Cupcake

Af Benjamin, Mikkel, Philip, Valdemar (Klasse A, Team 1)

Video-demo: https://www.youtube.com/watch?v=_Cq7ookIqNQ

Github: **Mikkel-ao**, **b-rino**, **Phils1993**, **V2S2P**

Projekt og rapport: 3/31/2025 Kl: 10:45

Indholdsfortegnelse

Indledning **3**
Baggrund 3
Teknologi valg 4
Krav **4**
Virksomhedens visioner 4
User stories 4
Aktivitetsdiagrammer **6**
ERD – Entity Relationship Diagram **8**
Overvejelser i forbindelse med databasen 8
Domænemodel **10**
Kommentarer til domænemodellen 10
Navigationsdiagram **11**
Formålet med UML-Tilstandsdiagram 11
Fordele ved UML-Tilstandsdiagram 13
Særlige forhold **14**
Session 14
Exceptions 14
Validering af brugerinput og sikkerhed i forbindelse med login 14
Brugertyper 15
Status på implantation **15**
Proces **17**
Gruppens udgangspunkt 17
Cupcake arbejdsplan 17
Evaluering af forløbet 18

Indledning

”Cupcake” er et full-stack web-development projekt, og omfatter derfor både frontend og backend udvikling. Målet er at integrere disse elementer, for at skabe en selvstændig hjemmeside der fungerer som en "rigtig" hjemmeside - altså en hjemmeside der kan guide kunden gennem login, køb af cupcakes, log out, samt andre forskellige sideelementer, såsom en ordreliste og kvittering.

Målgruppen for denne tekniske rapport er derfor fagfæller, der allerede besidder en vis forståelse herfor.

Baggrund

Olskers Cupcakes er en økologisk butik på Bornholm, der har fundet den perfekte opskrift. De har besluttet at lancere en webshop for at udvide deres forretning fra den fysiske butik. Kunderne vil kunne bestille online og hente deres varer i butikken senere.

Olskers Cupcakes ønsker, at kunderne kan oprette en bruger og logge ind på deres hjemmeside. Kunderne skal kunne vælge cupcakes, både bund og top, samt angive antallet af cupcakes der ønskes. Der skal også være mulighed for at fjerne en ordrelinje, så kunderne kan ændre deres ordre. Kunderne skal kunne se deres tidligere bestillinger, mens en administrator skal have adgang til både at se alle kunder og ordrer i systemet. Administratoren skal også kunne fjerne en ordre fra systemet, for at sikre at ugyldige ordrer ikke skaber forstyrrelser.

Teknologi valg

I Cupcake-projektet er der blevet brugt følgende teknoloiger:

1. Java - IntelliJ 2024.3.5 (corretto-17 Oracle OpenJDK version-23)
2. JDBC (HikariCP)
3. PostgreSQL
4. Javalin
5. Thymeleaf
6. HTML
7. CSS

Krav

Virksomhedens visioner

Olskers Cupcakes har hidtil været begrænset til at eksistere som et fysisk sted, hvilket har reduceret deres potentiale kundebase. En online butik vil åbne op for nye muligheder og give dem en billig og nem adgang til at nå flere kunder. Det er netop derfor, at de har valgt at samarbejde med os om at udvikle en hjemmeside. Målet er at etablere Olskers Cupcakes som et velkendt navn i Danmark og potentielt også på internationalt plan.

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 Postgres, så en kunde kan betale for sine ordrer.

US-4: Som kunde kan jeg se mine valgte ordrelinier 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).

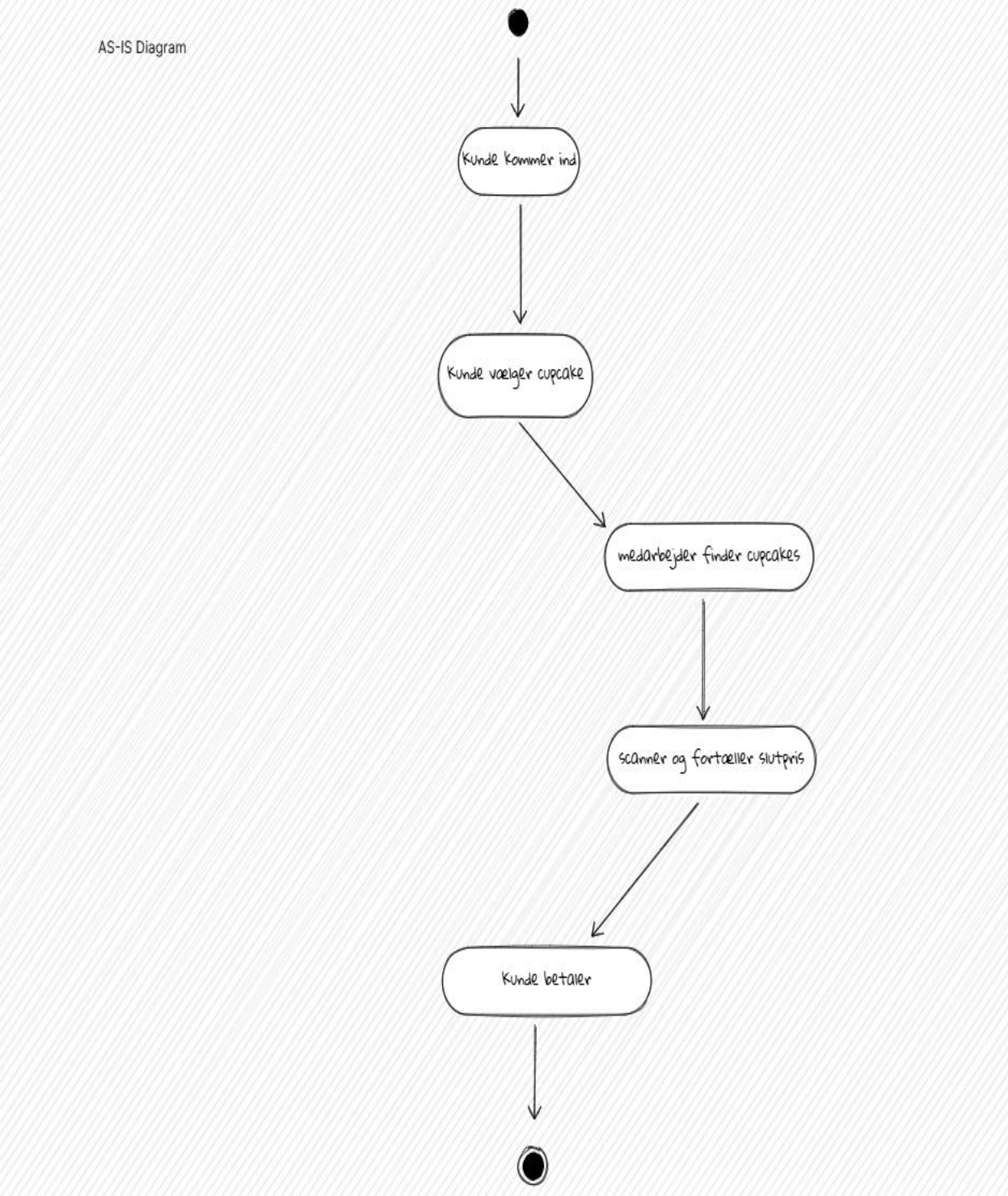
US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

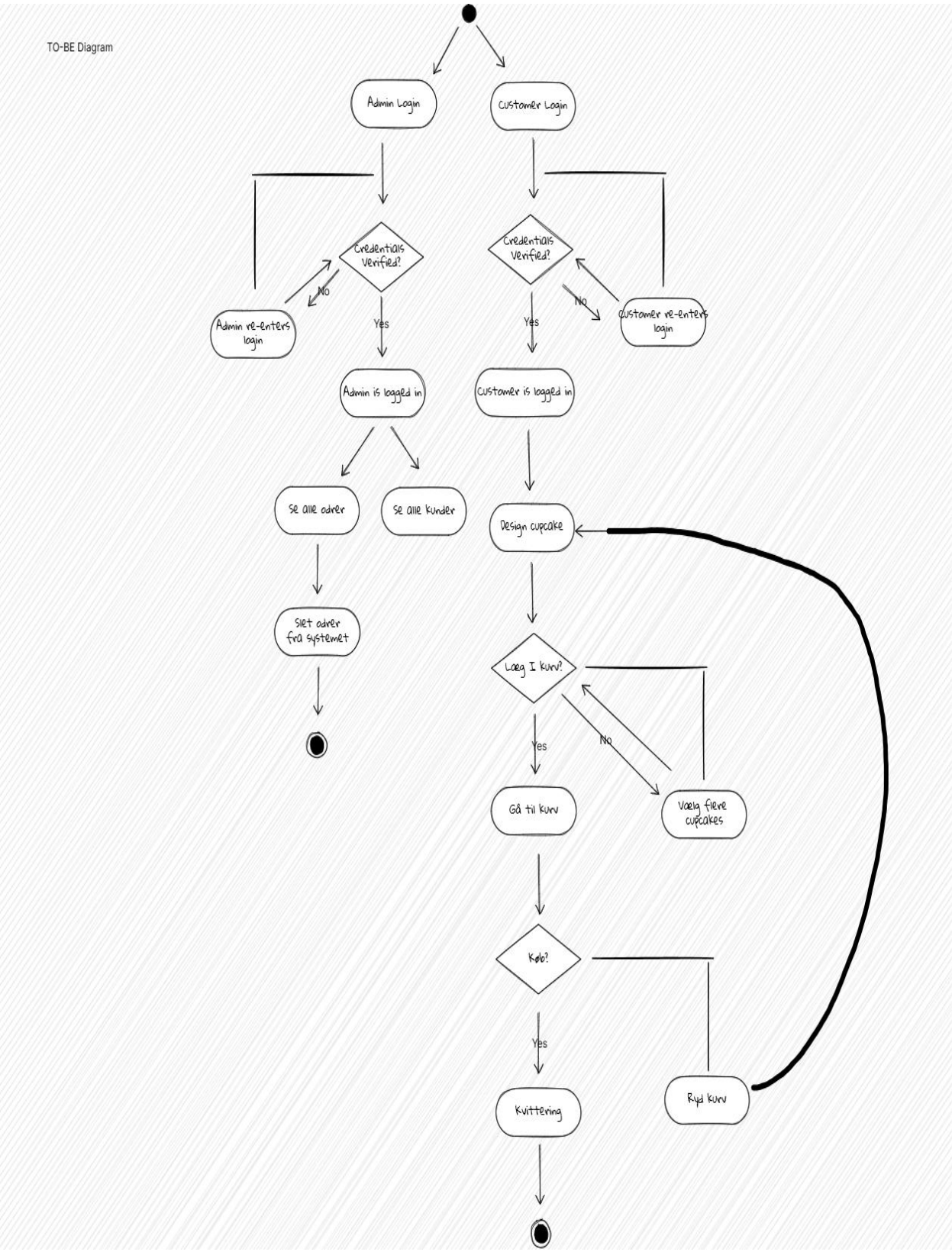
US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

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

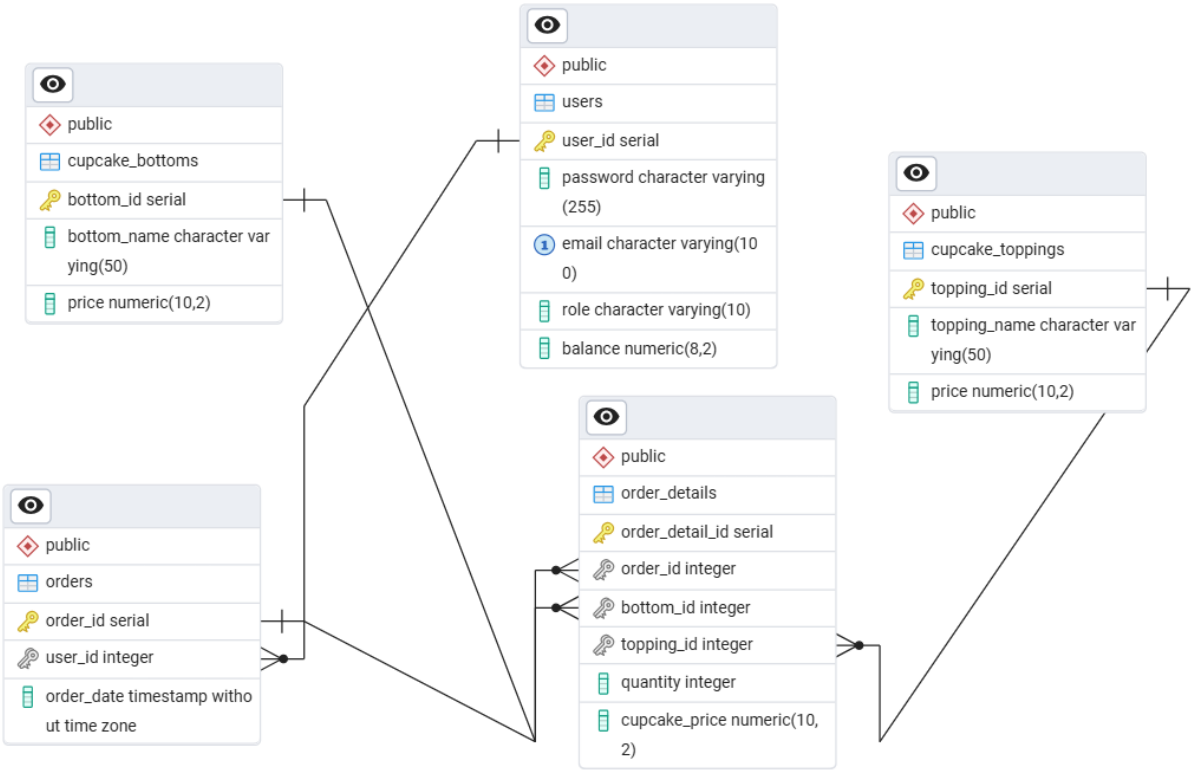
US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde udgyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Aktivitetsdiagrammer




TO-BE Diagram
Admin Login
Customer Login
Credentials Verified?
No
Yes
Admin re-enters login
Credentials Verified?
No
Yes
Customer re-enters login
Admin is logged in
Customer is logged in
Se alle ordrer
Se alle kunder
Slet ordrer fra systemet
Design cupcake
Læg I kurv?
Yes
No
Gå til kurv
Vælg flere cupcakes
Køb?
Yes
Kvittering
Ryd kurv

ERD – Entity Relationship Diagram



Figur 1

Overvejelser i forbindelse med databasen

Der er oprettet 5 tabeller, som tilsammen indeholder data om Olskers kunder, ordrer og cupcakes.

I tabellerne er der auto-genereret id'er. Dette er med til at sørge for unikke id'er. Det er vores primære nøgler, som har auto-genereret nøgler: Buttom_id, user_id, topping_id, order_id, order_details_id. Dette sikrer, at data om ordrer og kunder gemmes korrekt uden overlap eller datakonflikter.

Projektet kræver at der findes en administrator, og vi har derfor valgt at tilføje en "role" kolonne i users tabellen, som default er "customer". Det er kun ejerne/udviklingerne, som kan ændre rollen til "admin" i databasen.

Ingen af tabellerne i vores database har mange-til-mange relationer. Relationerne i databasen består primært af en-til-mange relationer, som fx:

- user_id i tabellen users har en-til-mange relation til user_id i tabellen orders, da én bruger kan oprette flere ordrer.

Vi har forsøgt at have en simpel og let overskuelig database, for at undgå konflikter.

Navngivningen af tabeller og attributter følger en stringent tilgang for at sikre læsbarhed og konsistens. Fx er attributten email valgt frem for username som brugeridentifikation, baseret på kravene i projektet. Derudover gemmes brugerens adgangskode som et hashed password for at sikre sikkerhed og dataintegritet.

Vi overvejede om email skulle være primær nøgle (naturlig nøgle), men valgte i stedet en syntetisk nøgle, da den også bruges som fremmednøgle, og vi derfor fandt det nemmere.

Databasen opfylder første og anden normalform, men afviger fra tredje normalform. Dette skyldes at tabellen order_details er afhængig af tre primære nøgler for at beregne cupcake_price. Denne struktur tillader prisændringer i tabellerne cupcake_bottoms og cupcake_toppings, uden at historiske ordredata påvirkes. Dette design prioriterer fleksibilitet over striks normalisering.

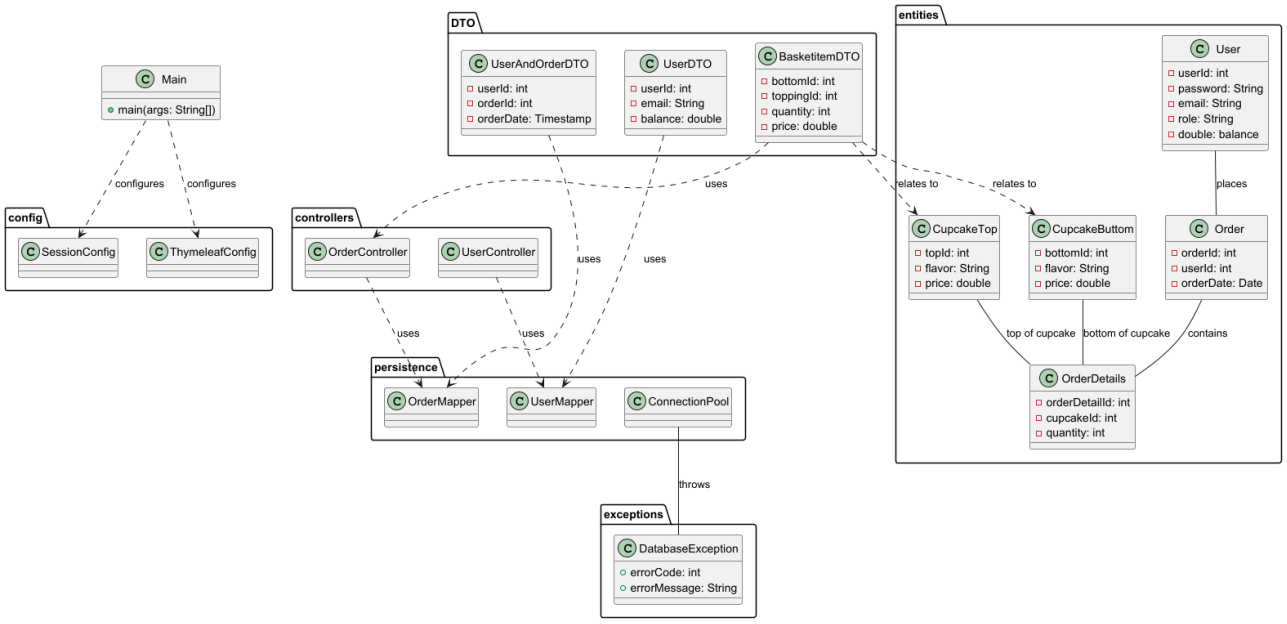
Implementerede constraints inkluderer:

- Primære nøgler: Sikrer entydig identifikation af rækker.
- Not NULL: Garanterer, at kritiske kolonner altid indeholder en værdi.
- Unikke nøgler: F.eks. e-mailadressen i tabellen users, for at hindre gentagelser.
- Default værdier: Brugeren får automatisk en startbalance på 200, som alternativ til en regulær betalingsgateway.

Mulige forbedringer:

- Implementer begrænsninger på price og quantity, så f.eks.. price > 0 og quantity > 0.
- Udvid databasen med leveringstabeller, f.eks.. en shipping_order tabel med attributter som packed (Boolean) og carrier (f.eks. GLS).

Domænemodel



Figur 2

Kommentarer til domænemodellen

Vi har lavet pakker og fulgt en simpel og overskuelig opsætning af Java programmet. Hvis vi ikke havde lavet denne opsætning, havde programmet været uoverskueligt og ubalanceret.

Ud fra vores efteranalyse af Java-koden, har vi opdaget at klasserne ”Orders”, ”CupcakeTop” og ”CupcakeBottom” ikke bliver brugt. Dette er fordi vi i stedet bruger vores DTO’er (”Data Transfer Objects”) til at objektivisere det data, som vi henter fra databasen. Dette har været nødvendigt, da vi har haft brug for at behandle data fra forskellige entities på én gang.

Selvom klasserne kan fjernes, er de efterladt som dokumentation for selve designprocessen.

Navigationsdiagram

Formålet med UML-Tilstandsdiagram

Formålet med diagrammet er at modellere navigationsflowet i det system, vi har lavet. Det skal give en klar visualisering af de forskellige tilstande, som en bruger kan befinde sig i, samt de overgange, der sker, når brugeren foretager en handling. Denne tilgang bør resultere i et mere struktureret og forståeligt design af programmet.

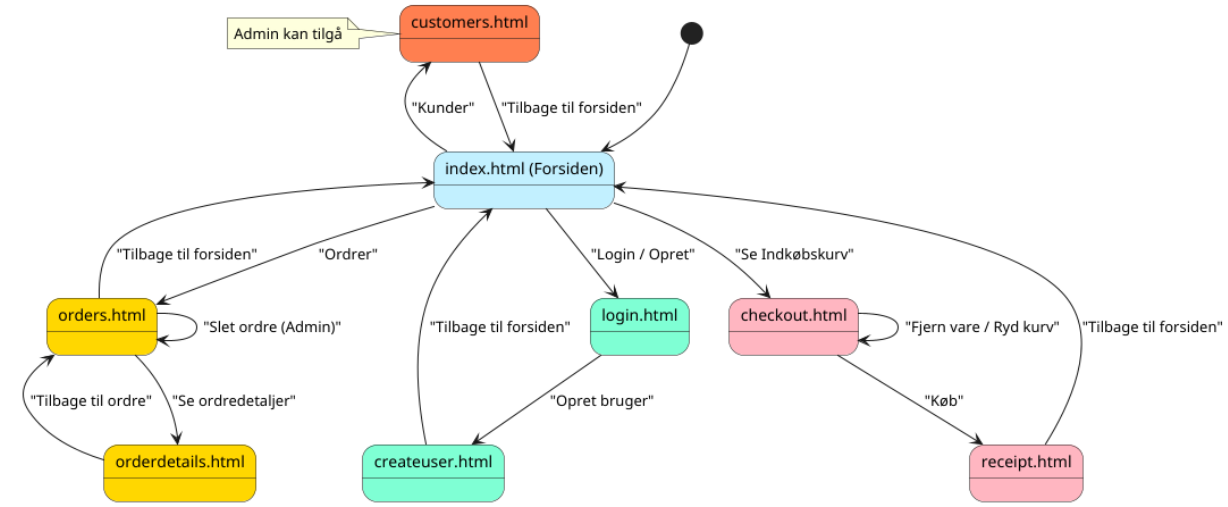
Nøgleelementer i vores tilstandsdiagrammer:

1. **Tilstande:**
   Hver webside, såsom "index.html" og "login.html", er repræsenteret som tilstande. Den **"Initial state"** bliver repræsenteret i form af en fyldt sort cirkel og er den første forbindelse til "index.html", som fungerer som vores forside.

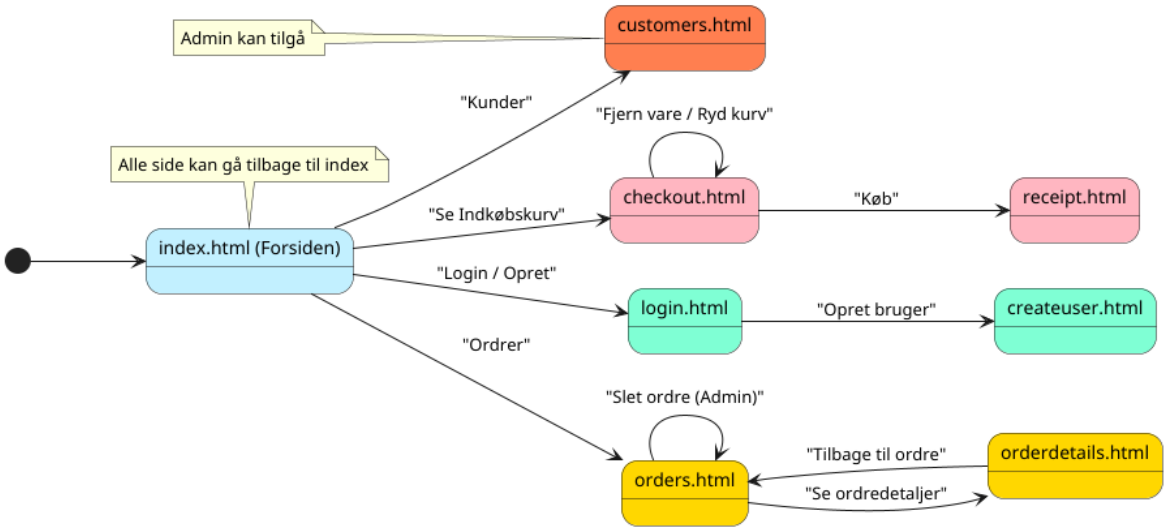
2. **Overgange:**
   Pilesymbolerne viser, hvordan brugeren navigerer mellem siderne, f.eks. fra "index.html" til "login.html" ved at klikke på "Login / Opret". Det er især i oversigtsdiagrammerne, hvor disse overgange giver en klar forståelse af navigeringen for brugeren.

3. **Betingelser og handlinger:**
   Overgange kan afhænge af betingelser. I "LoginProces" går brugeren kun videre til "LoginSucces", hvis legitimationsoplysningerne er korrekte.

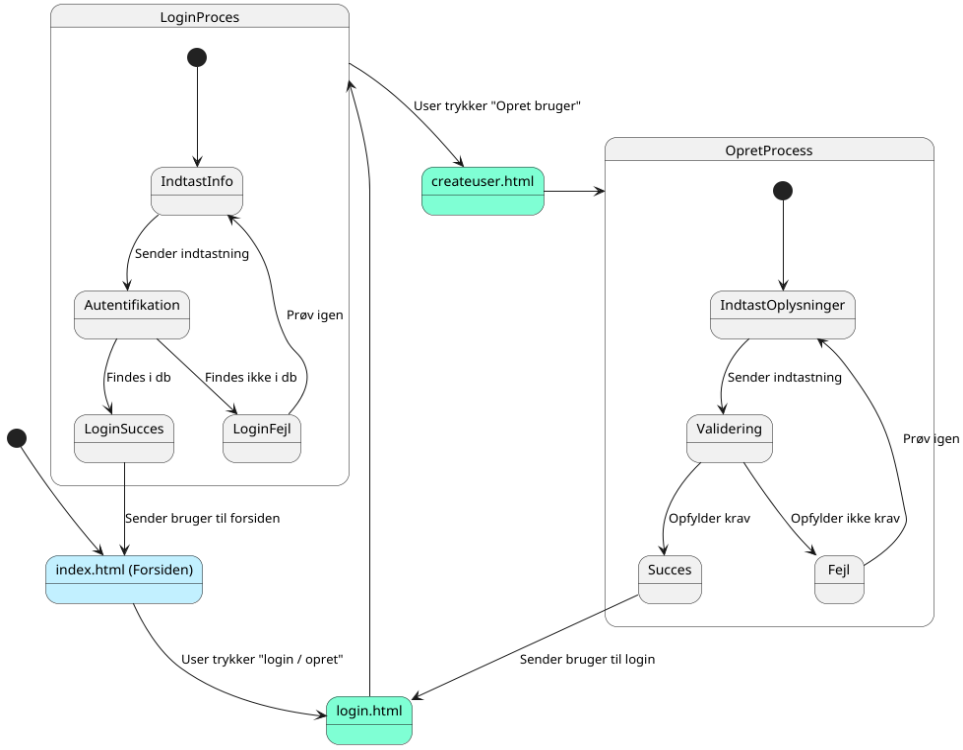
4. **Begrænsede tilstande:**
   Nogle tilstande, som "Customers", er kun tilgængelige for administratorer, hvilket er angivet i oversigtsdiagrammerne. Det samme gælder i "Orders", hvor administratorer har muligheden for at slette en ordre.



Figur 1: Oversigtsdiagram - Inklusiv overgange til index



Figur 2: Oversigtsdiagram - Implementation hvor index altid kan tilgås



Figur 3: Tilstandsdiagram - Login / Opret bruger processen

Fordele ved UML-Tilstandsdiagram

Eftersom tilstandsdiagrammet er blevet lavet senere i projektets levetid, må vi erkende, at flere af fordelene først opleves på et sent tidspunkt i arbejdsprocessen. Gevinsten ved at lave diagrammerne har været den klarhed, som en visuel repræsentation kan give. Det visuelle element har været med til at identificere potentielle problemer, og hvis der havde været tid til det, kunne det også have været med til at forbedre brugeroplevelsen. UML-diagrammerne kan nemlig hjælpe med at danne et klart og vedligeholdelsesvenligt design, som gør det lettere at forstå navigationen gennem de forskellige HTML-sider og dermed bidrage til at forbedre brugeroplevelsen.

Særlige forhold

Session

Ved succesfuldt log ind gemmes følgende oplysninger på brugeren i sessionen: **ID**, **e-mail** og **rolle**.

ID: Det sikrer at vi udfører de valgte operationer på den rigtige bruger (opdaterer balancen, viser de rigtige ordrer osv.). Derudover kræver det også at der findes et bruger ID i sessionen, for at man kan tilgå ordrer siden, samt foretage et køb.

E-mail: For at vise brugeren deres e-mail på hver side.

Rolle: Bruges i flere tilfælde til at begrænse eller udvide hjemmesiden funktionalitet, alt efter om det er en ”admin” eller ”customer” der er logget ind.

Disse gemte brugerdata bliver ryddet fra sessionen, idet en bruger logger ud.

Exceptions

Igennem hele projektet er der fokus på fejlhåndtering, herunder et skærpet fokus de steder, hvor der er en forhøjet risiko for fejl - hvilket der som udgangspunkt er i alle metoder, der har med kommunikation med databasen at gøre.

Ved brug af ”exceptions” sørges der for, at brugeren aldrig ”sidder fast” på hjemmesiden, men bliver sendt til en passende side, med en passende fejlmeddelelse, skulle en fejl opstå.

Validering af brugerinput og sikkerhed i forbindelse med login

Validering: Det er kun ved oprettelse af bruger og login, at en bruger har ”frihed” til selv at udfylde felter.
Her sikrer koden at brugerinput (e-mail og password) bliver valideret, inden det behandles, hvilket hjælper med at forhindre SQL-injektion angreb.
Derudover sikres at den indtastede e-mail indeholder et normalt e-mailformat med både ”@” efterfulgt af et ”.” igennem HTML.

Fejlhåndtering: Hvis et login fejler, bliver brugeren præsenteret for en generel fejlmeddelelse, hvilket forhindrer en potentiel angriber i at få at vide, om det er e-mail eller

password der er forkert.

Brug af brugersession: Efter vellykket login gemmes brugerens ID, e-mail og rolle i sessionen, hvilket gør det nemt at genkende brugeren, uden at password også ligger i sessionen. Dette er for at styrke sikkerheden, og samtidig en nem måde at styre hvad den enkelte bruger har adgang til på hjemmesiden.

Brug af hashed passwords: I databasen gemmes brugerens password som hashed data, hvilket sikrer at brugernes passwords ikke kan aflæses direkte, selv hvis databasen skulle blive kompromitteret. Dette øger sikkerheden omkring brugerens oplysninger.

Brugertyper

Som beskrevet tidligere findes to typer roller i vores database: ”admin” og ”customer”. Ideen med at have disse to, er at en ”customer” KUN skal kunne tilgå data vedrørende sig selv, hvor en ”admin” har fuld kontrol over systemet. Nøgleforskellene er at en ”admin” kan se hele se kunder, ordrehistorikken, samt slette ordrer. En ”customer” kan kun se sine egne tidligere ordrer, og har derudover ikke mulighed for at slette disse.

Status på implantation

Overordnet set er projektet kommet fint i mål, men der er altid plads til forbedringer, såsom:

Begrænsning på brugerinput ved oprettelse af bruger:

Ved oprettelse af en bruger findes der i den nuværende kode ingen begrænsninger på valg af password. Her kunne man eventuelt sørge for at koden har en minimumslængde, samt eventuelt flere regler som skal overholdes, for at få sit password godkendt.

Login/brugersession:

Ved login gemmes brugerens ID som to forskellige attributter i den samme session. Dette er unødvendigt, da én gang er nok. Vi har dog valgt at beholde dem begge, indtil vi er sikre på at vi kun bruger den ene igennem hele koden.

Test:
Vi nåede ikke i mål med at teste vores system med hverken **unit test** eller **integrations test**. Dette ville være en opgave af høj prioritet ved videre arbejde med projektet, da det både er en sikkerhed for kunden (product owner), samt en måde at ”holde sin ryg fri” på som udvikler!

Refaktorering:

Der findes metoder rundt omkring i koden, som med fordel kan deles op i flere mindre metoder, så man sørger for at én metode ikke står for flere opgaver, samt bliver for lang og uoverskuelig.

Synlig balance:

Det er indtil videre kun ”admin” rollen, der kan se sin egen (og alle andres) balance. Det ville være oplagt at sørge for, at alle brugere kan se egen balance på forsiden – evt. ved siden af deres e-mail!

Besked til bruger:

Man får på nuværende tidspunkt ikke en besked, når man har tilføjet en vare til indkøbskurven. Denne tilføjelse ville øge brugervenligheden, da man godt kan blive i tvivl om man fik trykket på ”Læg i kurv”.

Proces

Gruppens udgangspunkt

Vi har været i samme gruppe siden starten af uddannelsen, og har i alle tidligere projekter mødtes fysisk på skolen, hvor vi har lavet hele projektet sammen. Dette havde både fordele og ulemper:
Fordelene var blandt andet, at vi ikke oplevede mange ”merge-konflikter”, og at alle vidste hvad der skete rundt omkring i koden.
Ulemperne lød især på mangel på effektivitet, da man i mange tilfælde sad mange om én pc.

MEN i dette projekt følte vi os godt klædt på til at prøve at gå en anden retning. Dette skyldtes især vores nyligt erhvervet viden om ”Pull Requests”, samt tildelingen af roller internt i teamet, som gjorde det nemt at forstå ansvarsområderne.

Cupcake arbejdsplan

UGE 1
Vi mødtes fysisk på skolen første dag, hvor vi skulle lave database og komme i gang med projektet.
Resten af uge 1 var det valgfrit om man ville arbejde hjemmefra eller komme på skolen – dog mødtes vi hver dag klokken 9 til et statusmøde på Discord, og man skulle derudover være til rådighed på Discord i resten af den ”normale arbejdstid”.

Vi havde derudover en aftale om at hvis det var for besværligt at arbejde ”remote”, ville vi hurtigt sadle om og mødes på skolen hver dag som vi plejede.

UGE 2
Vi mødtes fysisk på skolen mandag, hvor vi skulle lave det sidste styling og komme i gang med diagrammer og rapport.
Resten af uge 2 var det valgfrit om man ville arbejde hjemmefra eller komme på skolen – dog mødtes vi hver dag klokken 9 til et statusmøde på Discord, og man skulle derudover være til rådighed på Discord i resten af den ”normale arbejdstid”.

Igennem hele projektet havde vi en aftale om at det kun var ”Tech Lead” der mergede, og alle andre skulle lave ”Pull Requests”.

Evaluering af forløbet

Al vores tvivl om hvorvidt det var for besværligt at holde styr på det hele, blev manet til jorden allerede på første ”hjemme-arbejds-dag”. Her oplevede vi nemlig, at det var ganske optimalt at arbejde sammen over Discord, samt at effektiviteten var meget højere end hvad vi var vant til fra tidligere. Vi fortsatte derfor på denne måde resten af projektet.

De roller vi uddelegerede i starten af projektet, gjorde det nemt at finde ud af hvem man skulle gå til, hvis man vil have afklaret et spørgsmål. Især ”tech lead” spillede en nøglerolle, da det at have én person som har ansvaret for ”Pull Requests” og at få samlet koden, frigav en masse tid og energi til de resterende, som i sidste ende resulterede i en forhøjet produktivitet.

Den eneste umiddelbare udfordring var, at hvert enkelt gruppemedlem bliver nødt til at sætte sig godt ind i den kode de øvrige har lavet (og eventuel kommer med spørgsmål og/eller konstruktiv feedback), da man jo alle står til ansvar for produktet, samt skal kunne redegøre for dette.

Alt i alt var ”Cupcake” et forløb med rigtig mange succesoplevelser. Vi kom godt i dybden med ”webstack”, men vigtigst af alt lærte vi rigtig mange nye ting i arbejdet som gruppe. Brugen af roller i teamet og ”Pull Requests” gjorde at det føltes, som en meget mere professionel og optimal måde at løse en opgave på, end hvad vi tidligere har været vant til.

Dette vil vi helt klart tage med videre i de kommende projekter!