

Indholdsfortegnelse

| | |
|---|----|
| Indledning | 3 |
| Værdikædeanalyse af SynsPunkt ApS | 3 |
| Ledelsesstile | 4 |
| Artefakter, hvilke og hvorfor | 4 |
| System Sekvens Diagram..... | 5 |
| Domænemodel | 6 |
| Kravsspecifikation..... | 8 |
| Usecase liste og Userstories..... | 8 |
| Ikke funktionelle krav | 10 |
| Scrum | 10 |
| Vandfald vs. Agil | 11 |
| Plan for udvikling..... | 11 |
| User stories | 11 |
| Arkitektur | 12 |
| ER diagram | 14 |
| SQL beskrivelser | 15 |
| Databasebeskrivelser | 15 |
| Beskrivelse af færdigt salgssystem | 16 |
| Forsiden..... | 16 |
| Programkode | 18 |
| Salg | 18 |
| Varer | 22 |
| Opret Vare | 22 |
| Use Case beskrivelse | 23 |
| Vareinformationer..... | 23 |
| Programkoden/SQL - Vare..... | 24 |
| Programkoden/SQL – Vareinformationer..... | 25 |
| Use Case:..... | 26 |
| Vis Alle Varer | 27 |
| Programkoden/SQL | 27 |
| Salgsstatistik fil | 30 |
| Kunde Application | 34 |
| Konklusion..... | 36 |

| | |
|--|----|
| Bilag..... | 37 |
| Bilag 1 Porters Værdikæde | 37 |
| Bilag 2 Booking | 37 |
| Bilag 3 User stories..... | 39 |
| Bilag 4 Use case beskrivelser..... | 41 |
| Bilag 5 Illustrationer af programmet | 44 |
| Bilag 6 SSD..... | 46 |

Indledning

Sanne & Emeli

Vi har fået udleveret en case-beskrivelse fra en lokal virksomhed SynsPunkt ApS. Ud fra denne case har vi formuleret forskellige kravspecifikationer til deres fremtidige IT-system. Dette har understøttet en udarbejdning af forskellige hjælpeværktøjer til os, som har gjort det muligt for os at udvikle selve deres system.

Systemet er operativt og i stand til at løse hoveddelen af deres ønsker til deres system.

Værdikædeanalyse af SynsPunkt ApS

Sanne

Porters værdikæde består af flere elementer. Som visualiseret i bilag 1 er der både primære og støtte aktiviteter. I vores korte analyse af SynsPunkt ApS værdikæde vil vi hovedsageligt fokusere på den ene støtteaktivitet: teknologi.

Her har vi fokus på det hele vores case drejer sig om, virksomhedens opgradering af deres interne IT-systemer.

I casen bliver det gjort helt klart at SynsPunkt ikke har fulgt helt med i tidens generelle udvikling af IT-systemer. Og derfor er der stort fokus på at få virksomheden og de enkelte forretninger ”up-to-date”, inden for området. Dog uden at kvæle ejers lidt gammeldags sjæl, som også er en del af virksomheden.

Ejer har altid lagt stor vægt på kvaliteten af sine produkter – det skal også gøre sig gældende for de systemer og programmer der skal udvikles.

Det tidligere billede af SP

Som nævnt tidligere, har SynsPunkt ikke gjort meget i at benytte teknologi og IT-systemer ude i butikkerne, da der generelt ikke har været den store tiltro til IT. Alt ejers kommunikation foregår derfor på den ældre manér – med alm post, telefon eller fax. Dette har ikke været effektivt, da det også er gældende for vigtige informationer som butikkerne helst skal have samtidig, hurtigt og ens. Meget af dette tidspres kan afhjælpes ved hjælp af IT-programmer og systemer.

Som det ser ud nu, så har de enkelte butikker kundeinformationerne til at ligge fysisk – eller i hvert fald lokalt i den enkelte forretning. Dette giver udfordringer for kunderne når de henvender sig i en af de andre forretninger. Kunder har påtalt dette som værende negativt i

forhold til deres valg af optiker, da det bliver en langsommelig og besværlig proces at komme igennem ekspeditionen.

Ledelsesstile

Emeli

I SynsPunkt er det ejer der har stået for den daglige ledelse i alle butikkerne. Derudover har han også siddet med alt det administrative for hele virksomheden. Dette har betydet at ejer har haft rigtig mange opgaver, og ikke formået at nå ud til forretningerne uden for Fredericia.

Ejer har udnævnt en butiksbestyrer i hver butik, men det er stadig alle medarbejdere i den enkelte forretning der har skullet sørget for bestilling af varer, dekorationer i butikkerne, samt budgettering mm.

Da Ejer har mistet overblikket i alle sine opgaver og tiden til at kunne løse alt med kun telefonopkald og fax, er ledelsen blevet meget passiv. Han har ikke kunnet overskue sine mange opgaver på daglig basis og dette er endt i en situation, hvor nogle af butiksbestyrerne har valgt at markedsfører produkter som ikke har været godkendt af ejer. Blandt andet lavpris briller og kontaktlinser, hvilket, ifølge ejer, ikke har været SynsPunkts salgsstrategier, da de er rettet mod kvalitet og design. Butikkerne er dermed gået i modsat retning af kædens kerneværdier. Et af kendetegnene på en Laissez-faire ledelsesstil, der har taget overhånd, grundet manglende kommunikation og tid er nemlig mangel på ledelse.

Alle medarbejdere er blevet lært op til at kunne have stort ansvar, når de er på arbejde i butikken. Den demokratiske ledelsesstil vil her være meget effektiv, da alle er med ind over beslutninger i deres egne butikker. Det lader alle medarbejdere komme med ideer og holdninger, så der er plads til kreative løsninger. Samtidig med at det motiverer medarbejdere til at lære og dygtiggøre sig endnu mere. Ejer får mere overblik og kan følge med på et overordnet niveau, samt tage de store beslutninger, der omhandler ting såsom varesortiment i alle butikker. Der imens mindre beslutninger er op til de enkelte butikker.

Artefakter, hvilke og hvorfor

Sanne

Vi har valgt at benytte følgende artefakter; domænemodel, usecase diagram og system sekvens diagram.

Vi har valgt at udarbejde en domænemodel, da det var vigtigt for os at få visualiseret, hvordan SynsPunkt fungerer som forretning. Derudover er det et godt redskab at benytte over for kunden,

for at være sikre på at de og vi er enige om hvordan deres domæne ser ud, inden vi begynder at optimere på deres system.

Emeli

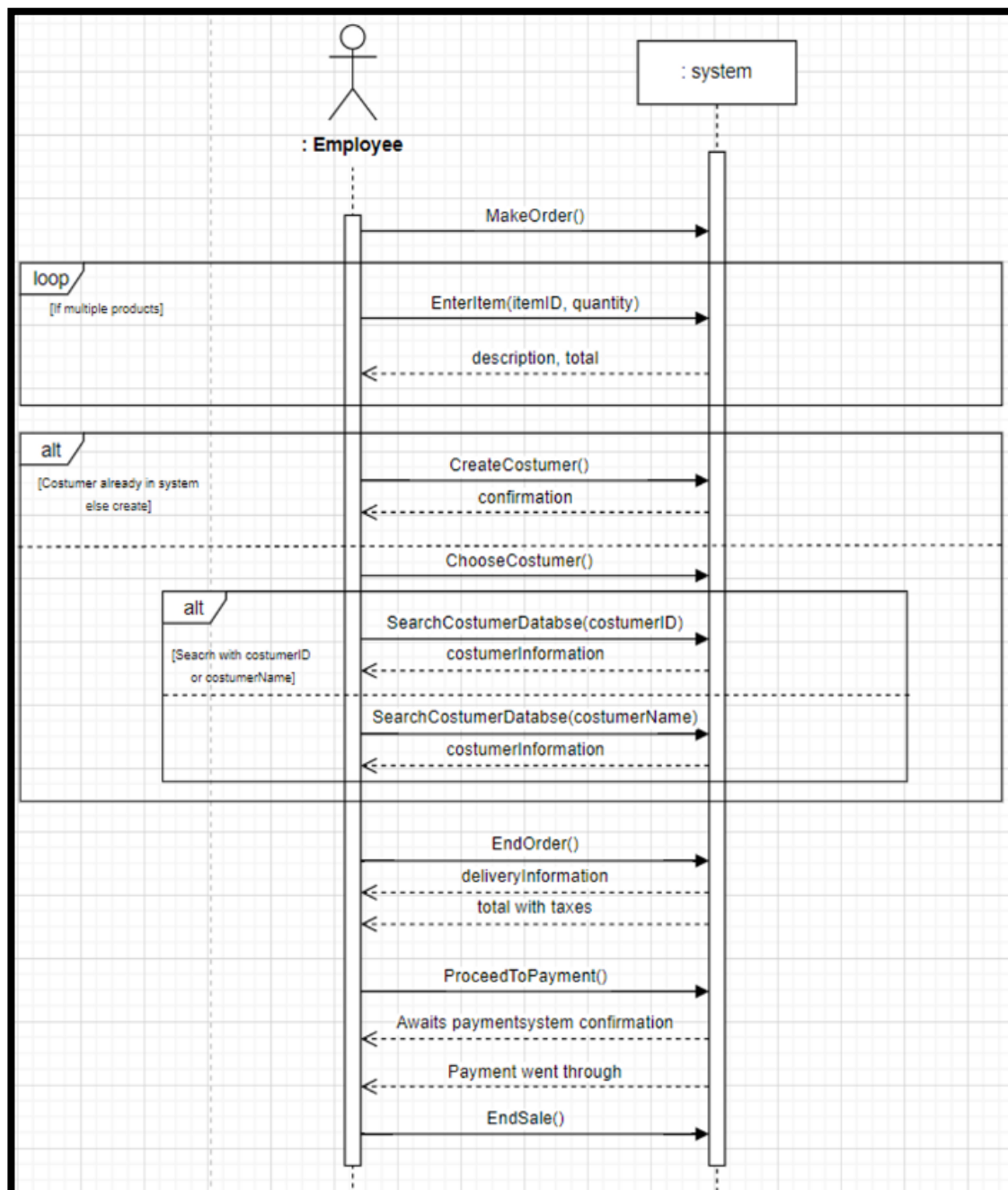
Usecase diagrammet giver overblik over de forskellige aktører som bruger programmet og hvilke Usecases de håndterer i systemet. I vores system gives alle brugere rettigheder til de forskellige funktioner (usecases) i programmet. Ejer af SynsPunkt skal være den eneste, som kan ændre varesortiment. Derfor er der taget udgangspunkt i usecase diagrammet. UC-diagrammet bliver beskrevet senere i rapporten, da det giver mere mening efter usecase listen er præsenteret.

System Sekvens Diagram

Emeli

Selvom et SSD ikke altid er nødvendigt, har vi alligevel lavet 4 SSD'er til udvalgte usecases, fordi vi i vores tilfælde synes de er relevante. De viser, hvordan aktøren gennem hændelser interagerer med systemet. Som processen skred frem, arbejdede vi agilt og sørgede derfor hele tiden for at tilpasse og optimere vores programmerings løsninger, derfor er det ikke alle der er ens med programmet.

I rapporten har vi valgt at beskrive og medtage det ene diagram, der beskriver 'Opret Bestilling'. Resten ligger som bilag. Vær opmærksom på at metoden 'Gå til betaling' sender 'user' over i et helt andet betalingssystem, her startes selve salget, der tilhører bestillingen. Betalingssystemet har SynsPunkt i forvejen andet steds fra. Når betalingen er bekræftet i det andet system, tages 'user' tilbage til salgssystemet og kan afslutte salget der er en del af ordren. Som det kan ses på vores SSD, har vi valgt starte interaktionen med 'Make Order' hændelsen, efterfulgt af et LOOP, dette indikerer at det er muligt at gentage processen af at tilføje en vare, flere gange, før man går videre til at vælge en kunde. Det ligger som et ALT (alternativ), dette er i tilfælde af, at det er en ny kunde som ikke findes i systemet endnu. Så skal der være mulighed for at oprette en kunde med det samme i en ny Dialog, og dernæst vende direkte tilbage til bestillingen. Dette er selvfølgelig ikke nødvendigt, hvis kunden er kendt i systemet. Er det tilfældet, søges der bare på kunden.



Figur 1 Udarbejdet af Emeli

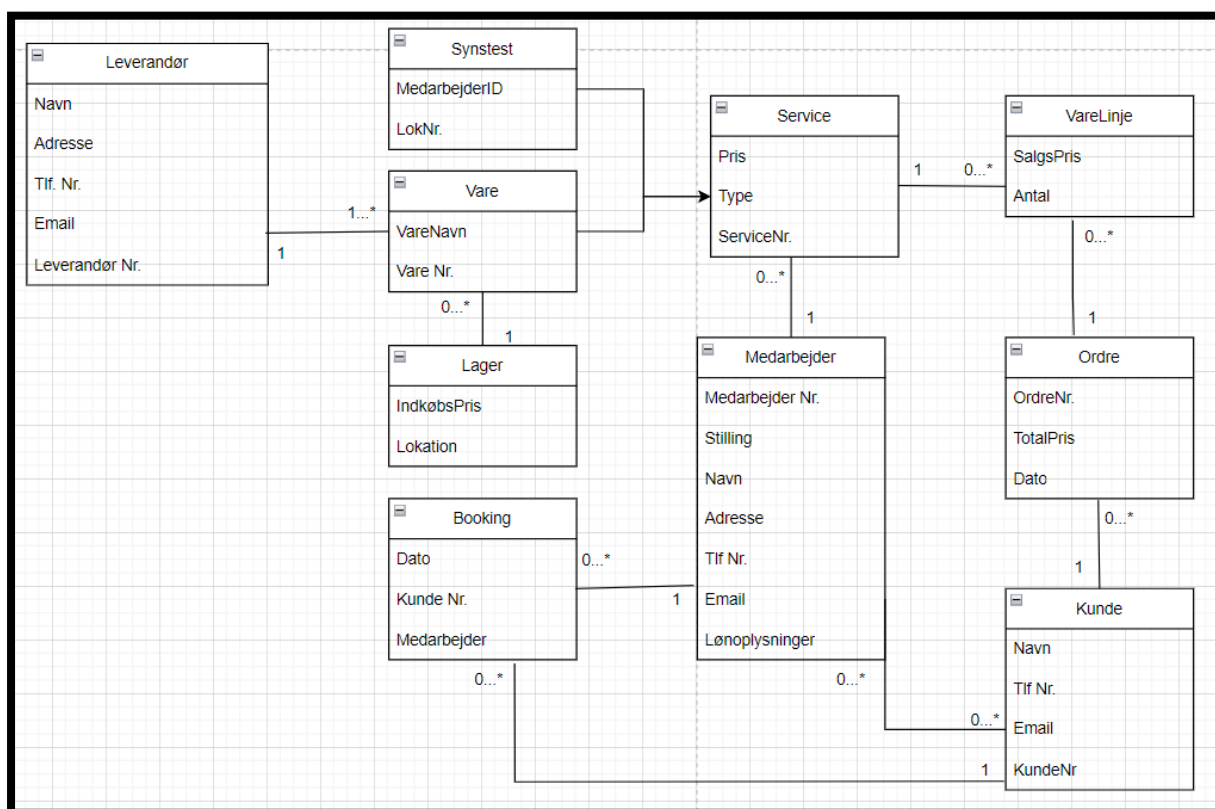
Domænemodel

Sanne

Vores DM består af 10 forskellige klasser – alle med hver deres attributter. Modellen er lavet ud fra vores forståelse af casen om SynsPunkt ApS, og hvordan vi har fortolket den. Vi har derfor arbejdet ud fra at de udbyder en service – den service kan bestå af en vare, eller en synstest. Varen har forbindelse til et varelager og til en leverandør af varen. Dette fordi vi

vurderer at de enkelte forretninger har deres eget lager og at når de bestiller nye varer hjem, er det direkte ved leverandøren.

Denne service har så forbindelse til en medarbejder. En medarbejder har samtidig forbindelse til klassen booking, og både booking og medarbejder har forbindelse til kunden. Dette for at illustrere at vi har antaget at medarbejderen booker for kunden, men at bookingerne er gemt på den enkelte kunde i systemet. Kunden har ydermere forbindelse til ordre, som går videre til varelinje, som slutter ved service. Den sidste del illustrerer salget der er til en kunde i systemet. Domænemodellen har ikke medtaget ”gadekunder”. Men vi har antaget at det forekommer – andet ville være underligt eftersom deres sortiment bl.a. også består af kikkerter – altså varer man blot kan komme ind og købe og ikke behøver at skulle igennem en synstest for at få, eller igennem en speciallavet bestilling, som en brille med styrke er. Disse kunder ville selvfølgelig kunne foretage et køb uden at skulle oprettes som kunde først.



Kravsspecifikation

Emeli og Sanne

For at kunne følge med på konkurrencen, har ejer derfor besluttet at opdatere sin virksomhed og indføre et salgs-IT-system. Dette er gjort med en simpel menu oversigt, hvorfra man kan komme til alle funktioner.

Grundet det manglende overblik har SynsPunkt ApS udtrykt, at de vigtigste funktioner for deres kommende salgssystem er følgende: At kunne strømline information ud til de tre butikker gennem IT-systemet. Dette er grunden til, at forsiden er sat op med to opslagstavler til medarbejderne med informationer de skal have i deres dagligdag. Her informationerne delt op, så der er en "tavle" med medarbejderinformationer, eks. Jubilæum og fødselsdage og en "tavle" med forretningsinformationer, eks. vareændringer, opdateringer i systemerne og generel forretningsinformation. Ejer giver udtryk for at det sociale miljø på arbejdspladsen er vigtigt og vi har derfor prioriteret, at medarbejdertavlen, samt forretningsinformationstavlen er lige synlige.

Derudover skal systemet kunne oprette og gemme kunde- og vareoplysninger i en database, så kunder har mulighed for ens betjening i alle butikker. Medarbejdere skal kunne oprette salg og bestillinger når de står med en kunde i butikken, samt gemme oplysninger på de respektive kunder i butikken. Butikschefen skal kunne tilgå salgsstatistikker for specifikke perioder. Der skal være rettigheder på henholdsvis chef og medarbejdere oprettet i systemet, så det kun er butikschefen, der kan rette vareoplysninger og varesortiment.

Usecase liste

Emeli

Ud fra ovenstående krav fra SynsPunkt har vi udarbejdet en UseCase liste, for at få overblik over alt der skal programmeres. Hver usecase, bør kun tilhøre en winform i projektet. Vi har dog valgt at nogle af vores Winforms har flere funktioner, fx Kundeinformationer. Denne medtager både 'find kunde', 'Rediger kundeoplysninger' og 'Slet kunde'. Grunden til dette er at man ved alle disse skal søge på en eksisterende kunde og tilgå de oplysninger, som ligger i kundens attributter. Derfor er det programmeret således, at man finder kunden og oplysningerne, hvorefter man kan bestemme, hvad der skal ske videre med en knap til henholdsvis 'Slet' og 'Gem'(gemmer redigerede oplysninger). UseCase listen er som følger:

Sanne, Linda, Emeli

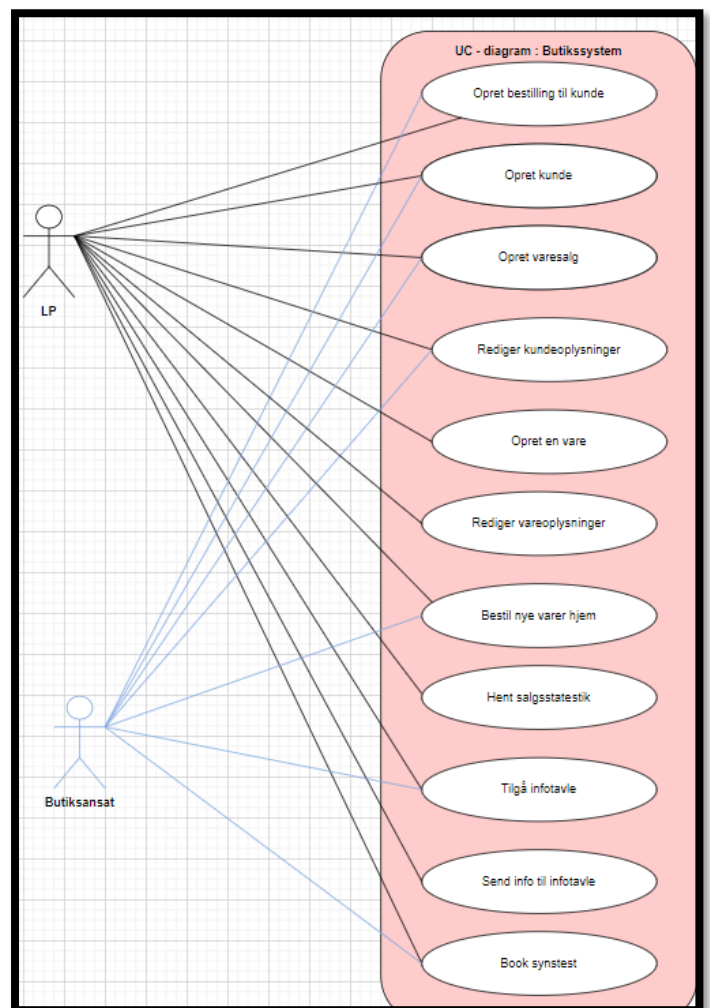
UseCase liste til butikssystem:

1. Opret bestilling til kunde.
2. Opret varesalg.
3. Opret en kunde.
4. Find Kunde
5. Slet kunde
6. Rediger kundeoplysninger.
7. Opret en vare.
8. Find vare.
9. Slet vare.
10. Rediger vareoplysninger.
11. Bestil nye varer hjem.
12. Hent salgsstatistik.
13. Tilgå infotavle.
14. Send info til infotavle.
15. Book synstest.
16. Aflys booking
17. Se bookinger
18. Tilgå Varelager
19. Returnér va

Uc-Diagram

Emeli

Dette har også været med til at strukturere vores arbejdsopgaver og sørge for at systemet er fuldt funktionelt inden tidsfristen for systemets release dato. Der er også lavet casual UseCase beskrivelser til alle usecases og samt fully-dressed usecase til Opret Bestilling, udvalgte vil blive gennemgået længere fremme med den tilhørende UI-form og koden dertil, så de fungerer efter hensigten. Derudover ses vores UC-diagram nedenfor, her er alle de mest relevante usecases placeret i en lyserød boks, med aktørerne: Butiksansat og ejer (LP). Forskellen på de to er, at ejer har tilgang til alle usecases, hvor medarbejderne kun kan tilgå udvalgte. Dette fremgår af diagrammet:



Figur 2 Udarbejdet af Linda

Ikke funktionelle krav

Linda

Ved SynsPunkt har de både funktionelle og ikke funktionelle krav. De ikke funktionelle krav er listet herunder:

| | |
|------------------------|---|
| Ikke funktionelle krav | <ul style="list-style-type: none">- Hurtigt og effektiv- Altid virker + tæt på 100% opetid- Meget brugervenligt- Nemt og meget intuitivt |
|------------------------|---|

Disse ikke funktionelle krav er for at understøtte LP's ønsker og egne udfordringer med IT generelt. LP vil have at systemet er nemt og hurtigt for både ham og hans medarbejdere at lære, samt nemt at lære nye medarbejdere op i.

Scrum

Emeli

Vi har som gruppe valgt at arbejde med SCRUM. Deadline fra vi begyndte at arbejde på projektet udgjorde samlet 4 uger. Derfor valgte vi at have 4 sprint, der hver især havde en varighed på en uge. Dette er ikke lange sprint, men da projektperioden heller ikke er lang, gør det at vi har nemmere ved at fordele og planlægge vores opgaver, med mulighed for at justere mellem hvert sprint. Dermed kan vi sørge for at alt bliver lavet i en rækkefølge der giver mening, og ingen går i stå med programmering fordi dele af et program mangler.

Vi brugte de første to dage i 1. sprint på at planlægge og formulere alle backlog opgaverne, samt hvordan de skulle fordeles ud over de 4 sprint. Som udgangspunkt, var første sprint tiltænkt systemudviklingsaspektet. Da det giver mening at planlægge, hvordan det nye IT-system skal se ud og fungere ved release inden det udvikles. Planen fungerede til start, men fordi det er første sprint, fejlbedømte vi, hvor meget det var muligt, at nå og vi måtte derfor skubbe 2 opgaver til 2. sprint. Dette kom også af at vi fejlbedømte, hvor meget tid der skulle til for at planlægge og strukturere alle arbejdsopgaverne. Det var dog forventet, at første sprint ville fungere som en test, for at afgøre hvor niveauet lå i forhold til, hvor meget det var muligt at nå inden for ét sprint.

2. sprint var tiltænkt oprettelse af database, opbygning af projektfilen, heriblandt oprettelse af praktiske lag-mapper, konstruktion af UI-laget, samt programmering af SQL. Det var dog her tidsplanen skred markant, dette skyldtes at vores GIT-applikation ikke fungerede, som resulterede at vi ikke kunne starte programmeringen. Vi måtte derfor påbegynde skriftlige

opgaver, som vi ellers havde lagt i 4. sprint. Så der ikke var spildtid, mens vi fandt en løsning på problemet.

Udover dette var der uforudsete omstændigheder i 3. sprint, som endte med at planen igen blev ændret og vi blev endnu mere presset på tid, da vi var længere bagud med programmeringsdelen, end vi havde forventet eller givet rum til, at vi kunne nå at rette op på. Vi måtte derfor ved sprintplanlægning til 4. sprint, omlægge hele planen og havde langt flere opgaver end forventet. Vi stod derfor i en situation, hvor sidste sprint, som oprindeligt var meningen, kun skulle indeholde virksomhedsopgaver og rapportskrivning, nu også havde programmerings opgaver, som skulle færdiggøres, for at systemet ville nå at være operativt.

Vandfald vs. Agil

Vi valgte ikke at arbejde ud fra en vandfaldsmetode, fordi vi fra start regnede med at der ville opstå situationer og ændringer. Og derfor ikke ville ende i en situation, hvor vi ikke kunne gå tilbage eller videreudvikle.

Da SCRUM er delvist agil, idet man kan nå at justere arbejdsopgaverne mellem hvert sprint, har vi haft mulighed for at have en læringsudvikling og sørge for, at vi ikke over- eller undervurdere, hvor meget det er muligt, at nå i løbet af en uges sprint. Det endte med at være en god beslutning at lægge 4. sprint i projektperioden, da vi fik meget brug for at om koordinere mellem sprintene, fordi der opstod flere uforudsete problemer allerede fra start, tidsplanen endte dermed også med at bevæge sig i en helt anden retning end oprindeligt planlagt.

Plan for udvikling

User stories

De userstories vi har arbejdet ud fra er baseret på listen for at sikre, at alle deres behov er opfyldt når de skal bruge salgssystemet i dagligdagen, disse har vi derefter samlet i en backlog.

Emeli, Sanne og Linda

| Epic User Story | Normal User Story |
|--|--|
| Som Ejer vil jeg have en varedatabase så der er ens varesortiment på kæden | Som ejer vil jeg kunne oprette nye varer i databasen Så jeg kan opdatere kædens varesortiment |
| | Som ejer vil jeg kunne redigere vareoplysninger Så de altid stemmer overens med gældende ændringer |
| | Som ejer |

| | |
|--|---|
| | vil jeg kunne redigere varens pris Så den altid stemmer overens med min markedsføring |
| | Som ejer vil jeg have statistik på butikkens varesalg Så jeg kan sikre det rette varesortiment |
| | Som ejer Vil jeg have historik på butikkens varesalg Så jeg sikrer økonomisk fremgang |
| | Som ejer vil jeg have statistik på kædens varesalg Så jeg kan sikre det rette varesortiment på kædeniveau |
| | Som ejer Vil jeg have historik på kædens varesalg Så jeg sikrer økonomisk fremgang |

Arkitektur

Linda

Vi har opdelt vores program i tre forskellige lag: præsentationslaget (UI), forretningslogiklaget (Service) og datalaget (Repository). Hvert lag har sine egne ansvarsområder og hjælper med at opretholde en klar adskillelse af opgaver.

Præsentationslaget er det øverste lag i 3-lags strukturen. Det er ansvarligt for brugergrænsefladen og interaktionen med brugeren. Det er her vores Winforms ligger. Dette lag fungerer som bindeleddet mellem brugergrænsefladen og forretningslogiklaget.

Heri ligger forms med navnene:

- HomePageForm
- CreateCustomerForm
- CreateProductForm
- EditCustomerSearchForm
- EditProductSearchForm
- ICreateBookingForm
- ICreateOrderForm
- ICreateSaleForm
- IReturnItemForm
- ISeeAllBookingsForm
- SalesStatisticsForm
- ShowAllProductsForm
- (Button Click Events mm.)

En lille note til navngivningen – de forms med I foran i navnet, er for at genkende de forms, som ikke er funktionelle. De har kun deres winform, og kode der forbinder dem med HomePageForm, men ingen yderligere kode.

Controlleren er et lag mellem præsentationslaget og forretningslogiklaget der er med til at håndterer brugerinput, udfører de nødvendige handlinger og opdaterer visningen efter behov.

Heri ligger klasserne: UIcontroller, CustomerController, ProductController

Forretningslogiklaget er det midterste lag i 3-lags strukturen. Dette lag håndterer forretningsreglerne og logikken i programmet. Det er ansvarligt for at behandle data fra præsentationslaget og træffe beslutninger baseret på disse data. I vores program er det forretningslogiklaget der bl.a. viderestiller til databasen.

Heri ligger klasserne: CustomerServices, ProductServices, SalesstatisticServices

Datalaget er det nederste lag i 3-lags strukturen. Det er ansvarligt for at håndtere dataadgang og -behandling. Det er datalaget der holder CRUD metoderne. I det her tilfælde har vi arbejdet med en lille smule dummydata i vores salgsstatistik, som ligger i en mappe og klasse for sig selv.

Strukturen for dette lag ser derfor således ud:

| Database | Repository |
|------------------|---|
| - DummyDataSales | - CustomerDatabaseSQL - ProductDatabaseSQL |

Derudover har vi en modelmappe, hvor vi har alle de modeller vi bruger i vores program.

Heri ligger følgende modeller: Customer, CustomerOrder, Product

3-lags strukturarkitekturen hjælper med at opnå en bedre organisering og opdeling af kodebasen, hvilket gør det nemmere at vedligeholde, teste og udvide programmet. Den klare adskillelse af bekymringer gør det også lettere at ændre eller erstatte en bestemt del af programmet uden at påvirke de andre lag.

I vores KundeApplikation har vi implementeret en lignende lagstruktur på en mindre skala, hvor vi ikke inkluderer controlleren.

Brugergrænsefladen er repræsenteret af vores form, kendt som "HomePageForm".

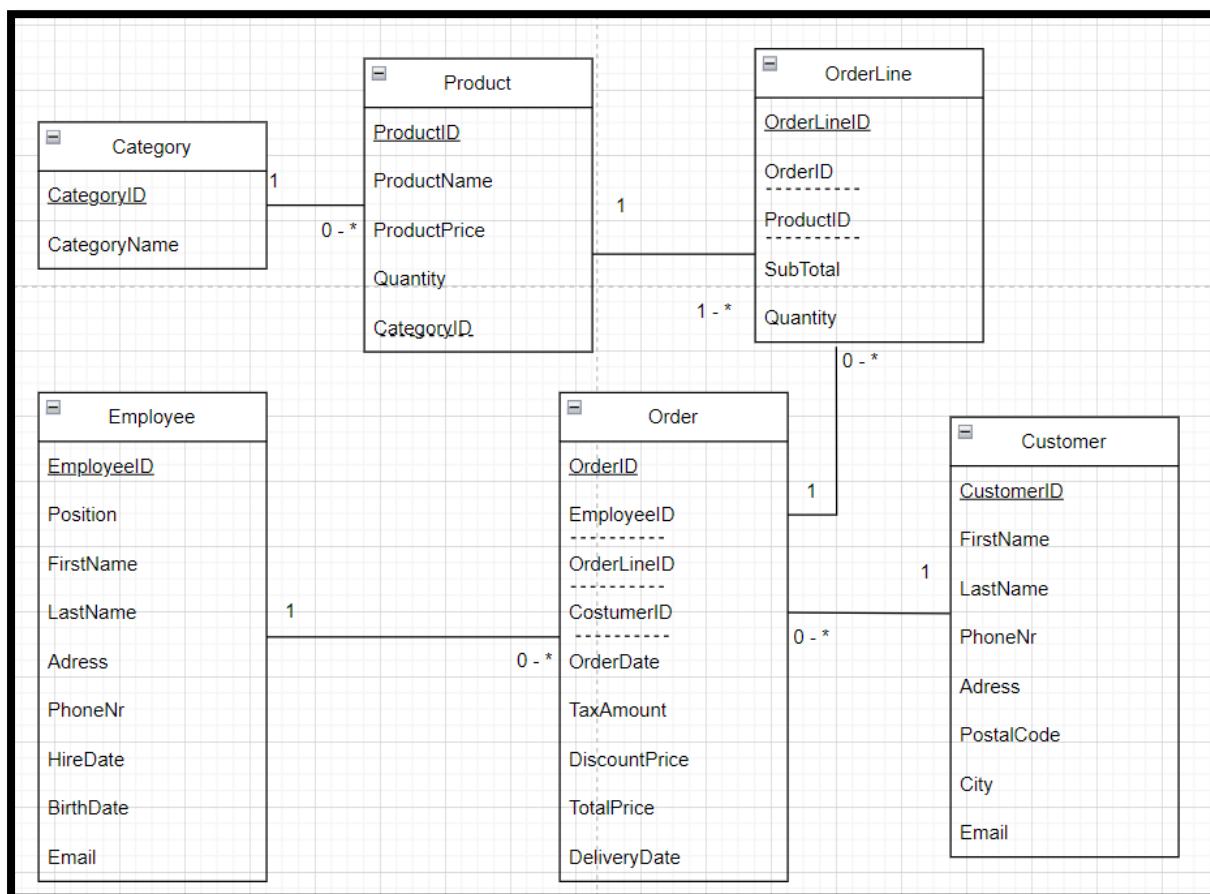
Forretningslogiklaget består af klassen "FrameServices", der indeholder den nødvendige forretningslogik. Databaselaget inkluderer klassen "CustomerAppSQL", som er ansvarlig for interaktion med databasen. Derudover har vi en "models" mappe, hvor vi har en klasse, der indeholder modeller til brug i applikationen, navngivet "Frame".

ER diagram

Emeli

Vi har udarbejdet et ER-Diagram for, visuelt at afbillede, hvordan den kommende database vil komme til at hænge sammen. Et af SynsPunkts krav til systemet er, at samle og opbevarer alle oplysninger i virksomheden, i en database. Dermed er det også nemmere at opsætte databasen, da alle attributter er indskrevet og relationer mellem de forskellige tabeller er bestemt på forhånd. Da vi oprettede databasetabellerne, tog vi udgangspunkt i nedenstående diagram. Vi har ændret en smule i databasen, da denne ikke fungerede med de påskrevne primary keys og foreign keys for at alt fungerede som det skulle. Vi har fjernet primary key på OrderLine og derved dens foreign key på Order.

Alt i databasen er operativt, dog har vi kun valgt at lave SQL og koble 'Order', 'Customer' og 'Product' til programkoden og systemets UI-lag, da vi mener disse er de mest relevante i forhold til de kravsspecifikationer. På ER-diagrammet er attributter som er Primary keys understregede for at tydeliggøre dette. Foreign keys er understreget med stiplede linje.



Figur 3 Dette ER-Diagram er udarbejdet af Sanne, Emeli og Linda

SQL beskrivelser

Sanne

Nedenfor er eksempler på SQL direkte i SSMS-programmet. Disse bliver brugt for at tilføje, ændre, manipulere mm med data i vores database.

SQL-kode der indsætter værdier i eksisterende tabel Customer. Der er valgt hvilke af attributterne der skal tilføjes. I dette eksempel er de alle valgt, med undtagelse af CustomerID, da systemet selv generere denne.

```
INSERT INTO Customer (FirstName, LastName, PhoneNumber, Email, CustomerAddress, PostalCode, City)
VALUES ('Hugh', 'Jackman', '49308230', 'MrWolverine@XMEN.com', 'Rævevejen 29', 6000, 'Kolding');
```

SQL-kode der opretter en ordre i vores Order tabel. Her har vi taget OrderID med for at visualisere at det er den samme ordre, som senere bliver brugt i OrderLine. Alle attributter er taget med, for at visualisere hvad de enkelte værdier betyder.

```
INSERT INTO Orders(OrderID, EmployeeID, OrderDate, TotalPrice, DeliveryDate, CustomerID)
VALUES (10, 2, '2023-01-14', 3188, '2023-01-22', 4);
```

SQL-kode der indsætter ordrelinjer på en ordre. Igen er alle attributter med for at visualisere hvad de forskellige værdier betyder.

```
INSERT INTO OrderLine (OrderID, OrderLineID, ProductID, SubTotal, Quantity)
VALUES (10, 1, 12, 795, 1),
       (10, 2, 13, 795, 1),
       (10, 3, 7, 799, 2);
```

Databasebeskrivelser

Sanne

Vores database består af 6 forskellige tabeller. Category, Customer, Employee, Order, OrderLine og Product.

I vores projekt er der kun forbindelse til Customer, Product og Order. De andre er taget med for at illustrerer vores ER diagram og fordi de er forbundet til hinanden med primær- og fremmednøgler.

Eksempel på SQL-kode, vi har benyttet til at oprette tabellerne:

Det er denne kode vi har brugt til at oprette vores Customer tabel, vi har bare ændret navnet på den senere.

```
CREATE TABLE CustomerTable(  
CustomerID INT IDENTITY(1,1) NOT NULL, PRIMARY KEY (CustomerID),  
FirstName VARCHAR(20) NOT NULL,  
LastName VARCHAR(50) NOT NULL,  
PhoneNumber VARCHAR(20) NOT NULL,  
Email VARCHAR(50) NOT NULL,  
CustomerAddress VARCHAR(50) NOT NULL,  
PostalCode INT NOT NULL,  
City VARCHAR(50) NOT NULL)
```

Bemærk at vi har sat systemet til selv at tildele værdien til CustomerID. Det har vi gjort ved at sætte IDENTITY(1,1) på. Det betyder at den første kunde der bliver oprettet, får CustomerID = 1. og den næste 2 osv. Da den øger med 1 hver gang. Derudover er CustomerID sat til at være primær nøgle. Alle værdierne er sat til at være NOT NULL hvilket betyder at de skal indtastes for at oprette kunden.

```
CREATE TABLE Product(  
ProductID int IDENTITY(1,1) NOT NULL, PRIMARY KEY (ProductID),  
ProductName varchar(50) not null,  
Price decimal(18,2) not null,  
Quantity int not null,  
CategoryID int not null, FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID))
```

Her gør det samme sig gældende med ProductID, det bliver også automatisk oprettet. Samt det er også primær nøgle.

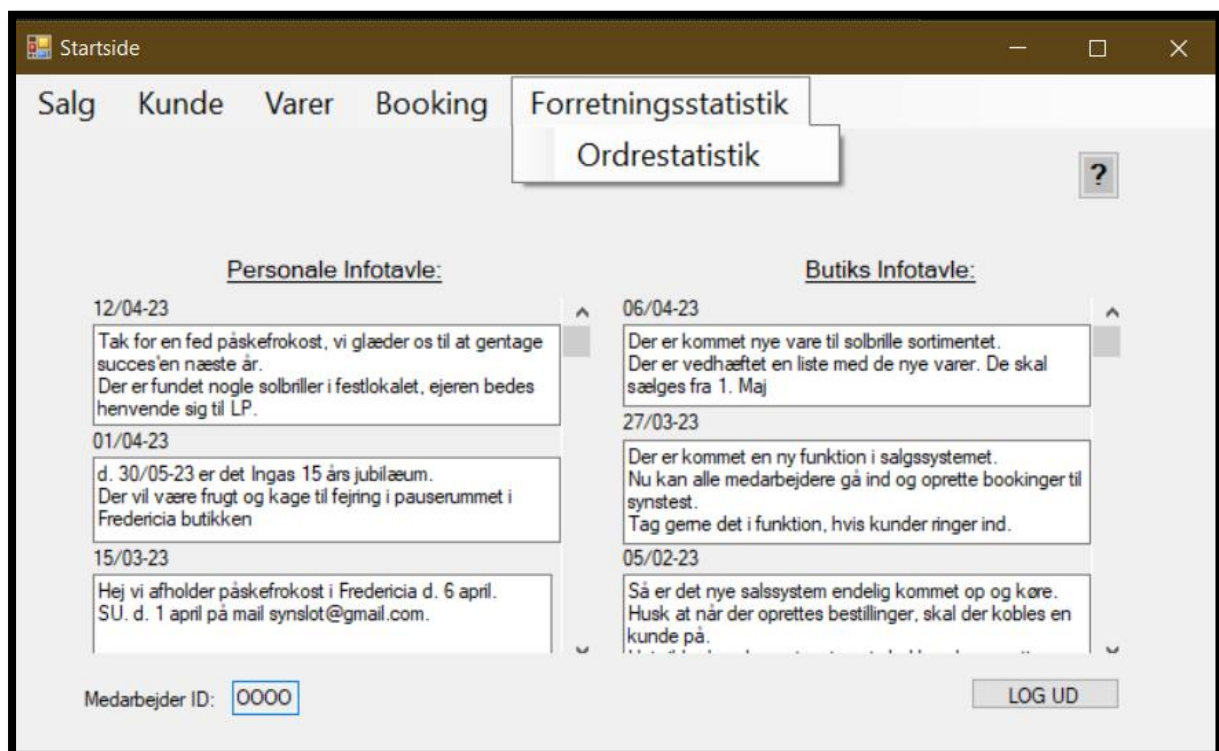
Som noget anderledes har denne en fremmednøgle – hvilket betyder at den er forbundet til tabellen Category med dens primærnøgle CategoryID. Dette for, på sigt, at kunne koble de forskellige varer på forskellige grupper. Dette er desuden et krav fra kunden.

Beskrivelse af færdigt salgssystem

Forsiden

Sanne

Vores system er sat op med Winforms, og disse er forbundet med hinanden. Der er en forside, som er den side man kommer ind på efter man har logget ind med sit eget medarbejder-nr. Ideen er, at der først er en loginside, og fra den kommer man ind på vores forside. Vores forside ser sådan her ud:



På systemets forside sker der flere forskellige ting. Øverst har vi en menu, som kan navigere brugeren videre til de andre forms. Nedenunder menuen ligger der to paneler. Disse bruges til at illustrere LPs mulighed for at kommunikere ud til sine medarbejdere på tværs af alle forretningerne. Der er to forskellige for at tydeliggøre, hvilken type information det drejer sig om. Den første skal LP bruge for at dele informationer om medarbejdere til medarbejdere. Eks. personalefester, jubilæer, fødselsdage og andre sociale arrangementer. Den anden skal LP bruge for at give informationer om selve virksomheden, som medarbejderne skal bruge i deres hverdag. Eks. prisændringer, ændringer i sortiment, ændringer og opdateringer i deres system osv.

Nederst til venstre vises medarbejder ID'et på den medarbejder der er logget ind. (0000) visualisere at det er rent output i feltet, og derfor ikke noget brugeren skal bruge – når programmet bliver taget i brug, vil det derfor ikke viser 0000, men være fyldt ud med den pågældende medarbejders ID.

Til højre er der en knap, som på nuværende tidspunkt lukker programmet, men ideen er at den logger medarbejderen ud, og returnere til login siden, så næste medarbejder nemt kan logge ind bagefter.

Som en hjælp til brugerne er der øverst til højre en knap med et spørgsmålstegn på – denne åbner en PDF op, som er en hjælpe-side der forklarer, hvad der sker i det gældende vindue – denne knap er på alle vores funktionelle forms i programmet, og åbner forskellige PDF'er op. Når man klikker på et af felterne i menuen, åbner den en fold-ud-menu, hvor man kan vælge hvad man ind på – uden at man forlader den første form – på den måde undgår vi frustrationer, over at man i første omgang klikkede forkert ind og så skal tilbage og klikke igen.

Menupunkterne Kunder, Varer og Forretningsstatistik er alle 3 fuldt funktionelle og er kodet færdige, her kan man oprette kunder og varer, de kan redigeres og slettes igen, og man kan få vist en oversigt over alle varerne. Oversigten kan ses både på skærmen og den kan printes ud til en .txt fil. I forretningsstatistik kan man søge på en specifik tidsperiode og så få vist hvor mange salg der har været i denne periode – denne kan også printes til .txt fil.

Programkode

Kodemæssigt er der ikke meget på forsiden. Den forbinder hovedsageligt kun til andre forms – sådan at når man klikker på et menupunkt så kan den åbne en ny form. Det ser således ud:

```
private void opretKundeToolStripMenuItem_Click(object sender, EventArgs e)
{
    CreateCustomerForm createCustomerForm = new CreateCustomerForm();
    createCustomerForm.ShowDialog();
}
```

Når man klikker på menupunktet, laver koden en ny instans af CreateCustomerForm, og af den kalder vi en indbygget metode, der hedder ShowDialog. Denne metode gør, at brugeren skal gøre sit arbejde færdig i den pågældende form. Man kan altså ikke være i gang med at oprette en kunde og så minimere det vindue for at åbne et nyt i samme program.

Salg

Emeli

Under menupunktet 'Salg' er mulighederne: Opret Salg, Opret Bestilling og Returnér Vare. Henholdsvis salg og bestilling, har vi valgt at skille ad, da det nævnes at der udover briller og kontaktlinser også sælges kikkerter. Yderligere antages det, at der sælges andre produkter såsom kontaktlinsevand, brille etuier, pudseklude, solbriller mm. Det skal derfor være muligt at oprette salg på kunder fra gaden. Disse kunder skal ikke nødvendigvis have bestilt briller eller kontaktlinser, som skal produceres eller laves først ud fra deres specifikke synstest og dermed er en bestilling ikke nødvendig.

Som det ses på Formen 'Opret Bestilling', er denne form lavet på baggrund af usecase nr. 1. Dette er en af de ikke funktionelle winforms i systemet.

Denne skal gøre det nemmere og ensartet for alle medarbejdere, at oprette bestillinger. Dette gøres, når en kunde har fået taget en synstest, fundet styrke, og i fællesskab med medarbejder, har fundet enten briller eller kontaktlinser, der skal bestilles hjem med de specifikationer der er tilpasset kunden. Briller er ikke til rådighed, medarbejder og leverandør er først fremstille og tilpasse brillerne med glas. Det samme gælder for kontaktlinser, som også skal bestilles hjem fra producenten.

Systembrugeren opretter en bestilling gennem tre 'trin'. Find vare, tilføj vare til bestilling, vælg kunde. Først indtastes et varenummer eller navn, hvorefter de varer der indeholder input i deres properties, vises i et datagridview. Når en vare vælges i datagridviewet, sender systemet varens informationer over i gruppeboksen 'Vare information', hvor alt er output ud fra de oplysninger som findes i systemet. Det er dermed muligt at se, om den valgte vare er den rigtige, samt om der er flere på lager i butikken. Hvis ikke, vælges en anden vare i datagridviewet. Når den rigtige vare er fundet, kan antal af denne indtastes i en inputtekstboks markeret med 'I', hvorefter der vises pris i output tekstboks markeret med 'o' nedenfor.

Når informationerne stemmer trykkes på knappen 'Tilføj til Bestilling'. Denne proces kan gentages for alle varer, som selve bestillingen skal bestå af. Når alle varer er tilføjet til bestillingen, kan en kunde kobles på ordren. Dette gøres på samme måde som en vare. En input tekstboks tager enten kundenummer eller kunde navn, derefter kan den respektive kunde

vælges. Hvis ikke kunden er oprettet i systemet, kan der trykkes på Opret Kunde knappen, som åbner Opret Kunde vinduet, hvorefter den oprettede kunde kan findes.

Vi har valgt at skrive en fully-dressed usecase beskrivelse til denne UseCase, der gøres opmærksom på at brugergrænsefladen blev lavet ud fra den casual beskrivelse, men den fully-dressed blev uddybet, og vi fandt frem til at levering også var essentiel. Derfor er dette leveringsvindue ikke lavet.

| | |
|-----------------------------|---|
| UseCase UC1: | Opret bestilling til kunde |
| Scope: | Salgssystem |
| Level: | User goal |
| Primary Actor: | Medarbejder i SynsPunkt |
| Stakeholders and Interests: | <ul style="list-style-type: none">- Kunde: Vil kunne gå i alle kædens butikker, og få samme optimale og hurtige hjælp, som i deres lokale butik. Og dermed ikke gå forgæves i butikkerne.- Medarbejder: vil gerne kunne oprette, tilgå tidligere ordre på alle kunder, samt kunde information, ved at have et fælles system for hele kæden. Dermed kan man spare tid og ressourcer, når man hjælper en kunde.- Ejer: Vil gerne kunne hente alle ordre i salgsstatistikker, have ordrene samlet på deres respektive kunder, så det giver optimale forhold for at kunne tilfredsstille alle kunder som kommer ind i butikkerne. |
| Preconditions: | Kunden skal være oprettet i systemet. |
| Postconditions: | Bestillingen er blevet gemt under den rigtige kunde i systemet og kan tilgås i alle kædens butikker. Varelageret er blevet opdateret. Bestillingen tages med i de nye salgsstatistikker. |
| Main succes scenario: | <ul style="list-style-type: none">- Medarbejder opretter ny bestilling- Tryk på "Opret Bestilling" i StartMenuen i systemet- Ny skærm åbner op.- Indtast/søg efter den/de valgte vare(r)- Tjek rette vare er valgt i output vare gruppeboks.- Tilføj varen til bestillingen, som er aftalt med kunde.- Gør dette indtil alle varer er tilføjet. |

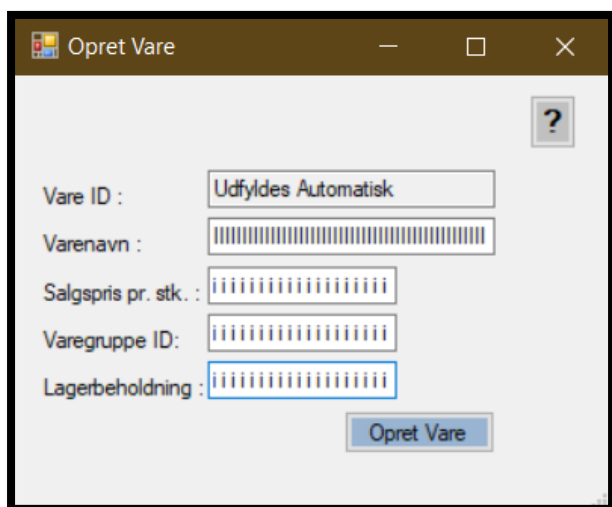
| | |
|-------------|--|
| | <ul style="list-style-type: none">- Indtast/ søg efter kundeoplysninger- Vælg specifik kunde, som ordren skal oprettes på.- Vælg OK – pop-up boks der spørger om alt indtastet er korrekt.- Systemet bekræfter valgt vare, samt kunde.- Vælg levering/afhentning i butik (ved briller kun afhentning)- Oplys kunden om leveringstid (skal dukke op automatisk i systemet)- Vælg OK – pop-up boks der spørger om alt indtastet er korrekt.- Tag imod betaling- Tryk knap: Afslut Bestilling. |
| Extensions: | <ol style="list-style-type: none">1. Kunden beder om at få afbrudt bestillingen, fordi kunde fortryder.<ul style="list-style-type: none">- Medarbejder afbryder og sletter bestillingen.- System kommer med pop-up og beder om bekræftelse.2. Kunden findes ikke i systemet.<ul style="list-style-type: none">- Medarbejder gemmer bestilling som kladde.- Går ud på forsiden, trykker Opret kunde.- Efter kunde er oprettet, genoptager medarbejder bestillingen.3. Vare kan ikke findes i systemet.<ul style="list-style-type: none">- Medarbejder, tjekke oplysninger igen.- Hvis der stadig er fejl, prøves anden søge-metode varenavn, vareID.- Hvis fejl ikke findes, kontaktes en medarbejder med flere rettigheder i system.- Ny medarbejder kan tjekke varesortiment.4. Bestillingen fejler og skal genoptages.<ul style="list-style-type: none">- Bestilling er gemt i kladder og kan genoptages, når systemet er oppe igen.- Er kladden ikke gemt, foretages alle trin i bestilling af vare igen.5. Kunde fortryder enkelt vare og beder om det fjernet. |

| | |
|--|---|
| | <ul style="list-style-type: none">- Medarbejder åbner varevinduet.- Vælger specifik varer som skal tages af bestilling.- Fjerner varer, og viser ny total pris. <p>6. Prisen på et produkt er slået ind forkert og skal redigeres.</p> <ul style="list-style-type: none">- Kontakt ejer, som kan give rettigheder til ændring af pris.- Hvis medarbejder ikke har rettigheder til prisændring, tag fat i medarbejder der har.- Vælg vare i varevinduet- Rediger pris.- - Vis ny total pris. |
|--|---|

Varer

Sanne

Under menupunktet 'Vare' er alle forms fuldt funktionelle, de er følgende: Opret Vare, Vareoplysninger og Vis alle Varer



Opret Vare

Denne kan, som navnet antyder, oprette varer i databasen. Det er denne form ejeren skal bruge for at oprette og gemme varens informationer i databasen. Der er flere felter, som skal udfyldes. Det første felt skal dog ikke indtastes, da den tabel i databasen er sat op til selv at tildele nyt varenr. til hver ny vare. Vi har det med i formen, for at illustrere at varen også får et varenr. Der er derfor skrevet i feltet at det automatisk udfyldes. De næste felter giver sig selv – vi har i

illustrationen sat i og Iér ind, så det er tydeligt hvilke felter der er numeriske og hvilke felter der er alfanumerisk.

| ProductID | ProductName |
|-----------|--------------------------|
| 8 | Linsevæske 360ml, Valmed |
| 6 | Softlens Kontaktlinser |
| | |

Use Case beskrivelse

| | |
|-----------|---|
| | Opret en vare. |
| Aktør(er) | LP (Ejer) |
| Handling | <ul style="list-style-type: none">- Tryk på "opret vare" i systemet- Indtast alle oplysningerne om varen- Pop-up boks der spørger om alle oplysninger er korrekte- Gem og OK |

Vareinformationer

I denne kan man søge i hele varedatabasen og finde frem til den vare man godt vil se. I søgefeltet kan man skrive både tal og bogstaver, og derfor søge på både navn og varenr. Realistisk set, ville de fleste nok søge på navnet – men her er muligheden også for at søge på varenr.

Når man har markeret en vare i datagridviewet, kan man klikke på "Vis Vareoplysninger" og få vist alle informationerne i tilhørende tekstfelter.

Når varens informationer står i felterne, kan de opdateres og slettes hvis det ønskes. Når man har klikket på gem eller slet vare, kommer der en dialogboks op og spørger om man er sikker på ændringerne. Her klikkes der OK eller Cancel/Annuller. Vælges OK bliver rettelserne opdateret ned i databasen, vinduet lukkes og brugeren vender retur til forsiden. Vælges i stedet Cancel/Annuller, vendes der retur til denne form, så man kan klikke/rette korrekt.

Programkoden/SQL - Vare

Som nævnt længere oppe er vores kode delt op i lag. Dette betyder at alt vores forretningslogik, som udgangspunkt, ligger i vores service lag. Servicelaget er forbundet med vores repository, som har kontakt med vores database. Vi illustrerer her processen med den ene af metoderne – CreateProduct.

Sådan ser koden ud i vores Controller lag – øverst oppe i Visual Studio har vi brugt using ButiksSystem.Models dette gør at vi ikke behøver at skrive Models.Product, hver gang vi referer til vores ProductClass, som ligger i Models.

```
public void CreateProduct(string productName, decimal price, int categoryId, int quantity)
{
    Product product = new Product(productName, price, categoryId, quantity);
    ProductServices productServices = new ProductServices();
    productServices.CreateProduct(product);
}
```

Også i servicelaget har vi i toppen skrevet using ButiksSystem.Models

```
public void CreateProduct(Product product)
{
    ProductDatabaseSQL productDatabaseSQL = new ProductDatabaseSQL();
    productDatabaseSQL.CreateProduct(product);
}
```

I vores repository har vi igen gjort det samme, skrevet using ButiksSystem.Models

```
public void CreateProduct(Product product)
{
    string query = $"INSERT INTO Product (ProductName, Price, CategoryID, Quantity) " +
        $"VALUES (' " +
        $"{product.ProductName}', " +
        $"{product.ProductPrice.ToString().Replace(',', '.')}, " +
        $"{product.CategoryID}, " +
        $"{product.Quantity})";

    SqlCommand command = new SqlCommand(query, connection);
    connection.Open();
    command.ExecuteNonQuery();

    connection.Close();
}
```

Som det ses her, har vi skrevet selve metoden i repository, og blot kaldt dem op igennem lagene. Dette giver et bedre overblik over hele koden, når det hele er skrevet de samme steder.

I netop denne metode ligger alt vores logik i repository i stedet for service, det gør det fordi det er en af vores CRUD metoder, hvor der bruges SQL, og det er alt sammen samlet i repository.

Programkoden/SQL – Vareinformationer

Her er koden også delt i lag, hvor lagene har forbindelse med laget lige over og lige under.

Controller:

```
public void UpdateProduct(Product product)
{
    ProductServices productServices = new ProductServices();
    productServices.UpdateProduct(product);
}
```

```
public void DeleteProduct(Product product)
{
    ProductServices productServices = new ProductServices();
    productServices.DeleteProduct(product);
}
```

Servicelaget:

```
public void UpdateProduct(Product product)
{
    ProductDatabaseSQL productDatabaseSQL = new ProductDatabaseSQL();
    productDatabaseSQL.UpdateProduct(product);
}
```

```
public void DeleteProduct(Product product)
{
    ProductDatabaseSQL productDatabaseSQL = new ProductDatabaseSQL();
    productDatabaseSQL.DeleteProduct(product);
}
```

Repository:

```
public void UpdateProduct(Product product)
{
    string query = $"UPDATE Product SET " +
        $"ProductName = '{product.ProductName}', " +
        $"Price = '{product.ProductPrice}', " +
        $"CategoryID = '{product.CategoryID}', " +
        $"Quantity = '{product.Quantity}' " +
        $"WHERE ProductID = '{product.ProductID}'";

    SqlCommand command = new SqlCommand(query, connection);
    connection.Open();
    command.ExecuteNonQuery();

    connection.Close();
}
```

```
public void DeleteProduct(Product product)
{
    string query = $"DELETE FROM Product WHERE ProductID = '{product.ProductID}'";

    SqlCommand command = new SqlCommand(query, connection);
    connection.Open();
    command.ExecuteNonQuery();

    connection.Close();
}
```

Det er i disse to den egentlige kode ligger. Af samme årsag som i CreateProduct, så er det metoder bestående af SQL og deres logik ligger derfor i det nederste lag. Disse SQL sætninger er CRUD, som er basis metoder at tage i brug når der skal benyttes databaser.

Igen er ProductID ikke en variabel der kan redigeres i, men i stedet den programmet pejler efter. Denne metode fungerer på samme facon som CreateProduct. Den tager input fra UI i tekstbokse og gemmer disse info i product, som bliver taget med ned til Repository.

Metoden er lavet sådan at alle informationerne om varen bliver automatisk skrevet i tekstboksen i UI-laget, og så kan brugeren ændre i de ønskede felter. Når brugeren trykker på gem, gemmer programmet alle de informationer der ligger i tekstboksene og tager alle disse informationer med videre – på denne måde opdaterer databasen varen med samtlige informationer, også selvom nogen af dem evt. er de samme som før.

I DeleteProduct er der brugt en enkelt linje kode – denne kode kigger på hvilket varenr der er gemt i 'product' og sletter så alt info der hører til den. Som nævnt tidligere, så trækker metoden infoen om product ud fra Models.Product – det er denne som gemmer på informationerne hele vejen igennem koden – metoden bliver kaldt slavist op igennem hvert lag – som illustreret ovenfor.

Use Case:

| | Rediger vareoplysninger. |
|-----------|---|
| Aktør(er) | LP (Ejer) |
| Handling | <ul style="list-style-type: none">- Tryk på "redigér vare" i systemet- Tryk på det valgte felt som skal redigeres- Gentag ved alle felter der skal redigeres- Vælg OK- Pop-up boks der spørger om alle oplysninger er korrekte- Gem og luk |

Vis Alle Varer

Sanne

Denne form viser automatisk alle varerne der er i databasen. Hvis brugeren har brug for et udprint af alle varerne, kan der klikkes på knappen Print Alle Varer Til txtfil. Og denne .txt fil gemmes derefter automatisk i debug mappen tilhørende projektet.

Programkoden/SQL

I stedet for at koble databasen

direkte på vores datagridview har vi i denne form, valgt at vise, hvordan det ville se ud, med kode der kobler en SQL READ fra vores repository op igennem lagene til UI.

Vi tager udgangspunkt i koden til datagridviewet og knappen der printer listen til .txt-fil.

```
private void ShowAllProductsForm_Load(object sender, EventArgs e)
{
    ProductController productController = new ProductController();
    List<Product> products = productController.GetAllProducts();
    dgv_showProductInfo.DataSource = products;
}
```

```
private void btn_printAllProductsToTxtfile_Click(object sender, EventArgs e)
{
    ProductController productController = new ProductController();
    productController.PrintDataGridViewToFile(dgv_showProductInfo);

    this.Close();
}
```

Disse to stykker kode viser, det samme som vist tidligere, at der laves nye instanser af de forskellige klasser igennem lagene, som der her forbindes imellem. I ShowAllProductsForm vises yderligere at datagridviewet skal vise indholdet af products, som er en liste af alle produkterne i databasen.

```
public List<Product> GetAllProducts()
{
    ProductServices productServices = new ProductServices();
    productServices.GetAllProducts();

    return productServices.GetAllProducts();
}
```

```
public void PrintDataGridViewToFile(DataGridView dataGridView)
{
    ProductServices productServices = new ProductServices();
    productServices.PrintDataGridViewToFile(dataGridView);
}
```

| | ProductID | ProductName | ProductPrice | Quantity | CategoryID |
|---|-----------|-----------------------|--------------|----------|------------|
| ▶ | 10 | Alm sort stel, Syn... | 700 | 66 | 1 |
| | 12 | Basic Grant Stel | 795 | 20 | 1 |
| | 13 | Basic Stel Red | 795 | 30 | 1 |
| | 5 | Cartoon Red Brill... | 6999 | 20 | 1 |
| | 9 | Kontaktlinse etui,... | 30 | 298 | 7 |
| | 3 | Leopard Monster... | 59 | 53 | 7 |
| | 8 | Linsevæske 360... | 40 | 75 | 6 |
| | 7 | Panaramic Glass,... | 799 | 0 | 5 |
| | 2 | Rayban Diva | 799 | 25 | 3 |

I disse to forbindes der fra controller til services

```
public List<Product> GetAllProducts()
{
    ProductDatabaseSQL productDatabaseSQL = new ProductDatabaseSQL();
    productDatabaseSQL.GetAllProducts();

    return productDatabaseSQL.GetAllProducts();
}
```

I denne bliver der forbundet ned til repository

```
public void PrintDataGridViewToFile(DataGridView dataGridView)
{
    string filePath = ("printallproducts.txt"); //Txt filen kommer til at ligge direkte i denne sti,
    using (StreamWriter writer = new StreamWriter(filePath))
    {
        // Write column headers to the file
        foreach (DataGridViewColumn column in dataGridView.Columns)
        {
            writer.Write(column.HeaderText.PadRight(32));
        }
        writer.WriteLine();

        // Write rows to the file
        foreach (DataGridViewRow row in dataGridView.Rows)
        {
            foreach (DataGridViewCell cell in row.Cells)
            {
                writer.Write(cell.Value.ToString().PadRight(32));
            }
            writer.WriteLine();
        }
    }
}
```

Denne ligger i service og det er heri logikken for, hvordan vi har kodet selve .txt-filen ligger. Først navngives en string variabel 'filePath', som skal have værdien af filens placering. Den er skrevet ind kun med navnet på filen og på den måde bliver den lagt direkte i debugmappen. StreamWriter er en indbygget klasse, som har sine egne funktioner vi her gør brug af for at oprette filen.

I det første foreach statement bliver der beskrevet, hvad overskrifterne på kolonnerne skal være – de er det samme som i databasen, og til slut bruges PadRight, som diktere, med tal-indikateren bagved, hvilken bredde der skal være imellem kolonnerne.

I det andet foreach statement bliver der beskrevet hvad der skal fyldes i selve teksten – her er det de enkelte celler i tabellen fra databasen der skal sættes ind. Da nogen af disse indeholder numeriske værdier benytter vi ToString metoden for at konvertere dem til strings så de kan bruges heri. Igen benyttes PadRight for at opsætte filen pænt.

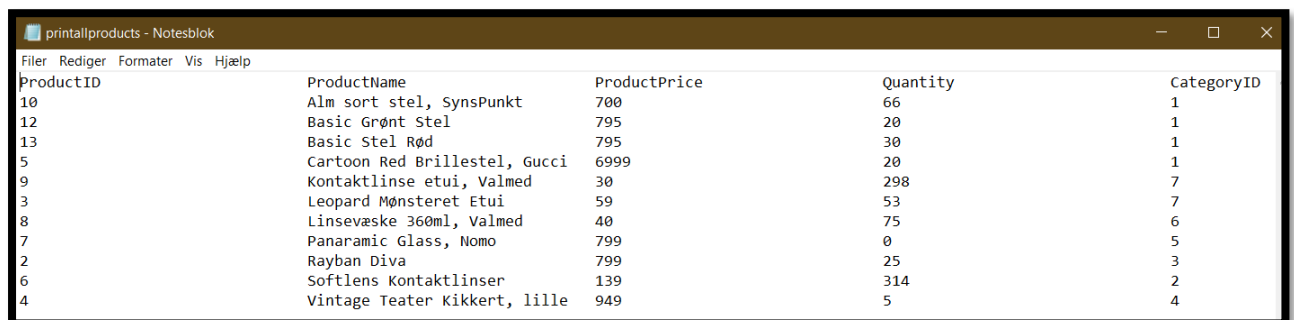
Metoden printer resultatet i datagridviewet ud, og har derfor ikke direkte adgang til databasen, og derfor heller ikke vores repository.

```
public List<Product> GetAllProducts()
{
    List<Product> result = new List<Product>();
    string query = "SELECT ProductID, ProductName, Price, CategoryID, Quantity FROM [Product] ORDER BY ProductName ASC";
    SqlCommand command = new SqlCommand(query, connection);
    connection.Open();
    SqlDataReader reader = command.ExecuteReader();

    while (reader.Read())
    {
        int productID = reader.GetInt32(0);
        string productName = reader.GetString(1);
        decimal productPrice = reader.GetDecimal(2);
        int categoryID = reader.GetInt32(3);
        int quantity = reader.GetInt32(4);
        Product product = new Product(productID, productName, productPrice, categoryID, quantity);

        result.Add(product);
    }
    reader.Close();
    connection.Close();
    return result;
}
```

Ovenstående metode, bruges til at vise data i datagridview, er en af CRUD-metoderne. Dette er en SELECT statement som laver en liste over de valgte attributter fra klassen Product og derefter sortere listen alfabetisk på productnavnet. Selve metoden returnerer til sidst listen, som visuelt kan ses i formen.



| ProductID | ProductName | ProductPrice | Quantity | CategoryID |
|-----------|-------------------------------|--------------|----------|------------|
| 10 | Alm sort stel, SynsPunkt | 700 | 66 | 1 |
| 12 | Basic Grønt Stel | 795 | 20 | 1 |
| 13 | Basic Stel Rød | 795 | 30 | 1 |
| 5 | Cartoon Red Brillestel, Gucci | 6999 | 20 | 1 |
| 9 | Kontaktlinse etui, Valmed | 30 | 298 | 7 |
| 3 | Leopard Mønsteret Etui | 59 | 53 | 7 |
| 8 | Linsevæske 360ml, Valmed | 40 | 75 | 6 |
| 7 | Panaramic Glass, Nomo | 799 | 0 | 5 |
| 2 | Rayban Diva | 799 | 25 | 3 |
| 6 | Softlens Kontaktlinser | 139 | 314 | 2 |
| 4 | Vintage Teater Kikkert, lille | 949 | 5 | 4 |

Salgsstatistik fil

Emeli

I denne form kan ejer udtrække salgsstatistikker fra en hvilken som helt tidsperiode. Dette er gjort med to 'datetime pickers' der repræsenterer henholdsvis en startdato og en slutdato, hvor programmet, så filtrere alle ordre og returnere en liste i en tekstfil. Tekstfilen indeholder de udvalgte salg, stillet op i et tabelview. Første kolonne repræsenterer Kunde ID, derefter kommer kunde navn, dato for salget og sidst købet. I bunden er alle køb, som er

| CustomerID | CustomerName | OrderDate | TotalPrice |
|------------|-----------------|------------|------------|
| 16 | Peter Jensen | 15-03-2023 | 8500 |
| 47 | Johs Møller | 18-03-2023 | 5020 |
| 69 | Henrik Skov | 24-03-2023 | 7540 |
| 18 | Camilla Ottesen | 10-04-2023 | 3990 |
| 21 | Maria Petersen | 25-04-2023 | 6370 |

trukket ud, blevet summeret og viser total summen for hele den valgte periode. 'User' vælger først datoer, dernæst trykkes på knappen 'Vis alle salg i valgt periode'. Dette sortere i datagridviewet, som fra form load fyldes med data. Vi har valgt at lave dummy data i stedet for at opkoble dette på databasen, det er alt sammen placeret i en metode 'CreateDummyDataSales'. Den ligger mappen: Database i Salgssystem projektet. Grunden til dette er at vi mener, at vi allerede har vist SQL, denne ville også benytte en Read metode. Et eksempel på Dummy data kan ses nedenfor:

```
public static List<CustomerOrder> CreateDummyDataSales()
{
    List<CustomerOrder> listOfSales = new List<CustomerOrder>();

    var orderDate = DateTime.Now.Date;
    Customer customer = new Customer(
        "Eigil",
        "Olsen",
        "+45 12 34 56 78",
        "Boulevarden 25",
        7100,
        "Vejle",
        "Hej22@gmail.com", 2);
    CustomerOrder sale1 = new CustomerOrder(customer.FirstName + " " + customer.LastName, 10, orderDate, 4534);
}
```

Rækkefølgen er bestemt af CustomerOrder Modellens constructor for at gemme oplysningerne modellen properties.

```
public CustomerOrder( string customerName, int costumerID, DateTime orderDate, decimal totalPrice)
{
    TotalPrice = totalPrice;
    OrderDate = orderDate.Date;
    CustomerID = costumerID;
    CustomerName = customerName;
}
```

Ovenfor ses omtalte constructor. Metoden der sorterer på datoerne, tager to parametre og returnere en liste. De to datetime værdier bliver sendt med fra brugergrænse fladen. Der oprettes først en ny liste, som er listen der returneres i 'CreateDummyDataSales' metoden. Datetime værdierne sammenlignes så med alle de datetime værdier der tilhører alle salg. Til dette bruges LINQ where(); metoden. Den ser således ud:

```
public class SalestatisticServices
{
    /// <summary>
    /// Creates a list of DummyData within specific timeinterval, that contains customerOrders.
    /// Sorts the list by orderdate.
    /// </summary>
    /// <param name="startdate"></param>
    /// <param name="enddate"></param>
    /// <returns></returns>
    1 reference
    public List<CustomerOrder> GetCustomerOrdersWithinGivenTime(DateTime startdate, DateTime endDate)
    {
        var listOfCustomerOrders = DummyDataSales.CreateDummyDataSales();

        if (startdate == default && endDate == default)
        {
            return listOfCustomerOrders.Where(x => x.OrderDate == DateTime.Today).ToList();
        }
        else
        {
            var ordersWithinTimePeriod = listOfCustomerOrders.Where(x =>
                x.OrderDate >= startdate && x.OrderDate <= endDate).ToList();
            return ordersWithinTimePeriod.OrderBy(x => x.OrderDate).ToList();
        }
    }
}
```

Læg mærke til at der her er et tydeligt eksempel på fejlhåndtering. En try/Catch bruges, den sørger for at et scenarie hvor en user ikke indtaster datoer men bare trykker direkte på 'Vis...' knappen, så vælges dagsdato og alle salg derfra, vises i stedet. Herefter skal alle de udvalgte Salg udskrives i en tekstfil. Metoden der gør dette ser sådan ud:

```
public void CreateSalesFile(List<CustomerOrder> listOfCustomerOrders, DateTime startDate, DateTime endDate, DataGridView dataGridView)
{
    try
    {
        List<CustomerOrder> customerOrders = listOfCustomerOrders;

        string FilePath = ("Salgsstatistik.txt"); //Prints the file directly in the debugger folder - so all can access it.

        using (StreamWriter writer = new StreamWriter(FilePath))
        {
            writer.WriteLine("SALGSSTATISTIK 2023" + "          Fra dato: " + startDate + "          Til Dato: " + endDate);
            string salesfileHeadings = "\nKundenummer    Kundenavn          Dato          Køb\n"; ;
            writer.WriteLine(salesfileHeadings);

            if (listOfCustomerOrders == null)
            {
                SaleStatisticServices.CreateSalesFileDataGridView(dataGridView, startDate, endDate);

                return;
            }
            foreach (var item in listOfCustomerOrders)
            {
                writer.WriteLine(string.Format("{0}          {1}          {2}          {3}",
                    item.CustomerID, item.CustomerName, item.OrderDate.Date, item.TotalPrice.ToString()));
            }

            decimal sumOfPrices = listOfCustomerOrders.Sum(x => x.TotalPrice);
            writer.WriteLine("\n          I alt kr.          " + sumOfPrices);
        }
    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message, "Der er sket en fejl, data kunne ikke overføres til fil", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Her er fejlhåndteret 2 gange. Først i en Try/Catch. Her kører den først koden, fejler det, åbnes en error messagebox. Denne Oplyser 'user' om at der er sket en fejl og smider en exception. I det stykke af kode ,som først testes er et if-statement. Den sørger for at programmet håndterer et tilfælde hvor, 'user' gerne vil have den fulde liste ud og derfor, slet ikke indtaster dato eller beder om at få vist inden for et tidsrum. Dette er muligt da den i stedet kalder en anden metode ' CreateSalesFileDataGridView', den printer alle objekter fra Datagridviewet til en fil der i stedet kaldes "AlleSalg". Metoden vises herunder:


```
public static void CreateSalesFileDataGridView(DataGridView dataGridView, DateTime startDate, DateTime endDate)
{
    StreamWriter file = new StreamWriter("AlleSalg.txt");
    try
    {
        //The document starts up here, sets up main headings and information.
        file.WriteLine("SALGSSTATISTIK 2023" + "          Fra dato: " + startDate.Date + "          Til Dato: " + endDate.Date);
        file.WriteLine("Kundenummer    Kundenavn          Dato          Køb");

        string sLine = "";

        if (dataGridView.RowCount == 0 || dataGridView.ColumnCount == 0)
        {
            MessageBox.Show("Der er ikke nogle data i Datagridview, Intet overføres til fil", "Fejlmeddelse", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            for (int r = 0; r <= dataGridView.Rows.Count - 1; r++)
            {
                for (int c = 0; c <= dataGridView.Columns.Count - 1; c++)
                {
                    sLine = sLine + dataGridView.Rows[r].Cells[c].Value;
                    if (c != dataGridView.Columns.Count - 1)
                    {
                        sLine = sLine + "          ";
                    }
                }
                file.WriteLine(sLine);
                sLine = "";
            }
            file.Close();
        }
    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message, "Der er sket en fejl, data kunne ikke overføres til fil", MessageBoxButtons.OK, MessageBoxIcon.Error);
        file.Close();
    }
}
```

Resultatet er følgende filer:

Salgsstatistik - Notesblok

Filer Rediger Formater Vis Hjælp

SALGSSTATISTIK 2023

Fra dato: 01-02-2023 00:17:58

Til Dato: 19-05-2023 00:17:57

| Kundenummer | Kundenavn | Dato | Køb |
|-------------|------------------|---------------------|-------|
| 12 | Emma Roberts | 10-02-2023 00:00:00 | 4000 |
| 17 | Hanne Kjær | 13-02-2023 00:00:00 | 5400 |
| 41 | Julie Kaspersen | 20-02-2023 00:00:00 | 9450 |
| 35 | Mathilde Johnson | 25-02-2023 00:00:00 | 2500 |
| 23 | Daniel Dahl | 01-03-2023 00:00:00 | 5000 |
| 22 | Thomas Nielsen | 02-03-2023 00:00:00 | 6050 |
| 16 | Peter Jensen | 15-03-2023 00:00:00 | 8500 |
| 47 | Johs Møller | 18-03-2023 00:00:00 | 5020 |
| 69 | Henrik Skov | 24-03-2023 00:00:00 | 7540 |
| 18 | Camilla Ottesen | 10-04-2023 00:00:00 | 3990 |
| 21 | Maria Petersen | 25-04-2023 00:00:00 | 6370 |
| I alt kr. | | | 63820 |

Denne fil er med valgte datoer. Den anden fil der dannes, ligger under bilag nr. 5.

Kunde Application

Emeli

Spørgsmål til din kommende brille!

1. Hvad er maksimal pris på stel?
☐ 1500 kr ☒ 3000 kr ☐ 5000 kr ☐ 8000+ kr

2. Hvilken farve skal dit brillestel
☐ Sort ☐ Gennemsigtig ☒ Rød ☐ Brun

3. Hvilken facon skal stellet have?
☐ Rund ☐ Firkantet

4. Hvordan har du det med mønstret stel?
☐ Mønster ☐ Intet Mønster

5. Skal stellet have næsepuder?
☐ Med Næsepuder ☐ Uden Næsepuder

6. Skal stellet være tykt eller tyndt?
☐ Tykt ☐ Tyndt

| ProductID | Name | FrameColour | FrameFacon | Thickness | NosePads | Pattern | ProductPrice |
|-----------|---------|-------------|------------|-----------|----------------|---------|--------------|
| 1 | Rainbow | Rød | Rund | Tykt | Uden Næsepuder | Mønster | 995,00 |
| 6 | DKStel | Rød | Firkantet | Tykt | Uden Næsepuder | Mønster | 895,00 |
| 9 | EmeliK | Rød | Rund | Tykt | Uden Næsepuder | Mønster | 1499,95 |

SynsPunkt forespurte et system, hvor kunder kan selvbetjene og dermed spare tid med medarbejderne. Det har vi lavet i ovenstående Winform. Til venstre ses en gruppeboks med 6 vejledende spørgsmål. Her er det meningen at hvert spørgsmål først bliver tilgængeligt når de har svaret på spørgsmålet før. Ellers virker programmeringskoden ikke. Da den starter som en liste med alle brillestel. Denne liste bliver derefter opdateret ved brug af LINQ ved hver spørgsmål. Og så bliver en opdateret liste returneret og gemt i en Property i servicelaget. Til start i form koden, oprettes en variabel og dernæst bruges Switch statements ved hver OK knap `button_click`. Dermed kan den teste, hvilken Radiobutton der er 'checked'. Hver case opdaterer variabelens indhold og sender dette videre til servicelaget. Dette sammenlignes med oplysningerne gemt i databasen.

Read metoden der læser den fulde liste ud, ser sådan ud (bemærk sql-stringen, SELECT, er rykket for at gøre det lettere læseligt på et skærmbillede):

Den returnerede liste kaldes til servicelaget i metoden 'ReadAllFrames'. Nedenfor den metode ses metoderne der er tilhørende hvert spørgsmål, der er medtaget de to første, da de fint beskriver, hvordan alle metoderne gemmer

```
public List<Models.Frame> ReadAllFrames()
{
    List<Models.Frame> allFrames = new List<Models.Frame>();

    string query = $"SELECT ProductID, " +
        $"      ProductName, " +
        $"      FrameColour, " +
        $"      FrameFacon, " +
        $"      FrameThickness, " +
        $"      Nosepads, " +
        $"      FramePattern, " +
        $"      ProductPrice FROM Frame";

    SqlCommand command = new SqlCommand(query, connection);
    connection.Open();
    SqlDataReader reader = command.ExecuteReader();

    while (reader.Read())
    {
        Models.Frame product = new Models.Frame(
            (int)reader["ProductID"],
            reader["ProductName"].ToString(),
            reader["FrameColour"].ToString(),
            reader["FrameFacon"].ToString(),
            reader["FrameThickness"].ToString(),
            reader["NosePads"].ToString(),
            reader["FramePattern"].ToString(),
            (decimal)reader["ProductPrice"]);

        allFrames.Add(product);
    }
    reader.Close();
    connection.Close();

    return allFrames;
}
```

lister i properties og den property bliver så kaldt i næste spørgsmåls metode. Resten ligger i bilag. Øverst ses alle properties med lister.

```
public List<Frame> AllFrames { get; set; }  
2 references  
public List<Frame> FramesWithinPrice { get; set; }  
2 references  
public List<Frame> FramesWithColour { get; set; }  
2 references  
public List<Frame> FramesFacon { get; set; }  
2 references  
public List<Frame> FramesPattern { get; set; }  
2 references  
public List<Frame> FramesNosePad { get; set; }  
1 reference  
public List<Frame> FramesThickness { get; set; }  
5 references  
public List<Frame> ReadAllFrames()  
{  
    CustomerAppSQL customerAppSQL = new CustomerAppSQL();  
    List<Frame> allFrames = customerAppSQL.ReadAllFrames();  
  
    AllFrames = allFrames;  
    return allFrames;  
}  
  
1 reference  
public List<Frame> GetFramesPrice(int priceMaxSelected)  
{  
    List<Frame> framesWithinprice = AllFrames.Where(x => x.ProductPrice <= priceMaxSelected).ToList();  
    FramesWithinPrice = framesWithinprice;  
    return framesWithinprice;  
}  
  
1 reference  
public List<Frame> GetFramesColour(string colour)  
{  
    List<Frame> framesSpecificColour = FramesWithinPrice.Where(x => x.FrameColour == colour).ToList();  
    FramesWithColour = framesSpecificColour;  
    return framesSpecificColour;  
}
```

Som det sidste og første er form koden, denne modtager alle indtastninger og input fra kunden:

```
private void HomePageForm_Load(object sender, EventArgs e)
{
    this.frameTableAdapter.Fill(this.saanneha_dk_db_databaseDataSet.Frame);
}

1 reference
private void NextBt_1_Click(object sender, EventArgs e)
{
    // FrameServices frameServices = new FrameServices();
    List<Frame> allframes = frameServices.ReadAllFrames();
    int maxPrice;

    switch (true)
    {
        case true when RB_MaxPrice1500.Checked:
            maxPrice = 1500;
            break;
        case true when RB_MaxPrice3000.Checked:
            maxPrice = 3000;
            break;
        case true when RB_MaxPrice5000.Checked:
            maxPrice = 5000;
            break;
        case true when RB_MaxPrice8000.Checked:
            maxPrice = 8000;
            break;
        default:
            maxPrice = 20000;
            break;
    }

    Dgv_Frames.DataSource = frameServices.GetFramesPrice(maxPrice);
}
```

Konklusion

Emeli og Sanne

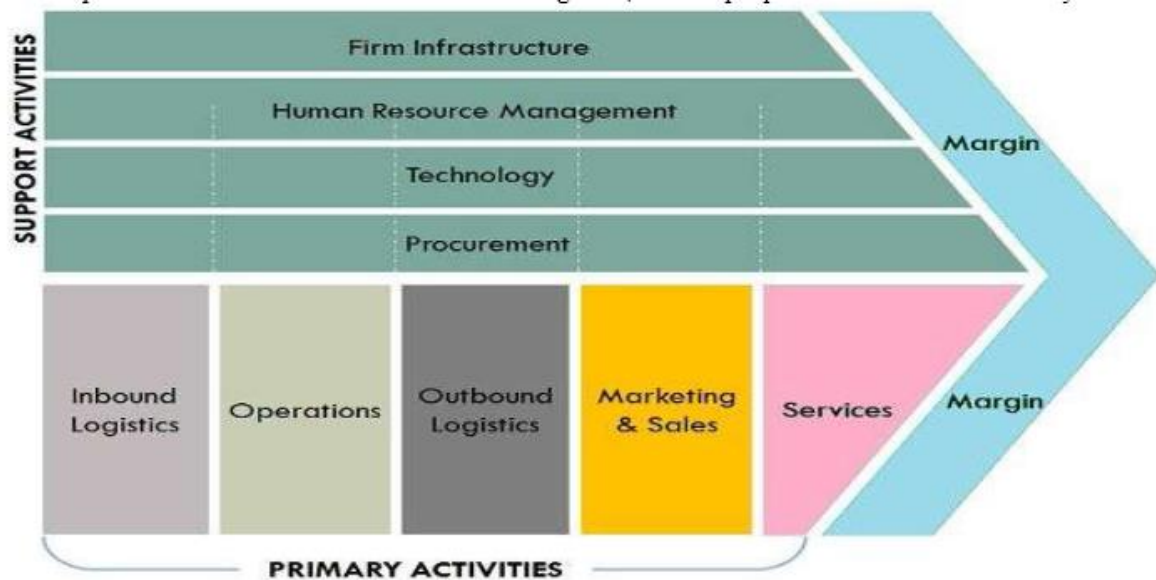
Ud fra de opstillede kravspecifikationer, har vi udarbejdet et program, der løser SynsPunkts problemstillinger. Ejer har mulighed for at strømline informationer til butikker, samtidig er det muligt at tilgå alle kunder og varer ude i de enkelte butikker, da alle disse informationer er samlet i en database. Denne database kan opdateres af både medarbejdere og eje. Intentionen med programmet er at opfylde ejers krav om at det kun er ham der kan tilføje og rette i varesortimentet. Dette er afbilledet med medarbejderID, som skal repræsentere de forskellige grader af rettigheder til programmet. Denne del er dog ikke funktionel endnu.

Programmet er gjort så enkelt og overskueligt som muligt, for at imødekomme virksomhedens mål for at kunne oplære nye medarbejdere hurtigt i det nye system.

Vi har udarbejdet et ekstra program, da ejer ønsker et intelligent rådgivningsprogram, som kunder kan benytte, og derved kan medarbejderen optimere deres ekspeditionstid. Programmet er kun en prototype, og mangler flere komponenter, for at kunne fungere optimalt ude i butikkerne. Dette ville kunne videreudvikles efter at salgssystemet er taget i brug.

Bilag

Bilag 1 Porters Værdikæde



Bilag 2 Booking

Linda

Opret/Slet Booking

Indtast Kunde Nr. / KundeNavn :

| | Kunde Nr. | Kunde Navn |
|---|-----------|------------|
| * | | |

09:50:00

juni 2023

| ma | ti | on | to | fr | lø | sø |
|----|----|----|----|----|----|----|
| 29 | 30 | 31 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |

I dag: 01-06-2023

Vælg Kunde

Opret Ny Kunde

Opret Booking

Slet Booking

Se alle Bookinger

På dette billede ses den side, som medarbejderen får adgang til ved oprettelse af en booking eller aflysning af en booking. Her kan medarbejderen assistere kunden med at reservere en tid enten til personligt fremmøde i butikken eller til telefonisk bestilling. Først skal medarbejderen indtaste kundens navn eller kundenummer, hvorefter en liste vil blive genereret med forhåbentlig kun én eller to kunder, hvis søgningen er præcis nok. Medarbejderen skal herefter markere kunden og vælge "Vælg kunde", medmindre kunden endnu ikke er oprettet. Hvis dette er tilfældet, kan medarbejderen trykke på "Opret ny kunde", hvorefter han bliver omdirigeret til oprettelsesprocessen for en ny kunde. Når kunden er oprettet, kan medarbejderen finde kunden og vælge "Vælg kunde". Derefter har medarbejderen tre valgmuligheder: "Opret booking", "Slet booking" eller "Se alle bookinger". Slet-knappen vil kun være aktiv, hvis kunden allerede har reserveret en tid. "Se alle bookinger" er primært til medarbejderne, så de kan holde styr på deres tidsplan. I de fleste tilfælde vil medarbejderen vælge en dato og et tidspunkt og trykke på "Opret booking", hvorefter den valgte tid vil blive indsat i kalenderen.

| <u>Medarbejder:</u> | <u>KundeID:</u> |
|---------------------|---|
| Per: | |
| 10.15 - 10.30 | <input type="text" value="0000000000"/> |
| 10.30 - 10.45 | <input type="text" value="0000000000"/> |
| 11.00 - 11.15 | <input type="text" value="0000000000"/> |
| Lasse: | |
| 12.15 - 12.30 | <input type="text" value="0000000000"/> |
| 13.00 - 13.15 | <input type="text" value="0000000000"/> |
| Henrik: | |
| 15.15 - 15.30 | <input type="text" value="0000000000"/> |
| 15.30 - 15.45 | <input type="text" value="0000000000"/> |

Luk

Hvis medarbejderen trykker på knappen "Se alle bookinger" på det ovenstående billede, åbner systemet kalenderen for den valgte dato og medarbejderen bliver omdirigeret til en side, hvor de forskellige medarbejders bookinger for den pågældende dag vises sammen med de tilhørende kunde-ID'er. Medarbejderen kan slå disse kunde-ID'er op i kundedatabasen, hvis de f.eks. har behov for at aflyse en booking. Hvis medarbejderen derimod trykker på "Booking" og også vælger "Se bookinger" i menuen på forsiden, bliver medarbejderen omdirigeret til en lignende side, men denne vil altid vise bookingerne for dagens dato.

Bilag 3 User stories

Sanne, Emeli, Linda

| Epic User Story | Normal User Story |
|--|--|
| Som medarbejder vil jeg have et bookingsystem Så jeg kan holde styr på bookinger | Som medarbejder vil jeg kunne oprette en bestilling i systemet Så jeg kan gøre det hurtigere |
| | Som medarbejder vil jeg kunne booke en synstest til mine kunder Så de kan få foretaget en synstest |
| | Som medarbejder vil jeg have et bookingsystem Så jeg kan overskue dagens aftaler for butikken |
| | Som medarbejder vil jeg have et bookingsystem Så jeg kan booke nye aftaler |

| Epic User Story | Normal User Story |
|--|---|
| Som ejer vil jeg have en kunde database Så alle medarbejdere kan tilgå al kunde data | Som medarbejder vil jeg kunne redigere kundeoplysninger Så kundens info altid er korrekt |
| | Som medarbejder vil jeg vil jeg kunne oprette kunde i system Så kunden kan få hjælp i alle butikker fremover |
| | Som medarbejder vil jeg kunne tilgå ordrehistorikken på en kunde Så jeg hurtigt og nemt kan finde det info jeg skal bruge for at hjælpe |

| Epic User Story | Normal User Story |
|--|---|
| <p>Som ejer</p> <p>Vil jeg have en kundeapplikation til butikkerne så salg kan optimeres</p> | <p>Som kunde</p> <p>vil jeg have et system/app hvor jeg kan filtrere min søgning af briller</p> <p>Så jeg kan få hjælp til spore mig ind på hvilken type jeg vil have</p> |
| | <p>Som kunde</p> <p>vil jeg have et system/app hvor jeg kan filtrere min søgning af kontaktlinser</p> <p>Så jeg kan få hjælp til spore mig ind på hvilken type jeg vil have</p> |
| | <p>Som kunde</p> <p>vil jeg kunne tilgå mig en knap</p> <p>Så der helt automatisk kommer en medarbejder hen til mig</p> |
| | <p>Som medarbejder</p> <p>vil jeg have et system/app kunden kan benytte</p> <p>Så min tid kan optimeres</p> |
| | <p>Som medarbejder</p> <p>vil jeg have et system/app kunden kan benytte så jeg bedre kan hjælpe kunden helt i mål.</p> |
| | <p>Som ejer</p> <p>Vil jeg have en kundeapplikation til butikkerne så jeg kan tiltrække en bredere kundegruppe</p> |

| Epic User Story | Normal User Story |
|--|--|
| <p>Som ejer</p> <p>vil jeg have en informations platform</p> <p>Så alle butikker har samme informationer tilgængelig</p> | <p>Som medarbejder</p> <p>vil jeg kunne tilgå et centralt sted at hente mine informationer</p> <p>Så jeg altid kan leve op til det der forventes</p> |
| | <p>Som ejer</p> <p>vil jeg kunne opdatere informationsplatformen</p> <p>Så butikkerne kan strømlines</p> |
| | <p>Som ejer</p> <p>vil jeg kunne redigere i informationsplatformen</p> <p>Så jeg kan sikre det kun er opdateret info der er tilgængelig</p> |

Bilag 4 Use case beskrivelser

Sanne

| | |
|-----------|---|
| | Opret bestilling til kunde. |
| Aktør(er) | Medarbejder |
| Handling | <ul style="list-style-type: none"> - Tryk på "bestil varer" i systemet - Ny skærm åbner op - Indtast/søg efter den/de valgte vare(r) - Indtast/ søg efter kundeoplysninger - Vælg levering/afhentning i butik (ved briller kun afhentning) - Oplys kunden om leveringstid (skal dukke op automatisk i systemet) - Vælg OK – pop-up boks der spørger om alt indtastet er korrekt. - Tag imod betaling - Afslut bestilling |

| | |
|-----------|--|
| | Opret en kunde. |
| Aktør(er) | Medarbejder |
| Handling | <ul style="list-style-type: none"> - Tryk på "opret kunde" i systemet - Indtast alle oplysningerne om kunden - Pop-up boks der spørger om alle oplysninger er korrekte - Gem og OK |

| | |
|-----------|--|
| | Rediger kundeoplysninger |
| Aktør(er) | Medarbejder |
| Handling | <ul style="list-style-type: none">- Tryk på "redigér kunde" i systemet- Tryk på det valgte felt som skal redigeres- Gentag ved alle felter der skal redigeres- Vælg OK- Pop-up boks der spørger om alle oplysninger er korrekte- Gem og luk |

| | |
|-----------|--|
| | Bestil nye varer hjem. |
| Aktør(er) | LP + Medarbejder (Butiksbestyreren) |
| Handling | <ul style="list-style-type: none">- Tryk på "Bestilling af varer" i systemet- Ny skærm åbner op- Tryk på find vare – funktion hvor man åbner en ny skærm op med søgefunktion efter varenr. og varenavn- Resultaterne af søgningen vises i en tabel i et datagridview- Markér den rigtige vare og vælg OK- Vend tilbage til tidligere skærm, hvor varen bliver tilføjet i ny tabel i et datagridview- Vælg antal- Gentag processen indtil alle varer der ønskes bestilt er valgt- Tryk OK- Dialogboks åbner op og spørger om alt er korrekt- Vælg OK- Gem og Luk |

| | |
|-----------|---|
| | Tilgå infotavle. |
| Aktør(er) | Medarbejdere + LP |
| Handling | <ul style="list-style-type: none">- Info tavle skal være på forsiden af medarbejdersystemet- Selve infotavlen fylder halvdelen af forsiden- Denne indeholder de enkelt beskeder ejer har sendt ud- Den enkelte besked vises med overskriften og de første 2 linjer af teksten- Selve beskeden kan trykkes på og der åbnes et pop-up vindue med beskeden i som kan lukkes efter læsning (kryds i hjørnet som alm browser vindue) |

| | |
|-----------|--|
| | Send info til infotavle |
| Aktør(er) | LP |
| Handling | <ul style="list-style-type: none"> - Simple beskeder skal kunne indsættes i systemet - Når ejer er logget ind i systemet er der en knap der hedder "indsæt besked" - Når denne besked trykkes på åbnes nyt vindue med tekstbokse i. - Første tekstboks er en overskrift - Anden tekstboks er selve beskeden - Disse skal være "enabled" så ejer kan skrive i dem. - Når disse er udfyldt trykkes på OK knap - Dialogboks åbner op og spørger om alt er korrekt - Vælg OK - Gem og Luk - Beskeden vises med det samme i infotavlen |

| | |
|-----------|--|
| | Book synstest |
| Aktør(er) | Medarbejdere |
| Handling | <ul style="list-style-type: none"> - Åben "Booking" i systemet - Vælg "Book Synstest" - Nyt vindue åbnes op - Der åbnes en kalender hvor der vælges dato og tidspunkt - Indtast/søg efter kundeoplysninger og tilføj disse - Indtast medarbejder (den medarbejder kunden skal bruge) - Vælg OK - Dialogboks åbner op og spørger om alt er korrekt - Vælg OK - Gem og Luk |

| | |
|-----------|--|
| | (Send besked til kunde.) |
| Aktør(er) | E-mail System |
| Handling | <ul style="list-style-type: none"> - Ved endt booking skal systemet selv sende en SMS/Mail til kunden. - Denne besked indeholder info om: - Hvilken booking - Hvornår bookingen finder sted - Info, i tilfælde af at kunden har brug for at aflyse/ændre aftalen. - Ny besked med en reminder om aftalen sendes ud 24 timer før aftalen finder sted. |

Bilag 5 Illustrationer af programmet

Opret Kunde

Kunde ID : Udfyldes Automatisk ?

Fomavn :

Efternavn :

Tlf.nr. :

Email :

Adresse :

Postnummer :

By :

Opret Kunde

Rediger Kunde

Kundeinformationer ?

Indtast KundeNr / KundeNavn : to

OK

| | CustomerID | FirstName | LastName |
|---|------------|-----------|------------|
| | 4 | Tom | Felton |
| ▶ | 10 | Tom | Hiddleston |
| | 13 | Tom | Hanks |
| | 14 | Tom | Holland |
| | 15 | Tom | Hardy |
| | 16 | Tom | Cruise |

Vis Kundeoplysninger

Kundeoplysninger

Kunde ID : 10

Fornavn : Tom

Efternavn : Hiddleston

Tlf.nr. : 45983217

Email : TheTricster@Asgard.net

Adresse : Lokesvej 58

Postnummer : 6623

By : Vorbasse

Slet Kunde Gem

Opret Salg

| <u>VareNr. fra strekcode:</u> | <u>Varenavn:</u> | <u>Antal:</u> | <u>Pris pr stk:</u> | <u>Totalpris:</u> |
|-------------------------------|---|----------------------|---|---|
| <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text" value="0000000000"/> |
| <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text" value="0000000000"/> |
| <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text" value="0000000000"/> |

Pris i alt

Forsæt til betaling

Returnér Vare

| <u>VareNr. fra strekcode:</u> | <u>Varenavn:</u> | <u>Antal:</u> | <u>Pris pr stk:</u> | <u>Totalpris:</u> |
|---|---|----------------------|---|---|
| <input type="text" value="00000000000000000000"/> | <input type="text" value="0000000000"/> | <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text" value="0000000000"/> |
| <input type="text" value="00000000000000000000"/> | <input type="text" value="0000000000"/> | <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text" value="0000000000"/> |
| <input type="text" value="00000000000000000000"/> | <input type="text" value="0000000000"/> | <input type="text"/> | <input type="text" value="0000000000"/> | <input type="text" value="0000000000"/> |

Pris i alt

Godkend Returning

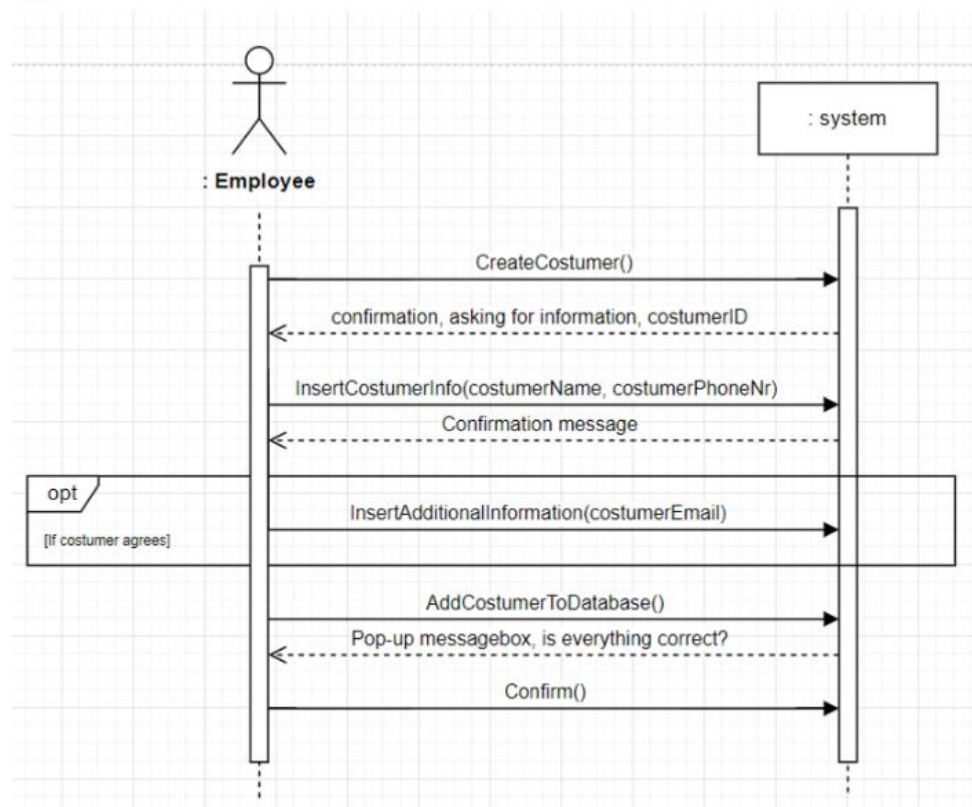
| AlleSalg - Notesblok | | | | |
|----------------------------------|----------|-----------|---------------------|------|
| Filer Rediger Formater Vis Hjælp | | | | |
| SALGSSTATISTIK 2023 | | | | |
| Kundenummer | | Kundenavn | Dato | Køb |
| 10 | Eigil | Olsen | 02-06-2023 00:00:00 | 4534 |
| 21 | Maria | Petersen | 25-04-2023 00:00:00 | 6370 |
| 18 | Camilla | Ottesen | 10-04-2023 00:00:00 | 3990 |
| 69 | Henrik | Skov | 24-03-2023 00:00:00 | 7540 |
| 47 | Johs | Møller | 18-03-2023 00:00:00 | 5020 |
| 16 | Peter | Jensen | 15-03-2023 00:00:00 | 8500 |
| 22 | Thomas | Nielsen | 02-03-2023 00:00:00 | 6050 |
| 23 | Daniel | Dahl | 01-03-2023 00:00:00 | 5000 |
| 35 | Mathilde | Johnson | 25-02-2023 00:00:00 | 2500 |
| 41 | Julie | Kaspersen | 20-02-2023 00:00:00 | 9450 |
| 17 | Hanne | Kjær | 13-02-2023 00:00:00 | 5400 |
| 12 | Emma | Roberts | 10-02-2023 00:00:00 | 4000 |
| 24 | Mikkel | Laursen | 31-01-2023 00:00:00 | 8500 |
| 11 | Albert | Hansen | 25-01-2023 00:00:00 | 4534 |
| 15 | René | Jørgensen | 25-01-2023 00:00:00 | 4500 |
| 30 | Leila | Dyrmose | 16-01-2023 00:00:00 | 6400 |
| 13 | Maja | Sørensen | 10-01-2023 00:00:00 | 8000 |
| 14 | Jacob | Lassen | 03-01-2023 00:00:00 | 2500 |

Her er scenariet hvor user trykker direkte på print knappen.

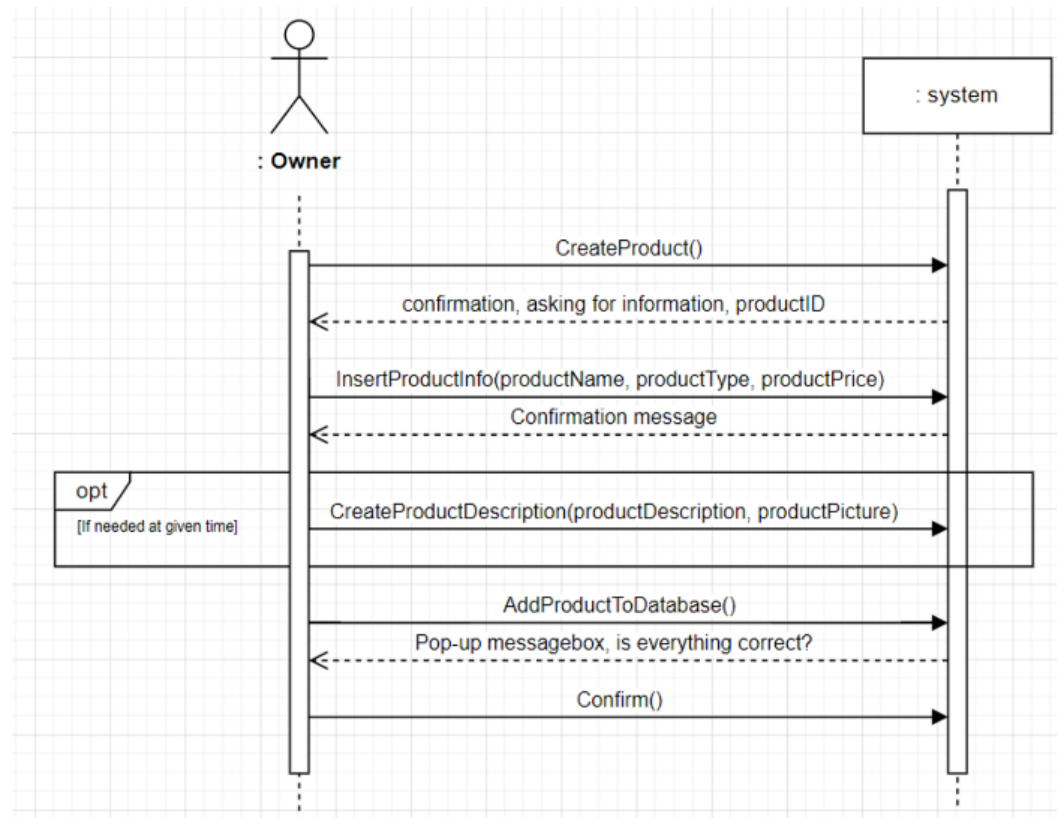
Bilag 6 SSD

Emeli

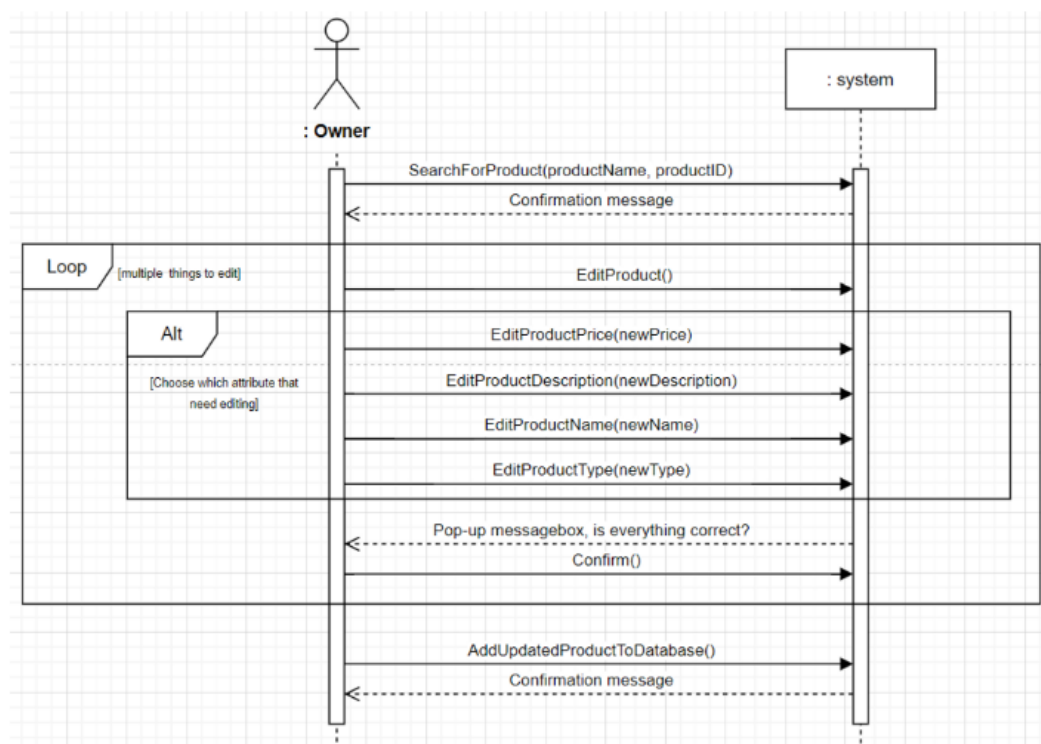
Opret en kunde.



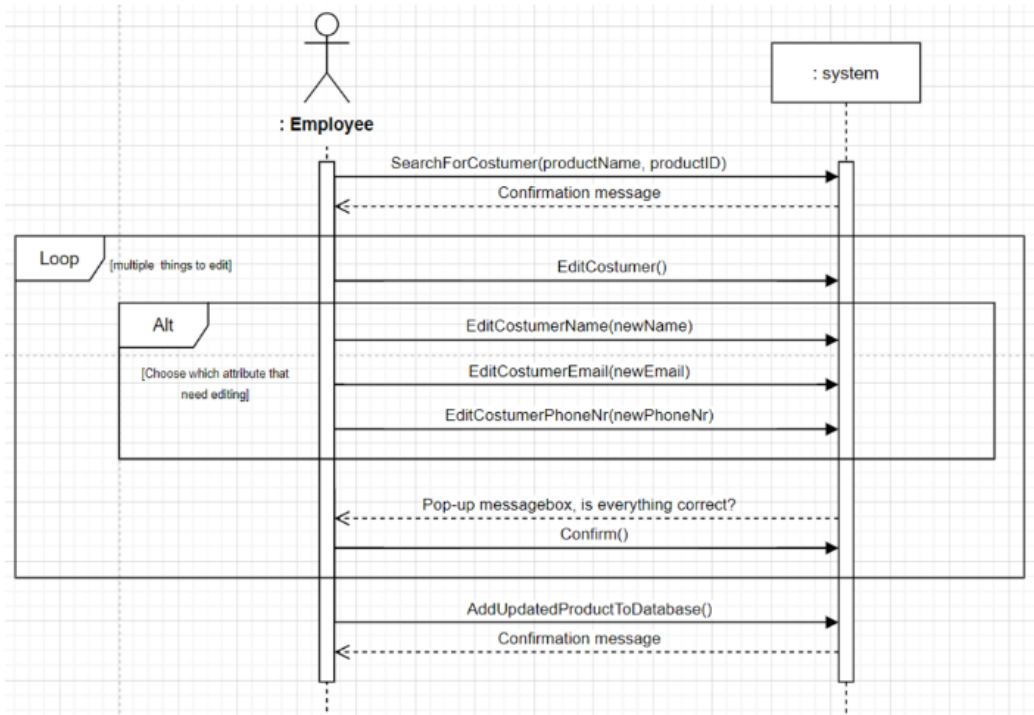
Opret en vare.



Rediger vare



Rediger kundeoplysninger



Opret salg til gadekunder

