iCupCake

Deltagere:

Jesper Rusbjerg cph-jr???@cphbusiness.dk github: JesperRusbjerg

Pernille Halberg Lørup cph-pl160@cphbusiness.dk github: pernillelorup

Mette Rytt cph-mr383@cphbusiness.dk github: metterytt

Rasmus Helsgaun cph-rh161@cphbusiness.dk github: Neytrax

2. semester, datamatiker A-klasse

16. marts 2018

Indledning

Dette projekt omhandler konstruktionen af en webshop, der sælger cupcakes. Grundlæggende kan man som kunde i denne butik vælge mellem forskellige bottoms og toppings, som så bliver sat sammen til én cupcake. Man kan bestille flere cupcakes, og cupcakes i forskellige sammensætninger af bottoms og toppings. Den særlige ved denne butik er at éns ordre er klar til afhentning i samme øjeblik man har trykket ’Bestil’ – altså ingen ventetid, men heller ingen levering.

Baggrund

Den (fiktive) virksomhed, som skal bruge dette system, har tidligere forsøgt sig med levering af cupcakes med posten – en løsning som af forskellige årsager viste sig ikke at fungere. Derfor skal nu denne webshop søsættes med en fornyelse af konceptet.

Krav til systemet omfatter:

* Man skal som kunde kunne registrere sig og siden logge ind for at handle i webshoppen
* Man skal kunne sammensætte cupcakes med valg af bottoms og toppings
* Betaling foregår ved at der sættes penge ind på kundens konto, noget som foreløbig bliver varetaget af et andet system. Dog skal dette system håndtere at trække det beløb fra kundens konto som der er handlet for
* Ligeledes skal systemet håndtere at man ikke kan handle for mere end det beløb der står på éns konto

Teknologi

MySQL

Netbeans 8.2 Patch 2

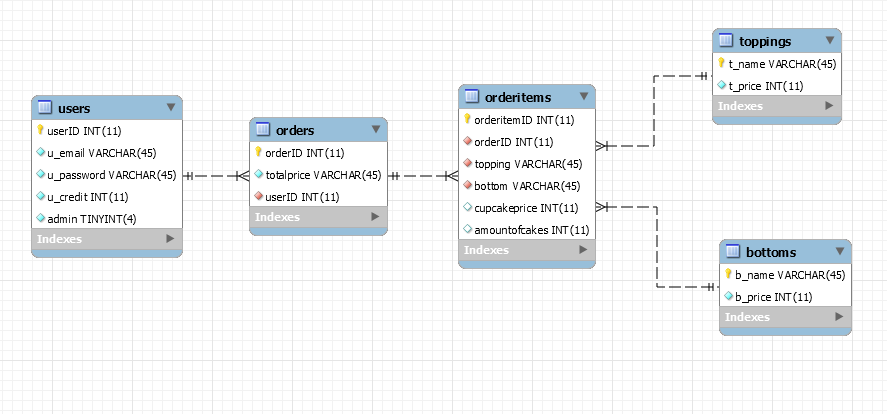
DigitalOcean

Tomcat

Linux Ubuntu

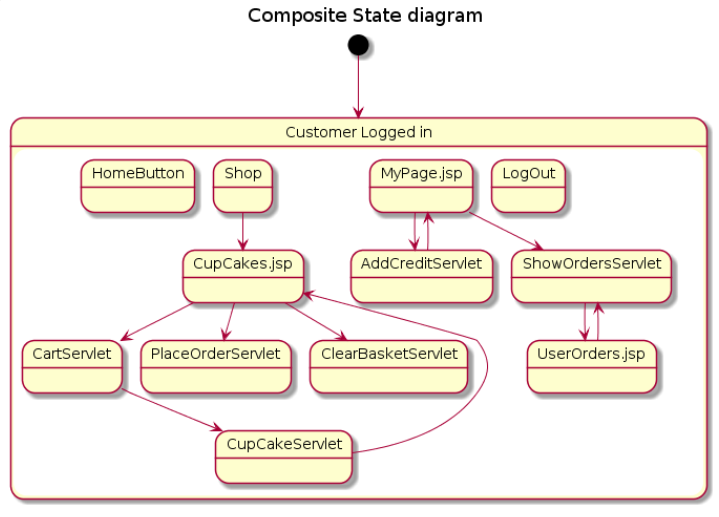
Er i tvivl om hvor mange af disse vi skal skrive på. Det må vi spørge om.

Foreløbigt ER-diagram (skal måske ændres iht. TODO listen):

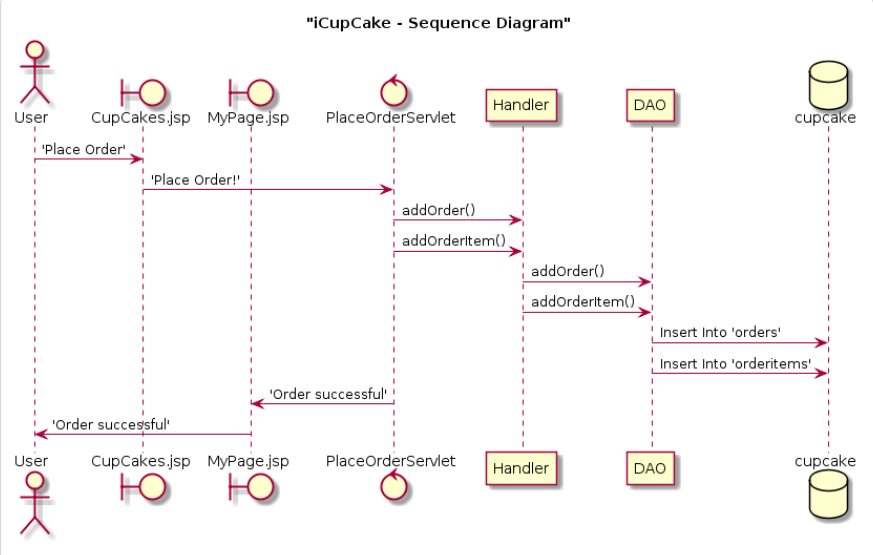


Navigationsdiagrammer

Composite State diagram for kunde logget ind:



Sekvensdiagram for cupcake-køb:



Dette sekvensdiagram beskriver flowet idet en bruger har valgt en cupcake og derefter trykker på ’Place Order’. Handler og DAO returnerer void på metoderne der tilføjer ordrer til databasen, hvorfor PlaceOrderServlet efter disse kald fortsætter med en requestForward til MyPage.jsp. Denne giver så brugeren besked om en succesfuld ordre.

MyPage.jsp kan også nås uden at man netop har placeret en ordre. Der er således en if-sætning i begyndelsen af MyPage.jsp der evaluerer på om der ligger en attribut i requesten med navnet ’ordersuccess’. Hvis ja, udskrives teksten om at ordren var succesfuld, og man har derudover muligheder som vanligt på MyPage.jsp.

Teknologi valg

En kort beskrivelse af hvilke teknologier der er brugt (jdbc, mysql, …). Her er det *ikke* meningen at I skal beskrive disse teknologier, men man skal sige hvilke der er brugt sådan at den der skal overtage projektet ved hvilken software der skal bruges. Der skal versions numre på (Netbeans 8.2, ikke blot “Netbeans”).

Krav

Dette afsnit vender vi tilbage til senere i semesteret. Der er to dele i dette afsnit:

* Hvad er firmaets håb med dette system (hvad er deres vision for systemet eller hvilken værdi er det jeres system skal tilføre deres virksomhed).
* Scrum userstories (product backlog). Dette vender vi tilbage til efter påskeferien.

Domæne model og ER diagram

Det interesante ved denne domæne og database er at den langt hen af vejen er grundlaget for resten at systemet. Tabeller og relationer siger noget om *hvad* systemet arbejder med, ikke *hvordan*. Så det er godt sted at starte.

Som led i beskrivelsen af Domæne eller ER diagram skal man have følgende med:

* Diagram over hele modellen. Det er vigtigt at få plads til alle tabeller og alle relationer. *Det kan så betyde at man ikke kan få plads til alle attributter på de enkelte tabeller. Dem kan man slå op i databasen, så det er ikke så vigtigt*
* Hvis nogle af tabellerne ikke er på 3. normal form vil det være almindeligt at nævne det, og forklare hvorfor det er gjort (tidspres eller anden overvejelse).
* Hvis der anvendes 1-1 relationer kan man beskrive hvorfor man ikke blot har en tabel.
* Hvis nogle tabeller implementerer en mange-mange relation vil det være normalt at beskrive det.
* Hvis der er flere veje at nå fra et sted til et andet, vil det nemt gøre det svært at holde databasen konsistent. Hvis I har gjort det alligevel så skal I forklare hvorfor.
* Hvis der er tabeller, hvor man benytter andet end et automatisk generetet ID som nøgle skal man forklare det.

Det er interessant at beskrive hvilke overvejelser der ligger til grund for de konkrete valg der er i ER modellen (fremmednøgler, constraints, triggers, osv)

Navigationsdiagram

Det som brugeren oplever er en række websider, hvor man kan indtaste oplysninger og navigere videre til andre sider. I større systemer kan det være svært at bevare overblikket over hvilke sider der er, og hvordan man kommer fra den ene til den anden. Navigationsdiagrammet er beregnet på at vise dette på en mere overskuelig måde. Som led i beskrivelsen af navigationsdiagrammet skal følgende med:

* Oversigtsdiagrammet. Hvis det bliver for stort må man dele det op. Men det er vigtigt at der er et overordnet diagram.
* Hvis man har benyttet sig af en “fælles navigations bar” i toppen af alle sider, skal man forklare det.
* Hvis nogle sider kun kan nåes af nogle brugere (dem der har konto, dem der er logget ind, dem der arbejder i butikken,…), så skal det fremgå.
* Navne på jsp sider skal fremgå, og hvilke servlets der bringer en fra den ene side til den næste.

Navigationsdiagrammer laves som UML Tilstandsmaskiner [state charts](http://plantuml.com/state-diagram). Bemærk specielt at det, der hedder “composite state”, er meget velegnet til at indkapsle at man er logget ind.

Sekvens diagrammer

De fleste programmører kan læse de enkelte metoder i et program, mens det kan være svært at skabe sig et overblik over hvordan programmet virker på overordnet plan. Et sekvens diagram bruges til at vise hvordan et typisk forløb foregår, eller til at vise et særligt svært særtilfælde. *Man kan aldrig dokumentere hele programmet med sekvensdiagrammer, man vælger altid nogle interessante eksempler*.

Et eksempel på et typisk forløb kunne være at brugeren præsenteres for indkøbssiden. Her skal der vises følgende:

* Selve diagrammet, startende med jsp-siden eller servlet.
* Brug de rigtige klassenavne og metode navne
* Undlad argumenter til metoderne

I forklaringen til diagrammet skal du særligt lægge vægt at beskrive hvilke grene af if-sætninger der er brugt i de enkelte metoder.

Særlige forhold

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

* Hvilke informationer gemmes i session
* Hvordan håndterer man exceptions. Det kommer vi tilbage til senere i semesteret.
* Hvordan man har valgt at lave validering af brugerinput.
* Hvordan man har valgt at lave sikkerhed i forbindelse med login.
* Hvilke brugertyper, der er valgt i databasen, og hvordan de er brugt i jdbc
* … andre elementer

Husk: det er bedre med 2 linjers dokumentation end ingen.

Status på implementation

Dette afsnit skal liste hvor langt man er nået med implementationen. Typiske ting man kan have sprunget over er:

* Man har ikke nået at lave alle de jsp sider man har med i navigationsdiagrammet.
* Man har ikke nået at lave alle CRUD metoderne til alle tabellerne
* Man har ikke fået stylet sine sider
* Man har fundet en fejl “i sidste øjeblik”, men har ikke haft tid til at rette det. - F.eks. at man har brugt session forkert, sådan at man på en af siderne kan komme ind uden at være logget ind.
* tests der fejler på afleveringstidspunkt
* … andre mangler

Test

Der skal være lavet test. Vi har ikke i dette projekt haft fokus på at teste, så det skal ikke med. Du skal dokumentere tests ved at beskrive i tabel form:

* Hvilke klasser er testet
* Hvilke metoder der er testet
* Dækningsgrad af dine tests for de valgte metoder og klasser