**Navn & Email:**

Mads Voss: [Cph-mv138@cphbusiness.dk](mailto:Cph-mv138@cphbusiness.dk)

Elias Z. Laghouila: [Cph-el63@cphbusiness.dk](mailto:Cph-el63@cphbusiness.dk)

Jonatan Magnus Bakke: [Cph-jb281@cphbusiness.dk](mailto:Cph-jb281@cphbusiness.dk)

Mathias Igel: [Cph-mi73@cphbusiness.dk](mailto:Cph-mi73@cphbusiness.dk)

**Indledning:**

Cupcake Project er en sammensætning af en database og et NetBeans projekt som samarbejder for, at lave en Web Application hvorpå det er muligt at købe cupcakes. Databasen indeholder forskellige tabeller hvor der bliver trukket data fra og lagt ind, som bliver brugt i NetBeans projektet. Systemet fungerer således at dataen bliver behandlet på følgende måde:

Database 🡪 DataMapper 🡪 Servlets 🡪 .jsp filer

**Baggrund:**

Enhver virksomhed som ønsker at sælge cupcakes ville kunne anvende denne Web App, formålet med Web App’en strækker dog ikke længere end at sælge cupcakes.

Krav stillet til produktet:

* Side hvor man kan oprette en bruger
* Loginside
* Lave en cupcake og bestille den
* Kurv som kan indeholde mere end en cupcake
* Layout
* Faktura
* Admin side

**Teknologiske valg:**

NetBeans IDE 8.2

MySQL Workbench 6.3.9

JDBC MySQL

Bootstrap

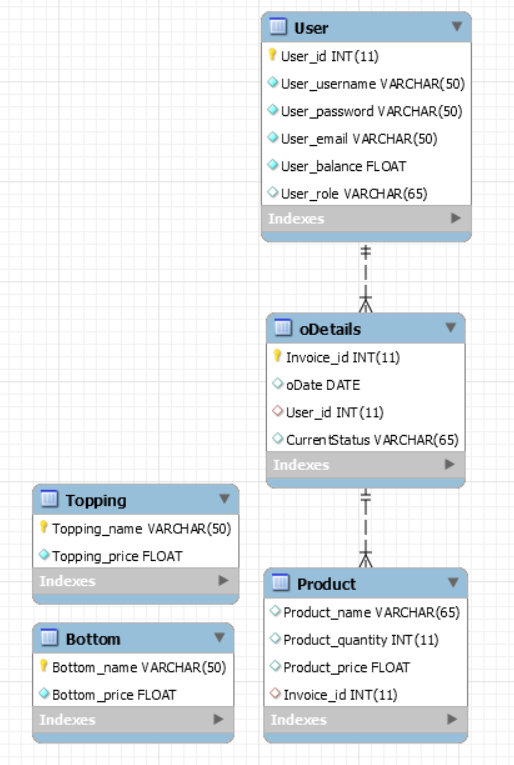
Grunden til vi har valgt at bruge NetBeans er fordi det er en af de mest populære open source IDE’s på markedet. Det vil sige at der er masser hjælp at hente hvis man skulle opleve problemer med ens IDE.

**Domæne model og ER Diagram:**

Vi har valgt at vores Primary Keys ikke må være lig ’null’, udover det bruger de alle sammen automatisk genererede ID’s, hvilket vil sige at de ikke kan blive ens.

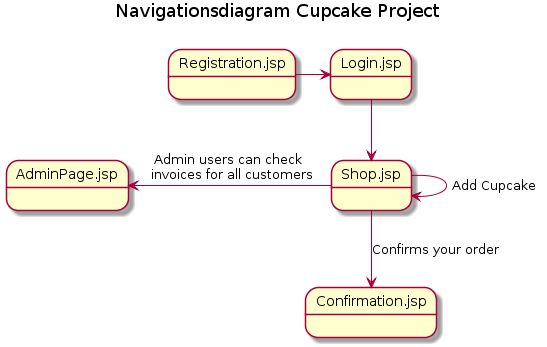
Vi har valgt at give en Foreign Key til Product fra oDetails, så vi kan have en en-mange relation fra oDetails til Product. Derved kan man have flere produkter i den samme ordre.

Grunden til vi ikke bruger automatisk genereret ID på vores bunde og toppe, er fordi vores navne er unikke. Hvis vi havde haft mere tid havde vi designet det sådan at der også var kommet en ordre som havde fungeret ligesom oDetails, og så havde oDetails indeholdt elementerne fra Topping & Bottom.



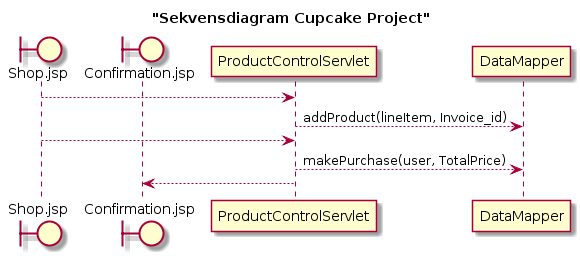
**Navigationsdiagram:**

Grunden til vi har valgt at lave en topmenu er for at gøre det lettere at navigere rundt. Vi synes også det var pænere som design at lave en topmenu. Vi har en Admin page hvor man kan se alle de ordrer der har været, denne side kan kun tilgås af brugere med administrator rettigheder. Vi har valgt at skrive menuen i en jsp fil, så hvis der skal rettes på siderne skal man kun lave om i en enkelt fil. Hvis der havde været mere tid til projektet havde vi separeret produkt hjemmesiden og kurven, stadig så man kunne se hvor mange varer der var blevet lagt i ens kurv. Men at man skulle ind på ens kurv for at se ens ordre inden man bestilte den.



**Sekvensdiagram:**

Vi har lavet det sådan, at addProduct opretter en cupcake til oDetails på databasen vha. et Invoice\_id som er unikt for alle brugere. Vores makePurchase holder styr på brugerens penge, som i vores program hedder balance. Dens primære funktion er at ændre CurrentStatus inde i oDetails til ’closed’ når kunden har færdiggjort sit køb, på den måde kan vi holde styr på brugerens penge og hvilke ordre der er åbne og lukkede i vores system.



**Særlige forhold:**

Gemt i session:

* Current user
* Shopping cart
* Bottom
* Topping
* Invoice

Grunden til vi har valgt at gemme tingene i session er så man undslipper unødigt arbejde, det vil sige at de forskellige ting ikke bliver oprettet flere gange end højest nødvendigt. Man slipper derved at lave flere kald og spilde ressourcer.

Resten bliver gemt i vores database, så deres informationer ikke forsvinder til næste gang de logger ind. Vi ved ikke hvordan vores exceptions håndteres. Vi validerer bruger-input fra vores Backend. Man kunne forbedre sikkerheden på vores projekt ved at hashe password, grunden til at det er en god ide at hashe passwords er så selv hvis der er en der får adgang til vores user tabel kan de ikke bruge informationer til noget fordi det er hashet. Siden kunne også havde blevet gjort mere sikker ved brug af SSL, så andre ikke kan se den sensitive trafik der foregår mellem serveren og brugeren. Vi bruger prepared statements som er en sikkerhed imod SQL injections. Vi har valgt at dele vores brugere op i to roller, Customers og Admins. Grunden til vi har gjort dette er så Admins har rettigheden til at tjekke invoices fra alle kunder. Hvis vi havde mere tid havde vi også tilføjet så Admins havde mulighed for at redigere i nogle af tabellerne gennem hjemmesiden. Det vil sige at man ikke skulle have fat i produktudviklerne hver gang der skulle laves minimale ændringer.

**Status på implementation:**

**Fungerer**:

* Login/Registration
* Check out
* Shopping cart
* Validering på login/registration
* DataMapper
* Menu der fungerer på alle siderne
* Klikke på invoices på admin siden
* Confirmation.jsp viser fakturaen

**Mangler**:

* JavaScript til validering af bruger input, så siden ikke reloader hver gang.
* Fejlside hvis man f.eks. bruger flere penge på cupcakes end hvad der ligger i ens balance.
* Informere brugeren om hvad man har gjort forkert under registrering/login hvis der er nogle krav der ikke bliver opfyldt. Som f.eks. hvis brugeren skriver et forkert password, fungerer siden p.t. så den bare reloader i stedet for at sige at passwordet var forkert.
* Vi mangler at style Admin.jsp
* Tilføjet en ordre til vores database, så der var bedre sammenhæng mellem oDetails – Products og vores Bottom og Toppings.

**Test:**

Vi har ingen tests lavet i dette forløb.