|  |
| --- |
| Aarhus Universitet |
| Village Smart Grid |
| DAB HandIn 4 |

Tobias Kjær Henriksen auID

Jacob Krutzhals auID

Mikkel Overgaard auID

Indhold

[Introduktion 2](#_Toc500591633)

[Design overvejelser 2](#_Toc500591634)

[Antal databaser 2](#_Toc500591635)

[Type af databaser 2](#_Toc500591636)

[Design af klasser – hvad skal de kunne 2](#_Toc500591637)

[General funktionalitet 2](#_Toc500591638)

[UI eller GUI? 2](#_Toc500591639)

[Implementeret design 2](#_Toc500591640)

[Valgte databaser 2](#_Toc500591641)

[SQL 2](#_Toc500591642)

[NoSQL 2](#_Toc500591643)

[Klasser – hvem gør hvad 2](#_Toc500591644)

[Hvad gør programmet 2](#_Toc500591645)

[Test af design 2](#_Toc500591646)

[Hvordan virker det? 2](#_Toc500591647)

[Diskussion 3](#_Toc500591648)

[Problemer? 3](#_Toc500591649)

[Pros and cons smart grid 3](#_Toc500591650)

[Vores løsning en god løsning 3](#_Toc500591651)

[Konklusion 3](#_Toc500591652)

# Introduktion

I denne opgave skal der udvikles et system til at kunne håndtere et Smart Grid. Et Smart Grid kan anvendes i mindre byer eller mellem lande til at kunne handle strøm med hinanden. Idéen bag en hel by som anvender Smart Grid er at alle indbyggere, virksomheder samt landbrug producer strøm og de forbruger selvfølgelig også strøm, det betyder at alle i byen er Prosumers. En prosumer kan så enten bruge for meget strøm, så bliver prosumeren nød til at købe strøm fra prosumers, hvis prosumeren derimod generere for meget strøm, kan prosumeren sælge dette til andre prosumers. Det som denne opgave fokuserer på er at lave det system, som de forskellige prosumers kan købe og sælge strøm over. System vil indeholde tre databaser, to SQL og en NoSQL database. Hvor diverse info omkring, prosumers, Smart Griddet og priser vil bliver persisteret. Der lægges et REST Api nedover så data kan tilgås ved brug af http kald, og der oprettes et konsol program til at prosumers kan ”logge” på og bytte strøm. Hvis der ikke er nok strøm at købe for griddet og man ikke selv producere nok er det muligt at købe det der mangler fra et National Power Grid. Det samme gælder hvis der er for meget strøm i overskud vil National Power Grid også have mulighed for at købe det ekstra strøm

# Design overvejelser

I dette afsnit vil der blive gennemgået hvilke design overvejelse vi har haft under vejs. Vi vil komme ind på hvorfor vi har valgt det antal databaser, som vi har valgt. Beskrive hvilke typer af databaser og hvorfor vi har overvejet disse til de forskellige funktioner. Hvor vi er kommet op med designet vedrørende klasser, beskrivelse af funktionalitet programmet skal have og om overvejelse om det skulle sættes op som en GUI eller blot en UI.

Antal databaser

Først havde vi overvejet om det var muligt at lave dette system med blot to databaser, da vi bare skulle have en måde at persisterer prosumers og transactions på. Disse to tog dog ikke højde for at prisen på strøm kan ændre sig, for eksempel ved udbud og efterspørgsel. Derfor var det nødvendigt at have en tredje database til at kunne holde styr på priserne og hvilke ombytnings værdier der skal være fra strøm til tokens. Men da vi kun har fået to forskellige databaser givet af universitet vil SmartGridInfo med priser i blot være en tabel i vores SQL database.

Type af databaser

Der skulle bruges to typer af databaser en SQL og en NoSQL, vi har fået givet to af universitetet en SQL database og en Azure Document Database. I dette afsnit skrives om hvilke overvejelser vi har haft vedrørende hvilket ansvar de forskellige databaser skal have og hvorfor at vælge den ene frem for den anden.

### SQL

Det giver mest mening at persistere vores prosumer data, adresse data og info vedrørende priser i SQL databasen. Da vores prosumer og adresse er klart defineret er det mest oplagt at persistere dette data i en SQL. Der er valgt at have prosumer data og adresse data i to forskellige tabeller, hvor disse blot relaterer til hinanden. Der kan sagtens være adresser uden prosumers, men ikke prosumers uden adresser. Priserne er også klart defineret derfor har vi det SQL database. Transaktion data er valgt ikke at persisteret i en SQL da disse kan ændre sig undervejs og hvad man gerne vil have persisteret i den kan også ændres. Det hele vil blive sat op med EnitityFramework og efter Code-First, ved brug af migrations.

### NoSQL

Vi har kun arbejdet med en Azure Document Database i I4DAB faget, så derfor var denne type af NoSQL mest oplagt til dette system. Her har vi valgt at persister vores transactions. Der vil blive oprette en collection for, Power, PendingTransactions og CompletedTransactions. Da disse kan ændre sig var det oplagt at persistere disse i en document database. Vores prosumer og adresse er klart defineret derfor er det ikke nødvendigt at persister dette data i en NoSQL, men det ville også være muligt at have en collection for disse.

Design af klasser – hvad skal de kunne

Der er oprettet forskellige modeller for vores system. Vores prosumers har alle et Id, så det gør det nemt at skelne den fra hinanden, et navn, en adresse, et adresse id og en prosumer type. Prosumer typen bestemmer om en prosumer er en boligejer, landmand eller virksomhed. Vores adresse indeholder blot et Id, vejnavn, vejnummer samt by og postnummer. SmartGridInfo indeholder data såsom ombytningsratioer fra strøm til tokens, prisen fra National Power Grid og et tidspunkt hvornår disse priser er oprettet.

Beskriv tankegang omkring klasser, vis klassediagram

## General funktionalitet

Hvad de skal have ansvar for

## UI, GUI eller web client

# Implementeret design

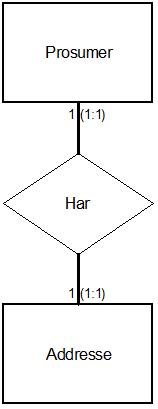
I denne del vil der blive gennemgået hvordan vi har implementeret vores design. Der vil blive gennemgået hvordan de valgte databaser er implementeret, hvordan klasserne er implementeret, hvordan programmet kører med noget essentiel kode og om hvordan vores UI er sat op.

## Valgte databaser

I denne del beskrives hvordan vores SQL og NoSQL databaser er implementeret og realiseret. Der vil blive gennemgået hvad de hver især har til ansvar for.

### SQL

Vores SQL har til ansvar for at persister vores prosumer data, adresse data og SmartGridInfo data. Vores prosumers har en adresse, men vores SmartGridInfo skulle have været en database for sig selv. Men dette er blot en tabel i den database vi har fået udleveret af universitetet. Denne er sat op med Enitity Framework og ved brug af Code-First migrations. Vores prosumer og adresse data er kodet i en configuration.cs i dens seed method. Så hvem der er i vores Smart Grid Village ligger fast. Der er oprettet 33 husstande, 10 virksomheder og 3 landbrug som vores prosumers. Der er også oprettet et SmartGridInfo som bare er basis information. Vores forskellige entities er kun prosumer og adresse i forhold til hinanden, er er lavet følgende ERD for hvordan det er sat op.



Figur , viser ERD for Prosumer og Adresse

Vi har valgt at sætte op så der er en prosumer pr. adresse

Hvad har denne til ansvar for. ERD diagram

### NoSQL

Hvad har denne til ansvar for

## Klasser – hvem gør hvad

Klassediagram forklar det

## Hvad gør programmet

Nogen essensiel kode?

# Test af design

Kort intro

## Hvordan virker det?

Forklar her

# Diskussion

Kort intro

## Problemer?

Aint no problems here son

## Pros and cons smart grid

Hvad er godt og dårligt ved smart grid.

## True Smart Grid Vs Vores

Hvordan er vores anerledes.

## Vores løsning en god løsning

Er det nu også det?

# Konklusion

Hvad har vi lært? NOTHING