# Icon Description automatically generated with medium confidence

**Deltagere**:

Navn: Angeliza Medina Quintana

E-mail: [cph-aq37@cphbusiness.dk](mailto:cph-aq37@cphbusiness.dk)

GitHub: <https://github.com/Angeliza-Medina>

Navn: Jean Andre Philippe Agerholm Suenson

E-mail: [cph-js507@cphbusiness.dk](mailto:cph-js507@cphbusiness.dk)

GitHub: <https://github.com/Philippe16>

**Projektperiode**:

08/11/2021 – 19/11/2021

Indholdsfortegnelse

[1](#_Toc88209815)

[Indledning 2](#_Toc88209816)

[Teknologi 3](#_Toc88209817)

[Projektkrav 4](#_Toc88209818)

[Virksomhedens forventning til projektet 4](#_Toc88209819)

[Usecases 4](#_Toc88209820)

[Projekt-diagrammer 6](#_Toc88209821)

[Aktivitetsdiagram 6](#_Toc88209822)

[Domæne model 9](#_Toc88209823)

[EER-diagram 9](#_Toc88209824)

[Navigationsdiagram 11](#_Toc88209825)

[Særligt benyttede forhold i projektet 13](#_Toc88209826)

[Liste af værdier som gemmes på session 13](#_Toc88209827)

[Håndtering af Exceptions 13](#_Toc88209828)

[Implementerings-status 14](#_Toc88209829)

[Arbejdsproces 15](#_Toc88209830)

# Indledning

Udfra følgende fiktive scenarie, har vi i den seneste uges tid arbejdet på at udvikle en java-baseret hjemmeside til en fiktiv kunde, *Olsker Cupcakes*.

*Olsker Cupcakes* er en ny virksomhed i form af en cupcake-shop beliggende på Olsker, Bornholm.

Den nye virksomhed, som ønsker sig en hjemmeside, har inviteret nogle hipster-udviklere fra København med det formål at få aftalt, hvad deres kommende hjemmeside skal indeholde. Derudover ønsker *Olsker Cupcakes* at give udviklerne et indgående indblik i, hvordan selve cupcake-shoppen er, da de ønsker et tilsvarende visuelt udtryk samt stemning for deres hjemmeside.

Efter besøget, står vi med en løs mockup til en enkelt side, samt et par krav i form af usecases.

Yderligere detaljer ang. hjemmesidens funktionalitet, design samt forslag til forbedringer, er efterfølgende op til os at indsamle/bringe op.

# Teknologi

Til denne opgave har vi anvendt følgende teknologi:

Håndtering af kode: Intellij (Ultimate Edition) v. 2021.2.2

Udarbejdelse af en statisk version af projektet, samt tegning af domæne model: Visual Studio Code v. 1.62.2

JDBC: Tomcat v. 9.0.54

Database: MySQL (Community Edition) v. 8.0.26

Tegning af diagrammer: Draw.io – (app.diagrams.net)

Design og mockup: Adobe XD (Community Edition) v. 45.0.62.17

Billede redigering: ArtRage Studio Pro v. 3.5.12

Planlægning, håndtering og fordeling af opgaver: Trello - (Trello.com)

Tekstprogram til rapportskrivning: Microsoft Word v. 2110

Uploading af projekt & versionskontrol: Github – (Github.com)

Kommunikation imellem gruppemedlemmer: Discord v. 9114 & Facebook Messenger 2021

# Projektkrav

## Virksomhedens forventning til projektet

Virksomheden ser gerne at de første 6 usecases (som kan forefindes herunder) bliver implementeret.

Efter endt forløb, står virksomheden gerne med en visuel appellerende hjemmeside, som samtidig også er funktionel.

Formålet med hjemmesiden er at give folk muligheden for at stifte bekendtskab til *Olsker Cupcakes*, samt give deres kunder muligheden for at bestille cupcakes online, sådan at deres bestilling står klar til afhentning, når de efterfølgende kommer forbi.

## Usecases

Herunder er de oprindelige 9 usecases som vi fik udleveret:

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
2. Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en ordre
3. Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer
4. Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris
5. Som kunde eller administrator kan jeg logge på systemet med e-mail og kodeord. Når jeg er logget på, skal jeg kunne min e-mail på hver side (evt. i topmenuen, som vist på mockup’en)
6. Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt
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
8. Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre
9. Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt

# Projekt-diagrammer

## Aktivitetsdiagram

Aktivitetsdiagrammer viser forløbet for en aktivitet fra start til slut.

Der er 2 typer af aktivitetsdiagrammer, As-is og To-be.

As-is: Giver et billede på, hvordan en aktivitet bliver udført her og nu.

To-be: Er en model for, hvordan vi gerne vil have aktiviteten til at være efter at have implementeret nogle ændringer.

Starten for aktiviteten er markeret med en sort cirkel, mens slutningen er markeret med en sort cirkel med et hvidt omrids.

Diagrammet er opdelt i såkaldte ”svømmebaner”, hvoraf aktivitetens aktører fremgår i hver deres bane.

En oval repræsenterer en handling, som bliver udført af den aktør, hvis bane ovalen befinder sig på.

En rombe forgrener handlingsforløbet i 2 eller flere grene. Dette sker, når aktørens efterfølgende handling er baseret på et bestemt udfald.

Et rektangel kaldes for et ”artefakt ”. Artefakter repræsenterer objekter som bliver flyttet rundt og ændret på/tilføjet til mellem de forskellige aktører.

Diagram

Description automatically generated

Foroven ses et aktivitetsdiagram, som viser hele forløbet fra at en kunde kommer ind til *Olsker Cupcakes’* fysiske shop, til at kunden forlader shoppen med en æske cupcakes.

I dette diagram har vi 2 aktører; der er kunden, som køber cupcakes og en butiksassistent, som assisterer kunden med sit køb.

Modellen er forholdsvis simpel og er i princippet hele vejen igennem en interaktion mellem kunden og butiksassistenten.

Diagram, schematic

Description automatically generated

Aktivitetsdiagrammet foroven viser til gengæld, hvordan en kunde fremover får mulighed for at bestille sine cupcakes online via *Olsker Cupcakes’* nye hjemmeside.

Sammenlignet med det forrige diagram, ses det tydeligt at der nu er kommet mange flere trin til, før at kunden kan bestille sine cupcakes. Til gengæld er man som kunde ikke længere afhængig af at skulle blive betjent af en butiksassistent, da denne aktør nærmest er sat helt ud af spil i denne model.

I stedet opstår der her en tæt interaktion mellem kunden og en ny aktør, nemlig systemet/hjemmesiden.

Denne model kræver at kunden har et operativsystem samt en netværksforbindelse.

## Domæne model

Et domæne model er et visuelt billede på, hvordan et system er bygget sammen.

Den er med til at danne et overblik over hvilke overordnet klasser projektet indeholder og deres relationer til hinanden. Hvad der fx implementerer og hvad der extender, kan ses udfra den type uml pil der er blevet brugt til at forbinde to klasser.

Et domæne model kan laves forholdsvis tidligt i et forløb, da den ikke indeholder særlig mange detaljer.

Med tiden vil domæne modellen selvfølgelig ændre sig, da der som regel tilføjes og fjernes klasser og attributter i takt med at systemet vokser og tager form.

Diagram

Description automatically generated

Vores domæne model er endt med at se således ud.

Et større billede kan findes i vores projekt mappe Report/Diagrams/domain\_model/Domain Model.png

Modellen er blevet genereret i Visual Studio Code ved hjælp af et plantUML plugin.

Med et program som har så mange klasser og relationer som dette, er det rigtig nyttigt at have et domæne model ved hånden.

## EER-diagram

EER-diagram står for ”enhanced entity-relationship diagram”.

Den giver et samlet billede af, hvordan éns database er designet, hvilke tabeller databasen består af, tabellernes forhold til hinanden, tabellernes attributter og dertil deres datatyper.

Diagram

Description automatically generated

Foroven ses et diagram over vores database. Diagrammet har vi genereret ved hjælp af MySQL Workbench.

Det er en MySQL database som er opdelt i 7 tabeller, hvoraf de alle på en eller anden måde er forbundet.

Alle relationer, som vi har benyttet i vores database, er af type ”én-til-mange” relation.

Et konkret eksempel hertil, kunne være at en række fra ”cupcaketoppings” kan være refereret til af flere ”orderitems”:

”Flere elementer i en ordre (cupcakes) kan referere til den samme chokolade topping”.

Ved at opdele vores data i mange enheder (tabeller) og efterfølgende forbinde dem med ”en-til-mange” relationer, har vi forsøgt så vidt muligt at undgå redundant data, samt opnå mere konsistens. Tabellen ”statuses” er et godt eksempel på netop afvigelse en masse redundant data samt vedligeholdelse af konsistens.

I vores ”statuses” tabel findes der kun tre status typer, ”pending”, ”completed” og ”cancelled”.

Der kan ikke udefra blive tilføjet mere data til denne tabel.

Da alle ordre har en status, ville mange statustyper gå igen, hvilket vi har undgået ved i stedet at lave en reference til statustypen fra ”orders” tabellen. Vi har derved samtidig opnået sikring af konsistens, da man på den måde ikke kan skrive en anden type af status, end dem som er defineret i statustabellen.

Vores database er designet ud fra den 3. normal form.

## Navigationsdiagram

Diagram

Description automatically generated

Vores hjemmeside består af i alt 10 jsp sider.

Vi har en index, home, about, shop, contact, shoppingBasket, signUp, signIn, account og error side.

Som med alle andre hjemmesider, er index siden den første, man som bruger, bliver sent hen til. I vores tilfælde har vi valgt at lave vores index til en ”splash screen”, som man automatisk bliver omdirigeret fra efter 1,8 sekund.

Efter at man er blevet omdirigeret fra index, bliver man sendt over til vores homepage. Derfra og fra alle andre sider (med undtagelse af signUp, signIn og account) kan man ved hjælp af navigationsbaren tilgå index, home, about, shop, contact, shoppingBasket, signUp, signIn og account.

I navigationsbaren har vi brugt et avatar ikon, som fører brugeren hen til signIn siden, hvis brugeren ikke i forvejen er logget ind. I det tilfælde at brugeren allerede er logget ind, fører ikonet brugeren over til account siden.

For at komme over til signUp siden, er man nødt til først at klikke ind på signIn og derfra klikke på et link.

Fra både signIn og signUp, har vi en knap, som fører brugeren tilbage til home.

Den sidste side, vi har, er en error side. Den kan udelukkende tilgås, hvis vores system kaster alle andre exceptions end en UserException.

# Særligt benyttede forhold i projektet

## Liste af værdier som gemmes på session

Fra at en client opretter forbindelse til vores hjemmeside, gemmes følgende data i sessionen:

* ”shoppingBasketItems”, <OrderItem>
* ”shoppingBasketTotalPrice”, double
* ”shoppingBasketTotalCupcakes”, int
* ”shoppingBasketItemCount”, int
* ”shoppingBagMsg”, String:
* ”user", User
* "orderConfirmed", Boolean
* ”customerOrders”, Order
* ”cupcakeShop”, CupcakeShop
* ”cupcakeToppings”, <CupcakeTopping>
* ”cupcakeFlavors”, <CupcakeFlavor>
* ”role”, String
* ”email”, String
* -”error”, String

## Håndtering af Exceptions

Måden hvorpå vi håndterer execptions er ved at benytte os af UserException.

UserException er en klasse, som medfulgte i startkoden. Den extender Exception klassen og tager imod et parameter af datatypen String i sin constructor.

Det parameter benytter vi til at sende vores fejlbesked til brugeren ved at gemme beskeden i en session variabel kaldet ”error”, som vi derefter kan tilgå og fremvise på vores jsp sider.

# Implementerings-status

Vi har nået at få implementeret alt som hører til bruger/kunde delen, hvilket vil sige usecase: 1, 2, 3, 4, 5 og 8.

Slut produktet som vi er endt med, er som følgende:

* En visuel tilfredsstillende og dynamisk hjemmeside
* Sider hvorpå en bruger henholdsvis kan oprette en konto, logge ind og logge ud. Kontooplysningerne hentes og gemmes på vores database
* En side hvor man kan sammensætte sin egen cupcake og derefter tilføje den til sin kurv
* En side hvor man kan gennemse, redigere (fjerne) og købe de varer man har lagt i sin kurv, hvorefter bestillingen gemmes på vores database
* En side hvor en bruger, som er logget ind, kan finde sine kontooplysninger herunder fornavn, efternavn, e-mail og konto saldo. Alle oplysninger hentes fra vores database
* En side hvor en bruger, som er logget ind, kan se en oversigt samt detaljer over alle de bestillinger som pågældende har foretaget

Hvad vi dog ikke har formået at nå i denne omgang, er usecase 6, 7 og 9, som alle hører til admin delen. Dette har udelukkende været grundet mangel på tid:

* Sider med oversigt over henholdsvis alle bestillinger og alle brugere, som kun kan tilgås af en bruger der er logget ind med en admin rolle
* En side hvorpå man kan se en liste over alle ordre tilhørende en specifik kunde, som kun kan tilgås af en admin som er logget ind
* En side hvor admin kan tilføje et beløb til kundernes konto. Også en side som kun kan tilgås af en admin bruger som er logget ind

Derudover har der været andre ting vi også gerne ville have nået så som:

* At vi gerne ville havde tilføjet et ”pickup date” input felt til køb af cupcakes siden. Databasen er sådan set allerede klar til at modtage denne data
* Tilføje mere validering af input
* At gøre hele hjemmesiden mobil venlig. Nogle af siderne er, men ikke alle
* At refaktorerer vores kode

# Arbejdsproces

Planen for vores arbejdsform og selve projektets forløb var at være på bølgelængde fra starten af. Derfor valgte vi at bruge god tid på opgavebeskrivelsen og derefter lave en mockup (design), så vi begge vidste, hvordan hjemmesiden helt præcis skulle se ud/fungere.

Dette gjorde vi inden vi begyndte at kode og derefter designede vi html til de enkelte sider, så det var lige til, når vi skulle lave JSP sider og koble det sammen med koden.

Vi udelte opgaver via Trello og mødtes fysisk og online, så vi konstant vidste hvad den anden gik i gang med og ikke snublede over hinanden. Samtidig brugte vi GitHub, hvor vi havde hver vores developer branch, så vi hver især kunne teste og arbejde på hver vores delopgaver hver for sig.

Selve arbejdsmoralen og kommunikationen er gået rigtig godt. Vi har ikke været i tvivl om, hvad der skulle laves eller hvordan slutproduktet skulle være. Vi har været til stede for og hjulpet hinanden, hvor vi nu kunne.

Vi har lært, hvordan det er bedst for os at arbejde og hvor hinandens styrker og svagheder ligger. Det har fungeret godt at vi har kunne guide hinanden og kunne sige, hvis der var noget man havde svært ved eller ikke forstod.

Det at vi kun har været to i gruppen har på mange punkter været rigtig godt, men som forventet også kommet med sin andel af udfordringer.

Hvad der har været specielt godt ved kun at være to, er at vi begge har fået muligheden for at komme godt rundt omkring alle de forskellige funktioner, som projektet rummer og derved tilegnet en masse erfaring derfra. Der var rigeligt med delopgaver, som vi begge kunne tage fra og rode med. At aftale og blive enige om diverse beslutninger, har også været ligetil, da vi kun har haft to holdninger at forholde os til, hvilket selvfølge også kan anses for at være mindre godt på visse punkter.

Den største udfordring var som forventet tiden. Der skal jo laves meget mere per person, når delopgaverne kun kan fordeles imellem to og det vidste vi godt fra starten af.

Vi har igennem dette forløb fået et godt billede af, hvordan vi skal håndtere et projekt som denne i fremtiden og hvad der virker for os og hvad der ikke gør, sådan så vi er optimeret og får det bedste produkt, som vores færdigheder rækker til i sidste ende.