Cupcake prøveeksamen

Deltagere:

Christoffer Ikizek Wegner, cph-cw109@cphbusiness.dk , dofinator

Sebastian Godsk Hansen, cph-sh497@cphbusiness.dk, sebbergo

Phillip Thomas Isenborg Andersen, cph-pa124@cphbusiness.dk , fiske-halsen

Lukas Bang Stoltz-Andersen, cph-ls369@cphbusiness.dk , LukasBangStoltz

Sumit Dey, cph-email?, github navn?

Klasse: D

Dato: 29/3/2019

Indledning

Vi har fået til opgave at lave et system til en webshop, der sælger cupcakes. Vi har gennem vores teori om backend, frontend, databaser og servere, udviklet en funktionel cupcake webshop. Vi har gjort det muligt for en virksomhed, at sælger cupcakes over en website, i stedet for en fysisk butik. Måden vi har gjort det på, vil blive beskrevet uddybende i nedenstående afsnit.

Baggrund

Vi har landet et programmerings job med udgangspunkt i en funktionel webshop fra ”Oslker Cupcakes”, et iværksætter start-up med fokus på økologiske cupcakes, logeret på Bornholm.

Efter indsamling af krav fra bageriet, er de funktionelle krav blevet fastsat.

Følgende krav:

Kundekrav:

- Som kunde kan jeg logge på websiden med e-mail og kodeord. Når kunden er logget ind skal kundes e-mail følge kunden på hver side.
- Som kunde skal man kunne oprette en konto for at kunne betale og gemme ordrer
- Som kunde skal man kunne bestille og betale cupcakes ud fra følgende valgte kriterier. Valg af bund, valg a topping og valg af antal stk.
- Som kunde kan man se sine valgte ordrelinjer i en indkøbskurv og dermed se samlede pris.
- Som kunde kan man fjerne ordrer fra indkøbskurven.

Administrator Krav:

- Som administrator kan man se alle kunder i databasen og dertil deres tilhørende ordrer.
- Som administrator kan man opdatere/slette kunders informationer gennem MySQL, dertil tilføje valuta til kunders konto.
- Som administrator kan man fjerne ordre, så der ikke vil opstå tilfælde med ugyldige ordrer hvor kunder ikke har betalt.

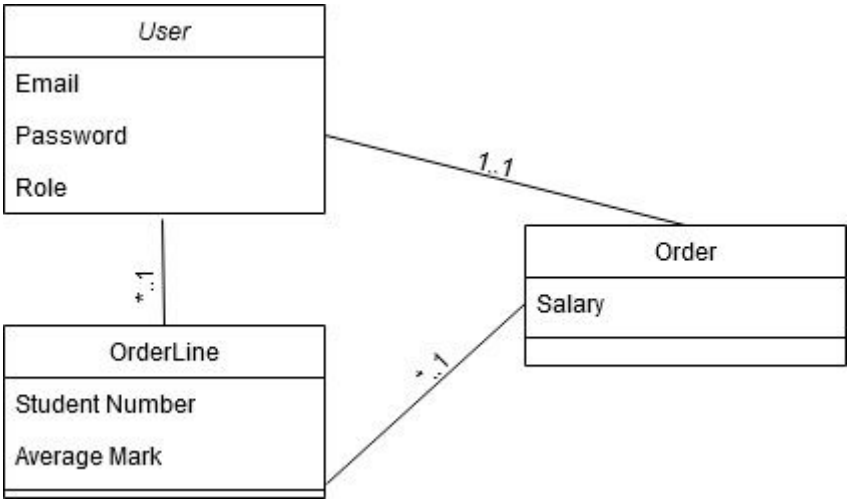
Valg af teknologi:

- MySQL Workbench 8.0 CE
- IntelliJ IDEA 2019.3.3 Ultimate
- Apache Tomcat 8.5 Tomcat 8
- Digital Ocean linux server

Vision:

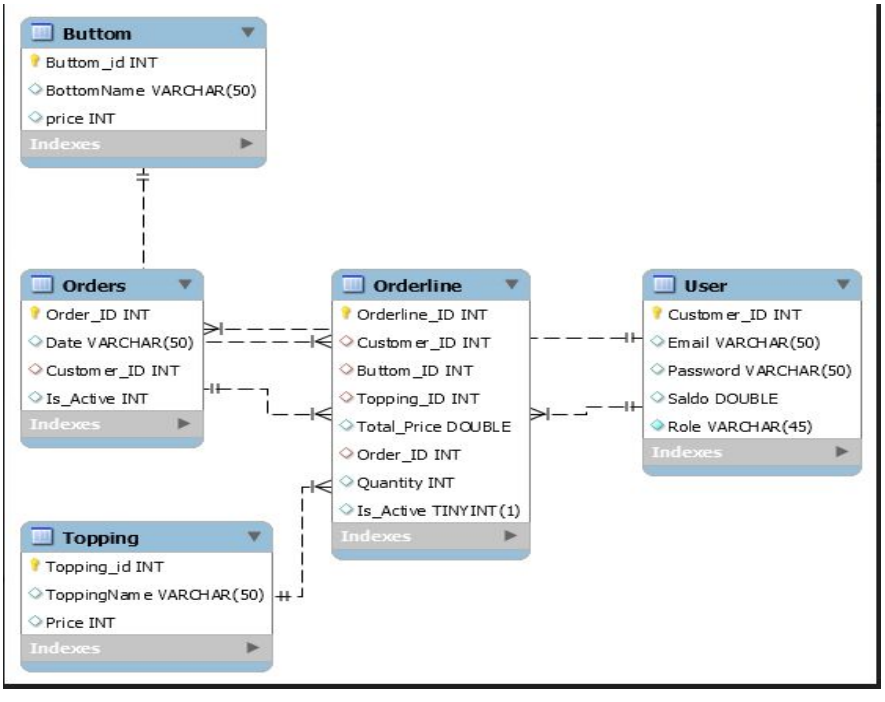
Firmaets håb med systemet, er at skabe et nemmere miljø for både forbruger og dem selv. Igennem hjemmesiden, vil bestillinger af cupcakes være mere struktureret for firmaet, og de vil derfor kunne producere dem mere effektivt. Samtidigt har kunden et større overblik over firmaets udvalg af cupcakes, og kan lettere rette/oprette ordrer. Dertil vil systemet hjælpe firmaet med at ekspandere, og få større kundebase og indkomst.

Domæne Model:



Vi har udviklet en domænemodel ud fra vores java klasser. Hvis man kigger på modellen, kan man se at vi bla. har en user klasse. En user har en email, et password og en role. Derudover har en user mange ordrelinjer, som tilhører den samme ordre. En overordnet ordre tilhører altså én user. Vi har udviklet vores domæne model i forhold til udviklingen af vores databases tabeller, som vil blive beskrevet i nedenstående afsnit.

EER diagram:



VI har designet vores tabeller i databasen, sådan at vi i alt har 5 tables, herunder Buttom, Orders, Orderline, User og Topping.

Vores buttom table indeholder kolonnerne Buttom_ID, som fungerer som primary key, ButtomName og til sidst en pris.

Hvis vi går videre til Topping tablet, kan vi se at den har kolonnerne Topping_Id, ToppingName og en pris. Topping_Id fungerer som primary key i dette table.

Vi bruger altså en primary key i de to ovennævnte tables, til at gøre hver enkelt buttom eller topping unik. Vi har endvidere designet Buttom og Topping tablet, med udgangspunkt i det "menukort" vi har fået udleveret som materiale.

Hvis vi går videre til user tablet, kan vi se at det indeholder kolonnerne Customer_Id, Email, Password, Saldo og Role. Customer_id fungerer her som primary key. Derudover bruger vi kolonnen "role" til at adskille en admin og en normal kunde, når der bliver logget ind på webshoppen.

Når en user laver en ordre, skal ordren først igennem et mellemled. Det table har vi kaldt Orderline. Dette table har vi gjort brug af, så at man i en ordre kan have mange ordre. Hvis vi kort kigger på det, kan vi se at det indeholder kolonnerne Orderline_Id, Customer_id, Buttom_Id, Topping_Id, Total_Price, Order_ID, Quantity og Is_Active. Vores primary key her er Orderline_Id.

Customer_Id er en foreign key til parent table User. Det samme gælder Buttom_Id, Topping_Id og Order_Id. Disse er dog foreign keys til tabellerne Buttom, Topping og Order_id.

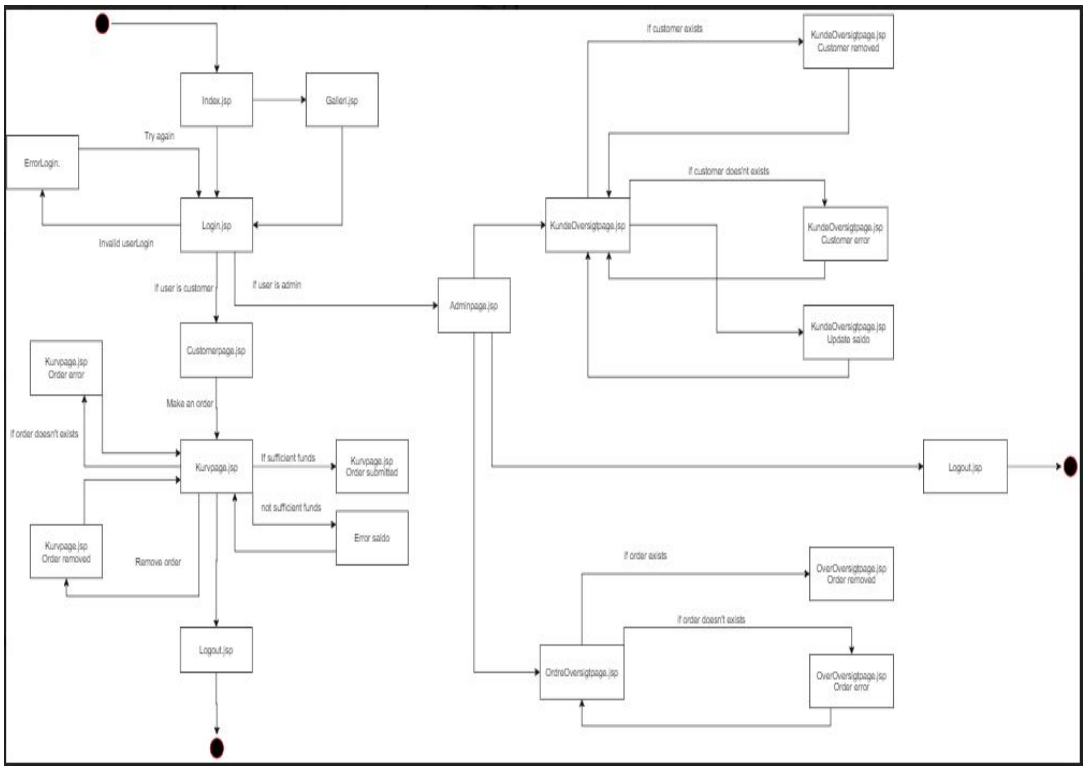
Vi har gjort brug af foreign keys, for at sikre at disse foreign keys, skal eksistere i parent tabellerne. Vi har altså her gjort brug af mange til mange relation, da kunder kan have samtlige ordrelinjer.

Hvis vi tager et videre kig på vores table Orders, så har vi kolonnerne Order_Id, Date, Customer_Id og Is_active. Vores primary key her er Order_Id. Foreign key er Customer_Id til tabellen User. Vi har her også gjort brug af en mange til mange relation, da vi har forbundet ordre til flere ordrelinjer. En Ordre kan altså have mange ordrelinjer.

Man kan kort sige, at vores Orderline table fungerer som et "junction table", da den fører tilbage til mange sider, altså både User tabellen og Orders vha. foreign keys.

Endvidere bider man også mærke i, at vi har gjort brug af en boolean "Is_Active" i både Orders og Orderline tabellerne. Dette har vi først og fremmest gjort i Orderline, så de er default aktive, det vil sige, at når en kunde tilføjer noget til kurven, er de aktive. Så snart de køber denne ordre, så vil de blive inaktive. Det samme gælder i tabellen Orders.

Navigationsdiagram

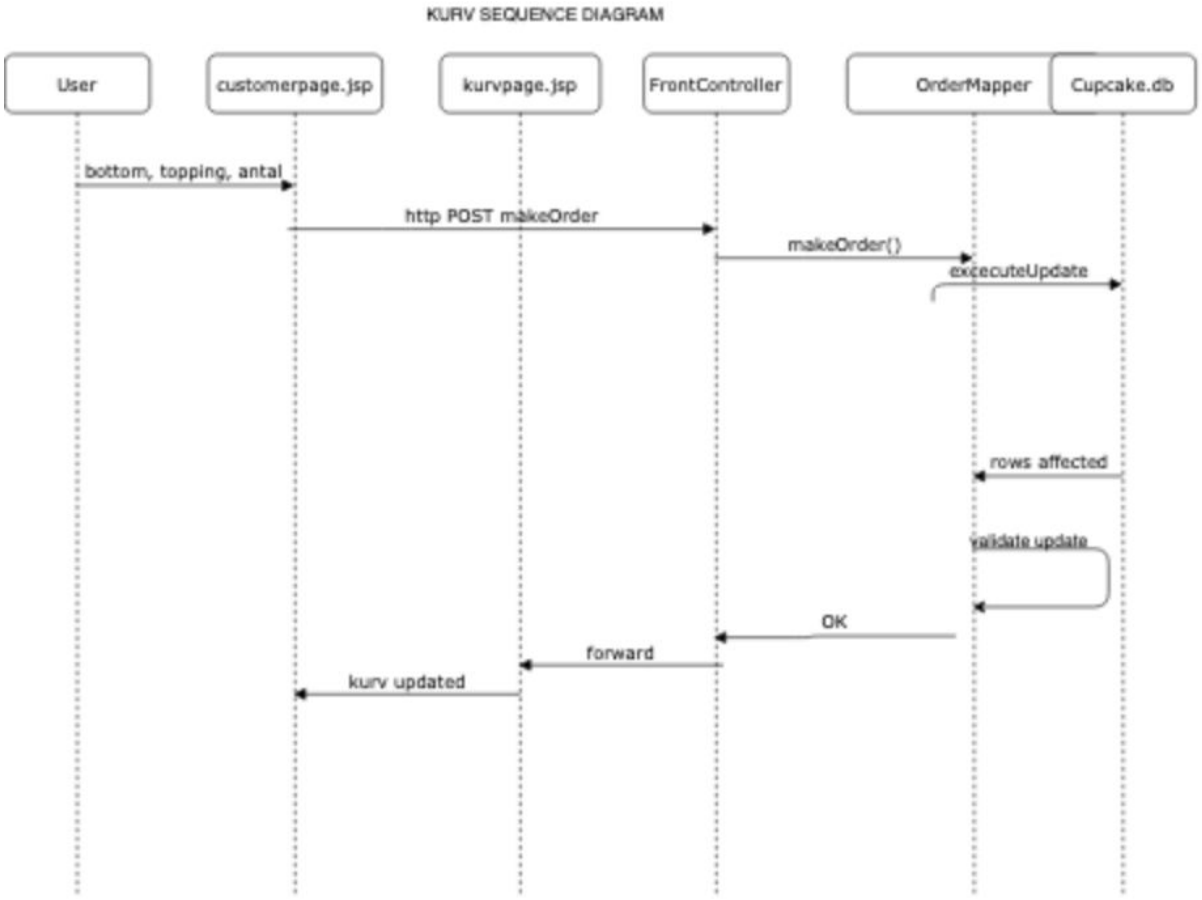


I vores navigationsdiagram er vores første user scenarie, at kunden bliver promptet for brugernavn og kodeord. De oplysninger bliver sendt til serveren som bekræfter hvorvidt brugeren eksisterer i systemet eller ej. Hvis kunden ikke eksisterer bliver man sendt tilbage til loginsiden med en errorbesked, hvor man kan prøve igen. Er loginoplysningerne korrekte bliver kunden videresendt til customerpage, hvor man har mulighed for at tilføje en cupcakeordre til kurven samt fjerne en ordre, hvis dette skulle være tilfældet. Kurven, bliver opdateret i serven både når en ordre tilføjes og når en ordre tilføjes. Ved begge muligheder bliver man sendt tilbage til customerpage hvor man kan gentage processen. Hvis man til sidst vil gennemføre sin ordre, så vil serveren gå ind og tjekke kundens saldo og verificere om hvorvidt købet kan gennemføres. Kan købet ikke gennemføres pga. for lav saldo bliver man sendt tilbage til kurven, hvor man bliver nødt til at fjerne en eller flere ordrelinjer. Hvis købet kan gennemføres bliver ordren gemt i serveren og man kan til sidst logge ud.

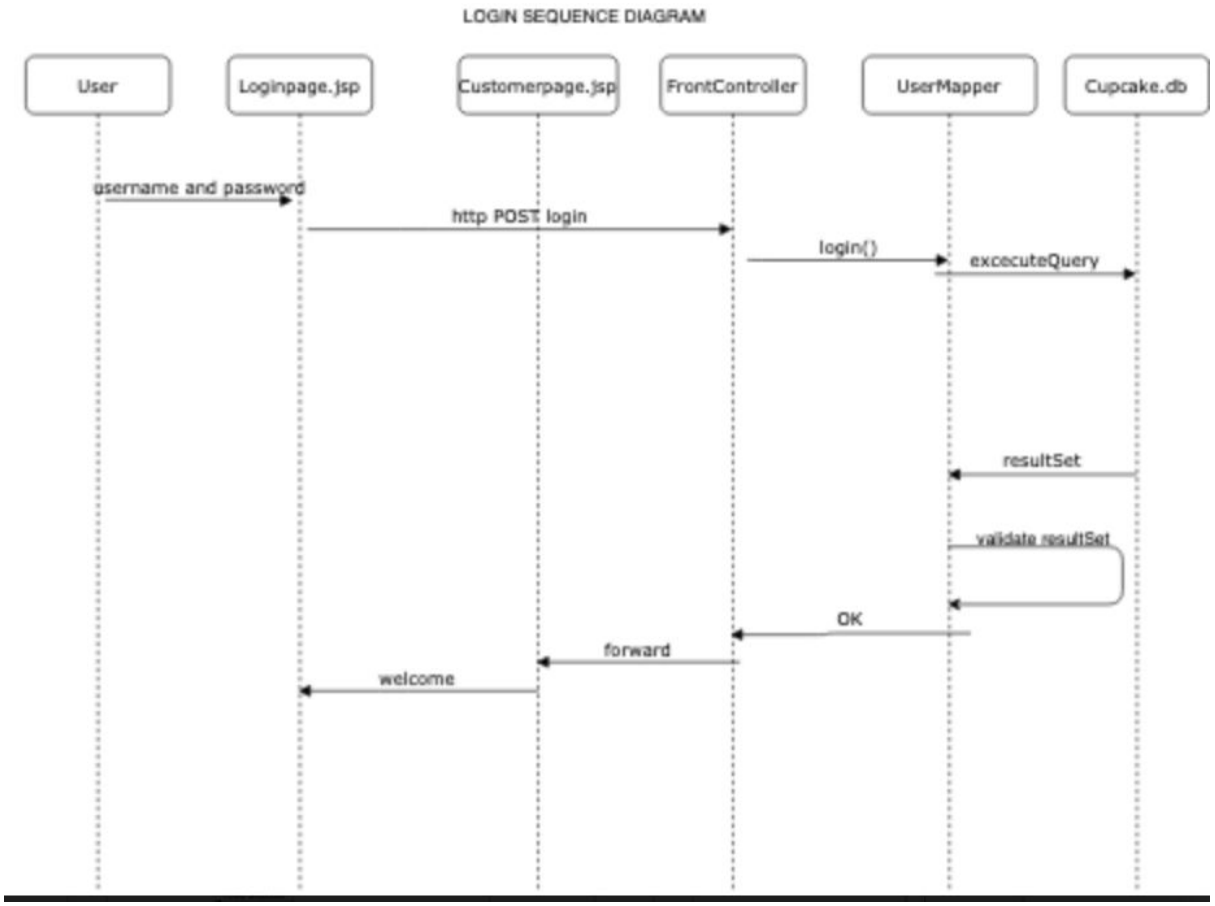
Hos admin bliver man promptet for email og kodeord. Disse oplysninger bliver sendt til serveren for at bekræfte om adminen findes, hvis ikke bliver man sendt tilbage til loginsiden

med en errorbesked om at prøve igen. Hvis oplysningerne er korrekte bliver man sendt videre til adminpage, hvor man har mulighed for at se alle ordre, se kunderne og opdatere kunders saldoer. Skulle admin have brug for at fjerne en ordre, kan vedkommende vælge ordreoversigtpage, hvor man kan fjerne ordre i systemet. Serveren går ind og ser om ordren findes og opdaterer databasen, hvis ordren findes og man bliver dernæst sendt tilbage til adminpage. Omvendt hvis ordren ikke findes bliver man også sendt tilbage til adminpage med en errorbesked. Det samme gælder også for at opdatere en kundens saldo. Til sidst har admin mulighed for at logge ud, for at slutte sin sessionen.

Sekvensdiagram

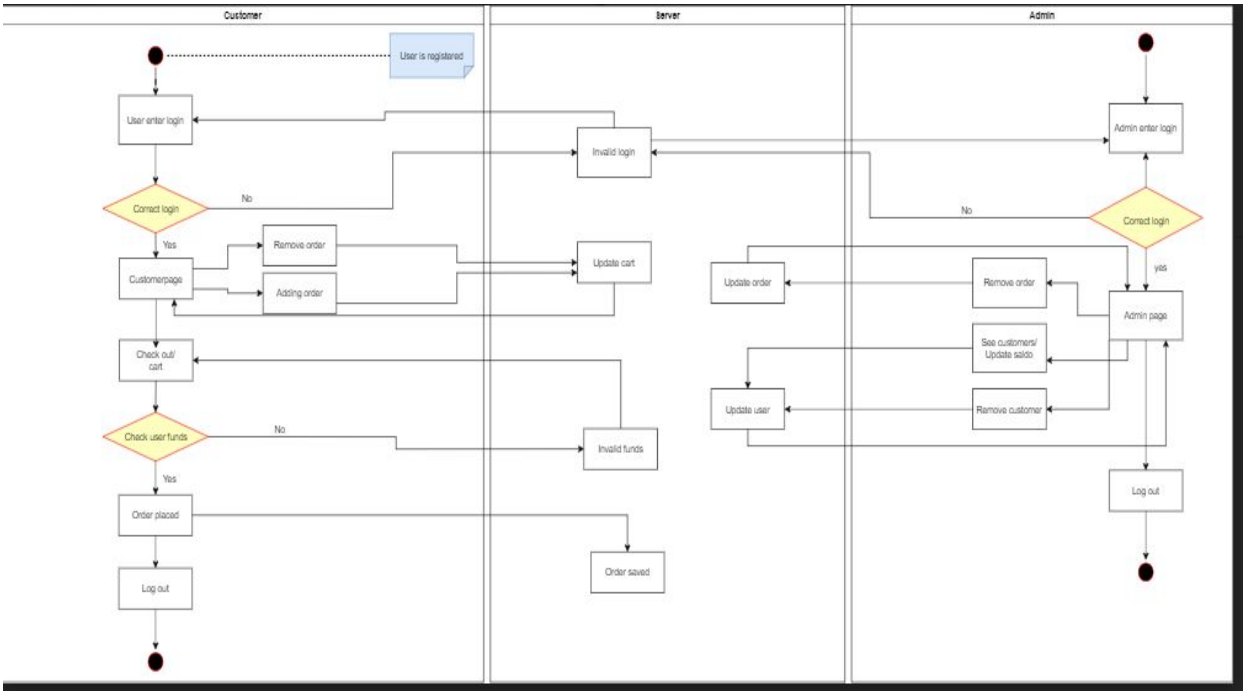


I ovenstående diagram får vi et overblik over hvordan systemet tilføjer en ordre til kundens kurv. Kunden står på customerpage.jsp siden, hvor der er 3 forskellige kategorier med toppings, bottoms og antal. Den hidden value vi har givet med, som er makeOrderLine, går indover FrontControlleren og ind i klassen, hvor den tager de 3 variabler ud med request og tilføjer en ordre med metoden MakeOrder til databasen. Den returnerer derefter kurven med den tilføjede ordre. Hvis ikke den får alle 3 variabler med, altså at kunden har glemt at vælge en kategori, så sender den kunden tilbage til customerpage.jsp med en error besked.



Login sekvensdiagrammet tager username og password indtastet fra useren. Herefter går den ind over FrontControlleren og ind i Login. Her får den fat i de indtastede variabler fra useren og checker i databasen om disse variabler tilhører en eksisterende kunde/administrator. Hvis dette ikke er tilfældet logger den ind henholdsvis som rollen useren har eller sender dig tilbage til loginsiden igen.

Aktivitetsdiagram



Kunde

Vi har lavet aktivitetsdiagram med udgangspunkt i tre swimlanes, henholdsvis Customer, Server og Admin. Hvis vores customer er registreret og forsøger at logge ind, kan der ske to ting - Han kan skrive sit login forkert, eller rigtigt. Serveren checker inputtet, og validere loginnet. Hvis loginnet ikke stemmer overens, bliver vores customer dirigeret tilbage til loginsiden. Hvis kunden logger succesfuldt ind, har han mulighed for at fjerne, eller tilføje cupcakes (ordrer) til sin kurv. Når kunden tilføjer ordrer, opdatere serveren kurven. Kunden kan herefter vælge at checke ud, og betale for sin kurv. Her bliver kundes konto checket - hvis der ikke er nok penge, dirigeres han tilbage på Customerpage, ellers bliver ordren oprettet, og gemt i serveren. For at slutte sin session kan kunden logge ud.

Admin

Ligesom kunden, kan adminen logge ind, hvorefter hans login bliver checket af serveren. Hvis loginnet er forkert, bliver han sendt tilbage til loginsiden, ellers bliver han dirigeret til Admin page. Her kan han fjerne en ordrer fra systemet, som serveren efterfølgende opdatere og fjerner. Derudover kan han se en liste over kunder, og opdatere deres saldo, eller slette dem fra systemet. Det går ligeledes ind over serveren. For at slutte sin session kan Admin logge ud

Særlige forhold

- Vi bruger session til at gemme variabler som vi skal bruge flere gange igennem systemet, som f.eks. userens email. Vi bruger den i navigationsbaren på alle sider, så man kan se hvilken email det er blevet brugt til at logge ind med.

- Vi bruger error beskeder til at håndtere exceptions. Så i stedet for at programmet crasher eller sender en besked, så sender vi useren hen på samme side som de var

på førhen sammen med en error besked, for at informere useren om hvad de gjorde forkert. Det her hænger også sammen med vores valg af vurdering af brugerens input. Så vi kan håndtere forkert indtastet input.

- Sikring af sikkerhed ved vores login bliver gjort gennem vores database. Det vil sige at den metode der sikrer at useren eksisterer checker det gennem vores database, for at se om userens input af email go password er et eksisterende match inde i databasen. Ellers sender den useren tilbage til login siden så useren kan registrere sig. I forhold til at lave flere administratorer har vi sikret det sådan, at man udelukkende kan lave flere ved at have adgang direkte til databasen, så man kun kan adde det der.

Status på implementation

- Hjemmesidens overordnede design kunne forbedres en del, men vi har haft funktionalitet i fokus.