Cupcakes

Gruppenavn: MIC

https://github.com/CK2800/cupcakes

Udarbejdet af:
Claus, Henrik og Jesper
Efteråret 2018

Navn:	Cph-email:	Github-navn:
Claus Kramath	cph-ck83@cphbusiness.dk	@CK2800
Henrik Michael Agger	cph-ha104@cphbusiness.dk	@HenrikAgger
Jesper Petersen	cph-jp284@cphbusiness.dk	@JesperPetersenDK

Klasse:
118dat2ae

Indhold

Indledning

Projektet omhandler salg af cupcakes hvor en kunde kan bestille produkter via. en hjemmeside kodet i Netbeans.
Rapportens formål er at gøre tankerne bag udviklingen let tilgængelig for udviklere, der senere måtte tilgå projektet.

Baggrund

Virksomheden som skal bruge systemet, ønsker at sælge cupcakes, som kunder kan bestille online.

Virksomheden ønsker et bestillingssystem, hvor kunden kan vælge mellem forskellige kombinationer af cupcakes bestående af top og bund, logge ind samt betale for produkterne.

Teknologivalg

Til projektet er valgt følgende programmer:

- Netbeans 8.2
- JDBC (bundle ver. 8.0.12)
- Workbench 8.0 CE
- MySQL RDBMS 14.14 (dist. 5.7.23)
- Apache Tomcat 8.0.27
- Bootstrap 3
- Maven 3.1

Løsningens MySQL RDBMS er hostet på en linux maskine på Digital Ocean, www.digitalocean.com. Maskinen kører Ubuntu 16.04.5 baseret på Linux 4.4.0-134-generic GNU/Linux kernel.

Krav

Der skal være tre sider.

1. En login side.
2. En bestillingsside.
3. En betalingsside.

Der skal være en tilhørende login-side hvor kunden kan oprette username og password. Når brugernavn og password er oprettet, skal det være muligt at logge ind. Der skal være et link til produktsiden.

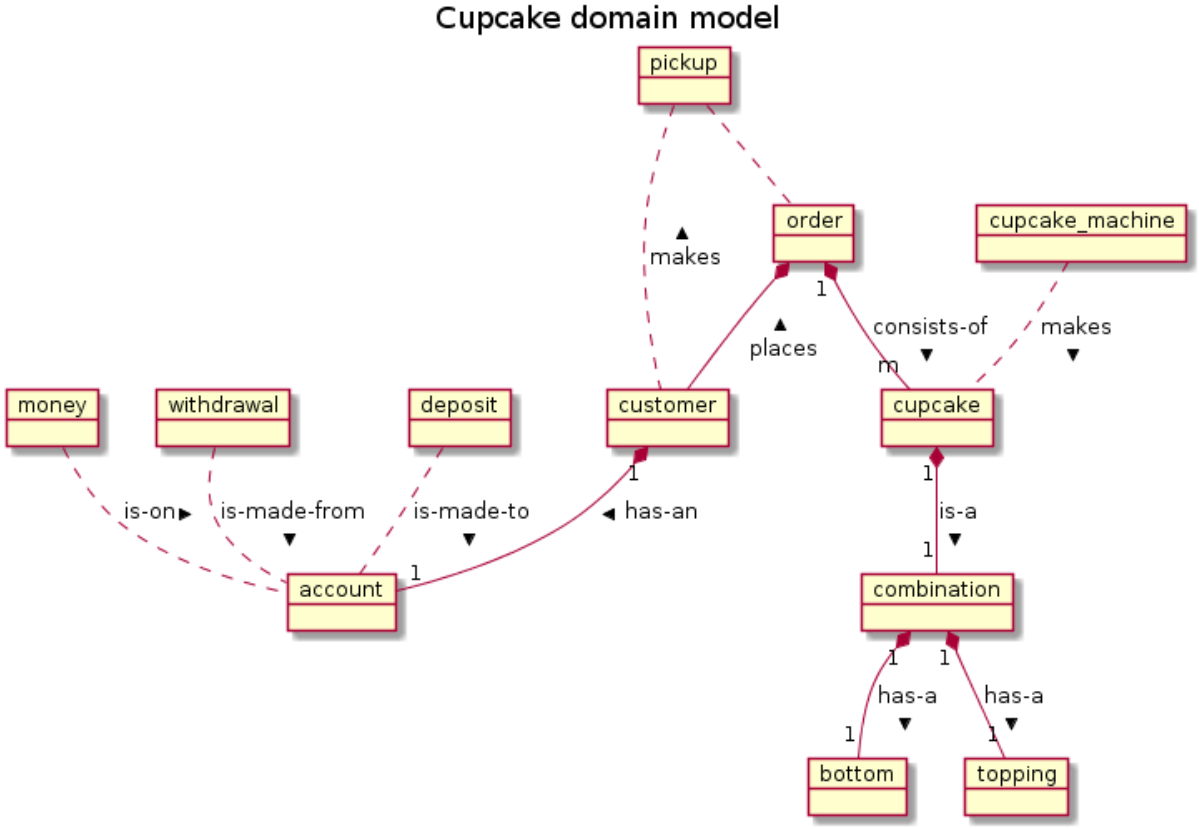
Kunden skal via. en bestillingsside kunne bestille cupcakes, hvor kunden kan vælge mellem forskellige kombinationer af top og bund, samt mængde af de forskellige cupcakes. Siden skal vise den samlede pris, på de valgte varer. Der skal være et link til betalingssiden efter produkterne er blevet valgt.

På betalingssiden skal der være mulighed for at betale med Credit card, 30 dages kredit og betaling via. PayPal/Stripe eller andre betalingsløsninger. Kreditkortet skal forespørge kunden om kundens navn, kortnummer og udløbsdato samt CVC mv.

Ikke funktionelle krav kan være juridiske forhold omkring købet med kreditkortet, samt øvrige indkøbsforhold som programmet skal kunne håndtere.

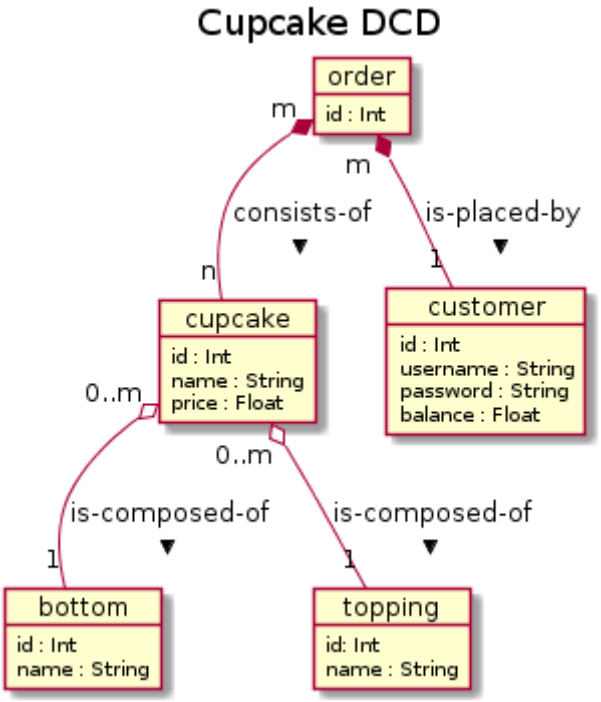
Domænemodel

For at løse et problem i domænet, må man forsøge at forstå dette. I opgaven er forretningsgangen for cupcake shoppen beskrevet, og denne beskrivelse danner grundlag for en lingvistisk analyse, som bl.a. bruges til at opdage ting/entiteter i domænet samt hvordan disse entiteter arbejder/bruges.
Efter denne analyse er følgende domænemodel etableret:



I domænemodellen findes flere 1-1 relationer mellem de fundne entiteter, f.eks. ml. customer->account og cupcake->combination->bottom/topping. Ofte kan den ene entitet i relationen omdannes til en attribut i den anden entitet og nogle af disse 1-1 relationer kan derfor fjernes. Eksempelvis har vi på klassediagrammet helt fjernet account-entiteten og i stedet tilføjet en attribut på customer som hedder 'balance'.
Under arbejdet med at omdanne domænemodel til (design)klassediagram, er der klasser, for hvilke vi ikke ønsker at gemme data, som ikke har fundet vej til klassediagrammet, f.eks. cupcake_machine og combination.

Klassediagram



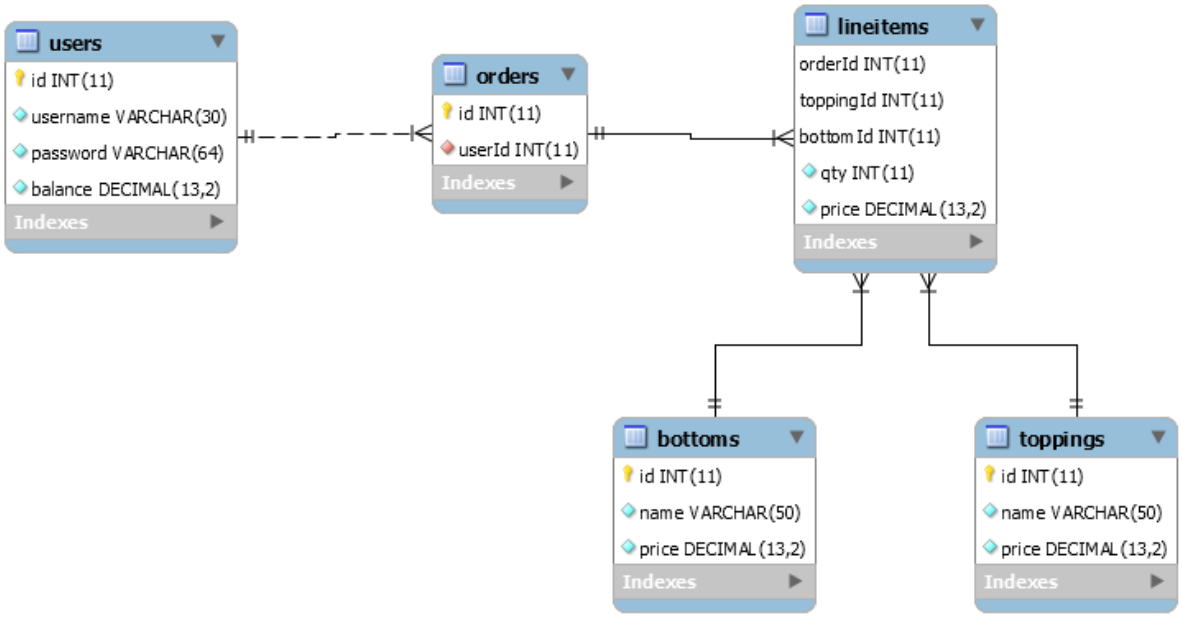
Database

Databasen er afledt af (design)klassediagrammet, da det illustrerer de klasser der gemmes data for. Den er således opbygget af 5 tabeller som er normaliseret til 3. normalform idet:

1. værdierne i kolonnerne er atomare, har samme domæne, kolonnenavnene er unikke i hver tabel og rækkefølgen af kolonnerne er ligegyldig.
2. Ingen kolonner er afhængige af en delprimærnøgle.
3. Ingen kolonner er afhængige af en kolonne udenfor primærnøglen.

E/R-diagram

Herunder ses det udarbejdede E/R diagram for vores cupcake database. Et E/R diagram er et logisk/strukturelt diagram, der beskriver tabellerne i den tilhørende database, samt relationerne mellem de enkelte tabeller, primærnøgler og datatyper.



Tabeller

users tabellen beskriver en bruger, dennes kodeord og kontostand.
Det er besluttet at lade attributten 'username' være unik. Grunden er, at en bruger ellers ville kunne logge ind ved at taste forkert kodeord, som tilfældigvis er kodeordet til en anden konto med samme brugernavn.
Attributten 'password' har datatypen VARCHAR(64) aht. senere implementering af hashing af kodeord med SHA256 som konverteres til hexadecimal streng.

orders tabellen knytter ordrer og brugere sammen.

lineitems tabellen er en lookup tabel som implementerer mange-til-mange relationerne mellem orders og bottoms- samt orders- og toppings tabellerne, idet en ordre kan have mange bunde og mange toppe. En bund eller en top kan findes på mange ordrer. Se også klassediagrammet.
Attributten 'price' er medtaget, således at en vares pris på et givent tidspunkt lagres. På denne måde sikres at senere visning af en ordre eller faktura fortsat har korrekt pris, trods evt. efterfølgende prisændringer. Dette har også regnskabsmæssig betydning. Bemærk at 'price' er stk-prisen for den givne cupcake.
Primærnøglen er sammensat af fremmednøglerne orders.id og products.id.

bottoms- og toppings tabellerne beskriver hver enkel cupcake del med navn og pris.

Implementerede begrænsninger

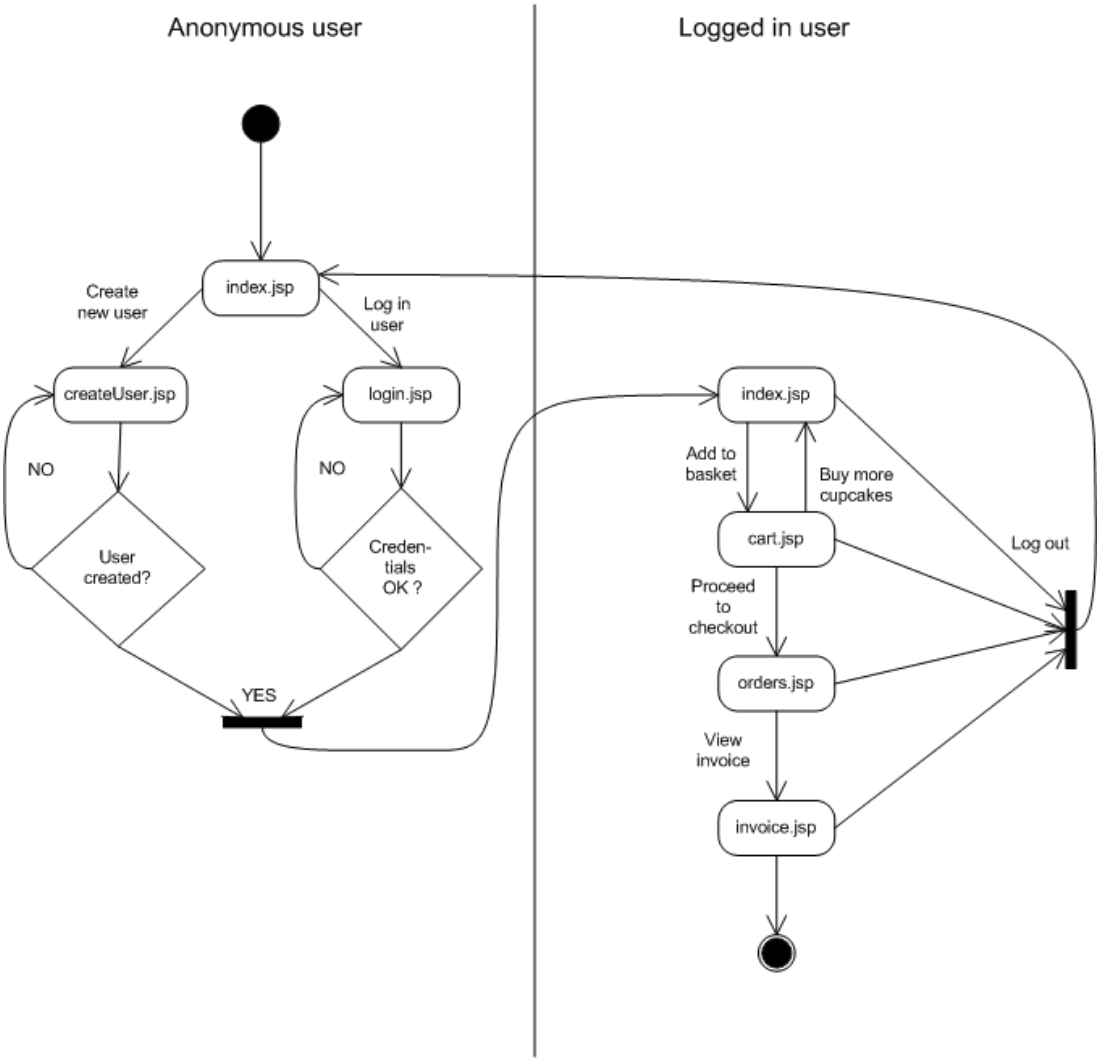
- Fremmednøglen *fk_orders_users* i tabellen orders tillader ikke at users slettes hvis de refereres af orders, fordi en order ikke skal kunne eksistere uden en user. En ordres detaljer skal til enhver tid kunne udskrives af regnskabsmæssige hensyn, herunder også information vedr. køber (user).
- Fremmednøglen *fk_products_producttypes* i products tabellen tillader ikke at producttypes slettes hvis de er refereret i products, fordi koden skal kunne foretage check af om et lige antal bunde og toppe er bestilt. Ligeledes vises disse informationer også på fakturaen.
- Fremmednøglen *fk_lineitems_orders* i lineitems tabellen tillader at lineitem slettes hvis order slettes. Hvis en ordre annulleres, er der ingen grund til at gemme information om ordrens varelinier.
- Fremmednøglerne *fk_lineitems_toppings* og *fk_lineitems_toppings* i lineitems tabellen tillader ikke at bottoms eller toppings slettes hvis det er refereret i lineitems. Dette ville medføre ufuldstændig information for en ordre, idet data om varen på varelinjen ville være tabt.

Navigationsdiagram

Herunder ses hvorledes navigationen i applikationen foregår.
Alle sider har en topmenu.

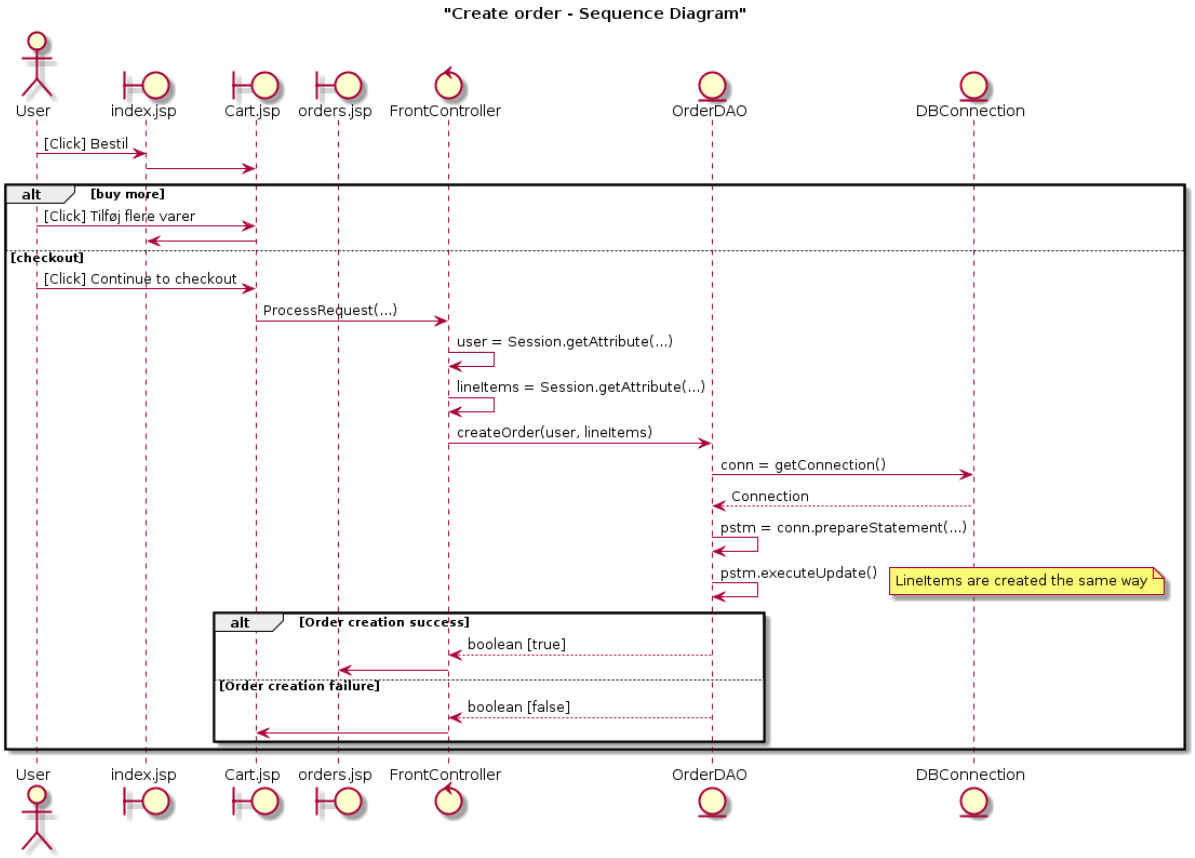
Toppen (kurv):
Link over til checkout område som gør det muligt, at kunne betal for varerne.
Forside, Log in og checkout:
Indeholder link til forside samt til create user og log in.
Create user:
Der er mulighed for, at kun gå over til forside samt log in siden i menu'en
Bemærk: At index1.jsp er det samme som index.jsp. Det vil sige at det er index.jsp.

Der indholder 2 stk af samme filnavn: cart.jsp.
viewtop/cart.jsp: Det er toppen af siden som indeholder hvad kurven har af vare. Det er som udgangspunkt at vise "hvor" mange vare der er tilføjet til kurven.
cart.jsp: Det er sidens indhold som viser hvor mange vare man har tilføjet til kurven.



Sekvensdiagram

Nedenfor vises et sekvensdiagram for købsprocessen, fra valg af cupcake til oprettelse af ordre og dermed afslutning af selve købet.

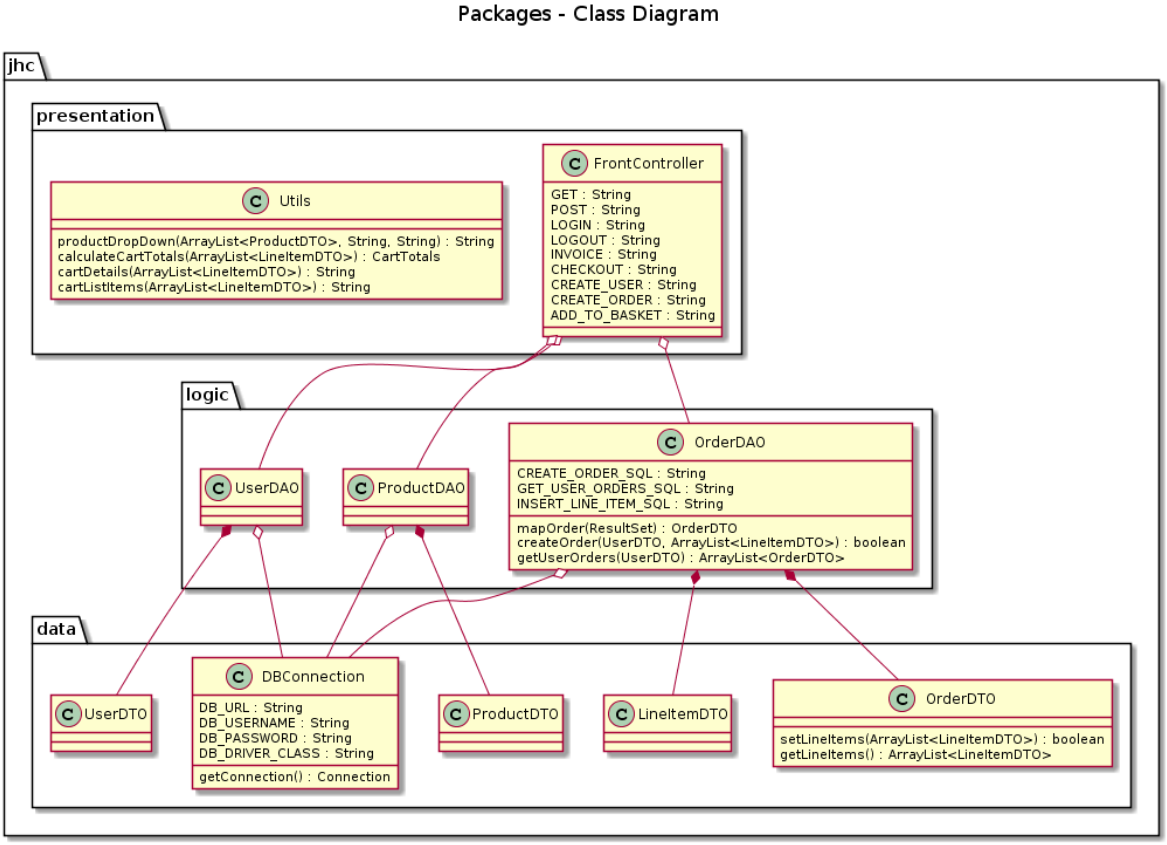


I overgangen fra brugergrænseflade til backend, i kaldet fra Cart.jsp til FrontController.ProcessRequest(...), switches på værdien af HTTP requestets origin. I dette tilfælde skal origin være FrontController.CREATE_ORDER.
Før FrontController'en kan kalde OrderDAO.createOrder(...) tjekker koden om der findes en user, om der findes en liste af line items og om listen har indhold. Alle tre betingelser skal være opfyldt. OrderDAO.createOrder returnerer true hvis order OG lineitems oprettes korrekt i databasen, ellers returneres false. Afhængig af denne returværdi, leder FrontControlleren brugeren til orders.jsp (ved korrekt oprettelse) eller tilbage til Cart.jsp (ved fejlet oprettelse).

Arkitektur

Applikationen er implementeret i en 3-lags arkitektur som består af et data-, logic- og presentation-lag. Lagene har ansvaret for hver sin del af applikationen.
I logic-laget findes DAO-klasser som danner bindeled mellem presentation- og data-lagene. Når klasser i presentation-laget skal bruge data, sker det via kald på DAO-klasser som returnerer objekter af DTO-klasser.

Nedenstående klassediagram viser arkitekturen og projektets packages.



FrontControlleren har en række strengkonstanter til hjælp når en html-formular eller et link etableres, som medfører at et request skal igennem FrontControlleren. På den måde kan evt. stavefejl mv. undgås og kodning af FrontControllerens switch eller html-formularens action forenkles.

Utils klassen i presentation-laget etableredes bl.a. for at samle dannelsen af html-elementer baseret på objekter af diverse klasser. Således undgås alt for meget kode i view (*.jsp servlets), med enklere kildekode til følge.

Af diagrammet fremgår det, at der ikke er en korresponderende DAO- for hver DTO-klasse. Det blev besluttet at lade DAO-klasser arbejde på flere DTO-klasser hvor det konceptuelt giver mening. Således kan OrderDAO arbejde både på OrderDTO og LineItemsDTO fordi en ordre, udover sine egne attributter, konceptuelt også består af en række varelinjer.
I diagrammet er OrderDTO udstyret med metoder til at sætte og hente ordrens LineItemDTO samling.

Det blev således fravalgt at have en enkel datamapper-klasse med ansvar for forespørgsler på alle DTO-klasserne. Fravalget begrundedes med at dens mange forskellige metoder hurtigt ville føre til en overfyldt (videns-)klasse med uklart defineret ansvar, især i takt med et stigende antal DTO-klasser, hvis systemet senere udvides.

DAO-klasser får forbindelse til databasen vha. DBConnection klassen. Ved hjælp af konstanter i denne klasse, kan tilgang til en anden database nemt konfigureres, uden at skulle ændre andre steder i koden.

Særlige forhold

I det følgende er særlige forhold beskrevet, der bør iagttages under videreudvikling af applikationen:

- Session: Indeholder information om indlogget bruger (UserDTO) og hvilke varer man har tilføjet til varekurven (ArrayList<LineItemDTO>).
- Brugerinput: Valideres ved at trimme leading og trailing whitespace. Sql injection håndteres bl.a. i UserDAO.getUser(...) ved at lade et objekt af PreparedStatement klassen indsætte modtagne argumenter i sql-sætningen. Det er besluttet at validere input mod et regulært udtryk, for at sikre indhold af store og små bogstaver, særlige tegn mv.
- Brugertyper: På vores “Frankfurt”-server oprettedes en alm. bruger vha. MySql. Brugeren er oprettet med flg. rettigheder:
  GRANT USAGE ON *.* TO 'cupcakeUser'@'%'
  samt
  GRANT ALL PRIVILEGES ON `cupcakes`.* TO 'cupcakeUser'@'%'

Implementeret status

Flg. sider / funktionalitet mangler:

- Validering af brugerinput mod et regulært udtryk.
- Kunne fremvise ordrestatus, f.eks. igang, afhentet o.lign.
- Håndtering af udsolgte/udgåede varer i cupcake shoppen.
- Adskillelse af users i alm. brugere og administratorer.
- Administrative funktioner som f.eks. anullere ordre, ændre stamdata for bruger, varer mv.
- Indbetale penge på kontoen.
- Debitere kontoen ved køb.

Flg. sider / funktionalitet er implementeret:

- Brugers varekurv vises i topmenuen på baggrund af sessionsvariabler.
- Funktionalitet til at logge ind og ud.
- Ordre kan oprettes.
- Visning af kurv på cart.jsp med detaljeret indhold om varer.
- Liste af ordrer for en indlogget kunde.
- Faktura for en ordre.