

## CDIO del 3

---

Projektopgave forår 2017

Projektnavn: CDIO 3

Gruppe: 23

Afleveringsfrist: Onsdag d. 10. maj 2017

Denne rapport er afleveret via Campusnet (der skrives ikke under).

Denne rapport indeholder 6 sider ekskl. forside og bilag.

---



Christian Niemann, s165220



Mads Pedersen, s165204



Frederik Værnegård, s165234



Viktor Poulsen, s113403

# 1 Timeregnskab

Deltager	Timer
Christian Niemann	12
Frederik Værnegaard	12
Mads Pedersen	7
Viktor Poulsen	16
I alt for alle deltagere	47

Tabel 1: Timeregnskab pr. deltager

Opgave	Timer pr. opgave
Design	8
Impl.	31
Test	0
Dok.	8
Andet	0
Ialt	47

Tabel 2: Timeregnskab pr. opgave

# Indhold

<b>1</b>	<b>Timeregnskab</b>	<b>1</b>
<b>2</b>	<b>Problemformulering</b>	<b>3</b>
<b>3</b>	<b>Design</b>	<b>4</b>
3.1	Sekvensdiagram . . . . .	4
<b>4</b>	<b>Implementering</b>	<b>5</b>
4.1	Frontend . . . . .	5
4.2	Backend . . . . .	6
<b>5</b>	<b>Konklusion</b>	<b>6</b>
<b>6</b>	<b>Bilag</b>	<b>1</b>
6.1	Kildekode . . . . .	1

## 2 Problemformulering

I CDIO delopgave 3 bliver vi bedt om at lave et web interface med tilhørende backend, der skal fungere tilsvarende CDIO delopgave 1. Modulet skal implementeres som en *Single Page Application* med en REST-backend og en HTML/CSS/JavaScript frontend. Web interfacet skal fungere som et bruger-administrationsmodul, som skal kunne tilgås gennem en almindelig webbrowser (evt. via en login-side), og står ubeskyttet. Enhver med adgang til modulet kan oprette enhver type brugere, der omfatter en af de fire følgende; Admin, Pharmacist, Produktionsleder og Laborant. Svarende til rollerne i vores databaseprojekt, kaldet Admin, Pharmacist, Foreman og Operator. Rettighederne for de forskellige brugertyper er endnu underordnet for denne delopgave, men kunne lyde således:

*"Brugerne med rollen Admin har lov til at oprette brugere, se alle brugere, opdatere brugernes data og nulstille deres kodeord samt lukke for brugeres adgang. Alle andre brugere kan rette i deres egne data og kodeord."*<sup>1</sup>

Datalaget for modulet skal som minimum implementeres med et indbygget transient datalag - og kan evt. udbygges med en MySQL database.

---

<sup>1</sup>Udvidet opgave fra den udleverede problemformuleringen

## 3 Design

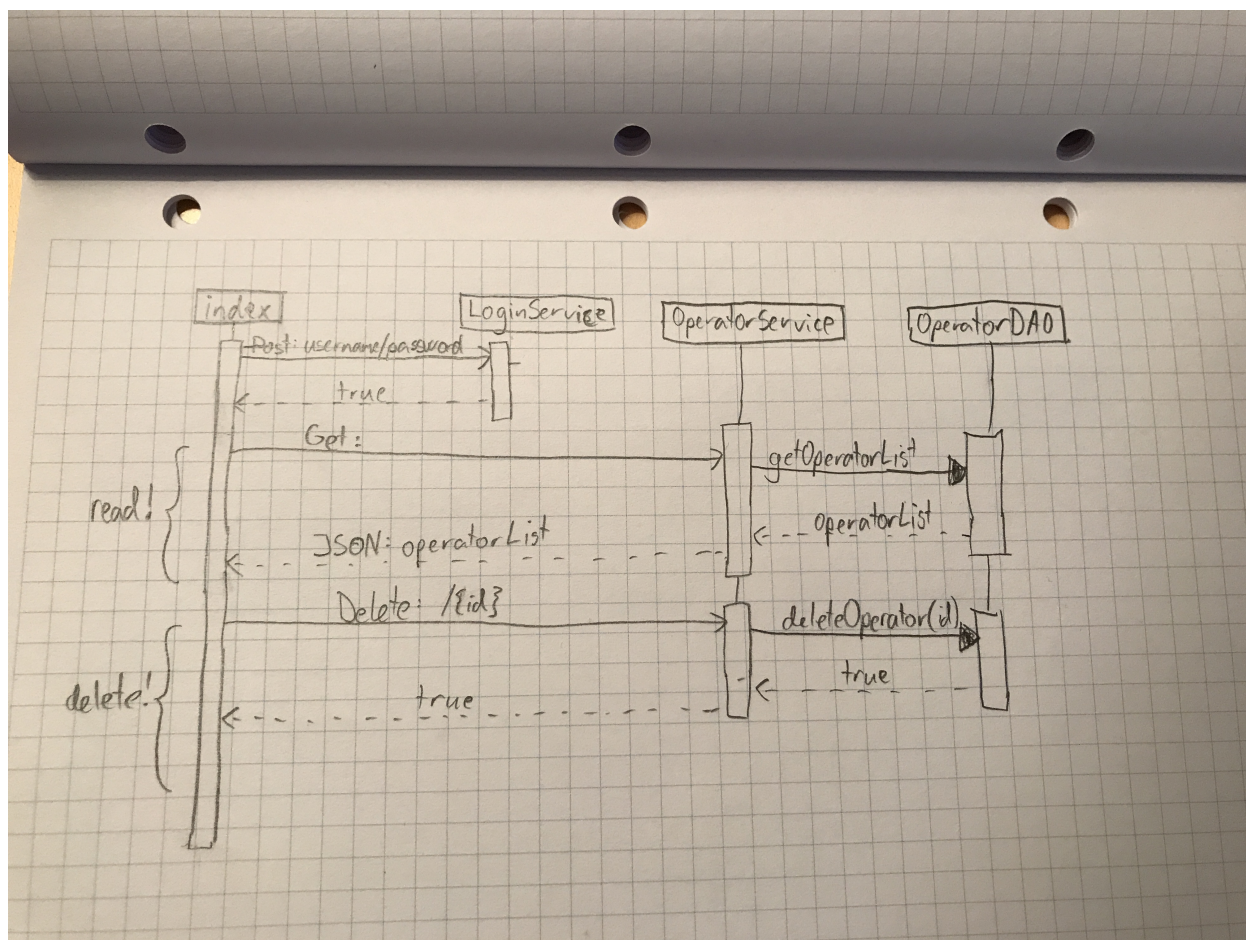
### 3.1 Sekvensdiagram

På Figur 1 ses et sekvensdiagram over vores system. Programmet starter via en browser på vores index side, hvorfra der bliver sendt et Ajax kald af typen 'POST', der indeholder username og password. LoginServicen i denne version af programmet svarer altid med 'true', og index siden loader da en brugeradministrationsside.

På den brugeradministrationsside kan man ved at trykke på forskellige knapper, udføre de fire CRUD funktioner. På sekvensdiagrammet har vi vist hvordan 'read' og 'delete' forløber.

'Read' udføres via en simpel 'GET' anmodning til OperatorService, der kalder metoden getOperatorList i OperatorDAO klassen, og får returneret listen af brugere, listen sendes til web interfacet som JSON objekter.

'Delete' har vi lavet sådan at man sender en 'DELETE' anmodning med det id'et på den bruger der skal slettes som en PathParam. OperatorService kalder da deleteOperator metoden, med det angivne id. Metoden returnerer et 'true' hvis sletningen går som forventet, dette 'true' kan evt. returneres videre op til web interfacet, hvis man vil give brugeren en lille 'alt gik godt' besked.













Figur 1: Sekvensdiagram

## 4 Implementering

### 4.1 Frontend

Til webinterfacet (frontend) har vi valgt at bruge en række frameworks og biblioteker til at øge brugervenligheden på tværs af enheder og skærmstørrelser. Til at give et responsivt design har vi anvendt Materialize og JQuery. Til dynamisk at opdatere data i vores webinterface har vi brugt en javascript-implementering af mustache-templates, som gør at vi kan genanvende layouts flere gange til at præsentere varierende data.

Webinterfacet starter med at kalde API'et på `/api/v1/operator` for at få alle operatører i systemet. Herefter indsætter vi denne data i vores mustache template, som genererer et layout som vist nedenfor.

DTU Gruppe 23						Logout
ID	Name	Initials	CPR	Admin	Role	Action
1	Viktor Poulsen	VP	1234567890	✓	operator	 
2	Frederik	FV	1234567890	—	foreman	 
3	Christian Niemann	CN	1234567890	✓	pharmacist	 
4	Mads Pedersen	MP	1234567890	—	operator	 
						
© 2017 DTU						

Figur 2: Tabeloversigt i Webinterfacet

Herefter har vi de resterende HTTP Methods knyttet op til Redigér (PUT), Slet (DELETE) og Tilføj (POST) samt en dynamisk webform, som kan udfyldes/redigeres for at oprette/ændre brugere i systemet. Hver gang en ajax requests sendes til API'et re-genererer vi tabellen med vores mustache-template.

## 4.2 Backend

Til webinterfacet har vi lavet en tilhørende backend. Backendten indeholder blandt andet vores CRUD funktioner, som kan 'POST', 'GET', 'PUT' og 'DELETE'. Vi har ikke udbygget denne opgave, med vores database, da der vil kunne forekomme problemer, når man skulle bruge hjemmesiden på en anden computer. Vi har derfor en ArrayList, hvori vi har vores brugere, dette er blot navnene på gruppemedlemer. Nedenstående kode viser brugerne, som de står i kildekoden.

```
private static ArrayList<OperatorDTO> operatorList = new
    ArrayList<OperatorDTO>();
    static {
        operatorList.add(new OperatorDTO(1, "Viktor Poulsen", "VP",
            "1234567890", "test", true, "operator"));
        operatorList.add(new OperatorDTO(2, "Frederik", "FV",
            "1234567890", "test", false, "foreman"));
        operatorList.add(new OperatorDTO(3, "Christian Niemann",
            "CN", "1234567890", "test", true, "pharmacist"));
        operatorList.add(new OperatorDTO(4, "Mads Pedersen", "MP",
            "1234567890", "test", false, "operator"));
    }
```

Vores backend kan ydermere validere logins på siden og om information man indtaster på siden, er valid, f.eks CPR-nr . I denne version af programmet vil kildekoden, der validere logins, blot returnere true ligemeget hvad.

## 5 Konklusion

Af denne delopgave har vi stykket et simpelt webinterface (frontend) sammen med vores API (backend), der sørger for kunne administrere de forskellige brugere gennem vores modul. Til webinterfacet gør vi brug af forskellige frameworks, heraf Materialize, JQuery, samt Mustache-templates. Fra vores webinterface kan vi kalde ned til API'et via en AJAX request, der indeholder vores CRUD funktioner - og som tilgår de OperatorDTO objekter vi skal hente op til interfacet, der senere hen vil blive koblet på en database. Vores nuværende modul ligger selvfølgelig op til videreudvikling og udarbejdelse af den sidste opgave, delopgave 4 (CDIO Final), der finder sted i 3-ugers-perioden. Her vil der som allerede nævnt blive koblet en database på, samt udarbejdet yderligere administration for det overordnede system, mht. rettigheder for brugertyper og afvejningssystemet, der blev udviklet i delopgave 2.

## 6 Bilag

### 6.1 Kildekode

Al kildekode samt udviklingshistorik kan gennemgås på <https://github.com/DTU23/23cdio3>