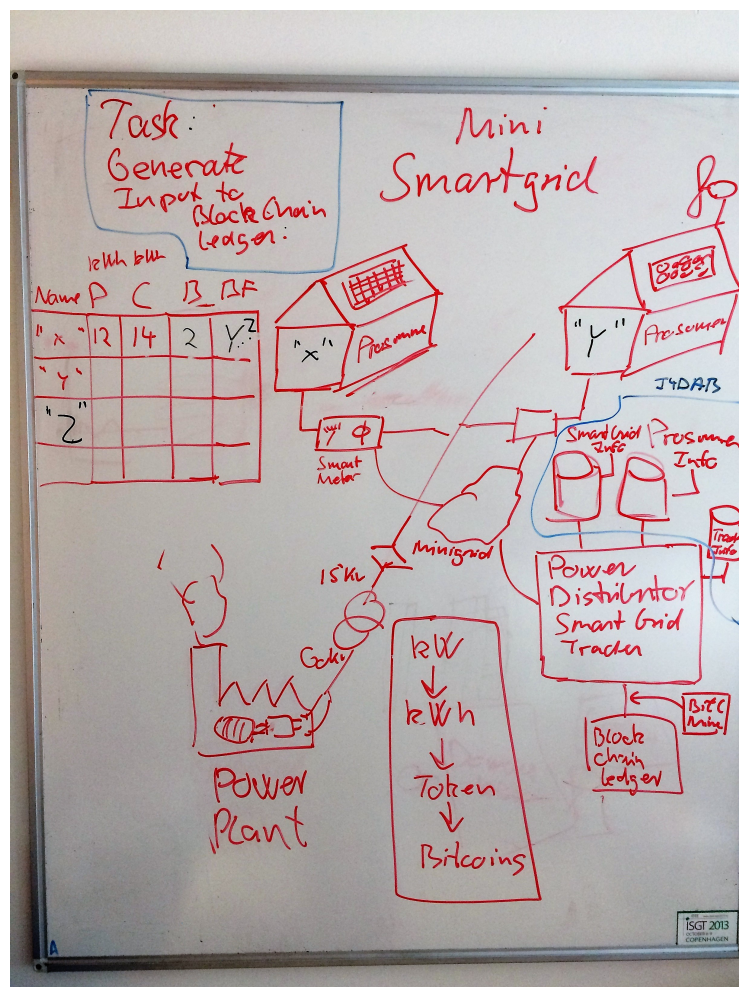


# DAB - Handin 4

## Gruppe 3



Studienummer	Navn	Studieretning
201610450	Rasmus Brask Rolsted	IKT
201606374	Jakob Levisen Kvistgaard	IKT
20119144	Ramtin Asef-Nakhai	IKT
201408675	Parweiz Hagshenas	IKT

**Dato:** 13-12-2018

## Indholdfortegnelse

<b>1 Indledning</b>	<b>3</b>
1.1 Links til Databaser . . . . .	3
1.2 Case . . . . .	3
1.3 Krav . . . . .	3
1.3.1 Problemdomænets definitioner . . . . .	3
1.3.2 Kravspecifikation . . . . .	3
1.3.3 Tekniske Krav . . . . .	4
<b>2 Design</b>	<b>5</b>
2.1 Entity Relationship Diagram (ERD) . . . . .	5
2.2 Database Studio Design (DSD) . . . . .	5
2.3 Domain Drive Design (DDD) . . . . .	5
<b>3 Konklusion</b>	<b>8</b>

# 1 Indledning

## 1.1 Links til Databaser

Vi har benyttet samme MS SQL Database, til begge "databaser", da de alligevel kører uafhængigt af hinanden.

- SQL database:
- Azure DB:

## 1.2 Case

## 1.3 Krav

### 1.3.1 Problemdomænets definitioner

- Databasen Smart Grid Info: indeholder en beskrivelse af konfigurationen af det givne Mini Smart Grid.
- Databasen Prosumer Info: indeholder oplysninger og beskrivelse af de "Prosumers" som er med i den givne Mini Smart Grid.
- Databasen Trader Info: indeholder oplysninger om, hvorledes der er handlet, hvorledes der handles lige her og nu og hvorledes der skal handles fremover. Tabellen i øverste højre hjørne af billedet er en skematisk beskrivelse af nogle af oplysninger i Trader Info DB.
- Smart Meter: En IOT enhed eller mere konkret en enhed, som måler og opsamler de konkrete data omkring strøm, strømproduktion, strømforbrug og status iøvrigt for den pågældende Prosumer.
- Power Distributor/Smart Grid Trader: Systemet som er forsyningsselskabet bag Mini Smart Grid bruges til at overvåge, styre og kontrollere Smart Grid. Trader delen står for at håndtere de indbyrdes salg mellem de enkelte Prosumers.
- Power Plant: symbolisere en typisk central elforsyning uden for Mini Smart Grid.

### 1.3.2 Kravspecifikation

- Udvikles 3 databaser - Trader Info, Prosumer Info og Smart Grid Info.
- De nævnte databaser skal udstyres med Front End REST API, hvor nødvendige metoder afhænger af interaktionen med resten af systemet.
- MiniSmartGrid i denne opgave kaldes 'Village Smart Grid', til 33 husstande og 12 virksomheder/landbrug.
- Der gælder en række karakteristika, som har relevans for MiniSmartGrid i forhold til hver overordnet type husstands-prosumer og virksomhedsprosumer. Disse skal kunne registreres.
- Afregning skal ske på basis af kWh-blokke. Smart Meter måler her og nu strøm og spænding på Prosumers elstik.
- Differensen mellem ind- og udgående blokke aflæses i forhold til en given tidsperiode kaldte afregningsvinduet.
- Der er en "bryder" mellem "The Village Smart Grid" og resten af Danmark.
- Resten af Danmark skal betragtes som en et stort Smart Grid 'The National Smart Grid'.
- Hver Prosumer i Mini Smart Grid er kendt via sin kobberforbindelse (sit elstik), sine karakteristika og sit Smart Meter.
- Afregningsprincip er det princip som Prosumers bruger til at afregne kWh-blokke indbyrdes.

- Der kan inddrages et salgsvindue eller et købsvindue, hvor et salgsvindue er den tid og det tidsrum en prosumer vil stille/stiller en mængde kWh-blokke til rådighed, og hvor et købsvindue er det er den tid og det tidsrum en prosumer vil købe kWh.blokke.
- I et givent afregningsvindue gælder en bestemt pris for en kWh-blok. Hvis priserne for en kWh-blok ønskes dynamiske i forhold til et bestemt elmarked gøres aflæsningsvinduet kortere men ønskes en mere fast afregningsstruktur gøre afregningsvinduet længere.
- En Prosumer har mulighed for selv at bestemme, hvor mange kWh-blokke, der ønskes købt eller solgt, samt hvilke købs- og salgsvinduer der er gældende.
- Alle handler afsluttede, igangværende og kommende kendte handler skal registreres.
- Alle afregninger der er gennemført , under gennemførelse og som er kendt til at ville blive gennemført, skal registreres.

### 1.3.3 Tekniske Krav

- Den tekniske platform der udvikles på er Microsoft .NET. Nyeste gældende versioner.
- Enten .NET Standard eller .NET Core eller et miks af begge.
- ADO.NET Entity Framework, EF eller EF Core, benyttes mod SQL databaser. Nyeste gældende versioner.
- Azure Cosmos DB SQL API .NET benyttes Dokumentdatabaser. Nyeste gældende version.
- Der må gerne udvikles en eller flere testklienter som bruger de udviklede REST API'er.
- Tilsvarende må Swagger Swashbuckle tilføjes REST API servererne for test af REST API.
- Tilsvarende med predefinerede request der sendes via REST klienter som Postman REST Client eller Advanced Rest Client må bruges.
- Der udvikles på SQL Express LocalDB og Azure CosmosDB SQL API Emulator.

## 2 Design

Der er til denne løsning benyttet 1 NoSQL-database og 2 SQL-databaser for at overholde kravet om mindst 1 af hver jf. kravspec.

Gruppen har valgt at lade Traderdatabasen være NoSQL, da dokumentdatabasen blev vurderet som den mest fleksible, idet den ikke er så låst fast som SQL-databaserne.

### 2.1 Entity Relationship Diagram (ERD)

### 2.2 Database Studio Design (DSD)

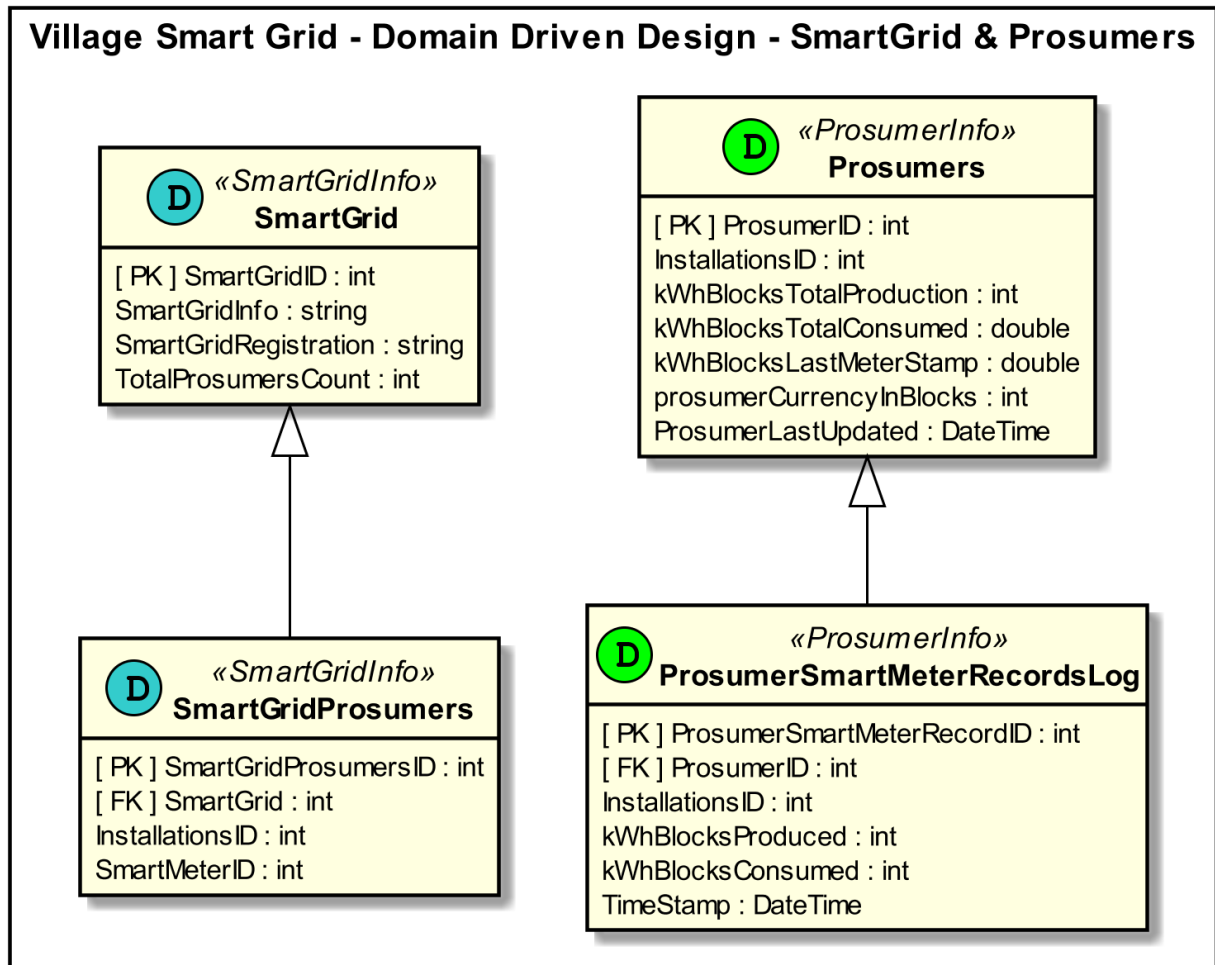
### 2.3 Domain Drive Design (DDD)

I vores DDD har vi lavet 2 diagrammer, hvor der i det ene er vores Smartgrid og Prosumer, mens der i den anden er vores Trader.

I det første diagram, SmartGrid er der 2 klasser, hhv. SmartGrid og SmartGridProsumer. SmartGrid indeholder udover en PK også information om den valgte smartgrid og hvor mange prosumers, der er.

SmartGridProsumers indeholder info omkring prosumers for hvem, der er i dette grid. Blandt andet deres InstallationsID og SmartMeterID.

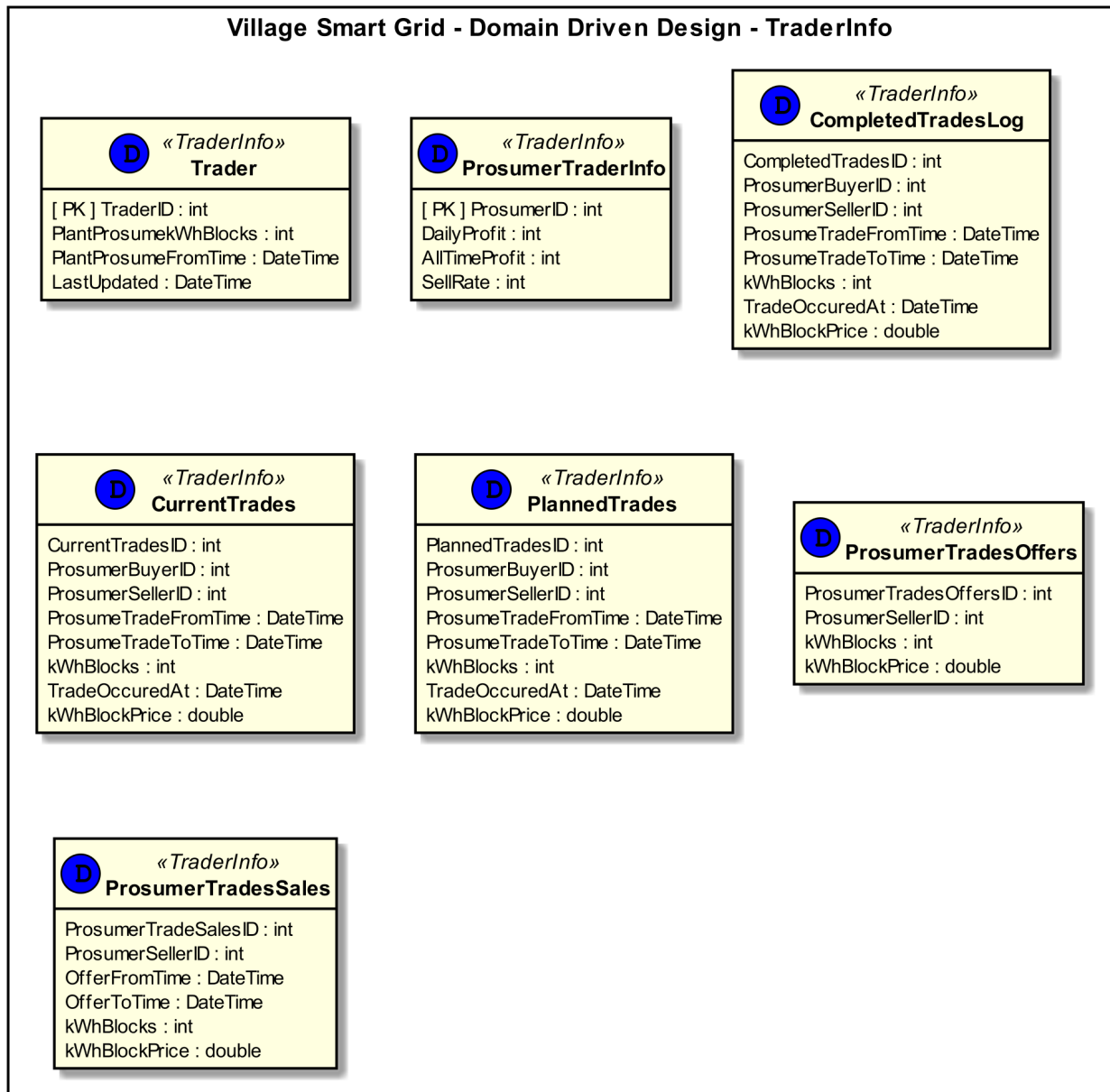
I Prosumers findes ligeledes 2 klasser, hhv. Prosumers og ProsumersSmartMeterRecoresLog. Den første indeholder udover et ProsumerID bl.a. info omkring total produktion og total forbrug af kWh-blokke. Den anden klasse indeholder tidsstemplede elementer med prosumerID for hver gang, der sendes data fra en given elmåler/smartmeter.



Figur 1: Opstilling af DDD-diagram af ProsumerInfo og SmartGridInfo.

I det andet diagram findes der noget flere klasser, da det er en del mere kompleks. Vi vil i det følgende kort beskrive hver klasse:

- Trader: info omkring trader-delen ift. grid, total brug af kWh fra plant og hvis den er negativ viser den forbrug.
- ProsumerTraderInfo: Hver prosumers info omkring totalproduktion, profit og "sellrate".
- CompletedTradesLog: En ny række for hver enkel handel for en given periode med køber, sælger og pris for, der er aftalt.
- CurrentTrades: Igangværende trades, køberID, sælgerID samt antal kWh blokke og pris.
- PlannedTrades: Fremtidige trades, køber- og sælgerID samt stamps fra og til og antal blokke.
- ProsumerTradesSales: Liste over specielle tilbud fra en prosumer, det pågældende tidspunkt, antal blokke, bestemt af en tidsmarkør for hvornår folk kan købe til prisen.
-



Figur 2: Opstilling af DDD-diagram af TraderInfo.

### 3 Konklusion

Som der er gjort bemærket, så er det lykkedes at lave et system bestående af tre databaser:

- En documentDB
- To relationelle databaser

ihybh