Cupcake Projekt

- Morten Bomholt Mikkelsen, cph-mm769@cphbusiness.dk , Github: mbm1337
- Mustafa Altinkaya, cph-ma763@cphbusiness.dk , Github: altinkaya
- Mounir Salem, cph-ms848@cphbusiness.dk , Github: ETHMUNI

Rapport dato: 2. November 2023

Projekt dato: 30 Oktober 2023

Indholdsfortegnelse

Indledning

Projektet sigter mod at realisere en online platform, der tillader brugerne at bestille og tilpasse deres cupcakes og administrere ordrer.
De identificerede user stories inkluderer funktioner som bestilling og betaling af tilpassede cupcakes, oprettelse af profiler, administrations muligheder for ordre- og kundehåndtering samt betalingsadministration.
Projektet fokuserer på at omsætte de aftalte user stories til en fuldt fungerende hjemmeside, der imødekommer behovene for både kunder og administratorer. Hvert trin i udviklingen sigter mod en velfungerende prototype, der gradvist udvider funktionaliteten og opfylder kundernes og virksomhedens krav.

Baggrund

Olsker Cupcakes består af et par hipsters fra København som har åbnet et økologisk bageri fra Bornholm, som har ramt den helt rigtige opskrift.

Kundens krav:

- Bestille og betale for cupcake med en valgfri bund og top
- Køre forbi Olsker og hente sin ordre
- Oprette en profil og kunne betale og gemme en ordre
- Som administrator indsætte beløb på en kundes konto direkte i Postgres, så kunde kan betale sin ordrer
- Som kunde se valgte ordrelinjer i en indkøbskurv så kunden kan se den samlede pris.
- Logge på systemet med email og kodeord
- Når man er logget på se sin email på hver side. Evt. i topmenu
- Som administrator se alle ordrer i systemet
- Som administrator se alle kunderne og deres ordrer i systemet
- Som kunde skal man kunne fjerne en ordrelinje fra indkøbskurv
- Som administrator kunne fjerne en ordrer hvis kunden ikke har betalt

Teknologi valg

- Javalin 5.6.1
- Java 17
- HTML
- CSS
- Runtime Environment: Corretto-17.0.8
- Thymeleaf 3.1.2
- Postgres Database

Krav

Vision og værdi systemet giver til Olsker Cupcakes via. vores system:

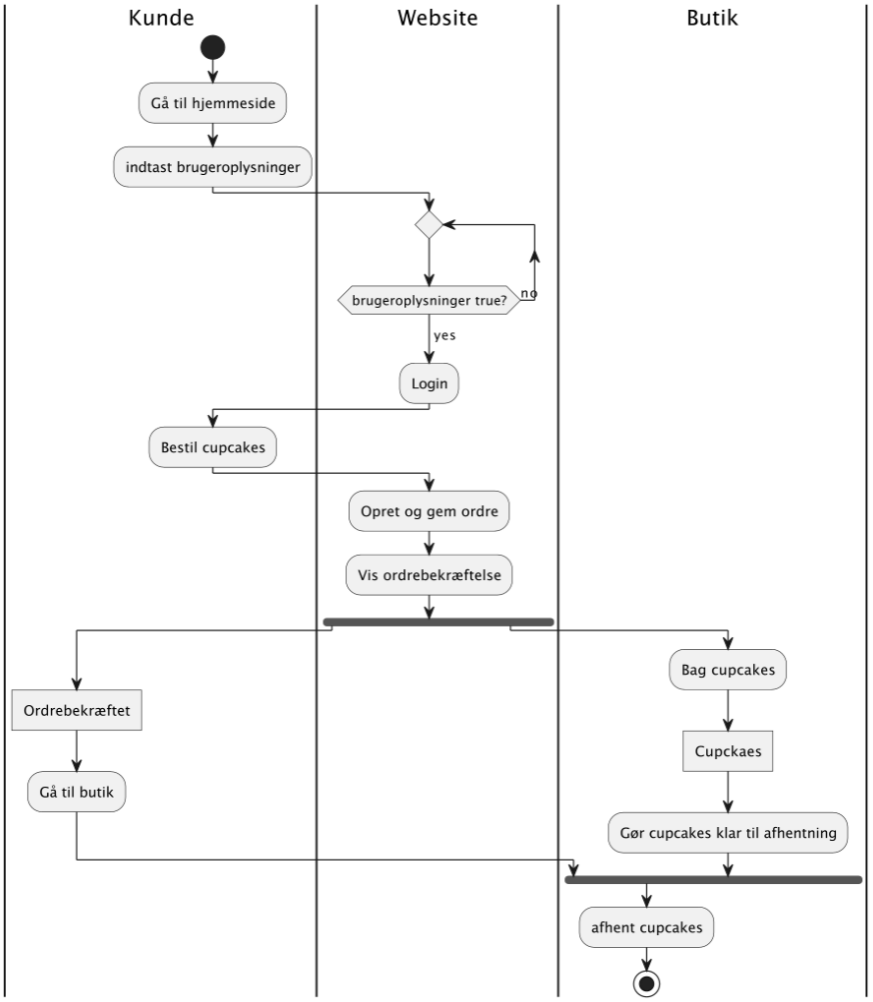
Systemet sigter mod at give kunderne en problemfri oplevelse, hvor de let kan bestille og tilpasse deres cupcakes efter deres præferencer. Ved at tillade brugerdefinerede bestillinger og let adgang til ordrehistorik vil systemet skabe en mere tilfredsstillende købsoplevelse.
Gennem systemet vil Olsker Cupcakes kunne effektivisere ordreprocessen og administrere kundekonti på en mere strømlinet måde. Det vil hjælpe med at reducere manuelle fejl og sikre nøjagtighed i bestillinger og betalinger.

Det forventes, at et mere tilgængeligt og brugervenligt online bestillingssystem vil tiltrække flere kunder og samtidig fastholde eksisterende kunder gennem en mere personlig service og brugerdefinerede tilbud.
Ved at samle data om kundeadfærd, præferencer og salgs mønstre vil systemet give Olsker Cupcakes mulighed for at trække værdifuld indsigt ud af disse data. Dette kan hjælpe dem med at træffe informerede forretningsbeslutninger og optimere deres tilbud i overensstemmelse hermed.

- 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 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

Kan også findes i projektet under models



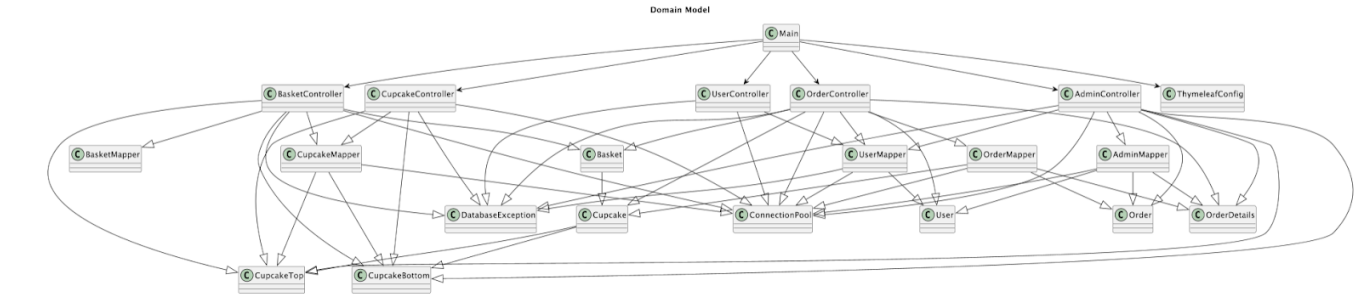
Her ses den ønskede proces ved en bestilling.

- Når brugeren kommer ind på siden, mødes de af en login funktion.
    - Websiden kontrollerer de indtastede oplysninger og sender dig videre, hvis de er korrekte.
- Brugeren vælger og bestiller sine cupcakes.
    - Websiden opretter en ordre og gemmer oplysninger i databasen. Sender ordrebekræftelse til brugeren. Samt ordre til butikken.
- Kunden tager sin ordrebekræftelse og tager til butikken. Butikken gør ordre klar til kunden.
- Kunde henter ordre i butikken.

Domæne model og ER diagram

Domænemodel:

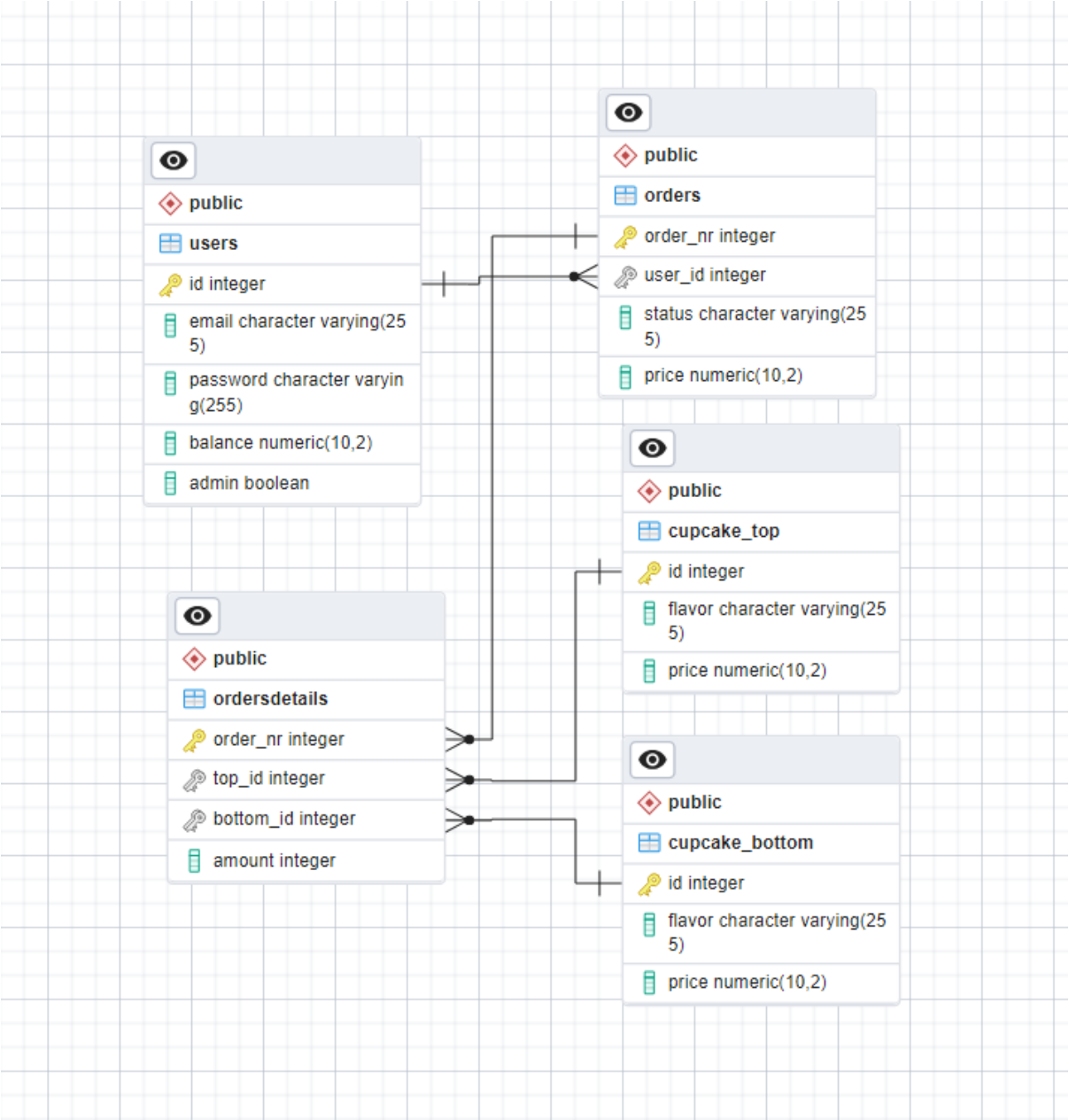
Kan også findes i projektet under models



Dette diagram viser hvordan de forskellige klasser i programmet relaterer og arbejder sammen.

- Main: Dette er den centrale klasse i programmet. Den har forbindelse til forskellige controller-klasser og konfigurations klassen for Thymeleaf .
- Controller-klasser:
    - UserController: Håndterer brugerrelaterede operationer som f.eks. login, oprettelse, opdatering og sletning af brugere.
    - OrderController: Ansvarlig for operationer relateret til ordrer, som inkluderer håndtering af kurve og brugerinformation.
    - CupcakeController: Styrer operationer vedrørende cupcakes, såsom tilføjelse, opdatering og sletning.
    - AdminController: Behandler operationer, der er specifikke for administrator brugere.
    - BasketController: Håndterer operationer vedrørende indkøbskurven, herunder tilføjelse, opdatering og sletning af elementer.
- Mapper-klasser:
    - AdminMapper: Mapper data fra programmet til databasen og omvendt for AdminController.
    - UserMapper: Mapper data fra programmet til databasen og omvendt for UserController.
    - CupcakeMapper: Mapper data fra programmet til databasen og omvendt for CupcakeController.
    - OrderMapper: Mapper data fra programmet til databasen og omvendt for Basket- og OrderController.
- Andre hjælpeklasser:
    - ConnectionPool: Håndterer forbindelsen til databasen ved at administrere en pulje af forbindelser.
    - DatabaseException: En klasse til at håndtere undtagelser, der kan opstå i forbindelse med databasen.
- Domæneklasser:
    - User: Repræsenterer en bruger i systemet.
    - Order: Representerer en ordre i systemet.
    - OrderDetails: Indeholder detaljerne om en ordre, såsom hvilke cupcakes der er inkluderet.
    - CupcakeTop og CupcakeBottom: Repræsenterer top- og bund delen af en cupcake.
    - Basket: Indeholder information om indkøbskurven og dens indhold.
    - Cupcake: Repræsenterer en komplet cupcake med både top- og bund del.
- ThymeleafConfig: Konfigurations Klasse for Thymeleaf-skabeloner.

ER diagram:



Dette diagram viser en oversigt over vores database.

Den opfylder de 3 normalformer ved at:

Ingen felter i kolonner gentager sig selv.

Der findes kun en primærnøgle i hver tabel.

Ingen felter uden for primærnøglen er afhængige af hinanden.

I tabellen orderdetails, benytter vi ikke et auto genereret id. Det gør vi ikke af da vi kan genbruge det fra tabellen orders, hvor order_nr er vores auto generede id.

Fremmednøgler (Foreign Keys):

- Fremmednøglerne i ordersdetails og orders refererer til primærnøglerne i cupcake_bottom, cupcake_top og users. Dette etablerer forbindelser mellem tabellerne, hvilket tillader relationer mellem dataene.

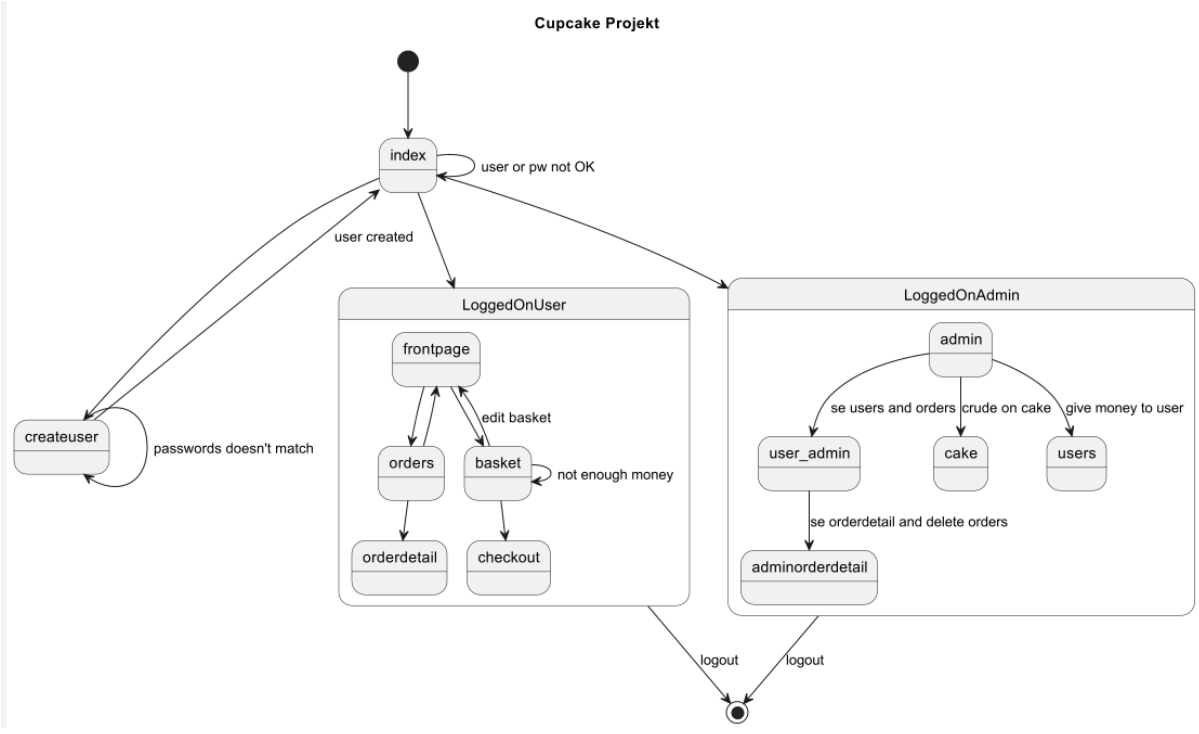
Constraints:

- NOT NULL constraints anvendes i alle vores felter.. Dette sikrer, at disse felter ikke kan være tomme. Det betyder også at vi opfylder 1. normalform

Entiteter og deres attributter:

- Der er identificeret entiteter som cupcake_bottom, cupcake_top, users (tidligere customer), ordersdetails og orders. Disse entiteter repræsenterer forskellige elementer i systemet og har tilknyttede attributter, f.eks. smag (flavor), pris (price), e-mailadresse (email), adgangskode (password), saldo (balance), osv.

Navigationsdiagram



Dette er et navigationsdiagram, der repræsenterer brugerinteraktioner.

- index: Dette er startpunktet for applikationen. Brugere kan logge ind, og hvis brugeroplysningerne ikke er korrekte, forbliver de på "index". De har også mulighed for at gå til "createuser" for at oprette en ny brugerkonto.

- createuser: Dette er stedet, hvor brugere kan oprette en ny konto. Hvis der opstår problemer, som f.eks. ikke-matchende adgangskoder, kan de blive på "createuser". Når en konto er oprettet, går de tilbage til "index" med beskeden "user created".
- LoggedOnAdmin: Dette er en separat tilstand for administratorbrugere. Her kan administratorer interagere med brugere, kager, ordreoplysninger osv. De kan navigere mellem forskellige muligher som "users", "cake", "user_admin", og "adminorderdetail".
- LoggedOnUser: Dette er en separat tilstand for almindelige brugere. Brugere kan udføre handlinger som at se forsiden, administrere kurven, se ordrer og gå til kassen.
- man kan logge ud af systemet når som helst og hvor som helt logud er ikke afhængig er nogen processor som skal fuldføres før man logger ud

I dette diagram kan brugere bevæge sig mellem forskellige tilstande afhængigt af deres handlinger og roller. Administratorer og almindelige brugere har forskellige interaktionsmuligheder i systemet.

Særlige forhold

Vi gemmer brugerens basket(kurv) informationer i en session inden de afgiver en ordre. Det har vi gjort fordi så kan kunden fortsætte shoppingen, rediger kurven og gennemfører købet senere uden at miste de valgte varer.

Primært håndterer vi exceptions med vores DatabaseExceptions klasse. Denne klasse bruges til at skabe mere skræddersyede exception-objekter. Når der opstår en fejl under database transaktioner, kaster vores kode DatabaseException-objekter. Disse objekter indeholder brugermeddelelser, der kan tilpasses, så brugerne kan forstå fejlen, f.eks. "Kunne ikke udføre forespørgsel: Ikke nok penge på kontoen".

Status på implementation

- Vi kunne havde lavet mere stylesheet men vi gik op i funktionalitet først så der var ikke så meget til at at fin pudse vores stylesheet
- Vi havde en ide om at kunne ændre status på ordrerne via admin panelet og det eneste det sådan set krævede var at vi skulle lave en dropdown menu med value også opdater vores sql string. Men da tiden var ude og det ikke var et krav i use casen, fravalgte vi dette, men vi havde det i tankerne.
- også mangler vi funktionen til at ændre bruger status til admin.

Proces

Vi planlagde at arbejde i et godt team miljø med god kommunikation, med regelmæssige “sprint-planlægninger” og hyppige statusmøder. Dette gjorde at vi hurtigt kunne tilpasse uforudsete ændringer hurtigt.

Vi testede primært, om vi kunne fungere kun med “sprint-planlægninger”. Det virkede meget godt til netop dette projekt, da vi fik klaret alle user cases. Dog overvejer vi, om vi kan bruge sprint-planlægningsmetoden til et større projekt. Næste gang vil vi sandsynligvis gøre det anderledes ved at kombinere vores “sprint-planlægningsmetode” med en mere langsigtet planlægning.

Vi har arbejdet ud fra vores Figma Mockup. Designet ligner ikke 1:1 med vores færdige produkt da diverse ændringer opstod.
Figma Mockup:
https://www.figma.com/file/6FdvQpj7grJBq8FnumMxIL/Cupcake-mockup?type=whiteboard&node-id=0%3A1&t=B4MpgqsoOTF0c63K-1