Otto Dyva Toubro
cph-ot@cphbusiness.dk
Github: OttoDyva

Ryan Zachariasen
cph-rz72@cphbusiness.dk
Github: RyanZachariasen

Bekhan Mahauri
cph-bm185@cphbusiness.dk
Github: BekhanM

Cupcake

CPH Business: DATB 2. Semester 02/04/2024
https://github.com/RyanZachariasen/CupcakeBRO

Inholdsfortegnelse

Indledning

Projektet Cupcake er en opgave uddelt på 2. semester, hvor man skal lave en hjemmeside for et bornholmsk cupcake firma. En kunde skal selv kunne vælge deres ønskede cupcakes, bestille dem, og derefter hente dem i butikken. Projektet skal opfylde nogle funktionelle krav som f.eks. en loginmetode og en administratorrolle som kan se dele af hjemmesiden en normal kunde ikke kan. Der er derudover også nogle ikke-funktionelle krav som skal opfyldes som f.eks. et mockup i Figma eller lignende, programmet skal virke på både computerskærm og for en iPhone 12 og databasen skal normaliseres til 3. normalform. Hele projektet skal så afleveres og derefter fremlægges.

Baggrund

Olsker Cupcakes er et iværksætterprojekt fra byen Olsker på Bornholm, som med deres økologiske fremgangsmåde har fundet den helt rigtige opskrift på de perfekte cupcakes. Et par hipstere fra København har efterfølgende været forbi, for at lave et mockup af den hjemmeside vi skal lave til deres virksomhed.

Som krav til hjemmesiden, vil de gerne have en sammenhængende hjemmeside, hvori man kan bestille en cupcake med valgfri bund og top. Efterfølgende skal man så kunne vælge hvor stort et antal man skal have af hver cupcake, hvorefter kunden skal kunne se alle deres cupcakes i deres indkøbskurv med en samlet pris, - inden de betaler.

Kunden skal også kunne oprette en bruger med e-mail og adgangskode, så man kan gemme ordrer og betale for dem. De vil derudover også gerne have en administratorrolle, som skal kunne gennemse alle kunder og ordrer i systemet. Administratoren skal kunne gå ind i databasen og tilføje penge til kundens konto, så kunden kan betale for deres ordre.

Som en ekstra tilføjelse vil de gerne have, at administratoren kan fjerne gamle ordrer som ikke er blevet betalt. Både kunden og administratoren skal kunne se deres email på hjemmesiden, for at vide hvilken konti de er på.

Teknologivalg

Vi i GruppeBRO har brugt de følgende teknologier :
Docker Desktop 4.27.1
pgAdmin/postgres 4:8.3
JDBC/PostgreSQL 42.7.2
Intellij IDEA 2023.2.5 (Ultimate Edition)
Maven 4.0.0-alpha-13
Javalin 6.1.3
ThymeLeaf 3.1.2 Release
Amazon corretto-17

Vores Cupcake program har flere komponenter, som vi bruger til at få alt til at køre. Til databasedelen brugte vi Docker Desktop, Postgres, og pgAdmin, for at gemme information i databasen. Med brug af PostgreSQL driveren havde vi koblet vores Intellij med databasen, så at ThymeLeaf og Javalin kunne køre som det skulle. Maven/pom.xml filen brugte vi til at importere ThymeLeaf og Javalin repositories så backend kunne køre. Vi programmerede i Java Version 17 eftersom det viser sig at være en noget mere stabil version.

Krav

Olsker Cupcakes håb for projektet er, at få en sammenhængende og nem hjemmeside som deres kunder nemt kan tilgå, som skal udvide deres kundebase. Hjemmesiden skal hjælpe kunden igennem processen til at købe deres ønskede cupcakes, som de derefter kan hente i butikken, efter deres cupcakes er blevet bestilt.

User cases

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 ordre.

US-3: Som administrator kan jeg indsætte beløb på en kundes konto direkte i Postgres, 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 se 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.

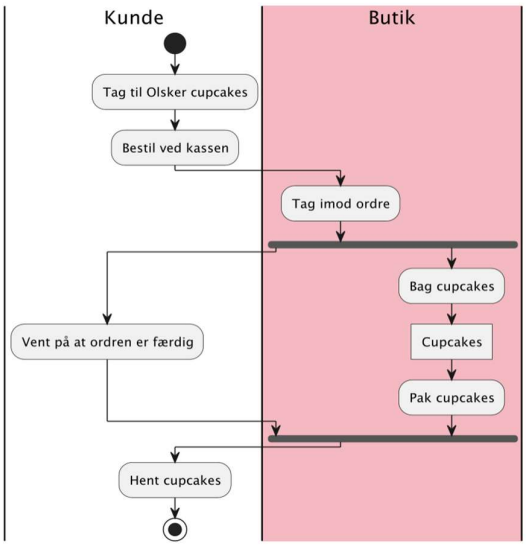
US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

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

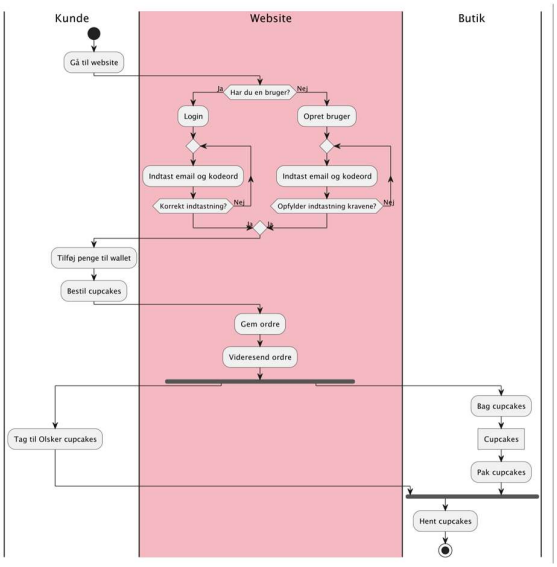
US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Aktivitetsdiagram

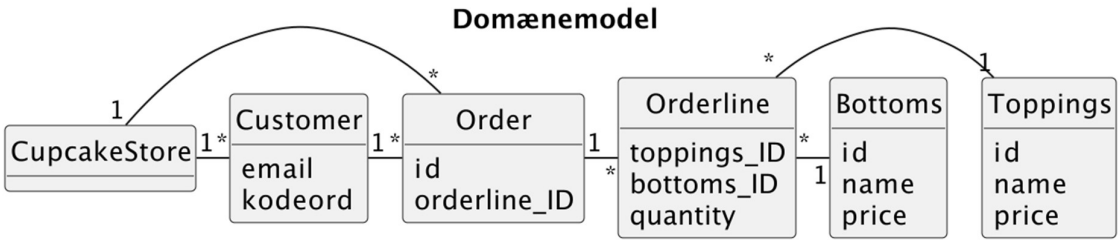
AS-IS



TO-BE

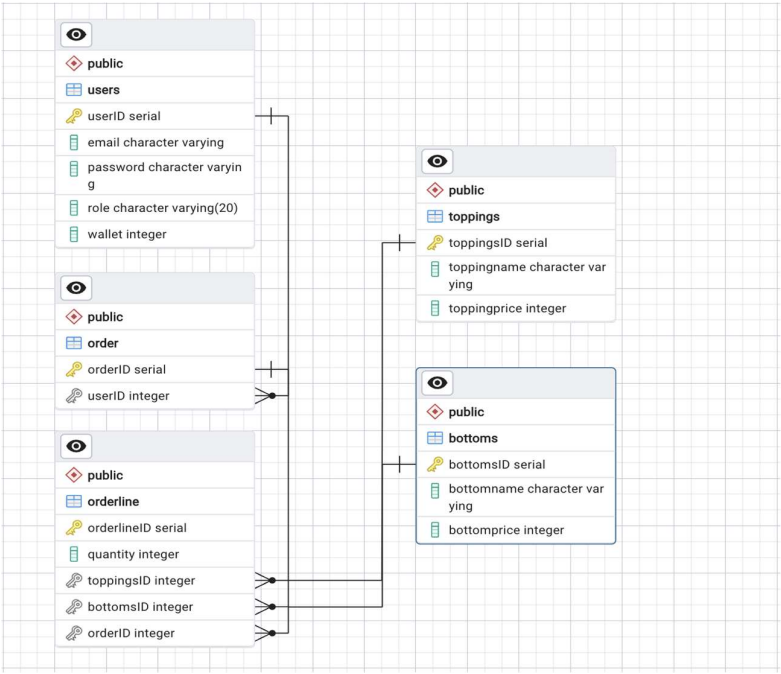


Domænemodel



Med domænemodellen er det vigtigt at man laver den før man begynder at kode noget som helst. Det vigtige her er at få styr på de her relationer og multiplicitet, så vi har haft lidt overvejelser omkring hvordan vores projekt skulle se ud og hvordan det ellers kunne have set ud.
En af vores overvejelser har været relationen mellem Orderline, Toppings og Bottoms. Vi har valgt at sige, at mange ordrelinjer kan have 1 topping og 1 bund. Dette er fordi ordrelinjen kun kan have en af hver og derfor ville mange til mange kun passe, hvis hver ordrelinje kunne have flere forskellige toppings og bunde.

ERD-diagram



Vi har vha. PostgreSQL skabt et 'Entity-Relationship Diagram' med formålet at skabe et klart og præcist overblik over vores datamodellering til brug i vores Cupcake Project. De fleste af vores database-tabeller følger 3.NF. Dvs. data og de enkelte attributter er organiseret i separate tabeller, primært for at undgå overflødige gentagelser.

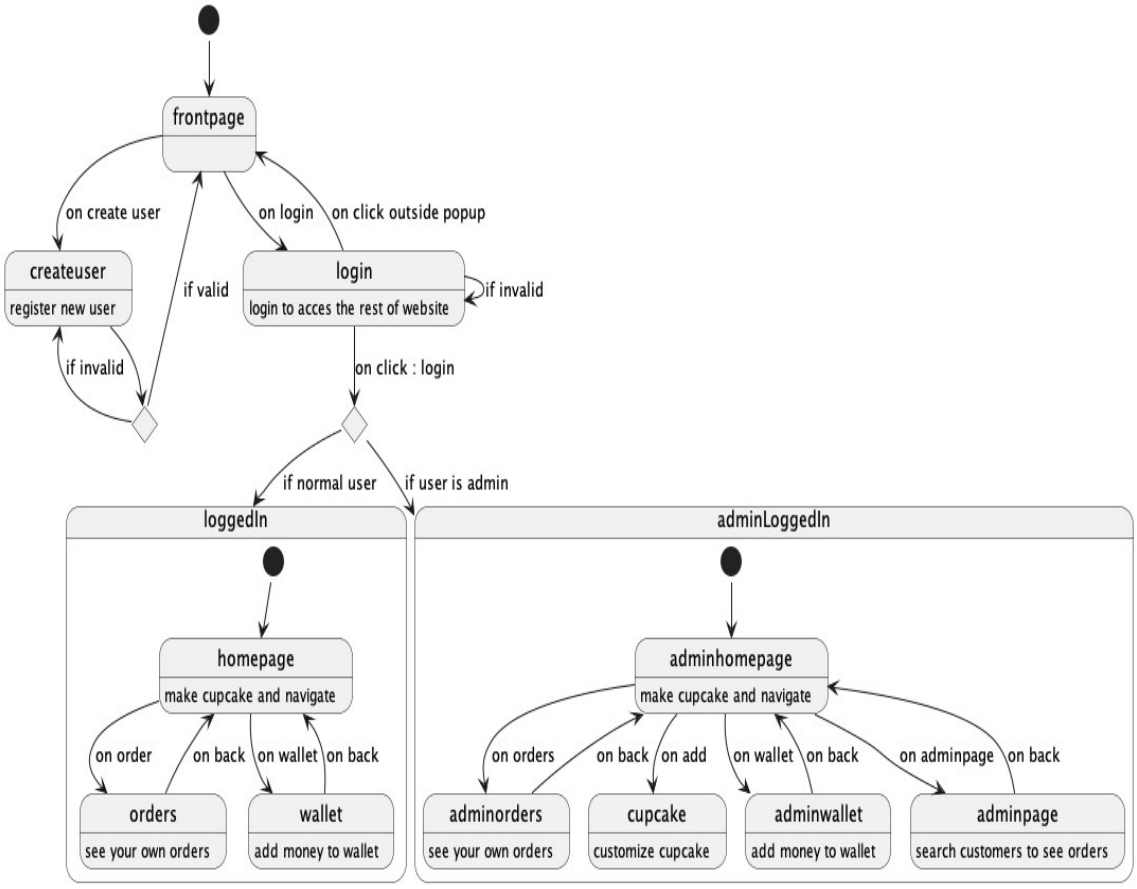
Tabellerne er lavet med simplicitet i tankerne, der skal særligt lægges mærke til *orderline* tabellen, som fungerer som en mellem relations tabel mellem *orders*, *bottoms* og toppings.

Ift. relationer i vores ERD ses det at vi har besluttet os for at køre med en 1:M relation, da dette passer bedre i vores program. Én *user* kan have mange *orders*, til gengæld tilhører hvert *order* blot én *user*. Det samme ses i følgende relationer:
Order - orderline relation*, Orderline - bottom* relation, *Orderline - topping* relation.

I vores databaser bruger vi godt brug af fremmednøgler med det primære formål at opretholde en sammenhæng mellem tabellerne for bl.a. øget simplicitet og fleksibel brug i vores mappers. Som eksempel kan man se på vores *user - order* relation. *UserID* i *order* refererer til *Users* tabellen for at identificere den tilknyttede *User*. Det samme ses i følgende relationer: *Order - orderline* relation*, Orderline - bottom* relation, *Orderline - topping* relation.

Samlet set opnår vores ERD en tydelig struktur, og hjælper med at visualisere og forstå vores database opstilling.

Navigationsdiagram



Særlige forhold

I frontpage.html har vi indført et login "pop-up", som kommer frem når man trykker på login-knappen. Det er blevet lavet med forskellige css-styles og en smule javascript, som giver en animering når man åbner den, ved at bruge en getElementById() metode, som man derfor kan åbne og lukke som ønsket.

Når man er valideret som admin popper der et admin page op, hvori du kan indtaste email på en bruger i systemet, og se alle deres ordre. Dette gøres ved hjælp af en SQL-statement i mapperen som tager fat i det nødvendige data/orderline, og søger efter emailen vha. SQL's LIKE funktion. Dette bliver efterfølgende routet gennem *OrderlineControlleren*, som displayer dataen efter man har indtastet emailen i vores adminpage.html.

Status implementation

Vi har ikke nået at lave nogle af de uddelte User-Stories, som opgaven bad os om. De er US-4, US-8 og en del af US-2. I US-4 og US-2 skal man kunne betale for en ordre; det kræver at man skal have en indkøbskurv, som vi desværre også mangler.

Det har påvirket vores program på den måde, at der er manglende user experience, da brugeren ikke har mulighed for at kunne skifte mening før en ordre er placeret. I stedet for gemmes ordren direkte i databasen og kundens "wallet" opdateres ikke. Kunden har heller ikke mulighed for at kunne justere ordren i vores nuværende program. Kunden vil blive tvunget til at bede en admin om at slette ordren, men det kan admin heller ikke som US-9 stiller krav til, fordi ordren er fastlagt i programmet.

Vores hjemmeside lider også af fejl, såsom når man forsøger at oprette en cupcake og efterlader felterne tomme, går den tilbage til forsiden i stedet for at informere brugeren at felterne er tomme.

Når man indsætter penge i sin "wallet" og derefter vender tilbage til homepage, viser hjemmesiden den tidligere ikke-opdateret saldo og ikke den nye, selvom det bliver gemt i databasen.

Proces

Da vi begyndte projektet havde vi en god ide om hvordan vi ville komme i gang. Vi aftalte at to begyndte på backend og en arbejdede med frontend. På den måde ville frontend kunne fortælle backend hvad der var brug for, før at han kunne arbejde videre på specifikke dele. Vi startede derfor med frontpage og implementerede herefter den næste side, kunden ville komme ind på.

I praksis virkede det rigtigt godt med at alle vidste hvad vi var i gang med og hvad der skulle løses. Problemerne opstod da vi skulle prøve at implementere vores user cases, da vi endte med at blive alt for opslugte i det vi var i gang med og derfor glemte at få styr på hvad der skulle være med i vores projekt. Vi begyndte også at tilføje metoder, der ikke var blevet bedt om, som for eksempel vores wallet.html, som ikke var en del af opgaven, men kun noget vi syntes var smart. Derudover var vi begyndt på at kode, før vi havde lavet både aktivitetsdiagrammerne, navigationsdiagrammet og domænemodellen, hvilket ikke var gået, hvis vi havde en fysisk kunde, som vi skulle kommunikere med.

I det hele taget mener vi at vores arbejdsproces gik godt, vi er stærke til at kommunikere og finde de bedste løsninger. Vi skal være bedre til at følge opgavebeskrivelsen og følge de normer der er for et projekt. Vi bliver til tider for opslugte i hvad vi finder interessant og derfor når vi ikke at løse de krav der er blevet stillet.

Hele processen og at arbejde på projektet har været en god oplevelse, vi har lært meget om hvordan vi alle arbejder bedst og især hvordan man skal forberede sig på projektet, inden man begynder at kode.
Næste gang vil vi især fokusere på at arbejde på alle vores diagrammer og modeller, inden vi begynder, så vi får en bedre ide om hvordan projektet skal opstilles. Vi vil også være bedre til at følge vores funktionelle krav og ikke bare begynde at kode, uden at have en plan for hvordan vi vil løse de stillede krav til opgaven.