

Dokumentationsrapport
4. Semester - Gruppe 1



Rieder, Kasper Jensen, Daniel V. Clausen, Ole Konstmann, Mia
201310514 201500152 20115758 201500157

Kloock, Michael Møller Høj, Christoffer Søby Vejleder:
201370537 201407641 Poul Ejnar Rovsing

9. december 2016

Indhold

1 Indledning	6
1.1 Koncept	6
2 Kravspecifikation	7
2.1 Aktørbeskrivelser	7
2.2 Epics	7
2.3 User Stories	8
2.3.1 Kunde	8
2.3.2 Butiksadministrator	9
2.3.3 Systemadministrator	9
2.4 Systembeskrivelse	10
2.5 Ikke funktionelle krav	11
2.6 Projektgrænsning	12
3 Systemarkitektur	14
3.1 Domænemodel	14
3.2 Arkitektur valg	16
3.2.1 Desktop applikation	16
3.2.2 Central server	16
3.2.3 Database	17
3.2.4 Web applikation	17
3.2.5 Programmeringssprog	17
3.2.6 HTTP	18
3.2.7 Bootstrap	18
3.2.8 jQuery	18
3.3 Brugergrænseflader	19
3.3.1 Desktop applikation	19
3.3.2 Web applikation	20
3.4 Database	20
3.4.1 Brugerdatabase	21
3.4.2 Butiksdatabase	22
3.5 Database API	23
4 Design & Implementering	25
4.1 Database	25
4.1.1 Brugerdatabase	25
4.1.2 Butiksdatabase	25
4.2 Database API	28
4.2.1 Store database API	29
4.3 Desktop Applikation	37
4.3.1 Model laget	37
4.3.2 ViewModel laget	39
4.3.3 View laget	40
4.4 Web applikation	51
4.4.1 Forretningslogik laget	52
4.4.2 Præsentationslaget	53
5 Test	56

5.1	Anvendte test frameworks	56
5.1.1	NUnit	56
5.1.2	NSubstitute	56
5.2	Visuel test	56
5.3	Unit test	56
5.3.1	Desktop applikation	57
5.3.2	Webapplikation	57
5.4	Integrationstest	58
6	Accepttest	61
6.1	Desktop applikation	61
6.1.1	Opret vare	61
6.1.2	Opret vare uden valgt varekategori	61
6.1.3	Slet vare	62
6.1.4	Indsæt plantegning til butikken	62
6.1.5	Tilføj sektion til butikkens plantegning	63
6.1.6	Rediger sektion	63
6.1.7	Slet sektion	64
6.1.8	Tilføj vare til sektion	64
6.1.9	Slet vare fra sektion	65
6.2	Web applikation	65
6.2.1	Vis vare på butikkens plantegning	65
6.3	Ikke funktionelle krav	66

Læsevejledning

Denne dokumentationsrapport beskriver udviklingsforløbet for produktet Locatiles, som anvendes til søgning af vareplaceringer i butikker. I denne rapport bliver visionen for produktet præsenteret og efterfølgende blive afgrænset til at passe tiden afsat til et 4. semesterprojekt.

I denne rapport er brødtekst skrevet på dansk, dog er diagramteksterne skrevet på engelsk for at stemme overens med koden.

I løbet af rapporten vil der jævnligt refereres til database API eller DbAPI. Med dette menes specifikt "Store Database API" delen af database API.

Termliste

Main Window	Hovedmenuen i desktop applikationen vil i flere diagrammer blive refereret til som “Main Window”.
Manage Items	Vinduet hvor varer administreres bliver igennem rapporten refereret til som “Manage Items” eller “Administrer varer”
Create Item	Vinduet i desktop applikationen, hvor varer oprettes bliver i rapporten refereret til enten som “Create item” eller “Opret varer”.
Manage Storesections	Vinduet hvor sektioner administreres, bliver igennem rapporten refereret til som “Manage Storesections” eller “Administrer sektioner”.
Edit Storesection	Vinduet hvor sektioner redigeres, bliver igennem rapporten refereret til som “Edit storesection” eller “Rediger sektion”.
Create storesection	Vinduet hvor sektioner oprettes, bliver igennem rapporten refereret til som “Create Storesection” eller “Opret sektioner”.
Manage Itemgroups	Vinduet hvor varegrupper administreres, bliver igennem rapporten refereret til som “Manage Itemgroups” eller “Administrer varegrupper”.
Create Itemgroup	Vinduet hvor varegrupper oprettes, bliver igennem rapporten refereret til som “Create itemgroup” eller “Opret varegrupper”.
Edit Itemgroup	Vinduet hvor varegrupper redigeres, bliver igennem rapporten refereret til som “Edit Itemgroup” eller “Rediger varegruppe”
Manage floorplan	Vinduet hvor plantegninger administreres, bliver igennem rapporten refereret til som “Manage floorplan” eller “Administrer plantegninger”
Plantegning	En plantegning er et billede af butikkens indretning, hvor reoler og gangarealer ses oppefra. Oversættelse: Floorplan
Sektion	En sektion er et punkt på plantegningen hvorpå der kan placeres et vilkårligt antal varer. Oversættelse: Store Section
Vareplacering	Lokationen af den sektion, som varer er tildelt
Reol	En reol er en fysisk reol som varer står på i en butik
Varegruppe	En varegruppe er en samling af varer der hører under samme kategori, f.eks. mejeri. Oversættelse Item Group
Over varegruppe	En varegruppe kan tilhøre en over varegruppe, f.eks. kan varegruppen “ost” tilhøre over varegruppen “mejeri”. Oversættelse: Parent Item Group
Admin	Admin er en forkortelse for administrator. Igennem rapporten bruges begge begreber omskifteligt
Epics	Ordet dækker over en samling af relaterede userstories for systemets funktionalitet
CRUD	CRUD er et udtryk for operationerne “Create”, “Read”, “Update” og “Delete”

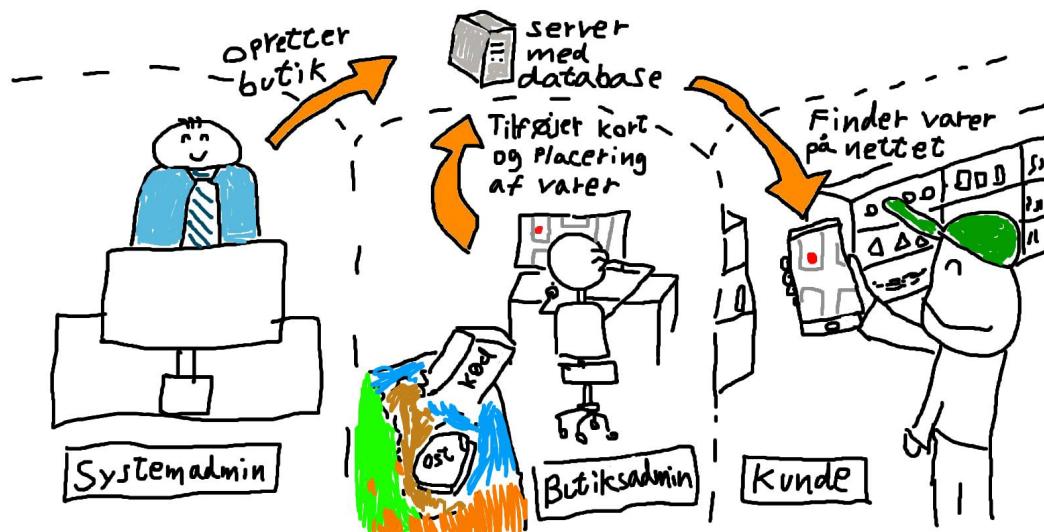
API	API er et udtryk for “Application Programming Interface”, hvilket er et software moduls grænseflade som en klient kan gøre brug af for at tilgå modulets funktionaliteter
Bruger Database	Bruger databasen indeholder bruger oplysninger som tilknytter brugere af systemet til specifikke butiksdatabase

1 Indledning

Som kunde kan det være svært at finde en ønsket vare i en forretning. Dette kan skyldes at man er i en fremmed butik, eller man er på udkig efter en eksotisk vare. Oftest vil kunden herefter lede efter en medarbejder for at få hjælp til at finde varen, hvilket vil tage tid, samt forstyrre medarbejderen i sit arbejde.

For at afhjælpe dette problem udvikles Locatiles, et produkt hvor kunder via en grafisk brugergrænseflade får mulighed for at søge på en vare ved at indtaste varens navn i et søgerfelt. Når en vare er blevet fundet i databasen, vises dennes placering på plantegningen i web applikationen. Denne søgning skal kunne foretages via en web applikation, som kan tilgås fra kundens pc i hjemmet, kundens smartphone eller en stander i butikken. Produktet yder en Quality of Life service, som butikkerne kan bruge til at give kunderne et ekstra incitament til handle hos dem.

1.1 Koncept



Figur 1: Koncepttegning af Locatiles

På figur 1 ses en oversigt af systemets brugere og hvordan de anvender systemet.

En systemadministrator opretter en butiksdatabase til butikken samt login for en butiksadministrator på en server. Butiksadministratoren indsætter en plantegning af butikken og sektioner på plantegningen. Disse sektioner kan for eksempel repræsentere en reol eller en køler. Sektioner kan tildeles varer som derved får en lokation i butikken. Butiksadministratoren har også mulighed for at tildele varegrupper til varer. Derefter har kunden mulighed for at søge efter en bestemt vare, ved brug af en web applikation, som tilgås via internettet. Hvis varen eksisterer og er tildelt en sektion, får kunden varens placering vist som et punkt afmærket på butikkens plantegning.

2 Kravspecifikation

I de følgende afsnit defineres de funktionelle og ikke funktionelle krav til Locatiles. Dette sker i form af aktørbeskrivelser, epics og en system beskrivelse. Epics bliver udspecifieret til user stories, og user stories bliver udspecifieret til brugerscenarier. Disse brugerscenarier lægger til grund for de accepttests, afsnit 6, s. 61, som udføres på systemet senere i udviklingsforløbet.

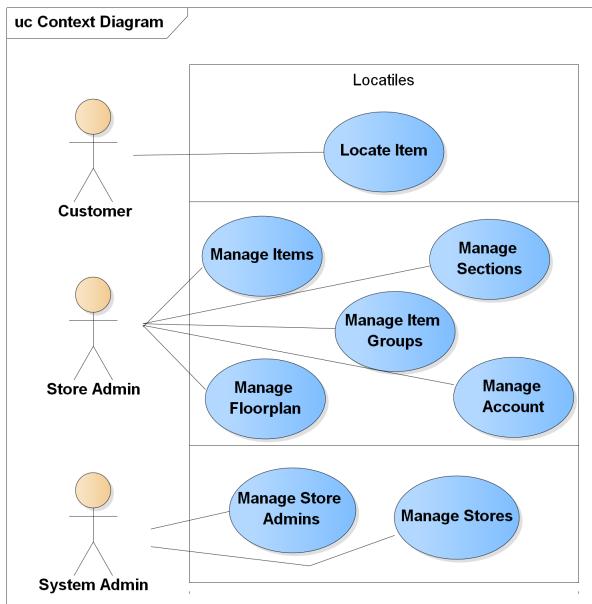
2.1 Aktørbeskrivelser

Følgende er en beskrivelse af de aktører der interagerer med systemet.

Kunde	Denne aktør vil interagere med systemet igennem en web applikation, for at finde en vare i butikken
Butiksadministrator	Denne aktør vil tilgå butikkens database, for at vedligeholde butikkens information, såsom vareliste, varegrupper, plantegninger og sektioner
Systemadministrator	Denne aktør vil oprette butikkernes databaser, samt vedligeholde butiksadministratorernes login informationer.

2.2 Epics

Til beskrivelse af systemets ønskede funktionaliteter er der blevet opstillet en række epics [1] som tager udgangspunkt i hvad de tre brugere ønsker af funktionalitet. I dette afsnit er disse epics opstillet i et kontekst diagram, figur 2, med en tilhørende beskrivelse. Hver epic repæsenterer en relateret samling af user stories som specificeres i afsnit 2.3, s. 8.



Figur 2: Kontekst diagram over Locatiles' epics

Følgende er beskrivelser for de epics, der optræder i kontekst diagrammet foroven. Hver epic beskriver et ønske som en aktør har til systemet, samt hvad de vil få ud af disse ønsker.

Locate Item	Som kunde vil jeg kunne søge efter en vares placering i butikken.
Manage Items	Som butiksadministrator vil jeg kunne tilføje, fjerne og redigere varer i butikkens databasen.
Manage Item Groups	Som butiksadministrator vil jeg kunne tildele varegrupper til varer for at kategorisere disse.
Manage Floorplan	Som butiksadministrator vil jeg kunne tilføje og redigere plantegninger for at give kunden et overblik over butikken.
Manage Sections	Som butiksadministrator vil jeg kunne tilføje og fjerne varer fra sektioner for at give en vare en lokation.
Manage Account	Som butiksadministrator vil jeg kunne ændre password for min konto, for at få et nyt.
Manage Store Admin	Som systemadministrator vil jeg kunne oprette butiksadministratorer i systemet og associere dem med butikker, så nye brugere kan logge ind.
Manage Stores	Som systemadministrator vil jeg kunne oprette butik databaser i systemet, så nye butiksadministratorer kan gøre brug af servicen til deres butik.

2.3 User Stories

Hver epic er blevet delt ud i user stories, som mere specifikt beskriver de funktionaliteter der ønskes af det endelige system. Disse user stories er kategoriseret efter brugerne og er yderligere blevet prioriteret efter MoSCoW princippet i projektafgrænsning, afsnit 2.6, s. 12.

2.3.1 Kunde

En kunde kan via forskellige typer af søgning bruge systemet til at finde en eller flere varers placeringer. En kunde har følgende ønsker til systemet:

- Som kunde vil jeg kunne søge efter en af butikkens varer, for at kunne se varens placering
- Som kunde vil jeg kunne søge efter en varegruppens placering, for at danne et overblik af disse varers placeringer
- Som kunde vil jeg kunne se min egen placering på butikkens plantegning, for at relatere det til produkters placering
- Som kunde vil jeg gerne have en rutevejledning til den søgte vare

2.3.2 Butiksadministrator

En butiksadministrator har ansvar for at administrere varer, varernes lokationer i butikken og plantegningen over butikken.

En butiksadministrator har følgende ønsker til systemet:

- Som butiksadministrator vil jeg kunne logge ind og ud af systemet, for at undgå uautoriseret adgang
- Som butiksadministrator vil jeg kunne opdatere plantegningen til butikken, for at placere sektioner på den
- Som butiksadministrator vil jeg kunne tilføje, redigere samt fjerne sektioner på butikkens plantegning, så jeg kan bestemme hvor varer kan placeres
- Som butiksadministrator vil jeg kunne oprette og slette varer i butikkens database, for at vedligeholde kundens søgedatabase
- Som butiksadministrator vil jeg kunne redigere allerede eksisterende varer, for at vedligeholde deres information
- Som butiksadministrator vil jeg kunne tilføje og fjerne varer fra en sektion, for at ændre lokationen på en vare i butikken
- Som butiksadministrator vil jeg vise relevante reklamer for mine kunder når de søger på en vare så kunden har mulighed for at se butikkens aktuelle tilbud
- Som butiksadministrator vil jeg kunne oprette, redigere og slette varegrupper for at kategorisere butikkens varer

2.3.3 Systemadministrator

En systemadministrator har ansvar for at oprette nye butikker i systemet og give en butiksadministrator adgang til systemet.

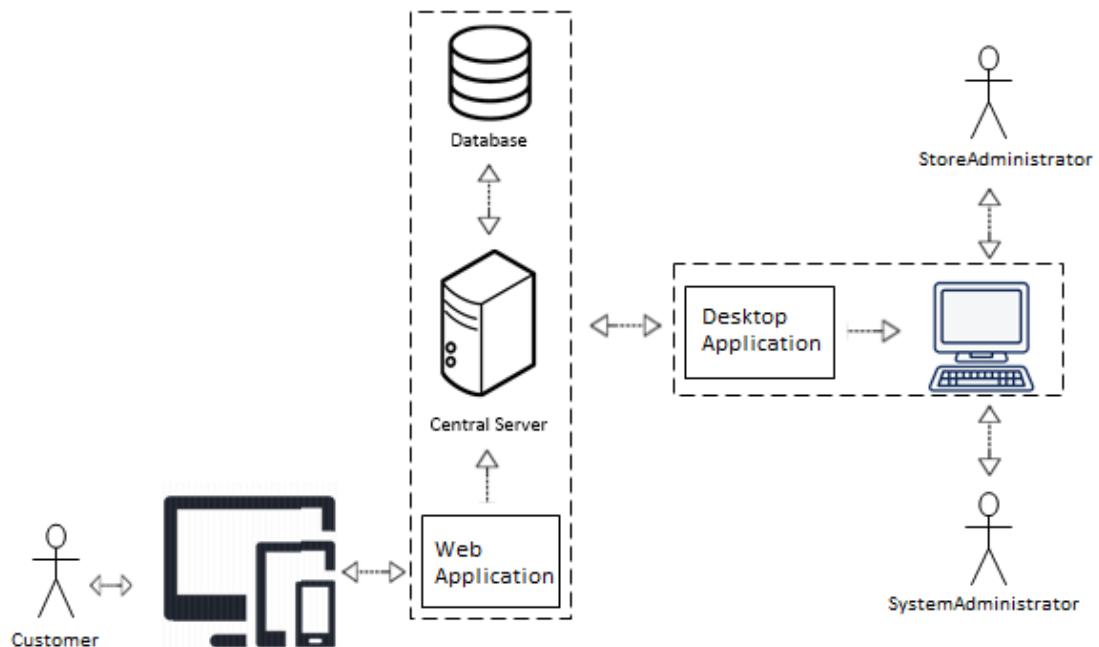
En systemadministrator har følgende ønsker til systemet:

- Som systemadministrator vil jeg kunne logge ind og ud af systemet, for at undgå uautoriseret adgang
- Som systemadministrator vil jeg kunne oprette og nedlægge butiksdatabase i systemet, for at vedligeholde aktive butikker
- Som systemadministrator vil jeg kunne oprette og nedlægge butiksadministratorer i systemet, for at vedligeholde aktive butiksadministratorer

2.4 Systembeskrivelse

I dette projekt udvikles systemet til vare administrering og lokalisering i dagligvare butikker. Herunder beskrives systemet samt typiske brugerscenarier hvori produktet indgår.

Figur 3 viser de centrale dele som indgår i systemet samt deres relationer. Den stipede afmærkning omkring den centrale server angiver at databasen og web applikationen bliver hosted på denne. På PC'en hostes desktop applikationen som også angives med en stiplet afmærkning. For at web applikationen og desktop applikationen kan tilgå databasens data udvikles der et Application Programming Interface (API), som beskrives nærmere i afsnit 4.2, s. 28.



Figur 3: Illustration af systemets centrale dele

Et typisk brugerscenarie for kunden, ville være at kunden står i en forretning og leder efter en vare. Kunden kan ikke finde varen og vælger derfor at gøre brug af Locatiles via egen telefon eller en af butikkens standere.

I web applikationen anvender kunden søgefeltet ved at indtaste varens navn. Herefter vises en liste af varer som indeholder søgeræksten, hvor kunden vælger den vare som søges efter. Herefter vises dennes placering på plantegningen. Kunden kan nu finde varen ud fra markeringen på plantegningen.

Et typisk brugerscenarie for butiksadministratoren er når der skal oprettes en ny plantegning for butikken efter en omrøkering af samtlige af butikkens varer.

Butiksadministratoren logger ind på systemet, vælger administrerer plantegning og indsætter en ny plantegning. Derefter placerer butiksadministratoren nye sektioner på plantegningen. Herefter kan varer tilføjes til de ny oprettede sektioner. Butiksadministratoren gemmer de nye ændringer så disse kan ses af kunden.

Et typisk brugerscenarie for systemadministratoren er når en ny klient køber produktet. Systemadministratoren opretter en ny butik i databasen samt en ny butiksadministrator. Systemadministratoren logger ind i systemet, vælger at oprette en butik i menuen og herefter at oprette en butiksadministrator. Systemadministratoren informerer klienten om de nye login detaljer.

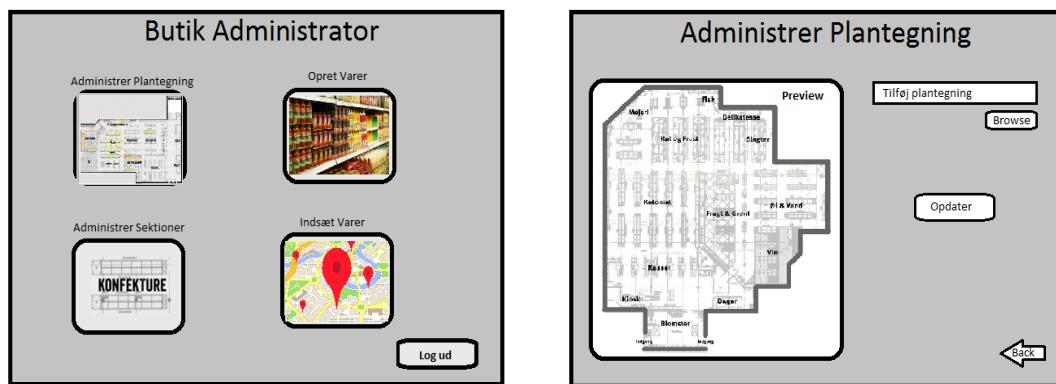
2.5 Ikke funktionelle krav

For at specificere produktet yderligere er der opsat en række ikke funktionelle krav, der stiller krav til kvaliteten af systemet.

1. Desktop applikationens størrelse må ikke overstige 200MB
2. Desktop applikationen skal kunne acceptere plantegninger af filtypen .jpg
3. Desktop applikationen skal kunne afvikles på Windows 10

Brugergrænseflade skitser

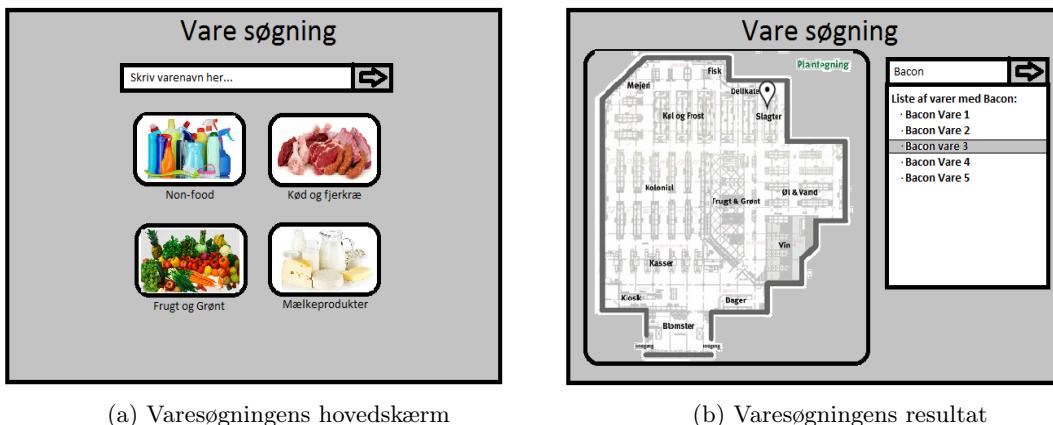
Følgende afsnit viser skitser for systemets brugergrænseflader.



(a) Desktop applikationens hovedskærm

(b) Desktop applikationens plantegning administrator

Figur 4: Skitse for desktop applikationen



Figur 5: Skitse for web applikation

2.6 Projektafgrænsning

Følgende afsnit indeholder en MoSCoW analyse af de definerede user stories. Disse er prioriteret i kategorierne 'must have', 'should have', 'could have' og 'won't have' ifølge MoSCoW principippet [2]. Denne prioritering tydeliggør hvilke funktionaliteter der er vigtigst for dette projektforløb.

Must have:

- Som butiksadministrator vil jeg kunne opdatere plantegningen til butikken, for at placere sektioner på den
- Som butiksadministrator vil jeg kunne tilføje, redigere samt fjerne sektioner på butikkens plantegning, så jeg kan bestemme hvor varer kan placeres
- Som butiksadministrator vil jeg kunne oprette og slette varer i butikkens database, for at vedligeholde kundens søgedatabase
- Som butiksadministrator vil jeg kunne tilføje og fjerne varer fra en sektion, for at ændre lokationen på en vare i butikken
- Som kunde vil jeg kunne søge efter en af butikkens varer, for at kunne se varens placering

Should have:

- Som butiksadministrator vil jeg kunne oprette, redigere og slette varegrupper for at katagorisere butikkens varer
- Som butiksadministrator vil jeg kunne logge ind og ud af systemet, for at undgå uautoriseret adgang
- Som butiksadministrator vil jeg kunne redigere allerede eksisterende varer, for at vedligeholde deres information

Could have:

- Som systemadministrator vil jeg kunne logge ind og ud af systemet, for at få adgang til systemet

- Som systemadministrator vil jeg kunne oprette og nedlægge butikker i systemet, for at vedligeholde aktive butikker
- Som systemadministrator vil jeg kunne oprette og nedlægge butiksadministratorer i systemet, for at vedligeholde aktive butiksadministratorer
- Som kunde vil jeg kunne søge efter en varegruppens placering, for at danne et overblik af disse varers placeringer

Won't have:

- Som butiksadministrator vil jeg vise relevante reklamer for mine kunder når de søger på en vare så kunden har mulighed for at se butikkens aktuelle tilbud
- Som kunde vil jeg kunne se min egen placering på butikkens plantegning, for at relatere det til produkters placering
- Som kunde vil jeg gerne have en rutevejledning til den søgte vare

Efter implementering af de højest prioriteret funktionaliteter vil det endelige produkt omfatte følgende dele:

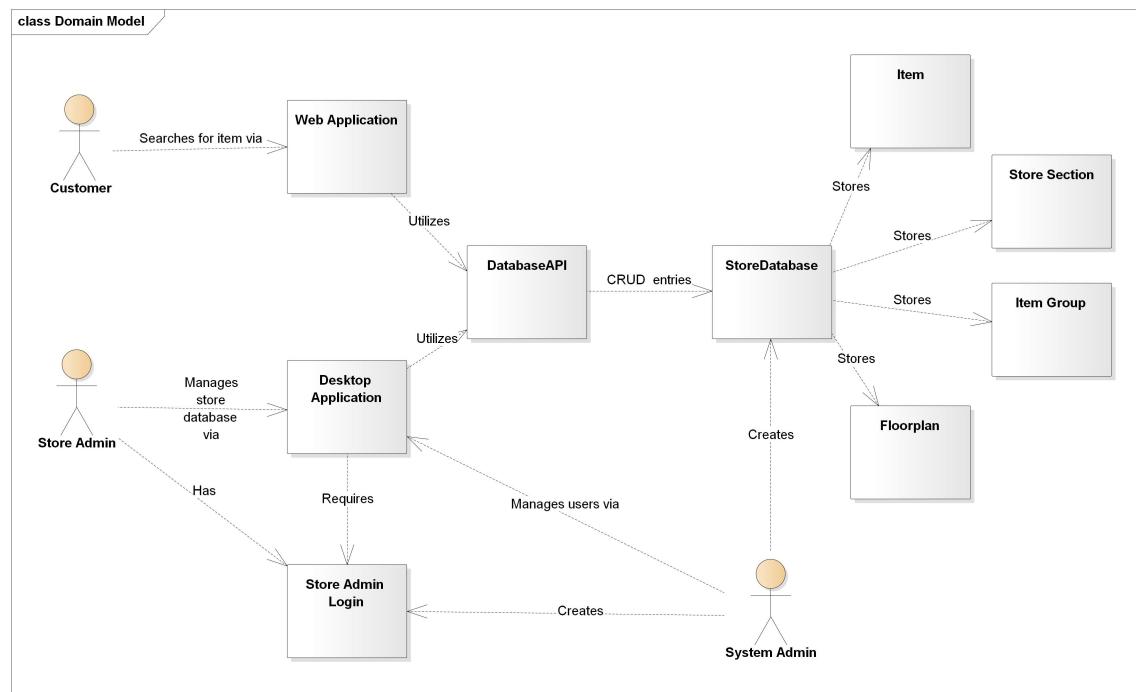
- En web applikation, hvor en kunde kan søge på en vare, via et søgefelt, og få vist dennes placering
- En desktop applikation, hvor en butiksadministrator kan oprette en plantegning, sektioner, varegrupper og varer
- En database, som indeholder plantegning, varelister og vare placeringer
- En API, til tilgang af databasens data

3 Systemarkitektur

I systemarkitekturen beskrives systemets grænseflader, databasernes struktur samt API'en til tilgang af databasen. Grænsefladerne beskrives ved hjælp af en domænemodel og databasens struktur defineres ved hjælp af entity relationship diagrammer. Under arkitekturforløbet tages der en række valg, som påvirker produktet senere i forløbet. Disse valg og deres begrundelser fremvises også under dette afsnit.

3.1 Domænemodel

Domænemodelen på figur 6 illustrerer aktørernes interaktioner med systemet, interaktioner mellem elementerne i systemet samt systemets grænseflader.



Figur 6: Domænemodel for Locatiles

Følgende er beskrivelser til domænemodellens aktører og elementer:

System Admin	Systemadministratoren står for at oprette en butiksadministrators login samt oprette tilhørende butiksdatabase.
Store Admin	Butiksadministratoren kan administrere varer, varegrupper, plantegninger og sektioner på butikkens database.
Customer	Kunden kan via web applikationen søge efter varer med henblik på at få vist varens placering i butikken.

Web Application	Web applikationen bruges af kunden til at søge efter en vares placering i butikken.
Desktop Application	Desktop applikationen bruges af systemadministratoren og butiksadministratoren. Systemadministratoren bruger applikationen til at oprette brugere til systemet. Butiksadministratoren bruger applikationen til at administrere varer, varegrupper, plantegninger og sektioner.
Admin Login	En brugerkonto for en butiksadministrator.
Database API	En API der bliver brugt af web applikationen og desktop applikationen til at tilgå databasen. API'en afskærmer databasen fra klienten, for at opnå en lavere kobling mellem klientens kode og potentielle datakilder.
Store Database	Databasen hvori alt data omkring varer, sektioner, varegrupper og plantegninger for en enkelt butik er gemt.
Item	Data gemt i databasen, indeholder oplysninger om en enkelt vare i butikken.
Store Section	Data gemt i databasen. En sektion er et punkt på plantegningen som kan tildeles varer. En sektion repræsenterer en fysisk reol, køledisk eller vare-ø.
Item Group	Data gemt i databasen. En varegruppe er en indeling af lignende varer under et overordnet navn.
Floorplan	Data gemt i databasen. En plantegning for butikken hvorpå der kan placeres sektioner.

3.2 Arkitektur valg

Det er under arkitekturfasen at mange kritiske beslutninger for et produkt bliver taget, da disse beslutninger udgør produktets fundament og som alle valg senere i udviklingsforløbet bygger på. I sidste ende er disse beslutninger de dyreste at rette op på efter de er blevet taget. Derfor brugte gruppen tid i starten af projektforløbet på at overveje samt diskutere hvilke teknologier der skulle bruges for at kunne realisere systemet. Følgende afsnit beskriver de teknologier og frameworks der bliver brugt under udviklingen af produktet samt begrundelser for disse valg.

På baggrund af figur 3 i projektets systembeskrivelse, kan følgende centrale elementer identificeres:

- Desktop applikation
- Central server
- Database
- Web applikation

Desktop applikationen og web applikationen udvikles ved brug af .NET frameworket [3] med hoveddelen af udviklingen foregående i Visual Studio. Dette framework tilbyder mange fordele til projektet, som vil blive yderligere specifiseret i dette afsnit.

3.2.1 Desktop applikation

Desktop applikationen er i ikke funktionelle krav, afsnit 2.5, s. 11, specificeret til at skulle kunne afvikles på Windows 10.

På baggrund af dette blev Microsofts WPF [4] teknologi valgt til at designe udseendet og udvikle desktop applikationen i. WPF er en del af .NET frameworket og har nogle stærke fordele for projektet:

- Applikationens brugergrænseflade kan designes visuelt i XAML med yderligere drag and drop funktionalitet direkte inde fra Visual Studio
- Der eksisterer et stort sæt af prædefinerede kontroller som kan gøres brug af uden ekstra programmeringstid
- Der er mulighed for at designe WPF applikationer ved brug af GUI arkitekturmønstre såsom MVVM. Dette gør det muligt at skrive genanvendeligt forretningslogik

Ved at bruge WPF kan der spares meget tid på udvikling af brugergrænsefladen til desktop applikationen, samtidig med at den kan få et professionelt udseende. Dette giver mere tid til at kunne fokusere på udvikling af forretningslogik.

3.2.2 Central server

Til hosting af den centrale server blev en SQL Server [5] hos Microsoft Azure [6] brugt.

Hosting hos Microsoft Azure giver god mening for projektet, da:

- Det ligger indenfor Microsofts økosystem og er derfor velintegreret med Visual Studio, hvor man kan administrere tilknyttede databaser, web applikationer og sin Azure konto

- Microsoft tilbyder Developer Program Benefit, hvilket er et 12 måneders program, hvor der gives 200 kr hver måned til services, såsom hostede databaser eller web applikationer
- Microsoft Azure tilbyder hosting af både SQL servere samt ASP.NET web applikationer, hvilket der skal bruges til projektet
- Microsoft Azure tilbyder en skalérbar service, i det tilfælde at forbruget pludseligt vokser

3.2.3 Database

Til projektet gøres brug af en MS SQL database. En relationel database giver god mulighed for at modellere butik- og brugerdbaserne.

Til hosting af SQL databasen bruges Microsofts Azure MS SQL database, da det som førnævnt er det muligt at få en gratis SQL database hostet via Developer Program Benefit. Da denne er en del af Microsoft Azure, er det også velintegreret med Visual Studio, hvilket betyder at databasen kan administreres derfra.

Til kommunikation med databasen gøres brug af ADO.NET [7]. ADO.NET er også en del af .NET frameworket og tilbyder en nem måde at kommunikere med datakilder på.

3.2.4 Web applikation

Web applikationen bliver udviklet ved brug af teknologien ASP.NET [8], som også er en del af .NET frameworket.

ASP.NET har gode fordele for projektet, da:

- ASP.NET applikationer kan hostes på Microsoft Azure, hvilket vil sige at det kan betales via førnævnte Developer Program Benefit
- Da ASP.NET også er en del af Microsofts økosystem med Azure kan ASP.NET applikationer deployes fra Visual Studio
- Der er mulighed for at designe ASP.NET applikationer ved brug af GUI arkitekturmønstre såsom MVC, hvilket betyder at det kan deles om en fælles kodebase med WPF desktop applikationen
- ASP.NET kan bruges til at implementere en dynamisk hjemmeside der kan skalere dets brugergrænseflade efter størrelsen på brugernes skærme

3.2.5 Programmeringssprog

Til implementering af desktop applikation og web applikationen gøres der brug af C#.

For projektet gav det god mening at bruge C#, da:

- Det er et garbage collected sprog, hvilket simplificerer memory management i stor grad. Dette minimerer risikoen for hukommelsesfejl i systemet og kan derfor være med til at effektivisere udviklingstiden for at opnå et stabilt produkt
- Det er en integreret del af .NET frameworket. Desktop applikationen og web applikationen gør brug af teknologierne WPF og ASP.NET, hvilket primært gør brug af C#.

- Der kan skrives klassebiblioteker, såsom database API, som let kan deles mellem kodebaserne for desktop applikationen og web applikationen

Overordnet set er det forsøgt at skabe et velintegreret miljø af arbejdsværktøjer som er med til at skabe et effektivt workflow for holdet samtidig med at produktet kan realiseres.

3.2.6 HTTP

Til kommunikation mellem web applikationen hostet på den centrale server og klienter, bruges der bagomliggende HTTP protokol.

3.2.7 Bootstrap

Til udvikling af web applikationens brugergrænseflade bruges Bootstrap [9]. Bootstrap gav nogle gode fordele for projektet, da:

- Det indeholder grid layout funktionalitet, hvilket betyder at projektets web applikation kan skalere ordenligt mellem mobile enheder og større skærme
- Det indeholder prædefinerede kontroller, så der spares tid på at opsætte brugergrænsefladen

3.2.8 jQuery

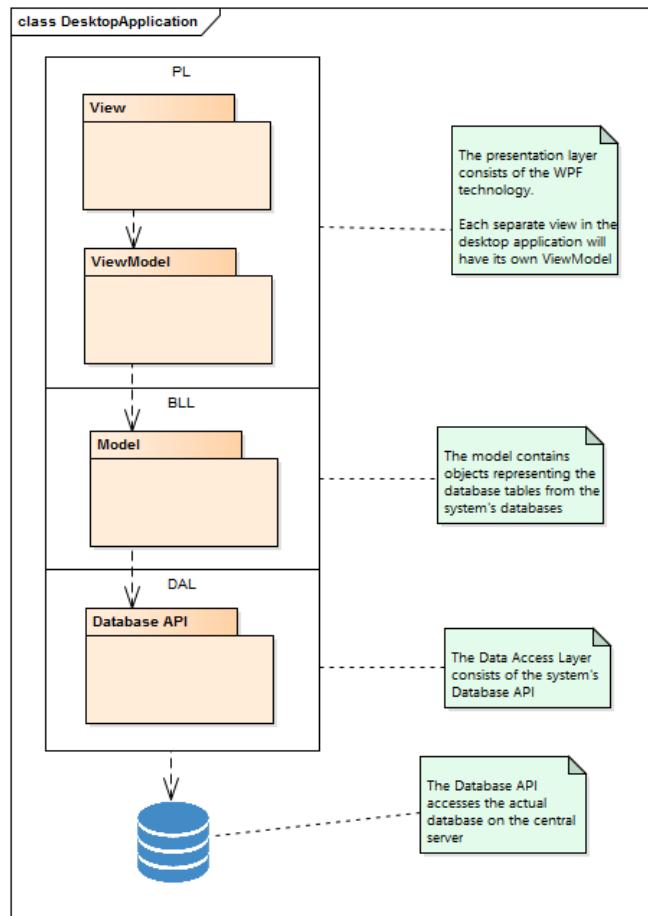
Til udvikling af web applikationen bruges jQuery [10]. jQuery har mange funktionaliteter, dog blev det i projektet primært brugt til at lettergøre JavaScript programmering af brugergrænsefladen. Dette skyldes at man med jQuery har mulighed for at tilgå web applikationens Document Object Model (DOM) på en mere effektiv måde.

3.3 Brugergrænseflader

Systemet indeholder to typer grafiske applikationer: web applikationen og desktop applikationen. Begge typer er opbygget med en 3-lags arkitektur, der separerer tre ansvar i de tre lag; presentation layer (PL), business logic layer (BLL) og data access layer (DAL). På denne måde kan forretningslogikken og datatilgangslogikken bruges på tværs af platforme. I dette system er det kun datatilgangslogikken, der deles mellem desktop applikation og web applikation, da der på hver platform er brug for forskellige funktionaliteter.

3.3.1 Desktop applikation

Desktop applikationen er opbygget ved brug af Model-View-ViewModel mønsteret, se bilag 1 - MVVM . På figur 7 ses en oversigt af hvilke moduler der indgår i applikationens tre lag samt en kort beskrivelse omkring modulernes indhold.



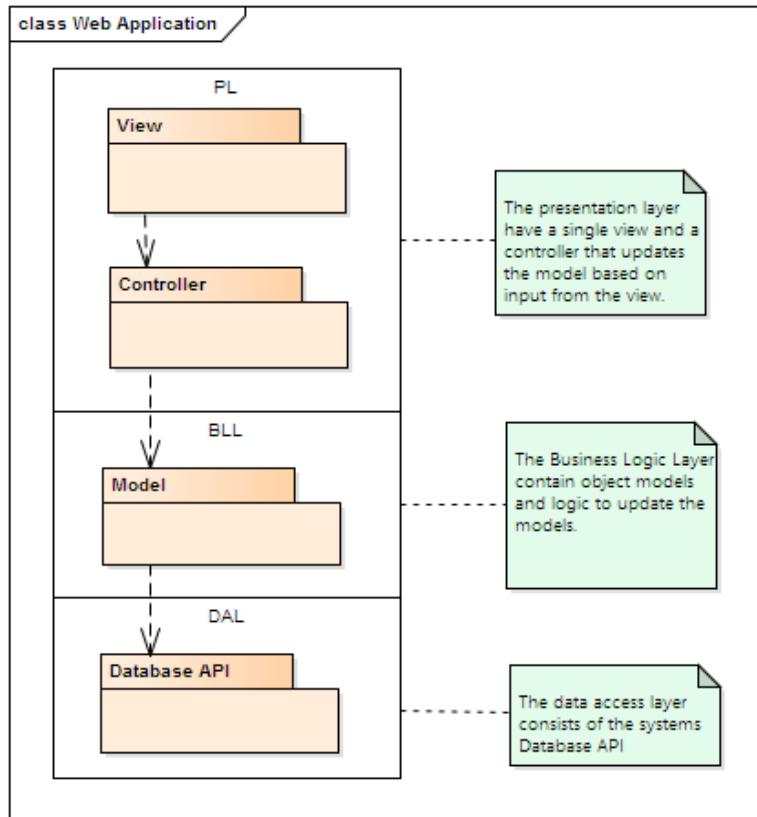
Figur 7: Desktop applikationens 3-lags arkitektur baseret på MVVM

Figur 7 præsenterer den idelle opbygning af desktop applikationen. Her vil de øvre lag gøre

brug af laget umiddelbart under den selv. I praksis gælder dette ikke, da der i nogle tilfælde ikke sker nogen databehandling i forretningslaget. Derfor sker det at viewmodellen forbipasserer forretningslaget ved at kalde direkte ned til datatilgangslaget.

3.3.2 Web applikation

Web applikationen er opbygget med Model-View-Controller mønsteret, se bilag 2 - MVC. På figur 8 vises de moduler der indgår applikationens tre lagt samt en kort beskrivelse af modulernes indhold.



Figur 8: Web applikationens 3-lags arkitektur baseret på MVC

Figur 8 viser strukturen for web applikationen. Web applikationen er udviklet med ASP.NET MVC. Da MVC er integreret i ASP.NET overholder web applikationen MVC strukturen.

3.4 Database

Følgende afsnit beskriver arkitekturen for bruger databasen samt de enkelte butikkers database. Dette dokumenteres ved hjælp af entity relationship digrammer (ERD).

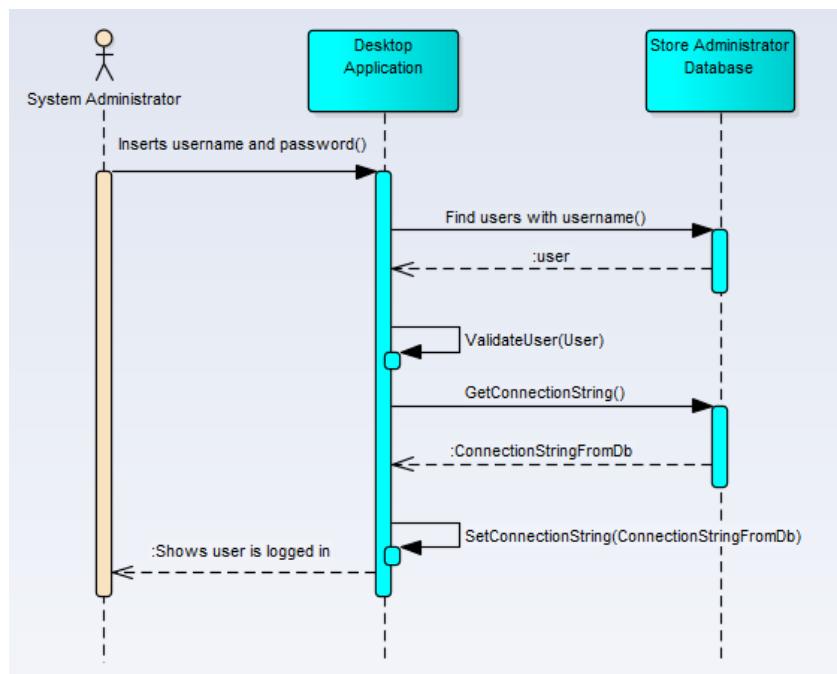
3.4.1 Brugerdatabase

På figur 9 ses ER diagrammet for bruger databasen. Denne database indeholder brugeroplysninger for alle butiksadministratorer, samt oplysninger til butiksadministratorens tilknyttede butiksdatabase.



Figur 9: ERD for bruger databasen

Bruger databasen består af én entitet, som indeholder brugeroplysninger, såsom brugernavn og password, samt en connection string til en brugers tilknyttede butiksdatabase. På figur 10 ses et sekvensdiagram, der viser hvordan desktop applikationen gør brug af bruger databasen til at validere brugere af systemet. Figuren illustrerer også sekvensen hvor desktop applikationen får en connection string til en butiksdatabase.

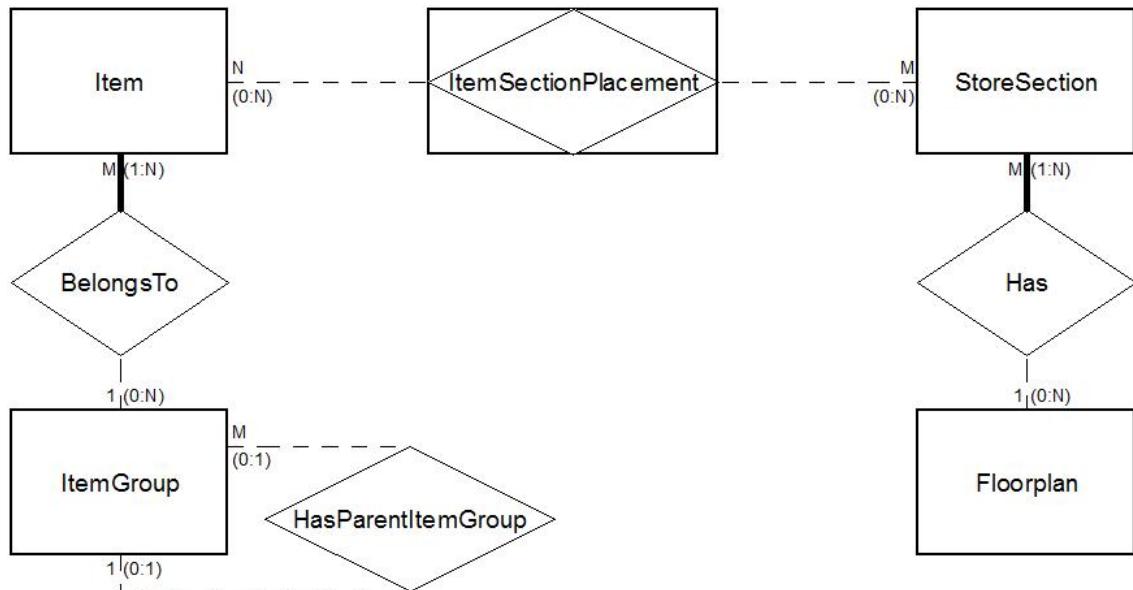


Figur 10: Sekvensen af desktop applikationens interaktion med bruger databasen

Under dette projektforløb vil bruger databasen ikke indgå som en del af det udviklede produkt og vil derfor ikke implementeret eller blive beskrevet yderligere.

3.4.2 Butiksdatabase

På figur 11 ses et ER diagram der beskriver butiksdatabasegens opbygning og relationerne mellem entiteterne. Hver butik har en instans af denne database.



Figur 11: ERD for butiksdatabase

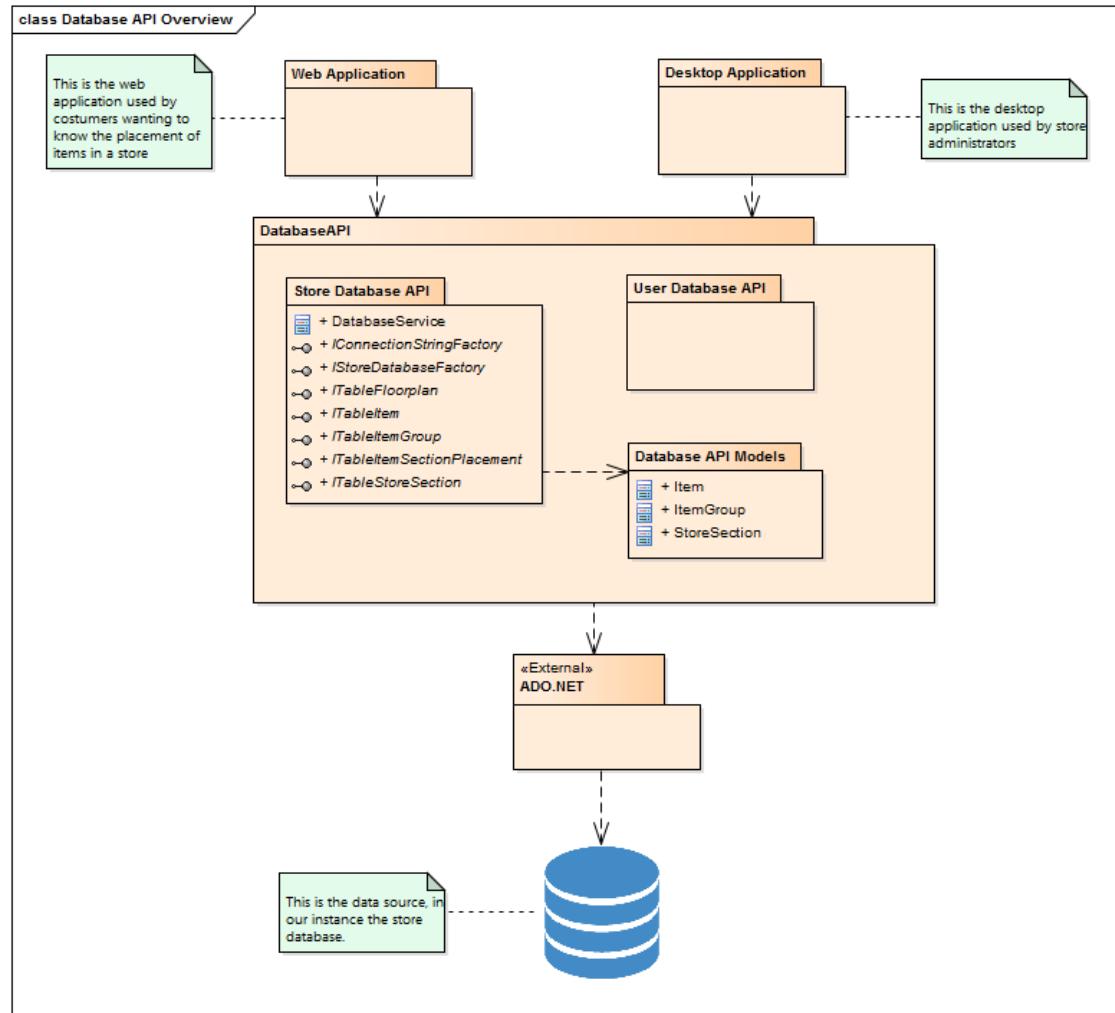
Butiksdatabase består af fire entiteter; item, itemgroup, storesection og floorplan. Ud fra designet, er der opsat nogle regler omkring entiteternes indbyrdes relationer. Reglerne er som følger:

- En plantegning kan have flere sektioner
- En sektion kan ikke eksistere uden at være tilskrevet en plantegning
- En sektion kan have flere varer
- En vare kan ligge på flere sektioner
- En vare skal tilhøre en varegruppe
- En vare kan ikke tilhøre mere end én varegruppe
- En varegruppe kan have flere varer
- En varegruppe kan være tildelt en over varegruppe

3.5 Database API

I dette afsnit beskrives arkitekturen for database API. Denne API tilgår data i databaserne, som web og desktop applikationerne anvender til forretnings- og præsentationslogik.

På figur 12 ses et pakkediagram for systemet. Disse pakkers indhold er beskrevet i tabel 1.



Figur 12: Pakkediagram af database API

Følgende tabel er en beskrivelse af ansvarsområderne for de pakker der indgår i database API.

Pakke	Ansvarsområde
Web Application	Giver kunden mulighed for at søge efter varer på butiks databasen
Desktop Application	Giver butiksadministratoren og systemadministratoren mulighed for at tilgå butiks databasen og bruger databasen
DatabaseAPI	Bruges af applikationerne til at tilgå databaserne
StoreDatabase API	Bruges af web og desktop applikationerne til at tilgå butiks databasen
UserDatabase API	Bruges af desktop applikationen til at tilgå bruger databasen
ADO.NET	Bruges af DatabaseAPI til at tilgå databaserne

Tabel 1: Beskrivelser for pakker som indgår i database API

Som nævnt under arkitekturafsnittet for bruger databasen, afsnit 3.4.1, s. 21, kommer det endelige produkt ikke til at indeholde en bruger database. Dermed kommer User Database API ikke til at indgå som en del af det færdige produkt.

4 Design & Implementering

Design og implementering af systemets komponenter er et resultat af de valg der blevet taget i planlægningen af systemets arkitektur. Her beskrives systemets komponenter i større implementeringsdetalje.

4.1 Database

Til persistering af data oprettes der to databaser, bruger databasen og butiks databasen. De følgende delafsnit beskriver databasernes design ved hjælp af ER diagrammer og beskrivelser af de tilhørende attributter.

4.1.1 Brugerdatabase

Denne database er ikke implementeret, men er designet så den let kan implementeres i tilfælde af fremtidigt arbejde. Bruger databasen indeholder én entitets klasse med loginoplysninger for en butiks administrator og en reference til den tilknyttede butiks database. Attributterne af denne entitets klasse beskrives yderligere i tabel 2.

StoreAdministrator	
PK	StoreAdministratorID
	UserName Password AssociatedDatabase

Figur 13: ERD med attributter for bruger databasen

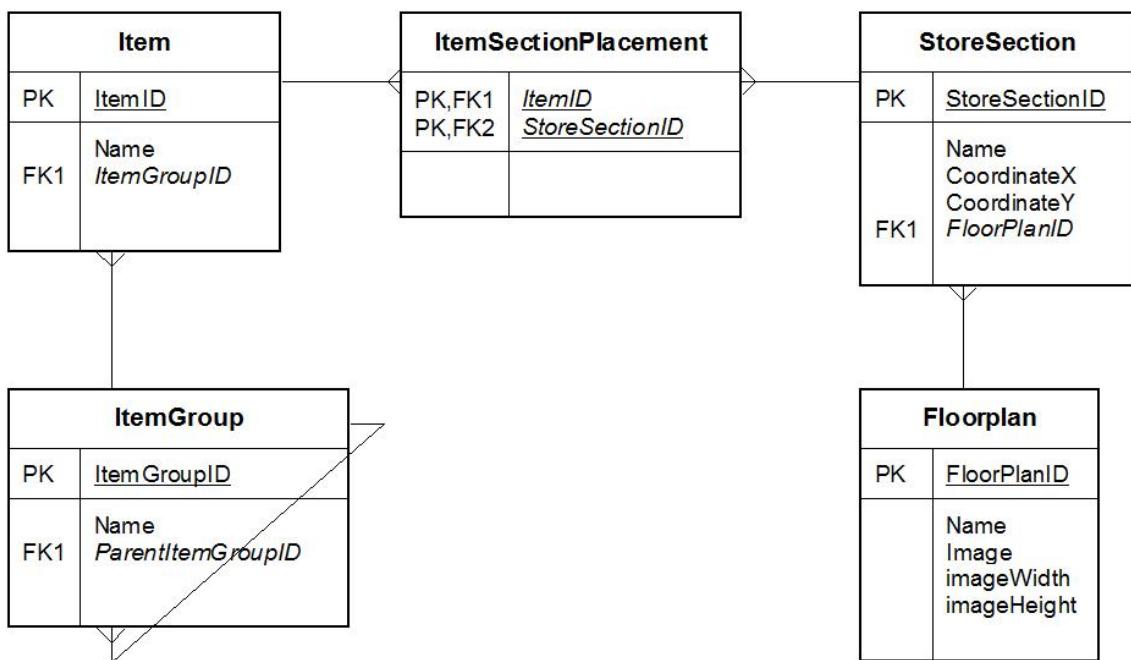
StoreAdministrator har følgende attributter:

Nøgle	Attribut	Type	Beskrivelse
PK	StoreAdministratorID	BIGINT	Række ID
	UserName	NVARCHAR	Indeholder butiks administrators brugernavn
	Password	NVARCHAR	Indeholder butiks administrators kodeord
	AssociatedDatabase	NVARCHAR	Butiks database tilknyttet til butiks administrator

Tabel 2: Beskrivelse af attributter i StoreAdministrator

4.1.2 Butiksdatabase

På figur 14 ses entitetsklasserne for butiks databasen. Hver entitetskasse har bestemte attributter som indeholder information omkring hver entitet.



Figur 14: ERD med attributter for butiks databasen

ItemGroup har følgende attributter:

Nøgle	Attribut	Type	Beskrivelse
PK	ItemGoupID	BIGINT	Række ID
	Name	NVARCHAR	Navnet på varegruppen
FK	ParentItemGroupID	BIGINT	ID'et på varegruppens over varegruppe

Tabel 3: Beskrivelse af attributter i ItemGroup

Item har følgende attributter:

Nøgle	Attribut	Type	Beskrivelse
PK	ItemID	BIGINT	Række ID
	Name	NVARCHAR	Varens navn
FK	ItemGroupID	BIGINT	ID på varegruppen som varen tilhører

Tabel 4: Beskrivelse af attributter i Item

ItemSectionPlacement har følgende attributter:

Nøgle	Attribut	Type	Beskrivelse
PK, FK	ItemID	BIGINT	ID på en vare
PK,FK	StoreSectionID	BIGINT	ID på en sektion

Tabel 5: Beskrivelse af attributter i ItemSectionPlacement

StoreSection har følgende attributter:

Nøgle	Attribut	Type	Beskrivelse
PK	StoreSectionID	BIGINT	Række ID
	Name	NVARCHAR	Sektionens navn
	CoordinateX	BIGINT	X-koordinat til sektionens placering
	CoordinateY	BIGINT	Y-koordinat til sektionens placering
FK	FloorPlanID	BIGINT	ID på plantegningen som sektionen er placeret på

Tabel 6: Beskrivelse af attributter i StoreSection

Floorplan har følgende attributter:

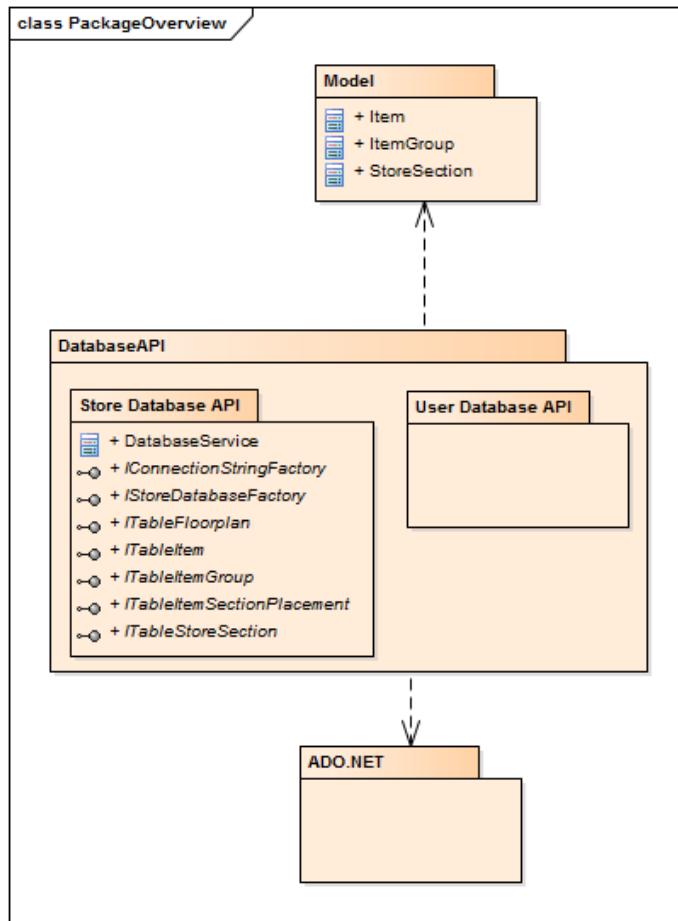
Nøgle	Attribut	Type	Beskrivelse
PK	FloorplanID	BIGINT	Række ID
	Name	NVARCHAR	Navn på plantegning
	Image	BLOB	Binær representation af plantegning
	imageWidth	BIGINT	Bredde i pixels på plantegning
	imageHeight	BIGINT	Højde i pixels på plantegning

Tabel 7: Beskrivelse af attributter i Floorplan

4.2 Database API

Følgende afsnit beskriver designet for database API, som bruges af web og desktop applikationerne til at tilgå databasens indhold.

På figur 15 ses et pakkediagram af Database API, dets moduler, og andre dele af systemet som den gør brug af.



Figur 15: Database API med dets moduler samt associeringer til andre dele af systemet

Her kan det ses at Database API består af to moduler, Store Database API samt User Database API. Modulerne repræsenterer tilgang til hver deres database i systemet; én til butikkens database og én til brugerdatabase.

På tabel 8 ses en beskrivelse af klassernes og interfacenes ansvarsområder.

Klasse/Interface	Ansvarsområde
DatabaseService	Denne klasse bruges af klienter eller viewmodeller til at udføre CRUD operationer på tabeller i butikkens database.
IConnectionStringFactory	Dette interface implementeres af konkrete connection-string factories. En connectionstring factory genererer en connectionstring til en specifik database samt database type.
IStoreDatabaseFactory	Dette interface implementeres af konkrete store database factories. Disse factories sørger for at oprette tabeller til den korrekte database type, afhængig af hvad klienten ønsker.
ITableFloorplan	Dette interface repræsenterer tabellen for plantegninger i butikkens database, og dets metoder bruges til at manipulere tabellens data.
ITableItem	Dette interface repræsenterer tabellen for varer i butikkens database, og dets metoder bruges til at manipulere tabellens data.
ITableItemGroup	Dette interface repræsenterer tabellen for varegrupper i butikkens database, og dets metoder bruges til at manipulere tabellens data.
ITableItemSectionPlacement	Dette interface repræsenterer tabellen for varers placeringer i butikkens sektioner, og dets metoder bruges til at manipulere tabellens data.
ITableStoreSection	Dette interface repræsenterer tabellen for butikkens sektioner, og dets metoder bruges til at manipulere tabellens data.

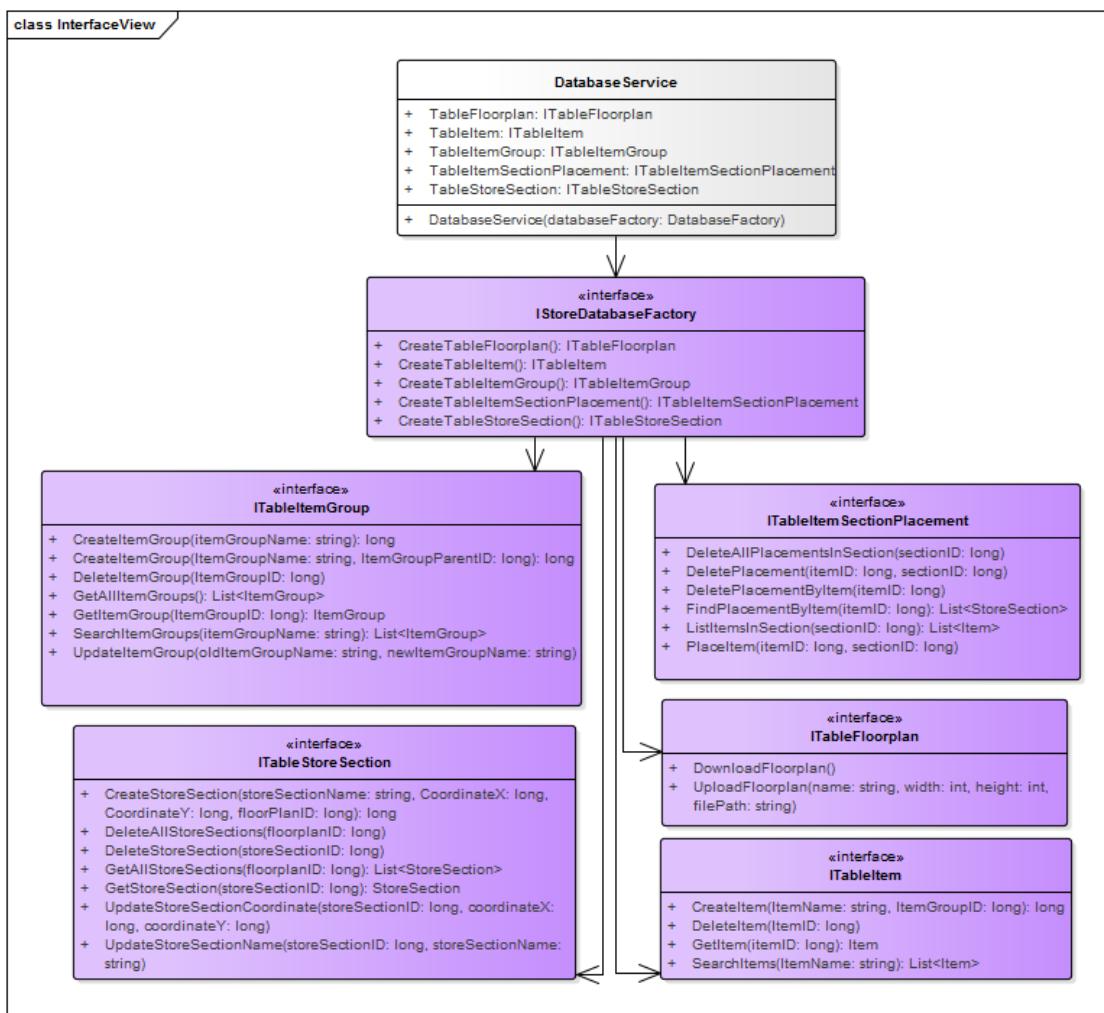
Tabel 8: Ansvarsområder for Store Database API klasser samt interfaces

Følgende afsnit beskriver interfaces samt klasser i disse to moduler i en større detaljegrad.

4.2.1 Store database API

Store Database API bruges til at tilgå en butiksdatabase, for at kunne udføre operationer som oprettelse af sektioner og søgning på varer.

På figur 16 ses en oversigt af klasserne som udgør Database API, DatabaseService klassen samt dets associerede interface IStoreDatabaseFactory. Interfaces som IStoreDatabaseFactory gør brug af, er også vist her.



Figur 16: Oversigt af grænsefladerne i database API

Følgende tabeller præsenterer klassebeskrivelser for figur 16.

StoreDatabaseService

På tabel 9 ses beskrivelser for attributterne af StoreDatabaseService.

Attribut	Type	Beskrivelse
TableFloorplan	ITableFloorplan	Bruges til at tilgå tabellen for butikkens plantegning, for at kunne opdatere og download den nuværende plantegning.
TableItem	ITableItem	Bruges til at tilgå tabellen for butikkens vare, for at kunne indsætte, opdatere og slette dem.
TableItemGroup	ITableItemGroup	Bruges til at tilgå tabellen for butikkens varegrupper, for at kunne indsætte, opdatere og slette dem.
TableItemSectionPlacement	ITableItemSectionPlacement	Bruges til at tilgå tabellen for butikkens placering af varer i sektioner, for at kunne indsætte varer i sektioner.
TableStoreSection	ITableStoreSection	Bruges til at tilgå tabellen for butikkens sektioner, for at kunne oprette og slette sektioner.

Tabel 9: Attributter til DatabaseService

På tabel 10 ses beskrivelser af metoderne for DatabaseService.

Metode	Parametre	Returværdi	Beskrivelse
StoreDatabaseService	databaseFactory : DatabaseFactory	Void	Constructor for klassen. Parametren databaseFactory skal angives, som bruges til at konstruere de konkrete tabel objekter afhængig af den ønskede database type.

Tabel 10: Metoder til StoreDatabaseService

IStoreDatabaseFactory

IStoreDatabaseFactory er en GOF Factory [11], der bruges til at instantiere tabel klasserne i DatabaseService klassen.

Metode	Returværdi	Beskrivelse
CreateTableFloorplan	ITableFloorplan	Denne funktion returnerer en konkret klasse, der implementerer ITableFloorplan
CreateTableItemGroup	ITableItemGroup	Denne funktion returnerer en konkret klasse, der implementerer interfacet ITableItemGroup
CreateTableItem	ITableItem	Denne funktion returnerer en konkret klasse, der implementerer interfacet ITableItem
CreateTableItemSectionPlacement	ITableItemSectionPlacement	Denne funktion returnerer en konkret klasse, der implementerer interfacet ITableItemSectionPlacement
CreateTableStoreSection	ITableStoreSection	Denne funktion returnerer en konkret klasse, der implementerer interfacet ITableStoreSection

Tabel 11: Metoder til IStoreDatabaseFactory

ITableStoreSection

ITableStoreSection er det interface der anvendes til at udføre CRUD operationer på storeSection tabellen i StoreDatabasen.

Metode	Parametre	Returværdi	Beskrivelse
CreateStoreSection	storeSectionName : string CoordinateX : long CoordinateY : long floorPlanID : long	long	Opretter en sektion som tilhører en floorplan. Funktionen returnerer store section ID'et
DeleteAllStoreSections	floorPlanID : long	void	Sletter alle sektioner på plan tegningen med den specificerede floorPlanID
DeleteStoreSection	storeSectionID : long	void	Sletter en sektion ud fra dens storeSectionID
GetAllStoreSections	floorPlanID : long	List<StoreSection>	Returnerer en liste af alle sektioner som tilhører en bestemt plan tegning ud fra floorPlanID
GetStoreSection	storeSectionID : long	StoreSection	Returnerer en sektion ud fra storeSectionID
UpdateStoreSectionCoordinate	storeSectionID : long coordinateX : long coordinateY : long	void	Ændrer sektionens koordinater til de angivne
UpdateStoreSectionName	storeSectionID : long StoreSectionName : string	void	Ændrer en sektions navn til det angivne ud fra sektionens storeSectionID

Tabel 12: Metoder til ITableStoreSection

ITableItemSectionPlacement

ITableItemSectionPlacement er det interface som anvendes til at udføre CRUD operationer på SectionPlacement tabellen i StoreDatabasen.

Metode	Parametre	Returværdi	Beskrivelse
DeleteAllPlacementsInSection	sectionID : long	void	Sletter alle vareplaceringer i sektionen med det specificerede sectionID
DeletePlacement	itemID : long sectionID : long	void	Sletter vareplaceringen med den specificerede itemID og sectionID
DeletePlacementsByItem	itemID : long	void	Sletter alle vareplaceringer der indeholder varen med det specificerede itemID
FindPlacementsByItem	itemID : long	List<StoreSection>	Returnerer en liste af alle sektioner indeholdende varen med det specificerede itemID
ListItemsInSection	sectionID : long	List Item	Returnerer en liste af alle varer indeholdt i sektionen med det specificerede sectionID
PlaceItem	itemID : long sectionID : long	void	Placerer varen med det specificerede itemID i sektionen med det specificerede sectionID

Tabel 13: Metoder til ITableItemSectionPlacement

ITableItemGroup

ITableItemGroup er det interface der anvendes til at udfører CRUD operationerne på ItemGroup tabellen i StoreDatabasen.

Metode	Parametre	Returværdi	Beskrivelse
CreateItemGroup	itemName : string itemGroupParentID : long	long	Opretter en ny ItemGroup i StoreDatabase ud fra itemName og itemGroupParentID
CreateItemGroup	itemName : string	long	Opretter en ny ItemGroup i StoreDatabase ud fra itemName
DeleteItemGroup	itemGroupID: long	void	Sletter en ItemGroup med specificeret itemGroupID
GetAllItemGroups		List<ItemGroup>	Returnerer en liste af alle eksisterende ItemGroups
GetItemGroup	itemGroupID : long	ItemGroup	Returnerer en ItemGroup med specificeret itemgroupID

Tabel 14: Metoder til ITableItemGroup

ITableItem

ITableItem er det interface der anvendes til at opretter, nedlægger og søger i Item tabellen i StoreDatabasen.

Metode	Parametre	Returværdi	Beskrivelse
CreateItem	itemName : string itemGroupID : long	long	Opretter et Item med navnet specificeret af itemName og varegruppen specificeret af itemGroupID
DeleteItem	itemID : long	void	Sletter varen specificeret af itemID
GetItem	itemID : long	Item	Returnerer et Item specificeret ved itemID
SearchItems	itemName : string	List<Item>	Returnerer en liste af vare som indeholder søgestrengen specificeret ved itemName

Tabel 15: Metoder til ITableItem

ITableFloorplan

ITableFloorplan er det interface der anvendes til at upload samt download butikkens plantegninger.

Metode	Parametre	Returværdi	Beskrivelse
DownloadFloorplan		void	Downloader den nuværende plantegning fra databasen som en lokal billedefil. Denne billedefil bruges af desktop applikationen for at kunne presentere plantegningen til butiksadministratorer
UploadFloorplan	name : string width : int height : int filepath: string	void	Uploader en plantegning med navnet specificeret af name. Filen til plantegningens billede er angivet af filepath, og billedets højde og bredde er specificeret af width og height.

Tabel 16: Metoder til ITableFloorplan

Brugte Design Patterns

DatabaseService gør primært brug af to design patterns. Repository Pattern [12] er det overordnede design pattern som DatabaseService repræsenterer. Dette betyder, at DatabaseService klassen bruges til tilgang af en butiksdatabase, for at kunne udføre CRUD operationer. Dette er præsenteret gennem en objekt-orienteret grænseflade.

DatabaseService klassen sørger også for at mappe de underliggende tabeller til en objekt-orienteret model, således at klienten ikke har noget kendskab til den underliggende datakilde. I teorien kunne DatabaseService repræsentere en hvilken som helst datakilde, så længe dets data mappes til objekter klienten forventer. Dette stemmer overens med de regler der gælder for at overholde Liskovs Substitutions Princip [13].

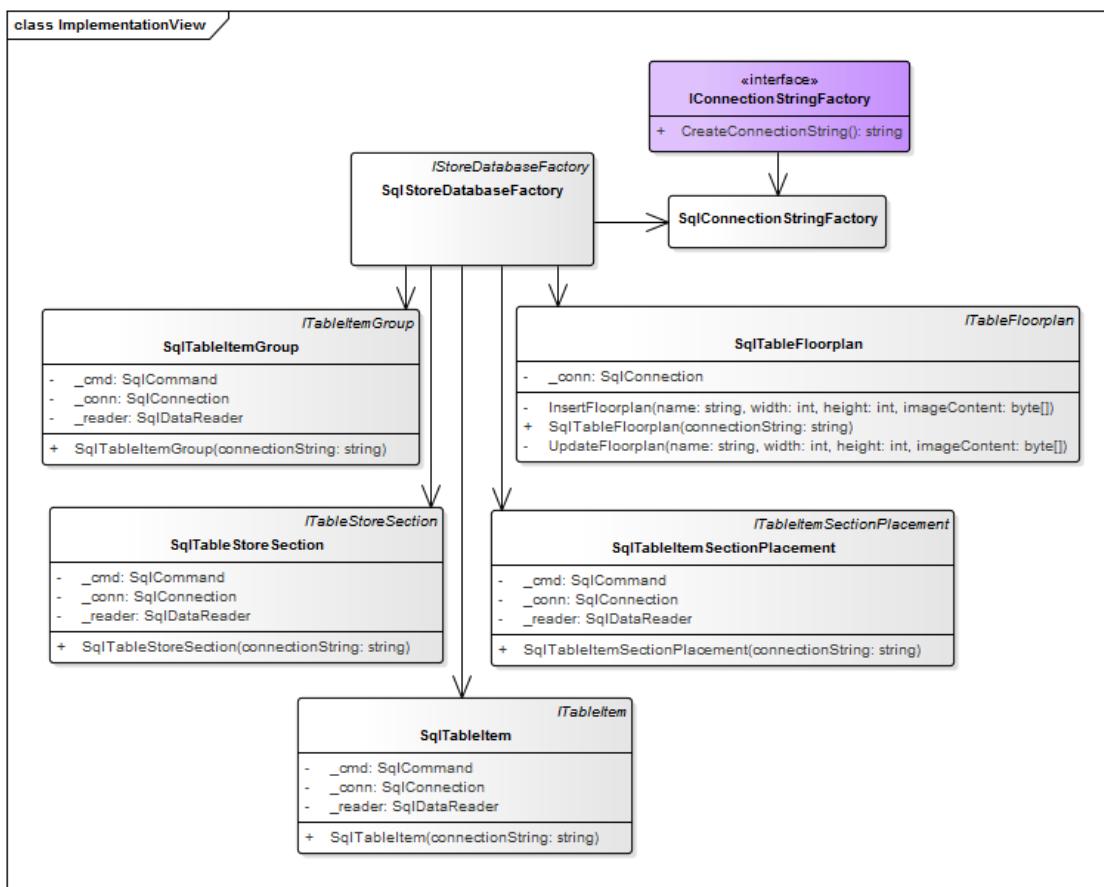
Ydermere gør DatabaseService brug af et Factory Pattern til konstruktion af de konkrete tabel klasser. I diagrammet på figur 16, indeholder ”DatabaseService” mange associeringer der skal konstrueres. Konkrete factories, som implementerer IStoreDatabaseFactory, konstruerer konkrete tabel klasser efter den bestemte type datakilde som DatabaseService gør brug af. Denne factory proces uddybes yderligere i sekvensdiagrammet på figur 18.

Implementering af Store Database API interfaces for MS SQL Databaser

Den nuværende implementering af Store Database API er en implementering til en MS SQL database.

På figur 17 ses klasserne der implementerer de forrige præsenterede interfaces af Store Database API for en SQL database. Yderligere er interfacet IConnectionStringFactory også introduceret på dette diagram.

Det tilsvarende interface som klasserne implementerer er indikeres øverst højre hjørne af klasserne. Interfacenes metoder er ikke vist og er implicit formodet implementeret. De metoder som stammer fra interfacene er ikke vist, så det er kun klassernes egne metoder som vises.



Figur 17: Oversigt af konkrete klasser som implementerer Store Database API

IConnectionStringFactory interfacet implementeres af konkrete ConnectionStringFactories, som genererer connectionstrings til specifikke databaser.

Følgende tabeller beskriver attributter og metoder af disse klasser. Alle klasserne har en constructor som tager en connectionstring til den database, som skal oprettes forbindelse til. Disse vil derfor ikke blive beskrevet yderligere.

Klasse/Interface	Ansvarsområde
IConnectionStringFactory	Dette interface bruges til at implementere konkrete connection string factories. Disse genererer en string med en connection string til en specifik database.
SqlConnectionStringFactory	Denne klasse genererer en connectionstring til butiksdbasen.

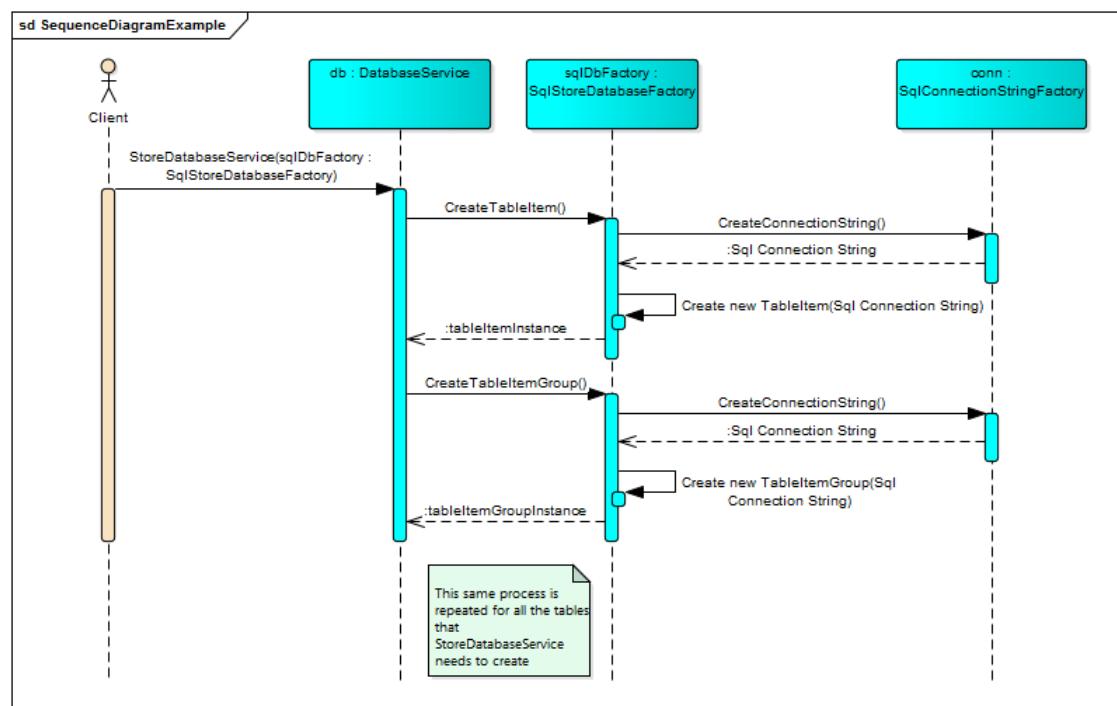
Tabel 17: Ansvarsområder for nyligt introducerede klasser og interfaces

Alle konkrete sqlklasser har desuden en eller flere af de følgende attributter:

Attribut	Type	Beskrivelse
_conn	SqlConnection	Bruges til at oprette en forbindelse til en SQL database med connection strengen specificeret igennem constructoren
_cmd	SqlCommand	Bruges til at specificere SQL sætningen der skal udføres på SQL databasen
_reader	SqlDataReader	Bruges til at aflæse returnerede tabel rækker fra SQL databasen efter at have eksekveret en SqlCommand på den

Tabel 18: Hyppigt brugte attributer i de konkrete SqlTable klasser

Figur 18 illustrerer byggeprocessen af tabelklasserne, TableItem og TableItemGroup, for StoreDatabaseService. Figuren viser kun oprettelsen af to tabelklasser. Det er dog den samme oprettelsesproces der sker for alle tabelklasser.

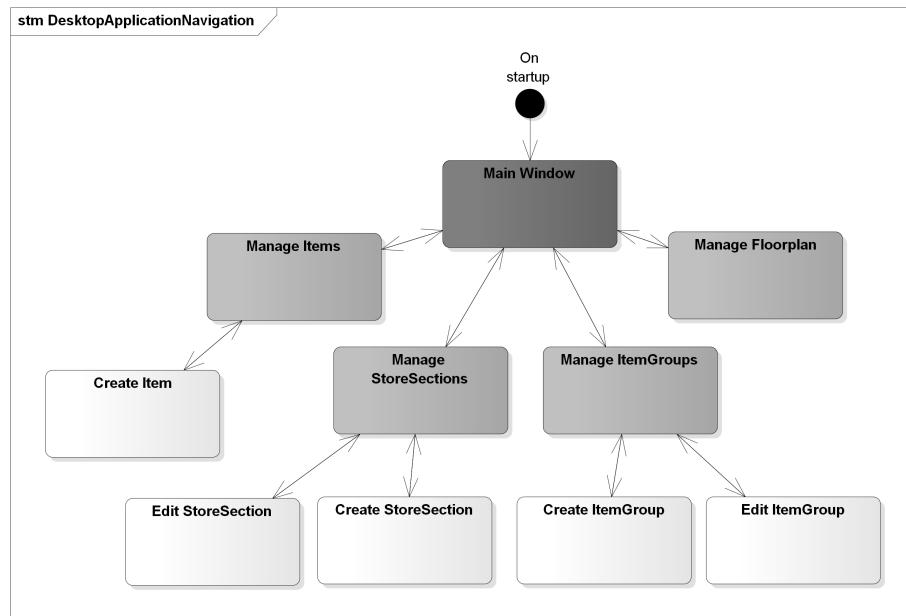


Figur 18: Byggeprocessen for oprettelse af tabel objekter

Oprettelsesprocess startes når en klient kalder `StoreDatabaseService` med en `SqIStoreDatabaseFactory`. Herefter bliver hver tabel klasse for en sql database oprettet.

4.3 Desktop Applikation

Dette afsnit dokumenterer designet af brugergrænsefladen for butiksadministratorens udsigt af desktop applikationen. Vinduer og de mest afgørende kontroller vil blive gennemgået med UML diagrammer, skærmdumps og beskrivelser. Desktop applikationen er implementeret efter et MVVM pattern, hvor hvert vindue fra hovedmenuen er et view. Disse views gør brug af kommandoer og properties som den tilhørende.viewmodel indeholder. Den data som viewmodellen fremviser kommer fra modellen, som tilgår databasen samt indeholder hjælpeklasser til at formater data.

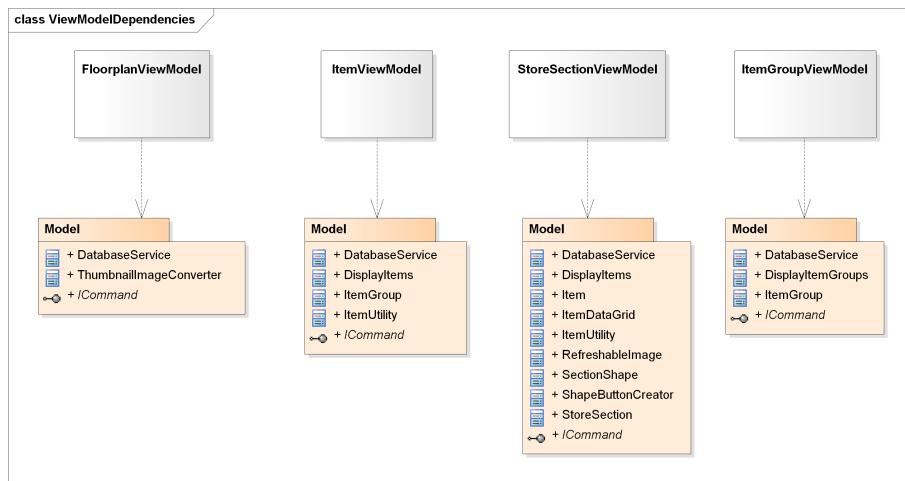


Figur 19: Tilstandsmaskine for desktop applikationen

På figur 19 ses tilstandsmaskinen for desktop applikationen hvor navigation mellem vinduerne tydeliggøres.

4.3.1 Model laget

Modellaget består af database API modeller samt en række af hjælpeklasser til at repræsentere data i det korrekte format. På figur 20 ses en grafisk repræsentation af modellaget, med tilhørende viewmodeller.



Figur 20: Modellag for desktop applikationen

Figur 20 viser model laget samt tilhørende viewmodeller. Hver.viewmodel gør brug af dele af modellen. Dette repræsenteres som pakker på figuren, som de forskellige viewmodeller er afhængige af. Indholdet af pakkerne er de klasser og interfaces som den bestemte.viewmodel gør brug af.

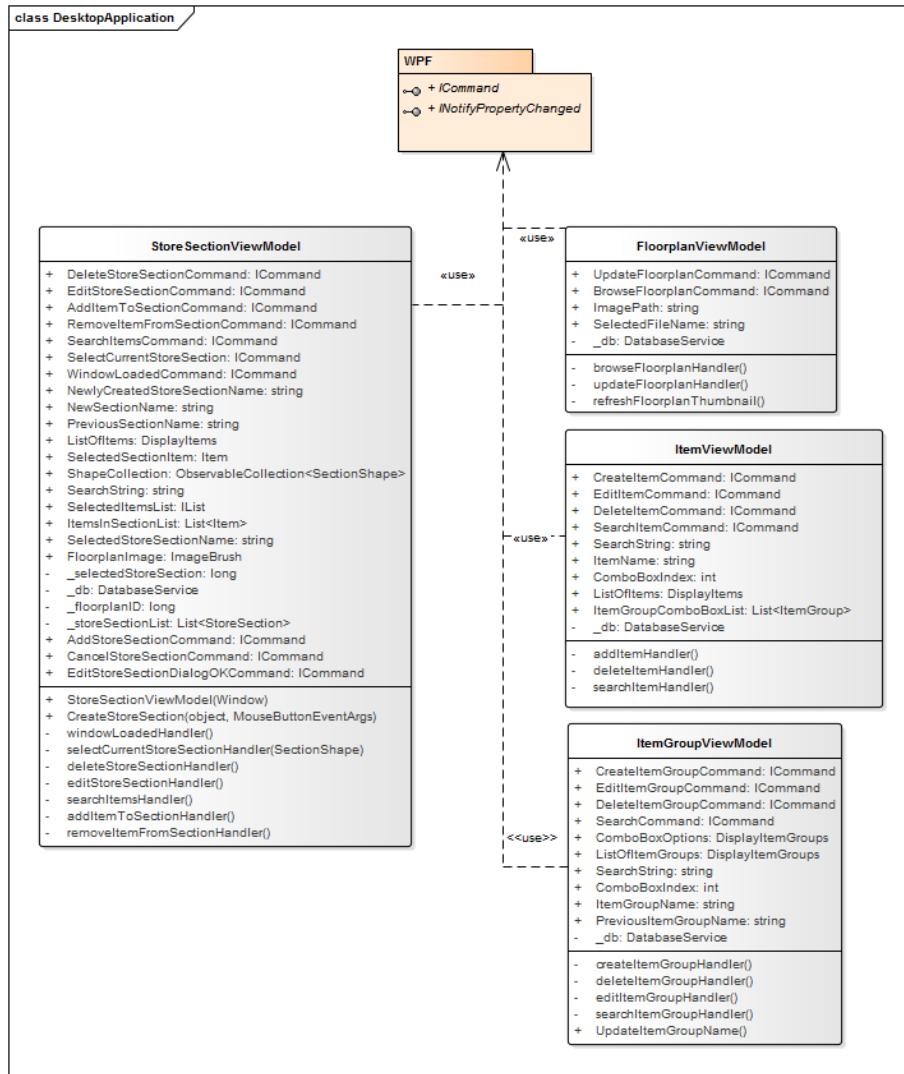
Følgende er en beskrivelse af klasser som modellaget består af.

Klasse Navn	Beskrivelse
DatabaseService	Database API model Bruges til tilgang af butiks databasen
Item	Database API model OO repræsentation af en vare
ItemGroup	Database API model OO repræsentation af en varegruppe
StoreSection	Database API mode OO repræsentation af en sektion
ThumbnailImageConverter	Sørger for at plantegninger kan opdateres på runtime
RefreshableImage	Repræsenterer et billede indlæst i RAM Bruge for at forhindre fil lås på det indlæste billede
DisplayItem	Indeholder information om en vare samt dens tilhørende varegruppe
DisplayItems	Indeholder information om en liste af DisplayItem
DisplayItemGroups	Indeholder en liste af ItemGroups
ItemUtility	Bruges til søgning af varer på databasen ud fra en søgestreng
ItemDataGrid	Udvidelse af DataGrid Bruges at kunne binde SelectedItems til en List property
SectionShape	Indeholder information om en sektion samt dens tilhørende shape
ShapeButtonCreator	Oprettet en shape til grafisk repræsentation af en sektion
ICommand	Bruges til kommandoer

Tabel 19: Klasse beskrivelse for modellaget

4.3.2 ViewModel laget

Viewmodel laget består af fire viewmodeller som hver præsenterer relevant data til de fire vinduer samt dialoger på brugergrænsefladen. Figur 21 ses et diagram hvor.viewmodelklasserne og deres tilhørende operationer og attributter er repræsenteret.



Figur 21: ViewModels til desktop applikationen

Figuren viser at alle viewmodellerne implementerer ICommand interfacet fra WPF. ICommand interfacet implementeres igennem bruget af RelayCommand [14], som gør at kommandoer fra et view kan bindes til en.viewmodel. Hver gang der sker en handling fra brugeren, som skal reageres på, bliver kommandoens tilhørende handler kaldt. Hvis viewmodellens properties ændres vil de relevante kontroller, som er bindet dertil, blive notificeret omkring dette og viewet vil blive opdateret.

Følgende er en beskrivelse af de.viewmodel klasser der indgår i viewmodellaget.

Klasse Navn	Ansvarsområder
StoreSectionViewModel	Står for håndtering og placering af sektioner plantegninger samt placering af varer på disse
ItemViewModel	Står for håndtering af varer blandt andet søgning, oprettelse og fjernelse
ItemGroupViewModel	Står for håndtering af varegrupper blandt andet søgning oprettelse og fjernelse
FloorplanViewModel	Står for opdatering af plantegning

Tabel 20: Klasse beskrivelse for viewmodellaget

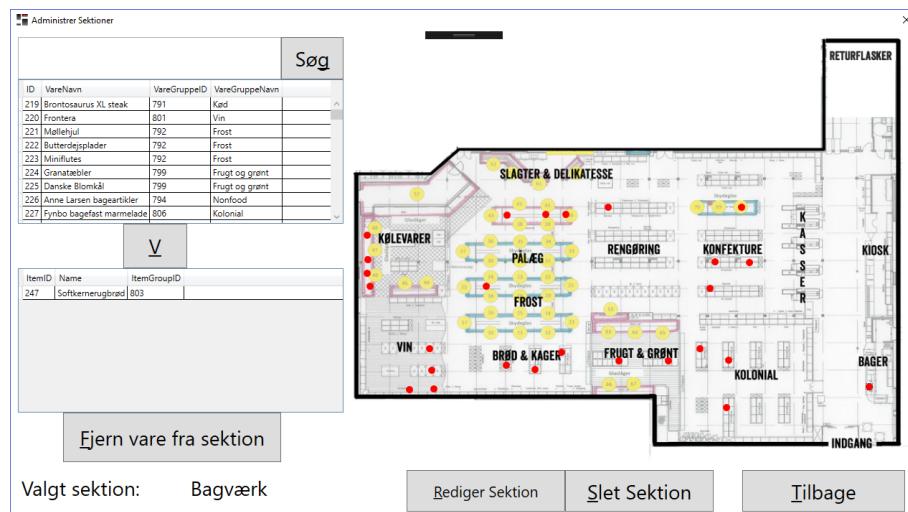
4.3.3 View laget

View laget består af fem vinduer, hvoraf den ene er hovedmenuen. Hovedmenuen omdirigerer brugeren til en af de fire andre vinduer, som har en tilknyttet.viewmodel. Dette gøres ved hjælp af click-events og vil ikke blive yderligere beskrevet. Brugeren kan fra disse vinduer åbne dialoger, som hver især også er et nyt view. Disse dialoger bruges til oprettelse af varer, varegrupper og sektioner.

Følgende afsnit beskriver de fire vinduer, der præsenterer data fra.viewmodelklasserne, samt de tilhørende dialoger. Kommandoerne, beskrevet i følgende tabel, er bundet til private handlers, som udfører de handlinger der skal udføres når en kommando bliver kaldt.

StoreSectionView

På StoreSection viewet bliver butikkens sektioner administreret. Her kan sektioner slettes og deres vareindhold kan administreres. Herfra kan der omdiriges til dialoger hvor sektion oprettelse og redigering kan ske.



Figur 22: StoreSection view

Følgende tabeller indeholder en beskrivelse af vinduets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Vindue	Indlæser sektioner, plantegning og varer fra databasen
Søg knap	Søger efter vare på databasen med den angivne søgestreng fra søge textbox
V knap	Indsætter de valgte varer fra vareliste til den valgte sektions vareliste
Fjern vare fra sektion knap	Fjerner den valgte vare fra sektionens varelisten
Rediger Sektion knap	Åbner dialog til redigering af sektionnavn
Slet Sektion knap	Sletter den valgte sektion fra plantegning området samt databasen
Sektion knap	Opdaterer den valgte sektion og sektions varelisten
Valgte Sektion label	Viser navnet på den valgte sektion
Søge textbox	Opdaterer SearchString med nyt brugerinput
Plantegning område	Fremviser plantegningen samt åbner dialog til oprettelse af sektion
Vareliste	Viser varer alle på databasen eller varer der passer til søgestrengen
Sektion vareliste	Viser varer i den valgte sektion

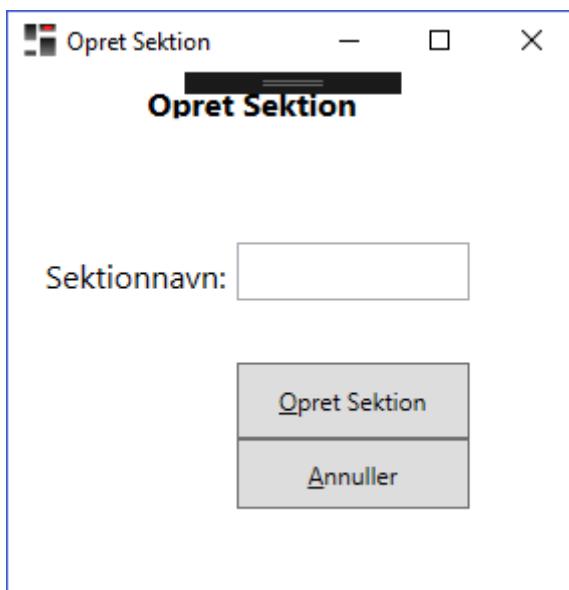
Tabel 21: StoreSection view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando /Event	Databinding
Vindue	WindowLoadedCommand	
Søg knap	SearchItemsCommand	
V knap	AddItemToSectionCommand	
Fjern vare fra sektion knap	RemoveItemFromSectionCommand	
Rediger Sektion knap	EditStoreSectionCommand	
Slet Sektion knap	DeleteStoreSectionCommand	
Sektion knap	SelectCurrentStoreSectionCommand	SectionShape.Shape
Valgte Sektion label		SelectedStoreSectionName
Søge textbox		SearchString
Plantegning område	CreateStoreSection(object, MouseButtonEventArgs)	FloorplanImage
Vareliste		ListOfItems.CurrentIndex ListOfItems SelectedItemsList
Sektion vareliste		ItemsInSectionList SelectedSectionItem

Tabel 22: StoreSection view kontroller med tilknyttede kommandoer, events og databinding

AddSectionView

På AddSection viewet kan der oprettes sektioner.



Figur 23: AddSection view

Følgende tabeller indeholder en beskrivelse af viewets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Opret sektion knap	Gemmer tekst fra Sektionnavn textbox i SectionName
Annuler knap	Lukker opret sektion dialogen
Sektionnavn textbox	brugerinput til sektionnavn

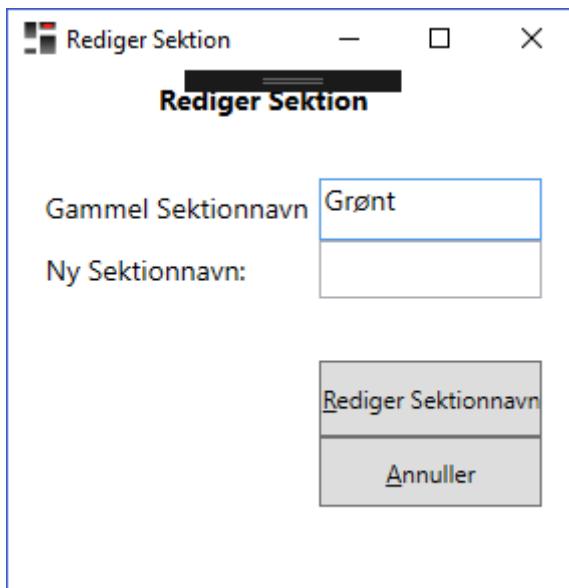
Tabel 23: AddSection view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Opret sektion knap	AddStoreSectionCommand	
Annuler knap	CancelStoreSectionCommand	
Sektionnavn textbox		NewlyCreatedStoreSectionName

Tabel 24: AddSection view kontroller med tilknyttede events og databindings

EditSectionView

På EditSection view kan en sektions navn redigeres.



Figur 24: EditSection view

Følgende tabeller indeholder en beskrivelse af viewets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Rediger sektionnavn knap	Lukker dialogen og angiver at den valgte sektions navn er ændret
Annuller knap	Lukker rediger sektion dialogen
Gammel sektionnavn textbox	Viser navnet på den sektion som skal ændres
Ny sektionnavn textbox	Brugerinput til ny sektionnavn

Tabel 25: EditSection view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Rediger sektionnavn	EditStoreSectionDialogOKCommand	
Annuller knap	CancelButton_Click(object, RoutedEventArgs)	
Gammel sektionnavn textbox		PreviousSectionName
Ny sektionnavn textbox		NewSectionName

Tabel 26: EditSection view kontroller med tilknyttede events og databindings

ItemView

På Item viewet bliver butikkens varer administreret. Her kan der søges efter varer og varer kan slettes. Herfra kan der omdirigeres til en dialog hvor nye varer kan oprettes.



Figur 25: Item view

Følgende tabeller indeholder en beskrivelse af vinduets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Opret vare knap	Åbner dialog til oprettelse af vare
Slet vare knap	Sletter den valgte vare fra vare listen
Søg knap	Søger efter vare på databasen med den angivne søgestreng fra søge textbox
Søge textbox	Opdaterer SearchString med nyt brugerinput
Vare liste	Viser alle varer på databasen eller varer der passer til søgestrengen

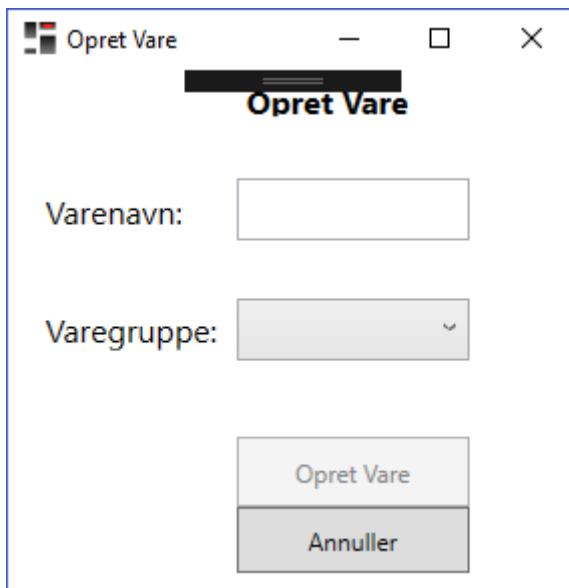
Tabel 27: Item view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Opret vare knap	AddItems_Click(object, RoutedEventArgs)	
Slet vare knap	DeleteItemCommand	
Søg knap	SearchItemsCommand	
Søge textbox		SearchString
Vareliste		ListOfItems.CurrentIndex ListOfItems

Tabel 28: Item view kontroller med tilknyttede kommandoer, events og databinding

AddItemView

På AddItem viewet kan der oprettes varer.



Figur 26: View til oprettelse af ny vare

Følgende tabeller indeholder en beskrivelse af viewets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Opret vare knap	Oprettet en vare med valgt varenavn og varegrupper
Annuler knap	Lukker opret vare dialogen
Vare navn textbox	Opdaterer ItemName med nyt brugerinput
Varegruppe combobox	Viser varegrupper på databasen

Tabel 29: AddItem view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Opret vare knap	CreateItemCommand	
Annuler knap	CancelButton_Click(object, RoutedEventArgs)	
Varenavn textbox		ItemName
Varegruppe combobox		ItemGroupComboBoxList ComboboxIndex

Tabel 30: AddItem view kontroller med tilknyttede kommandoer, events og databindings

ItemGroupView

På ItemGroup viewet administreres butikkens varegrupper. Her kan vare grupper slettes og søges på. Oprettelse og redigering af varegrupper sker i AddItemGroup og EditItemGroup viewet.



Figur 27: ItemGroup view

Følgende tabeller indeholder en beskrivelse af vinduets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Opret varegruppe knap	Åbner dialog til oprettelse af varegruppe og valg af overvaregruppe
Rediger varegruppe knap	Åbner dialog til redigering af varegruppe navn
Slet varegruppe knap	Sletter den valgte varegruppe fra varegruppelisten samt databasen
Søg knap	Søger efter varegruppe på databasen med den angivne søgestreng fra søge textbox
Søge textbox	Opdaterer SearchString med nyt brugerinput
Varegruppe liste	Viser alle varergrupper på databasen eller varegrupper der passer til søgestrengen

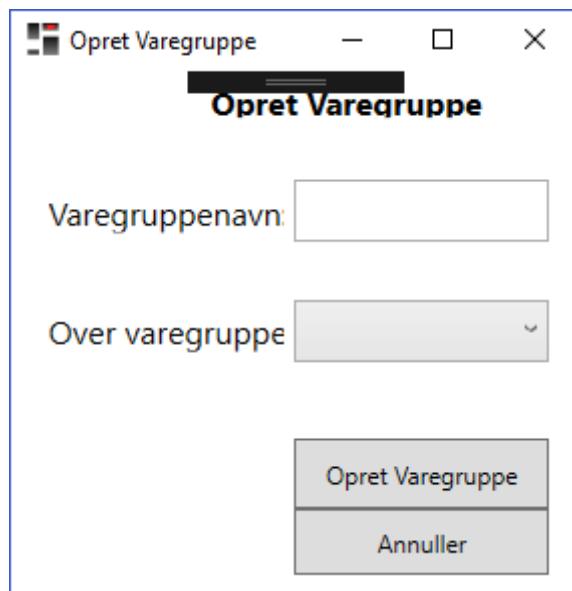
Tabel 31: ItemGroup view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Opret varegruppe knap	AddItems_Click(object, RoutedEventArgs)	
Rediger varegruppe knap	EditItems_Click(object, RoutedEventArgs)	
Slet varegruppe knap	DeleteItemGroupCommand	
Søg knap	SearchCommand	
Søge textbox		SearchString
Varegruppe liste		ListOfItems.CurrentIndex ListOfItems

Tabel 32: ItemGroup view kontroller med tilknyttede kommandoer, events og databinding

AddItemGroupView

På AddItemGroup viewet kan varegrupper oprettes.



Figur 28: AddItemGroup view

Følgende tabeller indeholder en beskrivelse af viewets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Opret varegruppe knap	Oprettet en varegruppe med valgt varegruppe navne og over varegruppe
Annuler knap	Lukker opret varegruppe dialogen
Varegruppe navn textbox	Opdaterer ItemGroupName med nyt brugerinput
Varegruppe combobox	Viser varegrupper fra databasen

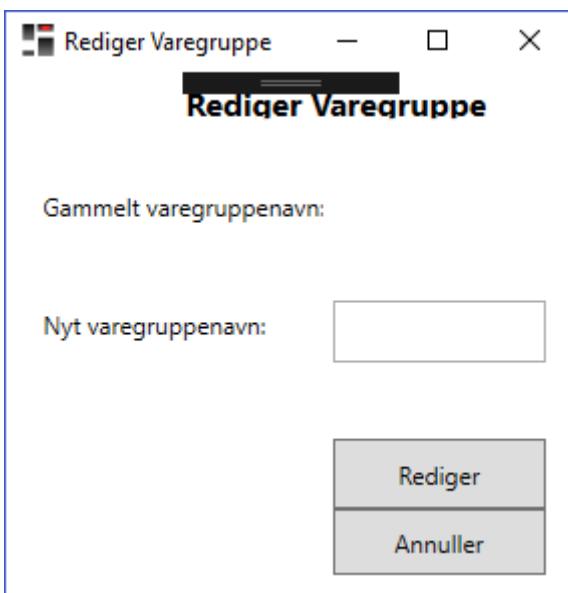
Tabel 33: AddItemGroup view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Opret varegruppe knap	CreateItemGroupCommand	
Annuler knap	CancelButton_Click(object, RoutedEventArgs)	
Varegruppe navn textbox		ItemGroupName
Varegruppe combobox		ComboBoxIndex ListOfItemGroups

Tabel 34: AddItemGroup view kontroller med tilknyttede kommandoer, events og databinding

EditItemGroupView

På AddItemGroup viewet kan varegrupper redigeres.



Figur 29: EditItemGroup view

Følgende tabeller indeholder en beskrivelse af viewets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Rediger knap	Opdaterer den valgte varegruppens navn
Annuler knap	Lukker rediger varegruppe dialogen
Gammel varegruppe navn label	Viser den valgte varegruppens navn
Varegruppe navn textbox	Opdaterer PreviousItemGroupName med nyt brugerinput

Tabel 35: EditItemGroup view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Rediger knap	EditItemGroupCommand	
Annuler knap	CancelButton_Click(object, RoutedEventArgs)	
Gammel varegruppe navn label		PreviousItemGroupName
Varegruppe navn textbox		ItemGroupName

Tabel 36: EditItemGroup view kontroller med tilknyttede kommandoer, events og databindings

FloorplanView

På Floorplan viewet uploades der en plantegning, hvorefter der vises et preview af denne.



Figur 30: Floorplan view

Følgende tabeller indeholder en beskrivelse af vinduets kontroller og de tilknyttede kommandoer, events og databindings.

Kontrol	Beskrivelse
Plantegning thumbnail	Viser den uploadedede plantegning
Sti textbox	Indeholder stien til den plantegning som skal uploades
Gennemse knap	Åbner stifinder til valgt af plantegning
Upload knap	Opdaterer databasen med en ny plantegning

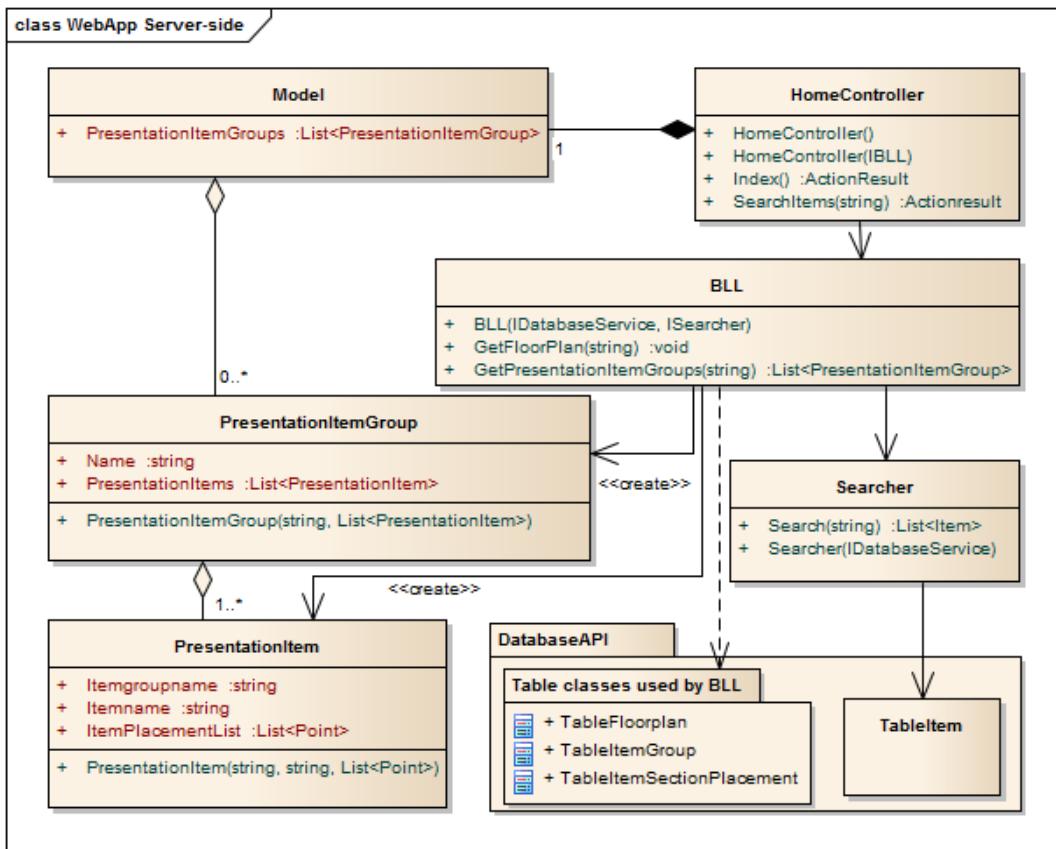
Tabel 37: Floorplan view kontroller samt beskrivelser

Kontrol	Tilknyttet Kommando/Event	Databinding
Plantegning thumbnail		ImagePath
Sti textbox		SelectedFileName
Gennemse knap	BrowseFloorplanCommand	
Upload knap	UpdateFloorplanCommand	

Tabel 38: Floorplan view kontroller med tilknyttede kommandoer og databindings

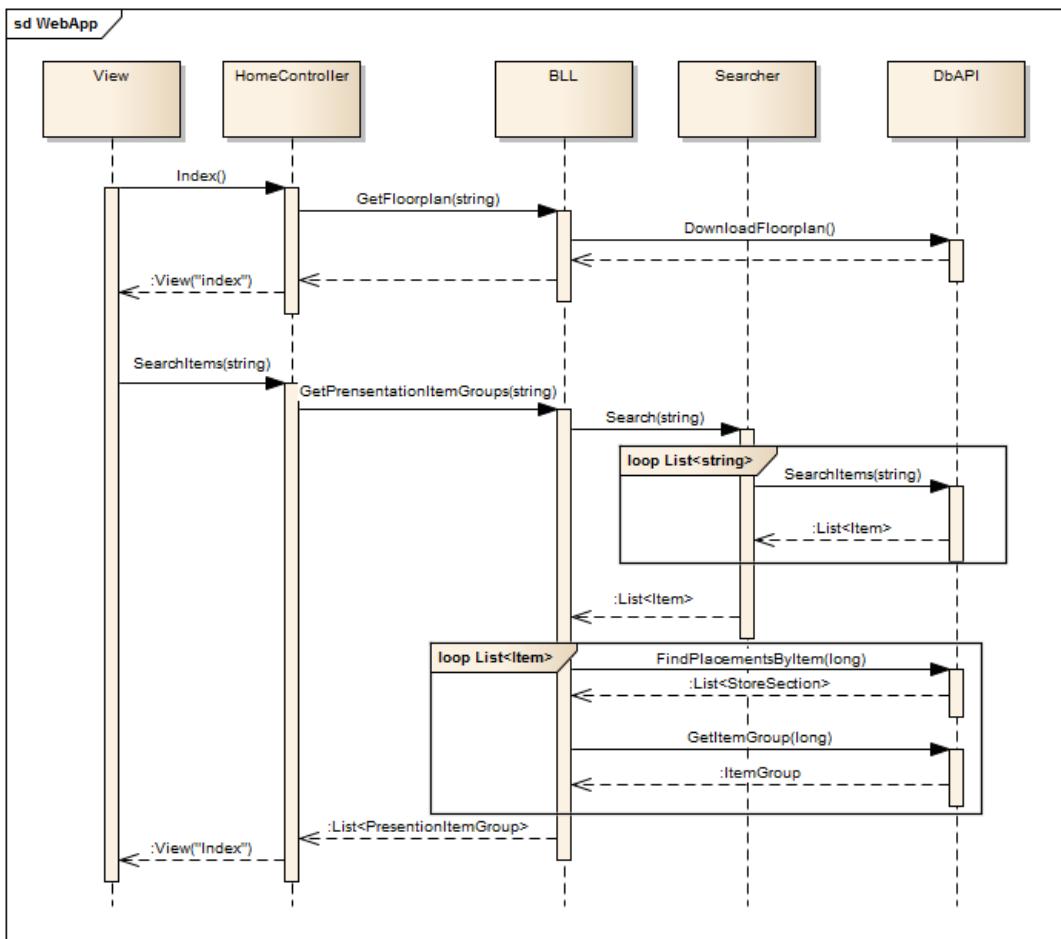
4.4 Web application

Dette afsnit har til formål at dokumentere designet af web applikationen. Web applikationen er designet i ASP.NET MVC. På figur 31 ses de klasser der bliver brugt på server-siden af web applikationen. På diagrammet ses det BLL klassen gør brug af dele af database API. Dette vises ved at database API pakken kun indeholder de klasser der er relevante for web applikationens server side. Dette er gjort for at spare plads på diagrammet og afspejler derfor ikke den egentlige implementering.



Figur 31: Klassediagram af de klasser fra webapplikationen som bliver brugt på server-siden

På figur 32 ses interaktionen mellem klasserne i web applikationen. Diagrammet illustrerer sekvensen af handlinger der sker ved første load af web applikationen og når der søges på en vare.



Figur 32: Sekvensdiagram af webapplikationen

I forbindelse med første load af webapplikationen hentes plantegningen fra databasen til webserveren ved et kald til GetFloorplan funktionen i BLL klassen. Når plantegningen er hentet vil denne kunne vises på brugergrænsefladen. Når en bruger har indtastet en søgetekst og trykket på søgeknappen, sættes en søgning i gang fra HomeController'en. Dette sker ved at der på BLL klassen kaldes GetPresentationItemsGroups. Denne funktion foretager en søgning på butiks databasen ved hjælp af Searcher klassen. Giver denne søgning resultater vil yderligere data omkring varen hentes, herunder dens tilhørende varegruppe og placering. Disse resultater inddeltes efter varegrupper i form af PresentationItems. I HomeController gemmes listen med PresentationItemGroups i Model objektet, hvorefter listen puttes i en viewbag, der kan tilgås fra viewet. Til sidst opdateres View, hvorved PresentationItemGroups-listen i viewbagen bliver loadet som html dokumenter.

4.4.1 Forretningslogik laget

Forretningslogik laget består af modeller for systemets objekter samt BLL og Searcher klassen. De sidstnævnte klasser indeholder metoder til at oprette præsentationsmodeller, som er objekter der repræsenterer varer og varegrupper. Det præcise indhold af disse klasses kan ses på

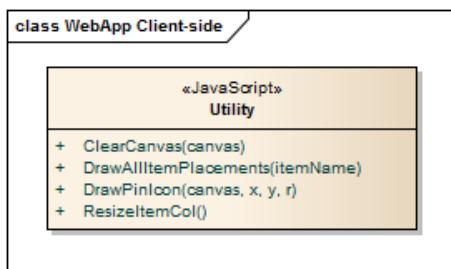
klassediagrammet på figur 31. Følgende tabel beskriver de klasser som indgår forretningslogik laget.

Klassenavn	Beskrivelse
DatabaseService	Database API model Bruges til tilgang af butiks databasen
Item	Database API model OO repræsentation af en vare
ItemGroup	Database API model OO repræsentation af en varegruppe
StoreSection	Database API model OO repræsentation af en sektion
PresentationItem	Præsentations model Udtræk fra butiks databasen bestående af varenavn, relateret varegruppenavn og placering Repræsenterer en vare i søgeresultatet
PresentationItemGroup	Præsentations model En samling af PresentationItems der tilhører samme varegruppe Repræsenterer søgeresultater i én bestemt varegruppe
Model	Præsentations model Indeholder en samling af PresentationItemGroups Repræsenterer alle søgeresultater samlet
BLL	Henter data fra butiks databasen via database API og opretter præsentations modeller
Searcher	Udfører søgning af varer på databasen Foretager operationer på søgeteksten inden søgning, således at mellemrum ekskluderes fra søgningen og der foretages en søgning for hvert ord, hvorefter resultaterne samles til et resultat

Tabel 39: Beskrivelse af klasser der indgår i forretningslogik laget

4.4.2 Præsentationslaget

På figur 33 ses den funktionspakke der bliver brugt på klientsiden af web applikationen. De forskellige funktioner bliver kørt i browseren og aktiveres via interaktioner med browseren. Yderligere beskrivelse af funktionerne ses i tabel 40.

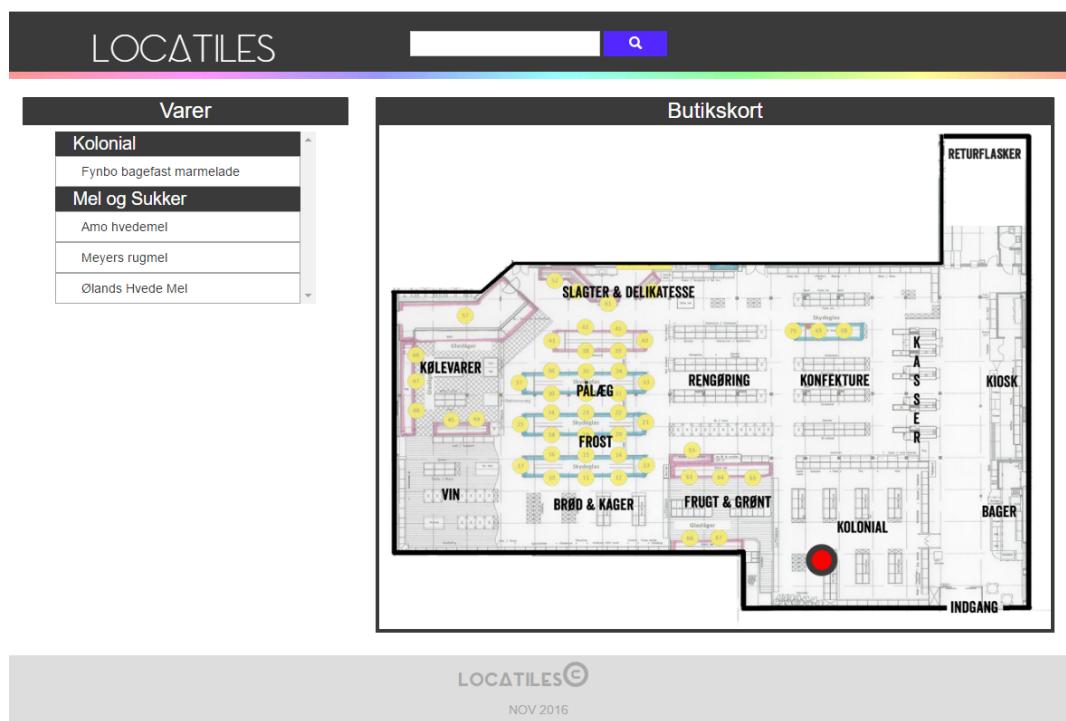


Figur 33: Funktionspakke brugt på klient siden af web applikationen

Funktion	Beskrivelse
ClearCanvas(canvas)	Fjerner eksisterende pin-ikoner fra plantegningen
DrawPinIcon(canvas, x, y, r)	Tegner et pin-ikon på et bestemt canvas ved en bestemt position med en bestemt diameter
DrawAllItemPlacements(itemName)	Henter en vares placeringer og kalder "DrawPinIcon" funktionen for at tegne vareplaceringerne
ResizeItemCol()	Opdaterer varelistens højde til at matche højden af plantegningen for at gøre siden mere mobilvenlig

Tabel 40: Beskriver javascript-funktionerne

Figur 34 viser web applikationen set fra en desktopskærm. Siden er opbygget af en række elementer der bruges til brugerinteraktion og visning af relevant information.



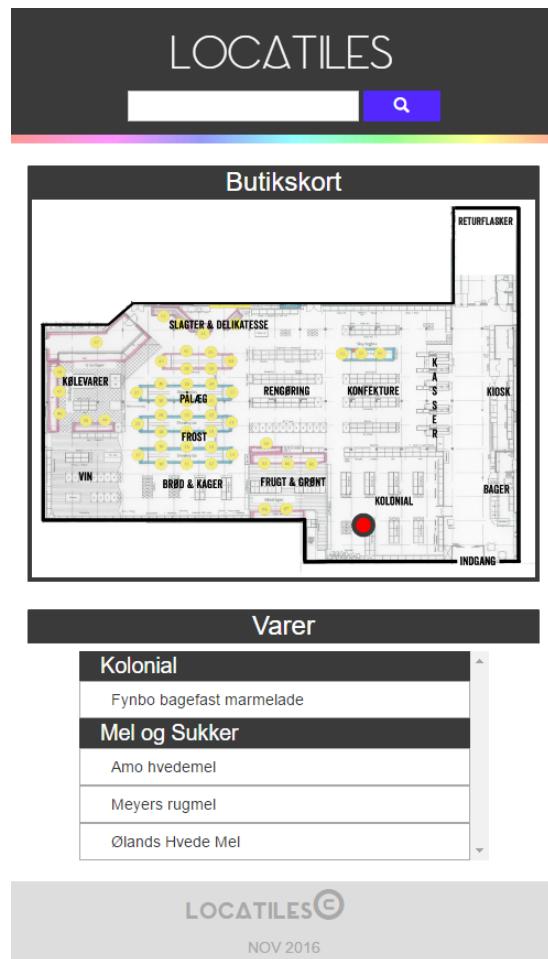
Figur 34: Web applikationen set på en skærm med desktop-størrelse

I tabel 41 ses beskrivelser omkring de elementer som indgår i applikationen.

Element	Tag	Beskrivelse
Søgefelt	Input	Bruges af kunden til at skrive søgestrenget med de varenavne kunden vil søge efter.
Søgeknap	Button	Aktiverer søgningen ved at sende søgestrenget til "SearchItems".
Vareliste	Div	Indeholder information fra alle "PresentationItemGroups" som sendes til viewet efter en søgning.
Vareknapper	Button	Aktiverer "DrawPinIcon"-funktionen som tegner de tilhørende placeringer af den pågældende vare på kortet.
Plantegning	Canvas	Viser plantegningen af butikken. Bruges også til at tegne pin-ikoner på.

Tabel 41: Beskriver elementernes type

Webapplikationen er designet med bootstrap for at opnå et responsivt design. På figur 35 ses web applikationen skaleret til skærmstørrelsen på en smartphone. Dette medfører at varelisten bliver flyttet til at være placeret under plantegningen.



Figur 35: Web applikationen set på en skærm med smartphone-størrelse

5 Test

I følgende afsnit beskrives de testtyper og metoder der er anvendt for at verificere at produktet overholde kravene sat i kravspecifikationen, afsnit 2, s. 7. Herunder beskrives også de frameworks anvendt under testforløbet.

5.1 Anvendte test frameworks

Under kvalitetssikringen og test af produktet er der anvendt frameworks til at automatisere unit tests samt integrationstest. Følgende afsnit beskriver de to frameworks der er blevet brugt under testforløbet, som blev valgt på baggrund af at de understøtter test af sourcekode skrevet på .NET platformen samt at de er blevet introduceret i kurset I4SWT.

5.1.1 NUnit

Til opsætning af tests er der blevet gjort brug af NUnit frameworket [15]. NUnit gør det nemt at implementere automatiserede unit tests i C#. Yderligere indeholder det også en test runner, så alle tests i en solution kan blive eksekveret og resultaterne vist.

5.1.2 NSubstitute

For at kunne verificere adfærd og tilstande efter en bestemt handling var foretaget er der blevet gjort brug af NSubstitute [16]. Med dette isolationsframework er det muligt at se på et objekts tilstand efter en handling og verificere resultatet. Frameworket gjorde det muligt at fjerne eksterne afhængigheder under testene for at sikre at testene blev udført under en kendt tilstand.

NSubstitute blev brugt til unit tests og integrationstests i løbet af projektet, for at have fuldt kontrol over “units under test” og “integration under test.”

5.2 Visuel test

Database API er ikke blevet testet via en automatiseret unit test, da denne er tæt knyttet til databasen. Dermed er der ikke mulighed for at isolere modulerne fra databasen og de er derfor testet visuelt.

5.3 Unit test

For at kunne lave integrationstest af produktets delsystemer, er der blevet foretaget unit tests af de moduler som udgør disse. Der er blevet foretaget unit test af modulerne som indgår i desktop applikationen og web applikationen.

5.3.1 Desktop applikation

For desktop applikationen er der foretaget unit test af viewmodellerne. Da disse er separeret fra brugergrænsefladen, er det blevet muligt at teste klasserne isoleret. I testene undersøges applikationens opførelse når kommandoer kaldes.

Unit under test	ItemGroup-ViewModel	ItemViewModel	Floorplan-Viewmodel	StoreSection-Viewmodel
Status	Gennemført	Gennemført	Gennemført	Ikke gennemført

Tabel 42: Status af unittest for desktop applikation

Under testforløbet er tre ud af de fire viewmodeller til butikadministratorens brugergrænseflade blevet testet. Dette skyldes at StoreSection viewmodellen indeholdte for mange WPF elementer til at denne kunne testes i isolation på højde med de andre viewmodeller.

5.3.2 Webapplikation

Alle de klasser i webapplikation, der indeholder adfærd er blevet unittestet, da risikoen for fejl forsaget af dem er langt større end for klasser uden adfærd. I tabel 43 ses det hvilke klasser der er testet.

Unit under test	HomeController	BLL	Searcher
Status	Gennemført	Gennemført	Gennemført

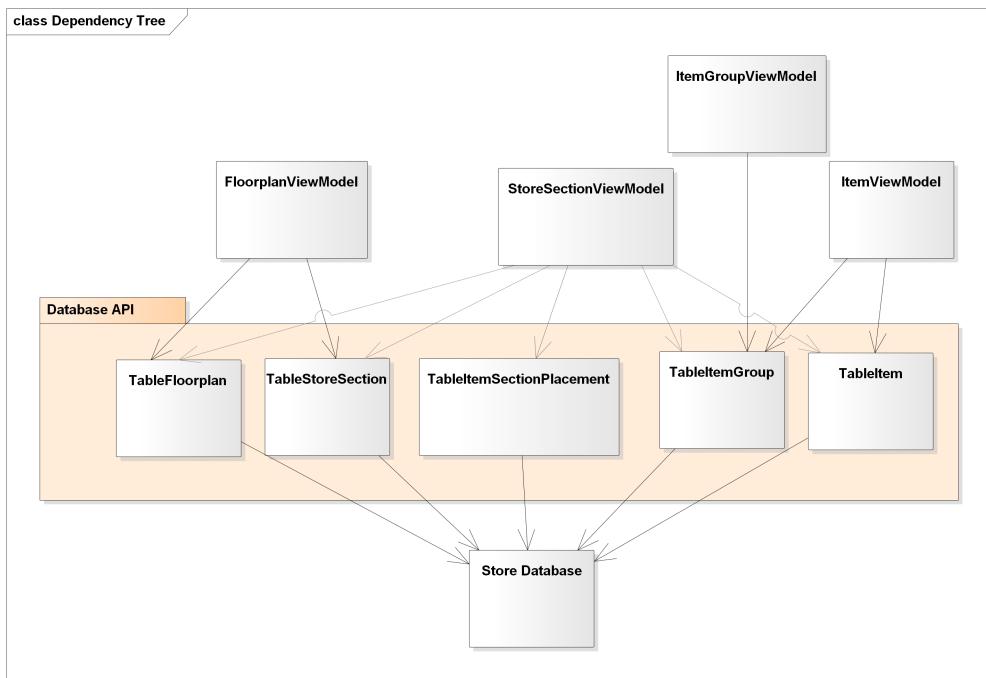
Tabel 43: Status af unittest for web applikation

5.4 Integrationstest

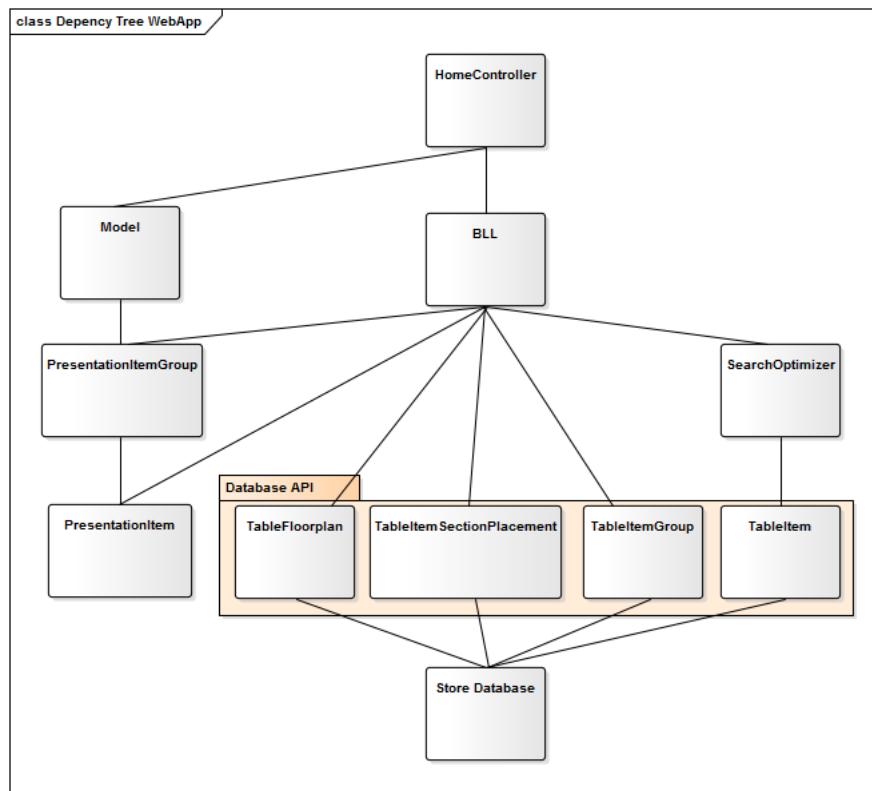
For at teste interaktionerne mellem modulerne som udgør systemet, er der blevet foretaget integrationstests. Test strategien for integrationstesten er bottom-up metoden [17]. For hver applikation er der lavet et afhængighedstræ som visualiserer afhængighederne mellem de enheder som testes. Ud fra afhængighedstræene og den valgte teststrategi er der defineret en række steps til udførelse af integrationstesten.

Integrationstesten er opdelt i tre faser, hvor den første tester database API og de sidste hver især tester desktop applikationen samt webapplikationen.

I første fase testes interaktionerne mellem butiks databasen og hver klasse i database API'en. Herefter testes der i anden fase interaktionen mellem butiks databasen og viewmodellerne via database API. Trejde fase består af test af interaktionerne mellem butiks databasen og HomeController via database API. På figur 36 ses et afhængighedstræ, som visualiserer afhængighederne mellem klasserne for desktop applikationen og på figur 37 ses afhængighederne for web applikationen. Udfra disse figurer er der defineret nogle trin til udførelse af integrationstesten, for hver af de tre faser.



Figur 36: Afhængighedstræ for første og anden fase



Figur 37: Afhængighedstræ for tredje fase

Følgende tabeller viser hvilke klasser der indgår i det pågældende step af integrationstesten. D angiver at den markerede klasse driver testen og X angiver at den markerede klasse indgår i testen.

Klasse \ Step nr.	1	2	3	4	5
TableItemSectionPlacement					D
TableFloorplan				D	
TableStoreSection			D		
TableItemGroup		D			
TableItem	D				
StoreDatabase	X	X	X	X	X

Tabel 44: Første fase af integrationstesten

Som det fremgår af figur 36 er StoreSection viewmodelen afhængig af samtlige klasser i database API'en, men på grund af den tætte kobling til WPF elementer er denne klasse dømt til at være ikke testbar. Dermed indgår klassen ikke i anden fase af integrations testen, som vist i tabel 45.

Klasse	Step nr.	6	7
ItemGroupViewModel		D	
ItemViewModel	D		
TableItemGroup	X	X	
TableItem	X		
StoreDatabase	X	X	

Tabel 45: Anden fase af integrationstesten

I tabel 46 ses testfaserne for tredje fase af integrationstesten. Klasser som er tilskrevet S er substituerede i det pågældende step.

Klasse	Step nr.	8	9	10	11
HomeController				D	
BLL			D	D	X
Searcher	D	S	X	X	
Model					X
PresentationItem				X	X
PresentationItemGroup				X	X
TableFloorplan			X	X	X
TableItemSectionPlacement				X	X
TableItemGroup				X	X
TableItem	X		X	X	

Tabel 46: Tredje fase af integrationstesten

6 Accepttest

I dette afsnit vil der for hver implementeret user story blive specificerede en accepttest, som skal sikre at kravene stillet i kravspecifikationen, afsnit 2, s. 7, er overholdt af systemet.

6.1 Desktop applikation

6.1.1 Opret vare

Brugerscenarie: Når butiksadministratoren vil oprette en vare i butikkens database, navigerer han til menupunktet Administrer varer. Herfra tilgår han undermenuen for oprettelse af varer og indtaster varens navn og vælger en varegruppe. Derefter klikker han på knappen “Opret Vare” for at tilføje varen til butikkens database.

Prækondition: Der skal være oprettet en varekategori ved navn “Fersk Kød”. Der findes en forbindelse til butikkens database. Brugeren er i hovedmenuen af desktopapplikationen.

Postkondition: En vare ved navn “Rieders Pølse” er blevet oprettet i databasen.

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Navigér til menupunktet Administrer Varer	Vinduet for administrering af varer er vist på brugergrænsefladen	Adminster Varer vinduet vises	OK
2	Klik på Opret Vare knappen	Dialogen for oprettelse af varer er vist på brugergrænsefladen	Opret Vare dialog vises	OK
3	Indtast varenavnet Rieders pølse i feltet for varenavn	Der står Rieders pølse i feltet for varenavn		OK
4	Tildel varen varegruppen Fersk Kød fra dropdown menuen	Varegruppe feltet viser Fersk Kød		OK
5	Klik på knappen der tilføjer varen til databasen	Brugergrænsefladen viser en bekræftelse på at varen er oprettet i databasen	Pop up bekræfter oprettelsen med vare navn	OK

Tabel 47: Accepttest for butikadministor opret vare

6.1.2 Opret vare uden valgt varekategori

Brugerscenarie: Når butiksadministratoren vil oprette en vare i butikkens database, navigerer han til menupunktet hvor varer administreres. Herfra tilgår han undermenuen for oprettelse af varer, og indtaster varens navn, og undlader at vælge en varegruppe. Derefter klikker han på en knap til at tilføje en vare.

Prækondition: Brugeren er i hovedmenuen af desktopapplikationen. Der findes en forbindelse til butikkens database

Postkondition: Varen er ikke blevet tilføjet til butikkens database

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Navigér til menupunktet Administrer Varer	Administrer Varer vinduet er vist på brugergrænsefladen	Administrer Varer vinduet vises	OK
2	Klik på Opret Vare knappen	Opret Vare dialogen er vist på brugergrænsefladen	Opret Vare dialog vises	OK
3	Indtast varenavnet Rieders pølse i Varenavn feltet	Der står Rieders pølse i Varenavn feltet		OK
4	Undlad at vælge en varegruppe fra dropdown menuen	Knappen for at oprette en vare er deaktiveret	Knappen er deaktiveret	OK

Tabel 48: Accepttest for opret vare, hvor der ikke er valgt en varekategori

6.1.3 Slet vare

Brugerscenarie: Når butiksadministratoren vil slette en vare fra butikkens database, navigerer han til menuen “Administrer Varer”. Herfra klikker han på den vare i listen han ønsker at slette. Når varen er markeret, klikker han på knappen “Slet Vare”.

Prækondition: Varen “Rieders Pølse” er oprettet i butikkens database. Brugeren er i hovedmenuen for desktopapplikationen, der er forbindelse til databasen.

Postkondition: Varen “Rieders Pølse” er blevet slettet

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Navigér til menupunktet Administrer Varer	Administrer Varer vinduet er vist på brugergrænsefladen	Administrer Varer vinduet er vist	OK
2	Klik på varen Rieders Pølse i listen over varer	Varen Rieders Pølse er markeret	Der blev søgt på varen Rieders Pølse, og denne blev er markeret	OK
3	Klik på knappen Slet Vare	Brugergrænsefladen udskriver at Rieders Pølse er slettet fra databasen	Der blev vist en dialogboks som meddelte at varen Rieders Pølse blev slette fra databasen	OK

Tabel 49: Accepttest for butiksadministrator slet varer

6.1.4 Indsæt plantegning til butikken

Brugerscenarie Når butiksadministratoren vil indsætte en plantegning i butikken, navigerer han til menuen “Administrer Plantegninger”. Derfra klikker han på knappen “Gennemse”, der åbner en dialogboks, hvorfra han kan finde den ønskede plantegning. Når den ønskede plantegning er valgt, klikker han på knappen “Upload”.

Prækondition: Brugeren er i hovedmenuen på desktopapplikationen, og der findes en forbindelse til databasen. Billedet “floorplanTest.png” findes på computeren, hvor desktop-applikationen bliver kørt.

Postkondition: Billedet floorplanTest.png er blevet indsæt i butikken.

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Naviger til menupunktet Administrer Plantegninger	Administrer Plantegninger vinduet er vist på brugergrænsefladen	Administer Plantegninger vinduet er vist	OK
2	Klik på knappen Gennemse	Windows stifinder på computeren er vist	Stifinder åbnes	OK
3	Find filen floorplanTest.png	Filen floorplanTest.png er markeret		OK
4	Klik Open/Åben knappen	Windows stifinder lukkes og stien til filen ses i tekstboksen over Gennemse knappen	Stien vises i tekstboksen	OK
5	Klik på knappen Upload	Plantegning på vinduet er blevet opdateret til det nye billede	Plantegningen opdateres	OK

Tabel 50: Accepttest for butiksadministrator indsættelse af plantegning

6.1.5 Tilføj sektion til butikkens plantegning

Brugerscenarie: Når butiksadministratoren vil indsætte sektioner på plantegningen, navigerer han til menuen “Administrer Sektioner”. Herfra klikker han et sted på plantegningen. Derefter viser brugergrænsefladen en dialogboks, hvor butiksadministratoren kan give sektionen et navn. Når dette er gjort, klikker han på “Opret Sektion”. Derefter er sektionen vist som en rød prik på plantegningen.

Prækondition: Brugeren er i hovedmenuen, og der findes en forbindelse til databasen. Derudover findes der en plantegning i databasen.

Postkondition: Der er oprettet en sektion kaldet “TestSektion” på plantegningen

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Naviger til menupunktet Administrer Sektioner	Administrer Sektioner vinduet er vist på brugergrænsefladen	Administrer Sektioner vinduet er vist	OK
2	Klik et vilkårligt sted på butikkens plantegning	Et rødt punkt markerer stedet hvor der blev klikket og dialogen for Opret Sektion	Opret Sektion dialogen vises	OK
3	Indtast TestSektion i feltet for sektionnavn og klik Opret Sektion	Sektionen TestSektion er blevet oprettet	Efter at trykke på det røde punkt ses at den valgte sektion er TestSektion	OK

Tabel 51: Accepttest for indsættelse af sektioner på plantegning

6.1.6 Rediger sektion

Brugerscenarie: Når butiksadministratoren vil redigere en sektion, navigerer han til menuen “Administrer Sektioner”. Herfra klikker han på den sektion han ønsker at redigere. Når sektionen er markeret, klikker han på “Rediger Sektion”. Derefter viser brugergrænsefladen en dialogboks, hvor butiksadministratoren kan skrive det nye navn på sektionen. Når dette er gjort, klikker han på “Rediger Sektionnavn”.

Prækondition: Brugeren er i hovedmenuen, og der findes en forbindelse til databasen. Derudover eksisterer der en sektion der hedder “TestSektion”.

Postkondition: Sektionen “TestSektion” er blevet omddøbt til “tSektion”.

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Naviger til menupunktet Administrer Sektioner	Administrer Sektioner vinduet er vist på brugergrænsefladen	Administrer Sektioner vinduet er vist	OK
2	Klik på sektionen med navnet TestSektion	Sektionen TestSektion er den valgte sektion	TestSektion er valgt	OK
3	Klik på Rediger Sektion	Dialogen for Rediger Sektion er vist på brugergrænsefladen	Rediger Sektion dialog vises	OK
4	Indtast tSektion i feltet Nyt Sektionnavn	Der står tSektion i feltet Nyt Sektionnavn		OK
5	Klik på Rediger Sektionnavn	Sektionen er omdøbt til tSektion	Efter at trykke på det røde punkt ses at den valgte sektion er tSektion	OK

Tabel 52: Accepttest for redigering af sektioner

6.1.7 Slet sektion

Brugerscenarie: Når butiksadministratoren vil slette en sektion, navigerer han til menuen ”Administrer Sektioner”. Herfra klikker han på den sektion han ønsker at slette. Når sektionen er markeret, klikker han på ”Slet Sektion”, hvorefter sektionen bliver slettet fra plantegningen.

Prækondition: Brugeren er i hovedmenuen, og der findes en forbindelse til databasen. Derudover er der oprettet en sektion, tSektion.

Postkondition: Den valgte sektion er blevet slettet fra databasen.

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Naviger til menupunktet Administrer Sektioner	Administrer Sektioner vinduet er vist på brugergrænsefladen	Administrer Sektioner vinduet er vist	OK
2	Klik på tSektion sektionen	Sektionen tSektion er vist som den valgte sektion		OK
3	Klik på Slet Sektion	Den markerede sektion er nu slettet	Det røde punkt er fjernet fra plantegningen	OK

Tabel 53: Accepttest for fjernelse af sektioner

6.1.8 Tilføj vare til sektion

Brugerscenarie: Når butiksadministratoren vil tilføje varer til en sektion, navigerer han til menuen ”Administrer Sektioner”. Derefter vælger han sektionen han vil tilføje varer til, ved at klikke på sektionen. Når sektionen er valgt, markerer han de varer han vil tilføje til sektionen i varelisten. Når alle varerne er valgt, klikker han på ”V”, hvorefter varerne er tilføjet til sektionen

Prækondition: Brugeren er i hovedmenuen, og der findes en forbindelse til databasen. Derudover findes varerne ”test1”, ”test2”, ”test3” i databasen. Sektionen ”tSektion” skal også findes på databasen

Postkondition: Varerne ”test1”, ”test2” og ”test3” er blevet tilføjet til sektionen ”tSektion”

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Naviger til menupunktet Administrer Sektioner	Administrer Sektioner vinduet er vist på brugergrænsefladen	Administrer Sektioner vinduet er vist	OK
2	Klik på sektionen med navnet tSektion	Sektionen tSektion er den valgte sektion		OK
3	Vælg varerne test1, test2 og test3 i varelisten	Varerne test1, test2 og test3 er markeret i varelisten	Søgefeltet bruges for at finde varerne og disse markeres	OK
4	Klik på knappen V	Varerne test1, test2 og test3 er blevet tilføjet til varelisten for sektionen tSektion	Varene er tilføjet til sektionen og vises når sektionen markeres	OK

Tabel 54: Accepttest for tilføjelse af varer til en sektion

6.1.9 Slet vase fra sektion

Brugerscenarie: Når butiksadministratoren vil fjerne en vase fra en sektion, navigerer han til menuen “Administrer Sektioner”. Derfra vælger han den sektion han vil slette varerne fra. Når sektionen er valgt, markerer han den vase han vil slette fra sektionens vareliste. Når denne er valgt klikker han på knappen “Fjern vase fra sektion”.

Prækondition: Brugeren befinner sig i hovedmenuen, og der findes en forbindelse til databasen. Derudover findes der en sektion med navnet “tSektion” der har varen “test1”.

Postkondition: Varerne “test1” er blevet slettet fra sektionen “tSektion”.

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Naviger til menupunktet Administrer Sektioner	Administrer Sektioner vinduet er vist på brugergrænsefladen	Administrer Sektioner vinduet er vist	OK
2	Klik på sektionen med navnet tSektion	Sektionen tSektion er markeret	tSektion er den valgte sektion	OK
3	Vælg vase test1 i sektionens vareliste	Varen test1 er markeret		OK
4	Klik på knappen Fjern vase fra sektion	Varen test1 er slettet fra sektionens vareliste	Varen er fjernet fra tSektions vareliste	OK

Tabel 55: Accepttest for fjernelse af en vase fra en sektion

6.2 Web applikation

6.2.1 Vis vase på butikkens plantegning

Prækondition: Varen “test vase” eksisterer i butikkens database og brugeren befinner sig i web applikationen.

Postkondition: Varen “test vase” er blevet vist på butikkens plantegning.

Step	Handling	Forventet Resultat	Observeret Resultat	Vurdering
1	Tryk på søgerfeltet for varer	Søgefeltet er aktivt		
2	Indtast "test vare" i søgerfeltet	"test vare" er indtastet i søgerfeltet		
3	Tryk på søgeknappen	En liste af fundne varer er vist		
4	Tryk på varen "test vare" i listen af fundne varer	Varens lokation vises på butikkens plantegning		

Tabel 56: Accepttest for søgning af vare igennem web applikationen

6.3 Ikke funktionelle krav

Krav: Desktop applikationens størrelse må ikke overstige 200 MB

Step	Handling	Forventet resultat	Observeret resultat	Vurdering
1	Navigér til applikationens eksekverbare fil i Windows stifinder			OK
2	Inspicér filens størrelse	Filens størrelse er mindre end 200MB	Desktop applikationen fylder 174 kB	OK

Tabel 57: Accepttest resultat for kravet; desktop applikationens størrelse må ikke overstige 200MB

Krav: Desktop applikationen skal kunne acceptere plantegninger af filtypen .jpg

Step	Handling	Forventet resultat	Observeret resultat	Vurdering
1	Åben desktop applikationen	Desktop applikationen er åben	Desktop applikationen er åben	OK
2	Navigér til menupunktet Administrerer Plantegninger	Menuen for administrering af plantegninger er vist	Det forventede menu punkt er vist	OK
3	Klik på knappen Gennemse	En windows stifinder dialog boks er vist på skærmen	Dialogboksen er vist	OK
4	Vælg en vilkårlig .jpg fil og klik på knappen Åben	Stien til den valgte fil er vist i tekstboksen over Gennemse-knappen	Ok	OK
5	Klik på knappen Upload	Billedet i menuen ændrer sig til det nyligt uploadedede billede	Billedet ændrede sig til det nyligt valgte .jpg billede	OK

Tabel 58: Accepttest resultat for kravet; desktop applikationen skal kunne acceptere plantegninger af filtypen .jpg

Krav: Desktop applikationen skal kunne afvikles på Windows 10

Step	Handling	Forventet resultat	Observeret resultat	Vurdering
1	Navigér til desktop applikationens eksekverbare fil på en Windows 10 maskine			OK
2	Dobbeltklik på den eksekverbare fil	Locatiles desktop applikation åbner	Applikationen åbnede som forventet	OK

Tabel 59: Accepttest resultat for det ikke funktionelle krav; Desktop applikationen skal kunne afvikles på Windows 10

Litteratur

- [1] Dan Radigan. Epics, stories, versions, and sprints. <https://www.atlassian.com/agile/delivery-vehicles>. Tilgået: 01-12-2016.
- [2] DSDM Consortium. Moscow prioritisation. <https://www.dsdm.org/content/moscow-prioritisation>. Tilgået: 21-11-2016.
- [3] Microsoft. .net documentation. <https://docs.microsoft.com/en-us/dotnet/>. Tilgået: 17-11-2016.
- [4] MSDN. Windows presentation foundation. [https://msdn.microsoft.com/en-us/library/ms754130\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx). Tilgået: 17-11-2016.
- [5] Microsoft. Sql server 2012. <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-v12-whats-new>. Tilgået: 17-11-2016.
- [6] Microsoft. Microsoft azure. <https://azure.microsoft.com/en-us/>. Tilgået: 17-11-2016.
- [7] MSDN. Ado.net. [https://msdn.microsoft.com/en-us/library/e80y5yhx\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/e80y5yhx(v=vs.110).aspx). Tilgået: 17-11-2016.
- [8] Microsoft. Asp.net. <https://www.asp.net/>. Tilgået: 17-11-2016.
- [9] Bootstrap. Bootstrap. <http://getbootstrap.com/>. Tilgået: 08-12-2016.
- [10] jQuery. jquery. <https://jquery.com/>. Tilgået: 08-12-2016.
- [11] dofactory. Abstract factory. <http://www.dofactory.com/net/abstract-factory-design-pattern>. Tilgået: 16-11-2016.
- [12] MSDN. Repository pattern. <https://msdn.microsoft.com/en-us/library/ff649690.aspx>. Tilgået: 16-11-2016.
- [13] Robert C. Martin. Solid. [https://en.wikipedia.org/wiki/SOLID_\(object-oriented_design\)](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)). Tilgået: 28-11-2016.
- [14] MVVM Light Toolkit. RelayCommand class. <http://www.mvvmlight.net/help/SL5/html/e5294f54-f460-8227-b228-ac63034b1294.htm>. Tilgået: 29-11-2016.
- [15] NUnit.org. Nunit. <https://www.nunit.org/>. Tilgået: 17-11-2016.
- [16] NSubstitute. Nsubstitute. <http://nsubstitute.github.io/>. Tilgået: 17-11-2016.
- [17] Robert V. Binder. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. Addison-Wesley, 1999.