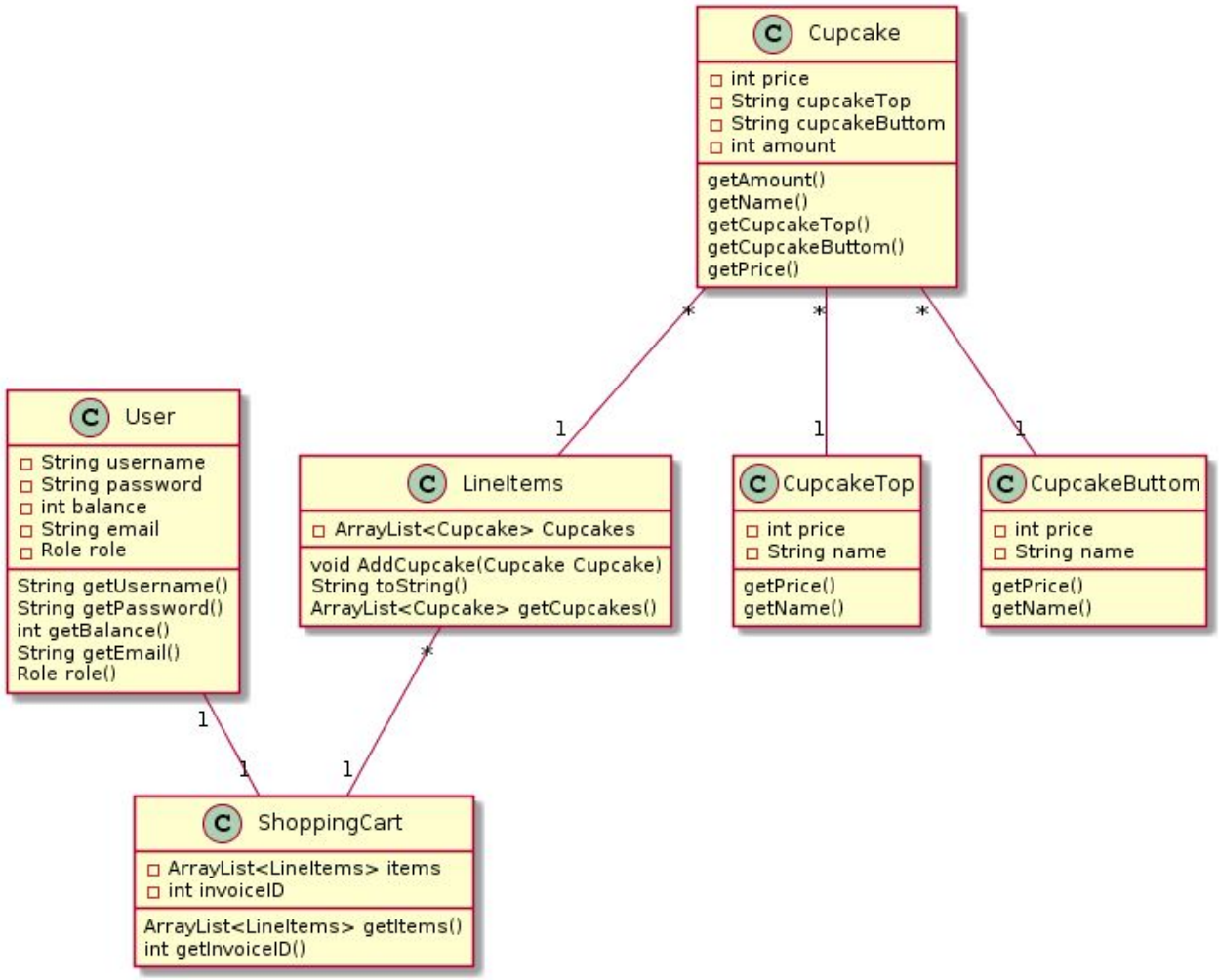
Teknologi valg

For at lave systemet har vi brugt følgende teknologier:

- Netbeans 8.2
    - Maven web application med html, jsp, css, servlet og regulære java klasser
- MySQL Workbench 8.0.15
- Apache Tomcat 8.0.27.0
- DigitalOcean server med Ubuntu 18.10
- MySQL 8.0.15 på DigitalOcean serveren

Domæne model og ER diagram

En domænemodel kan bruges til hurtigt at få et overblik over hvad der er i de enkelte klasser og hvordan deres indbyrdes forhold er.

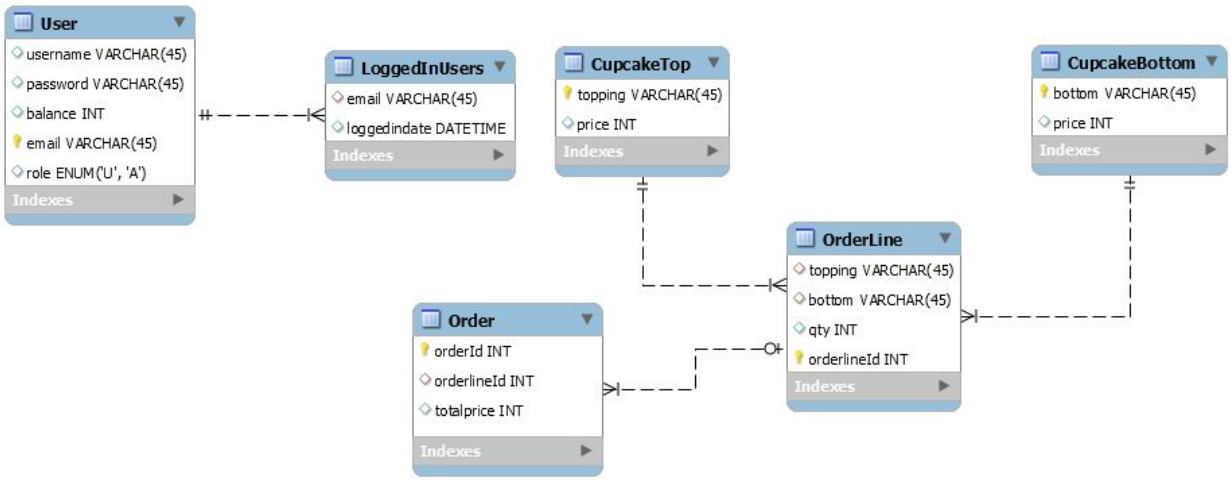


I vores domæne model kan man se at en Cupcake har en èn til mange relation til CupcakeTop, CupcakeButtom og LineItems. Det betyder at en Cupcake Kan have en CupcakeTop og en CupcakeButtom, men en CupcakeTop og en CupcakeButtom godt kan være på flere Cupcakes.

I forhold til LineItems, betyder det at LineItems kan bestå af mange Cupcakes, mens en Cupcake kun kan være på en LineItems. LineItems er derudover en èn til mange relation til ShoppingCart. Det vil sige at en ShoppingCart godt kan bestå af mange LineItems, men at LineItems kun kan være på en ShoppingCart.

Til sidst har ShoppingCart en èn til en relation til User, som betyder en User kan have en ShoppingCart, og en ShoppingCart er tilknyttet en User.

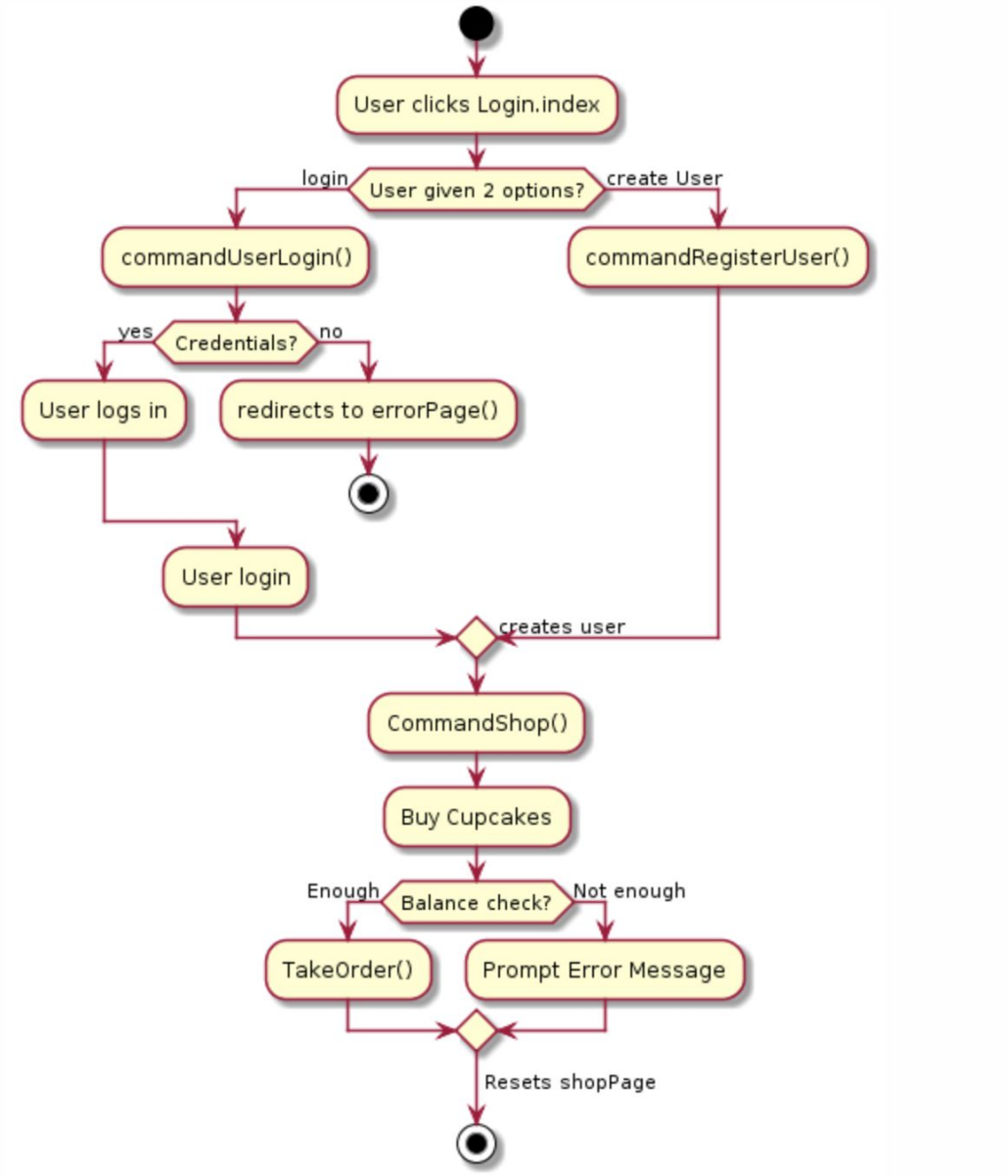
ER diagrammet giver et overblik over de forskellige tables og deres forhold til hinanden.
Vi har valgt at lave email på useren til primary key, fordi det er unikt, der kan ikke være to ens emails.



Man kan se vores tabeller i ER diagrammet overholder de 3. normal former:

1. *normal form*:
· Ved at vores kolonner kun indeholder en værdi.
· Alle værdier i den samme kolonne er af den samme type.
· Alle vores kolonner har et unikt navn.
2. *normal form*:
· Vores primary keys er ikke Candidate keys, det vil sige at de ikke er en kombination af to attributter. Derfor kan vores attributter ikke være afhængig af kun èn del af en primary key. Og derfor har vi ikke Partial Dependency.
3. *normal form*:
· For at være i den 3. normal form må man ikke have Transitive Dependency, det er når en non-prime attribut er afhængig af en anden non-prime attribut. Vores er afhængige af primary keys.

Navigationsdiagram



Følger man navigationsdiagrammet, vil brugeren blive mødt af en index side (Startside) hvor han/hun kun vil have mulighed for at gå videre til login/CreateUser page. I det endelige produkt ville disse to funktioner være delt ud på 2 pages. En til login og en create user (Kun vist hvis brugeren er logget ind som Admin).

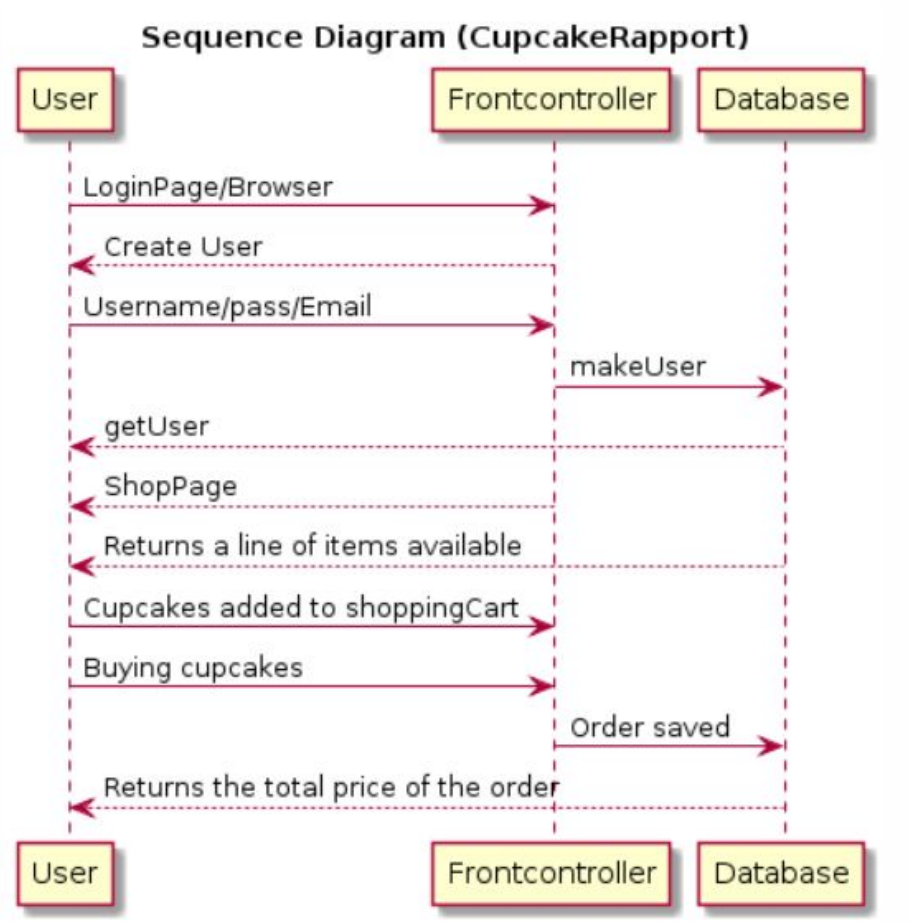
Når bruger vælger login, vil han/hun blive bedt om et e-mail og et password. Informationerne vil derefter blive sendt videre til systemet som vil gå ned i

databasen og checke om de er korrekte. Hvis ukorrekte informationer er givet, vil brugeren blive sendt til en error page. Hvis informationerne er korrekte, vil brugeren blive logget ind og sendt videre til shop pagen.

Vælger brugeren at create user, ville han/hun skulle indtaste username, password, balance, email og om den nye bruger er admin eller almindelig bruger. Brugeren vil derefter blive oprettet og den oprettede bruger vil derefter blive sendt videre til shop pagen.

På shop page vil brugeren have mulighed for at tilføje cupcakes til sin indkøbskurv og bagefter købe dem. Løbende som brugeren tilføjer cupcakes til sin kurv vil shop pagen opdatere så bruger kan følge med i hvor meget han/hun har i sin kurv og den totale pris. Når bruger er tilfreds med sit indkøb, vil han/hun trykke køb. Når brugeren gør dette, vil systemet checke om personens balance er høj nok til at kunne betale for brugerens indkøb. Er balancen høj nok vil systemet tage personens ordre og gemme den i databasen og derefter resette shop pagen, skulle brugeren have lyst til at lave endnu en bestilling. Er balancen ikke høj nok vil bruger blive promptet med en fejlmeddelelse som fortæller brugeren at der mangler penge.

Sekvens diagrammer



Et sekvensdiagram bruges til at vise samspillet mellem forskellige dele af et system over tid. Altså at vi starter i toppen hvor det første der sker, er at user sender et request om at se loginpagen, og det sidste der sker er at useren får at vide hvor mange penge han har brugt på cupcakes.
Vi har valgt at fokusere på tre dele af systemet som vi synes giver det bedste overblik over et normalt forløb gennem programmet. I vores udgave af diagrammet har vi valgt en process der starter med at oprette en ny bruger, men det kunne lige så godt have startet med at en bruger skulle logge ind, diagrammet vil se næsten ens ud.
Den første del af systemet vi har valgt at fokusere på, er useren, der repræsentere de ting brugeren interagere med, altså jsp siderne, og de input brugeren kommer med. Den anden del er frontcontrolleren som er den servlet der bestemmer hvilke sider der skal vises, og hvilke ting der skal hentes fra eller

sættes ind i databasen, afhængig af brugerens input. Den sidste del er selve databasen, som bliver brugt til at gemme information om brugerne, cupcakes og de ordrer der bliver lavet. Grunden til at vi har valgt netop disse tre dele af systemet er at de repræsentere de tre datalag. User repræsenterer præsentationslaget, frontcontrolleren repræsenterer logiklaget og databasen repræsentere datalaget. En forståelse af hvordan informationer og forespørgsler flyder mellem de tre lag hjælper i høj grad til at forstå den overordnede sammensætning af systemet.
Pilene mellem user, frontcontroller og database viser hvordan de tre dele interagerer med hinanden og pilenes retning viser i hvilken retning dataen flyder. Disse interaktioner kan enten ske via forespørgsler, fx når frontcontrolleren skal bruge username, password og email til at oprette en bruger, eller via metoder, fx når frontcontrolleren laver en user og gemmer den i databasen, eller når der bliver genereret en shop page ud fra den user information der bliver hentet i databasen.

Hvad er formålet med dette sekvensdiagram?
Formålet med sekvensdiagrammet er for at simplificere det, så enhver kan forstå hvad det er der foregår, så personen kan følge hele processen for hvad der skal ske. Hvis vi tager den slaviske rækkefølge, ser vi hvordan det kører. User kommer ind til en Loginpage/Browser (JSP-fil), Frontcontroller modtager denne information og svarer tilbage med at sige til User, at personen skal lave en bruger med navn, password og email. Når User så har fået disse informationer på plads og så trykker “create” så sender frontcontroller dette videre til databasen, da det er den som laver brugeren. Så den modtager en metode som så hedder “makeUser”, den tygger det her igennem og spytter så en anden metode ud “getUser”, når en user så bliver oprettet, bliver han automatisk logget ind, og derefter får han så lov til at komme videre til ShopPage. Denne næste page kaldes ShopPage (JSP-fil), som består af en linje af forskellige cupcakes med valg af top og bund, som kommer fra databasen via netbeans. Så nu kan User begå sig frit omkring i denne ShopPage og købe sig til alverdens cupcakes. Når User så skal købe, lægger personen det ind i en ShoppingCart, og bestiller ud ved at trykke på “buy”. Ordren bliver så sendt fra frontcontroller til databasen, som så sender den totale pris ud til User.

Særlige forhold

Vi har valgt at gemme brugeren som er logget ind i sessionen samt, et tomt Lineitems objekt, en liste med Cupcake Tops og Bottoms(Hentet fra databasen). Brugeren bliver gemt i session da vi i shoppen har brug for både brugers navn og balance for at kunne tage imod hans ordre, derudover har vi brug for listerne med Tops og Bottoms for at kunne vise brugeren hans muligheder for at customize hans cupcakes. LineItems bliver gemt i session da vi har brug for den i forhold til løbende at kunne opdatere shoppen som brugeren tilføjer cupcakes til sin indkøbskurv.

Som programmet ser ud nu, er der ikke data validering på shoppen så skulle en bruger vælge at indtaste et invalid input vil siden blive blank. Dette ville kunne håndteres ved checke inputtet i en try/catch før vi prøver at tilføje cupcaken til indkøbskurven. Derefter ville man kunne lave en custom exception som ville prompte bruger med en passende fejlmeddelelse som ville informere brugeren om fejlen personen har lavet.

I jsp siden Login/CreateUser har vi data validering via når en bruger logger ind, vil hans indtastede information blive checket i databasen og hvis brugeren ikke eksisterer eller passwordet er forkert, vil systemet fange en SQLexception og brugeren vil modtage en fejlmeddelelse. På samme måde vil brugeren modtage en fejlmeddelelse hvis han/hun prøver at oprette en bruger hvis primary key allerede existere i databasen(I dette tilfælde e-mail adressen).

I forhold til sikkerhed, har vi ikke haft erfaring med kryptering af data, så den eneste sikkerhedsforanstaltning vi har taget er at vi tager imod brugers password som hidden parameter så det ikke vises i URL når brugeren bliver videresendt til shop.jsp

Status på implementation

Vi er overordnet set tilfredse med hvor langt vi nåede med projektet. Med det sagt er der en række ting vi ikke fik implementeret og nogle ting som ikke fik den finpudsning de havde brug for.

Det første der kan nævnes, er statussen på vores database. Den indeholder nogle elementer der er levn fra tidlige ideer om hvordan vi skulle løse forskellige problemer. Blandt andet har brugerne både et username og en email, men vi bruger reelt set ikke usernamet til noget. Vi identificerer brugere på deres email når vi skal finde dem i databasen, de bruger deres email når de logger ind og vi fik først til sidst ændret shop pagen til at sige “welcome ‘username’” i stedet for “welcome ‘email’”, bare så vi følte at vi brugte username et eller andet sted. Derudover har vi også en tabel der hedder loggedInUsers som vi havde tænkt os at bruge til at holde styr på om man var logget ind, når man skifter rundt mellem siderne. Efter vi lærte om hvordan man kunne bruge session til at holde styr på det i stedet er tabellen blevet meningsløs. Vi har dog valgt at lade den stå, så vi senere kan kigge tilbage på hvordan vores tankeproces var i starten af projektet.

Vores user klasse har en attribut der afgør om en bruger er en regulær bruger eller en admin. Som projektet står nu, er der ingen forskel på om man er user eller admin. Vi havde planer om at admins skulle have adgang til flere jsp sider hvor de kunne se en oversigt over alle ordre eller alle brugere og hvornår de var blevet oprettet. Fx ville det give mening at kun admins kunne oprette andre adminbrugere og at nye folk der kommer til siden kun kan oprette regulære brugere. En ide var også at nogle af de eksisterende jsp sider skulle se anderledes ud hvis det var en admin der var logget ind, måske skulle de kunne købe cupcakes gratis!

Vi ville gerne have lavet en menulinje i toppen der gik igen på alle siderne, men vi stødte ind i problemer med at få den til at spille sammen med den css bootstrap template vi valgte at bruge. Der er levn af menulinjen i projektet. I samme dur ville vi gerne have brugt lidt mere tid på styling. Fx står prisen på toppe og bunde ikke lodret over hinanden på shop pagen og siderne kunne have haft gavn af nogle billeder, fx til illustration af de forskellige toppe og bunde.

Der blev lavet en errorpage der håndterer forkerte indtastninger ved oprettelse af ny brugere og ved login, men vi fik ikke overført brugen af errorpagen til forkerte indtastninger i shoppen.

Shop pagen har desuden et par mindre problemer:

Når en bruger tilføjer nye cupcakes til sin cart, er der ikke et tjek der ser om den cupcake allerede er i carten, så i stedet for bare at ligge quantity til den tidligere valgte samme cupcake, så bliver cupcaken bare tilføjet igen som en ny linje.

Der er ikke funktionalitet til at opdatere balance i databasen efter køb, og derfor heller ikke funktionalitet til at opdatere det sted hvor brugeren kan se hvor meget han/hun har tilbage på sin konto.

Der er ikke funktionalitet for en bruger til at sætte flere penge ind på sin konto, hvilket måske betyder mindre når de i forvejen ikke kan bruge de penge de har.

Det er meningen at shop pagen skal resette når man trykker “buy”. Som det er nu bliver ordren added til databasen, men for brugeren ser det ikke ud som om der sker noget og shopping carten bliver heller ikke tømt.