Technical University of Denmark

Rasmus Raakjær Vind

Robust software i et ugunstigt miljø

Lagerstyrings- og bestillingssystem til festivallen Musik i Lejet

Bachelorprojekt, februar til maj 2016

DTU Compute
Institut for Matematik og Computer Science

[This page intentionally left blank]

Robust software i et ugunstigt miljø

Lagerstyrings- og bestillingssystem til festivallen "Musik i Lejet"

Forfatter:

Rasmus Raakjær Vind – rasmus@raakjaer.dk

Supervisor:

Christian W. Probst – cwpr@dtu.dk

DTU Compute

Richard Petersens Plads

Bygning 324

2800 Kgs. Lyngby Denmark

www.compute.dtu.dk

Tel: (+45) 45 25 30 31

E-mail: compute@compute.dtu.dk

Udgivelses dato: 31-05-2016

Klasse: 1 (Public)

Udgave: 1. edition

Kommentar: Denne rapport er en del af uddannelsen til Diplomingeniør på Danmarks Tekniske Universitet.

Denne rapport repræsenterer 20 ECTS points.

FORORD

Denne rapport er en del af uddannelsen til Diplomingeniør IT på Danmarks Tekniske Universitet.
Projektet har strakt sig fra 1. februar 2016 til 31. maj 2016, dog har ideen til programmet været under udvikling siden december 2015, og er ligeledes blevet beskrevet i en rapport, jeg har udarbejdet ved DTU Compute med titlen ”Analyse af robuste systemer”.
Projektet er lavet i et samarbejde med non-profit festivallen Musik i Lejet, hvor produktet skal hjælpe dem med at styre lageret under opbygningen af festivallen.
Målet med denne rapport er, at give læseren en forståelse for de beslutninger, der er blevet taget, i forbindelse med programmets udvikling.

Et vist kendskab til programmering er en forudsætning for, at læseren kan få den fulde forståelse af rapporten. En person, som er uddannet diplom-IT på DTU vil som udgangspunkt have de nødvendige forudsætninger.

Jeg ville have haft svært ved at gennemføre dette projekt uden det dejlige netværk af personer, der på den ene eller anden måde har hjulpet mig igennem min tid på DTU.

Jeg vil gerne takke:
Christian W. Nielsen (diplom-IT Ingeniør) for at være en uundværlig støtte.
Min familie for at hjælpe til med at holde mig oven vande.
Mennesker med forbindelse til Scenelys, der har hjulpet med alt, fra en joke til at holde mig kørende i de sværeste situationer.

Rasmus Raakjær Vind
DTU – s100247

ABSTRACT

The music festival “Musik i Lejet” has always had problems with lost equipment during the period where the festival grounds is established.
The festival need a simple and robust program to handle the building material and equipment stockpile.

The festival grounds are located next to the northern beach of Zealand. This presents a couple of interesting limitations. There is no internet or mobile signal available.
All power come from generators. Water, sand and sea salt gets in everywhere.

The idea is to develop a program to run on a Windows based laptop. It is to be very robust against errors both by the code and the user. Periodic backup of the stored data is required and a quick method to set a restore from a backup to another computer in case of catastrophic hardware failure.

The administrator password for the prototype is “Hest”.

The 1.0 version of the program and the source code can be acquired by:

For a limited period of time (June 2016) by download link:

Source Code in a VS2015 Solution:

https://www.dropbox.com/s/jexofwsefk0d1cf/Quick_Codex_SouceCode_VS2015_Solution.zip?dl=0

WinForms v1.0:

https://www.dropbox.com/s/1564s9rcvtl587l/WF.v1.ReleaseBuild.zip?dl=0

WPF v1.0:

https://www.dropbox.com/s/3q6vsm6mz6lqt5o/WPF.v1.ReleaseBuild.zip?dl=0

Access to the Git repository:

User: read_access

email: yd6dkj5jf6@gmail.com

password: DTUafgangsprojekt

(https) git clone https://read_access@bitbucket.org/private100247/repo.git

(SSH) git@bitbucket.org:private100247/repo.git

The program require Microsoft SQL Server 2014 Express LocalDB to run.

Download at: https://www.microsoft.com/en-us/download/details.aspx?id=42299

RESUMÉ

Til materiale- og værktøjslageret på Musik I Lejet (MiL) ønskes et system som kan hjælpe med at holde styr på lagerbeholdning, udlån af værktøj samt generere bestillingslister på manglende materialer.

Nogle af festivallens udfordringer kommer af dennes lokation samt meget midlertidige arbejdsområder. MiL ligger på en parkeringsplads 5 meter fra havet og ca. 1 km fra nærmeste nabo. Der er intet fast internet eller mobil signal. Strøm levereres fra en generator, der jo kan være noget ustabil.

Med input og ønsker fra de ansvarlige på MiL, er der i samarbejde med dem udarbejdet en kravspecifikation, der ligger til grundlag for at skitsere et system, der skal kunne køre under disse specifikke forhold og gøre arbejdet med lagerstyring lettere.

Det planlagte system skal afvikles på en Windows baseret bærbar computer med minimum Windows 7. Det krævede hardwaretilbehør er en UPS enhed (til backupstrøm), en stregkode læser og en plastikkort printer (Standard id-kort plastik).

Softwaren skal designes så robust, at en fejl ikke lukker systemet. Der gennemføres periodiske backup af programmets data på computerens HHD med mulighed for backup på en evt. ekstern hukommelse. Det skal sikres, at en strømafbrydelse eller hardwarefejl - ikke betyder at data bliver korrupt, og at der hurtigt kan sættes et nyt system op fra en backup.

Administrator passwordet til denne prototype er ”Hest”.

Kildekode og version 1.0 af denne prototype kan findes ved:

I Juni 2016 via download link:

Source Code I en VS2015 Solution:

https://www.dropbox.com/s/jexofwsefk0d1cf/Quick_Codex_SouceCode_VS2015_Solution.zip?dl=0

WinForms v1.0:

https://www.dropbox.com/s/1564s9rcvtl587l/WF.v1.ReleaseBuild.zip?dl=0

WPF v1.0:

https://www.dropbox.com/s/3q6vsm6mz6lqt5o/WPF.v1.ReleaseBuild.zip?dl=0

Adgang til Git repository:

User: read_access

email: yd6dkj5jf6@gmail.com

password: DTUafgangsprojekt

(https) git clone https://read_access@bitbucket.org/private100247/repo.git

(SSH) git@bitbucket.org:private100247/repo.git

Programmet kræver Microsoft SQL Server 2014 Express LocalDB

Download: https://www.microsoft.com/en-us/download/details.aspx?id=42299

INDHOLDSFORTEGNELSE

1

INTRODUKTION

1.1 Om Musik i Lejet

Musik i Lejet (MiL) blev første gang afholdt i 2009 af en lille gruppe lokale beboere, der ville holde en tre dages strandfest. Herfra voksede det, drevet af frivillige kræfter, og er nu en non-profit 3 dages festival med mere end 10.000 betalende gæster og over 1.500 frivillige, der arbejder lige fra 2x8 timer til løbende over hele året. [1]

Den tekniske erfaring i forbindelse med gennemførelse af en festival er ret begrænset i MiL, da festivallen gennemføres på basis af næsten 100 % frivillige kræfter samt festivallens "unge" alder.
Det har været svært for ledelsen at styre opbygningen og nedtagningen af den meget ambitiøse festival plads.

Opbygningen af festivalpladsen starter ca. 14 dage før festivallen åbner og vokser støt. De sidste 7 dage inden festivallen er der over 200 frivillige på pladsen hver dag.

[1] www.musikilejet.dk

Et større areal lige ved stranden i Tisvilde ved stranden, (Fig.1) ca. 1600 m^2, bliver omdannet til en stor byggeplads. Dette stiller år efter år større krav til arrangørerne med hensyn til at uddelegere og automatisere flere opgaver.

Figur 1 Festivalpladsen – Google Earth (56°03'21.7"N 12°03'31.5"E)

1.2 Baggrund

Der bliver brugt mange ressourcer på indkøb og leje af såvel værktøj som store mængder byggematerialer. Desværre oplever festivallen, der drives af frivillig uden erfaring fra en stor byggeplads, at materialer og værktøj forsvinder.

Festivallen bruger derfor mange ressourcer på at holde styr på lageret, det man låner ud til frivillige, tidspunkt for leverancer/afhentninger med mere; alt sammen logistik som foregår på en stor, og meget kortlivet byggeplads.

Festivalens lokation giver en del problemer i forbindelse med at drive et lagersystem. Manglen på internet og stabil strøm er hovedproblemer for et sådant system. Dog spiller vejret også en rolle, da vind, regn, salt og sand har vist sig at være problematiske for de løsninger festivallen har brugt tidligere.

Lagersystemet skal kunne klare at:

- Blive anvendt af forskellige brugere, frivillige arbejdere med meget forskellige baggrunde.
- Folk laver fejl pga. et stærkt tidspres.
- Det i perioder vil blive brugt næsten konstant.

1.3 Motivation

I 2015 brugte festivallen en logbog til at holde styr på udlånte ting, og der var ingen kontrol med byggematerialer. Logbogen blev meget hurtigt uoverskuelig pga. mange forskellige brugere og senere blev en del af den ulæselig pga. regn. Dette resulterede i, at el-værktøj og byggematerialer for over 20.000 kr. forsvandt.

Tidligere år i 2014 og 2013 havde man tildelt alle bygningshold deres egen værktøjskasse og materialelager. Dette førte til større materialebestillinger end samlet set nødvendigt og et stort materialeoverskud, da festivallen var færdigbygget.

Lagerstyringen på MiL har indtil nu været såvel tids- som pladskrævende uden den har formået at reducere i nævneværdig grad i svindet af materialer og værktøj.

Efter selv at have været frivillig på MiL i 2014 og derefter en del af arrangørgruppen i 2015 og 2016 som ansvarlig for al strøm på festivallen, tilbød festivallen mig at, sideløbende med mit virke for dem, at skrive mit bachelorprojekt i samarbejde med dem.

Deres ønske var et lagerstyrings- og bestillingssystem som kunne hjælpe dem med at få mere styr på festivallens materialeforbrug og forhåbentlig skære nogle af udgifterne til forsvundet værktøj ned.

1.4 Problemformulering

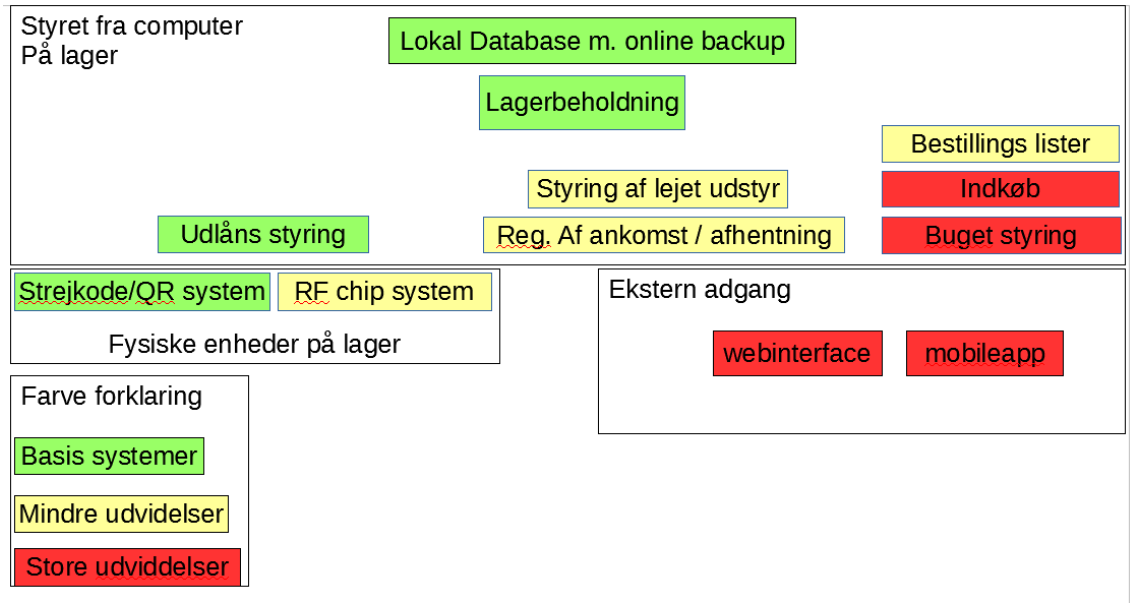
MiL har længe ledt efter en mere effektiv løsning til at styre lager og udlån, derfor har de tilbudt at finansiere hardware til dette program.

MiL ønsker et program til primært at styre udlån af værktøj under opbygningen af festivallen. Programmet skal kunne registrere, når en person låner noget af MiL og om dette stykke værktøj er blevet leveret tilbage.

Dette skal ske ved at fremsøge en person eller værktøj via programmet eller ved at scanne en stregkode. Direkte fra programmet skal udskrives et ”låner kort” og et stregkode kort til værktøj (på en plastikkort printer).

På grund af festivallens placering er det meget vigtigt at programmet ikke er afhængigt af internettet, da der er meget begrænset mobildækning og ingen fastnet telefon eller fiber linjer på stranden i Tisvilde. Desuden er der ikke mulighed for fast strøm. Hele festivallen kører på generatorer. Så et klart krav fra festivallen var at det kunne køre på en bærbar, så man var sikret imod nedbrud på grund af mindre strømafbrud.

Sammen blev der udarbejdet en meget simpel model som blev brugt til den videre idéudvikling. Se Figur 2.



Figur 2 Den første simple model tegnet af systemet.

På baggrund af ovenstående har MiL en række ønsker til programmet, hvor så mange som muligt skal med i første version. Efter endt projekt, er det planen fra MiL’s side, at videreudvikle på programmet.

MiL's ønsker i tilfældig rækkefølge:

- Mulighed for generel lagerstyrring, så det ikke kun er udlån, der bliver registreret
- At kunne generere bestillingslister og taste modtaget bestillinger ind.
- Afgrænse materialer på lageret så kun personer, der tilhører bestemte grupper/hold kan afhente dem
- En form for budgetstyring til de forskellige hold/grupper, så de ikke kan bestille mere end, hvad deres budget tillader
- Registrering af ind-lejet udstyr, f.eks. gaffeltruck, med returnerings tidspunkt og lejeaftale
- Mulighed for at importere person data direkte fra et Excel-ark eller lignende datafil, fra programmet "HEAP", som bruges til at registrerer alle, der arbejder på festivallen
- Importere lager data fra Excel-ark eller datafil
- Leverandøroplysninger
- Statistik over mest brugte værktøj og materialer
- Brugerdefinerede notifikationer ved lav lagerbeholdning.

1.4.1 Krav til programmet

Fra MiL's side har der været følgende krav:

- Programmet skal kunne køre uden nogen form for netværk
- Programmet skal kunne køre på en Windows 7 computer med lav hardware ydelse
- Programmet skal have en enkelt GUI, hvor de primære funktioner let kan tilgås
- Programmet skal bruge en database til lagring af data
- Programmet skal køre periodisk backup af data
- Programmet skal være robust bygget så størstedelen af fejl ikke får programmet til at miste data eller lukke ned
- Programmet skal gemme alle data ændringer med det samme, så der ved f.eks. ved strømafbrydelse ikke mistes data

- Programmet skal have et administrationsmodul som kan ændre direkte i data som er afgrænset med password
- Det skal være muligt at flytte databasen med data fra én computer til en anden og importere denne i programmet.

Underkrav aftalt mellem udvikler og MiL:

- Programmet skal så vidt muligt ikke tillade brugeren at lave ”ulovlige” handlinger eller handlinger der skader data
- Backup location skal kunne indtastes fra programmet
- I første udgave skal fejl vises med ”stack trace” for hurtig løsning af problemer i første live miljø. Dette skal dog senere ændres til at give brugbare fejlmeddelelser til en alm. bruger.

Hardware eller Software krav fra udvikler godkendt af MiL:

- Operativ system: Min. Windows 7
- Minimum .Net 4.5.2
- Microsoft SQL Server 2014 Express LocalDB
- Processor: 1 GHz
- Ram: 1 GB

1.5 Rapport oversigt

Analyseafsnittet kortlægger de tanker, overvejelser og beslutninger, der blev taget inden projektet gik over til selve udviklingsfasen. De vigtigste overvejelser som blev gjort i starten af projektet vil blive gennemgået, sammen med en kortlægning af de valgte teknologier.

Designafsnittet tager udgangspunkt i de valgte teknologier, og hvordan det blev planlagt, at de skulle implementeres. Der vil også blive givet en oversigt over de designprincipper som udviklingen efter planen skulle bruge.

Implementerings- og testafsnittene beskriver det endelige program. Hvilke funktioner, der er blevet implementeret pr. d. 31/05-2016, og hvordan de fungerer.

Under konklusionen kan man læse MiL's egen vurdering af programmet, samt en mindre oversigt over den planlagte videreudvikling.

2

ANALYSE

2.1 Risikoanalyse

At udvikle et helt nyt system som forventet taget i brug 5 måneder efter projektstart har en del medfødte risici.
MiL har aldrig selv været med til udvikle et software produkt, ligesom personerne fra MiL's side har meget begrænset IT-viden, og ingen erfaring med at udforme en kravsspecifikation. I et forsøg på at gøre MiL klar på de udfordringer, der er med et nyt system, er der udarbejdet en overordnet risikoanalyse med de typiske faldgruber for et softwareprojekt.

- **Ændringer i systemet eller krav**
  Når man udvikler tæt sammen med en kunde eller igennem test og generelt kendskab, der kommer med at udvikle et program, opstår der altid ændringer i det man forventede, da projektet startede. Her er det vigtigt, at man hurtigt finder ud af, hvad en ændring vil betyde for andre funktioner, hvor meget tid det vil tage at implementere den, og vurdere, hvordan der evt. skal prioriteres anderledes i forhold til andre funktioner eller udvidelser til programmet.

- **Afgrænsning af systemet, prioritering af udvidelser**
  Allerede fra start har MiL mange ting de ønsker i programmet, som ligger udenfor kernefunktionerne. Der vil uden tvivl i udviklingsperioden også komme flere idéer til nye funktioner til. MiL vil selvfølgelig gerne have så mange af disse funktioner med som muligt. Samtidig vil systemet være ubrugeligt for MiL hvis det ikke er klar til brug 1. juni. Det er derfor vigtigt at der er klare aftaler om prioritering af disse funktioner.

- **Sygdom**
  Ved enkeltmands projekter kan sygdom let komme i vejen for en rettidig aflevering.

- **Data tab eller større hardware nedbrud**
  I alle udviklingssituationer er det vigtigt at have backup af sit arbejde, da det er udviklerens ansvar at mindske denne risiko vha. ekstern versions-styring og backup.

- **Kode fejl**
  Al kode har fejl. Men man kan undgå mange og forsøge at sikre sig imod at disse fejl påvirker driften i selve koden. Oven i det er der en aftale mellem udvikleren og MiL om at udvikleren er tilstede under hele festivallen 2016 for at hurtigt kunne afhjælpe eventuelle fejl.

2.2 Usecases

I opstartsperioden af projektet blev der brugt usecases til at hjælpe MiL med at afdække deres forventninger og behov.
Der er 5 usecases hvor UC1, UC2 og UC3 er kerne funktioner imens UC4 og UC5 er til den højest prioriterede tilføjelse.
Alle usecases kan ses i deres helhed i appendix 8.1.

2.2.1 Aktørbeskrivelser

Udledt af arbejdet med usecases, er der identificeret 2 hovedaktører (Ekspedient og Lageransvarlig) og 1 sekundær aktør (frivillig).

2.2.1.1 Ekspedient

En person, hvis opgave det er at registrere alt, hvad der bliver udleveret og tilbageleveret på lageret. Ekspedienten vil i driftdagene have mest brugstid i programmet.

2.2.1.2 Lageransvarlig eller administrator

En person, der har et overordnet ansvar for lageret. Vedkommende har ansvaret for at oprette værktøj og materialer i systemet inden festivallen starter. Den ansvarlige har myndighed til at ændre direkte i data i systemet.

2.2.1.3 Frivillig

En frivillig, der arbejder på festivallen. Vedkommende er forudsætning for at de fleste aktioner i systemet sker, men vil ikke være en direkte bruger af systemet.

2.2.2 Usecase beskrivelser

Usecase 1, 2 og 3 beskriver hovedfunktionen i systemet. Først en frivillig, der låner et værkstøj. Og derefter returnerer det igen, og sidst at en person, der endnu ikke er oprettet i systemet, bliver oprettet.

Usecase 4 og 5 er de primære udvidelser af programmet som festivallen ønsker men også de største. Nemlig muligheden for direkte i programmet at lave bestillingslister og håndtere de enkelte grupper/holds budgetrammer.

Disse usecases blev udarbejdet inden selve kodningen af programmet gik i gang. De yderligere udvidelsesmuligheder, beskrevet i 1.4 problemformuleringen, er kommet til i løbet af udviklingsarbejdet og test (i samarbejde med MiL).

2.3 Stabilitet og robusthed

MiL har igennem hele forløbet været meget klare på, at systemet skal være robust nok til at kunne klare en del brugerfejl uden at ”vælte” og således at data vil have svært ved at gå tabt.

I denne sammenhæng er et robust program, ét der kan detektere og komme videre fra fejl på en acceptabel måde til mindst mulig irritation for brugeren.

I udviklingen skal der lægges meget vægt på de ”robuste principper”:

I. Paranoia
II. Dumhed (Stupidity)
III. Usikre implementeringer (Dangerous Implements)
IV. Umulige resultater (Can't happen).[2]

I. Paranoia

[2] Robust Programming af Matt Bishop. Department of Computer Science. University of California at Davis

Systemet stoler ikke på noget og det går ud fra det selv har fejl. Ved brug går systemet ud fra at alle input vil forsøge at smadre det. Kalder systemet andre funktioner, holder det øje med, at det lykkes.

II. Dumhed
Systemet går ud fra at en bruger eller en anden del af systemet intet ved om, hvad de laver. Systemet designes således at det kan håndtere parametre og inputs der er forkerte eller falske. Systemet giver udførlige og let forståelige fejlbeskeder.
Systemet stopper eller reparerer en handling i det øjeblik, der opstår en fejl, så der ikke ophober sig fejl.

III. Usikre implementeringer
Brugere eller andre systemer har ikke adgang til funktionsbiblioteker, datastrukturer, adresser eller størrelser på data. Systemet bruger interfaces til at sikre bagvedliggende funktioner. Systemet bruger ikke globale variabler. Systemet forventer ikke, at noget er statisk. Systemet forventer ikke, det har adgang til data, heller ikke selv om det har haft det på et tidligere tidspunkt.

IV. Umulige resultater
Systemet tager højde for, at det umulige kan ske og har metoder til at håndtere dette, hvis det alligevel sker.

2.4 WinForms eller WPF

WinForms som GUI, der er bygget oven på det eksisterende Windows API, er et af de mest brugte, testede og dokumenterede GUI systemer, der findes til Windows platformen.
Der findes utrolig mange 3. parts elementer både til at købe eller gratis tilgængelige.
Designer i Visual Studio regnes for at være en af de bedste og gør det meget let og hurtigt at udvikle en GUI til de fleste ikke grafisk krævende applikationer.

WinForms har siden 2014 været i det der kaldes ”maintenance mode”, det vil sige at Microsoft ikke længere udvikler nye funktioner til WinForms, men at de stadig retter fejl.

Den store tilgang af mobile enheder i de senere år har været baggrunden for et projekt, der tog WinForms og videreudviklede det til Xamarin.Forms, som kan anvendes på de tre største mobil operativ systemer, iOS, Android og Windows 10 mobile.

WPF blev lanceret i 2006 som et XAML baseret alternativ til WinForms, der skulle gøre det lettere at manipulere de grafiske elementer. Der blev også fokuseret mere på at adskille den bagvedliggende logik fra selve den grafiske del af applikationen. [3]

WPF har en stor fordel, når større datamængder skal vises på skærmen, ide WPF tager moderne grafiske hardware i brug, og har fra Windows 7 en stor fordel i ydeevne i forhold til WinForms.

Der er visse funktioner der ikke findes i WPF som gør i WinForms. Det muligt enten at lave denne funktion selv ellers findes der en ”Container” i WPF som kan indeholde et WinForms element.[4]

På mobilsiden af udvikling har WPF desværre ikke fulgt helt med. Der findes i skrivende stund et nyudviklet program, der kan konvertere WPF til Universal Windows Platform. Dette er dog stadig meget nyt og kan kun bruges til Windows 10 mobile platformen.

Dette har medført en del spekulationer om hvorvidt WPF er ved at blive droppet - dette benægtes dog af Microsoft. [5]

[3] http://www.wpf-tutorial.com/about-wpf/wpf-vs-winforms/
[4] http://www.infragistics.com/community/blogs/devtoolsguy/archive/2015/04/17/windows-presentation-foundation-vs-winforms.aspx
[5] http://pragmateek.com/is-wpf-dead-the-present-and-future-of-wpf/

2.5 Database

Da database teknologi skulle vælges, var det med programmets begrænsninger i mente. Programmet skulle køre uden internet, og kunne tage backup af sine data uden adgang til en online og ekstern database.
Der blev kigget på to forskellige systemer SQLite og LocalDB. Featuresættet på disse to er langt hen af vejen ens, dog med nogle få afvigelser.

SQLite kører som en del af programprocessen og kan installeres uden administratorrettigheder. Dens max database størrelse er 140 TB. Den har en noget mindre installationsstørrelse men kommer med nogle begrænsninger i featuresettet og typen af queries, som man kan bruge.[6]

LocalDB kører som en separat proces startet sammen med program processen. Maximale database er på 10 GB. LocalDB køre med den fulde SQL Server Express engine, og har ingen begrænsninger i forhold til xml indhold, Stored procedures, Transact SQL eller LINQ. [7]

2.6 Hardware

Da systemet blev diskuteret første gang, var udgangspunktet en smartphone App baseret på QR-kode eller en RF-id chip.

Smartphone App blev dog droppet på baggrund af følgende:

- Det stod hurtig klart, at man som minimum skulle understøtte både Android og iOS telefoner. Dette ville forlænge udviklingstiden meget
- Ligeledes ville den gennemsnitlige smartphone have problemer med at køre i tilstrækkelig tid på en opladning
- Krav om at data omkring lagerstatus og udlån skulle kunne synkroniseres løbende. F.eks. over et dertil opsat LAN som enheden skulle være tilkoblet, for at systemet ville virke.

[6] http://www.sqlite.org/different.html
[7] https://blogs.msdn.microsoft.com/sqlexpress/2011/07/12/introducing-localdb-an-improved-sql-express/

En enkelt Windows bærbar computer giver et ”single point of failure”, men en placering af denne, indenfor i værktøjs- og materialelageret, vil give den ly for vejret og den bedste mulighed for, i længere perioder, at være sat til strøm.

En RF-id chip har fordel i at være lille og ville let kunne sættes på alle materialer. Men en undersøgelse viste desværre, at for at få den nødvendige robusthed imod vind, slid og vejr, ville det blive meget dyrt.
En stregkode printet på et almindeligt stykke papir ville ikke holde længe på en byggeplads, og hvis denne skal lamineres efter print, vil det komme til at tage meget lang tid at markere materialer og værktøj op.
En stregkode printet på et standard plastikkort er en større investering, men det giver den ønskede robusthed, og tager ikke lang tid at printe.

3

DESIGN

3.1 Overordnet plan

Overordnet var det vigtigt for MiL at være en del af hele udviklingsprocessen, da de aldrig har været med til sådanne projekter før. Der blev lagt en plan, som minder lidt om principperne i en SCRUM agil udviklingsmetode.
Da de ledende personer i MiL aldrig har hørt om agil udvikling, blev det en meget modificeret og skrabet udgave deraf.

Ca. hver 14. dag blev der planlagt et fast møde, hvor MiL skulle præsenteres for programmets status. Ved hvert af disse møder, skulle planen for, hvilke features der skulle prioriteres, udarbejdes i samarbejde.

Herudaf voksede der noget, der lignede en backlog over features, som MiL ønskede, og fejl der blev fundet. Dette hjalp dem meget med at prioritere, hvilke funktioner de gerne ville have med i første version.

I anden halvdel af projektperioden blev det planlagt, at personer i MiL fik programmet ”med hjem”, hvor de kunne prøve at bruge det på egne maskiner. Til det næste møde kunne de så komme med deres indtryk, fejl og eventuelle forslag til ændringer.

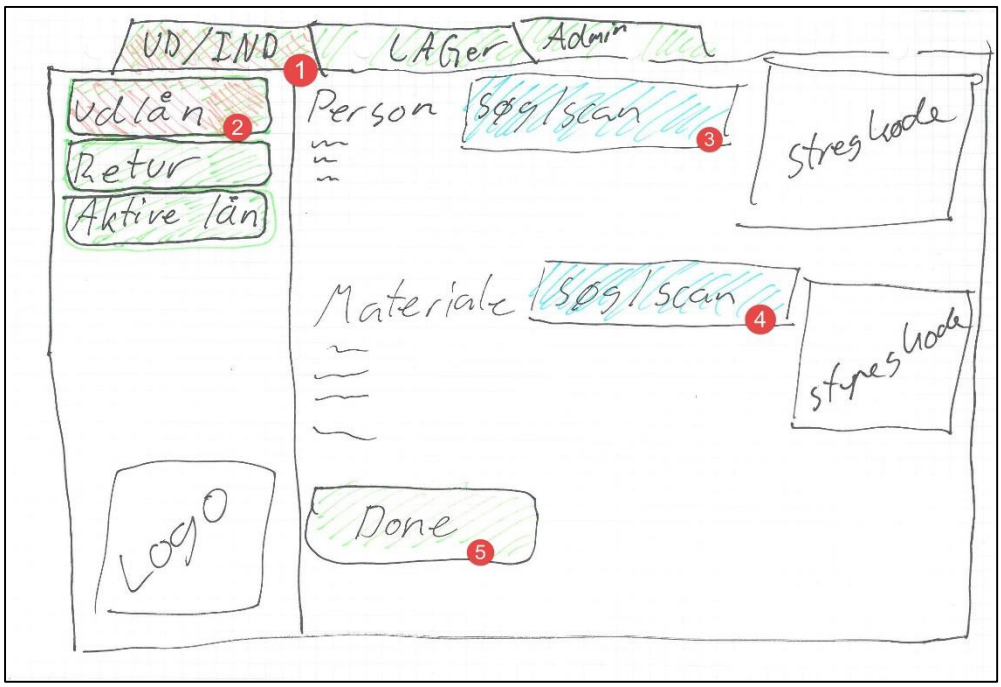
Det viste sig meget hurtigt at MiL’s ønsker til dette projekt ikke kunne lade sig gøre inden for den 3 måneder lange projektperiode. Derfor blev det planlagt, at placere udviklingen af de sidste ønskede funktioner og drifttest i måneden efter projektets afslutning.

3.2 Mockups

Der blev udarbejdet nogle få muckups af GUI og det primære udlånsflow. Målet var et meget enkelt og overskueligt aktivitets flow, så den ekspedient, der skal anvende programmet, ville kunne gøre dette med meget lidt oplæring. De uredigerede versioner af tegningerne kan findes i appendix 8.2.

3.2.1 Udlåns mockup og flow

Det primære udlånsflow bliver beskrevet på Figur 3. Grøn og orange farve markerer faneblad eller en knap. Blå er et tekstfelt.

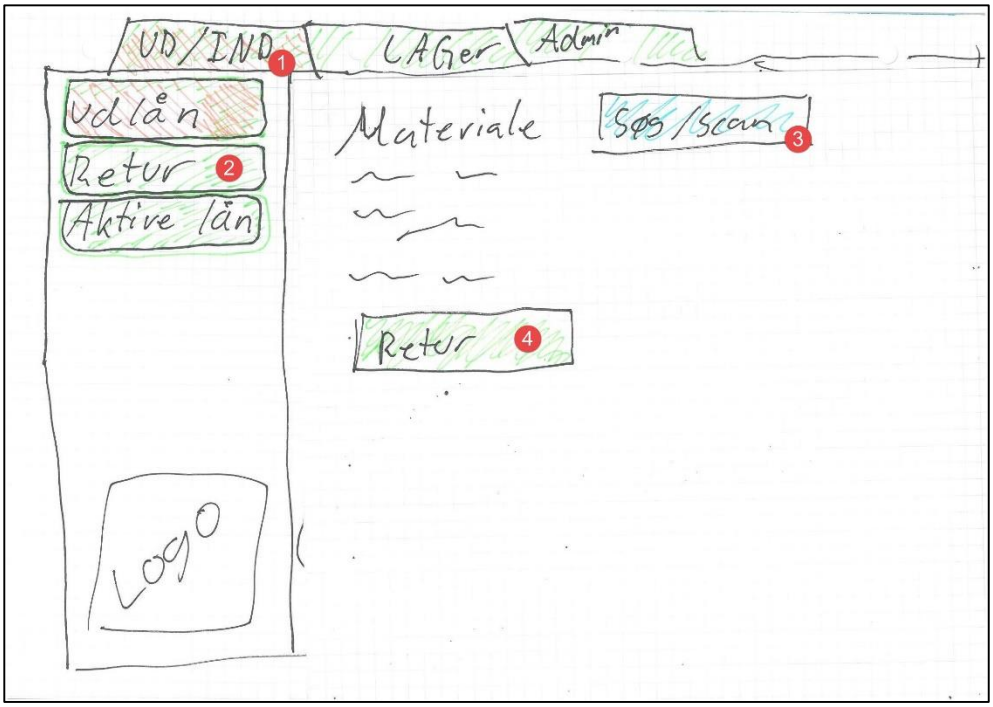


Figur 3 "Ud" Mockup (8.2.1) hvor brugerflow er indtegnet som nummer 1 til 5.

Punkt 1 og *punkt 2*: Brugeren navigerer til siden med udlån.
Punkt 3 og punkt 4: Brugeren scanner eller søger den ønskede person og materiale frem.
Punkt 5: Knap der kun kan trykkes på, hvis personen kan låne materialet. Bruger trykker på knappen og lånet er registreret.

3.2.2 Retur mockup og flow

Det primære returneringsflow bliver beskrevet på Figur 4. Grøn og Orange farve markerer faneblad eller en knap. Blå er et tekstfelt



Figur 4 Retur Mockup (8.2.2) hvor brugerflow er indtegnet som nummer 1 til 4

Punkt 1 og *punkt 2*: Brugeren navigerer til siden med retur.

Punkt 3: Værktøjet eller materialet scannes eller fremsøges.

Punkt 4: Hvis der er et aktivt lån med denne ting, registreres det som afsluttet, eller hvis det ikke er en udlånsartikel tilføjes det til lageret direkte.

3.2.3 Se aktive lån og find personen der har lånt materialet

For at kunne finde et manglende stykke værktøj blev der aftalt en metode til at se alle aktive lån og fremsøge info på den person, der har lånt værktøjet. Dette flow beskrevet på Figur 5 og Figur 6. Grøn og Orange farve markerer faneblad eller en knap. Blå er et tekstfelt.

Figur 5
Aktive lån 1 Mockup (8.2.3) hvor brugerflow er indtegnet som nummer 1 til 4

Punkt 1 og *punkt 2*: Brugeren navigerer til siden med aktive lån.

Punkt 3: Bruger vælger det ønskede lån og højreklikker.

Punkt 4: I højre klik menuen vælges vis person.

Figur 6 Aktive lån Mockup (8.2.4) hvor brugerflow er fortsat fra Figur 5 med numrene 5 til 7

Punkt 5: Et pop-op vindue åbnes med info på den valgte person fra lånet.
Punkt 6: Dette vindue skal kunne bruges til at finde andre personer igennem dette søgefelt, men er i denne sammenhæng ikke relevant.
Punkt 7: Når brugeren har den info han har brug for, kan vinduet lukkes.

3.3 GUI teknologi

Valget af GUI teknologi til dette projekt var ikke let. WinForms giver en hurtig måde at få noget på skærmen og med mulighed for senere at blive konverteret til at køre på mobile styresystem. Dog bliver programmer skrevet med WinForms meget hurtigt langsomme at afvikle, hvis der ikke lægges meget tid i at optimere det. Hertil kommer at det er det ret besværligt at få grafiske elementer til at fungere, som man ønsker det.
I modsætning hertil afvikles WPF langt hurtigere på moderne Windows systemer og giver en let mulighed for at skille GUI fra programlogikken. WPF har dog nogle svagheder, når man manipulerer direkte i data vist på skærmen.

I starten af projektet blev WinForms valgt primært på grund af, at der var langt større erfaring med denne teknologi fra udviklerens side. Sekundært for at det måske i fremtiden ville blive mindre besværligt at lave en mobil udgave af programmet.
Langt inde i processen blev det et stort ønske fra MiL's side af, at programmet så ud på en anden måde, og at det ville komme til at køre hurtigere i den første udgave. I samråd med udvikleren prioriterede MiL derfor, at der blev udviklet en WPF udgave af programmet, selv om der var en fungerende version, skrevet med WinForms.

Rapporten vil derfor i de følgende afsnit, hvor der er forskelle, skelne imellem WinForms versionen og WPF versionen. Her efter benævnt som **WF.v1** og **WPF.v1**.

Det overordnede design af GUI blev kortlagt tidligt med de mockup's fra det tidligere afsnit. Rent designmæssigt blev der lagt vægt på at adskille udlåns- og returdelen fra lagerinfo og administrationsdel.
I WF.v1 skal designet inddeles i 4 overordnede faner:

I. Loans
II. Stock
III. Orders
IV. Admin.

Programmets funktioner skal inddeles under disse faner. Dog underinddeles i emner som f.eks. udlån, returneringer og oversigt over udlån.
WPF.v1 er designet ligesom WP.v1 dog har de forskellige funktioner fået fane-blade i stedet for knapper, på grund af tekniske begrænsninger i WPF.
I begge versioner skal der gøres brug af at mindre funktioner og udvidet info bliver vist i et separat pop-up vindue.

3.4 Programarkitektur

Programmet bygges op meget simpelt med GUI forrest, der igennem et logiklag har forbindelse til en database.
I databasen er der flere forskellige tabeller. Der oprettes specifikke adgangslag, så en databaseforbindelse kun har adgang til en tabel af gangen.
I programlogik delen ligger dataforbindelserne og den generelle logik for programmet.

3.5 Database

Valget faldt på SQL 2014 Express LocalDB da den integrerede let og hurtigt med Visual Studio, og umiddelbart virkede som den bedst dokumenterede.
Ved projektstart var der en del usikkerhed om, hvordan databasen skulle bruges sammen med programmet; derfor kom det til at veje tungt i valget, at LocalDB har den fulde SQL Express engine.
Det at databasen ligger som en let tilgængelig fil .MDF talte også for valget.

3.6 Brugervenlighed

Der skal lægges stor vægt på at programmet er enkelt at bruge. For at give brugeren så kort vej som muligt til de funktioner, denne har brug for, skal programmet designes så at en specifik funktion kan tilgås med så få klik som muligt. Det skal forsøges at holde museklik i navigationen under 3 klik. Et navigationsflow kunne være:

1. Brugeren er på udlåns siden, men skal finde antal på et materiale på lageret
2. Her skal brugeren trykke på ”Stock” fanen og muligvis på knappen/fanen for at få en oversigt over alt der er på lager
3. Når denne side er fremme skal søgefunktionen automatisk være klar til, at brugeren indtaster en søgestreng og trykker søg/Enter
4. Herefter (hvis brugeren har brugt en søgestreng, der kan finde materialet) vises de ønskede data på skærmen.

For at hjælpe brugeren skal en funktion designes så intuitivt som muligt, og hvis der skulle ske en fejl fra brugerens side, skal programmet så vidt muligt fortælle, hvad der gik galt. I udlånsprocessen, hvor der både skal vælges et materiale og en person, der skal låne det, er det vigtigt, hvis et lån ikke kan gennemføres, at der gives en grund. Dette kunne f.eks. være, at der ikke er mere på lager eller at den frivillige ikke har rettigheder til at låne det specifikke stykke materiale.

Når brugeren scanner et ID eller selv indtaster det, skal programmet forsøge at finde det frem, når der er mulighed for det. F.eks. når et id bliver scannet, vil det 5 ciffer-id blive sat ind i den aktive tekstboks. Dette skal programmet opdage og forsøge at finde den indscannede med det samme, og sparer brugeren for et klik.

3.7 Fejlhåndtering

Da programmet skal kunne køre stabilt og med så få nedbrud som muligt er fejlhåndtering en vigtig del. Så vidt muligt og i det omfang det er nødvendigt, skal udviklingen følge de stabile principper beskrevet tidligere i rapporten (afsnit 2.3).
Data der bliver udvekslet imellem programmet og databasen bliver valideret af det tilhørende DataSet.

Eventuelle konverterings- eller SQL-fejl vil blive fanget i GUI elementet, der kaldte adgangsklassen. Her bliver den exception vist. Dog skal den i en videreudvikling fra prototypen kodes til at give en forståelig fejlmeddelelse, som kan læses af en person, der ikke er inde i hvordan koden fungerer.

Så vidt muligt skal det begrænses brugerens mulighed for at indtaste data som kan blive behandlet af programmet på en måde der giver fejl. F.eks. ved at man ikke kan indtaste bogstaver hvor programmet går ud fra at der bliver indtastet tal.

Igennem udviklingen skal statiske klasser og funktioner undgås så meget som mulig. Så vidt mulig skal en funktion ikke gå ud fra at noget andet i koden eksisterer uden at have testet om det gør først.

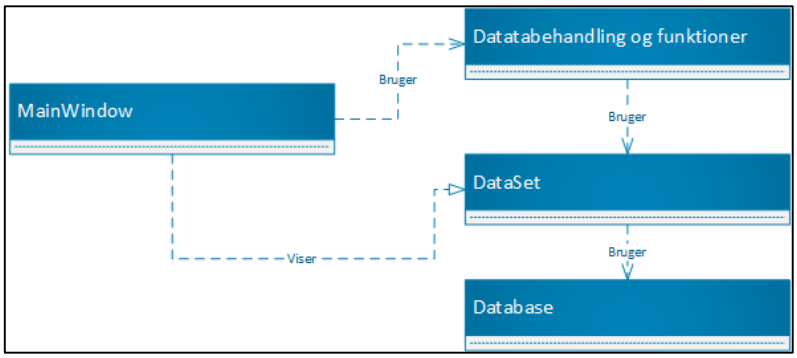
I prototypen bliver der gjort stor brug af meget brede ”catch” sætninger for at sikre imod uforudsete fejl. Dette er dog ikke praktisk i længden og skal i den videre udvikling laves om.

4 IMPLEMENTERING

4.1 Program struktur

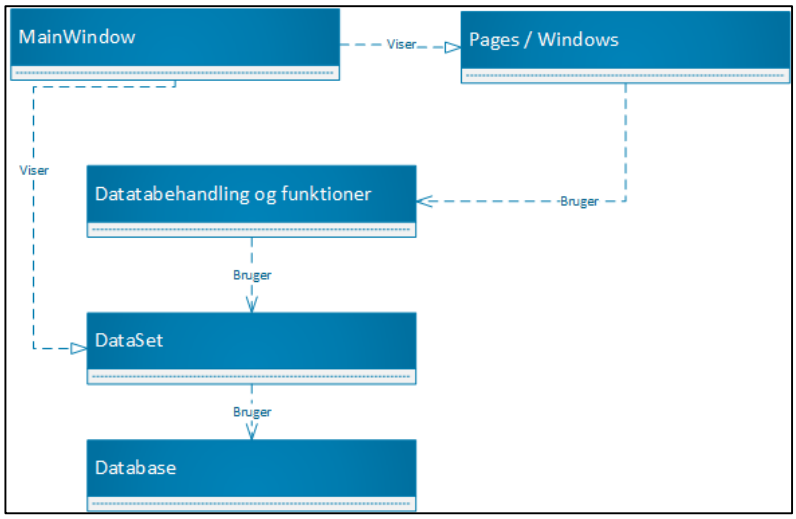
Da der er to prototyper af programmet, én med WinForms og én med WPF er det bygget en smule forskelligt op alt efter hvilken man kigger på.

Det meste af programlogikken er identisk i de to versioner. Dog som man kan se på Figur 7 og Figur 8, er det overordnede vindue noget mere adskilt fra databehandlingen og selve programlogikken.



Figur 7 WinForms version struktur

I WPF.v1 ligger programlogikken i WPF.sider og vinduer som opretter og bruger de DataSet og hjælpeklasser som den har brug for.
Der er dog en forbindelse i begge versioner, som går uden om det viste data flow. Dette sker i forbindelse med den periodiske backup af databasen. Her bliver der åbnet en forbindelse direkte til databasen fra backupklassen, som selv beder databasen om at lave en backup direkte i den fil sti, som det er sat til.



Figur 8 WPF version struktur

Et normalt data flow som f.eks. at ændre i et materiale i databasen, kan i begge versioner beskrives enkelt.
Brugeren navigerer til funktionen og får åbnet ”admin stock item” vinduet.
Her enten scanner eller indtaster brugeren ID på det materiale man vil ændre i.
I koden vil der blive undersøgt om alle felter er udfyldt.
Programmet prøver at indsætte de indtastede felter (Figur 9) i et ”stock_item.cs” objekt, og advarer brugeren om at alle ændringer er permanente, hvorefter programmet forsøger at gemme data i databasen.

```
private void button_submit_Click(object sender, RoutedEventArgs e)
{
    if (isEditingItem)
    {
        if (allFieldsFilled())
        {
            try
            {
                item.Name = textBox_item_name.Text;
                item.Amount = Convert.ToSingle(textBox_item_amount.Text);
                item.Unit = textBox_item_unit.Text;
                item.Type = textBox_item_type.Text;
                item.Place = textBox_item_place.Text;
                item.Price = Math.Round(Convert.ToDouble(textBox_item_price.Text), 2);
                item.Supplier = textBox_item_supplier.Text;
                item.Group = textBox_item_group.Text;
                item.Loan_item = Convert.ToBoolean(comboBox_Item_loan_item.SelectedValue);

                var result = MessageBox.Show(en.editStockItemIsPermanant(), en.areYorSure(),
                        MessageBoxButton.YesNo);

                if (result == MessageBoxResult.Yes)
                {
                    stockDB_access stockDB = new stockDB_access();

                    int res = stockDB.editStockItem(item);
                    Close();
                }

            }
            catch (Exception Ex)
            {
                MessageBox.Show(Ex.ToString());
            }
        }
```

Figur 9 – Udsnit af stock_item_admin.xaml.cs fra linje 190 i Quick Codex WPF projektet.

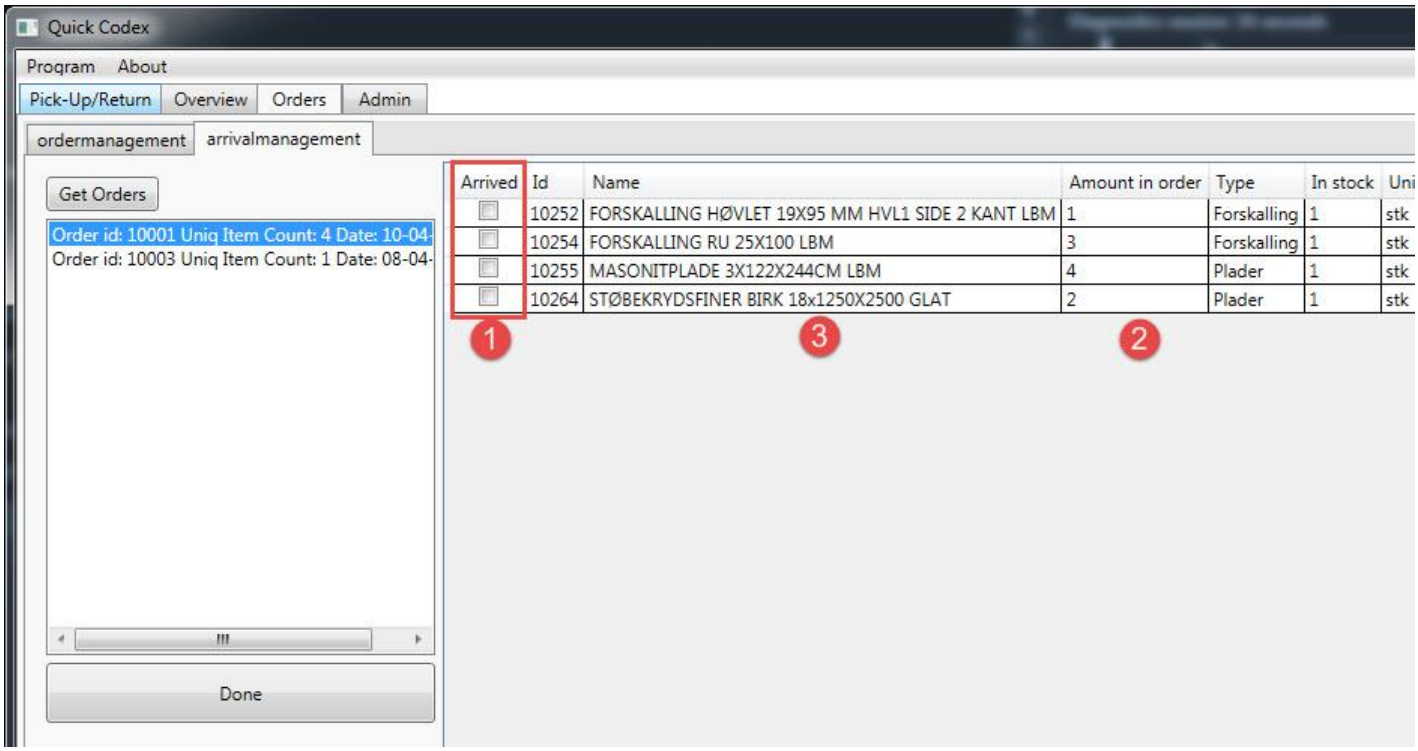
Hvis der sker en fejl under konvertering eller adgangen til databasen, vil fejlmeddelelsen i denne prototype være et udskrift af den fangede Exception som bliver vist for brugeren.

Det er dog forsøgt at minimerer dette ved at felter som ”price” bruger ”Regular Expressions” (Figur 10) til at begrænse hvad brugeren kan indtaste i feltet.

```
299         private void textBox_item_price_PreviewTextInput(object sender, TextCompositionEventArgs e)
300         {
301             e.Handled = new Regex("[^0-9.]+").IsMatch(e.Text);
302         }
```

Figur 10 Udsnit af stock_item_admin.xaml.cs fra linje 299 i Quick Codex WPF projektet.

En af databasens tabeller bliver brugt en smule anderledes end de andre. Når en bruger vil registrere, at en bestilt vare, fra en ordre, er ankommet, bruges en hjælpetabel, som viser en oversigt over, hvilke varer der var i bestillingslisten. Det letter markeringen af hvad der er ankommet, og hvad der ikke er (Figur 11).



Figur 11 Screen shot af WPF.v1 arrival management (1) til at markerer som ankommet, (2) hvis det ikke er det korrekte nummer. (3) hvis der er en ændring i selve materialet.

Brugeren har mulighed for at klikke på det, eller de, materialer som er ankommet. Her er også mulighed for at ændre antallet af det ankomne materiale. Hvis der er en forskel imellem det bestilte og det ankomne, kan dette også rettes direkte i denne tabel.

Hvis brugeren har ændret noget andet end hvor mange materialer, der er ankommet, vil vedkommende blive bedt om at oprette dette materiale i databasen. Hvis der er materialer i ordren, som endnu ikke er ankommet, imens andre er, vil disse blive overført til en ny ordre, med samme bestillingsdato som den tidligere.

Her kommer en af forskellene imellem WinForms og WPF frem. Hvor det i WinForms er meget enkelt at detektere ændringer, der sker direkte i data i GUI, er det en noget omstændelig proces i WPF.

På Figur 12 er WF.v1 til venstre og WPF.v1 til højre.

```
private void arrivalDataGridView_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex != -1)
    {
        if (e.ColumnIndex != 0 )
        {
            if (e.ColumnIndex != 4)
            {
                int row = e.RowIndex;
                arrivalDataGridView.Rows[row].Cells["edited"].Value = true;
            }
        }

    }
}

#endregion
```

```
226
227     private void arrivalDataGrid_CellEditEnding(object sender, DataGridCellEditEndingEvent
228     {
229         if (!e.Column.Header.Equals("Arrived") && !e.Column.Header.Equals("Amount in orde
230         {
231             if (e.EditAction == DataGridEditAction.Commit)
232             {
233                 string tmp = arrivalDataGrid.GetCell(e.Row, 1).ToString();
234                 tmp = new Regex(@"\d+").Match(tmp).Value;
235                 editedItemsList.Add(int.Parse(tmp));
236             }
237         }
238         else if (e.Column.Header.Equals("Arrived"))
239         {
240             if (e.EditAction == DataGridEditAction.Commit)
241             {
242                 string tmp = arrivalDataGrid.GetCell(e.Row, 1).ToString();
243                 tmp = new Regex(@"\d+").Match(tmp).Value;
244                 arrivedItemsList.Add(int.Parse(tmp));
245             }
```

Figur 12 Sammenligning i mellem WF og WPF i arrival_management Større version i appendix 8.4

I WF.v1 var det utrolig enkelt at have et midlertidigt felt, som kun blev aktiveret, hvis brugeren havde ændret i noget, som ville gøre det nødvendigt at oprettet et nyt materiale i databasen. Når brugeren trykker på ”Done” bliver alle rækker i tabellen gået i gennem for at se om ”edited” flaget er sat og programmet beder brugeren om at oprette et nyt materiale.

I WPF.v1 blev løsningen at lægge ID for de ændrede materialer i en liste for sig selv og derefter senere behandle disse data. Efter al sandsynlighed findes der en bedre løsning, men da dette umiddelbart virker, blev det ikke prioriteret særlig højt at finde en anden løsning, selvom om man i dette tilfælde ikke helt bruger WPF som det er tiltænkt.

Gruppernes budget bliver håndteret primært ved hjælp af et, eget udviklet, klasse bibliotek (Quick Codex Budget.dll). Det er et meget simpelt bibliotek, der giver mulighed for at give en gruppe et budget, hvor der, hver gang gruppen bestiller noget, bliver registreret, hvad de bestiller, og hvem der gjorde det. Det er som udgangspunkt ikke muligt i programmet at bestille for et større beløb end, hvad gruppen har til rådighed i deres projekt.
Dog er der indbygget i biblioteket, at denne funktion kan blive implementeret. I hjælpe klassen ”order_control.cs” bliver det håndteret, når en gruppe bestiller en ting. Gruppen har sit eget budgetobjekt som bliver kaldt til at udføre opgaven (Figur 13).

```
group = groupDB.getGroupByName(person.Group);
Tuple<bool, string> budgetResult = group.Budget.transaction(-(selected_stock_item.Price * amount), "Order: " + selected_order.Id + ". Item: " + selected_stock_item.Id, person.Name);
if (budgetResult.Item1)
{
    orderDBaccess orderDB = new orderDBaccess();
    orderDB.UpdateItemIDsList(selected_order);
    groupDB.updateBudget(group);
}
return budgetResult.Item2;
```

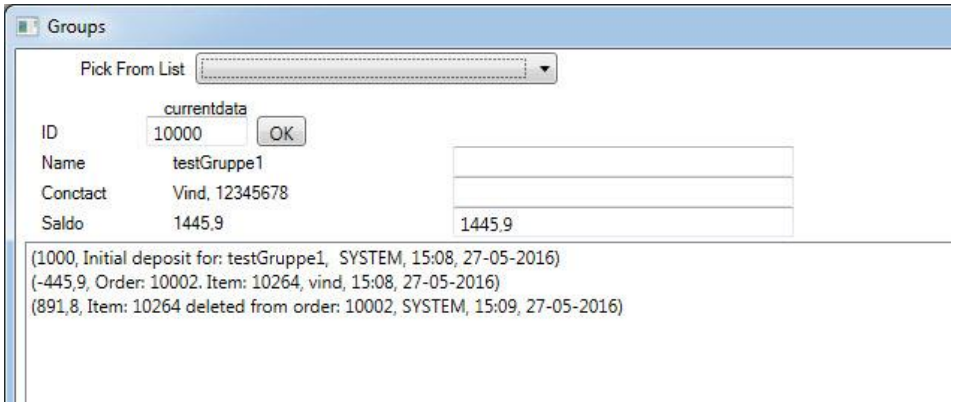
Figur 13 Udsnit af order_control.cs

I Budget (Figur 14) bliver transaktionen behandlet og lagt i en liste af tupler inde i objektet selv, og returnerer med besked om hvorvidt der var dækning og den nye saldo.

```
public Tuple<bool,string> transaction(double amount, string reference, string person)
{
    bool result = false;
    string returnVal = "transaction: ";
    double tmp = Saldo + Math.Round(amount, 2);
    if (tmp > 0)
    {
        result = addTransaction(Math.Round(amount, 2), reference, person, DateTime.Now.ToShortTimeString(), DateTime.Now.ToShortDateString());
        if (result)
        {
            returnVal = returnVal + ". New saldo: " + Saldo;
        } else
```

Figur 14 Udsnit af budget.cs i Quick Codex Budget

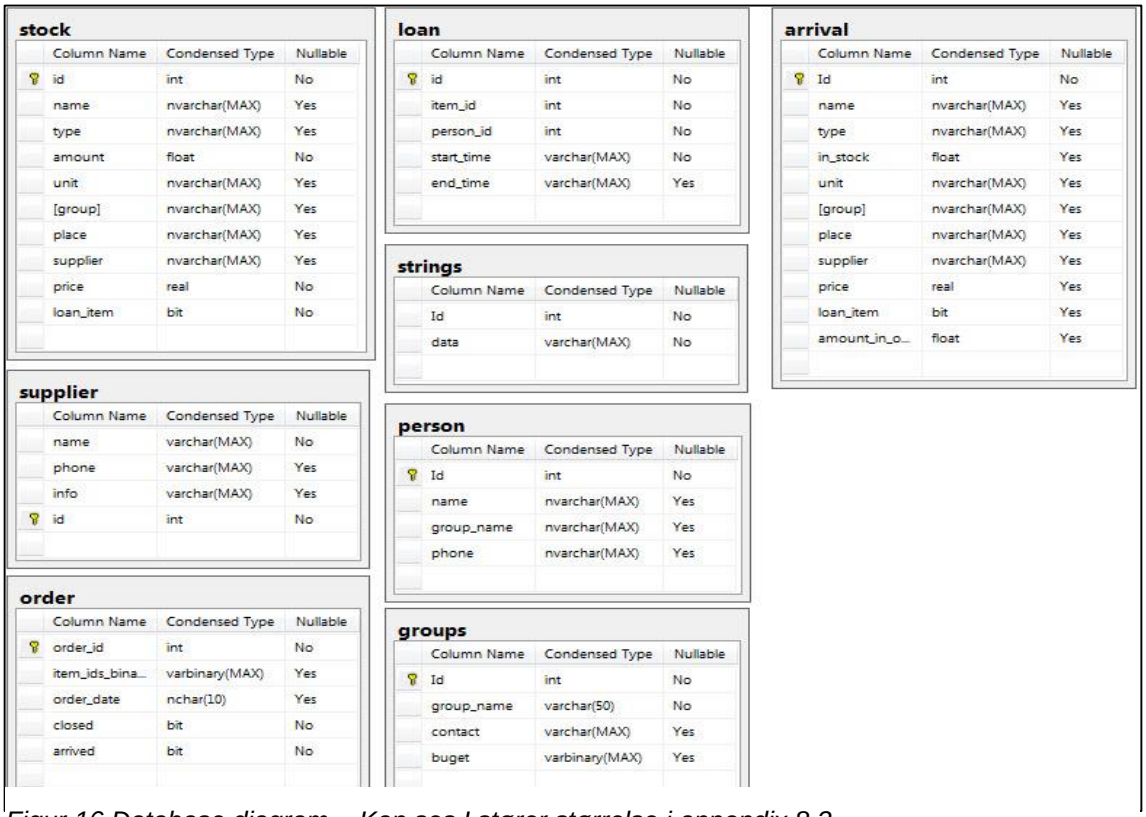
Alt dette resulterer i at en gruppe let kan få et overblik over, hvordan de har brugt deres penge. Dette bliver tilgået i administrationsdelen af programmet. (Figur 15)



Figur 15 Udsnit af Gruppe admin vindue

4.2 Database

På Figur 16 ses et overblik over tabellerne i databasen. De kan inddeles i to grupper tabeller til at gemme data i og arrival som kun bruges i forbindelse med at ordrer ankommer og bliver lagt ind i ”stock” tabellen.



Figur 16 Database diagram – Kan ses I større størrelse i appendix 8.3

Alle primær nøgler, med undtagelse af den i ”strings” tabellen, bliver autogenereret ved indsættelse af en ny linje.

1. ”Stock” tabellen indeholder data om de enkelte materialer som lageret håndterer.
2. ”Supplier” tabellen indeholder de firmaer som forsyner festivallen.
3. ”Loan” tabellen indeholder ID på den person og det materiale der er blevet lånt ud, samt start og en eventuel slut tid på lånet. Et lån som ikke er afsluttet har sin sluttid markeret med ”x”.
4. ”person” tabellen indeholder alle systemets lånere.
5. ”groups” tabellen indeholder de grupper eller de frivillige, som er en del heraf. Denne tabel indeholder også et binært felt, der har gruppes buget info.
6. ”order” tabellen indeholder alle materialeordrer i systemet. Her er også et binært felt, der indeholder en liste af tupler som hver har et id på det bestilte materiale, og hvor meget der er bestilt af det.
7. ”strings” indeholder en krypteret version af administrator passwordet.
8. ”arrival” bliver brugt når brugeren markerer materialer i en ordre som ankommet.

Inden programmet kommer i drift skal der i databasen laves nogle ændringer. Der er i denne prototype ingen relationer imellem tabellerne i databasen, dette skal ændres så at. F.eks. ”item ID” i loan tabellen har en relation til et ”id” i stock tabellen.

Alle forbindelser til databasen bliver håndteret af en autogenereret ”Data Set” lavet af Visual Studio. Disse gør det let at manipulere og validere data i hukommelsen og derefter at skrive det til databasen.[8] Disse bliver kun brugt direkte af GUI, når den henter data fra en database til et tabel i GUI’en. Alle andre kald går igennem en ”xxxDBaccess” (Her står xxx for navnet på den tabel, man ønsker adgang til) klasse, der sikre at dette sker korrekt.

[8] https://msdn.microsoft.com/en-us/library/8bw9ksd6.aspx

Særligt ved ”order” tabellen er dog, at der også er en hjælpeklasse med navnet ”order_control” som faciliterer er de mere komplekse opgaver, når et materiale bliver tilføjet til en ordre. Dette blev nødvendigt, da de forskellige gruppers budgetter også blev implementeret.

4.3 GUI teknologi

Som beskrevet tidligere findes programmet i to versioner, en WinForms og en WPF. Langt det mest programlogik er identisk i de to udgaver.

4.3.1 WinForms

WF.v1 er bygget op omkring et MainWindow som indeholder al GUI logik som ikke er et pop-up vindue.
Klik på en knap bliver fanget i en event og ved hjælp af nogle hjælpefunktioner udføres den ønskede handling.
I WinForms versionen bliver forms.DataGrid meget brugt. Disse viser og indeholder data fra databaserne, der så kan manipuleres og gemmes.

4.3.2 WPF

WPF.v1 er også bygget op omkring et MainWindow, dog er de fleste funktioner lagt ud i separate WPF pages for at minimere program logikken i hovedvinduet. Programmets primære funktioner tilgås igennem en WPF.Page. Det er disse sider, der opretter og bruger de kontakt- og hjælpe-klasser som den funktion har brug for.

5 TEST

5.1 Blackboxs test

Blackbox test blev udført i samarbejde med Christian W. Nielsen, en medstuderende på Diplom IT. For hver gang en ny feature blev implementeret, eller større fejl blev rettet, overtog Christian den seneste version af programmet og udførte test på det. Det var en stor fordel at bruge en med programmør erfaring til dette, da vi igennem både test og samtaler fandt rigtig mange fejl eller ting, der kunne optimeres.

5.2 Bruger test

Den første brugertest blev udført af MiL med udvikleren ved deres side. Dette gav flere ændringer end fejl som udbytte. Der blev fra MiL's side lagt mange timer i at teste for at hjælpe med at finde fejl. Dette medførte identifikation af en del fejl. Det var igennem disse test at mange af MiL's ekstra ønsker kom frem.

I starten brugte testeren en guide/liste til at teste funktioner (Figur 17). Senere gik testeren meget selvstændigt til arbejdet og havde kun brug for at få at vide, når der var en ny funktion, der skulle testes.

guide:
Installer "SqlLocalDB.msi"
Installer "Quick Codex 1.0"
Start Quick Codex ved at klikke på genvejen på dit skrivebord.

hvis du møder en fejl må du meget gerne tage et billede af teksten med din mobil og sende det tilbage til mig.
hvis du åbner programmet efter at have lukket det vil du få en meget lang fejl, denne er lige meget og vil ikke ske i virkeligheden (det er en fejl pga. de test data jeg lægger ind)

jeg vil meget gerne have du tester:

- låne en ting ud
  Du kan f.eks. bruge ID "10352" som materiale og "10002" i person.
- Se at du har lånt en ting ud ved at trykke på "Active Loans" (prøv her at højre klikke på det udlån du lige har lavet)
- Returnere den ting du lige har lånt ud.
- Prøv at finde en ting at låne ud ved at søge på navnet. f.eks skriv "ga" og tryk "Enter". Så skulle gaffatape komme op og du kan vælge det ved at dobbelt klikke på navnet.
- Udforsk hvad du kan se i fane bladet "Stock" pråv f.eks at bruge søge feltet.
- Tryk på fanebladet "Orders"
- Luk den åbne ordre ved at trykke "Close Order", (dette virker ikke 100% i nu, du lukker vinduet ved st trykke på "X" i top højre hjørne)
- Find en lukket ordre ved at trykke på "Arrival Management" og derefter "Get Orders"
- Tryk på ordre "10003" og ude til højre tryk et hak i "Has Arrived". Tryk "Done" denne ordre er nu lagt ind i systemet du kan se der er kommet en mere ved at klikke på "stock" fanen og finde materiale id "10252"
- Prøv at trykke på "ToggleAdmin" i toppen. password er "Hest". der efter tryk på "Administration" fanen.
- prøv de forskellige knapper: ændre og oprette nogle ting. du kan finde dem igen i "Stock" fanen.
- Tag en pause ved at trykke på "Help" i øverste venstre hjørne og tryk på "about" ;)
- Hvis du er modig tryk på "Program" og der efter "Settings" i top venstre hjørne.
- Her kan du ændre admin password og rode med backup af databasen
- Hvis du vil teste backup kan du trykke på den røde "Delete data In DB" -> tryk på "Program" og der efter "Settings"
- tryk på på "browse" under "no file selected"
- tryk på "Database1_backup.bak" også "open"
- tryk på "Restore"
- Hvis dette går galt kan du afinstalere programmet(via kontrol panel) og installere det igen ved at brugen filen "Quick Codex 1.0" som du hentede til at starte med.
- Ellers tryk rundt og se hvad der sker.

du har en bruger/"person" der har ID: 10002
vil du ikke lige svare når du ser denne mail.

Figur 17 En af de udarbejdede test guides. Denne test blev kørt på en tidlig version af WF.v1 programmet.

Det var en udfordring for udvikler såvel som MiL, at tilgangen til at teste et stykke software var meget forskelligt. F.eks. blev i begyndelsen meldt tilbage, at programmet slet ikke kunne køre på en MiL computer! Det viste sig dog, at testeren aldrig havde installeret et program før og ikke vidste at man skulle trykke ”næste” ved kørsel af ”.msi” filen.

5.3 Kendte fejl

I begge prototyper er der fejl. Den umiddelbart største kendte fejl er, at det er muligt at sætte et materiale til at høre til en gruppe, der ikke eksisterer.
I den videre udvikling er dette første prioritet at få rettet, så man ikke ender med en ugyldig henvisning. Denne ændring skal gennemføres helt ned på databasen hvor relationer imellem tabellerne skal oprettes. Det er i denne version heller ikke muligt helt at slette noget, der er blevet oprettet, f.eks. en person eller en gruppe osv.

5.3.1 WF.v1

- Der bliver åbnet en debug konsol, som bliver åbnet med programmet hver gang
- Der er en knap til at slette al data i stock tabellen som skal fjernes
- Vinduer har ikke korrekt opførsel når de ændrer størrelse
- Højre klik menu punkt ”flyt gruppe” i stock tabellen virker ikke
- Det er muligt at ændre i ”system” brugeren.
- Hvis man lukker programmet og åbner det igen kommer der en SQL fejl på grund af programmet forsøger at indsætte test data ind i databasen som allerede eksistere.

5.3.2 WPF.v1

- Tekst i administratormodulets klapper mangler
- Hvis man åbner vindue til at styre materialer eller personer uden administratorrettigheder, bliver der vist en fejl
- Det er muligt at ændre i ”system” brugeren
- Vinduet til administstion af lån data mangler tekst på knapper
- Ved første visning af order management tab vises der ”test” tekst
- Højre klik menuen i overview virker ikke
- Aktive lån tabellen viser alle lån
- Højre klik menu i aktive lån tabel virker ikke
- Det er muligt at ændre i backupsti uden at være logget ind som admin

- Det er muligt at indlæse en backup af databasen uden at være admin
- Flere vinduer opfører sig ikke korrekt når størrelsen på vinduet bliver ændret
- Det er ikke muligt at ændre en ordre fra ankommet til ikke ankommet i ordremanagement
- I priser sker der floating point errors.
- Hvis man lukker programmet og åbner det igen kan der komme SQL fejl på grund af programmet forsøger at indsætte test data ind i databasen som allerede eksistere.

6

KONKLUSION

6.1 Resultat

Programmet som skal anvendes på festivallen til sommer bliver WPF versionen. Festivallen kan bruge dette til at holde styr på deres materialelager og de udlånsting, der findes på festivallen. Budgetdelen vil helt sikkert give nogle frustrationer, da det er første gang at de forskellige grupper ikke har næsten ubegrænsede penge til indkøb. Dog har kassereren for festivallen udtrykt stor glæde ved netop denne funktion.

Programmet i den nuværende form løser hovedformålet, nemlig at holde styr på de udlån, der findes på festivallen. Gennem det tætte samarbejde med MiL forventer jeg, at de vil blive glade for programmet.
Det er en prototype, og der er mange ting, der kan forbedres. Dog udtrykker MiL, at de er rigtig glade for resultatet.

Jeg glæder mig meget til at arbejde videre med ”Quick Codex”, som programmet er blevet dømt hos MiL - og ser frem til at være med til driften af det på festivallen til sommer.

6.2 Udtalelse fra Musik i Lejet

Musik i Lejet har haft den udsøgte fornøjelse at have Rasmus Vind tilknyttet festivalen med øje for at udvikle et Lagersystem samt styringen af udlån af værktøj, og jeg kan berette at Rasmus har til fulde udført opgaven, han var sågar også velvillig til at ændre systemet undervejs da vi i løbet af processen fik yderligere behov end først antaget.

Musik i Lejet er en Non profit organisation hvilket betyder at vi kører med så få udgifter som muligt og ikke ville være i stand til at kunne købe os til et sådan program som Rasmus har udviklet, desuden er vores organisation struktureret ud fra et frivilligt synspunkt som ikke ses i professionelle firmaer, hvilket også resulterer i at vores produkt er indrettet herefter, og Rasmus har levet fuldt ud til vores forventninger.

Det der har imponeret os mest, er hans evne til at forstå vores behov, da vi ikke er fagmænd inden for dette område, og ikke har været særlig gode til at forklare vores behov.

Rasmus har 100 % stået for udviklingen selv, han har sågar leveret varen til tiden.

Rasmus har selv stået for udarbejdelse af rapporten, vi vurderer at ambitionsniveauet er yderst højt. Vi vurderer at programmet er yderst avanceret, dog utroligt nemt at bruge hvilket er en meget vigtig faktor for os, da vores frivillige bliver skiftende hold hver dag som skal bruge programmet.

Rasmus har en videre indgået en aftale med os, hvor han bliver ansvarshaver for at udføre strømopgaver for festivalen, og har indvilliget i at hjælpe med programmet hvis der skulle opstå problemer med programmet under festivalen.

Vi føler os utrolig privilegeret at have Rasmus med på holdet.

med venlig hilsen
Musik i Lejet.
V/***Christian Vivet***
produktionsansvarlig for tømrer og træ.

Kristian Grauengaard
Festival leder.

6.3 Videre udvikling

Sammen med MiL er der indgået en aftale om at videreudvikle denne prototype frem imod idriftsættelse medio juli. De nævnte fejl i tidligere afsnit skal rettes, og der er stadig flere features der skal implementeres.
Umiddelbart bliver der startet med at implementere punkterne fra indledningen her i en prioriteret liste:

1. Importere person data direkte fra et Excel ark eller lignende datafil fra programmet ”HEAP”, som bruges til at registrere alle, der arbejder på festivallen
2. Importere lagerdata fra Excel ark eller datafil
3. Bruger definerede notifikationer ved lav beholdning
4. Statistik over mest brugte værktøj og materialer
5. Registrering af ind-lejet udstyr, f.eks. gaffeltruck, med returneringstidspunkt og lejeaftale.

Efter dette års festival skal programmet evalueres internt i MiL, og her skal det besluttes om det skal udvikles mere.

7 REFERENCER

7.1 Fodnoter / Kilder

1. www.musikilejet.dk
2. Robust Programming af Matt Bishop. Department of Computer Science. University of California at Davis
3. http://www.wpf-tutorial.com/about-wpf/wpf-vs-winforms/
4. http://www.infragistics.com/community/blogs/devtoolsguy/archive/2015/04/17/windows-presentation-foundation-vs-winforms.aspx
5. http://pragmateek.com/is-wpf-dead-the-present-and-future-of-wpf/
6. http://www.sqlite.org/different.html
7. https://blogs.msdn.microsoft.com/sqlexpress/2011/07/12/introducing-localdb-an-improved-sql-express/
8. https://msdn.microsoft.com/en-us/library/8bw9ksd6.aspx

7.2 Liste over figurer

8

APPENDIKS

8.1 Usecases

8.1.1 Usecase 1

ID:	UC1
Titel:	Frivillig låner et stk. værktøj
Beskrivelse:	En frivillig beder om at låne en el skruemaskine. Han har sit personlige id og værktøjet er på lager.
Primær Aktør:	Ekspedient
Forudsætninger:	Værktøjet er på lager, Frivillig er oprettet I systemet, låne ekspedient tilstede.
Slutbetingelser:	Lånet er registeret med den pågældende frivillige og der er en mindre af det lånte stykke værktøj registreret på lageret.
Hoved senarie:	1. Ekspedient scanner værktøjets stregkode 2. Ekspedient scanner frivilliges stregkode 3. Ekspedient trykker gennemfør lån 4. Systemet registrerer lånet og viser en bekræftelses besked
Udvidelser:	1+2 a: Stregkoden bliver ikke genkendt af systemet ----- a1: En fejl-meddelelse vises på skærmen ----- a2: Ekspedienten søger enden på navn eller id og gennemføre lånet 1+2 b: Frivillig eller værktøj er ikke i systemet ----- b1: Frivillig henvises til ansvarlig for lager og udlån 4 a: Systemet kunne ikke registrer lånet ----- a1: Systemet viser en fejlmeddelelse med forklaring på

	hvorfor lånet ikke kunne registreres
Brugsfrekvens:	Mere end 10 gange om dagen
Status:	Færdig
Prioritet:	1 – Højeste

8.1.2 Usecase 2

ID:	UC2
Titel:	Frivillig afleverer lånt materiale
Beskrivelse:	En frivillig afleveret et tidligere lånt værktøj
Primær Aktør:	Ekspedient
Forudsætninger:	Materialet er registreret som udlånt I systemet
Slutbetingelser:	Udlånet er registreret som afsluttet. Materialet er registerret I lageret.
Hoved senarie:	1. Den frivillige afleverer værktøjet til ekspedienten. Ekspedienten scanner stregkoden. 2. Systemet registrerer at lånet er afsluttet og at materialet nu er på lager igen. Slutter med en bekræftende besked.
Udvidelser:	2 a: Systemet kan ikke fuldfører registreringen. ----- a1: Der vises en fejlmeddelelse så ekspedienten kan rette fejlen, evt. med hjælp fra den lager ansvarlige.
Brugsfrekvens:	Mere end 10 gange om dagen
Status:	Færdig
Prioritet:	1 - Højeste

8.1.3 Usecase 3

ID:	UC3
Titel:	Registrering af frivillig
Beskrivelse:	Ansvarlig for lager og udlån skal oprette en ny frivillig
Primær Aktør:	Ansvarlig for lager og udlån (her efter *Admin*)
Forudsætninger:	-
Slutbetingelser:	Den nye frivillige er oprettet i systemet
Hoved senarie:	1. Admin logger ind med Admin kodeord 2. Admin åbner form til oprettelse/ændring af frivillige 3. Admin indtaster de nødvendige oplysninger og trykker opret 4. systemet viser bekræftelses besked 5. Admin printer nyt lånerkort til den frivillige
Udvidelser:	1 a: Admin indtaster forkert kodeord ----- a1: Admin indtaster rigtigt kodeord ----- a2: Admin sætter kodeord til det oprindelige kodeord 4 a: systemet kan ikke oprette den frivillige og viser en fejl ----- a1: Admin retter fejlen i sin indtastning ----- a2: indtastning er rigtig men den frivilliges gruppe er ikke oprettet i systemet. Admin opretter den frivilliges gruppe og derefter kan den frivillige oprettes. 5 a: Fejl med printer / printer ikke en del af systemet ----- a1: Admin retter printer fejl og prøver igen ----- a2: Admin oplyser den frivillige dennes unikke 5 ciffer id kode som kan bruges i stedet for et lånerkort
Brugsfrekvens:	Meget I starten men burde falde efterhånden som alle frivillige bliver oprettet
Status:	Færdig
Prioritet:	1 - Højeste

8.1.4 Usecase 4

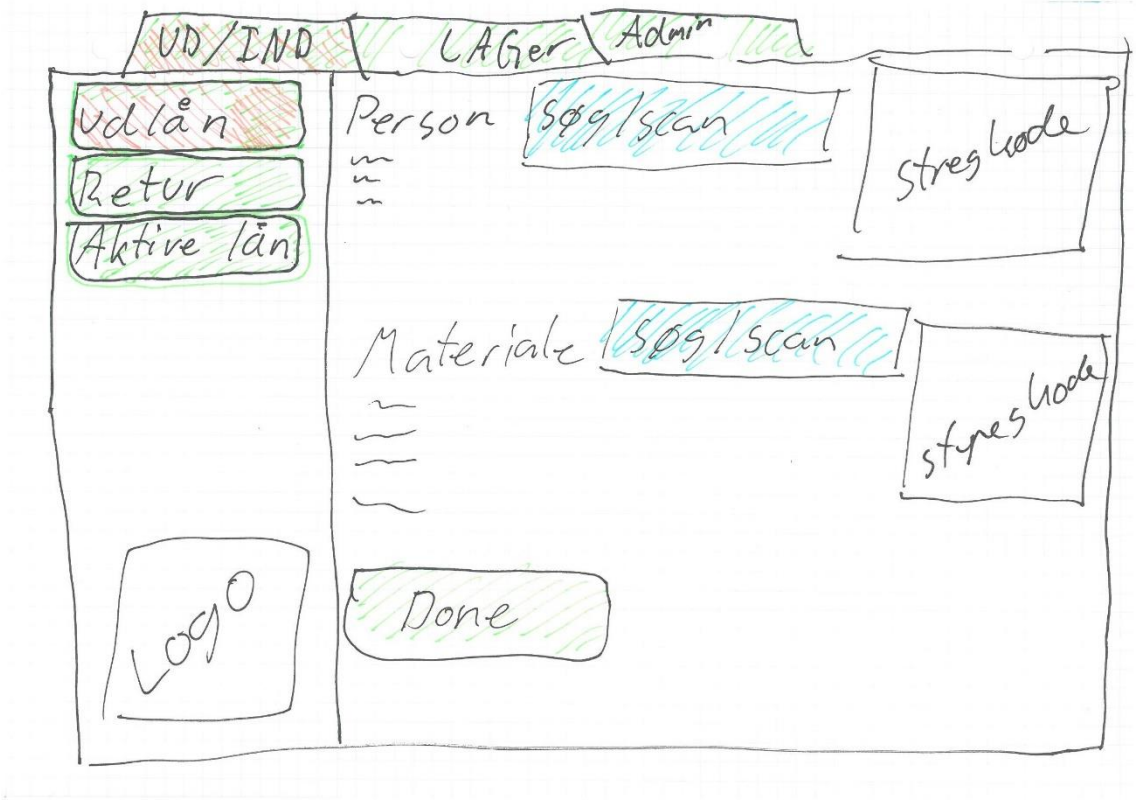
ID:	UC4
Titel:	Bestilling af et stykke materiale
Beskrivelse:	En frivillig beder om at der bliver bestilt mere af et materiale
Primær Aktør:	Ekspedient
Forudsætninger:	Materialet og frivillig findes I systemet.
Slutbetingelser:	Systemet registrerer en åben ordre med det ønskede materiale. Den frivilliges gruppe er blevet trukket materialets pris fra deres buget.
Hoved senarie:	1. Ekspedient finder det ønskede materiale i systemet og åbner formular til at tilføje det til en åben ordre 2. Ekspedient indtaster/scanner de nødvendige oplysninger inklusiv den frivilliges ID og trykker opret 3. Systemet registrerer, at der nu er en åben ordre med den ønskede ting og omkostningen er registreret i gruppens buget. Viser dette i en besked på skærmen.
Udvidelser:	1 a: Der findes ikke en åben ordre ----- a1: Systemet åbner formular til at oprette en ordre og ekspedienten gør dette 3 a: Der vises fejl at den frivilliges gruppe har ikke nok tilbage på deres buget til at bestille den ønskede ting. ----- a1: Den frivillige informeres her om og materialet bliver ikke bestilt ----- a2: Ansvarlig for lager og udlån giver tilladelse til at gruppen overskrider buget, taster sin kode og ordren bliver gennemført
Brugsfrekvens:	Mere end 10 gange om dagen
Status:	Færdig
Prioritet:	2 – Høj

8.1.5 Usecase 5

ID:	UC5
Titel:	Registrering af ankommet ordre
Beskrivelse:	En ordre er ankommet fra en specifik leverandør
Primær Aktør:	Ekspedient
Forudsætninger:	Ordren er registreret i systemet
Slutbetingelser:	De leverede materialer er registreret som ankommet og tilføjet i lageret. Der er ny ordre med same status som den gamle, der har de resterende materialer.
Hoved senarie:	1. Ekspedient finder ordren frem, markerer de ankomne varer og trykker OK 2. Systemet registrer de ankomne vare i lageret og opretter de varer, der endnu ikke ankommet i en ny ordre, der afventer ankomst.
Udvidelser:	1 a: Der er forskel imellem registreret data om varen og den ankomne ----- a1: Systemet opretter en ”ny” vare med de nye data. F.eks. pris eller gruppe. 2 a: Systemet kan ikke registrer varerne som ankommet ----- 2a: Der vises en fejl meddelelse og den oprindelige ordre er uændret.
Brugsfrekvens:	1-5 gange om dagen
Status:	Færdig
Prioritet:	2 - Høj

8.2 Mockup

8.2.1 ”Ud” mockup

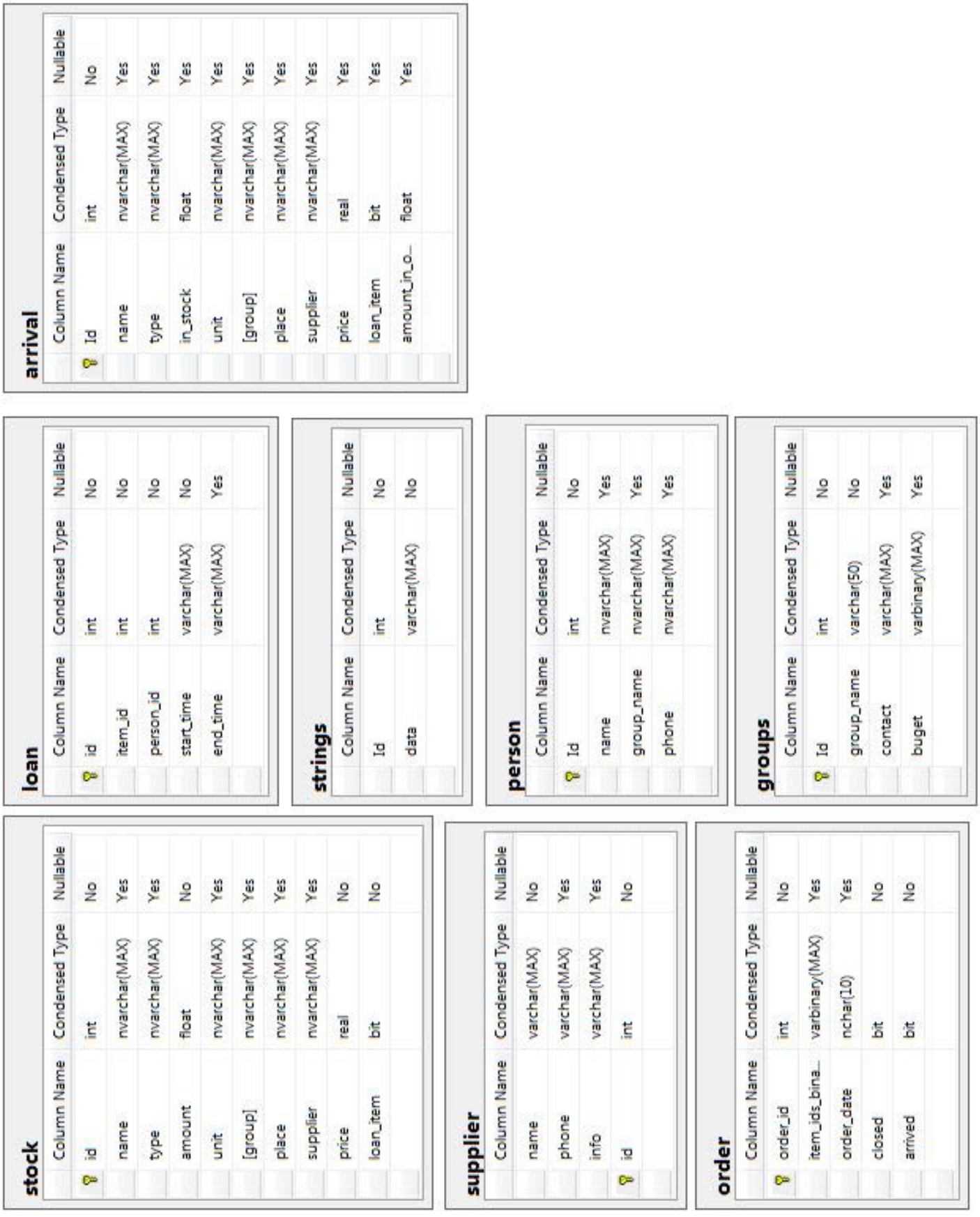


8.2.2 Retur mockup

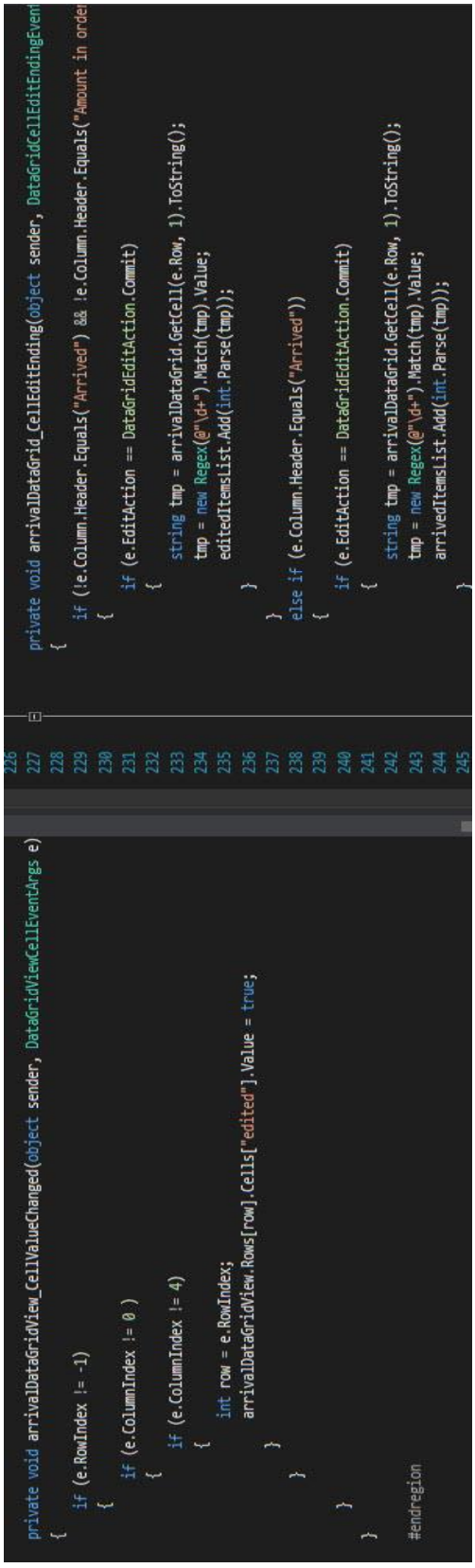
8.2.3 Aktive lån 1 mockup

8.2.4 Aktive lån 2 mockup

8.3 Database diagram



8.4 Sammen ligning af arrival_managment kode



```
private void arrivalDataGridView_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex != -1)
    {
        if (e.ColumnIndex != 0 )
        {
            if (e.ColumnIndex != 4)
            {
                int row = e.RowIndex;
                arrivalDataGridView.Rows[row].Cells["edited"].Value = true;
            }
        }
    }
}
#endregion
```

```
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
```

```
private void arrivalDataGrid_CellEditEnding(object sender, DataGridCellEditEndingEven
{
    if (!e.Column.Header.Equals("Arrived") && !e.Column.Header.Equals("Amount in orde
    {
        if (e.EditAction == DataGridEditAction.Commit)
        {
            string tmp = arrivalDataGrid.GetCell(e.Row, 1).ToString();
            tmp = new Regex(@"\d+").Match(tmp).Value;
            editedItemslist.Add(int.Parse(tmp));
        }
    }
    else if (e.Column.Header.Equals("Arrived"))
    {
        if (e.EditAction == DataGridEditAction.Commit)
        {
            string tmp = arrivalDataGrid.GetCell(e.Row, 1).ToString();
            tmp = new Regex(@"\d+").Match(tmp).Value;
            arrivedItemslist.Add(int.Parse(tmp));
        }
```

Robust software i et ugunstigt miljø,
Lagerstyrings- og bestillings- system til festivallen Musik i Lejet

Forfatter:
Rasmus Raakjær Vind – rasmus@raakjaer.dk

Supervisor(s):
Christian W. Probst – cwpr@dtu.dk

DTU Compute
Richard Petersens Plads
Bygning 324
2800 Kgs. LyngbyDenmark

www.compute.dtu.dk
Tel: (+45) 45 25 30 31
E-mail: compute@compute.dtu.dk