Cupcake

2. Semester

Prøveeksamensprojekt

Gruppe 7

Navn	Mail	GitHub navn
Alexander Jacob Ingemann Sarson	**cph-as485@cphbusiness.dk**	**AlexanderSarson**
Benjamin Paepke	**cph-bp133@cphbusiness.dk**	**Paepke-cph**
Mads Bredal Brandt	**cph-mb346@cphbusiness.dk**	**dublo144**
Oscar Nils Laurberg	**cph-ol38@cphbusiness.dk**	**OscarLaurberg**
Tobias Anker Boldt-Jørgensen	**cph-tb193@cphbusiness.dk**	**Tunoc**

GitHub repository

https://github.com/AlexanderSarson/CupCake

Droplet Server

http://134.209.233.32/CupCake

SonarQube

http://167.71.67.60:9001/dashboard?id=com.mycompany%3ACupCake

Indholdsfortegnelse

Indledning

Projektet *Cupcake* er et prøveeksamensprojekt, for 2. semester studerende, gående på uddannelsen Datamatiker på CPH-Business Lyngby. Projektet er en webapplikation, hvor det skal være muligt for en kunde at købe cupcakes, lave en bruger, ved brug af mail og kodeord. Projektet indebærer udarbejdelse af en frontend, hvor kunderne vil kunne fortage sine bestillinger, en logik del der manipulerer data til og fra frontend og backend. Derudover skal projektet også have en backend i form af en MySQL database, som holder al relevant data.

Projektet er udarbejdet i en gruppe af fem studerende, hvor arbejdsbyrden er fordelt ligeligt.

Baggrund

Projektet skal udforme en applikation for en virksomhed, der ønsker at sælge cupcakes. Firmaet der skal bruge applikationen, har en lokation hvor ordre skal afhentes, hvilke betyder at applikationen ikke har noget med levering at gøre. Da firmaet har en meget hurtig cupcake-maskine, der kan lave cupcakes i det de bliver bestilt, skal der heller ikke tage højde for produktionstid.

Applikationen skal kunne gøre følgende muligt for kunderne af Cupcake firmaet:

- Kunderne skal kunne vælge sine ønskede cupcakes, lave og betale[1] en ordre via applikationen.
- Kunderne skal kunne se valgmuligheder for at lave sin egen cupcake, som en sammensætning af en top og en bund.
- En kunde skal kunne oprette sig som bruger, med egen profil, på applikationen.
- Kunderne skal kunne se sine tidligere ordre.
- Kunderne skal ikke kunne færdiggøre en bestilling, uden at være logget ind på sin profil.

Teknologi valg

JDBC MySQL

For at danne forbindelse med vores MySQL database, gør vi brug af JDBC-version 8.0.17, dette er derfor et krav for at kunne tilgå database og derved for at kunne afvikle applikationen.

JUnit, Mockito & Selenium

For at kunne bedst teste og sikre stabilitet, gør vi brug af både JUnit-version 4.12, Mockito-version 3.0.0 og Selenium-version 3.5.3

Git & Github

Der er gjort brug af Git version 2.23.0 og Github til versionsstyring, hvilke også betyder at udviklingsmiljøet er underordnet, da ingen irrelevante filer deles.

Digital Ocean Droplet

Til lagring af vores web-applikation, gør vi brug af en Server, hosted af DigitalOcean. Serveren køre Ubuntu 19.04 x64 med 1 GB hukommelse og 25 GB lager plads. Severen køre Apache Tomat version 9.0.27 som håndterer HTTP kommunikationen med applikationen.

[1] Selve betalingen af ordren vil ikke blive inkluderet i dette projekt.

Azure DevOps

For at kunne sikre CI/CD gør vi brug af pipelinedelen af Azure DevOps til at teste og deploy vores kode, hvilket betyder at vi får fjernet fejlene "det virker på min maskine" og webapplikationen heletiden bliver opdateret på serveren.

Krav

Udgangspunktet for kravene, til systemet, er taget ud fra opgavebeskrivelsen[2] og i samtale med produktejerne[3]. Ud fra de formelle krav er der blevet dannet en række User Stories, som projektet er udviklet ud fra.

Funktionelle krav

Bruger

- Brugeren skal kunne oprette en konto.
- Brugeren skal kunne logge ind på sin konto.
- Brugeren skal kunne lave sin egen cupcake.
- Brugeren bestiller sine cupcakes.
- Brugeren tjekker tidligere ordre.
- Brugeren tjekker sin balance.

Administrator

- Administrator skal kunne se alle ordre og deres detaljer.
- Administrator skal kunne se alle registrerede brugere.

System

- Systemet skal kunne persistere brugerinformation.
- Systemet skal kunne persistere informationer om produkter, som bund og top til en cupcake.
- Systemet skal kunne persistere en brugers gennemførte ordre.
- Systemet skal tillade en bruger, der er logget ind, at gennemfører en ordre.
- Systemet skal kunne persistere brugerens indkøbsvogn gennem en hel session[4].

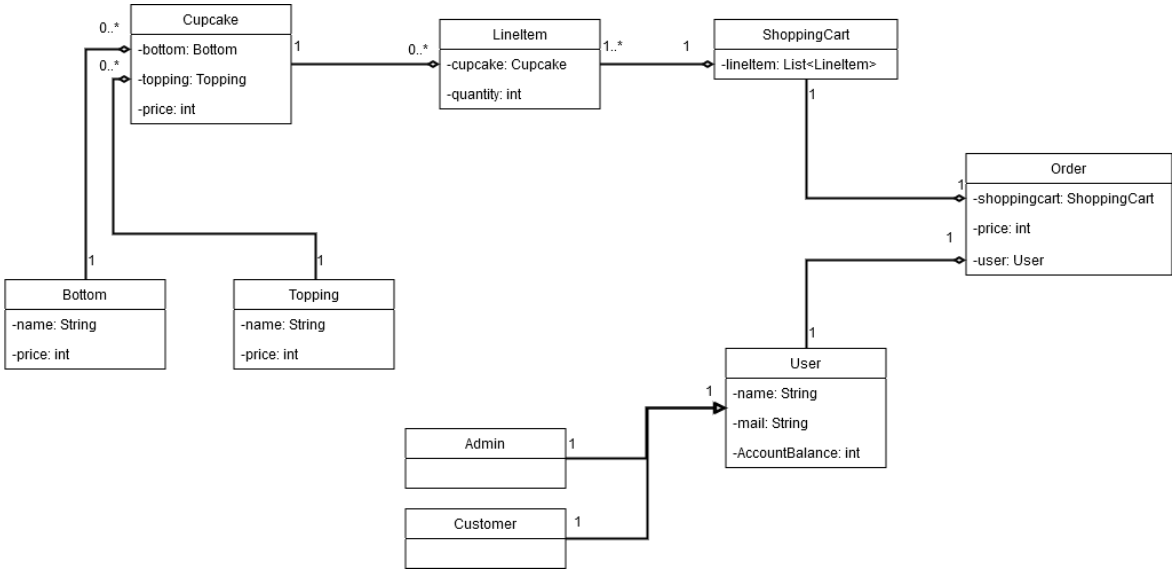
[2] Opgavebeskrivelse: https://datsoftlyngby.github.io/dat2sem2019Fall/uge41/CupCake.html
[3] Underviserne: Tina Marbjerg og Tue Hellstern
[4] En session er fra brugeren har tilsluttet sig webshoppen og til de forlader webshoppen igen.

Domænemodel og ER model

Domæne model

Domænemodellen er udarbejdet ved brug af opgavebeskrivelsen[5]. Tilgangen til at danne domænemodellen var at tage opgavebeskrivelsen og finde navneord og udsagnsord, for at danne et overblik over hvilke komponenter og processer der forekommer i firmaet, som skal sælge cupcakes. Igennem processen blev der fundet følgende komponenter, illustreret i Figur 1 Domænemodel.

- Brugere af systemet kan differentieres i to, henholdsvis Administrator og Kunde. Administrator har samme rettigheder som kunde, han kan købe cupcakes og se tidligere ordre. Derudover kan administratoren også se alle kunder og deres ordre.



Figur 1 Domænemodel

ER model

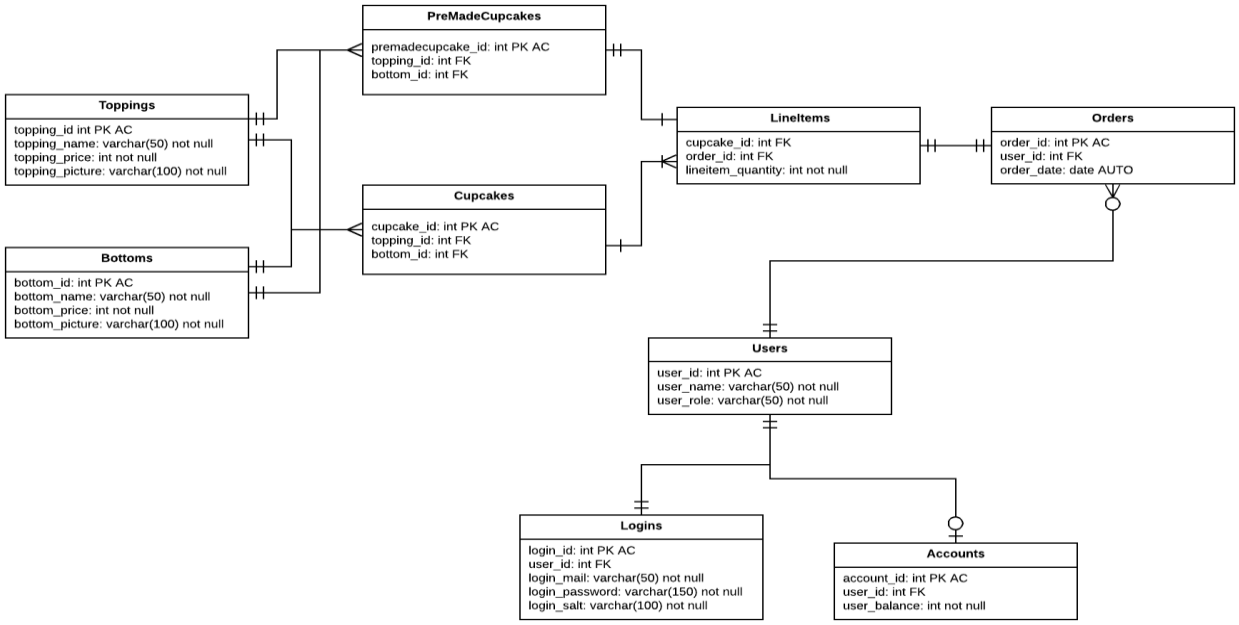
Lige som domænemodellen er ER modellen også udarbejdet ud fra opgavebeskrivelsen og Domænemodellen.

- Toppings og Bottoms tabellerne opfylder ikke kravene for 3. Normalform, da name og picture er afhængig af hinanden og vi burde måske have lavet en "flavour table", men tiden har ikke været der.
- Logins table opfylder ikke kravene for 3. Normalform, da password og salt er afhængig af hinanden, dette burde have været trukket ud til sin egen table, men tiden har ikke været der.
- Logins, Users og Accounts har en til en relation, dette ville vi have ændret, hvis tiden havde været der.

Databasen er designet ud fra at opnå 3. Normalform. Vi har brugt primary og foreign keys for at holde konsistensen gennem hele databasen. Vi kunne godt have tænkt os at have redesigned Toppings og Bottoms tabellen, da de indeholder de samme felter, men havde desværre ikke tiden til at komme op

[5] Opgavebeskrivelse: https://datsoftlyngby.github.io/dat2sem2019Fall/uge41/CupCake.html

på en god løsning.

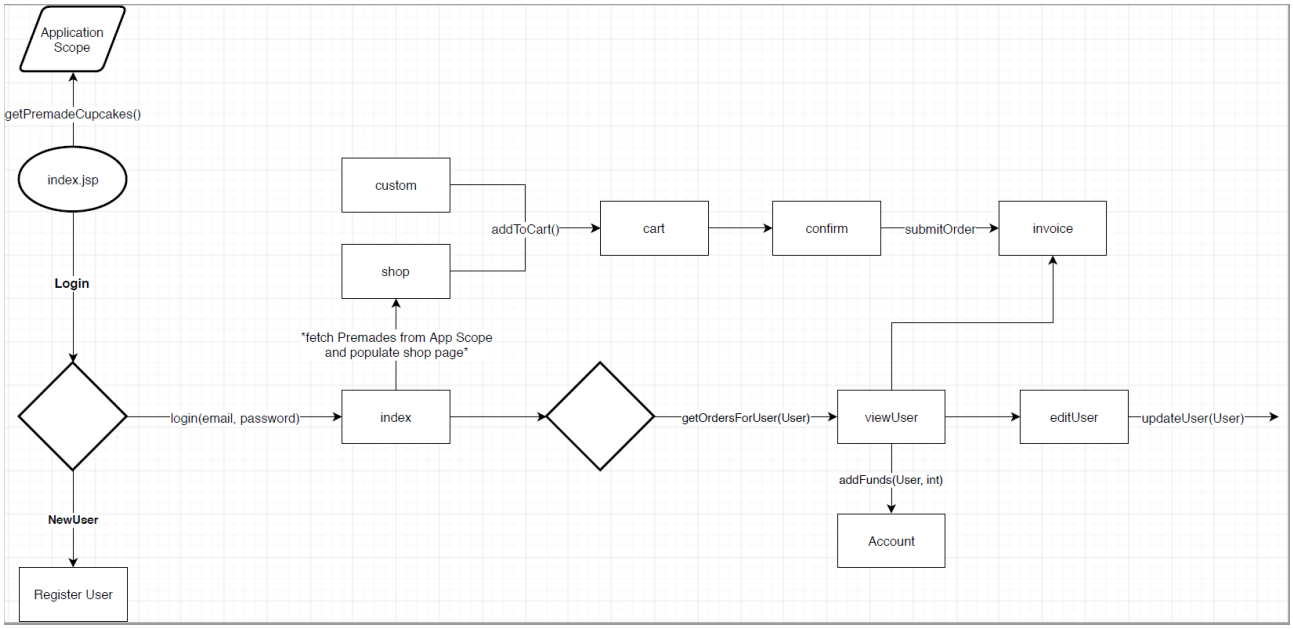


Figur 2 ER Model

Navigationsdiagram

Navigationen i systemet er udarbejdet omkring en navigationsbar, hvor alle navigations muligheder forekommer

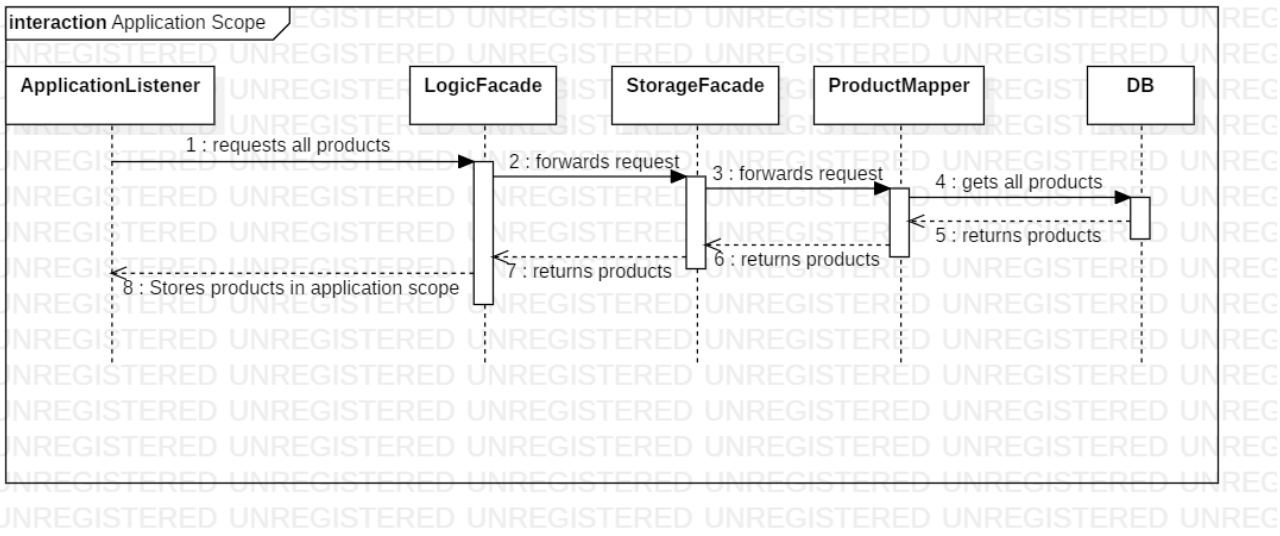
- Systemet gør brug af en navigations bar i toppen af alle sider. Det er derfor muligt fra alle steder i programmet at tilgå følgende sider: Index.jsp, mainShop.jsp, login.jsp og createUser.jsp.
- Er brugeren allerede logget ind, så erstattes login.jsp og createUser.jsp med en log ud funktion i navigationsbaren.
- Er brugeren en administrator har brugeren også adgang til adminPanel.jsp i navigationsbaren.



Figur 3 Navigationsdiagram

Sekvens diagrammer

Nedenstående sekvensdiagram viser hvordan vi går fra vores Application Scope ned igennem Logic facaden, Storage facaden, Produkt mapperen og Databasen, for at instansiere alle vores premade cupcakes, toppings og bottoms. Denne process bliver kørt 1 gang i det at vi deployer vores program på vores Droplet server, og skal redeployes i ændringer af at der bliver tilføjet en hel cupcake, topping eller bottom i databasen.



Figur 4 Application Scope

Særlige forhold

- Shopping cart og bruger bliver gemt i http sessionen.
- Exceptions bliver kastet op til presentation laget, hvor der bliver vist en meningsfuld besked til brugeren.
- Validering af brugerinput er lavet ved hjælp af JavaScript.
- Vi har valgt at kryptere brugernes passwords ved hjælp af salt 256.
- Vi har 2 forskellige type brugere. Kunde og administrator. Dette holder vi styr på via en kolonne i databasen som hedder rolle i users table.
- Vi har valgt at styre programflow via command patterns.
- Alt information i forhold til at forbinde til database ligger i en properties fil.
- Vi har valgt at tage Selenium og Mockito i brug, så vi kan opnå større code coverage og mindske risikoen for fremtidige fejl.

Status på implementering

Generelt

- Opsat server med sonarqube, som hele gruppen kunne tilgå hvor som helst, når som helst.
- Har ikke nået at implementere Junit categories, for at gøre build flow lettere.

Frontend

- Rent visuelt, er webshoppen meget simplificeret, funktionel stabilitet blev stillet højere og derved blev et lækkert design ikke implementeret.

Backend (Business)

- Til kryptering af kodeord, bliver der gjort brug af SHA-512. For at forbered sikkerheden var det planlagt at gøre brug af Salt[6]. Det er dog ikke blevet implementeret.

Backend (Database)

- DataSource, som styrer Connection Pooling[7], er i nuværende version ikke lavet som en Singleton, hvilke gør at det er muligt at have flere instanser af DataSource som ikke er optimalt.
- Følgende CRUD-operationer er ikke blevet implementeret:
    - Update Order
    - Delete Order
    - Delete User
    - Delete Product

[6] https://crackstation.net/hashing-security.htm set 31. oktober

[7] Dynamisk styring af forbindelser til database.

Test

Unit Tests

Systemet er udviklet ved brug Test-Driven Development, det har medført udarbejdelse af Unit Tests, til at definerer logikken i de mindre delkomponenter

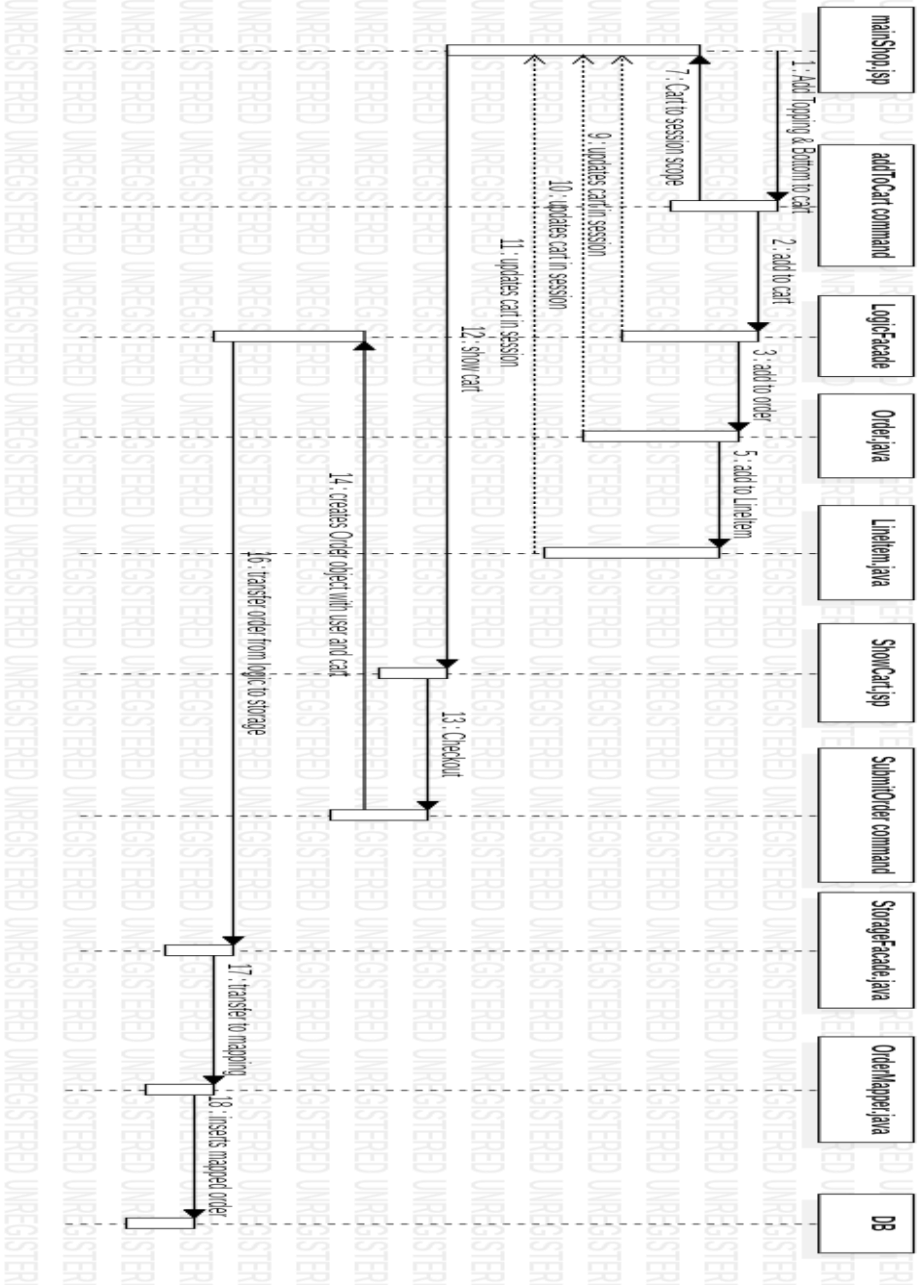
Behavior Tests

For at teste specifikt Mapper klasserne, som omdanner databaseinformationer til Java klasser, er der blevet gjort brug af Mockito frameworket for at kunne teste adfærden uden at skulle teste op imod en database.

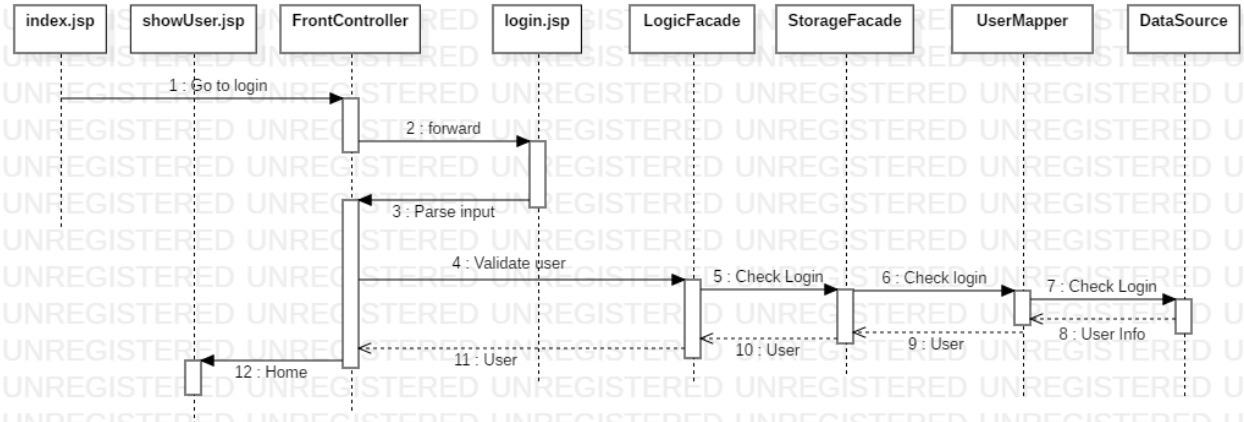
Integration Tests

Til at teste om vores system kan ændre i databasen, f.eks. oprettelse af brugere, oprettelse af ordre, eller ændre en bruger, er der blevet udarbejdet integrations tests. Integrations testene er med til at sikre at vi kan indsætte, hente og læse data på databasen. Ved brug af SQL Scripts genopbygger vi databasen efter hver test er kørt.

Bilag



Figur 5 Order Sequence



Figur 6 Login Sequence