Danmarks Tekniske Universitet

CDIO 3

Diplom Software

4. maj 2018

Gruppe 22

Gruppemedlemmer:

Thomas Mattsson - s175206
Rasmus Gregersen - s175221
Patrick Marcell Madsen - s175209
Søren Kaare Rasmussen - s175202
Dan Sørensen Drachmann- s155384

Tidsregistrering

CDIO_del3							
Time-regnskab	Ver. 2008-09-03						
Dato	Deltager	Design	Impl.	Test	Dok.	Andet	Ialt
30/4-2018	Dan		5				5
-	Rasmus		5				5
-	Patrick		1			4	5
-	Thomas		2			3	5
-	Søren		8				8
1/5-2018	Dan						0
-	Rasmus		2				2
-	Patrick						0
-	Thomas						0
-	Søren				3		3
2/5-2018	Dan		3				3
-	Rasmus		4				4
-	Patrick		2			1	3
-	Thomas		1		2		3
-	Søren				1		1
3/5-2018	Dan						0
-	Rasmus		1		1		2
-	Patrick					0,5	0,5
-	Thomas				1		1
-	Søren	1,5	0,5				2
4/5-2018	Dan						0
-	Rasmus						0
-	Patrick						0
-	Thomas						0
-	Søren				0,5		0,5
	Sum	1,5	34,5	0	8,5	8,5	53

Indholdsfortegnelse

1. Indledning

Til dette projekt har vores kunde bedt os om at lave en webapplikation, der skal kunne oprette brugere. Samt skal webapplikationen sende og hente data fra en backend/REST. Igennem projektet gøres der brug af HTML, jquery, jersey, ajax, json og java til at opnå kundens ønske, om at kunne oprette en bruger fra webapplikationen, der kan snakke sammen med backend systemet.

2. Analyse

2.1 Kravspecifikation

Der ønskes at lave et webapplikation, hvor man kan sende data til og fra REST. Nedenfor ses vores kravspecifikationer:

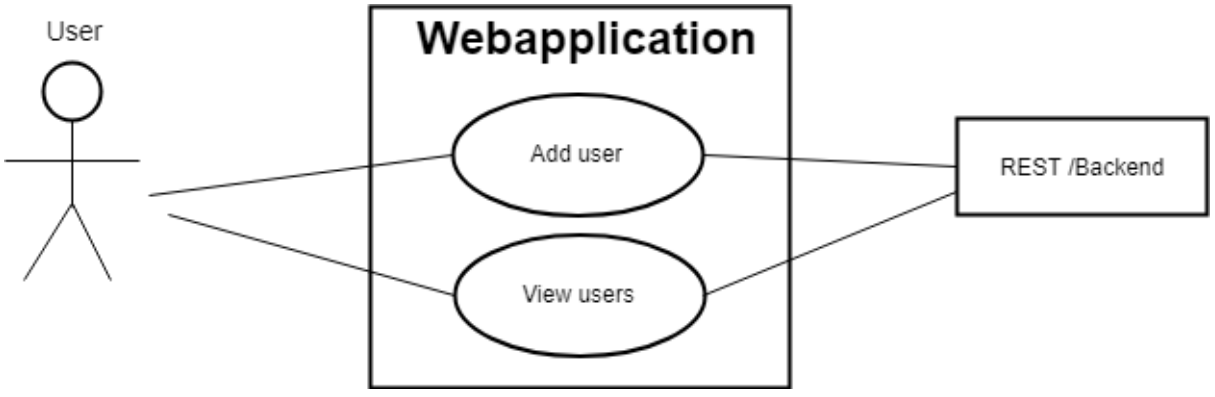
Funktionelle krav

1. Programmet skal kunne oprette en bruger.
2. Programmet skal kunne vise en liste og oprettede brugere.

Ikke-funktionelle krav

1. Webapplikationen giver besked til brugeren, hvilke inputs der mangler før brugeren kan oprettes.
2. Programmet skal køre på en webapplikation sammen med REST.
3. Webapplikationen skal være en single-page applikation.

2.2 Use case diagram



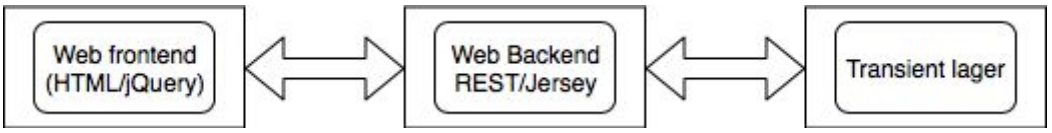
Ovenover er illustreret vores use case diagram over webapplikationen.
Brugeren har kun mulighed for at oprette en/flere brugere og se listen af brugere inde i webapplikationen ved at opfylde nogle felter, hvorefter de inputs, der er blevet indtastet er lavet om til en bruger-objekt inde i REST og kan derefter ses i hjemmesiden.

2.3 Use case beskrivelse - Fully dressed:

Use Case Sections	Comments
Use Case Navn	Add user
Scope	Webapplication
Niveau	User-goal
Primær Aktør	User
Interessenter	Ingen
Forudsætning	Serveren er startet og hjemmesiden er åbnet
Success garanti	Brugeren har oprettet en bruger
Primære Scenarie	**1.** Brugeren indtaster et naturligt tal i ID-feltet. **2.** Brugeren indtaster korrekte oplysninger på resten af felterne. **3.** Programmet samler dataen og laver et objekt ud fra de data, der er indtastet **4.** Den oprettede bruger kan ses på siden, så snart brugeren er oprettet
Alternativ flow	**1a.** Brugeren indtaster alt andet end et naturligt tal **1.** Hjemmesiden vil ikke gå videre før et korrekt id er indtastet **1-2b.** Brugeren lader felterne tomme **1.** Hjemmesiden kræver at alle felter er fyldt korrekt, før der kan oprettes en bruger ***a.** Hvilken som helst tid, hvis systemet crasher eller socket mister forbindelse: **1.** Brugeren bliver ikke skabt. 2. Brugeren skal genlæse siden.
Specielle krav	Webapplikationen giver besked til brugeren, hvilke inputs der mangler før brugeren kan oprettes. Programmet skal køre på en webapplikation sammen med REST. Webapplikationen skal være en single-page applikation.
Frekvens	Indtil brugeren er færdig med at oprette en bruger

2.4 Client-server relation

Dette projekt er en øvelse i at få en frontend, backend og et lager til at snakke sammen. En simpel illustration af dette findes nedenfor.

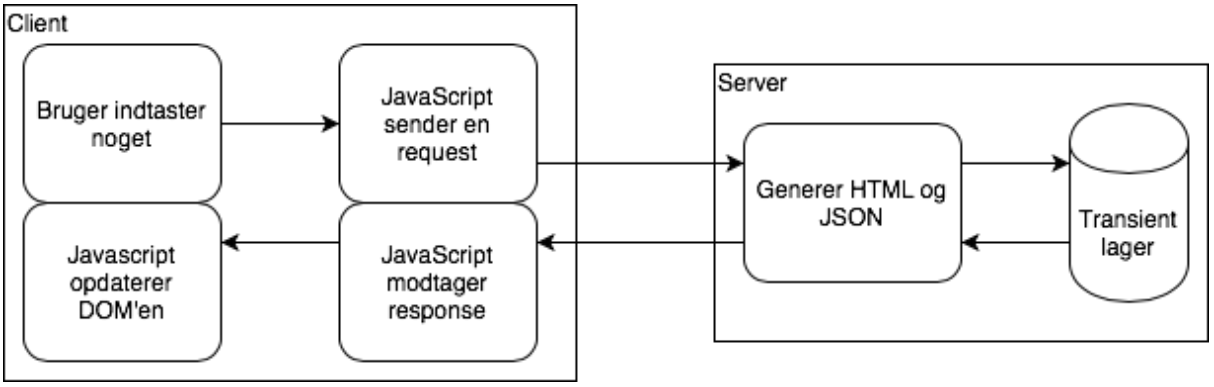


Figur 1[1]

Vi har en bruger som indtaster noget data, denne data bliver sendt med HTML videre til vores REST backend. Vores backend sætter så disse data ind i et transient lager, og den kan tilgå det. Spørger bruger så om andre data, vil backend også kunne få denne data fra lageret. Datalaget kan også udskiftes med et permanent lager, som f.eks. en MySQL database.

2.4.1 Fra request til response

Da vores hjemmeside er en single page application, bruger vi AJAX til at indlæse forskellige dele af hjemmesiden på forskellige tidspunkter. AJAX gør at vi ikke behøver at genindlæse siden for at opdatere information. En illustration af dette kan ses nedenfor.



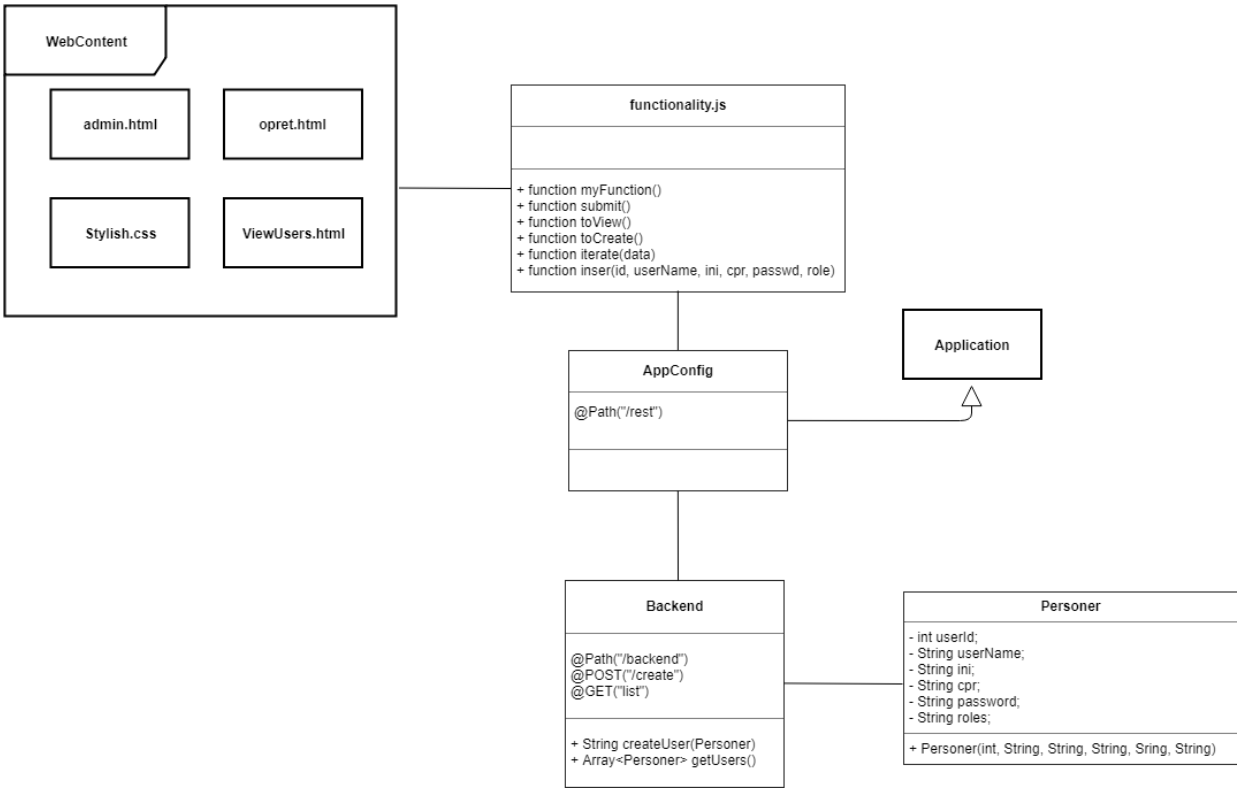
Figur 2[2]

[1] Figur 1 er inspireret af slide nr 2 fra lektion 1 i videregående programmering. https://docs.google.com/presentation/d/1cSBzcJ7Tk0P4xebWRpqDRMi94CY5wP1V1nesQ27G-pg/edit#slide=id.p31

[2] Figur 2 er inspireret af slide nr 3 fra lektion 11 i videregående programmering. https://docs.google.com/presentation/d/1bIecf5258EnfXwzT-MEtq3bGdXU6blWRYrcCjXEoK-A/edit#slide=id.g1eb1c731d1_0_99

3. Design

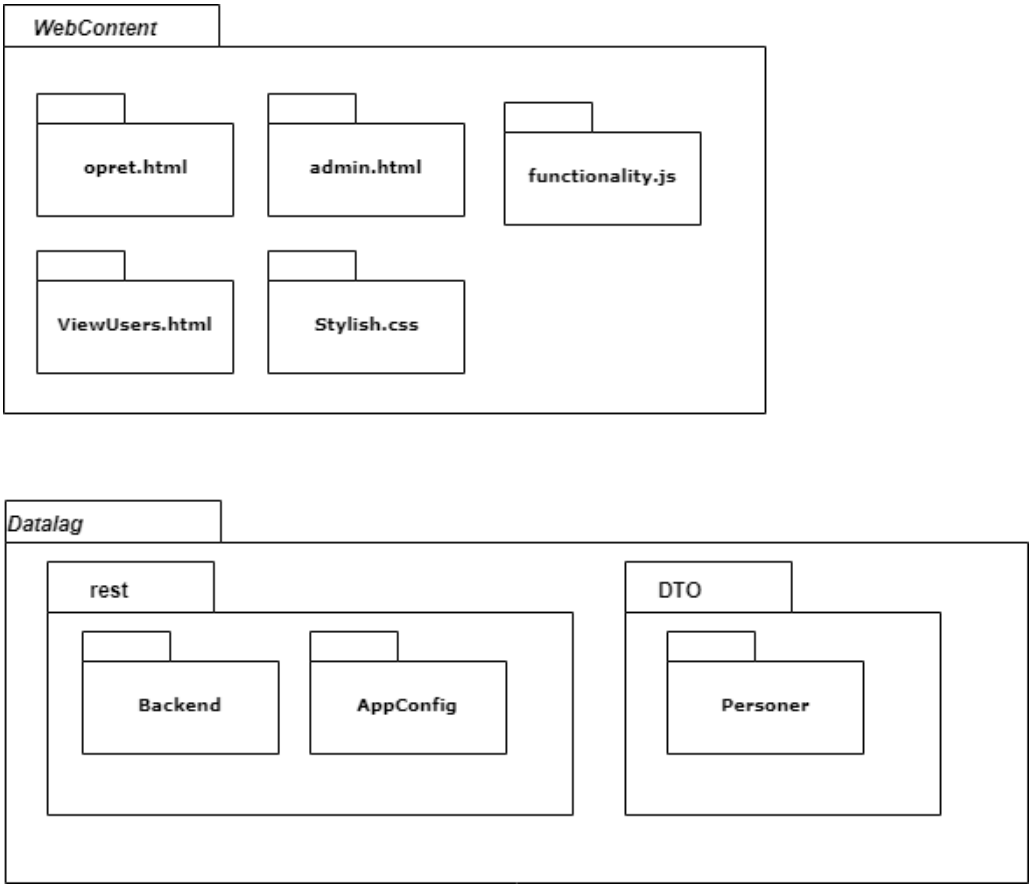
3.1 DCD



Figur 3 “DCD”

Ovenover ses designklassediagrammet over vores webapplication med vores datalag.

3.2 Pakkediagram



Figur 4 “Pakkediagram”

Ovenfor er illustreret pakkediagrammet, der viser to prmær pakker. WebContent indeholder de forskellige frontend filer såsom html, css og javascript. Den anden pakke er datalaget hvor backenden af programmet indeholder.

4. Konklusion

Vi har i dette projekt lavet en webapplikation, som har opfyldt kundens krav. Applikation kommunikerer igennem HTML til vores REST backend, som henter data fra et transient datalager. Da vores MySQL database stadig er under udvikling, valgte vi ikke at benytte den. I 3 ugers vil vi implementere en MySQL database samt give brugermodulet øget funktionalitet.