|  |
| --- |
| 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.

# 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 2 og nu 3 hvorfor?

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

Overvejelser?

### NoSQL

Azure eneste udleveret

Design af klasser – hvad skal de kunne

Beskriv tankegang omkring klasser, vis klassediagram

## General funktionalitet

Hvad de skal have ansvar for

## UI eller GUI?

Pros cons UI/GUI

# Implementeret design

Kort intro

## Valgte databaser

Kort intro

### SQL

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