Cupcake rapport

Gruppe C4 / 28.04.2021

Johan Karpantschof

cph-jk389@cphbusiness.dk

github@Capa2

Alexander Michelsen

cph-am8@cphbusiness.dk

github@alex124-bot

Indholdsfortegnelse

Indledning

Cupcake projektet er struktureret som en “rigtig” udvikler opgave. kunden ønsker en webshop hvor deres kunder kan bestille kager til afhentning. Dette skal afleveres som en web app, med tilhørende grafiske mockups, diagrammer og kildekode.

På et teknisk plan er målet er at bygge en web app som kan simulere en webshop med simple klient sessioner, som læser og skriver vedvarende data fra klientens interaktioner gennem applikationen og tilhørende database.

Appen er baseret på denne startkode: https://github.com/jonbertelsen/sem2-startcode

Baggrund

Kunde

Kunden "Olsker Cupcakes" er et nystartet økologisk cupcake bageri på Bornholm.

Funktionelle Krav

Kunden skal have mulighed for at:

- Logge ind med email og kodeord. Email skal herefter være synlig på alle sider
- Oprette en konto/profil for at kunne betale og gemme en en ordre
- Bestille og betale cupcakes med en valgfri bund og top til afhentning i butikken
- Se valgte cupcakes og deres samlede pris i en indkøbskurv
- Fjerne cupcakes fra indkøbskurven

Admin skal have mulighed for at:

- Logge ind med email og kodeord. Email skal herefter være synlig på alle sider
- Indsætte beløb på en kundes konto direkte i MySql
- Se alle ordrer i systemet med tilhørende data om kunden og valgte cupcakes.
- Se alle kunder i systemet med tilhørende data om tidligere ordrer
- Fjerne ordre fra systemet

Software

I. IntelliJ IDEA 2020.2.2 (Ultimate Edition)

Vores primære programmeringsværktøj til kodeskrivning. Vi har også brugt intellijs inbyggede værjtøjer til at køre vores app på en lokal server gennem produktionsfasen.

II. MySQL Workbench 8.0.22

Vi har brugt MySQL workbench til at udarbejde vores ER diagram som basis for vores database, og efterfølgende til at designe og manipulere vores database tabeller.

III. AdobeXD 39.0.12

Vi har brugt AdobeXD til at designe grafiske mockups af web appen.

IV. Git 2.31 & Git Bash 4.4

Vi har brugt git til at dele vores kode med hinanden under produktionsfasen, og til aflevering af det endelige projekt på github. Vi har brugt Git Bash som git-værktøj, samt adgang og håndtering af vores fjern server (Digitalocean droplet).

Teknologi

V. JSP, HTML, CSS & Bootstrap

Vi har brugt HTML og CSS med twitter bootstrap til den basale opsætning af web appen. HTML elementer genereres dynamisk gennem vores JSP dokumenter.

VI. Tomcat

Tomcat hoster vores JSP dokumenter som generer HTML sider baseret på kundens efterspørgsler i web appen. Data hentes og skrives fra databasen gennem Java applikationen.

VII. Java

Java applikationen styrer request og sessions data. Når en klient efterspørger eller postere data på web appen beregner java appen den korrekte respons og sender den nødvendige data videre til serveren (Tomcat) eller databasen (MySQL).

VIII. Maven

Java applikationen bygges med Maven. Java dependencies håndteres også af maven.

IX. MySQL

Information om produkter, ordre og kunder lagres i en MySQL database som tilgås gennem java applikationen.

X. DigitalOcean Droplet

Hele systemet bliver kørt og lagres på en ubuntu server, som køre på en digital ocean droplet.

Krav

Vision

Firmaet ønsker en tilgængelig web applikation, hvor deres kunder kan bestille vare til afhentning når det passer dem, så deres produkter kan være klar, når de kommer ned i butikken. Betaling skal kunne foregå gennem appen. Kunden skal også kunne tilgå sine tidligere ordrer.

Applikationen skal også have de nødvendige værktøjer for at firmaets ansatte kan håndtere ordrer og kunder i systemet. Disse data skal præsenteres på en fornuftig måde, så de ansatte kan få overblik over hvilke ordrer der skal laves, eller hvem deres kunder er.

User Stories

US-1: Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, sådan at jeg senere kan køre forbi butikken i Olsker og hente min ordre.

US-2: Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en en ordre.

US-3: Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer.

US-4: Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.

US-5: Som kunde eller administrator kan jeg logge på systemet med email og kodeord. Når jeg er logget på, skal jeg kunne min email på hver side (evt. i topmenuen, som vist på mockup'en).

US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

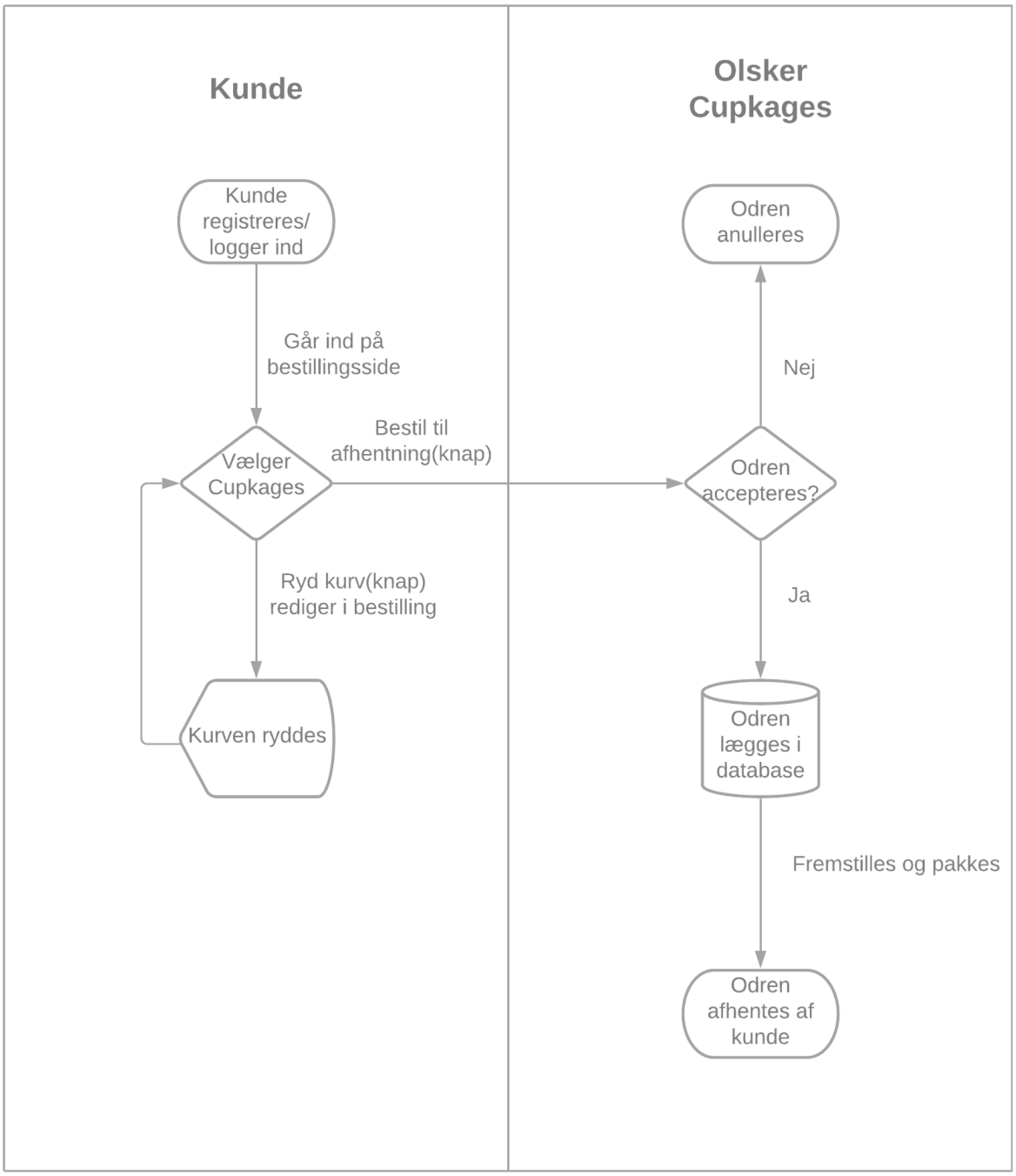
US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

US-8: Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.

US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

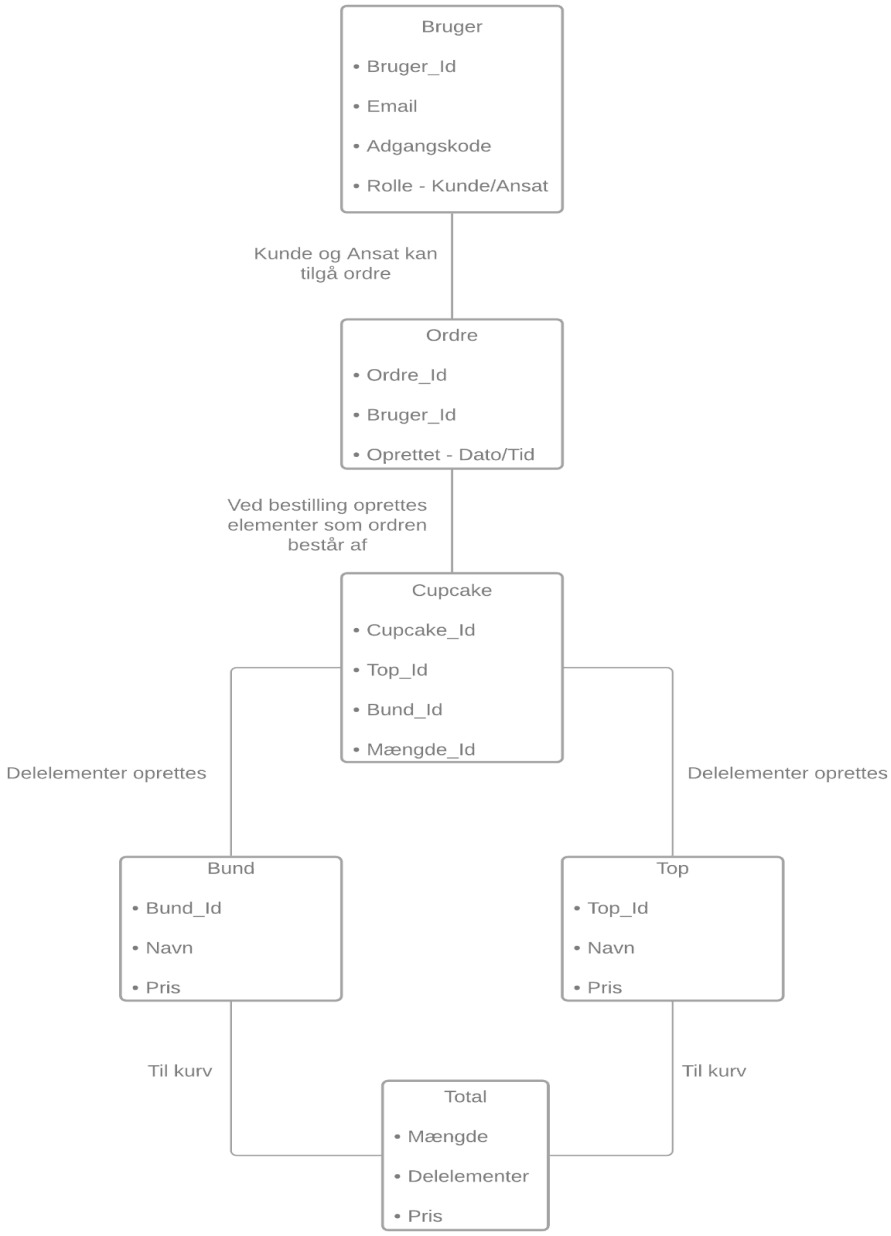
Aktivitetsdiagram

Aktivitetsdiagramet eller flow diagramet viser den process det er systemet skal varetage fra a til z og givne forskellige udfald samt cirkulærer bevægelser inden for systemet, med kunden på den ene side og virksomheden på den anden med systemet som forbindelsesfaktor gennem hele forløbet.



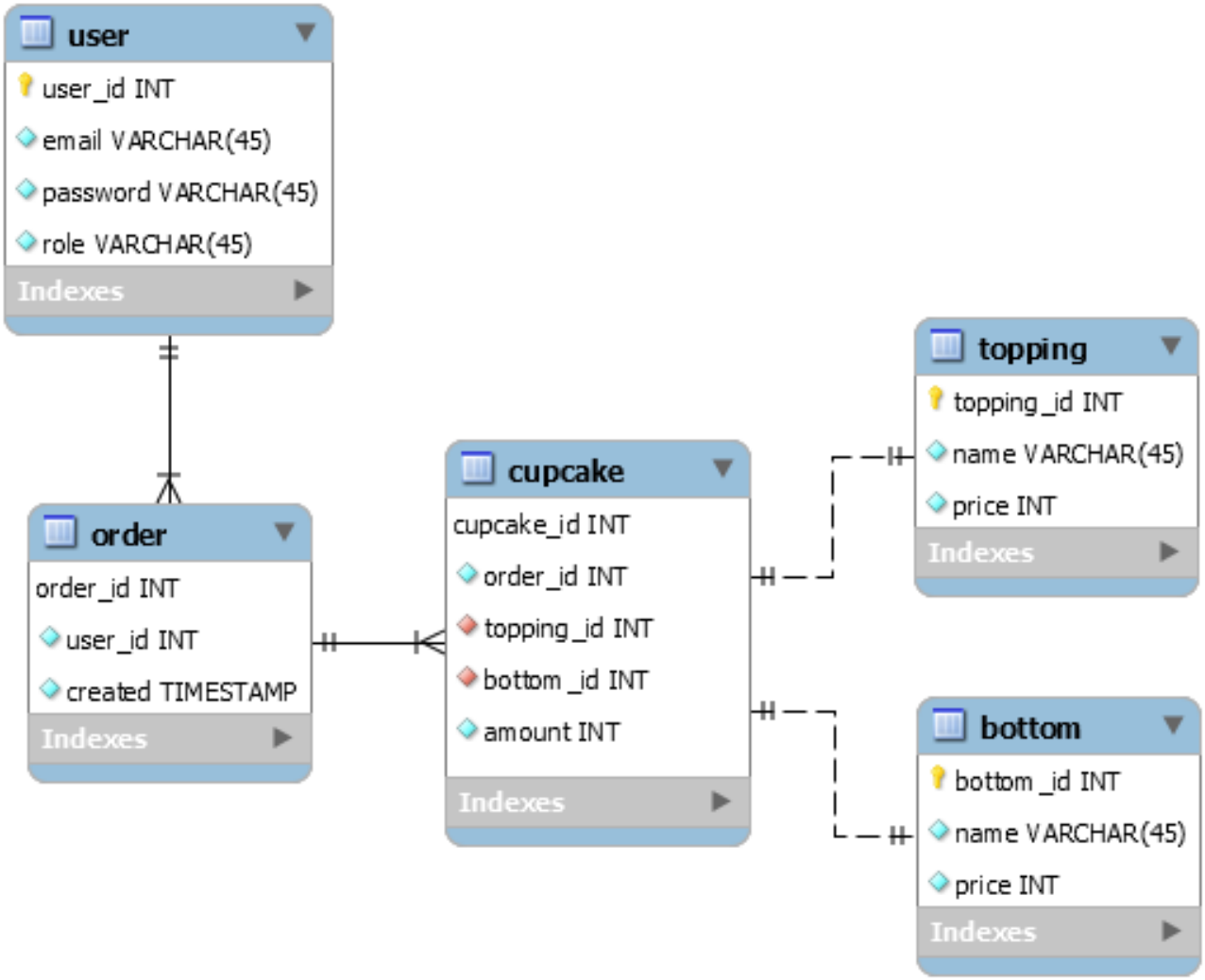
Domæne model

Modellen viser nogle af vores klasse kandidater med attributter. Den er ikke ment som et aktivitetsdiagram men har alligevel en lignende opdeling og sammenhæng da vi har designet vores projekt efter at tilpasse kundens behov, som kunne være at have det mest optimale flow i og af sammenhæng i klasser og attributter. Linierne er med til at beskrive relationer. Diagrammet er ment som én opfattelse af klasser og attributter for vores projekt og er ikke retningsbestemt.



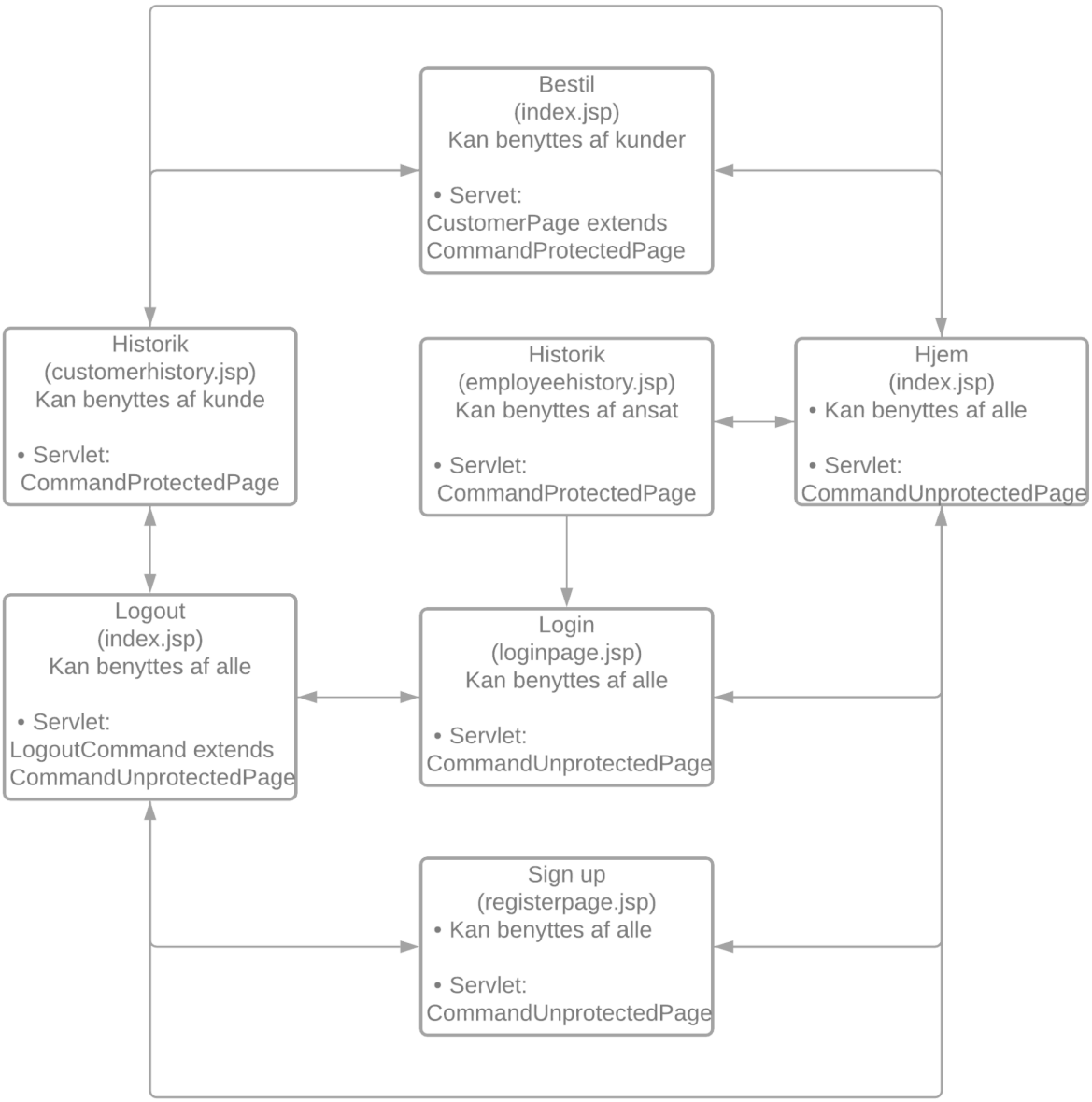
ER diagram

Vores ER diagram afbilleder vores database som model med primærnøgler, fremmednøgler og ellers relevante kolonner. Databasen hænger sammen med to en til mange relationer mellem user og order, og igen mellem order og cupcake. Der er også to en til en relationer mellem cupcake og henholdsvis topping og bottom. Vores ER diagram er på 2. Normalform og ikke 3. da flere kolonner er indirekte afhængige af nøglen.



Navigationsdiagram

Navigationsdiagrammet viser forbindelsen mellem de forskellige .jsp sites samt hvilke servlet de gør brug af samt om de extender andre servlets. Pilene angiver om man kan og om det er muligt ift. menu osv. at tilgå siden fra andre sider. Der er også med hvilken rolle man skal have for at kunne autoriseres til side tilgang.



Særlige forhold

Applikationsdata

Produktinformationer, dvs. top og bund og tilhørende pris m.m. hentes fra databasen til applikationen, så alle brugere såvel som admins kan tilgå det i forskellige kontekst.

Sessionsdata

Brugerdata, dvs. email og password gemmes i sessionen efter brugeren har indtastet sine oplysninger, såfremt disse kan blive valideret af databasen.

Exceptions

Den brugerdefinerede *exception* "UserException" kastes når et brugerinput eller interaktion ikke er gyldig, og indeholder et string som vises til brugeren som fejlmeddelelse.

Sikkerhed

Alle forespørgsler til databasen foregår gennem prepared statements, for at undgå SQL Injection. Der er ikke taget foranstaltninger for at sikre brugerdata i java applikationen.

Status på implementation

Vi har ikke nået at lave tests. Vi har ikke implementeret funktionalitet på admin siderne eller kunde historik siden.

Bestillingssiden er funktionsdygtig med mulighed for at tilføje redigere, og bestille produkter. Hver ordre bliver gemt i databasen, men der er stadig fejl i forbindelse med at gemme cupcake data i databasen.

Vi har stylet siderne.

Vi har deployet web appen på en droplet med tomcat server og mysql database:
http://104.131.165.37:8080/cupcake-1.0-SNAPSHOT/

Proces

Hvad var jeres planer for teamets arbejdsform og projektforløbet?

Vi havde ingen konkret plan for projektet. Som en 2 persons gruppe vidste vi begge skulle deltage i alle dele af løsningen.

Hvordan kom det til at forløbe i praksis?

Vi holdt dagligt hinanden opdateret om hvilke arbejdsopgaver vi var i gang med, og lavede to-do lister med diverse opgaver. Vi arbejdede typisk en opgave hver især, med mindre der var kompatibilitets problemer og eller behov for fejlfinding.

Hvad gik godt og hvad kunne have været bedre?

Der var flere dage - især i starten - hvor vi ikke havde nogen reel slagplan, og der ikke blev taget initiativ til lægge en. de dage arbejde vi lidt i blinde grundet den manglende kontakt - eller slet ikke. En stærkere og mere organiseret start ville have gjort en stor forskel.

Hvad har I lært af processen og hvad vil I evt. gøre anderledes næste gang?

Vi er gode til at arbejde, dele opgaver op, og vi er kompetente til at kode. Vi har svært ved at organisere os og dårlige til at overholde aftalte mødetidspunkter, hvilket leder til uoverskuelige arbejdsforhold og uproduktive dage eller ugyldig kode.

Næste gang vil vi afsætte mere tid til planlægning, og forsøge at introducere mere struktur i vores arbejdsgang og kommunikation.