

Cupcake



af

Hold: Cupcake Bornholm 1

Steen Sander Rybak - cph-sr347@cphbusiness.dk

Github: knogleknuser

Louis Odin Marini Knoblauch - cph-lk366@cphbusiness.dk

Github: FartFace500

Nikolaj Fjellerad - cph-nf70@cphbusiness.dk

Github: Nikolaj1992

Indholdsfortegnelse

Indholdsfortegnelse.....	1
Indledning.....	2
Baggrund.....	2
Teknologi.....	2
Krav.....	3
Aktivitetsdiagram.....	4
Domæne model og ER diagram.....	6
Domæne model.....	6
ER-diagram.....	7
Navigationsdiagram.....	10
Sekvensdiagram.....	12
Særlige forhold.....	13
Status på implementation.....	16
Proces.....	18
Bilag.....	19
Bilag 1 - Figma mockup.....	19
Frontpage - index mockup.....	19
Login mockup.....	20
Create account mockup.....	21
Cart/Kurv mockup.....	22
Bestillinger mockup.....	23
Bilag 2 - Source Code og Demo Video.....	24
Sourcecode på GitHub.....	24
Demo Video.....	24

Indledning

Som led i datamatikeruddannelsens 2. semester, er vi blevet bedt om at lave en hjemmeside til den fiktive virksomhed Olsker Cupcakes. Formålet med projektet er at få et indblik i, hvordan vi, som leverandør, kan stå overfor en kundeopgave og arbejde ud fra specifikke krav og ønsker.

Idéen er at få et indblik i hvordan et typisk projektforsløb kunne se ud, fra de første user stories og det første mockup (**se bilag 1**), til et brugbart produkt.

Baggrund

Olsker Cupcakes er en lille fiktiv lokal virksomhed, som udelukkende producerer og sælger cupcakes. Deres cupcakes sælges både i deres fiktive fysiske butik, men kan også bestilles og afhentes af kunden.

Særligt på bestillingssiden vil virksomheden gerne gøre det nemmere for deres kunder, ved at have en hjemmeside som er overskuelig, og tillader kunderne at kunne bestille cupcakes med frit valg af en række forskellige bunde og toppings.

Når bestillinger modtages kunne virksomheden ligeledes godt tænke sig at det skal være nemt at håndtere disse på samme hjemmeside.

Teknologi

For at kunne imødekomme kravene har vi anvendt de følgende teknologier:

Database:

- PostgreSQL: 42.7.2
- pgAdmin 4: 8.3
- Docker: 4.27.2
- pgJDBC: org.postgresql.Driver 42.6.0
- Jdbc: 4.2

Backend:

- IntelliJ IDEA 2023.2.3 (Ultimate Edition)
- Java: 17
- SDK: Amazon Correto: 17.0.8
- Maven: 4.0.0
- Maven-compiler-plugin: 3.10.1

- Maven-shade-plugin: 3.4.1
- Javalin: 6.1.3
- Javalin-rendering: 6.1.3
- HTTP: 1.1
- hikariCP: 5.1.0

Ukendt hvor nødvendige de følgende 4 teknologier er, men de står i pom filen.

- jackson: 2.17.0
- slf4j: 2.0.12
- junit: 5.10.2
- hamcrest: 2.2

Frontend:

- Thymeleaf: 3.1.2.RELEASE
- Thymeleaf-extra: 3.0.4.RELEASE
- HTML5
- CSS3
- FIGMA: 116.17.12 (for Mockup - vist i bilag 1)

Krav

Den fiktive virksomhed Olsker Cupcakes vision er at gøre deres produkter mere tilgængelige via online bestilling, samt give et lettere overblik over bestillinger for selve virksomheden.

Følgende user stories er udleveret som krav.

- **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.

De første seks af disse user stories er minimumskrav/funktionelle krav for hvad der forventes og er derfor need-to-have, mens de sidste tre anses som nice-to-have.

Der blev yderligere stillet en række ikke-funktionelle krav.

1. Der laves en mockup i Figma (**bilag 1**) eller lignende, som viser de websider den færdige løsning kommer til at bestå af.
2. Ordre, kunder og øvrige data skal gemmes i en database.
3. Databasen skal normaliseres på 3. normalform med mindre andet giver bedre mening.
4. Kildekoden skal deles på GitHub.
5. Det færdige produkt skal udvikles i Java 17, Javalin, Thymeleaf template engine, Postgres Database, HTML og CSS.
6. Websitet skal helst kunne fungere tilfredsstillende både på en almindelig skærm og på en mobiltelefon (iPhone 12 og lignende). Hvis det volder problemer, så lav kun jeres løsning til en laptop.

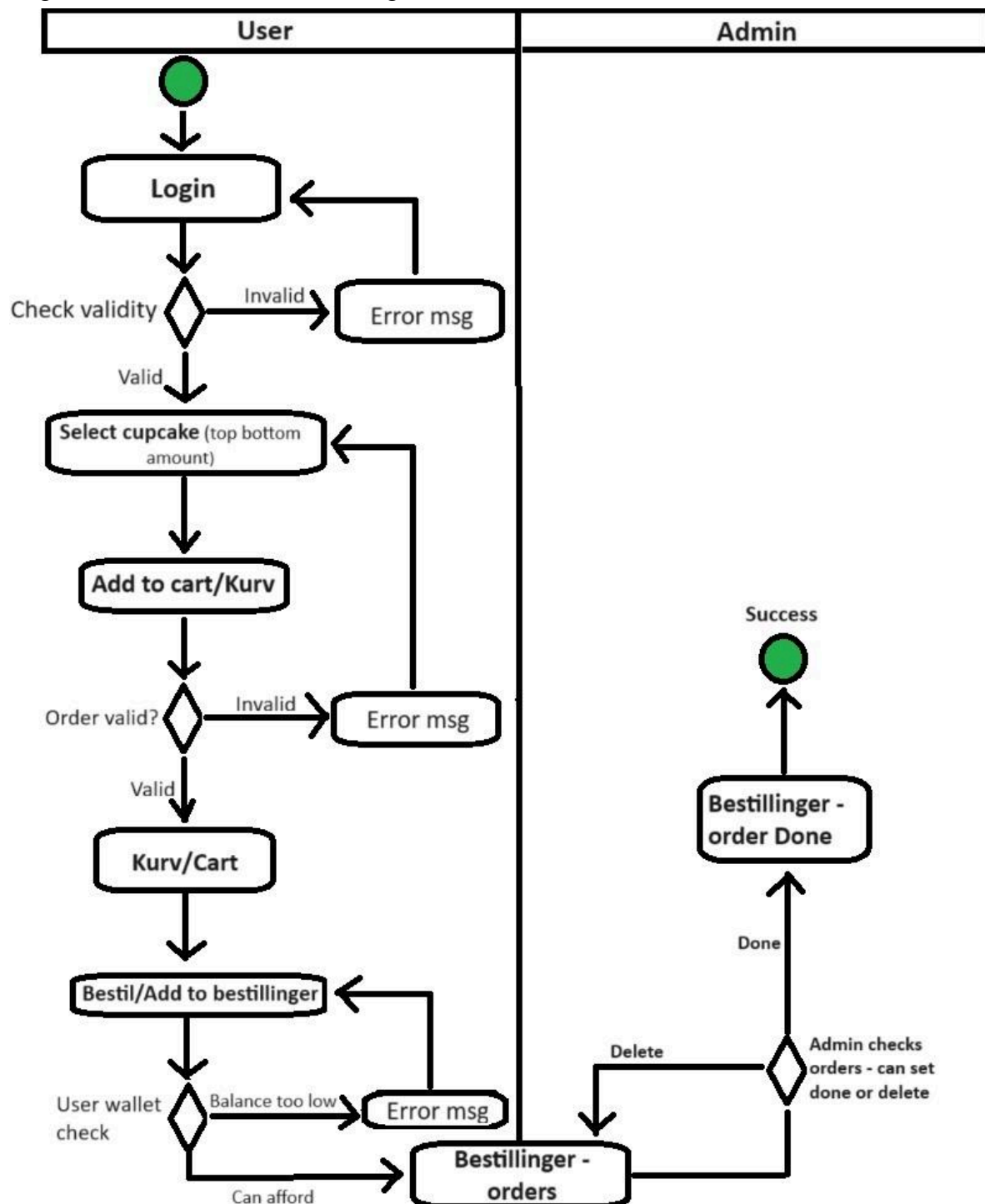
Aktivitetsdiagram

En lille note til dette diagram. Dette diagram blev lavet til sidst i stedet for først i forløbet. Dog vil diagrammet forsøge at vise et simpelt workflow fra start til slut, når man tilgår produktet og vil bestille cupcakes.

Vi har lavet et "TO-BE" aktivitetsdiagram, som viser hvordan vores workflow, i simpelt format, ser ud. Da man ikke kan fuldende et køb uden at være logget ind, starter vi i login-fasen. Her kan selve login'et fejle ved ugyldig input og medfører en fejlbesked. Herefter har brugeren mulighed for at "designer" de cupcakes, han/hun gerne vil købe og lægge dem i kurven. Igen kan der opstå fejl, hvis eksempelvis at antal ikke er specificeret.

Herfra kan brugeren tilgå kurven og se de valgte cupcakes samt pris, og vælge at sende bestillingen af sted. Her forekommer der et wallet-check, hvor systemet tjekker om kunden har råd til købet - endnu en fejlbesked kan opstå. Se under diagrammet for forklaring af admin-siden.

Bagkant skitse af et aktivitetsdiagram



Efter en bestilling er modtaget ved et succesfuldt wallet-check, kommer bestillingen ind i databasen og kan nu ses på bestillingssiden af både bruger og admin. Herfra kan admin, takket være admin status, se ordrer fra alle brugere og vælge om en ordre skal udføres eller slettes fra systemet (fx i tilfælde af at en fejlagtig ordre har sneget sig igennem).

Når en ordre sættes til udført af admin, har vi et succesfuldt workflow.

Domæne model og ER diagram

Efter at have modtaget kravene for hvad forventningerne til systemet er, så blev der begyndt at skitsere hvordan der tænkes at systemet ville fungere i form af en domænemodel og et ER-diagram. Når vi var tilfredse med vores skitser for hvordan systemet ville fungere, så gik vi i gang med at arbejde på udviklingen af systemet.

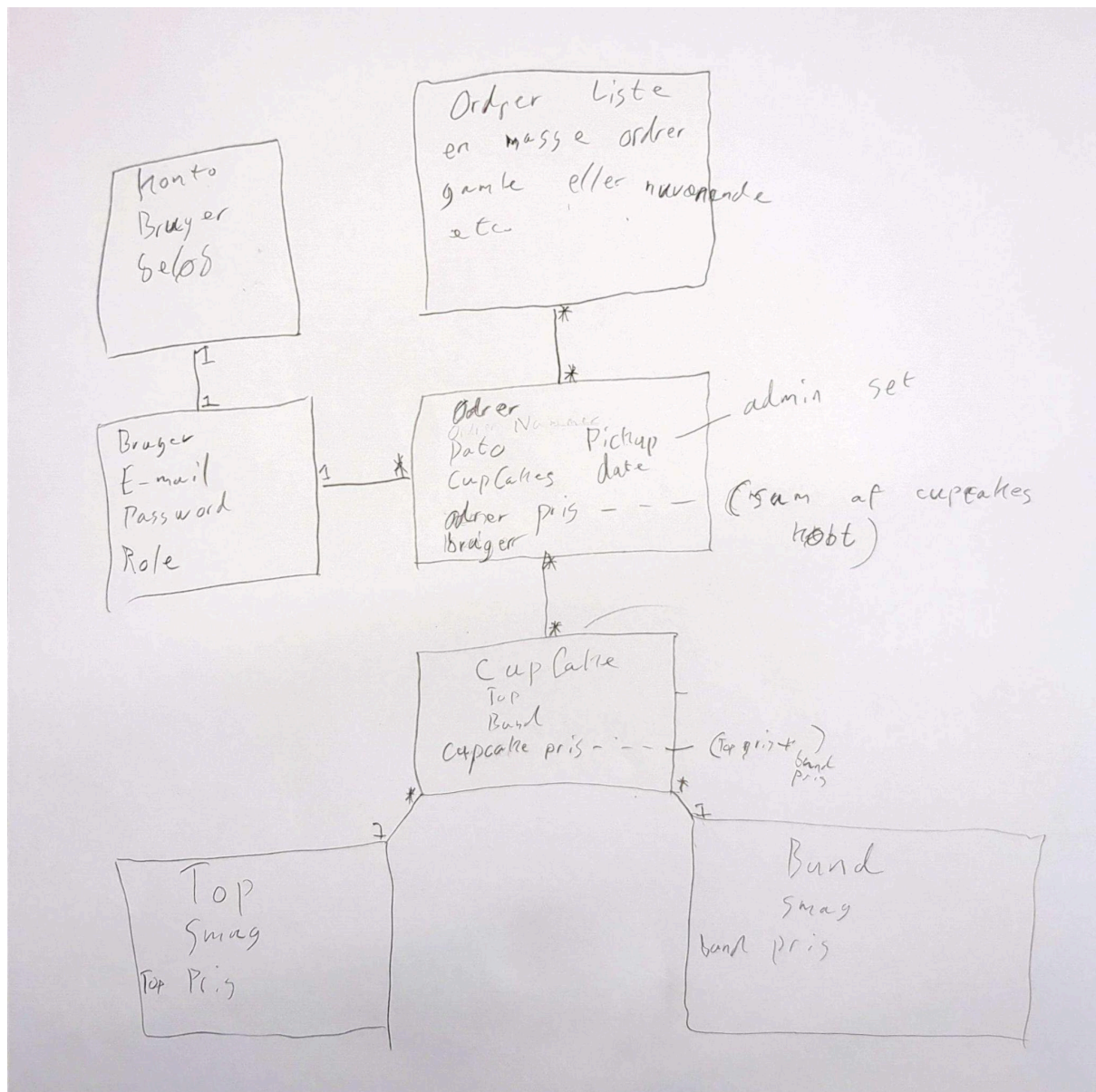
Domæne model

Klassekandidaterne som er kommet med i domænemodellen repræsenterer delelementer af hvad systemet skal arbejde med. Vi tænkte alle klasserne i modellen vil have deres egne tabeller i databasen, som systemet er tilkoblet.

Cupcake og ordrer liste klasserne er de eneste undtagelser for denne regel i domænemodellen, da de ikke har deres egne tabeller i databasen. Cupcake klassen bliver brugt til at skabe et cupcake objekt i systemet, som bruges til at transportere informationer på tværs af klasser i systemet. Kombinationen af cupcake objektets indhold vil blive gemt i databasen i to tabeller, ordrer linje og order, som vil referere til hinanden for at holde styr på en ordre.

Der er senere hen blevet oprettet en klasse kaldet 'OrderLine', som er en ordrer linje som indeholder den information en cupcake i en bestilling har. Ordrer linje klassen er dog tilkoblet en tabel i databasen. Denne ordrer linje er endt med at udskifte cupcake objektet ordrer klassen og resten af systemet.

Vores oprindelige domæne model



ER-diagram

Et ER-diagram går mere i dybden end en domænemodel, derfor er det primært udvikleren som vil få gavn af det. ER-diagrammet indeholder alle tabeller og relationer som vi på tidspunktet af skitseringen tænker skal bruges og er relevante for systemet.

Her blev 'wallet' til sin egen tabel, da det forhåbentlig vil være nemmere at transformere til en række gemte betalingskort i forhold til at have en kolonne på 'user' med en pengemængde. Derudover har vi valgt at bruge integers til at repræsentere penge i øre, grundet at penge har brug for et præcisionsniveau som floating points ikke tillader og dermed kan vi ikke bruge en double.

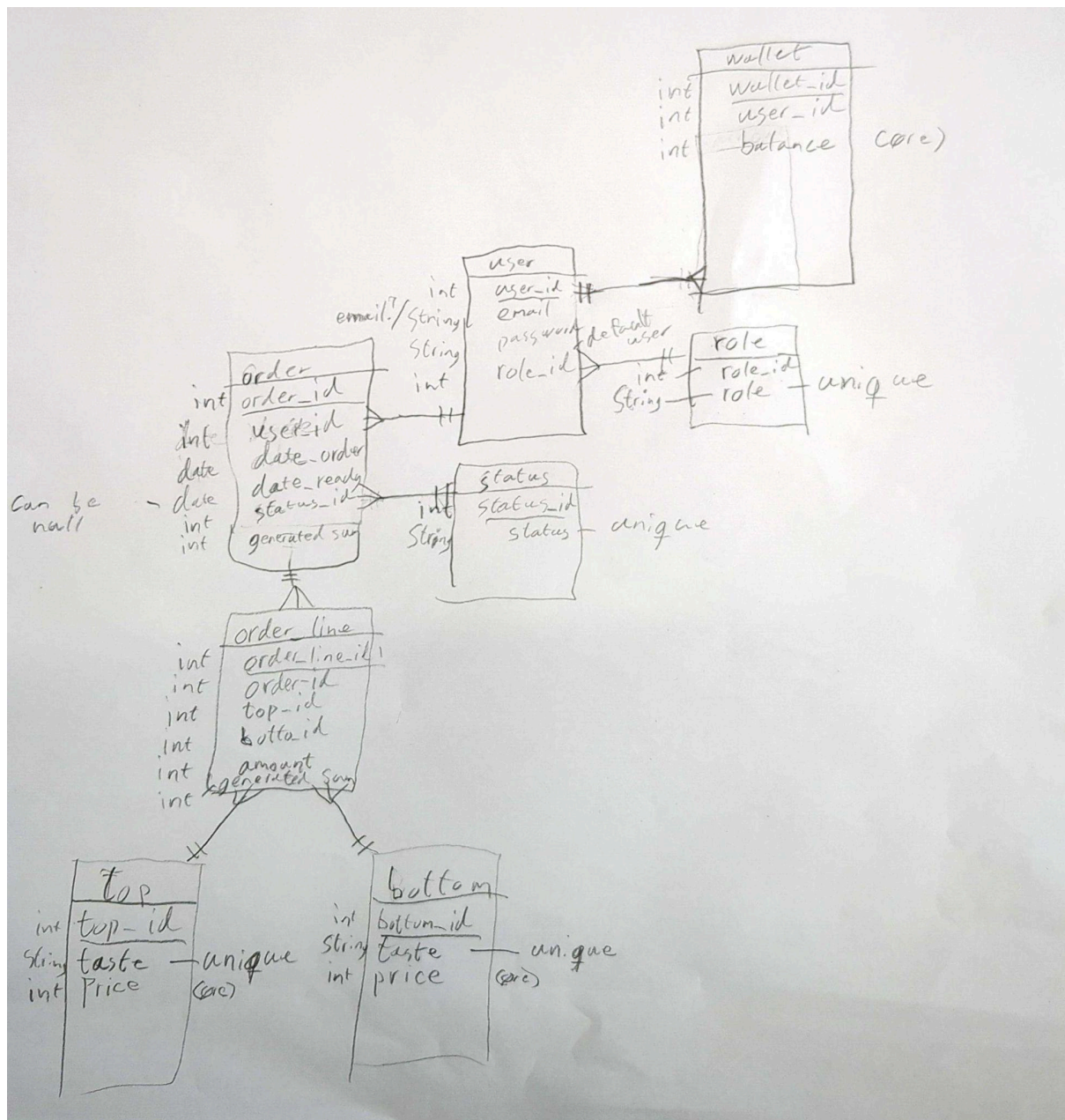
'status' og 'role' er også sine egne tabeller for at holde nemt styr på alle gyldige

status'er og roller i systemet, samt gøre det nemt at tilføje ekstra informationer der kunne høre en status eller rolle til.

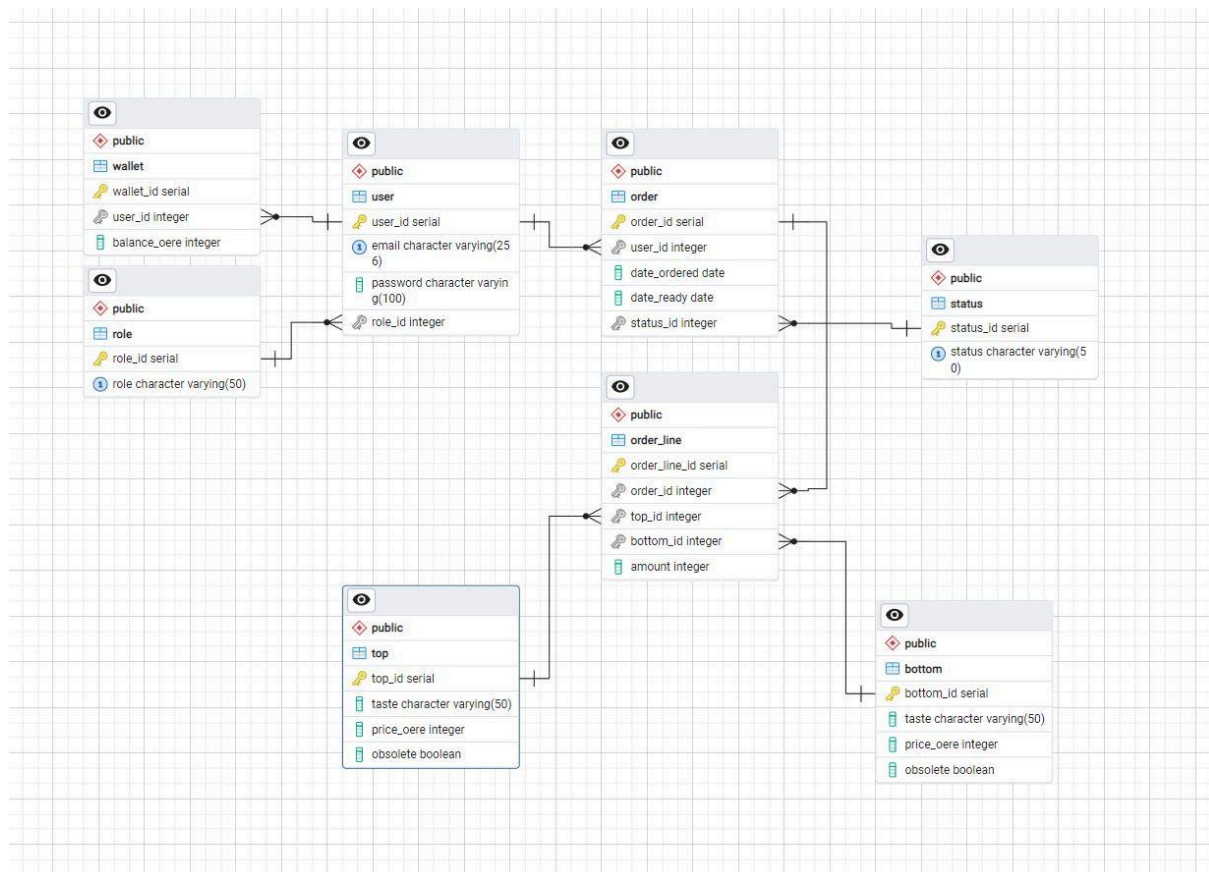
Under udviklingen af dette system kom der opdateret ER-diagram, da det var nødvendigt at foretage nogle tilføjelser til det oprindelige system. Den anden version har fået to tilføjelser i både 'top' og 'bottom'-tabellerne. Der blev tilføjet 'obsolete' kolonnen og 'taste' er ikke længere tvunget til at være unik for hver forekomst af den i tabellen.

Denne ændring til 'top' og 'bottom'-tabellerne blev foretaget fordi at der kom bekymringer om at en ændring på prisen af en top eller bund i databasen ville ændre på tidligere oprettede bestillinger i databasen. Dette kunne nemlig ende med at være et problem siden at priserne af tidligere bestillinger indeholder referencer tilbage til prisen på en hvis top eller bund i databasen. 'obsolete' attributtet ville give muligheden for at have en ny version af samme smag med en anden pris, så tidligere bestillinger vil kunne beholde den pris de havde på tidspunktet de blev oprettet og så at den samme smag i en nyere ordre kunne have en anden pris. Det betyder også at gamle toppe og bunde bør ikke slettes medmindre alle ordrer der bruger dem også slettes. Dette kunne ske hvis, for eksempel, alt information ældre end 10 år skal slettes

Første skitse af ER Diagram



Sidste version af ER diagram



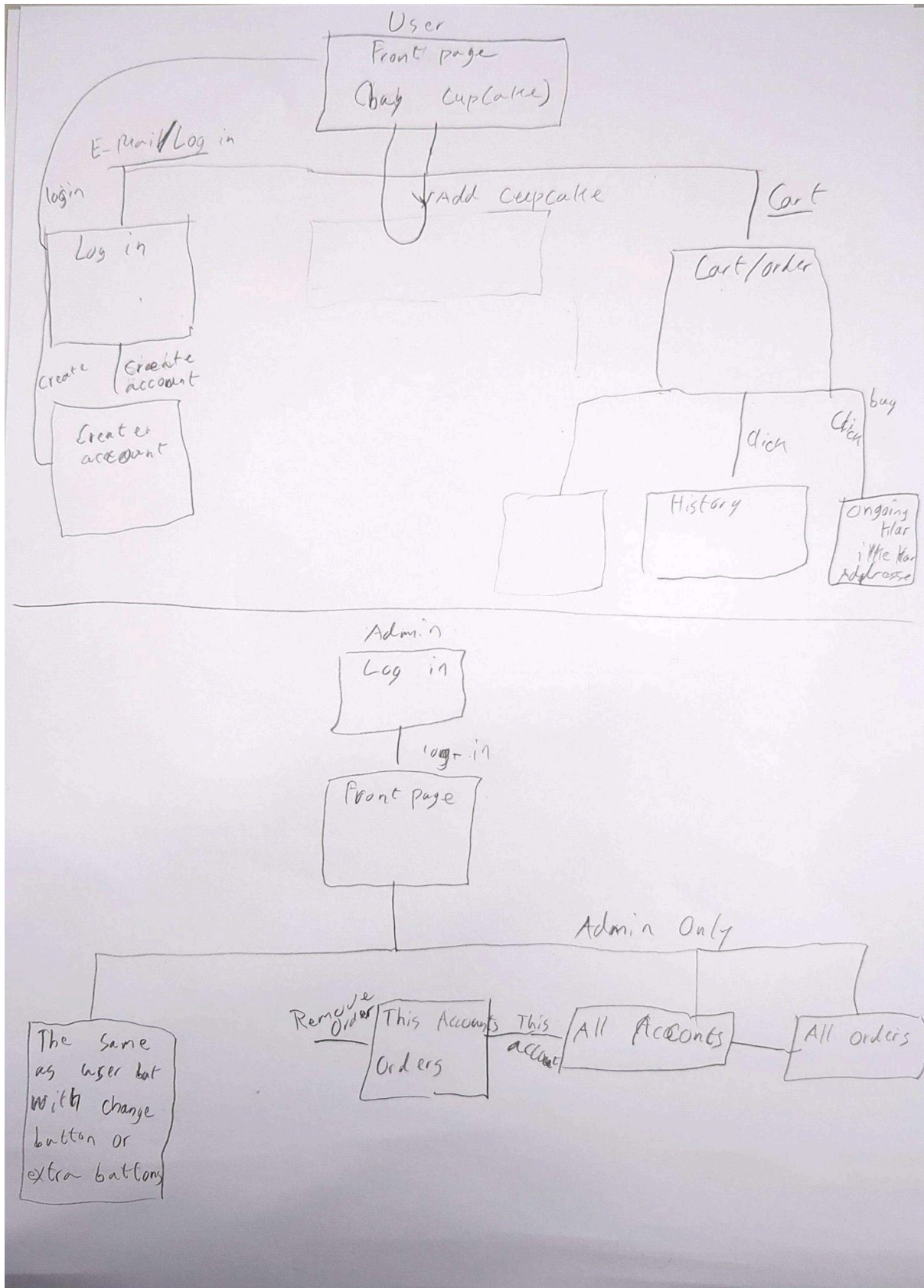
Navigationsdiagram

Navigationsdiagrammet vi har lavet er opdelt i både en administrator og en normal bruger del. Som en administrator er der muligheden for at slette, se og markere ordrer som klar til afhentning. Der er ikke specielle sider for administratorerne, men de kan under bestillinger se specifikke eller alle brugeres bestillinger og håndtere dem på samme side. Udover at en administrator kan håndtere ordrer så kan de funktionelt det samme som en normal bruger der er logget ind, i det at de kan bestille en cupcake.

Det er dog ikke et krav at man logge ind på hjemmesiden med det samme for at kunne tilgå siden hvor at man kan se hvad der i sin kurv og siden hvor at man opretter en cupcake der ender i ens kurv, men for at kunne bestille en cupcake eller se og have en købshistorik skal man være logget in. Senere i udviklingen af hjemmesiden er der blevet lavet den ændring, at en brugers købshistorik nu vil ligge under en delside som hedder 'bestillinger', i stedet for at man ville tilgå dem gennem kurven.

For at logge ind skal man blot trykke på knappen for at logge ind eller oprette en bruger som findes i højre side af navigationsbaren, og for at få adgang til sine bestillinger og kurv skal man kigge oven i den venstre side af navigationsbaren som går igen på hver side.

Vores oprindelige navigationsdiagram



Sekvensdiagram

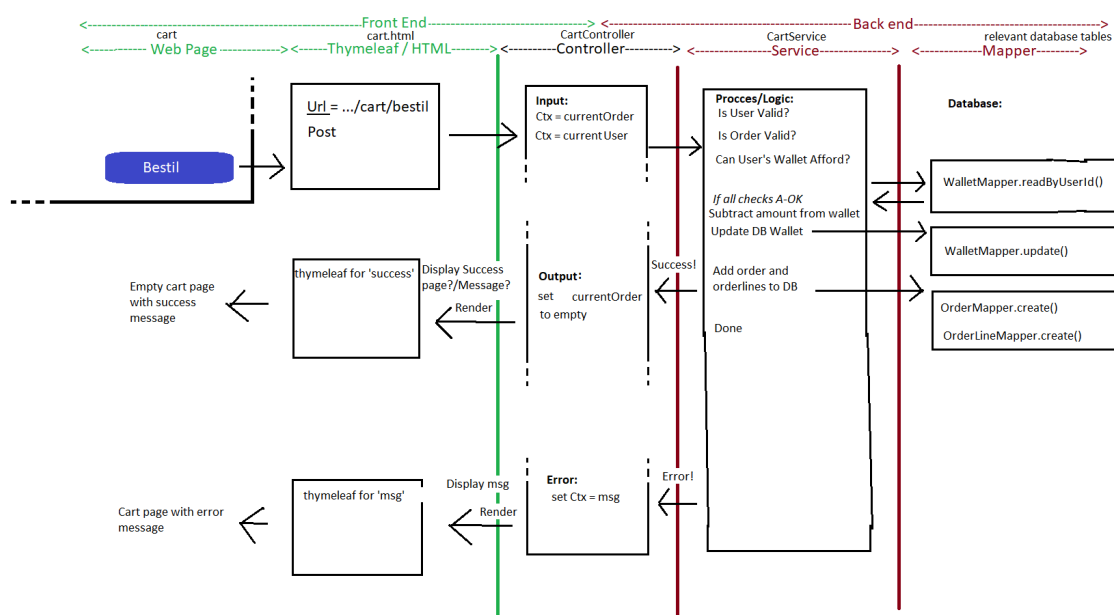
På starten af sidste kodedag for projektet (søndag den 07/04/2024) var der et mangler på overblik over koden. Hertil blev det følgende sekvens diagram udviklet for at gendanne overblikket.

På dette tidspunkt var bestil-funktionen ikke implementeret endnu.

Det er et sekvensdiagram for, hvordan vi behandler enhver form for HTTP request i vores projekt.

Specifikt viser den for eksempel sekvensen for når man bestiller en kurv og de relevante metoder, en feature, inklusiv dens metoder, der ikke var implementeret på tidspunktet diagrammet blev lavet.

Sidste kodedags sekvensdiagram



NOTE: 'create' metoder burde have en pil der peger tilbage til service, da disse metoder giver DTO'erne et ID.

NOTE: Det er værd at bemærke at det er med vilje blevet valgt ikke at eksplicit inkludere DTO objekter i dette diagram.

OBS: 'HistoryController' og 'HistoryService' er ikke så gode til at følge dette diagram og som resultat er de ret rodet.

Sekvens Diagrammet Uddybet

En kort, men dybere forklaring er at vores kode og mapper er struktureret efter hvilken webpage vi arbejder med, så på page'en 'cart' har vi 'cart.html' og 'cart.css' i spidsen af frontend.

Vi overfører input fra frontend og output/exceptions fra backend i controlleren kaldet 'CartController'.

Selve backend logikken og checks finder vi i servicen 'CartService'.

Allerdybest i backend har vi java mappers til databasen som ikke er sat op efter webpages men efter databasen. Derfor findes der ikke nogen 'CartMapper', da der ikke er en tabel i databasen kaldet 'cart'.

Til gengæld har vi tabeller for brugerens konto, 'wallet', brugerens ordre 'order' og brugerens ordre indhold 'order_line'. Derfor har vi netop CRUD på mappers til disse 3 tabeller på 'WalletMapper', 'OrderMapper' og 'OrderLineMapper'.

Det betyder at frontend og backend kun behøver at mødes i controlleren for håndtering af input, output og errors.

Desuden tillader det også en 3 opdeling mellem frontend, java web-backend og java database-backend, da web-backend AKA service, har ikke brug for også at arbejde med at lave CRUD til databasen, da de er sine egne ting under mappers.

Overordnet så hjalp sekvensdiagrammet gevaldigt, men der er stadig plads til yderligere struktur eller andre forbedringer til fremtidige projektor.

Særlige forhold

Informationer Gemt I Sessionen:

Som man kan se i Java interface'en 'CtxSessionAttributes' under 'constants', så har vi 3 session attributes.

1. Den nuværende bruger der er logget ind ('currentUser')
(Java Klasse='User')
2. Den nuværende kurv ('currentOrder')
(Java Klasse='OrderExtended')
3. Den nuværende ordrer historik ('currentOrderHistory')
(Java Klasse='UserExtended')

Derudover har vi konstanter for navne af web attributes under andre scopes, så som:

- parameter attributes under interface'en 'FormParam'
- request attributes under interface'en 'CtxAttributes'
- session attributes under interface'en 'CtxSessionAttributes' (som sagt)
- application attributes under interface'en 'CtxGlobalAttributes' (session i praksis)

Det er værd at bemærke at det blev aldrig regnet ud hvordan man faktisk satte et application-scope attribute og senere hentede den i Thymeleaf, som konsekvens er alle 'CtxGlobalAttributes', som er brugt i Thymeleaf, blevet sat som session

attributes.

Her påvirker det hovedsageligt 'BOTTOM_MAP' og 'TOP_MAP', da disse bruges på forsiden til at vise dropdowns over bunde og toppe til websidens cupcakes.

'SomeDtoExtended' Klasserne:

'OrderExtended' og 'UserExtended' er begge udvidelser til deres respektive Data Transfer Objects (DTOs) ('Order' og 'User'). De indeholder lidt ekstra information i form af en sum (af pris) og et map eller liste af deres sub-kategori, samt metoder brugt i thymeleaf til visning af strings på websiden.

Mere specifikt, så indeholder 'UserExtended' en sum pris for alle ordrer som denne bruger har lavet, med andre ord, hvor mange penge de har brugt på cupcakes i alt, metoder til strings til brug i thymeleaf, og et map af alle ordrer denne bruger har lavet i form af 'OrderExtended'.

'OrderExtended' indeholder en sum pris af netop denne ordre, metoder til strings til brug i thymeleaf, samt en liste af 'OrderLineExtended'.

En 'OrderLineExtended' er en 'OrderLine', der er i denne ordre.

'OrderLineExtended' indeholder kun en attribut i form af en sum for prisen af netop denne kombination af bund, top og antal samt nogle ekstra metoder til strenge til brug i thymeleaf. Med andre ord, mere eller mindre det samme som de andre metoder dog uden sin egen sub-map/list.

Exceptions:

Alle exceptions har en bruger besked, som bliver vist på den relevante side.

Alle exceptions udvider den abstrakte klasse 'UserFriendlyException' og er stort set identiske. De forskellige udvidelser findes kun i tilfælde af at differentiell behandling af exception'en er nødvendig, hvilket det aldrig blev i dette projekt.

Validering af Bruger Input:

Alt brugerinput bliver så meget som muligt valideret nede i den relevante service.

For eksempel, for siden 'login', for post requesten 'login' kan du finde bruger valideringen i LoginService.login(parameters...).

Denne opsætning gælder så vidt muligt for alle sider og requests.

CRUD:

Der er nogle særlige forhold med hensyn til CRUD, da den er blevet generaliseret i dette projekt for at spare tid, da det ellers ville være problematisk at implementere alle 48 CRUD metoder.

(6 metoder per tabel, 8 tabeller.)

(6 metoder minimum og ikke 5, da vi har 2 read metoder, en readAll og readSingle)

Alt CRUD bruger 'TemplateSharedCrud' for at kunne udføre sin funktion. Selve Crud metoderne og mapperne indeholder kun SQL og oversætter en DTO til et array af parametre i form af et Object Array.

Dette kan gøres takket være den følgende Java metode
`preparedStatement.setObject(index, parameterAsObject)`

Denne metode vil automatisk sætte den relevante datatype i databasen, dog der var problemer med Datoer og derfor finder man at den datatype bliver særbehandlet med en 'if instance of' og casting.

Derudover indeholder alle mappers 3 instans metoder og er singletons. Det betyder at kun mapperne og 'TemplateSharedCrud' har adgang til disse instans metoder, som alle er en implementation af interface'en 'TemplateDtoCreator'.

Disse 3 instans metoder findes for at fortælle 'TemplateSharedCrud' hvordan man netop laver denne DTO fra et 'ResultSet' og hvordan man assigner den et ID i tilfælde af at 'create'-CRUD bliver brugt. (Så vi får opdateret vores DTO med dens nye ID fra databasen)

En alternativ implementation kunne have været at vi i stedet for at putte disse metoder og implementering af interface'en 'TemplateDtoCreator' på selve mapperne og derfor gøre mapperne singleton, kunne refactor det om til at bruge en private static inner class i stedet. Denne kunne implementere 'TemplateDtoCreator' interface og de relevante metoder. Den inner class kunne så være en singleton. Dette ville gøre det nemt at konvertere mapper klasserne til instans klasser.

Derudover er 'TemplateSharedCrud' default access for at forhindre at noget andet end mappers sender SQL-strings afsted til databasen.

Som en sidste ting. Hvis en kolonne ikke tillader null-værdier og har en default, så er det den relevante mappers job at erstatte null-værdier med defaulten i de relevante metoder.

For eksempel, en 'user' har som default på 'role_id' værdien, dvs ID'et, 1 (hvilket er rollen 'user'). 'role_id' er en foreign key som ikke kan være null og har en default, derfor hvis du prøver at kalde 'create' på 'UserMapper' og giver en DTO 'User' med 'roleId' = null, så vil 'UserMapper' erstatte 'roleId' = null med 'roleId' = 1

'ConnectionPool':

En anden ændring er til 'ConnectionPool'. Dette skyldes at alt brug af 'ConnectionPool', udover hvad Javalin skal bruge, nu sker i netop kun én klasse, nemlig 'TemplateSharedCrud'.

Derfor bliver singleton instansen sendt direkte hen til 'TemplateSharedCrud' på server startup dog med en caveat. Den er i en wrapper klasse 'ConnectionPoolAccess' som kun tillader 'TemplateSharedCrud' at tilgå metoden 'getConnection()' på 'ConnectionPool' instansen.

Det betyder at 'ConnectionPool' ikke behøves at sendes med rundt i systemet, men

kan også gøre det mere besværligt at implementere flere 'ConnectionPools' i fremtiden, skulle det blive nødvendigt eller ønsket.

Status på implementation

Generelt er produktet en fuldt ud implementeret cupcake webseite der opfylder alle funktionelle og ikke-funktionelle krav. Dog, med nogle små problemer hist og her og nogle imperfektioner, dog ikke noget som blokerer eller forhindrer en user story beskrevet funktion.

Mere konkret, så er implementationen det følgende:

Færdigt:

- Postgres Databasen
- ALT CRUD
- Opfyldt alle Funktionelle Krav (Se 'Krav' for mere info)
 - Herunder tilkobling til databasen
 - Java Backend
 - HTML Layout
 - Thymeleaf Dynamiskhed
 - CSS Styling
- Opfyldt alle Ikke-Funktionelle Krav (Se 'Krav' for mere info)

Delvis Færdigt:

- Siden 'Kurv':
 - Mangler en bekræft side, så man ikke køber en ordre med det samme, men skal bekræfte
- Siden 'Bestillinger':
 - Har lange strings i stedet for tabeller i HTML.
 - (ADMIN) Kan ikke sætte et færdig tidspunkt udover i dag
 - (ADMIN) At fjerne en ordre refunderer ikke beløbet
- 'HistoryController' og 'HistoryService' er noget rod, der skal ryddes op i.
- Generelt er Java doc dokumentation og kode kommentarer ikke fyldestgørende.
- Fejl og Exceptions:
 - 90% af fejlbeskeder til brugeren er ikke blevet oversat til dansk. Vi skrev dem på engelsk og opdagede fejlen senere.

- Ikke alle fejlbeskeder følger fejlbeskedstilen sat af forsiden.
For eksempel har nogle lokale strings og et mangler på korrekt prefix
- Ikke alle sider kan vise flere fejl på en gang, selvom det ville være relevant. Forsiden kan dette, brug den som eksempel

Mangler:

- Ingen Automatiserede Tests, herunder Unit Tests eller Integrations Tests
- Ingen acceptkriterier på userstories
- Ingen konkret, nedskrevet, 'definition of done' udover en løs ide baseret på tidligere opgavers erfaringer, såsom:
 - Features virker (ud fra korte manuelle tests)
 - Features giver errors der forklarer hvis det ikke virker
 - Features er stylet
 - Features følger 'kode standarden' og vores overordnede struktur
- Ingen konkret, nedskrevet, kode standard, udover en løs ide baseret på tidligere opgavers erfaringer, så som:
 - Java camelcase
 - Konstanter skal være med stort
 - Undtagen hvis det er for variabelnavne brugt som javalin attributes
 - Alle postgres variabler er skrevet med småt i snakecase
- Web attributes i application scope blev aldrig implementeret korrekt og bruger derfor session scope som en placeholder
- Viewet 'view_order_detailed' og dens mapper samt DTO blev aldrig brugt og kan derfor slettes.

Kendte Bugs:

- Siden 'Frontpage':
 - Flexbox er ikke sat korrekt op, hvilket gør at i omkring 680px - 800px skærmstørrelse glider 'Tilføj' knappen til venstre og dropdowns til højre
 - Viser 'obsolete' bunde og toppe, hvilket giver store negative konsekvenser hver gang man opdaterer en pris, da det er endnu en ugyldig valgmulighed der kloner op i brugerens gyldige valgmuligheder
- Siden 'Kurv':
 - Godkendt bestilling viser ikke success beskeden
- Siden 'Bestillinger':
 - (ADMIN) bliver nødt til at klikke på 'sorter' for at opdatere i tilfælde af nye ordrer eller brugere. Det er ikke nok at genindlæse siden.

Proces

Projektforløbet startede med at lave en domænemodel, navigationsdiagram samt en skitsering af ER-diagram. Dernæst lavede vi mockup i Figma og havde dermed et overblik over hvordan database og hjemmeside kunne ende med at se ud, og fungere.

Vi valgte at anvende Trello, som er et webbaseret kanban-board, til at holde overblik over de elementer/delopgaver som indgik i projektet samt benyttede os af Trello's værktøjer til at prioritere opgaver.

Arbejdsformen blev således at man kunne sætte sig på et Trello-card (opgave) og sætte i gang, samtidig med at vi var i et Discord opkald, i hele arbejdstiden.

Noget vi måske skal blive bedre til, kunne være at begrænse os til "det minimalt acceptable produkt" i vores arbejde, især ved særdeles tidsbegrænsede projekter

Noget vi har lært af processen, og af projektet generelt, er vigtigheden af at holde pauser og sikre at vi alle er helt med på hvordan vores kode hænger sammen og fungere, for at undgå at miste overblikket over projektet i de senere stadier.

Til næste gang vil det nok være godt med et aktivitetsdiagram i starten af projektet.

Men overordnet set var det et godt samarbejde, hvor alle gruppemedlemmer tog ansvar og deltog engageret i projektet.


Bilag

Bilag 1 - Figma mockup

Frontpage - index mockup

Det udleverede mockup til Cupcake projektet

Det var "okay at ændre layoutet".



[Order](#) [Kunder](#) cupcakejohn@hotmail.com [KURV](#)

Velkommen ombord

Øens bedste cupcakes. Vælg og bestil her:

Vælg bund

Vælg top

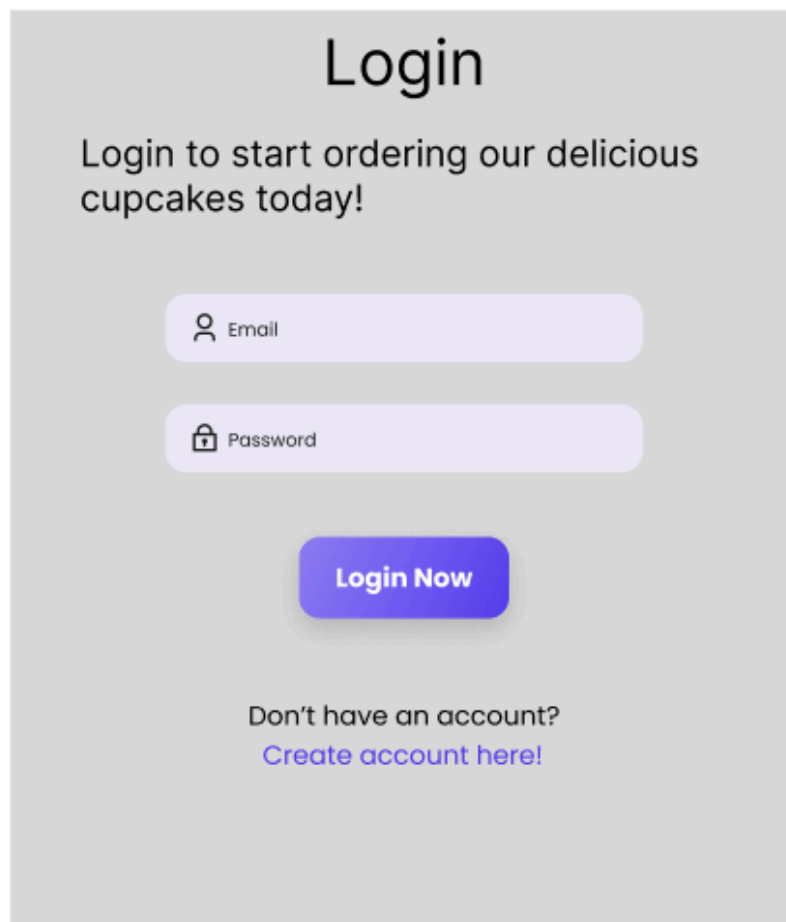
Vælg antal

Skriv en kommentar

Bestil

Login mockup

Figma mockup af login siden

A login form mockup on a light gray background. At the top, the word "Login" is centered in a large, black, sans-serif font. Below it, the text "Login to start ordering our delicious cupcakes today!" is centered in a smaller, black, sans-serif font. There are two input fields: the first is labeled "Email" with a person icon, and the second is labeled "Password" with a lock icon. Both fields are light purple with rounded corners. Below the input fields is a blue button with rounded corners and the text "Login Now" in white. At the bottom, the text "Don't have an account?" is centered, followed by a blue link "Create account here!" in a smaller font.

Create account mockup

Figma mockup af opret bruger siden

Logo!

Create Account

Create your account. Please use a valid email address.

Email

Password

Repeat password

Create

Cart/Kurv mockup

Figma mockup af cart/kuurv siden

Lad som om der er et logo her

Din bestilling:	
1x Cupcake med Chokolade bund og Vanilje top (12kr stk) 2x Cupcakes med Vanilje bund og Jordbær top (10kr stk) 1x Cupcake med Jordbær bund og Chokolade top (8kr stk)	
2x Cupcake 1 - Chokolade og Vanilje Chokolade bund: 6kr Vanilje top: 8kr Antal: 2 (14kr stk) Delsum: 28kr	
Total pris: 40kr	Bestil

Bestillinger mockup

Figma mockup af bestilling/historik siden

Lad som om der er et logo her

Ændre til kunde navn på admin version, måske lav det til en dropdown menu

admin version

Dine bestillinger:

Bestilling: 1

Dato for bestilling: 01/04/2024 14:32

Dato for afhentning: 03/04/2024 10:00

2x Cupcake 1 - Chokolade og Vanilje

Chokolade bund: 6kr

Vanilje top: 8kr

Antal: 2 (14kr stk)

Delsum: 28kr

Bestilling: 2 Slet

Dato for bestilling: 02/04/2024 11:28

Dato for afhentning: ikke sat Sæt

1x Cupcake 4 - Chokolade og Jordbær

Chokolade bund: 6kr

Jordbær top: 10kr

Antal: 1 (16kr stk)

Delsum: 16kr

Bilag 2 - Source Code og Demo Video

Sourcecode på GitHub

Public (Uden Git Log):

https://github.com/Steens-Amazing-Peeps/cupcake_public

Private (Med Git Log):

<https://github.com/Steens-Amazing-Peeps/cupcake>

(Kræver collaborator status, check for invites)

Det sensitive data er blevet skjult nu, men det kan stadig ses i Git Loggen.

Til alle fremtidige projekter vil der kun være et Public repository med Git Loggen, da det sensitive data aldrig vil blive gemt med Git igen.

Tak for forståelsen.

Demo Video

Kort demo video af websiden lavet i dette projekt:

<https://www.youtube.com/watch?v=8l2aHHiazOw>