



## CDIO -1

2. Marts 2019

**Medlemmer af gruppe 24:**



Christoffer Adrian Detlef  
(s185117)



Simon Andersen  
(s185083)



Jákup Viljam Dam  
(s185095)



Thaer Mhd Refaat Almalla  
(s170727)



Asama Hayder  
(s185099)

**Underviser:**

Stig Høgh, 02312 - Indledende programmering

**Github link:**

<https://github.com/asamahayder/CDIO-2.semester>

<b>1. Indledning</b>	<b>3</b>
<b>2. Vision</b>	<b>3</b>
<b>3. Krav</b>	<b>3</b>
3.1 Funktionelle krav	4
MoSCoW	4
Most have	4
Should have	4
3.2 Ikke funktionelle krav	4
<b>4. Analyse</b>	<b>5</b>
4.1 Use case diagram	5
4.2 Use case beskrivelse	5
Opret bruger (Casual)	5
Vis brugere (Casual)	5
Opdater bruger (Casual)	5
Slet brugere (Casual)	6
4.3 Domænemodel	6
4.4 Systemsekvensdiagram	6
<b>5. Design</b>	<b>7</b>
5.1 Design klassediagram	7
5.2 Design sekvensdiagram	10
<b>6. Implementering</b>	<b>10</b>
6.1 Kode eksempler	10
<b>7. Test af programmet</b>	<b>12</b>
7.1 Test af arraylist-løsning	12
Positiv test	12
7.2 Test af database-løsning	15
7.3 Test af data fil-løsning	15
<b>8. Konklusion</b>	<b>15</b>
<b>Bilag</b>	<b>15</b>

**!OBS! Grundet tidsmangler har rapporten store mangler! !OBS!**

## 1. Indledning

I dette projekt skal der fremstilles et administrationsprogram hvor det er muligt at oprette bruger, slette bruger, opdatere bruger og vise en liste af bruger. Det forventes ikke at programmet har et krypteret password og vi skal tænke på at dem som sidder med programmet, har adgang til dette. Dette betyder at der ikke skal laves en bruger og password metode for at bruge programmet. Vi vil i denne rapport følge dig igennem de vigtige beslutninger og overvejelser som der er taget i forhold til dette projekt.

Vi har på denne måde formået at fremstille et produkt som opfylder alle de forventninger som der er til programmet, ved at bruge de redskaber som vi har fået lært igennem de forrige projekter.

## 2. Vision

En vigtig operation i softwareudvikling er CRUD-systemet der kreerer, læser, opdaterer og sletter informationer. Dette system bliver brugt alle de steder hvor man skal gemme nogle oplysninger. Dog findes der forskellige metoder til at gemme oplysninger. F.eks. kan man gemme data i en database eller en fil gemt på computeren. Med dette projekt har vi et vision om at skabe et administrativt program der kan alle funktionerne i CRUD princippet hvor strukturen af systemet er bygget op efter trelagsmodellen. Desuden skal der være mulighed for at vælge mellem at gemme data på en database, en fil liggende på computeren samt data der bliver gemt i hukommelsen på computeren.

Ud fra hvad vi har lært i diverse fag har vi høj selvtillid til at skabe en database samt et data gemmende system på hukommelsen på computeren. Der er dog nogle begrænsninger med den løsning der gemmer på en fil i forhold til hvad vi har lært, men vi vil prøve på at lave løsningen så godt vi kan.

## 3. Krav

Til dette projekt er der ikke stillet særlig mange krav, dog er der nogle krav. Det er krav som selve programmet er bygget op på. Dette gør at det er nogle krav som er utrolig vigtige, da det vil betyde at man ikke vil kunne lave programmet, hvis det ikke laves. Til at finde ud af om dette er sandt, med at alle er meget vigtige, kan man bruge MoSCoW. MoSCoW er en meget god måde og finde ud af hvilke funktionelle krav der er vigtige og hvilke der ikke er lige så vigtige.

For de ikke funktionelle krav er der bliver brugt FURPS. Både FURPS og MoSCoW er begge prioriteringsmetoder, det vil sige at de begge er gode til at finde ud af hvilke ting i programmet er vigtigt og hvilke ikke er.

## 3.1 Funktionelle krav

### MoSCoW

#### Most have

- (K1) Programmet **skal** kunne oprette en brugere
- (K2) Programmet **skal** kunne vise brugere
- (K3) Programmet **skal** kunne opdatere en brugere
- (K4) Programmet **skal** kunne slette en brugere
- (K5) Programmet **skal** kunne hænge CRUD sammen med databasen
- (K6) Programmet **skal** kunne oprete, opdatere og slette en bruger i en Arraylist
- (K7) Programmet **skal** gemme alle bruger i en fil på computeren
- (K8) Programmet **skal** deles op i tre lags modellen

#### Should have

- (K9) Programmet **skulle gerne** tillade at et ID kun bliver brugt en gang
- (K10) Programmet **skulle gerne** tillade at et CPR-nummer kun bliver brugt en gang
- (K11) Programmet **skulle gerne** generer et nyt password til nye brugere

## 3.2 Ikke funktionelle krav

**Functional:** Skal kunne fungerer på alle computere, som har IntelliJ

**Usability:** Sprog, TUI, Responstid

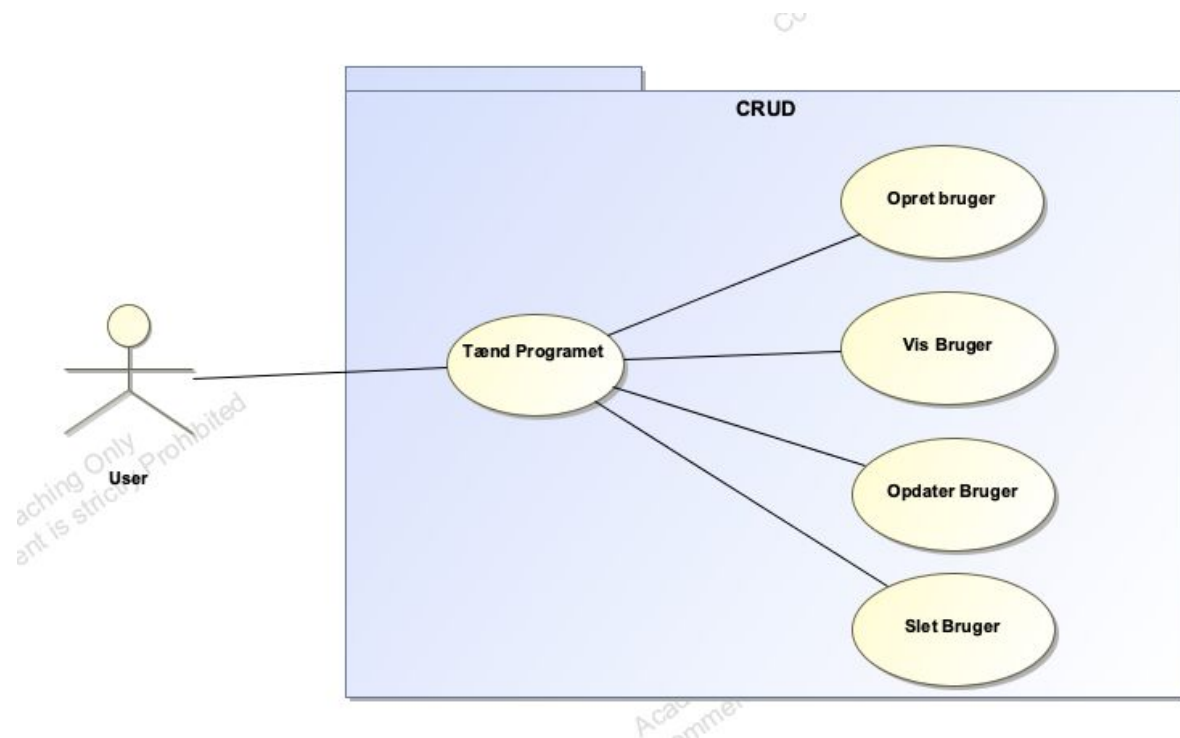
**Reliability:** Lav fejlfrekvens

**Performance:** Skal ikke påvirke computerens ydeevne mere end nødvendigt

**Supportability:** testability

## 4. Analyse

### 4.1 Use case diagram



### 4.2 Use case beskrivelse

#### Opret bruger (*Casual*)

Man skal kunne ud fra TUI'en vælge en mulighed som hedder Opret bruger, hvor man vil kunne oprette en bruger med username, ini, rolle og CPR-nummer.

#### Vis brugere (*Casual*)

Brugere som er gemt i databasen skal kunne hentes frem, og blive vist frem, så man vil kunne se, Id'et, navn, rolle, ini, CPR og Password.

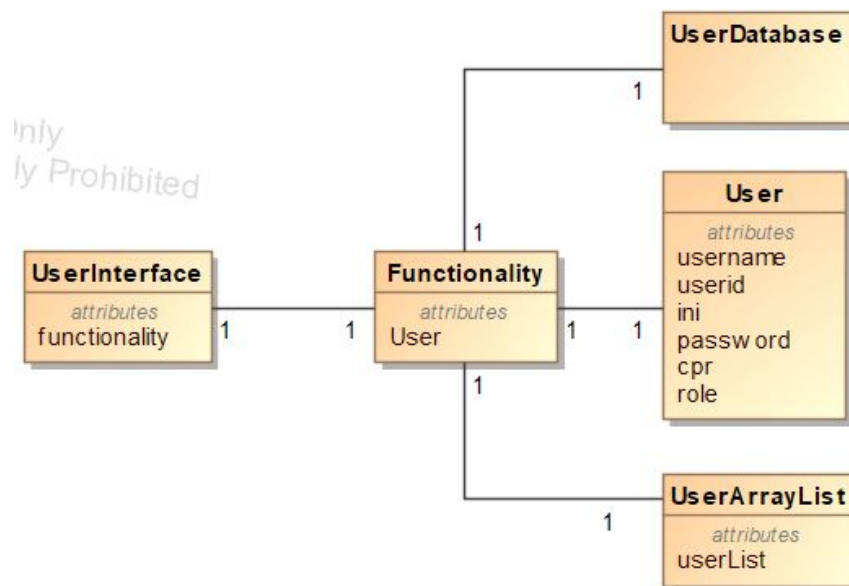
#### Opdater bruger (*Casual*)

Når man skal opdater en bruger, så skal man først vælge hvilket ID man vil ændre informationer for. Efter det skal man så skrive de nye informationer som tilhører det ID.

## Slet brugere (*Casual*)

For at kunne slette en bruger så skal man vælge hvilket ID man gerne vil have slette, og så bliver alle informationer og ID slette fra databasen.

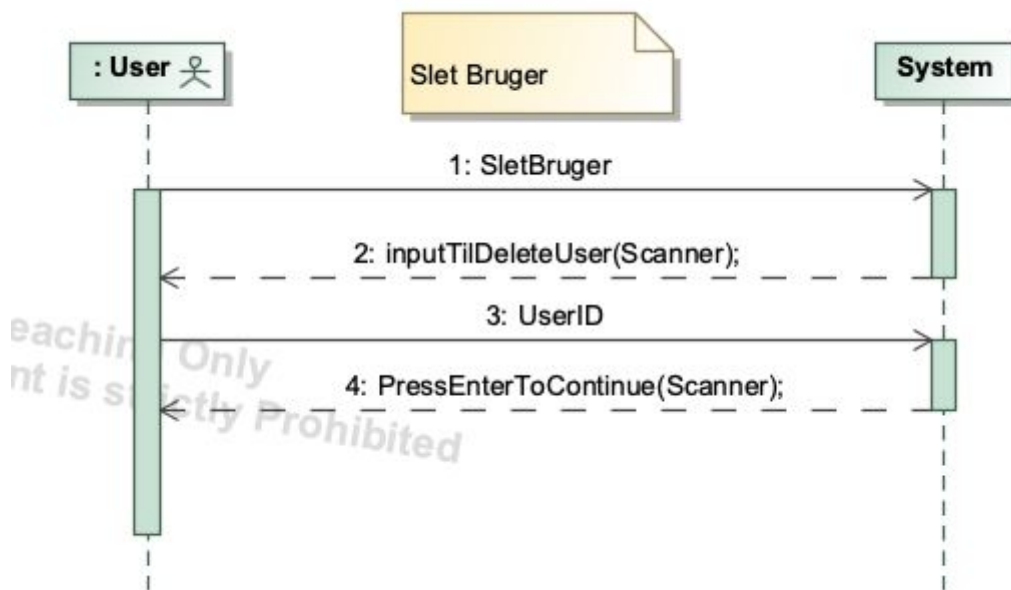
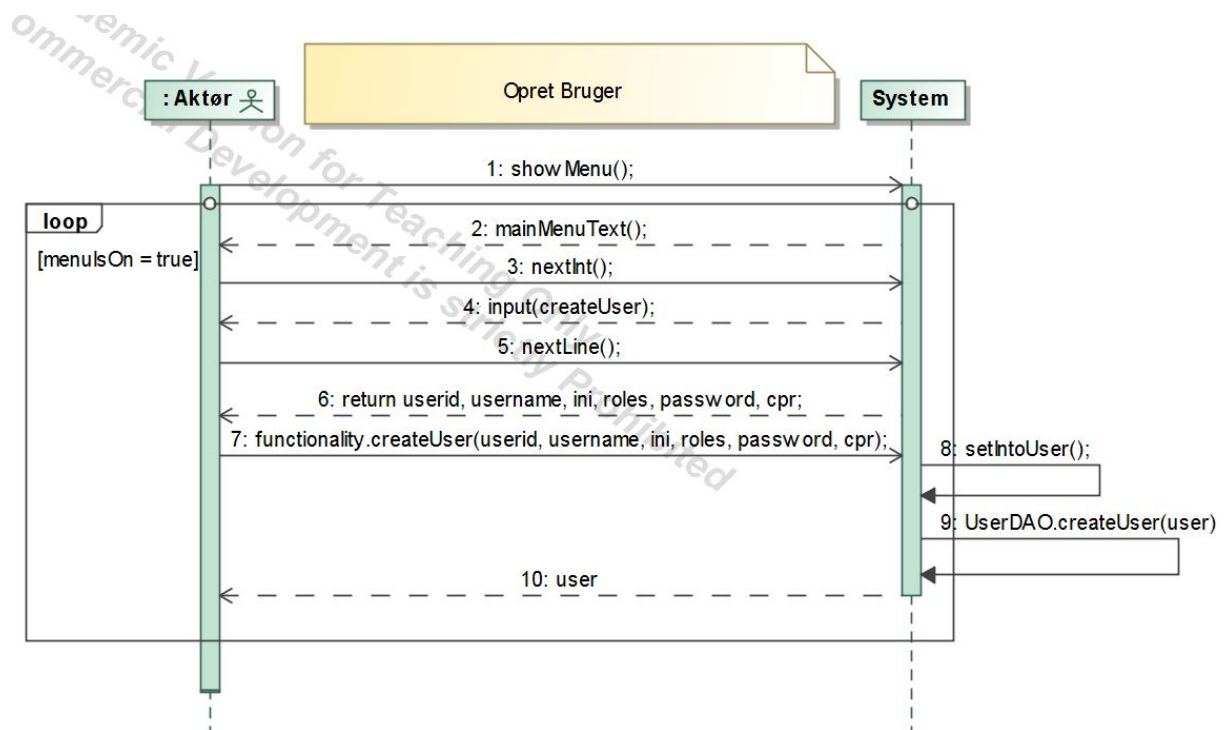
## 4.3 Domænemodel



Måden der laves en domænemodel på, er ved at se ud i den virkelige verden, og på den måde beskrive de klasser som vil være en del af selve systemet. Nu når der bliver arbejdet med et interface, så kan det godt blive lidt svært og forestille sig hvordan det kommer til at se ud. Men vi ved at der er en grænseflade, hvor man kan vælge de forskellige funktioner ud fra. I bagend er der ting som styrer hvad der skal ske når man vælger forskellige ting. Der bliver derfor lavet et par klasser der definerer hvad der kommer til at ske med den data som bliver valgt. DatabaseBruger vil sige at dataene bliver enten hentet eller indtastet ind i databasen, mens ArrayListBruger er klassen som definerer at der laves en liste over dataet mens programmet kører. Dataene kommer fra Bruger klassen, som er den vi opretter/sletter/ændrer i grænsefladen og derefter bliver sat ind i de andre datalag klasser.

## 4.4 Systemsekvensdiagram

Systemsekvensdiagrammet viser den dynamiske interaktion mellem aktøren og systemet hvor der bliver taget hensyn til de forskellige use-cases. Det tages som en black-box, som vil sige at brugeren ved kun de ting, som bliver sagt på skærmen og derefter sker et eller andet i baggrunden.



## 5. Design

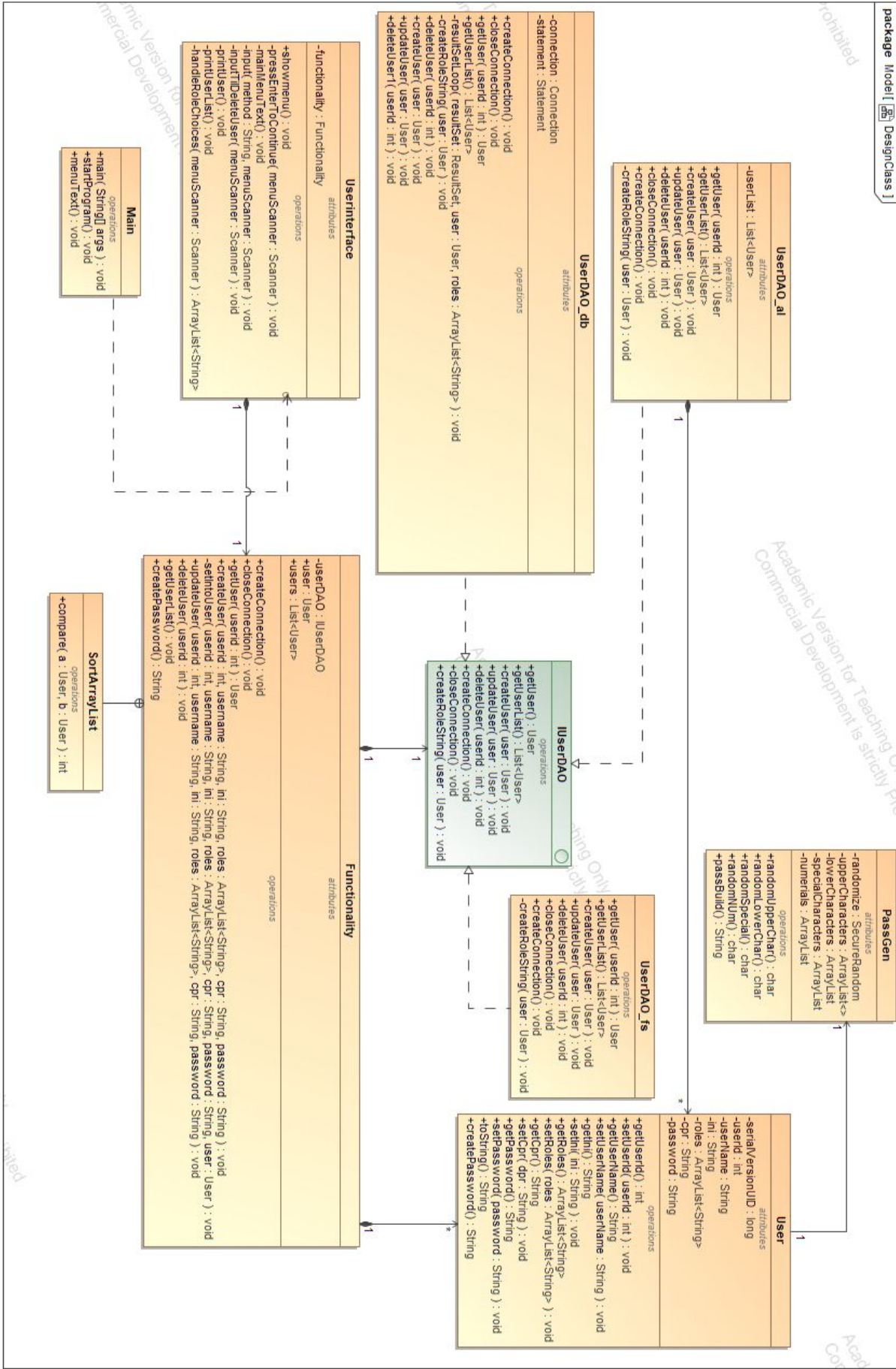
### 5.1 Design klassediagram

Det som design klassediagrammet kan bruges til at vise et statisk billede af hvordan hele systemet hænger sammen. Derudover viser diagrammet også hvilke klasser som bruger attributter eller metoder fra en anden klasse. Dette kan ses mellem de streger som der er mellem de forskellige klasser. Hvis man for eksempel kigger på

klassen Functionality, så er det den klasse sørger for at alle informationer, såsom role, username, osv. Bliver sendt videre til UserDAO\_al, UserDAO\_db eller UserDAO\_fs. Disse tre klasser sørger hver især for at komme brugeren på hver sin måde. Den første gemmer det i et array, den næste i en database og den sidste i en fil på ens computer. Til at kunne frem til at lave, slette eller opdatere en bruger, så er der klassen UserInterface. Den er lavet kun til at vise noget visuelt frem. Den bliver kun brugt til at genere den TUI som der skal bruges, når der bliver valgt en specifik ting.



package Model [  DesignClass ]



## 5.2 Design sekvensdiagram

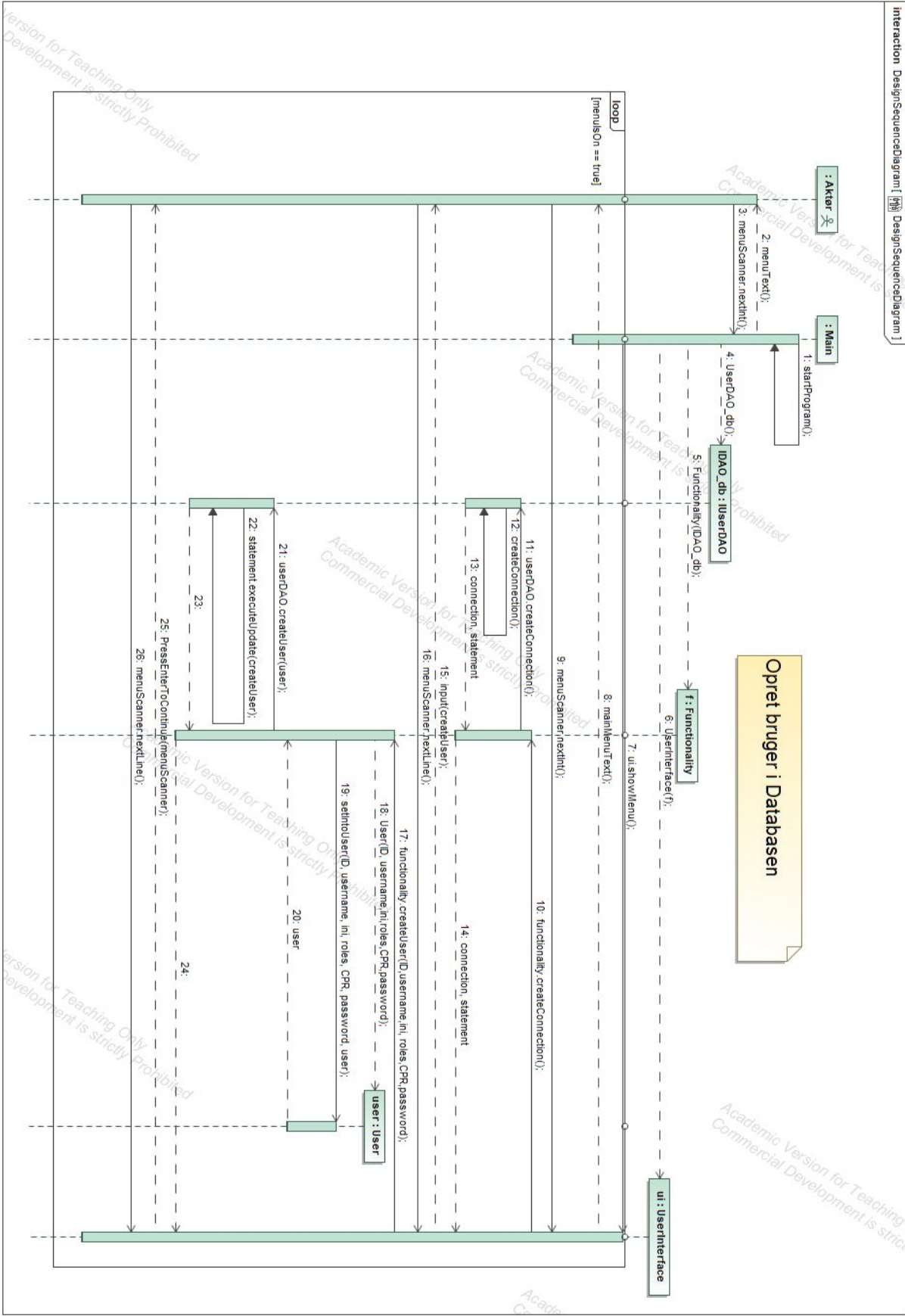
Design sekvensdiagrammet viser aktøren og hans interaktioner med objekterne i systemet. Man kan godt sige at der er ikke så mange interaktioner mellem de to, men vi ser at der sker en hel masse ting i baggrunden bagefter aktøren har begyndt på en aktion.

Diagrammet viser den dynamiske interaktion og hvilke metoder og objekter der bliver påvirket gennem den enkelte use-case. Der blev valgt at se på use-casen, "Opret bruger" hvor der blev også specificeret at brugeren skulle oprettes i databasen og ikke i arraylisten eller filsystemet. Man kunne dog lave et alternativ flow for de andre metoder at gemme data på, men det ville så få diagrammet til at være betydeligt større end hvad det er lige nu. Aktøren har den rolle i dette flow at indtaste data og følge med hvad der sker på skærmen. Systemet arbejder derefter ud fra de data som er inputtet, for at returnere det som aktøren vil gerne have.

Diagrammet viser først og fremst før aktøren overhovedet laver noget, at main klassen kalder på en metode som hedder, startProgram som har brug for et eller andet input at komme videre. Aktøren indtaster det data, som definerer at han vil gemme dataet i en database. Main klassen laver derefter nogle nye objekter, som bliver brugt til at udføre flere metoder til at udføre use-casen.

Enkelt sagt, laver programmet en connection til SQL databasen og derefter har brug for input fra aktøren igen for at indtaste brugerens værdier. Bagefter bliver det sat ind i User objektet som bliver derefter brugt til at hente dataet til at indsætte ind i databasen.

Der slutter så use-casen for "opret bruger"



## 6. Implementering

Projektets formål er at implementere et CRUD-system via trelagsmodellen. Vi har derfor implementeret en grænseflade/TUI hvor man styrer hvad programmet skal gøre. Herfra benytter grænsefladen metodekald fra funktionalitets klassen. Som laver metode kald fra vores 3 forskellige data klasser der henholdsvis indeholder fil-, database- og arraylist lagring af data. Man kan desuden vælge hvordan man vil gemme sin data i starten af programmet. Til sidst returnerer data klassen værdier til funktionaliteten som derved returnerer videre til grænsefladen der outputter dataet.

Et af kravene til opgaven var at gøre så programmet lavede et unikt ID selv til nye brugere som blev oprettet i systemet. Andre restriktioner som kontrollerer brugerinput er heller ikke blevet implementeret. F.eks som brugere navn kun må være mellem 2-20 tegn og initialer der kun må have 2-4 tegn.

### 6.1 Kode eksempler

Når man starter op for programmet, så bliver man mødt af en TUI, som viser en menu. På denne menu har man mulighed for at vælge hvilken måde man gerne vil gemme sine bruger på. Valget af hvilken måde man vil gemme det på, kører igennem et while loop. Koden til dette kan ses på følgende billede:

```
while (menuIsOn) {
    switch (menuTal) {
        case 1:
            IUserDAO IDAO_db = new UserDAO_db();
            f = new Functionality(IDAO_db);
            ui = new UserInterface(f);
            ui.showmenu();
            break;
        case 2:
            IUserDAO IDAO_fs = new UserDAO_fs();
            f = new Functionality(IDAO_fs);
            ui = new UserInterface(f);
            ui.showmenu();
            break;
        case 3:
            IUserDAO IDAO_al = new UserDAO_al();
            f = new Functionality(IDAO_al);
            ui = new UserInterface(f);
            ui.showmenu();
            break;
        case 4:
            menuIsOn = false;
            break;
    }
}
```

Når man så har valgt hvilket format man vil gemme brugerne på, så kommer der en ny menu op, hvor man får nogle nye muligheder. Her har man mulighed for at kunne vise en liste med alle brugerne, vælge en specifik bruger ud fra id, oprette en bruger, slette en bruger ud fra et ID, opdatere en bruger ud fra et ID eller afslutte programmet. Alt koden når man vælger en af dem, kan ses på billedet nedenfor.

```
while (menuIsOn) {
    int menuTal = 0;
    menuTal = menuScanner.nextInt();

    switch (menuTal) {
        case 1:
            System.out.println("——Show User List Menu——");
            functionality.createConnection();
            functionality.getUserList();
            printUserList();
            functionality.closeConnection();
            System.out.println();
            PressEnterToContinue(menuScanner);
            break;
        case 2:
            System.out.println("——Lookup User Menu——");
            System.out.println("Indtast userid som skal findes");
            int id = menuScanner.nextInt();
            functionality.createConnection();
            functionality.getUser(id);
            functionality.closeConnection();
            printUser();
            System.out.println();
            PressEnterToContinue(menuScanner);
            break;
        case 3:
            System.out.println("——Create User Menu——");
            functionality.createConnection();
            input( method: "createUser", menuScanner);
            functionality.closeConnection();
            System.out.println();
            PressEnterToContinue(menuScanner);
            break;
        case 4:
            System.out.println("——Delete User Menu——");
            functionality.createConnection();
            inputTilDeleteUser(menuScanner);
            functionality.closeConnection();
            System.out.println();
            PressEnterToContinue(menuScanner);
            break;
        case 5:
            System.out.println("——Update User Menu——");
            functionality.createConnection();
            input( method: "updateUser", menuScanner);
            functionality.closeConnection();
            System.out.println();
            PressEnterToContinue(menuScanner);
            break;
        case 6:
            System.out.println("——Goodbye!——");
```



## 7. Test af programmet

Hele programmet handler om CRUD systemet som vores test-cases er lavet ud fra og de diverse operationer som det skal kunne.

### 7.1 Test af arraylist-løsning

#### Positiv test

Når man skal teste hvordan systemet gemmer data med arraylists er man nød til at starte programmet op og oprette brugere før man kan begynde at slette, opdatere og hente oplysninger. Man starter med at se user interfacet hvor man kan tilgå alle de forskellige menuer og ud fra brugerinput vælger man hvilken menu man vil tilgå.

```
#####CDIO_del1#####
1: Se listen over databasen      #
2: Hent en specifik bruger      #
3: Opret en bruger              #
4: Slet en bruger               #
5: Opdater en bruger            #
6: Afslut programmet            #
#####
```

Når man vælger opret bruger går man igennem alle trin til at kreere en ny bruger.

```
-----Create User Menu-----
indtast ID:
1
indtast username:
test
indtast Initialer:
test2
vælg roller:
press 1 for 'Master', 2 for 'Student', 3 for 'Admin'
1
add more roles? press y for 'yes' or anything else for 'no'
no
indtast CPR:
1234567890
indtast noget for at generere password
1
Indtast noget for at forsætte
```

Efter dette trin kommer man tilbage til startmenuen hvor vi nu kan vælge at hente den bruger vi lige har lavet frem.

```
----Lookup User Menu----  
Indtast userid som skal findes  
1  
  
UserId: 1  
Username: test  
Initials: test2  
CPR: 1234567890  
Password: 1A==L7vr  
Roles: Master
```

Hvis vi laver flere brugere kan vi også hente hele listen:

```
----Show User List Menu----  
  
UserId: 1  
Username: test  
Initials: test2  
CPR: 1234567890  
Password: 1A==L7vr  
Roles: Master,  
  
UserId: 2  
Username: test2  
Initials: test3  
CPR: 0987654321  
Password: 8l_B8.hJ  
Roles: Student,  
  
UserId: 3  
Username: test3  
Initials: test4  
CPR: 5647382910  
Password: Z-dq-91U  
Roles: Admin,
```

Herefter kan vi slette bruger 2 og få listen igen som viser brugeren er blevet slettet.

```
UserId: 1
Username: test
Initials: test2
CPR: 1234567890
Password: 1A==L7vr
Roles: Master,

UserId: 3
Username: test3
Initials: test4
CPR: 5647382910
Password: Z-dq-91U
Roles: Admin,
```

Man kan også opdatere en bruger ved at vælge brugerens ID:

```
----Update User Menu----
indtast ID:
1
indtast username:
rettelse1
indtast Initialer:
rettelse2
vælg roller:
press 1 for 'Master', 2 for 'Student', 3 for 'Admin'
3
add more roles? press y for 'yes' or anything else for 'no'
n
indtast CPR:
0983475681
indtast noget for at generere password
```



Og når man henter bruger ID 1:

```
----Lookup User Menu----  
Indtast userid som skal findes  
1  
  
UserId: 1  
Username: rettelse1  
Initials: rettelse2  
CPR: 0983475681  
Password: 1J4+j?Yd  
Roles: Admin
```

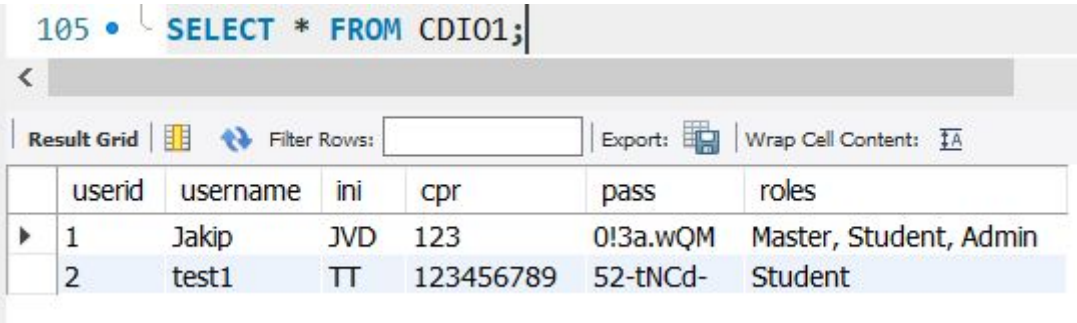
## 7.2 Test af database-løsning

Da både denne klassen som håndtere at tingene bliver gemt på databasen og den som håndtere at tingene bliver gemt i en arrayList, bruger den samme userinterface. Derfor vil dette ikke blive forklaret her igen, man kan læse kapitel 7.1 for det. Hvis man har valgt og skulle lave en ny bruger, så indtaster man bare informationer om brugeren som man bliver bedt om.

```
----Create User Menu----  
indtast ID:  
2  
indtast username:  
test1  
indtast Initialer:  
TT  
vælg roller:  
press 1 for 'Master', 2 for 'Student', 3 for 'Admin'  
2  
add more roles? press y for 'yes' or anything else for 'no'  
n  
indtast CPR:  
123456789  
indtast noget for at generere password  
  
Indtast noget for at forsætte
```

Efter man har indtastede disse informationer, så kan man gå ind på selve databasen, som i dette tilfælde bliver åbnet igennem SQL Workbench. Som man kan

se på billedet nedenfor, så er brugeren med de informationer man gav den i billedet ovenfor.



	userid	username	ini	cpr	pass	roles
▶	1	Jakip	JVD	123	0!3a.wQM	Master, Student, Admin
	2	test1	TT	123456789	52-tNCd-	Student

Når man opdatere en bruger, så skal man først vælge hvilket ID man vil ændre. Efter man har gjort det, så vil man have mulighed for at ændre alle personens informationer. UI'en for når man ændre en bruger kan ses på billedet nedenfor.

```
----Update User Menu----
indtast ID:
2
indtast username:
TestAf2
indtast Initialer:
TA2
vælg roller:
press 1 for 'Master', 2 for 'Student', 3 for 'Admin'
1
add more roles? press y for 'yes' or anything else for 'no'
y
press 1 for 'Master', 2 for 'Student', 3 for 'Admin'
2
add more roles? press y for 'yes' or anything else for 'no'
n
indtast CPR:
1234
indtast noget for at generere password

Indtast noget for at forsætte
```

Som man kan se nedenfor, så er der to billeder. Det første viser hvordan Databasen så ud før brugeren blev ændret, og det andet viser hvordan det ser ud efter en bruger er ændret.

	userid	username	ini	cpr	pass	roles
▶	1	Test1	T1	1234	OB1x!2w=	Master
	2	Test2	T2	1234	.Sbq09!X	Master

	userid	username	ini	cpr	pass	roles
▶	1	Test1	T1	1234	OB1x!2w=	Master
	2	TestAf2	TA2	1234	6r_N!5Ap	Master, Student

Som man kan se, så er alting blevet ændret, lige udover at brugeren stadig har det samme ID. Man vil også have mulighed for kun at ændre en af tingene, eller dem alle. Det eneste der ikke kan ændres er dog selve ID'et.

Hvis der er en bruger som ikke skal være der længere, så kan man også slette den bruger. For at gøre det, så skal man bare indtaste ID'et på den bruger man gerne vil have slettet. Som man kan se på billedet nedenfor, så forsvinder brugeren fra selve databasen.

	userid	username	ini	cpr	pass	roles
▶	1	Test1	T1	1234	OB1x!2w=	Master

Med alt dette kan man se at CRUD-systemet virker som det skal til at gøre de metoder som de skal.

## 7.3 Test af data fil-løsning

Vores test af data fil-løsningen virkede kun på windows computere fordi selve koden kun var lavet til windows. Men koden er lavet sådan så programmet laver en ny mappe på ens skrivebord hvor den efterfølgende lagre hver bruger man kreerer inde i programmet.

## 8. Konklusion

Grundet mangel på tid er rapporten ikke fuldendt. Derudover er der minor bugs i programmet og få funktioner er ikke implementeret i programmet. F.eks. Generer programmet ikke et unikt ID hver gang der bliver lavet en ny bruger. Samt generelt, mangler vi restrictions på de mange input brugeren laver. Dvs. at en bruger sagtens kan indtaste et id som ikke findes når deleteUser() bliver kørt, hvilket vil forårsage en fejl. Altså mangler programmet finpudsning og kvalitetstjek. Men ellers fungerer programmet som forventet.

# Bilag

Kode referencer:

Fra klassen PassGen:

[link 1](#) -

<https://stackoverflow.com/questions/599161/best-way-to-convert-an-arraylist-to-a-string>

[link 2](#) - <https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html>

[link 3](#) - <https://beginnersbook.com/2015/05/java-string-to-arraylist-conversion/>

[link 4](#) - <http://www.asciitable.com/>