



ELVSION

# Afsluttende Eksamensprojekt

UCL | Datamatiker

5. Semester

*Philip V. Nagel | Benjamin Telsing | Christian S. Svinding*

*Product Owner: Peter Lyck Ingerslev*

*Vejleder: Jakob Varring Jensen*

*I Samarbejde med*

**ENERGINET**

## Indholdsfortegnelse

Business Case	1
Formål	1
Problemstilling	1
Problemformulering	1
Forretningsmæssige baggrund	2
Interessenter	2
Scenarie oversigt	3
Valgte-Scenarie: 1-Scenarie	4
Forretningsmodel Lærredet	14
Værditilbud	14
Kundesegmenter	15
Kanaler	16
Kunderelationer	16
Nøgleaktiviteter	17
Nøgleressourcer	17
Nøglepartnere	18
Indtægtsstrømme	19
Omkostningsstrukturer	20
Systemudvikling	22
Metoder	22
Versionsstyring	24
Teknikker og Værktøjer	27
Kravindsamling	31
Use Cases	31
Ekstern Data	36
Funktionelle krav	40
Ikke-Funktionelle krav	42
Design	43
High Level	43
Low Level	47
Sprint Overblik	51
Sprint 1: 14 Okt. - 25 Okt.	51
Sprint 2: 28. Okt - 8. Nov	58
Sprint 3: 11. Nov- 22. Nov	65
Sprint 4: 25. Nov- 6. Dec	69

Sprint 5: 9. Dec- 20. Dec	75
Sprint 6: 23. Dec- 3. Jan	79
Test	81
Unit Tests	81
Performance Tests	83
Brugertest	84
Udrulning	88
Vedligeholdelse	88
Teknisk vedligeholdelse	88
Sikkerheds vedligeholdelse	89
Videre Udvikling	89
Machine Learning	89
Hvad kostede forbruget?	89
Energikilders produktionssted	90
Samarbejde med ældre pleje	90
ChatBot	91
Gamification	91
Opdatering af API key	92
Valg Af Tema	92
Best Practices for modeller	92
Genbrugelige Blazor Komponenter	93
Evaluering	93
Konklusion	94
Bilag	95
Hosted Web Applikation	95
UI First draft	95
Interaktionsdiagram: Visning af Elforbrugs data	96
Manuelt Unit Test Run	97
GitLeaks Analyse test	98
Scrum Board	99
Logging med Application Insights	100
Automatisk Pipeline Test Run Results	101
Branch struktur	102
Første uddrag af Azure Arkitektur	103
Pipeline Run Tests Code	104
Kode Analyse Pipeline	104
Performance test	105

## Ordbog

<b>Elleverandør</b>	En elleverandør er en virksomhed, der sælger elektricitet til forbrugere og virksomheder. Elleverandøren køber strøm på elmarkedet og fakturerer kunden for det faktiske elforbrug samt eventuelle abonnementomkostninger.
<b>Netselskab</b>	Et netselskab er en virksomhed, der ejer og vedligeholder elnettet i et bestemt område. Netselskabet har ansvaret for at transportere strøm via elnettet. De står også for drift, reparation og udbygning af el- infrastrukturen samt for måling af elforbrug på elmålerne. Netselskabet kan ikke vælges af forbrugeren, da det afhænger af hvor man bor.
<b>Transmission System Operator</b>	En organisation, der har ansvaret for at eje, drive og vedligeholde transmissionsnet for energi, som el, gas og brint. Det er kort sagt TSO'en som står for landets "motorveje" for energi.
<b>Energi trilemma</b>	En udfordring i energisektoren, hvor målet er at balancere tre nøgleområder: bæredygtighed, forsyningssikkerhed og økonomisk overkommelighed.
<b>Key Performance Indicator</b>	Key Performance Indicator: Er en måleenhed, der bruges til at evaluere, hvor godt en person, et team eller en organisation opnår specifikke mål.

<b>Forsyningssikkerhed</b>	Et udtryk, der beskriver evnen til at sikre en stabil og pålidelig levering af energi til forbrugere uden afbrydelser, selv under belastning stoppe eller uforudsete hændelser i energisystemet
<b>Klimaaftryk</b>	Et mål for den samlede mængde drivhusgasser, der udledes som følge af en aktivitet, et produkt, en virksomhed eller en persons handlinger, målt i CO <sub>2</sub> -ækvivalenter.
<b>Dashboard</b>	En visuel brugergrænseflade, der præsenterer data og nøgletal på en overskuelig og interaktiv måde.
<b>Webapplikation</b>	En softwareapplikation, der tilgås og bruges via internettet gennem en webbrowser, uden at brugeren behøver at installere eller opdatere programmet lokalt på deres enhed.
<b>Application Programming Interface</b>	Sæt af regler og protokoller, der gør det muligt for forskellige softwareapplikationer at kommunikere og udveksle data med hinanden.
<b>Brugerbaser</b>	Den samlede gruppe af brugere, kunder eller personer, der benytter en bestemt applikation, tjeneste eller platform.
<b>Frontend</b>	Den del af en applikation, som brugeren direkte interagerer med. Frontend omfatter det visuelle og funktionelle, der præsenteres i brugergrænsefladen.
<b>Brugergrænseflade</b>	Det visuelle og funktionelle lag af et system, en applikation eller et website, som brugere interagerer med.

<b>Komponent</b>	En genanvendelig byggeblok i Blazor-applikationer, der består af både brugergrænseflade elementer og logik.
<b>ApexCharts</b>	Et moderne og fleksibelt JavaScript-bibliotek til at oprette interaktive og responsive datavisualiseringer som grafer og diagrammer.
<b>Backend</b>	Den del af en applikation eller et system, der håndterer serverlogik, databehandling og kommunikation mellem frontend og databasen.
<b>Skalerbar</b>	Beskriver en applikations evne til at håndtere vækst eller øgede krav uden at miste ydeevne eller funktionalitet.
<b>Tariffer</b>	Priser eller gebyrer, der opkræves for bestemte ydelser eller produkter, ofte indenfor områder som energi, transport eller telekommunikation.
<b>Afgifter</b>	Beløb, der opkræves af staten på specifikke varer, tjenester eller aktiviteter, ofte for at regulere forbrug eller producere indtægter til staten.
<b>Automatisk skalering</b>	En proces, hvor et system eller en applikation automatisk justerer sine ressourcer for at imødekomme ændringer i belastning eller efterspørgsel.
<b>Logging</b>	Processen med at registrere og gemme information om systemets aktiviteter, fejl, advarsler eller brugerhandlinger i en logfil.

<b>Hosting Service Plan</b>	En specifik aftale eller pakke, der tilbydes af en hosting udbyder, og som definerer de ressourcer og funktioner, en kunde får adgang til, når de hoster en applikation, et website eller en tjeneste.
<b>Brugerengagement</b>	Et mål for, hvor meget og hvordan brugere interagerer med en applikation, et website eller en tjeneste.
<b>Continuous Integration og Continuous Deployment (CI/CD)</b>	En praksis og proces i softwareudvikling, der automatiserer integration, test og implementering af kode for at sikre hurtig og pålidelig levering af software.
<b>Gamification</b>	Brugen af spilelementer og principper som konkurrence, belønning og progression for at gøre oplevelser mere engagerende.
<b>DK1 &amp; DK2</b>	DK1 repræsenterer Vestdanmark og inkluderer Jylland og Fyn.  DK2 repræsenterer Østdanmark og inkluderer Sjælland og Bornholm.
<b>Personal Access Token</b>	Personal Access Token er en sikkerhedsmekanisme, der bruges til at autentificere din identitet og give adgang til en tjeneste eller et API uden at bruge din adgangskode direkte
<b>Azure Pipelines</b>	Azure pipelines bruges til at automatisere bygning, testning og deployment af applikationen.

**Pull Request**

Er en funktion i versionskontrollsystemer som Git, der bruges til at foreslå ændringer i en kodebase



# Business Case

## Formål

Formålet med projektet er at udvikle en applikation, som hjælper med at optimere og informere brugere om deres elforbrug. Applikationen skal samle data om elforbrug, elpriser og leverandørinformation for at give brugerne indsigt i deres forbrugsmønstre, besparelsesmuligheder og hvordan forbrugerens el er produceret. Samtidig skal den muliggøre for brugere at sammenligne leverandører, så det er nemt at få et overblik over deres muligheder.

## Problemstilling

Mange forbrugere har svært ved at forstå og har mangel på overblik til at optimere deres elforbrug, hvilket resulterer i højere omkostninger og spild af ressourcer. Der er ofte mangel på gennemsigtighed i elpriser og leverandørvalg, hvilket gør det udfordrende for forbrugerne at vælge den bedste løsning baseret på deres behov og miljømæssige overvejelser.

## Problemformulering

Hvordan kan forbrugeren få bedre forståelse og overblik over deres elforbrug, og hvordan kan øget gennemsigtighed i elpriser og leverandørvalg hjælpe dem med at træffe informerede beslutninger, reducere omkostninger og minimere ressourcespild.

## Forretningsmæssige baggrund

Energinet er en selvstændig offentlig virksomhed under Klima-, Energi- og Forsyningsministeriet. Energinet ejer og udvikler el- og gasnettet i Danmark for at indføre mere vedvarende energi, opretholde forsyningssikkerhed og sikre lige markedsadgang til nettene.

*Grøn energi for en bedre verden.*

Det er Energinets vision og det er det, de arbejder for hver dag. Energinet som Danmarks TSO<sup>1</sup> ejer, driver og udvikler rygraden i den danske energiforsyning: Transmissionsnettene, som er de overordnede net for el, gas og brint.

Energinet arbejder med at løse energiens trilemma.<sup>2</sup> Det vil sige, at de omstiller energisystemet til vedvarende energi med en høj forsyningssikkerhed<sup>3</sup> og til en pris, der kan betales.

## Interessenter

Peter Lyck Ingerslev fra Energinet, product owner for projektet.

---

<sup>1</sup> Transmission System Operator: En organisation, der har ansvaret for at eje, drive og vedligeholde transmissionsnet for energi, som el, gas og brint.

<sup>2</sup> Energi trilemma: En udfordring i energisektoren, hvor målet er at balancere tre nøgleområder: bæredygtighed, forsyningssikkerhed og økonomisk overkommelighed.

<sup>3</sup> Forsyningssikkerhed: Et udtryk, der beskriver evnen til at sikre en stabil og pålidelig levering af energi til forbrugere uden afbrydelser, selv under belastning stoppe eller uforudsete hændelser i energisystemet

## Scenarie oversigt

### 0-Scenarie

#### Beskrivelse

En hus/boligejer går ind på ElOverblik for at kunne se elforbrug for ønsket periode. Her kan husejeren se sit elforbrug på måneds og årsbasis.

Det er også muligt at få et overblik over ens klimaaftryk<sup>4</sup>, efter at have genereret en rapport.

#### Fordele

- Der er et godt overblik over månedligt og årligt elforbrug og klimaaftryk.
- Det er nemt at tilgå dataen.

#### Ulemper

- Det er ikke tydeligt at det er muligt at se ens forbrug specifikt pr. dag og time.
- Der er ikke mulighed for at sammenligne elleverandører.
- Klimaaftryksrapporten er på en helt anden side end dashboardet<sup>5</sup> og kræver at man genererer en rapport. Hvilket kan tage noget tid.

---

<sup>4</sup> Klimaaftryk: Et mål for den samlede mængde drivhusgasser, der udledes som følge af en aktivitet, et produkt, en virksomhed eller en persons handlinger, målt i CO<sub>2</sub>-ækvivalenter.

<sup>5</sup> Dashboard: En visuel brugergrænseflade, der præsenterer data og nøgletal på en overskuelig og interaktiv måde.

## 1-Scenarie

### Beskrivelse

En hus/boligejer går ind på ElVision, her kan brugeren se deres elforbrug pr. Time, dag, måned eller år. Dagens elpris vil blive præsenteret for brugeren samt et overblik over hvordan deres el er blevet produceret. Brugeren kan også få et overskueligt overblik og sammenligning af forskellige leverandører.

### Fordele

- Mulighed for at tydeligt se elforbrug for time og dag basis
- Der kan fremvises mere data
- Fremvisning af dagens elpriser
- Nemmere vise hvordan brugerens el er produceret.

### Ulemper

- Omkostninger
- Vedligeholdelse

## Valgte-Scenarie: 1-Scenarie

### Formål

At udvikle en webbaseret applikation<sup>6</sup>, der giver brugere adgang til deres strømforbrug, realtidsdata og historiske forbrugsdata ved at bruge forskellige API'er<sup>7</sup> til at hente data. Applikationen skal være tilgængelig som webapplikation og tilbyde en brugervenlig

---

<sup>6</sup> Webapplikation: En softwareapplikation, der tilgås og bruges via internettet gennem en web browser, uden at brugeren behøver at installere eller opdatere programmet lokalt på deres enhed.

<sup>7</sup> Application Programming Interface: Sæt af regler og protokoller, der gør det muligt for forskellige softwareapplikationer at kommunikere og udveksle data med hinanden.

grænseflade til visualisering af data, som hjælper brugerne med at forstå og optimere deres elforbrug.

## Forretningsmæssige løsningsbeskrivelse

ElVision kan positionere sig som et værdifuldt værktøj for hus/boligejere og andre energiforbrugere, der ønsker at træffe bæredygtige valg og reducere deres omkostninger. Ved at tilbyde en platform, der skaber gennemsigtighed i markedet og muliggør energibesparelser, kan ElVision tiltrække en bred brugerbase<sup>8</sup>.

## IT-mæssige løsningsbeskrivelse

### Teknologi Arkitektur

#### 1. Frontend<sup>9</sup>

- **Blazor:** Frontend vil udvikles med Razor components<sup>10</sup> i blazor, som bliver renderet med interactive server render mode, bortset fra Identity endpoints som af sikkerhedsmæssige årsager skal renders med static server side rendering mode i stedet.  
**MudBlazor:** MudBlazor vil bruges til at skabe en moderne og responsiv brugergrænseflade<sup>11</sup>. Komponenterne fra MudBlazor vil være effektivt til at lave en flot ensartet UI.
- **Apex Charts:** ApexCharts<sup>12</sup> vil blive brugt til applikationens datavisualisering med grafer, siden MudBlazors grafer manglede funktionalitet.

---

<sup>8</sup> Brugerbase: Den samlede gruppe af brugere, kunder eller personer, der benytter en bestemt applikation, tjeneste eller platform.

<sup>9</sup> Frontend: Den del af en applikation, som brugeren direkte interagerer med. Frontend omfatter det visuelle og funktionelle, der præsenteres i brugergrænsefladen.

<sup>10</sup> Komponent: En genanvendelig byggeblok i Blazor-applikationer, der består af både brugergrænseflade, elementer og logik.

<sup>11</sup> Brugergrænseflade: Det visuelle og funktionelle lag af et system, en applikation eller et website, som brugere interagerer med.

<sup>12</sup> ApexCharts: Et moderne og fleksibelt JavaScript-bibliotek til at oprette interaktive og responsive datavisualiseringer som grafer og diagrammer.

## 2. Backend<sup>13</sup>

- **.NET 8 ASP.NET Core:** Et ASP.NET Core projekt vil udgøre applikationens kerne og fungere som webserver, der håndterer alle indgående og udgående HTTP-forespørgsler. Det render Razor-komponenter og anvender Entity Framework til at administrere både Identity-databaseopsætning og kommunikation med Azure-databasen. Projektet indeholder også alle nødvendige services, herunder API-kald, datamanipulation, samt integrationer med Azure-tjenester som Key Vault og Application Insights.
- **Azure Automation Accounts:** Skal bruges til at automatisere rutineopgaver ved hjælp af runbooks, hvilket sparer tid og reducerer manuelle fejl. Det tilbyder centraliseret administration af scripts og processer, hvilket gør det nemt at implementere og vedligeholde automatiserede arbejdsflows. Derudover integreres det med andre Azure-tjenester, hvilket giver en skalerbar<sup>14</sup> løsning til at håndtere komplekse opgaver på tværs af miljøer.
- **Azure SQL Database:** Database håndteringen vil ske via Azure SQL Database, hvor vi opbevarer brugerne. Der vil også for hver bruger gemmes et token til ElOverblik API'er, så vi kan hente brugerens energiforbrugsdata.
- **Azure App Service:** Applikationer hostes på Azure App Service for at sikre skalerbarhed og pålidelighed. App Service tilbyder automatiseret skalering og sikkerhedsfunktioner.

---

<sup>13</sup> Backend: Den del af en applikation eller et system, der håndterer serverlogik, databehandling og kommunikation mellem frontend og databasen.

<sup>14</sup> Skalerbar: Beskriver en applikations evne til at håndtere vækst eller øgede krav uden at miste ydeevne eller funktionalitet.

### 3. Opbevaring af Secrets

- **Azure Key Vault:** Key Vault bruges til sikker håndtering af nøgler, API-adgangskoder og andre følsomme oplysninger. Dette sikrer, at applikationen er sikker og overholder gældende sikkerhedsstandarder.

### 4. CI/CD

- **Azure Pipelines:** Azure pipelines bruges til at automatisere bygning, testning og deployment af applikationen. CI/CD-pipelines oprettes for både frontend og backend for at sikre en kontinuerlig og stabil udviklingsproces.

### 5. Data fra ElOverblik

- API'er fra ElOverblik vil blive integreret direkte i backend. Disse data bruges til at hente information om elforbrug, som derefter præsenteres for brugeren via frontend.

### 6. Data fra Energi Data Service

- Energi Data Service er en digital platform udviklet af Energinet, platformen gør det muligt at tilgå data om energiforbrug, produktion og andre energirelaterede aspekter i Danmark. Denne API vil blive brugt til at hente data om elpriser, afgifter, tariffer og information om, hvordan el i Danmark er produceret.

### 7. Udviklingsmiljø

- **Visual Studio:** Udviklingen af applikationen foregår i Visual Studio, hvor alle nødvendige værktøjer og funktioner til både frontend og backend er tilgængelige.

### 8. Versionsstyring

- Git med Azure Repos ville vi bruge til versionsstyring af vores projekt. Vi har valgt Azure Repos, da vi allerede bruger mange andre Azure teknologier.

## Funktionaliteter

1. Brugerlogin: Brugere kan logge ind på appen, så data og information er specifik for brugeren.
2. Dashboardet viser brugerens elforbrug i grafform, sammenligninger over tid og mulighed for at se timeligt, dagligt, månedligt og årligt forbrug.
3. Dashboardet viser også brugerens energikilder og hvordan deres elforbrug er fordelt mellem dem.
4. Sammenligning af priser og mere på forskellige elleverandører.
5. Mulighed for at se elpris for dagen med tariffer<sup>15</sup> og afgifter<sup>16</sup>.

## Udrulning og vedligeholdelse

Applikationen implementeres på Azure med automatisk skalering<sup>17</sup> og overvågning, hvilket gør den nem at vedligeholde. Logging<sup>18</sup> og overvågning vil blive integreret ved hjælp af Azure Monitor og Application Insights for at sikre, at eventuelle fejl kan opdages og rettes hurtigt.

Skulle der blive brug for at skalere applikationen op eller ned, kan Hosting Service planen<sup>19</sup> for applikationen opgraderes eller nedgraderes til at tilpasse appens krav.

---

<sup>15</sup> Tariffer: Priser eller gebyrer, der opkræves for bestemte ydelser eller produkter, ofte indenfor områder som energi, transport eller telekommunikation.

<sup>16</sup> Afgifter: Beløb, der opkræves af staten på specifikke varer, tjenester eller aktiviteter, ofte for at regulere forbrug eller producere indtægter til staten.

<sup>17</sup> Automatisk skalering: En proces, hvor et system eller en applikation automatisk justerer sine ressourcer for at imødekomme ændringer i belastning eller efterspørgsel.

<sup>18</sup> Logging: Processen med at registrere og gemme information om systemets aktiviteter, fejl, advarsler eller brugerhandlinger i en logfil.

<sup>19</sup> Hosting Service Plan: En specifik aftale eller pakke, der tilbydes af en hosting udbyder, og som definerer de ressourcer og funktioner, en kunde får adgang til, når de hoster en applikation, et website eller en tjeneste.



## Forretningsmæssige effekter

### Støtte til den grønne omstilling

Applikationen kan fungere som et værktøj til at fremme bæredygtig adfærd blandt brugerne, hvilket kan positionere virksomheden og samfundet positivt. Dette kan åbne op for muligheder for støtteordninger og tilskud, da projektet bidrager til Danmarks klimamål og opfordrer til brug af vedvarende energikilder.

### Forbedret beslutningsgrundlag for brugerne

Ved at tilbyde data om elforbrug giver ElVision brugerne et solidt grundlag for at træffe informerede beslutninger om deres energiforbrug og valg af leverandør. Dette kan resultere i bedre brugeroplevelser, hvilket igen kan føre til positiv omtale og øget brugerengagement<sup>20</sup>.

## Gevinster

### Reduktion af el omkostninger for brugere

Brugerne får mulighed for at optimere deres elforbrug og skifte til billigere leverandører, hvilket kan føre til lavere elregninger. Dette vil gøre applikationen mere attraktiv og kan øge engagementet og anbefalinger fra tilfredse brugere.

### Fremme den grønne omstilling

Applikationen kan gøre det lettere for brugerne at vælge grønne energikilder ved at synliggøre bæredygtighed. Dette kan bidrage til samfundets generelle mål om at blive grønnere.

---

<sup>20</sup> Brugerengagement: Et mål for, hvor meget og hvordan brugere interagerer med en applikation, et website eller en tjeneste.

## Opfølgning

### Brugerfeedback og iterativ udvikling

Brugernes feedback indsamles løbende for at tilpasse og forbedre funktioner i applikationen. Dette sikrer, at applikationen fortsætter med at levere værdi og forbliver relevant for brugernes behov.

### KPI'er

KPI <sup>21</sup>	Antal aktive brugere
Hvorfor måles?	For at vurdere om applikationen lever op til brugerens behov. Et højt antal aktive brugere ville indikere, at appen er værdifuld for brugerne.
Hvordan måles?	Hvor mange der er logget ind og interageret med applikationen den seneste måned.
Hvem er ansvarlig for målingen?	Energinet.
Forventet Målingsdato	3 måneder efter den er gået i produktion.
Forventet Værdiinterval for måling	500+ aktive brugere inden for de første 3 måneder.
Måling	Det aktive engagement opgøres og præsenteres for ledelsen.
Handlingsplan i fald målingen ligger udenfor	Hvis antallet af aktive brugere er under det forventede niveau, skal der iværksættes en analyse af årsagerne til lav

---

<sup>21</sup> Key Performance Indicator: Er en måleenhed, der bruges til at evaluere, hvor godt en person, et team eller en organisation opnår specifikke mål.

forventet interval	brug. Som løsning hertil kunne der inddrages mere promovering af applikationen.
Ansvarlig for handlingen	Energinet

<b>KPI</b>	<b>Reduceret elforbrug blandt brugere</b>
Hvorfor måles?	For at vurdere, om applikationen bidrager til brugernes bevidsthed om energiforbrug og hjælper dem med at reducere deres elforbrug. Dette er en indikator for applikationens succes og bidrag til den grønne omstilling.
Hvordan måles?	Gennemsnitlig procentvis reduktion i elforbrug pr. bruger over 6 måneder ved at sammenligne brugerens elforbrug før og efter brug af applikationen. Data indsamles via API'er, der henter historiske og aktuelle forbrugsdata.
Hvem er ansvarlig for målingen?	Udviklingsteamet af applikationen, som her sørger for at analysere data.
Forventet Målingsdato	Første måling forventes 6 måneder efter applikationen er lanceret, og derefter hver 6. måned.
Forventet Værdiinterval for måling	10-25% gennemsnitlig reduktion i elforbrug blandt aktive brugere.
Måling	Den gennemsnitlige reduktion opgøres og præsenteres for ledelsen.
Handlingsplan i fald målingen ligger udenfor forventet interval	Hvis den gennemsnitlige reduktion er under 10%, skal der igangsættes en analyse af, hvorfor brugerne ikke ændrer adfærd. Der kan iværksættes forbedringer i

	brugergrænsefladen eller introduktion af yderligere vejledning i appen.
Ansvarlig for handlingen	Udviklingsteamet som der står til ansvar for applikationen og at analysere data'en.

<b>KPI</b>	<b>Antal brugere der skifter til grønnere leverandører</b>
Hvorfor måles?	For at måle effektiviteten af applikationens formål
Hvordan måles?	Kontrollere ændringer i brugerens leverandøraftale
Hvem er ansvarlig for målingen?	Energinet
Forventet Målingsdato	Hver måned
Forventet Værdiinterval for måling	5-10% af aktive brugere
Måling	N/A
Handlingsplan i fald målingen ligger udenfor forventet interval	Evaluer repræsentationen af miljøpåvirkning
Ansvarlig for handlingen	Energinet

KPI	Max Responstid for API-kald
Hvorfor måles?	For at sikre at appen ikke loader for langsomt. For at give en god brugeroplevelse.
Hvordan måles?	Med Azure Application Insights - Performance målinger
Hvem er ansvarlig for målingen?	Udviklerne
Forventet Målingsdato	01/04/2025
Forventet Værdiinterval for måling	1000-1500 ms
Måling	N/A
Handlingsplan i fald målingen ligger udenfor forventet interval	Optimerer håndtering eller revurderer om brug for en anden API, der er hurtigere.
Ansvarlig for handlingen	Udviklerne

# Forretningsmodel Lærredet

## Værditilbud

### At give et bedre indblik om elforbrug

Applikationen giver indsigt i, hvilke tidspunkter på dagen brugerne bruger mest energi, så brugeren på ud fra dette kan optimere forbruget. Brugere får adgang til grafer og visualiseringer, der hjælper dem med at forstå deres energiforbrug bedre, hvilket kan resultere i lavere regninger og en mere bæredygtig livsstil.

### At give et bedre overblik over elpriser fra forskellige leverandører

Applikationen samler og præsenterer elpriser fra en bred vifte af leverandører, hvilket gør det nemt for brugerne at sammenligne priser på tværs af markedet. Dette hjælper brugerne med at træffe informerede beslutninger og vælge den leverandør, der bedst passer til deres behov og budget. Gennemsigtigheden i prissætningen sikrer, at brugerne ikke betaler mere end nødvendigt.

### Daglige elpris

Applikationen giver brugeren indsigt i hvordan elprisen ser ud for dagen. Dette vil informere brugeren om hvornår det er bedst at bruge strøm til fx madlavning, se tv eller opladning af el bil.

## Energikilder

For Energi Data Service er der information tilgængelig om, hvordan el for region DK1<sup>22</sup> og DK2<sup>23</sup> er produceret. Dette kan applikationen hente og præsentere for brugeren relativt til deres forbrug det sidste års tid. Dette giver brugeren en god indsigt i hvordan strømmen er produceret i Danmark og kan hjælpe med at fremme mentaliteten til den grønne omstilling.

## Kundesegmenter

### Primære brugere

Primære brugere, der har ansvar for husholdningsøkonomien og ønsker at optimere deres energiforbrug og omkostninger. Dette segment inkluderer både eneboer og familier, der ønsker at få kontrol over deres elforbrug.

### Virksomheder

Små og mellemstore virksomheder, der ønsker at minimere deres driftsomkostninger og optimere energiforbruget. Virksomhederne kan drage fordel af applikationen ved at identificere besparelsesmuligheder og sammenligne forskellige leverandører for at finde den mest økonomiske løsning.

---

<sup>22</sup> DK1: repræsenterer Vestdanmark og inkluderer Jylland og Fyn

<sup>23</sup> DK2: repræsenterer Østdanmark og inkluderer Sjælland og Bornholm

## Kanaler

### Web app

En tilgængelig platform, hvor brugere kan få adgang til applikationen via en webbrowser. Web App'en giver brugerne mulighed for at analysere deres energiforbrug, sammenligne priser og interagere med applikationens funktioner fra enhver enhed med internetadgang.

## Kunderelationer

### Selvbetjening

Applikationen er designet til at give brugerne mulighed for at administrere deres egen konto, analysere data uden behov for ekstern assistance. Dette skaber en effektiv og brugervenlig oplevelse, hvor brugerne kan finde de oplysninger, de har brug for, når som helst.

### Gratis produkt

Applikationen tilbydes som en gratis tjeneste for at tiltrække brugere og skabe en stor brugerbase. Dette kan omfatte gratis grundlæggende funktioner, mens mere avancerede funktioner og tjenester kan tilbydes som betalte opgraderinger. Det gratis produkt hjælper med at opbygge tillid og relationer til brugerne, hvilket kan føre til konvertering til betalte tjenester i fremtiden.



## Nøgleaktiviteter

### Udvikling af applikationen

Design og opbygning af en brugervenlig applikation, der fungerer på webplatforme. Dette inkluderer udvikling af en intuitiv brugergrænseflade (UI), funktionalitet til datahåndtering og sikkerhed, samt løbende opdateringer for at forbedre ydeevne og funktionalitet. Det tekniske team skal også sikre stabil integration med eksterne datakilder som ELOverblik og Energi Data Service.

### Indsamling af elforbrug, elpriser, elproduktion og leverandørinformation

Opsætning af datastrømme fra forskellige kilder, herunder elpriser, elforbrug, elproduktion og leverandørplysninger. Dette indebærer at indsamle realtidsdata via API'er eller partnerskaber med elleverandører og andre datakilder.

## Nøgleressourcer

### Azure

Microsoft Azure bruges som cloud-platform til hosting af applikationen, datalagring og behandling. Dette giver mulighed for skalerbarhed<sup>24</sup>, fleksibilitet og sikkerhed, hvilket sikrer, at applikationen kan håndtere mange brugere og store mængder data effektivt. Azure tjenester, såsom databaser, analyseværktøjer og machine learning, kan integreres for at optimere brugeroplevelsen og dataanalysen.

---

<sup>24</sup> Skalerbarhed: Evnen for et system, netværk eller applikation til at håndtere øgede belastninger ved at tilføje ressourcer, som f.eks. servere eller kapacitet, uden at forringe ydeevnen.

## Arbejdskraft

Det team, der arbejder på projektet, udgør en vigtig ressource. Den samlede ekspertise inden for udvikling er afgørende for at sikre applikationens kvalitet og succes. Teamets samarbejde og evne til at løse problemstillingen vil være central for projektets fremdrift og effektivitet.

## Energinet

Finansiering fra Energinet kan bruges til at støtte udviklingen og driften af applikationen. Disse midler kan dække omkostninger til teknologi, marketing og udviklere, hvilket sikrer at projektet har de nødvendige ressourcer til at lykkes.

## Nøglepartnere

### Energinet

Den danske transmissionssystem operatør, som er ansvarlig for at sikre en stabil energiforsyning. Partnerskabet kan give adgang til data og ressourcer, samt støtte i at navigere i energimarkedet. Energinet kan også bidrage med legitimitet og anerkendelse i branchen.

### Energi Data Service

Dette partnerskab er vigtigt for at få adgang til pålidelige og opdaterede data, som er nødvendige for at kunne analysere og præsentere information til brugerne i applikationen.

## ElOverblik

En platform, der tilbyder sammenligninger af elpriser. Et samarbejde med ElOverblik kan give adgang til deres eksisterende data og brugermasse, hvilket kan styrke applikationens tilbud og rækkevidde.

## UCL

Samarbejde med uddannelsesinstitution, her inddrages de studerende Philip, Christian og Benjamin i projektet.

## Indtægtsstrømme

### Finansiering fra Energinet

Få støtte fra Energinet til udviklingen og driften af applikationen som et led i deres grønne omstillingsinitiativ. Dette kan være i form af direkte økonomisk støtte, ressourcer eller adgang til data.

### Offentlig tilskud

Undersøg muligheden for at ansøge om offentlige tilskud eller støtteordninger, der fremmer bæredygtighed og energieffektivitet. Dette kan give ekstra midler til drift og udvikling af web app.

## Omkostningsstrukturer

Vores omkostningsstruktur fokuserer på at have en meget skalerbar applikation, der effektivt kan tilpasse sig efter behov. Vi stræber efter at holde vores omkostninger realistiske og i overensstemmelse med praktiske scenarier, samtidig med at vi undgår unødvendige udgifter, hvor det ikke giver mening for et studieprojekt.

Vi har valgt en af de billigste hosting planer for at kunne teste applikationen og komme tættere på at implementere vores CI/CD processer. Denne tilgang giver os praktiske funktioner til udvikling og drift, samtidig med at vi holder omkostningerne lave. Selvom man kunne argumentere, at vi ikke nødvendigvis behøver hosting til alle formål, gør denne løsning det muligt for os at balancere mellem praksis, funktionalitet og omkostninger.

Ressource	Type	Plan	Pris
Elvision-db	Azure Database for MySQL flexible server	Compute Sku: Standard_B1ms (1 vCore) - DKK 83,02  Storage: selected 20 GiB - DKK 15,98/month  Auto scale IOPS (billed on usage in per million requests)	Forventet omkostning: DKK ~190/måned
Elvision-kv	Key Vault	Standard: DKK 0,207 (Per 10,000 operations)	Forventet omkostning: DKK <0,21/måned

Elvisionsa	Storage Account: Block blob storage	Performance: Standard  Storage Account Type: General Purpose V2  Access tier: hot  Redundancy: LRS	Forventet omkostning: DKK ~0,29/mlåned
Elvision-aa	Automation Account		Forventet omkostning: DKK 0/måned
Elvision-ai	Application Insights	Multi-step web tests: DKK 63,19/pr test	Forventet omkostning: DKK 0/måned (Ingen test kørt)
Hosting (ASP-rgstudEnergi Optimering-9a89)	App Service Plan	Plan: Free (F1)  ACU/vCPU: 60 minutes/day compute  Memory: 1GB  Remote Storage: 1GB	Forventet omkostning: DKK 0/måned
<b>Total måneds omkostning:</b>		<b>DKK ~190,50/måned</b>	
<b>Total omkostning:</b> Opdateret 02-01-2025		<b>DKK 840,69</b>	

# Systemudvikling

## Metoder

### Feature Driven Development

Feature Driven Development (FDD) er en agil udviklingsmetode, der fokuserer på at levere funktionalitet i små, håndterbare trin. FDD har været inspirationen til den måde vi har drevet vores projekt, da det kombinerer en struktureret tilgang men stadig er agil.

Vi begyndte med at lave en domænemodel for at opnå en fælles forståelse af den ønskede struktur i systemet. Det viste sig at være en udfordrende proces, da der var mange ukendte faktorer, som vi endnu ikke var sikre på. Samtidig var vi afhængige af en stor mængde ekstern data, der skulle integreres i applikationen. Som resultat havde vi ikke et typisk 'domæne', der kunne modelleres på traditionel vis.

Vi valgte derfor i stedet at fokusere på en liste over features, som systemet skulle have. Denne tilgang har hjulpet os med at nedbryde projektet i mere håndterbare opgaver og hovedfunktionerne Elforbrug, Elpris, Energikilder og Elleverandører kunne så nedbrydes i mindre opgaver og herfra gradvist bygges op.

Features blev planlagt, prioriteret og grupperet i vores Azure DevOps-backlog baseret på deres betydning for projektet. Denne metode gjorde det muligt for os at fokusere på de mest afgørende features først, hvilket sikrede tidligere succeser og styrkede motivationen. Samtidig gjorde denne tilgang det lettere at fordele opgaverne mellem os og ved at fokusere på specifikke og mindre features ad gangen kunne vi løbende præsentere konkrete resultater for vores product owner.

## SCRUM

Vi har valgt SCRUM som vores projektstyringsmetode, fordi den tilbyder en fleksibel og iterativ tilgang til udvikling, der sikrer kontinuerlig fremdrift og tilpasning til ændrede krav. Metoden fokuserer på samarbejde, transparens og løbende forbedringer, hvilket er godt for vores dynamik i teamet og projektets fremgang.

Nedenfor er en beskrivelse af de forskellige elementer i vores metode, som vi har tilpasset projektet og vores egne behov og ønsker.

### Daily Meetings

- Mødes hver dag kl. 8.
- Fokusområder:
  - Hvad skal vi lave i dag?
  - Er der nogen problemstillinger?
- Varighed: Fleksibel, typisk 10-30 minutter.

### Board

- Arbejdet organiseres via *Azure DevOps Boards*<sup>25</sup>.
- Sprog: Alt føres på dansk.
- Kolonner:
  - **Backlog:** Indeholder features og user stories.
  - **To Do:** Opgaver klar til at blive påbegyndt.
  - **In Progress:** Opgaver under arbejde.
  - **Review:** Opgaver klar til gennemgang.
  - **Done:** Færdige opgaver.

---

<sup>25</sup> Bilag: SCRUM Board

## Sprint Planning Meetings

- Planlægning af næste sprint.
- Afholdes den sidste dag i det aktuelle sprint.

## Sprints

- Varighed: 2 uger.
- Starter hver mandag og slutter fredag i uge 2.

## Sprint Retrospective Meetings

- Afholdes som en del af Sprint Planning-mødet.
- Fokus på at evaluere:
  - Hvad gik godt?
  - Hvad kunne gøres bedre?

## Rapportskrivning

- Udarbejdes løbende hver fredag.

## Versionsstyring

Versionsstyring er en essentiel del af vores udviklingsproces, da det sikrer samarbejde, sporbarhed og kontrol over kodebasen. Ved at implementere en klar branch-struktur og strenge regler for pull requests<sup>26</sup> har vi skabt en robust proces, der minimerer risikoen for fejl og forbedrer kvaliteten af vores software.

---

<sup>26</sup> Pull Request: er en funktion i versionskontrollsystemer som Git, der bruges til at foreslå ændringer i en kodebase



## Branch Struktur

For vores projekt har vi etableret en konvention og proces for vores Git branches, hvilket har sikret en struktureret og sikker branch-håndtering<sup>27</sup>. Dette har hjulpet os med at opretholde konsistens og beskyttelse af både development og main branchen mod uhensigtsmæssige pull requests. Alle ændringer i development og main branches kræver pull requests, hvor kun development må merges ind i main, hvilket kontrolleres gennem en pipeline. Derudover er der krav om en reviewer på alle pull requests for at sikre, at ændringer bliver gennemgået af en anden udvikler inden integration.

### Main-branchen:

Kan kun ændres via en pull request fra development-branchen, som er sikret via en pipeline.

### Development-branchen:

Kan kun ændres via en pull request.

### Feature-branches: ★

feature/branch-navn: Bruges til at tilføje nye funktioner eller funktionaliteter.

Eksempler:

- feature/user-authentication
- feature/payment-integration

### Bugfix-branches: 🐛

bugfix/branch-navn: Bruges for at rette fejl eller problemer.

Eksempler:

- bugfix/database-error
- bugfix/ui-styling-issue

---

<sup>27</sup> Bilag: Branch Struktur

**Hotfix-branches:** 🔥

hotfix/gren-navn: Bruges til kritiske rettelser, der skal deployeres hurtigt.

Eksempler:

- hotfix/security-issue
- hotfix/crash-fix

**Dokumentation-branches:** 📄

doc/dokumentations-emne: Bruges til ændringer i dokumentationen.

Eksempler:

- doc/update-readme
- doc/api-documentation

**Design-branches:** 📄

design/dokumentations-emne: Bruges til ændringer i designet af hjemmesiden.

Eksempler:

- design/dashboard-beautify
- design/mainlayout-coloring

## Teknikker og Værktøjer

### C# med Visual Studio

**Teknik:** Vi har valgt *C#* som vores programmeringssprog, da det er moderne og objektorienteret, hvilket gør det ideelt til udvikling på .NET-plattformen. *C#* er også det sprog vi hovedsageligt har siddet med gennem uddannelsens forløb og er derfor det programmeringssprog gruppen er bedst bekendt og effektiv med.

**Værktøj:** *Visual Studio* er vores valg af IDE til projektet, fordi det tilbyder et komplet værktøj til udvikling i *C#* applikationer og er velkendt for gruppens medlemmer.

### Azure DevOps

**Teknik:** Continuous Integration og Continuous Deployment (CI/CD)<sup>28</sup> - Ved at integrere CI/CD sikrer vi, at vores kodeændringer kontinuerligt testes og deployeres, hvilket reducerer risikoen for fejl og øger pålideligheden i vores udviklingscyklus. CI/CD understøtter også vores feature-driven development, idet vi arbejder i kontinuerlige features, som løbende bliver deployed.

**Værktøj:** Vi har valgt *Azure DevOps* til at implementere CI/CD, da det effektivt automatiserer og forbedrer vores udviklingsproces med hurtigere og mere pålidelig udgivelse af kode. At vælge *Azure DevOps* hjælper os også med integrering, da mange af vores andre ressourcer også er i Azure.

---

<sup>28</sup> Continuous Integration og Continuous Deployment (CI/CD): En praksis og proces i softwareudvikling, der automatiserer integration, test og implementering af kode for at sikre hurtig og pålidelig levering af software.

## App Service

**Teknik:** Platform as a Service (PaaS) gør, at vi kan fokusere på at udvikle og optimere vores applikationer uden at bekymre os om den underliggende server infrastruktur. Dette øger effektiviteten og gør det lettere at skalere og vedligeholde vores tjenester.

**Værktøj:** Azure App Service er vores valg af hosting platform, da den lader os bygge og hoste webapplikationer og API'er uden at bekymre os om server infrastrukturen. Det integrerer også godt med vores andre ressourcer, da det er en del af Azure miljøet og er en relativ billig løsning samt hurtigt og nemt kan skifte mellem serviceplaner for at tilpasse vores behov.

## Database

**Teknik:** Databasestyring - Metoder til at opbevare, organisere og hente data effektivt.

**Værktøj:** Azure Database for MySQL flexible Server - Vi har valgt at benytte denne database, da den tilbyder en høj grad af fleksibilitet og kontrol over databasen. Dette værktøj giver os mulighed for at tilpasse og skalere vores databaser efter behov. Flexible Server er også ret omkostningseffektiv gennem sine fleksible prismodeller, hvilket gør den ideel for både små og store applikationer og hjælper os med at opretholde vores applikationers ydeevne og pålidelighed.

## Key Vault

**Teknik:** Sikkerhed og adgangskontrol, key vaults er en teknik til at beskytte følsomme oplysninger og nøgler i applikationen.

**Værktøj:** Vi valgte *Azure Key Vault* til at håndtere og beskytte adgangskoder, nøgler og certifikater, hvilket styrker vores sikkerhed og adgangskontrol. Key Vault er designet til

at integrere problemfrit med andre Azure-tjenester, hvilket betyder, at vi kan beskytte og hente adgangsdata direkte fra tjenester som Azure App Service og Azure Functions uden at håndtere yderligere integrationer. Hvilket var ideelt for projektet.

## Blazor Web App

**Teknik:** Komponentbaseret web applikationsudvikling er en tilgang til at bygge web applikationer ved hjælp af komponenter og C#. Denne teknik muliggør bedre vedligeholdelse og skalerbarhed gennem genanvendelige komponenter.

**Værktøj:** Med Blazor Web App har vi valgt Blazor Server, som kører på serveren for hurtigere indlæsning samt at følsom logik håndteres på serveren, hvilket kan forbedre sikkerheden.

## MudBlazor

**Teknik:** Komponentbaseret UI-udvikling er en teknik til at skabe brugergrænseflader ved hjælp af genanvendelige komponenter, som fremmer effektivitet og konsistens i designet. Denne tilgang forenkler processen og giver teamet mulighed for at fokusere mere på funktionaliteten frem for applikationens udseende.

**Værktøj:** Her anvender vi værktøjet MudBlazor som er et komponent bibliotek, da det tilbyder komponenter til Blazor, hvilket gør det lettere og hurtigere at skabe moderne og responsive brugergrænseflader. Vi kunne især godt lide MudBlazors komponenter i forhold til andre alternativer som Bootstrap. Og MudBlazor har faktisk en default template som har MudBlazor allerede opsat med Identity for et Blazor Server projekt, hvor der ellers kan være en række problemer med static server og interactive server render modes.

## Apex Charts

**Teknik:** Data Visualisering - Ved at bruge data visualisering kan vi hurtigt identificere mønstre og tendenser i brugerens data, hvilket hjælper brugere med at træffe informerede beslutninger om deres elforbrug, holde sig informeret om elpriser eller se hvordan deres el er produceret.

**Værktøj:** Apex Charts er det værktøj, vi har valgt til at implementere vores data visualiseringer. Det er et kraftfuldt JavaScript-bibliotek, der giver os mulighed for at lave interaktive grafer og diagrammer. Med Apex Charts kan vi nemt integrere og tilpasse vores visualiseringer til at vise vores elforbrugsdata i forskellige tidsperioder og formater, hvilket forbedrer brugeroplevelsen og gør det nemmere for brugerne at forstå deres forbrugsmønstre. Vi brugte Apex Charts i stedet for MudBlazors integreret charts, da der var flere muligheder og features som ikke endnu er implementeret i MudBlazor.

## Entity Framework Core

**Teknik:** Object-Relational Mapping (ORM) er en teknik der letter interaktionen mellem objektorienterede programmeringssprog og relationelle databaser ved at omdanne database rækker til objekter i programmeringssproget.

**Værktøj:** Entity Framework Core - Vi bruger Entity Framework Core som vores ORM, da det effektivt forenkler arbejdet med databaser gennem .NET-objekter og gør CRUD operationer mere intuitive og håndterbare.

## Kravindsamling

Kravindsamling er en essentiel del af udviklingsprocessen, da det sikrer, at løsningen opfylder brugernes og interessenternes behov. For at indsamle de nødvendige krav har vi haft løbende samtaler med vores Product Owner (PO), som har en dybdegående forståelse af projektets mål og muligheder.

## Use Cases

På baggrund af samtaler med PO har vi identificeret og prioriteret features, som er blevet dokumenteret i form af use cases. Use cases bruges som et værktøj til at beskrive funktionelle krav i en struktureret form, hvilket gør det lettere at omsætte kravene til tekniske løsninger.

## Visning af standard el forbrugsdata

<b>Aktør</b>	Bruger
<b>Formål</b>	Brugeren ønsker at se deres historiske elforbrug for at forstå deres forbrugsmønstre.
<b>Forudsætning</b>	Brugeren er logget ind, og applikationen har adgang til eloverblik.
<b>Hovedforløb</b>	Brugeren går ind på web applikationen for at se sit elforbrug. Systemet henter brugerens forbrugsdata. Systemet viser data'en frem i grafisk form, herunder det daglige forbrug for den sidste måned.
<b>Efterforløb</b>	Brugeren kan nu se og analysere sit elforbrug.

<b>Undtagelse</b>	Hvis der ikke er tilgængelige data for en valgt periode, informerer systemet brugeren.
-------------------	--

## OC: Visning af standard el forbrugsdata

<b>Operation</b>	Visning af standard el forbrugsdata
<b>Cross References</b>	Use Case: Visning af standard el forbrugsdata
<b>Preconditions:</b>	<ul style="list-style-type: none"> <li>• Brugeren er logget ind i webapplikationen.</li> <li>• Applikationen har adgang til eloverblik.</li> </ul>
<b>Postconditions:</b>	<ul style="list-style-type: none"> <li>• Systemet viser elforbrugsdata for den standard tidsperiode i form af en graf.</li> </ul>

## Visning af specifik el forbrugsdata

<b>Aktør</b>	Bruger
<b>Formål</b>	Brugeren ønsker at se deres historiske elforbrug for at forstå deres forbrugsmønstre.
<b>Forudsætning</b>	Brugeren er logget ind, og applikationen har adgang til ElOverblik.
<b>Hovedforløb</b>	Brugeren går ind på webapplikationen for at se sit elforbrug. Systemet henter brugerens standard forbrugsdata. Systemet viser data'en frem i grafisk form. Brugeren kan nu vælge mellem tidsintervallerne dagligt, ugentligt, månedligt og årligt. Brugeren kan så få detaljerede oplysninger om deres forbrug fra en given



	startdato til en given slut dato, som systemet så præsentere for brugeren.
<b>Efterforløb</b>	Brugeren kan nu se og analysere deres elforbrug for det specifikke tidsinterval.
<b>Undtagelse</b>	Hvis der ikke er tilgængelige data i en valgt periode, informerer systemet brugeren.

## OC: Visning af specifik el forbrugsdata

<b>Operation</b>	Visning af specifik el forbrugsdata
<b>Cross References</b>	Use Case: Visning af specifik el forbrugsdata
<b>Preconditions:</b>	<ul style="list-style-type: none"><li>• Brugeren er logget ind i webapplikationen.</li><li>• Applikationen har adgang til eloverblik.</li><li>• Brugeren har valgt tidsinterval</li><li>• Brugeren har valgt start og slut dato</li></ul>
<b>Postconditions:</b>	<ul style="list-style-type: none"><li>• Systemet viser elforbrugsdata for den valgte tidsperiode i form af en graf.</li></ul>

## Sammenligning af elleverandører

<b>Aktør</b>	Bruger
<b>Formål</b>	Brugeren ønsker at sammenligne elleverandører for at finde den bedste pris eller den mest bæredygtige løsning.
<b>Forudsætning</b>	Brugeren er logget ind, og applikationen har adgang til aktuelle elpriser og leverandørdata.
<b>Hovedforløb</b>	<p>Brugeren vælger "Sammenlign leverandører" i menuen, hvorefter systemet præsenterer en liste over elleverandører med deres forskellige priser, bæredygtighedsindeks og CO2-udledning.</p> <ol style="list-style-type: none"><li>1. Brugeren vælger kriterier for sammenligning (f.eks. laveste pris, bedste bæredygtighed).</li><li>2. Systemet viser en rangering af leverandører baseret på de valgte kriterier.</li></ol>
<b>Efterforløb</b>	Brugeren kan vælge den leverandør, der passer bedst til deres behov.
<b>Undtagelse</b>	Hvis der mangler data fra en eller flere leverandører, informerer systemet brugeren om dette.

## OC: Sammenligning af elleverandører

<b>Operation</b>	Sammenlign elleverandører
<b>Cross References</b>	Use Case: Sammenligning af elleverandører
<b>Preconditions:</b>	<ul style="list-style-type: none"> <li>• Brugeren er logget ind i webapplikationen.</li> <li>• Applikationen har adgang til aktuelle elpriser og leverandørdata.</li> </ul>
<b>Postconditions:</b>	<ul style="list-style-type: none"> <li>• Systemet viser en rangering af elleverandører baseret på de valgte kriterier (pris, bæredygtighed osv.).</li> </ul>

## Visning af energikilder

<b>Aktør</b>	Bruger
<b>Formål</b>	Brugeren ønsker at se deres energikilder for at forstå hvordan deres strøm er produceret.
<b>Forudsætning</b>	Brugeren er logget ind, og applikationen har adgang til ELOverblik.
<b>Hovedforløb</b>	Brugeren går ind på webapplikationen for at se sine energikilder. Systemet henter brugerens energikilde data. Systemet viser data'en frem så brugeren kan se hvordan deres el er produceret.
<b>Efterforløb</b>	Brugeren kan nu se deres energikilder
<b>Undtagelse</b>	Hvis der ikke er tilgængelige data for den givne elmåler, informerer systemet brugeren.

## OC: Visning af energikilder

<b>Operation</b>	Vis energikilder
<b>Cross References</b>	Use Case: Visning af energikilder
<b>Preconditions:</b>	<ul style="list-style-type: none"><li>• Brugeren er logget ind i webapplikationen.</li><li>• Applikationen har adgang til eloverblik.</li><li>• Applikationen har adgang til Energi Data Service.</li></ul>
<b>Postconditions:</b>	<ul style="list-style-type: none"><li>• Systemet viser data om energikilder, hvilket gør det muligt for brugeren at se, hvordan deres el er produceret.</li></ul>

## Ekstern Data

Applikationens funktionalitet afhænger af data fra eksterne kilder, både API kald og offentligt tilgængelige json filer. Nedenfor er et overblik over hvor vi har diverse data fra.

### API'er

#### ElOverblik API

Link: <https://eloverblik.dk/customer/overview>

ElOverblik API er en tjeneste, der tilbydes af Energinet, som giver adgang til data om elforbrug og elproduktion i Danmark.

API Kald		
Navn	Hvad den bruges til	Link
Data-sharing	Redirect brugeren til ELOverbliks login side og videre til API-adgang	<a href="https://www.eloverblik.dk/customer/data-sharing">https://www.eloverblik.dk/customer/data-sharing</a>
Metering Points	Bruges til at få brugerens elmåler (Metering point) og forbrugerinformation.	<a href="https://api.eloverblik.dk/customerapi/api/meteringpoints/meteringpoints">https://api.eloverblik.dk/customerapi/api/meteringpoints/meteringpoints</a>
GetTimeSeries	Bruges til at hente tidsserier af forbrug på brugeren. Dette API kald er essentiel for alt dashboard funktionalitet som bruger disse 'time series'.	<a href="https://api.eloverblik.dk/customerapi/api/meterdata/gettimeseries/%7BdateFrom%7D/%7BdateTo%7D/%7Bperiod%7D">https://api.eloverblik.dk/customerapi/api/meterdata/gettimeseries/%7BdateFrom%7D/%7BdateTo%7D/%7Bperiod%7D</a>

## Energi Data Service API

Link: <https://energidataservice.dk/datasets>

Energi Data Services API er en offentlig tjeneste fra Energinet, der giver adgang til realtidsdata og historiske data om elnettet i Danmark.

API Kald		
Navn	Hvad den bruges til	Link
DeclarationOnGridMix	Bruges til at hente information om, hvordan elektriciteten i DK1 eller DK2 er produceret eller importeret.	<a href="https://www.energidataser vice.dk/tso-electricity/DeclarationGridmix">https://www.energidataser vice.dk/tso-electricity/DeclarationGridmix</a>
Datahub Pricelist	Bruges til at hente spot priserne for de individuelle elselskaber.	<a href="https://energidataservice.dk/tso-electricity/DatahubPricelist">https://energidataservice.dk/tso-electricity/DatahubPricelist</a>
Elspot prices	Bruges til at hente spot priserne på el i Danmark.	<a href="https://energidataservice.dk/tso-electricity/Elspotprices">https://energidataservice.dk/tso-electricity/Elspotprices</a>

## Andre Kilder

### Elpris

Link: <https://elpris.dk/data>

Elpris.dk er en hjemmeside, der prøver at gøre det lettere for brugere at få et overblik og sammenligne elprodukter og elpriser på markedet. De har ikke en direkte en API, men de har data tilgængelig som vi har kunne hente i form af json filer.

JSON filer		
Navn	Hvad den bruges til	Link
Products	Bruges til at hente de forskellige produkter fra diverse elleverandører	<a href="https://elpris.dk/data/products_543.json?cacheBust=1734694626156">https://elpris.dk/data/products_543.json?cacheBust=1734694626156</a>
DistributionAreaCharge	Bruges til at finde ud af hvilken pris der er i de forskellige distributions areas	<a href="https://elpris.dk/data/distributionAreaCharge_543.json?cacheBust=1734694626010">https://elpris.dk/data/distributionAreaCharge_543.json?cacheBust=1734694626010</a>
Price Calculation Function	Bruges til at udregne elprisen med tariffer, afgifter og moms.	<a href="https://elpris.dk/data/priceCalculationFunction.js">https://elpris.dk/data/priceCalculationFunction.js</a>
ZipCodes med distributionAreas	Bruges til at finde ud af hvilken distribution area man er del af ud fra ens postnummer	<a href="https://elpris.dk/data/static.json?cacheBust=1734694984517">https://elpris.dk/data/static.json?cacheBust=1734694984517</a>
Standard forbrugsprofiler	Bruges til at finde gennemsnitlig priser baseret på ens boligprofil.	<a href="https://elpris.dk/data/configuration.json?cacheBust=1734694984706">https://elpris.dk/data/configuration.json?cacheBust=1734694984706</a>

## Funktionelle krav

### Visualisering af elforbrug

Systemet skal kunne præsentere brugernes elforbrug i historiske data (time, dagligt, månedligt, årligt) via grafer. Dette skal hjælpe brugerne med at identificere perioder med højt forbrug.

### Sammenligning af leverandører

Systemet skal kunne sammenligne forskellige leverandører på blandt andet pris, og præsentere disse oplysninger for brugerne, så de kan tage en beslutning om eventuelt leverandørskifte.

### Fremvisning af Elpriser

Systemet skal vise dagens elpriser, så brugeren kan få et overblik og planlægge deres energiforbrug for dagen.

### Visualisering af Energikilder

Systemet skal vise brugerens energikilder for deres region (DK1/DK2). Brugerens forbrug skal relateres til den producerede elektricitet, så forbrugeren kan se, hvilken andel de forbruger.

### Fremvisning af omkostninger

Systemet skal give brugeren et overblik over de estimerede omkostninger ved at anvende forskellige apparater i hjemmet baseret på den aktuelle elpris. Overblikket skal inkludere typiske hverdagseksempler som omkostningerne ved at se TV i en time eller tage et varmt bad på 10 minutter.



## Gamification

Systemet skal inkludere elementer af gamification for at motivere brugeren til at reducere deres energiforbrug. Dette kan eksempelvis opnås ved at visualisere brugerens forbedringer over tid eller ved at sammenligne deres forbrug med andre brugere i tilsvarende boligstørrelser. Funktionen skal præsenteres på en engagerende måde, der fremhæver positive resultater for at få brugeren til at fortsætte med energibesparende tendenser.

## Dashboard

Systemet skal kunne præsentere et dashboard, der viser brugerens elforbrug, elpriser, og hvordan brugerens el er blevet produceret.

## Integration med eksterne API'er

Systemet skal kunne hente data fra eksterne kilder som ElOverblik og Energi Data Service for at sikre, at brugerne præsenteres for de mest opdaterede elpriser og forbrugsdata. Systemet skal kunne opdatere disse data automatisk.

## Ikke-Funktionelle krav

### Skalerbarhed

Systemet skal kunne skalere, så backend og databaser kan håndtere en stigende mængde brugere uden at miste ydeevne.

### Sikkerhed

Systemet skal sikre brugerdata. Data som API nøgler i databaserne skal krypteres for at sikre brugernes data.

### Ydeevne

Systemet skal sikre, at responstiden for indlæsning af dashboard og visning af grafer er minimal, så der opnås en hurtig og smidig brugeroplevelse. API-kald til eksterne tjenester skal optimeres for at minimere indlæsningstiden.

### Brugervenlighed

Systemet skal have en intuitiv og nem navigerbar brugergrænseflade, som også er let at bruge for brugere uden teknisk viden. Indtil videre skal UI'et kun passe til computer skærmstørrelse.

# Design

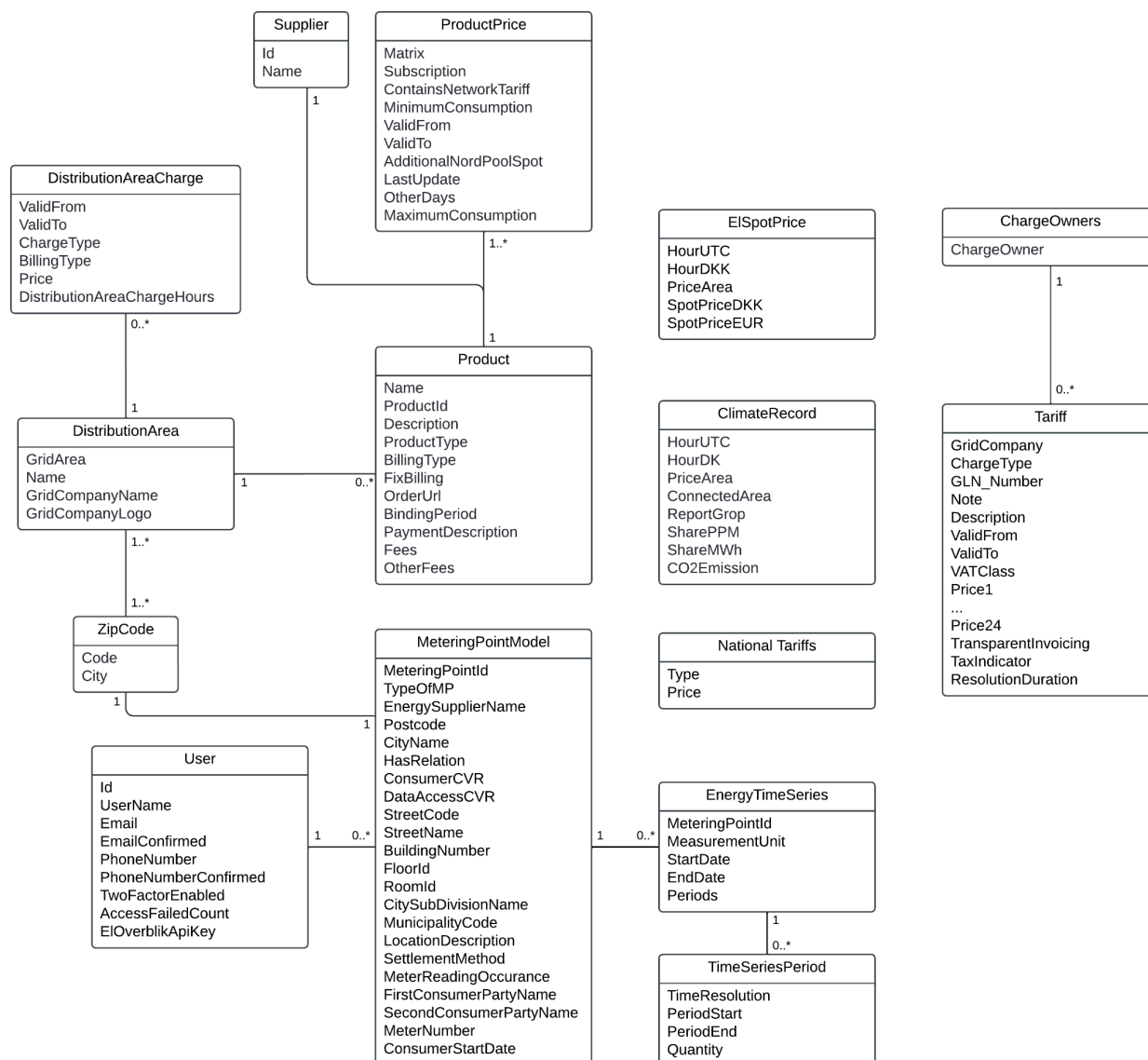
Designfasen i udviklingsprocessen er afgørende for at sikre en velfungerende og skalerbar løsning. I dette kapitel præsenterer vi både vores high-level og low-level design.

## High Level

High-Level Design giver et overblik over systemets arkitektur og fokuserer på den overordnede opbygning af systemet.

### Domænemodel

Vi har udviklet en domænemodel for at skabe en fælles forståelse af forretningsdomænet og sikre en struktureret tilgang til systemets logik. Modellen hjælper med at adskille forretningsregler fra datahåndtering og UI. Derudover understøtter den en klar og konsistent måde at arbejde med vores data på, så vi undgår misforståelser og fejl.



## Azure Arkitektur Diagram

Vi har valgt at benytte Azure Cloud-løsninger for at sammensætte vores system, hvilket giver os mulighed for at skabe en løsning, der både er let at integrere og skalerbar i fremtiden.

## Første uddrag

Vores oprindelige idé til opsætningen af vores Azure-arkitektur var at benytte to separate webapplikationer. Den ene skulle fungere som en produktions-webapp, som er den live-version, brugerne har adgang til. Den anden skulle være en udviklings-webapp, der fungerer som vores testmiljø. På denne måde kunne vi teste nye ændringer og se, hvordan de påvirker hjemmesiden, uden at risikere at påvirke live-miljøet<sup>29</sup>.

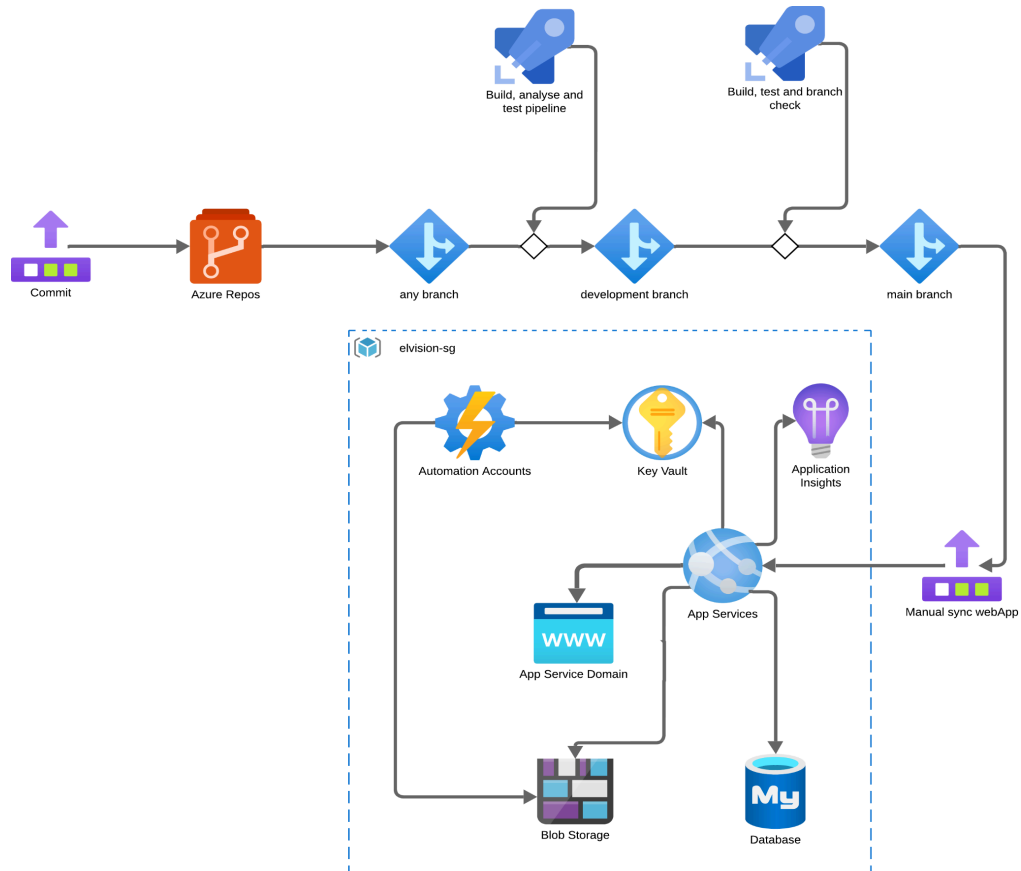
## Endelige resultat

Vi endte med at droppe den første version af arkitekturen på grund af flere udfordringer, herunder problemer med rettigheder og af økonomiske årsager. Den endelige Azure-arkitektur er vist på det nedenstående billede.

I stedet for at have et separat testmiljø valgte vi at teste lokalt ved hjælp af *Build Solution* i Visual Studio, hvilket fungerer fint for vores behov. Den nuværende Azure-arkitektur viser, hvordan vi går fra det lokale miljø til live-miljøet. Den viser processen fra branch-merging til opdatering af web app'en samt samspillet mellem de forskellige ressourcer.

---

<sup>29</sup> Bilag: Første uddrag af Azure Arkitektur



## Low Level

Low-Level Design går i dybden med specifikke dele af systemet. Her præsenteres tekniske detaljer, der understøtter high-level designet.

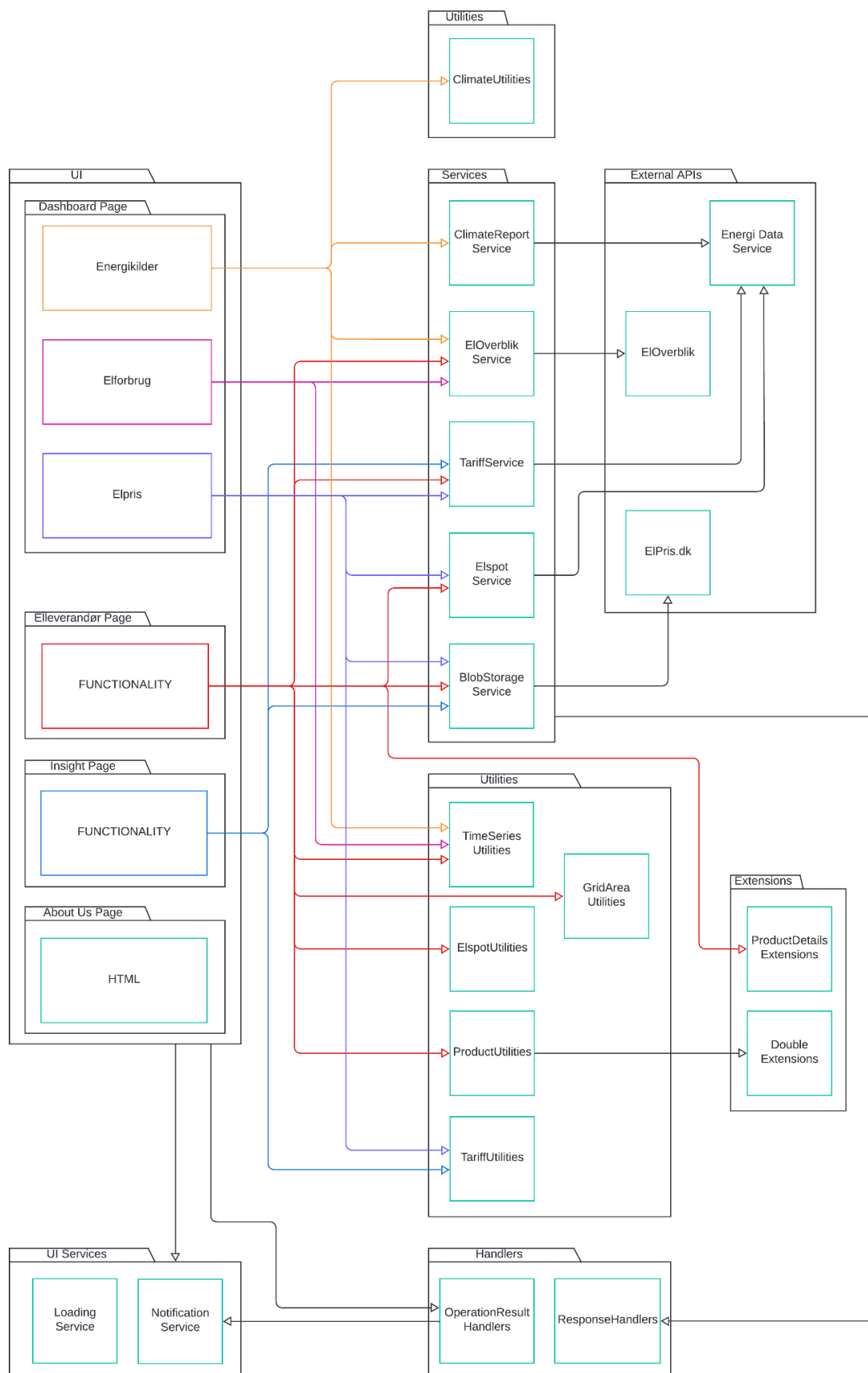
## Web App Arkitektur

**Blazor-komponenter:** Disse fungerer som UI-laget og kalder tjenester (services) for at få data og håndtere brugerinteraktioner. På den måde holdes forretningslogik og datamanipulation adskilt fra præsentationslaget.

**Services:** Disse fungerer som mellemlid mellem Blazor-komponenterne og backenden. De kalder API'er og repository-laget, hvilket gør, at Blazor-komponenterne ikke behøver at kende detaljerne om API'et eller databasen. Services indeholder forretningslogik og håndterer eventuelle transformationsopgaver, f.eks. formattering af data til eller fra API'er.

## Systemafhængigheds Model

Vores Systemafhængigheds model viser hvordan brugergrænseflade, services, utilities og eksterne API'er interagerer med hinanden. Den fremviser dataflow og afhængigheder, hvor UI-komponenter som Dashboard-siden henter data fra services, der igen bruger utilities og eksterne API'er til at levere funktionalitet.

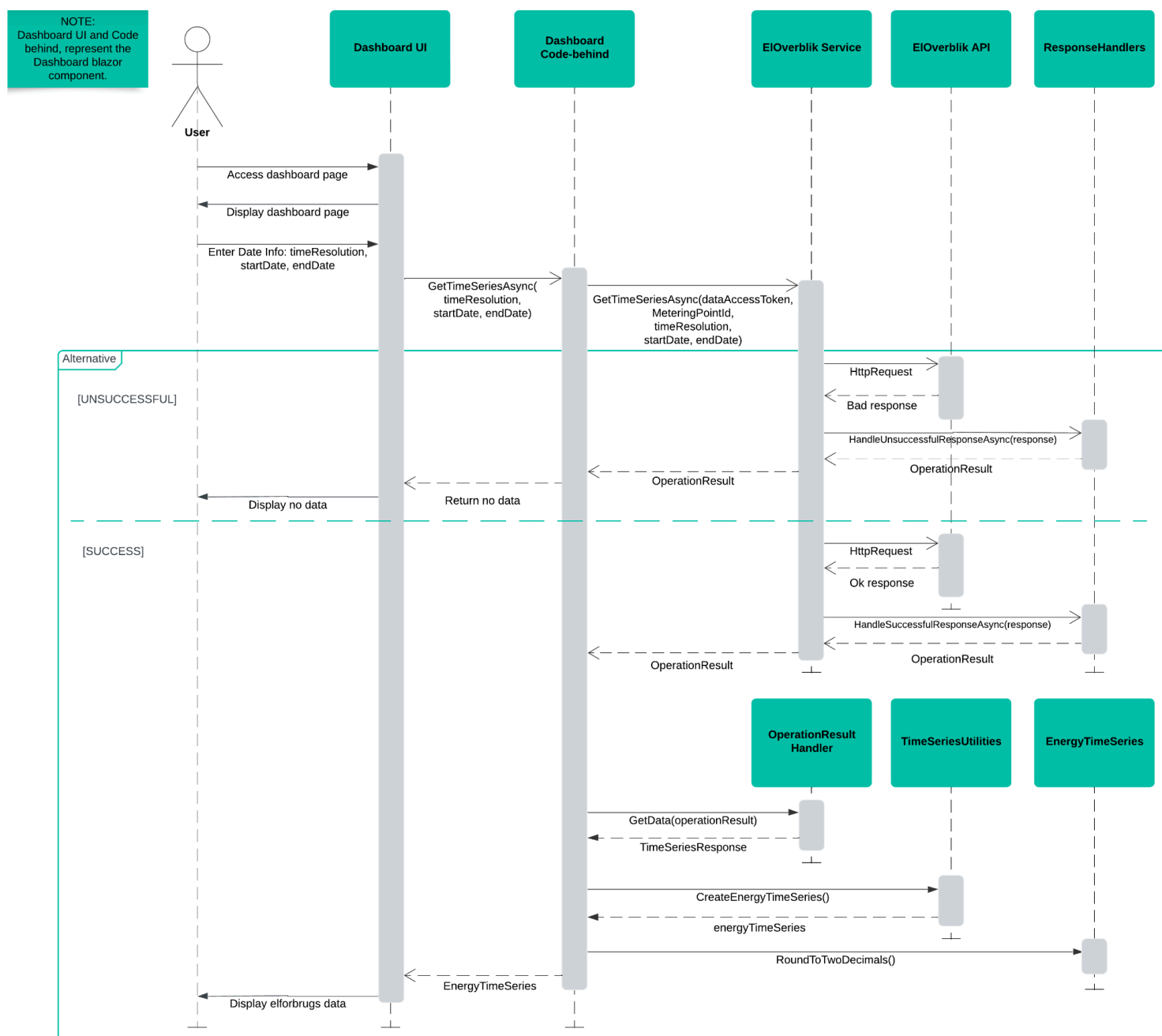




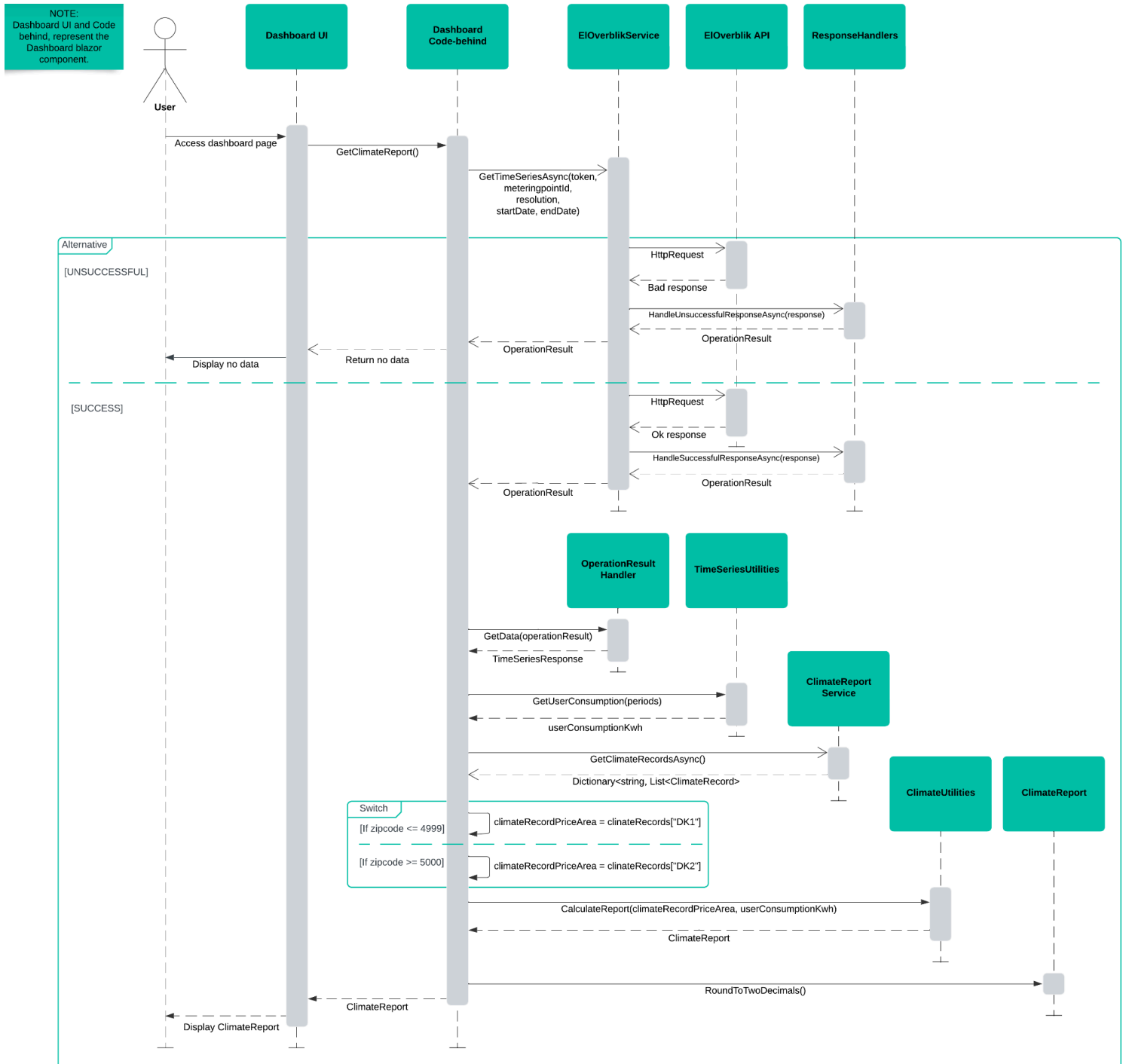
## Interaktionsdiagrammer

Vores interaktionsdiagrammer viser, hvordan aktører interagerer med systemet, og hvad der så sker på systemets side som følge.

### Visning af specifik Elforbrugs data



## Visning af Energiforbrugs kilder



# Sprint Overblik

I dette kapitel præsenteres en samlet oversigt over de aktiviteter, der er blevet gennemført i løbet af projektforløbet. Vi forklarer nærmere de specifikke aktiviteter og beskriver faste elementer som sprint planlægning og retrospektiv analyse. Gennem disse beskrivelser giver vi indblik i vores arbejdsproces, herunder de udfordringer, vi har mødt og de succeser, vi har opnået.

## Sprint 1: 14 Okt. - 25 Okt.

### Sprint Plan

I første sprint handler det om at få styr på fundamentet for projektet. Først skal vi have opsat en rapport struktur og etableret standarder for Git, kode og udviklingsmiljø. Derefter skal vi have lavet business case, fml og kravspecifikationer og udarbejde de vigtigste use cases.

Vi skal have opsat udviklingsmiljøet i Azure Portal med DevOps, og starte implementeringen af Blazor-projektet med MudBlazor. Samtidig skal vi prøve at finde relevante API'er og opsætte en database med Entity Framework. Vi skal også have påbegyndt domænemodellen med de vigtigste domæner.

Til sidst skal vi prøve at implementere hentning af data fra ElOverblik for at sikre, at vi kan arbejde med reelle data fremadrettet.

## Aktiviteter

### ElOverblik API

I dette sprint fik vi succes med at etablere forbindelse til ElOverblik. Vi har nu mulighed for at hente en brugers API-token samt information om brugerens elmåler. Ud fra denne elmåler kan vi så kalde en tidsserie, som returnerer brugerens elforbrug i formaterne år, måned, dag eller time for den valgte periode.

### Azure portal

Vi har fået oprettet vores udviklingsmiljø i Azure Portal, med vores egen ressourcegruppe. Vi har konfigureret en MySQL-server, Key Vault, Storage Account og en Azure WebApp. Azure web app'en bliver deployet med koden som ligger i vores Azure Repository main branch, dette er muligt ved at give vores web app adgang til repositoret via en PAT<sup>30</sup>.

Vi stødte på udfordringer med rettigheder, da vores ressourcegruppe var placeret i Energinets tenant, hvilket begrænsede vores adgang til at arbejde med alle ressourcer. Dette blev dog løst i løbet af sprintets anden uge, efter vi tog kontakt til Energinet og fik forhøjet vores rettigheder.

---

<sup>30</sup> Personal Access Token: er en sikkerhedsmekanisme, der bruges til at autentificere din identitet og give adgang til en tjeneste eller et API uden at bruge din adgangskode direkte

## UI Design

I dette sprint fik vi udarbejdet et first draft af UI designet til applikationen.<sup>31</sup>

## Azure DevOps

### Scrum board

DevOps har en funktion kaldet Boards, som vi bruger til planlægning af backlog og opgaver. Denne funktion understøtter vores Scrum-processer. Vi har organiseret vores sprints i 2-ugers intervaller, hvor hver sprint har sin egen backlog med tilhørende opgaver. Dette giver os et klart overblik over både gennemførte og kommende opgaver.

### Git branch konvention

Vi har etableret en branch konvention for vores udviklingsproces for at sikre konsistens og effektivitet. Konventionen er designet sådan, at *Main*-branchen repræsenterer den udrullede version af applikationen, og kun *Development*-branchen kan merges ind i *Main*. *Development*-branchen fungerer som udviklings- og testmiljøet, hvorfra der kan oprettes specifikke branches til forskellige formål, såsom */feature*, */hotfix*, */bug* eller */development*.<sup>32</sup>

### Pipelines

Vi har lavet nogle forskellige pipelines, så som en build pipeline som køres hver gang man laver en pull request, hvor den skal være gennemført før man kan complete pull request. Det var også meningen at build pipeline, skal køre noget kodeanalyse/linting, men det virker ikke helt endnu.

Vi har også lavet en anden pipeline som kører på alle pull requests til main. Den kontrollerer, om den branch som prøver at merge ind i main, er 'development' branchen, for at sikre at det kun er 'development' branchen som kan merge ind i main.

---

<sup>31</sup> Bilag: UI first draft

<sup>32</sup> Bilag: Visualisering af Git Branch Konventioner

Denne pipeline skaber mere sikkerhed i forhold til ikke at kunne merge ind i main ved en fejl, og sikre at alt, som bliver merged ind i main, først er testet i development branchen.

🔍 main ▾    🔒 ELVision / pipelines/build\_&\_check\_merge\_branches.yml \*

```

1  trigger: none
2
3  pool:
4  | - vmImage: windows-latest
5
6  variables:
7  | - buildConfiguration: 'Release'
8  | - solution: '**/*.sln'
9
10 steps:
11 | - powershell: |
12 |   | - $branchName = "$(System.PullRequest.SourceBranch)"
13 |   | - Write-Host "Source branch is: $branchName"
14 |   | - $branchName = $branchName -replace 'refs/heads/', ''
15 |   | - if ($branchName -ne "development") {
16 |   |   | - Write-Host "Pull requests can only come from 'development' branch"
17 |   |   | - exit 1
18 |   |   }
19 |   | - displayName: 'Ensure branch is development'
20 |
21 |
22
23 Settings
24 | - task: DotNetCoreCLI@2
25 |   | - displayName: Build
26 |   | - inputs:
27 |   |   | - command: 'build'
28 |   |   | - projects: '$(solution)'
29
30 Settings
31 | - task: VSTest@3
32 |   | - inputs:
33 |   |   | - testSelector: 'testAssemblies'
34 |   |   | - testAssemblyVer2: |
35 |   |   |   | - **\Tests\bin\**\Tests.dll
36 |   |   | - searchFolder: '$(System.DefaultWorkingDirectory)'
37 |   |   | - codeCoverageEnabled: true
38 |   |   | - diagnosticsEnabled: true

```

Vi ville også have lavet en publish pipeline, som automatisk skubber alle ændringer fra development og main op i hver deres Azure app service, for altid at have et opdateret cloud miljø at teste det i. Men på grund af de omkostninger, der ville være for at hoste en web app til testmiljøet, gik vi ikke denne vej.

Det var heller ikke muligt at lave en publish pipeline på grund af nogle ekstra rettigheder, som vi ikke kunne få fra Energinet. Derfor bliver vi nødt til manuelt at gå ind og sync web appen, med main branchen i Azure.

## Database og Entity Framework

Som nævnt tidligere har vi opsat en MySQL database via Azure Portal. For at få vores applikation til at sende data til databasen, har vi oprettet en DbContext ved brug af Entity Framework med en connection string direkte til databaseserveren. For at opdatere databasen opretter vi en migration ved hjælp af Entity Framework og anvender den derefter på den valgte database.

Dette giver os mulighed for at undgå at skrive direkte SQL kode. Vi kan også nemmere håndtere database strukturer ved hjælp af migration og vi har nemmere ved at rulle databasen tilbage ved fejl.

## MudBlazor

Vi har fået opsat MudBlazor på vores Blazor applikation, som gør det muligt for os at inkludere MudBlazors komponent bibliotek, som er udviklet specifikt til Blazor. Det tilbyder et bredt udvalg af brugergrænseflade komponenter, hvilket gør det lettere at bygge en moderne og responsiv brugeroplevelse i vores applikation.

## Find interessante api kald

Vi har fundet mange forskellige api kald på Energi Data Service, ELOverblik og ElprisenLigeNu<sup>33</sup>, som vi kan bruge til at skabe funktionalitet.<sup>34</sup>

## Azure Arkitektur diagram

Vi har lavet et azure arkitektur diagram for at skabe et overblik over hvilke ressourcer vi bruger, og hvordan de snakker sammen.<sup>35</sup>

## Use Cases

I dette sprint har vi skrevet tre centrale use cases for applikationen.

- Vi har en use case hvor brugerne kan få indsigt i deres forbrugsmønstre for forskellige tidsperioder.
- Vi har også lavet en use case som går ud på at brugerne kan se deres elforbrug fordelt på energikilder som sol og kul via en pie chart.
- Endelig har vi lavet en use case som omhandler en sammenligningsfunktion for elleverandører, så brugerne kan finde den mest økonomiske eller bæredygtige løsning.

## FML

Vi har udarbejdet et forretningsmodel lærred med det formål at visualisere, analysere og forbedre vores forretningsmæssige argumentation for projektet.

---

<sup>33</sup> Bemærk: ElprisenLigeNu bruges ikke i applikationens endelige version.

<sup>34</sup> Se kapitel: Ekstern Data under Kravindsamling

<sup>35</sup> Se kapitel: Design - High Level - Azure Arkitektur



## Business Case

Vi har også udarbejdet en Business case for projektet. Dette er gjort for at fremhæve projektets værdi og berettigelse. Dette giver ledelsen et klart overblik over de forventede omkostninger, fordele og risici, hvilket gør det lettere at vurdere, om projektet bør investeres i.

## Kravspecifikationer

Vi har lavet yderligere kravspecifikationer ud over use cases, for at uddybe kravene på et mere teknisk plan.

## Retrospektiv

### Features

- Etableret forbindelse til ELOverblik API, her henter vi brugerens API-token og elmålerinformation.
- Implementeret funktionalitet til at hente brugerens elforbrug som tidsserier (år, måned, dag, time).

## Udfordringer

Vi stødte på nogle udfordringer med opsætningen af vores Azure resource group, da der opstod problemer med de rettigheder, vi havde til at konfigurere vores Azure miljø. Dette resulterede i lidt frem og tilbage med Energinet for at få det til at lykkes, men vi formåede til sidst at finde en løsning.

Derudover havde vi også nogle vanskeligheder med rapporten. Her havde vi svært ved at finde ud af hvordan vi bedst kunne opsætte dens struktur, samt hvad der burde inkluderes for at sikre, at rapporten er informativ, men ikke består af unødvendig fylde.

## Succeser

Generelt har vi skabt et godt arbejdsmiljø, som vi alle kan præstere godt i. Vi har været gode til at lave nogle aftaler som vi alle er enige i og kan regne med hinanden opfylder.

## Forbedringer

Vi kan fokusere på at optimere vores daglige scrum møder, så vi får mere ud af dem og kan afslutte dem hurtigere. På den måde kan vi hurtigere komme i gang med dagens opgaver.

## Sprint 2: 28. Okt - 8. Nov

### Sprint Plan

I denne sprint handler det om at få implementeret dashboardet med MudBlazor, hvor brugeren skal kunne se sit elforbrug, elpriser og energikilder. Samtidig skal vi færdiggøre pipeline, som skal stå for analysen af koden.

Vi videreudvikler domænemodellen og ser på, hvordan vi kan præsentere forbrugernes energikilder, for eksempel ved at vise, at 30 % af elforbruget stammer fra vindenergi og 20 % fra vandkraft. Derudover skal vi udarbejde use cases til dashboardet, som SSD og SD diagrammer.

Til sidst sætter vi Azure Application Insights op, så vi kan overvåge og analysere applikationen.

### Aktiviteter

#### Vis elforbrug med tidsinterval

Vi har implementeret Apex Charts til at visualisere brugerens elforbrug via et søjlediagram. Diagrammet giver brugeren mulighed for at filtrere og få vist forbruget

på time-, dags-, måneds- eller årsbasis samt vælge et specifikt tidsinterval.

Elforbrugsdataen hentes direkte fra brugerens elmåler via ElOverblik, hvor vi trækker data som en tidsserie. Vi planlægger fremtidige forbedringer for denne feature, for at give brugerne en endnu bedre forståelse af deres elforbrug.

## Vis elpriser

Vi implementerede funktionalitet til at fremvise elpriser ved brug af apex charts, via Elprisenligenu<sup>36</sup> api kaldet. Det er et API kald som giver en forudsigelse af el-prisen uden moms og afgifter(også kaldet elspotprisen).

Apex charts er et komponentbibliotek som fokuserer på, at udvikle charts, hvor man blandt andet kan 'hover' med musen for at få yderligere information. Hvilket var en feature, vi meget gerne ville have.

## Kode Analyse Pipeline

Vi har lavet en pipeline i azure devops, som bliver kørt hver gang der er en pull request til development. Pipelinen har til formål at Bygge og Analysere koden. At køre build i pipelinen, sørger for at nuget packages bliver restored og projektet kan bygge uden fejl, i det miljø som det skal køre i<sup>37</sup>.

---

<sup>36</sup> [Elpris API - Åben og gratis](#) - vi bruger ikke længere dette API kald. Nu bruger vi spotprisen fra energi data service og regner selv en mere troværdig elpris ud.

<sup>37</sup> Bilag: Kode Analyse Pipeline

Mend Bolt, kontrollerer alle tredjeparts afhængigheder for sikkerheds sårbarheder.

Med Microsoft Security DevOps kan man tilvælge forskellige ting man vil scanne efter, for eksempel AntiMalware.

GitLeaks scanner efter secrets som er hardcoded i koden, så man sikrer at man ikke ved et uheld leaker nogle keys.<sup>38</sup>

## Elleverandør page

### Data fra Elpris.dk

I sprint 2 fandt vi ud af at vi ikke kunne få data fra elleverandører som har direkte kontakt til kunden, men i stedet kun fra netselskaber, via Energi Data Service. Det var lidt en krise for os, da vi havde forventet at kunne få den data vi ville have brug for, der.

Vi kiggede derfor rundt på andre hjemmesider som havde den funktionalitet som vi var interesserede i. Vi spurgte nogle forskellige, hvordan de havde gjort osv. Samt kiggede på hvilke filer de indhentede på siden, og fandt tilfældigvis *elpris.dk* som havde alt deres data liggende som json filer og javascript, helt åbent og tilgængeligt.

Der er blandt andet *products.json* som indeholder alle elleverandører, med en masse information og priser.

Der er også en javascript fil som hedder *pricecalculatorfuntions.js*<sup>39</sup>, hvilket er et langt script som udregner elleverandørpriser med tariffier, skatter osv.

En komplikation ved at få data fra elpris.dk, er at vi afhænger 100% af at de har korrekt data. Umiddelbart opdaterer de deres data ofte, men vi kan ikke verificere om det er helt korrekt. Vi kan prøve at sammenligne noget af deres data med andres data, hvilket helt klart er vores bedste mulighed, da det er meget omfattende selv at indsamle dataen, og det er der hverken tid eller rettigheder til.

---

<sup>38</sup> Bilag: GitLeaks Analyse test

<sup>39</sup> Vi har omskrevet den her fil, men originalen kan findes i [wwwroot/js](http://wwwroot/js).

## Price Calculator Functions

Som beskrevet ovenfor, fandt vi *pricecalculatorfunctions.js*, som indeholder alle udregninger i forhold til elleverandør priser med dataen fra *products.json*. Det har vi udnyttet ved at læse det grundigt igennem, omskrevet det til C# og fået det til at køre i vores eget system. Vi har lavet 2 filer, *ProductUtilities.cs* og *ProductDetailUtilities.cs*, som indeholder den omskrevne kode fra *pricecalculatorfunctions.js*.

## Runbook i Azure Automation Account

For at støtte udviklingen og vedligeholdelsen af vores ProductUtilities(omskrevne kode fra *pricecalculatorfunctions.js*), har vi oprettet et PowerShell-script, der dagligt køres som en runbook i en Azure Automation Account. Scriptet er designet til at hente de nødvendige JSON-filer, som ProductUtilities er afhængig af for at fungere. Grunden til, at scriptet kører en gang dagligt er, at mange af disse JSON-filer løbende bliver opdateret. Vi sikrer, at vores data holdes opdateret ved at sammenligne de nye JSON-data med de eksisterende filer, så vi kun opdaterer filerne, hvis der er reelle ændringer. Alle JSON-filer gemmes i vores Storage Account i en privat container, hvorfra de kan tilgås af vores Blazor-projekt. Følgende er et udkast af runbook powershell scriptet.

```
# Fetch and upload JSON files directly to Azure Storage
foreach ($gridArea_url in $gridArea_Urls) {
    $fileName = [System.IO.Path]::GetFileName($gridArea_url)
    $uri = "$baseUri" + "/" + "$fileName" + "?" + "$sasToken"

    # Download the JSON content
    $jsonContent = Invoke-WebRequest -Uri $gridArea_url -Method GET -UseBasicParsing

    # Convert bytes to plain text (UTF-8 encoding)
    $jsonString = [System.Text.Encoding]::UTF8.GetString($jsonContent.Content)

    # Save the content to a local file
    $jsonString | Out-File -FilePath "$fileName" -Encoding utf8

    $jsonObject = $jsonString | ConvertFrom-Json | ConvertTo-Json -Depth 10 -Compress
    $storageAccountFile = Invoke-WebRequest -Uri $uri -Method Get -UseBasicParsing
    $storageAccountFileContent = $storageAccountFile.Content

    $storageAccountFileObject = $storageAccountFileContent | ConvertFrom-Json | ConvertTo-Json -Depth 10 -Compress

    if ($jsonObject -ne $storageAccountFileObject){
        Write-Output "$fileName has changed, uploading new file"

        UploadToStorage -fileName $fileName -uri $uri

        Write-Output "Uploaded $fileName to $containerName"
    }else{
        Write-Output "$fileName not changed, skipping upload"
    }
}
```

## Klimaaftryk fra ELOverblik

Noget tid før projektets start har der været en opdatering af ELOverbliks API-kald for klimaaftryk rapport. En bruger skal nu på ELOverblik generere en emissionsrapport med kaldet "POST <https://eloverblik.dk/emissions/api/report/calculateemissionsjob>". Når rapporten så er generet, kan brugeren hente rapporten med "GET <https://eloverblik.dk/emissions/api/report/getemissionsreportjobs>". Problemet er, at som med alle de andre API-kald til ELOverblik, skal Authorization headeren sættes med et bearer token. Men, det token, som er krævet for de to nævnte API-kald, er ikke en af de to tokens for brugeren vi er bekendte med. Det er et tredje token som er et login session token fra *MitID*<sup>40</sup>. Det er således ikke muligt for os på nuværende tidspunkt at få fat i disse kald. Vi har sendt en mail til DataHub supporten for at få mere information om dette og om der er en måde vi kan få data'en på.

## Retrospektiv

### Features

De tilføjede features til applikationen:

- Elforbrug: Implementering af Apex Charts for at vise brugerens elforbrug for forskellige tidsintervaller (time, dag, måned, år). Før blev MudBlazors grafer brugt.
- Elpris: Brugt ElprisLigeNu<sup>41</sup> API for at vise elpriser ved hjælp af Apex Charts. API'en giver elpris data uden moms og afgifter. Apex Charts leverer interaktive grafer med hover-funktion for yderligere information.
- Elleverandør side: Vi har implementeret en side hvor man kan se priser fra forskellige leverandørers elaftaler. Dataen er hentet fra Elpris.dk, da de havde frit tilgængelige json filer liggende på hjemmesiden, som de selv bruger til

---

<sup>40</sup> MitID: er en dansk digital signatur, der bruges til sikker identifikation og adgang til offentlige og private tjenester.

<sup>41</sup> Bemærk: ElPrisenLigeNu bruges ikke længere.

samme formål. Der er både data med en masse elaftaler og en javascript fil, med komplekse udregninger som er blevet omskrevet til C#.

## Udfordringer

### Tasks

Midt i sprintet oplevede vi et tilbageslag, hvor det føltes som om, vi ramte en mur og ikke rigtig kom videre. Vi havde udfordringer med at fordele opgaver, da vi begyndte at mangle opgaver at arbejde med. Vi skal fremadrettet blive bedre til at specificere opgaver mere detaljeret og tage hånd om de mindre spændende opgaver, da de også er nødvendige for projektets fremdrift.

### API'er

Vi havde nogle problemer i den sidste halvdel af dette sprint med at meget af den information vi havde regnet med at kunne hente ret nemt gennem API-kald, lige pludselig ikke var muligt og at vi måtte finde en anden løsning. Vi fik dog fundet løsninger til det meste og afventer stadig svar fra DataHub support om adgang til klimaaftryks rapporter.

## Forbedringer

### Pull request / Merge requests

Generelt for gruppen skal vi være bedre til at kontrollere diverse pull og merge requests, for at sikre at der ikke sker uhensigtsmæssige ændringer i koden. Gruppen har fået lidt en tendens til at bare approve uden at kontrollere, hvad ændringerne er. Dette forårsagede en situation hvor en del af dashboardet i vores main branch, pludselig ikke længere var funktionel.



## Omkostningsstyring

Vi skal have kultureret en tendens til at ofte kontrollere vores omkostningsoversigt for at skabe et bedre overblik over vores forbrug af ressourcer. Det er gået fint indtil videre, men ved at være bedre kan vi fange uventede omkostninger hurtigere. Vi havde et scenarie hvor der var blevet tilføjet en ressource for at løse et automation problem, det virkede dog ikke og siden blev slettet. *Men selvom siden blev slettet, fortsatte hosting planen.* Dette blev derfor ikke opdaget i et par arbejdsdage og fik forårsaget unødvendige omkostninger.

## Succeser

Vi havde god fremgang i dette sprint, og til slut i sprinten havde vi også et produktivt møde på Energinet hvor mange forskellige personer i virksomheden kom for at se vores projekt og var meget tilfredse med det.

## Sprint 3: 11. Nov- 22. Nov

### Sprint Plan

I denne sprint skal vi fokusere på at færdigudvikle vores diagrammer samt integrere energikilder for brugeren. Dette indebærer, at vi skal kortlægge, hvor meget af forbrugerens elektricitet der kommer fra forskellige ressourcer som vind, sol, vandkraft osv. Disse data skal vises i et diagram på vores dashboard for nem adgang og læsbarhed.

Vi skal finde ud af hvordan vi bearbejder tariffer fra energi data service. Vi var til et møde med Peter vores PO<sup>42</sup>, samt flere fra Energinet, hvor vi fandt ud af hvor vi kunne få tariffer fra energi data service. Det er et uoverskueligt datasæt, som kan virke meget rodet.

---

<sup>42</sup> Product Owner: Personen som 'ejer' vores produkt.

## Aktiviteter

### Energikilder

Det er lykkedes os at få udviklet Energikilde featuren, vi fik noget mere dokumentation på hvordan ELOverblik gør, som hjalp os lidt på vej. Vi endte med at hente den totale produktion for brugerens region, og så beregner vi hvor meget forbrugernes forbrug udgør af den producerede energi. På den måde kan vi præsentere et gennemsnitligt overblik over, hvordan brugerens energi er produceret.

### Tariffer og Afgifter

Til at starte med, da vi undersøgte forskellige API-kald, kunne vi ikke helt se os ud af det kald, som indeholder tariffer og afgifter fra Energi Data Service<sup>43</sup>. Men med inspiration fra et github repository<sup>44</sup>, Har vi nu fundet ud af hvordan vi kan udvinde tariffer og afgifter fra Energi Data Service.

Vi har nu mulighed for at inkludere tariffer og afgifter i vores spotpris beregninger, hvilket gør vores elpris graf mere præcis og repræsentativ for brugerens reelle elpriser. Dette opnås ved at identificere brugerens tilhørende *ChargeOwner*<sup>45</sup> baseret på deres postnummer. Herefter foretager vi et API-kald til Energi Data Service med informationer fra den valgte ChargeOwner. Dette giver os adgang til de 24 timelige tariffer samt 3 "additional tariffs", som udgør afgifterne.(se *Elvision/Services/TariffService.cs*)

### Dark Theme

Vi har ændret applikationens theme til dark theme for en mere behagelig brugeroplevelse.

---

<sup>43</sup> Api kald for tariffer og afgifter: <https://www.energidataservice.dk/tso-electricity/DatahubPricelist>

<sup>44</sup> [Github Repository med inspiration til at hente tariffer](#)

<sup>45</sup> ChargeOwner: Repræsenterer det netselskab som brugeren er tilknyttet.

## Dashboard grafer til Components

For at få indført mere single responsibility, prøvede vi at implementere nogle komponenter for de forskellige grafer på dashboardet. Dette endte dog med at være ret krævende at løse, og gav nogle nye problemer. I sidste ende endte vi med at stoppe udviklingen i den retning og rullede tilbage til den tidligere løsning.

## Retrospective

### Features

De tilføjede features til applikationen:

- **Energikilder:** Implementering af energikilde-funktionen. Data om den totale produktion i brugerens region hentes, og forbrugernes andel beregnes for at vise en gennemsnitlig oversigt over energiens oprindelse.
- **Tariffer og Afgifter:** Tilføjelse af tariffer og afgifter i elpris beregningerne for mere præcise og realistiske elpris grafer.
- **Dark Theme:** Implementering af dark theme for en mere behagelig brugeroplevelse og visuel forbedring af applikationen.

### Udfordringer

I dette sprint har vi haft nogle udfordringer. Philip havde brugt sin Energinet konto indtil nu fordi han havde nogle flere privileges i Azure som vi kunne drage fordel af, men nu hvor den konto er blevet off-boarded og han skal bruge sin EDU mail, er vi stødt på det problem at den konto af en eller anden årsag er blocked. Han kan således hverken tilgå vores Azure resource group eller starte projektet selv lokalt på grund af at han skal have adgang til key vaulten i ressourcegruppen.

Vi har også haft nogle udfordringer med external services i den periode her. Energi Data Service har været nede et par gang, ElOverblik har også haft problemer, og ligeledes

har Azure DevOps været nede så vi ikke har kunne pull eller push vores branches. Selvfølgelig da vi så på samme dag valgte at bare lave modeller, siden vi ikke kunne andet. Men så var vores modelværktøj også nede... Man kan altså godt sige for dette sprint at vi har haft lidt udfordringer.

Vi havde også nogle udfordringer med at opdatere vores hosted hjemmeside. Vi kunne alle køre vores applikation lokalt uden problemer, men når vi skulle sync det på azure så fik vi en fejl som ikke gjorde det muligt. Som følge af dette prøvede vi at reconnect til webapp'en, og her løb vi så også ind i flere problemer. Efter et godt stykke tid, lykkedes det dog gruppen at få det op og køre igen.

## Forbedringer

I det vi har haft mange udfordringer i dette sprint med eksterne services, kunne det være vi burde lave en backup plan, hvis vi engang igen løber ind i det problem. For eksempel kunne man gemme nogle datasæt, så man ikke er lige så afhængig af eksterne data, når man tester.

Vi har også planlagt at optage en gennemgang af applikationen, så vi har den som backup hvis der nu skulle være en service nede den dag vi har eksamen.

## Succeser

I dette sprint har vi haft flere vigtige succeser:

- **Energikilder:** Vi har udviklet en funktion, der viser brugerens energiforbrug i forhold til regionens samlede produktion.
- **Tariffer og afgifter:** Spotpris beregninger inkluderer nu daglige tariffer og afgifter, hvilket giver mere præcise elpriser.
- **Dark Theme:** Applikationen er opdateret med et mørkt tema for en bedre brugeroplevelse.

- Problemløsning: Trods udfordringer med eksterne services og Azure-ressourcer fik vi applikationen op at køre igen gennem godt samarbejde.

## Sprint 4: 25. Nov- 6. Dec

### Sprint Plan

For dette sprint skal vi have opdateret vores energikilde graf så dens loading time bliver meget hurtigere, det tager på nuværende tidspunkt alt for lang tid (~7 sekunder) før dashboardet kan loades, fordi den stadig er igang med at få API kaldet og behandle data'en.

Udover dette skal vi også have opdateret Elprisen, så den er tættere på realiteten. På nuværende tidspunkt er der ikke medregnet diverse tariffer og afgifter. Spotprisen på el har også været hentet fra ElPrisenLigeNu, og skal fremadrettet hentes fra Energi Data Service, da det er en mere troværdig kilde af data. Vi skal have refaktoreret noget af koden, herunder skal OperationResult og ResponseHandlers omskrives.

Der er også noget kode som ikke længere bliver brugt, der skal fjernes. Vi skal også have lavet noget mere modellering af systemet for at kunne dokumentere, hvordan diverse dele hænger sammen.

Vi skal også have lavet nogle unit tests til dele af koden, som har mange udregninger, da vi tænker der kan være meget tid at spare der.

## Aktiviteter

### Refaktorering af kode

Vi har gennemført en omfattende refaktoreringsproces for at forbedre kodens kvalitet, struktur, testbarhed. En væsentlig del af arbejdet har været en omskrivning af `OperationResult` og `ResponseHandlers`.

Derudover har vi fjernet ubrugt og redundant kode, hvilket har resulteret i en mere ren og effektiv kode. I forbindelse med refaktoreringsarbejdet har vi også lagt vægt på at forbedre læsbarheden og optimere eksisterende logik, så koden er lettere at forstå.

### Performance forbedringer

Indhentningen af vores energikilder har været en langsom proces. Udfordringen var, at rapporten blev hentet hver gang dashboardet blev initialiseret, hvilket resulterede i en ventetid, før dashboardet kunne bruges. For at løse dette problem har vi ændret processen, så rapporten nu hentes ved programmets opstart, og alle data gemmes i en liste. Dette betyder, at opstarten tager lidt længere tid, men dashboardet bliver til gengæld hurtigt tilgængeligt efterfølgende.

### Kvalitetsforbedringer

- Vi fik i dette sprint forbedret elprisen på dashboardet, ved at inkludere tariffer og afgifter. Der blev også skiftet hvor vi henter elspotprisen fra. Tidligere hentede vi fra `ElPrisenLigeNu` og nu henter vi fra `Energi Data Service` som er en mere troværdig kilde af data.
- Vi har implementeret en feature som giver brugeren mulighed for hurtigt at vælge en af tidsperioderne: i går, sidste 7 dage, sidste måned eller dette år.

## Opsætning af logging med Application Insights

Vi har sat logging op med application insights, så man har mulighed for at se alle logs direkte i application insights. Vi har særligt haft fokus på det, i forhold til at kunne logge, om en specifik background-task har kørt som forventet. Der er også mulighed for at se andre logs, så som error, opstart og nedlukningslogs.

Man kan query efter specifikke logs, for eksempel efter vores background task logs<sup>46</sup>

## Dokumentation og modellering

- Applikationsoverblik: Vi har lavet en model som giver et overblik over systemet og hvordan det er kædet sammen. Vores systemafhængigheds model<sup>47</sup> viser applikationen delt op i UI, Services, Utilities, Handlers og de eksterne API'er. Her bliver der vist de forskellige relationer mellem komponenterne.
- Vi har arbejdet videre med det, der oprindeligt var planlagt som en domænemodel. Undervejs blev det dog tydeligt, at en traditionel domænemodel ikke gav tilstrækkelig indsigt i systemet, da nogle dele for eksempel ikke havde nogen relationer. Derfor har vi tilpasset modellen, så den bedre afspejler systemet, selvom vi ikke længere vil betegne den som en domænemodel. Modellen er fortsat under udvikling.

---

<sup>46</sup> Bilag: Logging med Application Insights

<sup>47</sup> Se kapitel: Design, Low-Level, Systemafhængigheds Model

## Encrypt/decrypt

Vi har udviklet en metode, der krypterer og dekrypterer brugerens API-nøgle, som gemmes sikkert i databasen. Dette er vigtigt for at overholde GDPR og sikre, at følsomme data forbliver beskyttede.

### Kryptering:

1. Når en API-nøgle skal gemmes, bruger vi AES<sup>48</sup>, som er en stærk og sikker krypteringsalgoritme.
2. Før krypteringen genererer vi tilfældige salt- og IV-værdier(initialization vector). Disse sikrer, at selv hvis samme nøgle krypteres flere gange, vil resultatet altid være unikt.
3. Vi bruger en PassPhrase, som fungerer som nøgle til krypteringen. PassPhrasen er en tilfældigt genereret streng af tegn.
4. Data krypteres ved hjælp af AES med en nøgle på 128 bit, som genereres sikkert ud fra PassPhrasen ved hjælp af metoden PBKDF2<sup>49</sup>.
5. Den krypterede tekst gemmes sammen med salt- og IV-værdierne i databasen.

### Dekryptering:

1. Når en API-nøgle skal bruges, hentes den krypterede tekst fra databasen sammen med salt- og IV-værdierne.
2. Ved hjælp af den samme PassPhrase og PBKDF2 genskabes den nøgle, der blev brugt til at kryptere data.
3. AES-dekryptering bruges til at gendanne den oprindelige API-nøgle.

---

<sup>48</sup> AES: Advanced Encryption Standard

<sup>49</sup> PBKDF2: Password-Based Key Derivation Function 2



**PassPhrase:**

- For at sikre, at PassPhrasen holdes hemmelig, gemmes den sikkert i Azure Key Vault, som er en tjeneste til håndtering af følsomme nøgler og værdier.
- Azure Key Vault sikrer, at PassPhrasen ikke eksponeres i koden eller applikationen.

**Hvorfor er det vigtigt?**

Denne proces sikrer, at brugerens API-nøgle altid er krypteret, både når den gemmes og bruges. Selv hvis databasen kompromitteres, kan nøglen ikke genskabes uden adgang til både de unikke salt/IV-værdier og PassPhrasen fra Azure Key Vault. Dermed beskytter vi data i overensstemmelse med GDPR og moderne sikkerhedsstandarder.

## Retrospective

### Features

De tilføjede features til applikationen:

- Implementering af en feature, der giver brugeren mulighed for hurtigt at vælge specifikke tidsperioder (i går, sidste 7 dage, sidste måned, dette år).
- Implementering af automatiske tests i Azure Pipelines for at sikre kvaliteten af koden og forhindre regressionsfejl.

## Udfordringer

I dette sprint havde vi nogle problemer med nogle bugs som begyndte at vise sig. For elforbruget havde vi lige pludselig et problem med at labels ikke var korrekte. Der gik lidt tid før vi fik løst problemet, men det mest bekymrende var at det var opstået ud af det blå, og vi er usikre på om det er på grund af ElOverblik, som har haft en ændring i deres data/api-kald eller om det bare er fordi vi har lavet en fejl med en merge conflict.

Vi har igen i dette sprint haft lidt problemer med de eksterne datakilder, specifikt Energi Data Service, som ikke altid er samarbejdsvillige.

Der er desværre også stadig problemer med Philips konto til Energinets portal, dette betyder derfor stadig at han ikke har haft mulighed for at teste applikationen lokalt og ikke har adgang til vores Azure resource group. Energinets Service desk er stadig i gang med at kigge på problemet, men det begynder at blive lidt stramt med den resterende tid af projektet.

## Forbedringer

Vi har oplevet udfordringer med de eksterne API'er og services som Energi Data Service. For at minimere afhængigheden af deres stabilitet, bør vi overveje at implementere en backup løsning eller implementere mere caching. Med vores nuværende backlog kan det nok ikke nås nu. Men det er et emne til videreudvikling og hvis applikationen skulle gå live.

## Succeser

I dette sprint har vi haft flere vigtige succeser:

- Performanceforbedringer: Dashboardet loader nu næsten øjeblikkeligt ved at hente data ved programmets opstart og cache det i hukommelsen.
- Kvalitetsforbedringer i Elprisen: Tariffer og afgifter er blevet inkluderet, og data hentes nu fra Energi Data Service for større præcision.

- Refaktoring og Optimering: Vi har omskrevet *OperationResult* og *ResponseHandlers*, fjernet ubrugt kode og forbedret kodens struktur og læsbarhed.
- Logging med Application Insights: Logging sikrer nu bedre overvågning af baggrundsopgaver, fejl og systemevents.
- Modellering og dokumentation: Vi har udviklet en systemoversigt model, der giver klarhed over applikationens komponenter og relationer.
- Sikkerhed: API-nøgler krypteres sikkert med AES og opbevares i Azure Key Vault for at overholde GDPR og sikre følsomme data.
- Unit Tests: Vi har skrevet unit tests for vigtige beregninger og sat dem til at køre automatisk i vores Azure Pipelines. Dette sikrer høj kodekvalitet og reducerer risikoen for fejl.<sup>50</sup>

## Sprint 5: 9. Dec- 20. Dec

### Sprint Plan

I det kommende sprint fokuserer vi på at færdiggøre applikationen, så vi kan skifte til fuld fokus på rapporten. Vi skal tilføje en ny funktion til dashboardet, der giver brugerne mulighed for at sammenligne deres tidligere forbrug, eksempelvis sammenligning måned for måned fra sidste år, med deres nuværende årsforbrug.

Derudover vil vi undersøge muligheden for at implementere gamification<sup>51</sup>, som kan motivere brugerne til at reducere deres forbrug. Det kan for eksempel være en indikator, der viser en procentvis ændring i forbrug over de seneste 30 dage.

Vi planlægger også at tilføje informative statistikker i toppen af dashboardet. Disse skal inkludere data om det samlede forbrug for indeværende år samt det gennemsnitlige månedlige forbrug.

---

<sup>50</sup> Referer til Test kapitlet efter sprints

<sup>51</sup> Gamification: Brugen af spilelementer og principper som konkurrence, belønning og progression for at gøre oplevelser mere engagerende.

I sprintets sidste uge vil vi skifte fokus til rapportskrivning og opdatering af modellerne, så alt kan blive klart til den endelige aflevering.

## Aktiviteter

### Gamification

Vi har tilføjet en oversigtsrække over vores elforbrugsgraf, som fremhæver nogle nøgletal for brugerens elforbrug. Her kan man se den samlede mængde kWh, man har brugt i år, ens gennemsnitlige månedlige forbrug, samt hvor stor en procent forskel der har været i dit forbrug de seneste 30 dage sammenlignet med de foregående 30 dage. Disse nøgletal giver et hurtigt overblik over ens forbrug, og kan være med til at hjælpe med at motivere brugeren til at sænke sit forbrug.

### Elforbrug sammenligning

I dette sprint har vi udviklet en ny funktion til elforbrugs grafen, der giver brugerne mulighed for at se, hvor meget strøm de brugte i den inspicerede periode for et år siden. Funktionen gør det muligt at sammenligne elforbruget helt ned til daglig basis.

### Energi Indsigt

Vi har udviklet en ny side kaldet "Energi Indsigt", hvor brugerne kan få indsigt i, hvad forskellige dagligdags aktiviteter koster baseret på de aktuelle elpriser. Eksempler inkluderer prisen på et 10-minutters varmt brusebad, opladning af en elbil eller en opvask. Funktionen gør det lettere for brugerne at træffe informerede beslutninger om deres energiforbrug.

## Bug: elforbrug graf labels

Vi har arbejdet på at løse et problem med vores elforbrugsgraf, hvor labels blev vist forkert. Problemet skyldtes, at grafen blev indlæst med data fra det foregående år først, hvilket resulterede i forkerte datoer på labels. Vi løste dette ved at ændre rækkefølgen, det bliver indlæst, så grafen nu indlæser den korrekte data først, og labels bliver sat korrekt.

## Prioritering af rapportskrivning

Grundet tidspres har vi måtte prioritere arbejdet med vores rapport over mindre fejlrettelser i systemet. Vi har haft fokus på at balancere arbejdet mellem udvikling og dokumentation for at sikre, at vi leverer et kvalitetsprodukt samtidig med en velformuleret rapport.

## Modeller og diagrammer

Vi har i dette sprint også fået lavet opdateringer af diverse modeller og diagrammer som vi har lavet igennem projektet. Her med fokus på interaktionsdiagrammer og systemafhængigheds modellen.

## Retrospective

### Features

De tilføjede features til applikationen:

- Implementering af nøgletal, der viser samlet kWh for nuværende år, gennemsnitligt månedligt forbrug og forskellen i forbrug de seneste 30 dage.
- Implementering af en *Energi indsigt* side hvor brugeren kan se hvor meget det vil koste med dagens elpriser, at tage f.eks. et 10 minutters varmt brusebad, at lade elbilen eller starte opvasken.

- Tilføjelse af feature til elforbrugsfunktionalitet, så brugeren nu kan se hvor meget strøm de brugte samme dag, måned eller året før. Dette giver brugeren mulighed for at sammenligne perioder side om side.

## Udfordringer

I dette sprint har vi oplevet yderligere problemer med vores elforbrugsgraf, hvor labels blev vist forkert. Problemet opstod på grund af måden, grafen blev indlæst på i vores dashboard. Den var sat til at indlæse grafen for det foregående år først, hvilket medførte, at labels blev vist for sidste års datoer i stedet for de aktuelle datoer. Vi løste problemet ved at indlæse den korrekte graf først, så labels nu bliver sat korrekt.

Vi har været under tidspres, da vi har mange ting, vi gerne vil arbejde på i vores produkt, men vi er nødt til at fokusere på rapporten i stedet. Dette betyder, at vi må undlade at rette mindre bugs og fejl i systemet.

I anledning af rapportskrivningen har vi i gruppen haft noget debat angående hvordan rapporten skulle sættes op, som har skabt tvivl om rapportens opsætning. Dette løste vi ved at snakke med vores vejleder for at komme på rette spor igen.

## Forbedringer

Som nævnt tidligere har vi oplevet tidspres i forbindelse med rapporten, og derfor skal vi fremover blive bedre til at prioritere den i forhold til selve produktet.

I forhold til vores tvivl omkring rapporten burde vi have kontaktet vores vejleder tidligere. Dette ville have givet os bedre muligheder og tid til at tilpasse os, hvis der var noget, der ikke levede op til forventningerne.

## Succeser

Vi var meget motiverede i forbindelse med implementeringen af gamification, hvilket i sig selv er en succes. Det bidrog også til, at vi fik mange flere idéer til applikationen og nye måder at udvikle den på.

Det var desuden en stor succes at holde møde med vores vejleder, hvor vi fik svar på de spørgsmål, vi havde angående rapporten og vores modeller.

## Sprint 6: 23. Dec- 3. Jan

### Sprint Plan

Planen for det sidste sprint er at tage en kort juleferie for at samle energi og vende tilbage med et friskt perspektiv på rapporten. Her vil vi rette op på formalia og korrektur, samt foretage eventuelle sidste justeringer, der måtte være nødvendige inden afleveringen.

### Aktiviteter

#### Rapport

I løbet af dette sprint har vi haft fokus på at færdiggøre rapporten. Vi har arbejdet med at gennemføre korrektur, sikre en god opsætning samt finpudse de enkelte kapitler. Derudover har vi dedikeret tid til at rette op på formalia og sikre, at rapporten lever op til de gældende krav, så den står klar til aflevering.

#### Modeller

I løbet af det sidste sprint har vi også fokuseret på at finpudse de sidste detaljer på vores modeller. Vi har gennemført de nødvendige opdateringer. Samtidig har vi fået færdiggjort de resterende modeller.

## Produkt

Som en del af det sidste sprint har vi også lavet de sidste justeringer af produktet. Dette arbejde har primært bestået af lidt refaktorering. Derudover har vi omdøbt nogle variabler og klasser for at sikre bedre navngivning samt fjernet overflødige og ubrugte klasser.

## Retrospective

### Features

Der blev ikke tilføjet nogle features i det sidste sprint af projektet.

### Udfordringer

I det sidste sprint har vi ikke haft de større udfordringer. Arbejdet har primært været præget af træthed, som en naturlig følge af processen med at gennemlæse rapporten igen og igen.

### Forbedringer

En mulig forbedring kunne have været at reducere træthed i gruppen ved at i stedet for at alle i teamet læste hele rapporten fra start til slut, kunne vi have opdelt arbejdet i sektioner. For eksempel kunne en person have fokuseret på den første tredjedel af rapporten, en anden på den midterste del, og en tredje person på den sidste del. Dette kunne have lettet arbejdsbyrden, men også gjort processen mere effektiv ved at give hver person mulighed for at dykke mere detaljeret ned i deres del. En afsluttende gennemlæsning kunne derefter sikre sammenhæng på tværs af hele rapporten.

Hvis korrekturlæsningen var blevet fordelt over flere sprints, kunne det have mindsket presset i den sidste fase. Det ville også have givet mulighed for at opdage og rette



stavefejl tidligere. Dog havde vi meget omskrivning her til sidst i projektet. Så det kunne måske bare være endt ud i dobbelt arbejde i stedet.

## Succeser

For det sidste sprint af projektet havde vi en fin række generelle succeser med rapportskrivning og modellering som kan ses i 'Aktiviteter', og den selvfølgelig største succes: at vi kan kalde projektet og rapporten for færdig.

## Test

### Unit Tests

Vi har skrevet nogle unit tests til dele af vores kode, som vi vurderede til at have mest brug for det. Testprojektet er oprettet som et separat projekt i samme solution.

Vi har mange små udregninger til forskellige dele af systemet, som kan være uoverskuelige, blandt andet vores udregninger af produktpriser for elleverandører og energikilder. Unit tests er med til at sikre at de forskellige dele som arbejder sammen giver det forventede resultat. Det kan i længden spare os meget tid under fejlsøgning.

I løbet af udviklingen af unit tests, er noget kode blevet omskrevet til mindre dele, for at gøre det nemmere at teste.

```
[Fact]
0 references | salgaard, 31 days ago | 1 author, 3 changes
public void GetValidProductPrice_Should_ReturnPrice_When_AnnualConsumptionIsInRange()
{
    // Arrange
    var product = new Product
    {
        ProductPrices = new List<ProductPrice>
        {
            new ProductPrice
            {
                MinimumConsumption = 1000,
                MaximumConsumption = 5000,
                Matrix = new List<Matrix> { new Matrix { VolumeFrom = 0, VolumeTo = 5000, Amount = 200.0 } }
            },
            new ProductPrice
            {
                MinimumConsumption = 5001,
                MaximumConsumption = 10000,
                Matrix = new List<Matrix> { new Matrix { VolumeFrom = 5001, VolumeTo = 10000, Amount = 300.0 } }
            }
        }
    };
    double annualConsumption = 4000;

    // Act
    var result = ProductDetailUtilities.GetValidProductPrice(product, annualConsumption);

    // Assert
    Assert.NotNull(result);
    Assert.Equal(200.0, result.Matrix[0].Amount);
}
```

På billederne ovenover, er der en af vores unit tests som sikrer at metoden `GetValidProductPrice()` returnerer null, hvis årligt forbrug ikke er inden for produktets maximum og minimum.

I vores Azure pipelines, kører vi altid vores tests som et krav for gennemførelse<sup>52</sup>. Det gør, at vi ikke er nødt til hele tiden at køre dem manuelt<sup>53</sup>, men i stedet kan sikre at det sker automatisk.

På alle pull requests, er det et krav at pipelinen er gennemført med succes. Vores tests bliver dermed altid automatisk kørt<sup>54</sup>, hvilket sikrer en højere kvalitet og pålidelighed af den kode som vi merger ind i development og main.

<sup>52</sup> Bilag: Automatisk Pipeline Test Run Results

<sup>53</sup> Bilag: Manuelt Unit Test Run

<sup>54</sup> Bilag: Pipeline Run Tests Code

## Performance Tests

Vi har integreret Azure application insights, så vi kan se forskellige ting, for eksempel performance af page loads, database loads eller http requests. Det er blandt andet muligt at se hvilke requests kommer fra vores cloud miljø kontra lokale miljø, da vi har lavet roller som indikerer hvor de er fra.

I bilag<sup>55</sup> ses et billede som viser HTTP requests sendt fra applikationen, hvor det er muligt at se hvor lang tid requesten tager og hvor mange af dem som er sendt. Det bruger vi f.eks. til at se hvad der får en bestemt side til at loades langsomt, og om tiden det tager at loades en side er inden for det vi vurderer til at være acceptabelt<sup>56</sup>.

Vi har DeclarationTransmissionGridMix(Klima Rapport) API-kaldet, som gjorde indlæsningen af Dashboard siden meget langsom. Vi valgte derfor at lave en background service<sup>57</sup> som bliver tilføjet som en hosted service i Program.cs. Den kører ved midnat hver dag og gemmer data'en i ClimateReportService i en liste, men da vi skulle hente 2 versioner af klima rapporten(en for DK1 og en for DK2), ligger de i hver deres liste i en dictionary. Det er en singleton service, så der vil kun være en instans af servicen og alle brugere kommer til at hente den samme liste. Det er perfekt, da klimarapporten ikke er for hver person, men generelt for regionen.

I stedet for at kalde api'en hver gang man indlæser siden, henter den så bare den liste som er gemt i servicen.

Som kan ses på billedet, er der nogle gange enkelte andre requests som tager utroligt lang tid. Nogle af dem er hvor man henter rigtig meget data, og andre kan være fordi der skete en fejl under kaldet. Der er mulighed for at lave yderligere caching for at optimere indlæsningen.

---

<sup>55</sup> Bilag: Performance Test

<sup>56</sup> Se kapitel - KPI: Max Responstid for API-Kald

<sup>57</sup> Elvission/Services/ClimateReportBackgroundService.cs

## Brugertest

For at sikre, at vores applikation lever op til brugernes behov og forventninger, har vi valgt at gennemføre brugertests. Disse tests giver os værdifuld indsigt i, hvordan brugerne oplever og interagerer med applikationen. På den måde kan vi finde frem til, hvad svagheder applikationen har, som vi udviklere kunne have overset.

Test #1		
Opgave	Resultat	Konklusion
#1 Du skal oprette en bruger på ElVision med din ElOverblik API nøgle.	Brugeren havde lidt svært ved at finde hvor de kunne oprette en konto, herefter gik det fint også med API nøgle guiden, udover at trin 1 måske var lidt blandet sammen med headeren, da brugeren overså trin 1.	Brugeren fik relativt hurtigt løst opgaven. Det kunne dog være at trin 1 skulle gøres mere tydeligt ved at adskille det visuelt fra overskriften, fx med en line break. Tilføj evt. en tydeligere "Opret konto"-knap på forsiden, da brugeren ikke kunne finde denne.
#2 Du skal undersøge dit elforbrug. Find dit elforbrug for de sidste 30 dage.	Brugeren gjorde dette ved at vælge en periode i datepickeren på 30 dage, istedet for at bruge 'hurtig valg' funktionen.	Brugeren valgte at vælge perioden manuelt med datepickers fordi de forså "hurtigvalg" funktionen.  Overvej at fremhæve eller gøre "hurtigvalg" mere

		intuitivt og synlig.
<i>#3 Du skal nu undersøge dit elforbrug for en bestemt periode og et bestemt tidsinterval. Tidsinterval i timer, med start dato d. 1 Marts og slut dato d. 5 Marts</i>	Brugeren gjorde dette uden problemer.	Ingen problemer med denne opgave, ingen forbedringsforslag.
<i>#4 Du skal nu prøve at se din forbrugsændring, har der været en forbedring eller forværring i dit forbrug?</i>	Brugeren fandt forbrugsændringen hurtigt, men begyndte at prøve og se om de kunne ændre for at se forbrugsændring perioden ved at skifte periode.	<p>Brugeren fandt forbrugsændringen uden problemer, men forventede mulighed for at ændre perioden for at analysere forbrugsændringer yderligere.</p> <p>Man kunne tilføje en funktion, der gør det muligt at ændre perioden for forbrugsændringer direkte, eller man kunne gøre det klart at perioden er fastlåst, med en forklarende tekst eller indikator.</p>

<p><i>#5 Du skal nu kontrollere elprisen for i dag, hvad er prisen pr. kWh kl 17:00?</i></p>	<p>Brugeren startede på Elforbrugs siden, men scrollede ikke ned til Elpris grafen, men skiftede først til en anden side. Da de gik tilbage til oversigten, scrollede brugeren dog direkte ned til den.</p>	<p>Brugeren fandt elprisen, men overså oprindeligt elpris grafen på siden og gik derfor unødigt til en anden side, inden de scrollede ned på siden til grafen.</p> <p>Det er svært at se hvordan dette kunne forbedres, udover måske at gøre Elforbrugs grafen mindre så brugeren kan se elprisgrafen uden behov for at scroll ned.</p>
<p><i>#6 Du skal nu se på dine energikilder, hvor stor en procent er atomkraft? Og hvor mange kWh udgør atomkraft?</i></p>	<p>Brugeren fandt hurtigt grafen men troede først at sol var atomkraft fordi farven var lidt den samme. Brugeren så dog hurtigt selv fejlen, da de hoverede over energikilden og kunne se både % og kWh uden problemer.</p>	<p>Brugeren fandt energikilde grafen hurtigt, men blev kortvarigt forvirret af farverne, da sol og atomkraft havde lignende nuancer. Brugeren kom dog hurtigt tilbage på sporet ved at bruge hovedfunktionen.</p> <p>Overvej at bruge mere tydelige farver eller ihvertfald skifte Sol og Atomkraft så de er lettere</p>

		at skelne mellem.
<i>#7 Du skal nu prøve at se, hvad din estimerede gennemsnitspris ville være på en Energi Fyn plan.</i>	Brugeren var først lidt fortabt i hvor de skulle gå hen for at lede efter det og først tjekkede Energi indsigt. Bagefter gik brugeren dog til Elleverandør siden og brugte hurtigt søgefunktionen til at finde Energi Fyn, men brugeren missede dog knappen til at gøre prisen ud fra brugerens forbrug.	<p>Brugeren oplevede forvirring om, hvor de skulle starte, og overså knappen for at beregne prisen baseret på deres forbrug.</p> <p>Man kunne gøre det tydeligere med <i>Beregn estimeret pris</i> knappen.</p>
<i>#8 Du skal nu prøve at se hvad det ville koste at lade din Tesla Model 3 med den nuværende elpris.</i>	Brugeren startede med at sige "Fisk", men efterfølgende gik direkte på Energi Indsigt og fandt ret hurtigt hvad det ville koste at lade en Tesla Model 3 nu.	Opgaven gik næsten som håbet, ingen anbefalinger på forbedringer ud fra testen.

## Udrulning

Vores udrulning vil fungere som en *staged deployment*. Dette betyder, at vi opsætter et testmiljø (staging), der er en komplet kopi af produktionsmiljøet. Her kan brugere teste systemet så grundigt som muligt for at identificere og rette fejl og bugs.

Efter testfasen er afsluttet og valideret, kan applikationen udrulles til det egentlige produktionsmiljø, hvor den er tilgængelig for alle brugere.

Denne proces giver os mulighed for at minimere fejl og mangler i applikationen ved at validere funktionalitet og ydeevne i et miljø, der ligner det endelige produktionssetup

## Vedligeholdelse

### Teknisk vedligeholdelse

#### Bug fixes

For at opretholde applikationens funktionalitet skal vi løbende rette eventuelle fejl, der opstår, når applikationen går live.

#### Hold API integrationer opdateret

Som vi allerede har erfaret under projektet, kan der godt komme opdateringer til API og hvordan de fungerer, eller nye features hertil. Derfor er det vigtigt at vedligeholdelses-teamet holder sig opdateret på de brugte API'er og kan bringe applikationen up-to-date hurtigst muligt efter en API ændring.



## Sikkerheds vedligeholdelse

### Sårbarhedsscanning

Vi har allerede implementeret en kode analyseværktøj, som kører automatisk hver gang vi merger en branch til både development- og main. Denne proces sikrer, at vores applikation bliver løbende scannet for sårbarheder ved hver opdatering. På denne måde kan vi identificere og fikse sårbarheder i applikationen.

## Videre Udvikling

### Machine Learning

Ved brug af machine learning til at opfange mønstre i brugerens forbrug kan vi lave anbefalinger til forbrugeren. Har brugeren tendens til at lave mad hver dag omkring kl.18? Anbefal et billigere tidspunkt. Sætter de el bilen til opladning direkte når de kommer hjem fra arbejde? Anbefal at vente til at lade om natten. osv

### Hvad kostede forbruget?

Ved at bruge elprisen time for time med brugerens målte forbrug kan man beregne de samlede omkostninger for brugerens elforbrug i den forgangne uge for dagens timer. Ved at levere et mere præcist estimat af, hvad dette forbrug har kostet, kan man give brugeren bedre indsigt i deres omkostninger for el. Dette giver en mere realistisk forståelse af, hvilke forbrugsmønstre der har størst økonomisk betydning, og dermed kan brugeren identificere, hvor der er mulighed for besparelser.

## Energikilders produktionssted

Energikilder api kaldet gav ikke bare information om hvor meget var produceret af hver type energi, men også hvorfra f.eks. Sverige, Norge, Holland, Tyskland osv. Det kunne være meget interessant for brugeren at se hvor deres energi kommer fra.

## Samarbejde med ældre pleje

Applikationen kunne tilpasses, så ældre borgere får mulighed for at give samtykke til at dele deres el forbrugsdata med relevante parter såsom hjemmeplejen eller pårørende. Dette ville give kommunen og plejepersonalet et værktøj til at holde øje med de ældres trivsel og hjælpe med at sikre, at de har det godt.

### 1. **Overvågning af madlavning:**

Hvis der ikke registreres elforbrug omkring aftensmad, f.eks. ved brug af komfur, kan det indikere, at den ældre ikke har fået lavet mad. Dette kan være et tegn på, at de har brug for hjælp eller støtte.

### 2. **Overvågning af daglige rutiner:**

El Forbrugsdata kan hjælpe med at identificere ændringer i de ældres daglige mønstre. Hvis elforbruget pludselig falder eller bliver uregelmæssigt, kan det være et tegn på, at noget er galt, f.eks. sygdom eller svækket tilstand.

### 3. **Hurtig identifikation af problemer**

Hvis der registreres usædvanligt højt elforbrug, kan det være et tegn på en fejl som et efterladt komfur eller et defekt elektrisk apparat. Dette kan hjælpe med at forhindre potentielt farlige situationer som brand eller kortslutning.

### 4. **Automatiske advarsler**

Systemet kan sættes op til at sende automatiske notifikationer til plejepersonale eller pårørende, hvis der registreres afvigelser i elforbruget, som kan indikere, at noget er galt.

## ChatBot

Udvikling af en chatbot, der kan hjælpe brugerne med at få direkte svar på spørgsmål om elforbrug, som for eksempel: *"Hvilke ting i mit hjem bruger mest strøm?"* eller *"Trækker mit TV stadig strøm, selv når det er slukket, men stadig tilsluttet?"* ChatBotten ville fokusere på relaterede spørgsmål om energiforbrug og tilbyde rådgivning eller information, som brugerne kan finde nyttig i deres hverdag.

## Gamification

For applikationen ville vi gerne have implementeret nogle former for gamification. Denne implementering ville være med til at fremme engagement i brugerne for at motivere dem til at skære ned på deres el forbrug. Vi har allerede implementeret et enkelt feature til dette, som giver brugeren et procenttal, som fortæller om de har brugt mere eller mindre strøm de sidste 30 dage, i relation til de 30 forrige dage. Andre måder man kunne implementere gamification kunne være:

- Gennemsnit relativt til region (DK1, DK2). Hvis der kan findes noget gennemsnitsforbrug for regionerne og ud fra en lignende hus/lejlighedsstørrelse.
  - f.eks: *"Du bruger 15% mindre strøm end gennemsnittet i din region!"*
- Lad brugeren sætte specifikke mål som trackes i appen.
  - Reducerer forbrug med 10% den næste måned.
  - Reducerer årlige totale forbrug med 1000 kWh
- Notifikationer med anerkendelse af brugerens forbedringer. At give brugeren et klap på skulderen for deres indsats, kan være med til at fremme denne tilgang og sørge for at brugeren er motiveret til at lave forbedringer.
  - f.eks: *"Godt gået! Du har brugt 12% mindre strøm denne måned."*

- Med samtykke fra brugere kunne man også inkludere dem i et form for leaderboard, og her sammenligne brugere og deres nedskæring i forbrug ugentligt, månedligt og årligt.

## Opdatering af API key

Det er på nuværende tidspunkt kun muligt at fjerne og opdatere en API key ved at slette brugeren helt og genoprette sig i systemet igen. Der bør her implementeres en mulighed for at en aktiv bruger kan opdatere deres API nøgle til ElOverblik direkte i webappen, da der er mulighed for at brugerens API nøgle er udløbet (nøglerne er valid i 1 år). Denne feature burde ikke tage alt for meget udviklingstid. Vi valgte dog i gruppen at fokusere på andre opgaver, da den nuværende løsning er funktionel og acceptabel for et studieprojekt.

## Valg Af Tema

Mulighed for brugeren til at kunne vælge om man vil have dark eller light UI tema.

## Best Practices for modeller

I løbet af projektet har vi nogle gange haft lidt for meget fart på, for eksempel når det kom til at lave nogle klasser til serialization og deserialization af json data. Vores klasser er meget rodet og forvirrende, da vi bruger én klasse, både til json og business logikken, hvor vi i stedet skulle have lavet separate domæne klasser.

Det vi skulle have gjort er at lave DTO(Data Transfer Object) klasser til json formål og domæne klasser til business logik formål.

Det ville gøre at vi har en bedre opdeling af ansvar i vores modeller, og gøre det mere læseligt og håndgribeligt at arbejde med senere i koden.

## Genbrugelige Blazor Komponenter

I vores UI, har vi meget lange filer, som gør det uoverskueligt. Der kan man dele koden ud i mindre komponenter som både gør det mere overskueligt og læseligt, men også gør det nemmere at teste og arbejde videre på i fremtiden.

## Evaluering

I løbet af projektet er vi stødt på mange forskellige problemer, lige fra almindelige motivationsproblemer til rettighedsproblemer. Vi har formået at komme igennem alle problemer på den bedst mulige måde.

Et af de største problemer, vi mødte var da Philips Energinet konto blev off-boarded og hans UCL konto var af ukendte årsager blokeret på Energinets login tenant. Han havde således ikke adgang til vores Azure-miljø og derfor ikke på samme måde kunne hjælpe med udviklingen af selve applikationen. Dog vendte vi dette problem til en fordel, idet vi altid havde en, der kunne fokusere på modellerne og rapporten for de features, vi udviklede, samt tilgængelig til noget par-programmering. Dette har givet os den erfaring, at vi i fremtidige projekter bør være hurtigere til at gå i en anden retning i stedet for at regne og håbe på at virksomheden selv kan fikse det.

Dette havde selvfølgelig betydet at der ikke ville være ret meget samarbejde med virksomheden eller kunne drage fordel af deres ressourcer til projektet. Men til gengæld kunne vi have fuld produktionskraft gennem hele projektet. Dog kan det nok også konkluderes at det er et ret specielt scenarie vi er stødt på hvor hverken UCL support eller Energinets Service desk har kunne løse problemet i tide, eller forstå hvorfor det overhovedet var sket.

Vi har udviklet os meget inden for SCRUM og vores evne til at planlægge sprints, hvilket har gjort, at vi har følt os godt med i forhold til projektets tidsplan. Dette har hjulpet os med at håndtere projektet uden at føle os stressede eller pressede.

Vores evne til at planlægge udviklede sig markant, jo længere vi kom ind i projektet. I starten oplevede vi til tider et fald i motivationen, fordi vi løb tør for opgaver, før sprintet var afsluttet. Vi følte vi havde planlagt tilstrækkeligt, men nogle uger har simpelthen været for hurtigt færdige eller haft backlog items som ikke viste sig at være mulige. Men efterhånden som projektet skred frem, blev vi bedre til at planlægge og definere opgaver, hvilket øgede vores motivation og engagement i projektet som helhed.

## Konklusion

Vores applikation er udviklet for at hjælpe forbrugere med at opnå bedre overblik og forståelse af deres elforbrug. Ved at integrere data fra ElOverblik og Energi Data Service giver platformen indsigt i forbrugsmønstre gennem grafer og nøgletal. Applikationen gør det muligt for brugerne at få indsigt i deres forbrug, energikilder, elpriser og sammenligne leverandøraftaler, hvilket giver dem værktøjer til at træffe informerede valg og reducere både omkostninger og forbrug.

Information om forbrugerens energikilder bidrager til en mere bæredygtig tankegang, mens energi indsigt giver brugeren konkrete priser på dagligdags aktiviteter, hvilket gør deres elforbrug mere relaterbart. ElVision fremmer også øget gennemsigtighed i elmarkedet gennem leverandørsiden, hvilket tidligere har været svært tilgængeligt. Sammensætningen af disse funktioner hjælper forbrugere med at optimere deres energiforbrug og træffe mere bæredygtige valg.

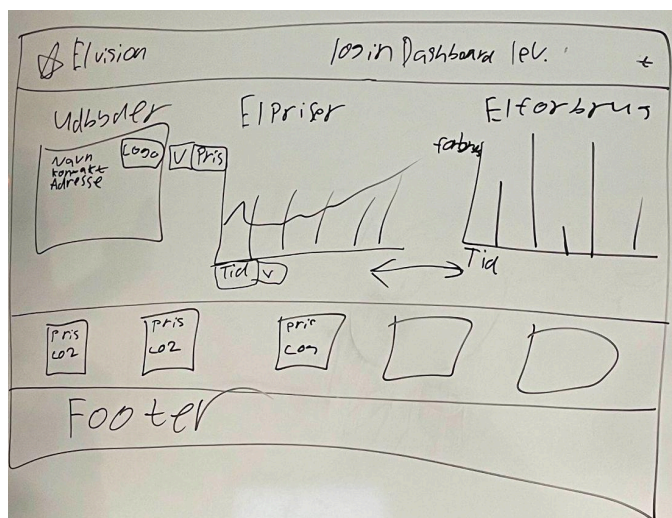
# Bilag

## Hosted Web Applikation

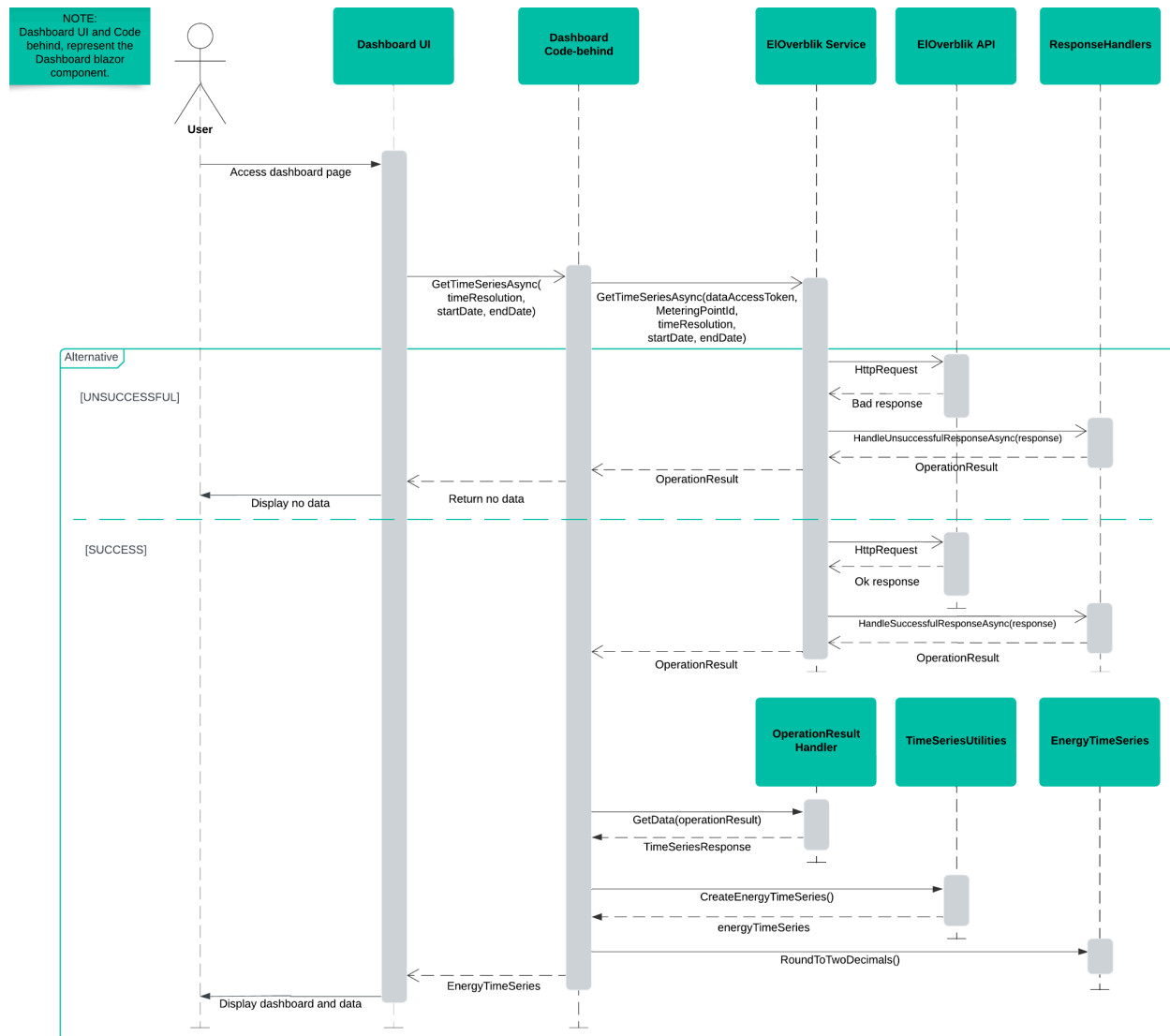
Bemærk at applikationen kan tage noget tid om at starte op første gang

<https://elvision-wa-gsecfufpf7fvf4e0.northeurope-01.azurewebsites.net/>

## UI First draft



# Interaktionsdiagram: Visning af Elforbrugs data





## Manuelt Unit Test Run

Ready 0 Warnings 0 Errors

Test	Duration	Tr
Tests (20)	799 ms	
Tests (20)	799 ms	
CalculateClimateReportTests (8)	263 ms	
CalculateReport_ShouldReturnCorrectUserShares	4 ms	
GroupClimateRecords_ShouldGroupRecordsCorrectly	1 ms	
NormalizeReportGroupCode_ShouldReturnExpectedCode (6)	258 ms	
NormalizeReportGroupCode_ShouldReturnExpectedCode(inputCode: "...)	< 1 ms	
NormalizeReportGroupCode_ShouldReturnExpectedCode(inputCode: "...)	< 1 ms	
NormalizeReportGroupCode_ShouldReturnExpectedCode(inputCode: "...)	258 ms	
NormalizeReportGroupCode_ShouldReturnExpectedCode(inputCode: "...)	< 1 ms	
NormalizeReportGroupCode_ShouldReturnExpectedCode(inputCode: "...)	< 1 ms	
NormalizeReportGroupCode_ShouldReturnExpectedCode(inputCode: "...)	< 1 ms	
GetDailyAverageTests (3)	274 ms	
GetDailyAverage_ShouldHandleSingleElspotPrice	2 ms	
GetDailyAverage_ShouldReturnCorrectAverages	272 ms	
GetDailyAverage_ShouldReturnEmptyList_WhenNoElspotPrices	< 1 ms	
PriceCalculatorHelpersTests (9)	262 ms	
BuildConsumptionProfilePercentage_Should_ReturnCorrectPercentage_...	< 1 ms	
BuildConsumptionProfilePercentage_Should_ReturnEqualDistribution_W...	< 1 ms	
BuildConsumptionProfilePercentage_ShouldReturnDefaultProfile_WhenH...	< 1 ms	
CalculateAnnualProductPrice_Should_CalculateCorrectPrice_WithFixedBill...	< 1 ms	
CalculateAnnualProductPrice_ShouldReturnCorrectPrice_ForVariableBilling	2 ms	
CalculateNetworkFactors_Should_CalculateCorrectFactors_When_TariffsA...	< 1 ms	
CalculatePriceFactor_ShouldReturnCorrectValue	< 1 ms	
GetValidProductPrice_Should_ReturnNull_When_NoPricesInRange	< 1 ms	
GetValidProductPrice_Should_ReturnPrice_When_AnnualConsumptionIs...	260 ms	

Group Summary

**Tests**

Tests in group: 20

⌚ Total Duration: 799 ms

**Outcomes**

✅ 20 Passed

## GitLeaks Analyse test

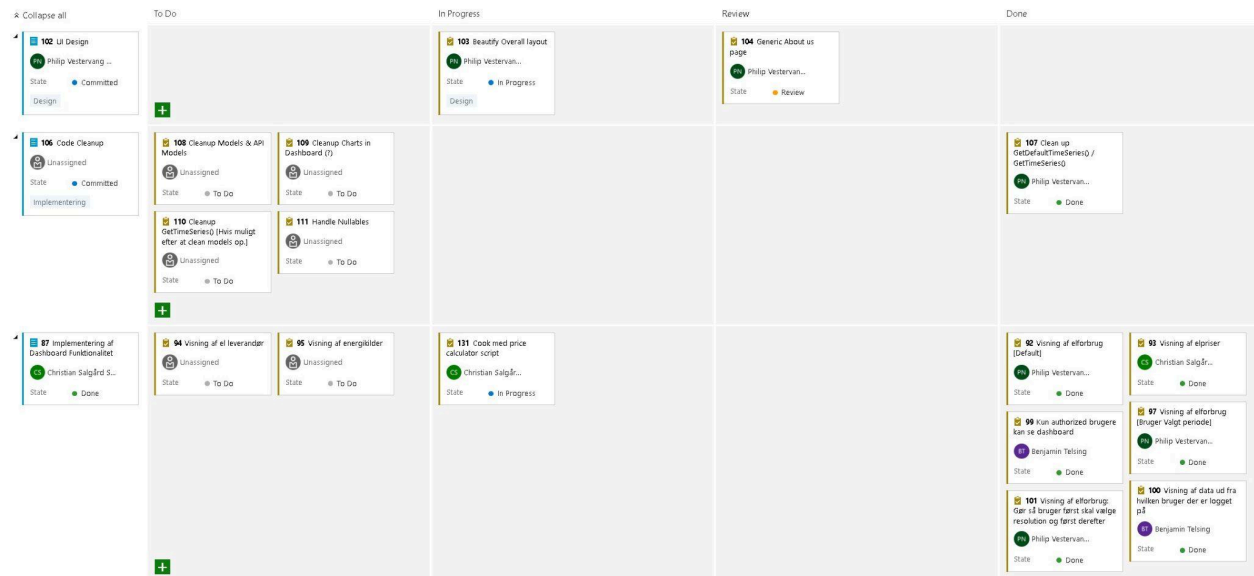
På det følgende billede, kan det ses hvor GitLeaks har fundet nogle secrets i vores kode. Bemærk at vi har sløret billedet for at beskytte sårbare informationer.

Summary Scans Code Coverage Mend Bolt

Filter by keyword

Path	Details ↑
▼ CSCAN0092 / CSCAN0043 4	
1 D:\a\1\s\EIVision\EIVision\appsettings.json	CSCAN0092 / CSCAN0043 has detected secret for file D:\a\1\s\EIVision\EIVision\appsettings.json. 3 User Id=E1Vision;Password=
1 D:\a\1\s\EIVision\EIVision\appsettings.json	CSCAN0092 / CSCAN0043 has detected secret for file D:\a\1\s\EIVision\EIVision\appsettings.json. 4 User Id=E1Vision;Password=
1 D:\a\1\s\EIVision\EIVision\bin\Debug\net8.0\appsettings.json	CSCAN0092 / CSCAN0043 has detected secret for file D:\a\1\s\EIVision\EIVision\bin\Debug\net8.0\appsettings.json. 3 User Id=E1Vision;Password=
1 D:\a\1\s\EIVision\EIVision\bin\Debug\net8.0\appsettings.json	CSCAN0092 / CSCAN0043 has detected secret for file D:\a\1\s\EIVision\EIVision\bin\Debug\net8.0\appsettings.json. 4 User Id=E1Vision;Password=
▼ generic-api-key 2	
1 D:\a\1\s\EIVision\EIVision\appsettings.json	generic-api-key has detected secret for file D:\a\1\s\EIVision\EIVision\appsettings.json. 5
1 D:\a\1\s\EIVision\EIVision\bin\Debug\net8.0\appsettings.json	generic-api-key has detected secret for file D:\a\1\s\EIVision\EIVision\bin\Debug\net8.0\appsettings.json. 5

# Scrum Board



## Logging med Application Insights

New Quer... \*...X +

Run

Time range : Last 24 hours

Limit : 1000

```
1 traces
2 | where message startswith "Climate Report Task has run"
3 | project timestamp, message
```

Results

Chart

timestamp [UTC] ↑↓	message
> 12/6/2024, 9:08:26.970 AM	Climate Report Task has run at 12/06/2024 10:08:26
> 12/6/2024, 8:55:26.566 AM	Climate Report Task has run at 12/06/2024 09:55:26
> 12/6/2024, 8:55:04.626 AM	Climate Report Task has run at 12/06/2024 09:55:04
> 12/6/2024, 8:55:02.518 AM	Climate Report Task has run at 12/06/2024 09:55:02
> 12/6/2024, 8:55:00.422 AM	Climate Report Task has run at 12/06/2024 09:55:00
> 12/6/2024, 8:54:58.318 AM	Climate Report Task has run at 12/06/2024 09:54:58
> 12/6/2024, 8:54:56.214 AM	Climate Report Task has run at 12/06/2024 09:54:56
> 12/6/2024, 8:54:54.104 AM	Climate Report Task has run at 12/06/2024 09:54:54
> 12/6/2024, 8:54:51.996 AM	Climate Report Task has run at 12/06/2024 09:54:51
> 12/6/2024, 8:54:49.903 AM	Climate Report Task has run at 12/06/2024 09:54:49

# Automatisk Pipeline Test Run Results


Summary

1 Run(s) Completed ( 1 Passed, 0 Failed )

20

Total tests

+20



20

0

0

Passed

Failed

Others

100%

Pass percentage

↑ 100%

24s 597ms

Run duration ⓘ

↑ +24s 597ms

0

Tests not reported

Bug

Link

Test run

Column Options

Filter by test or run name

Tags

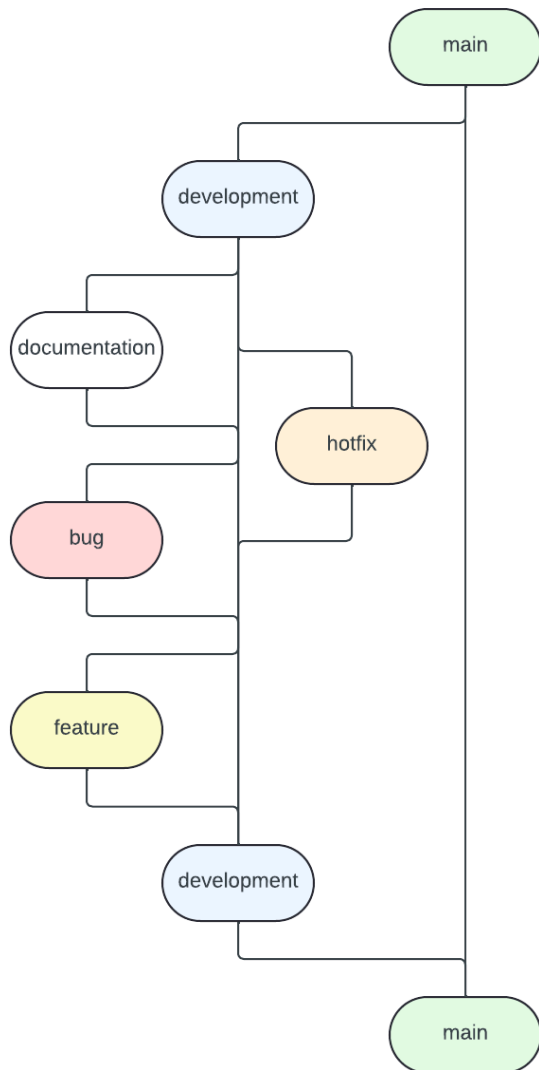
Test file

Owner

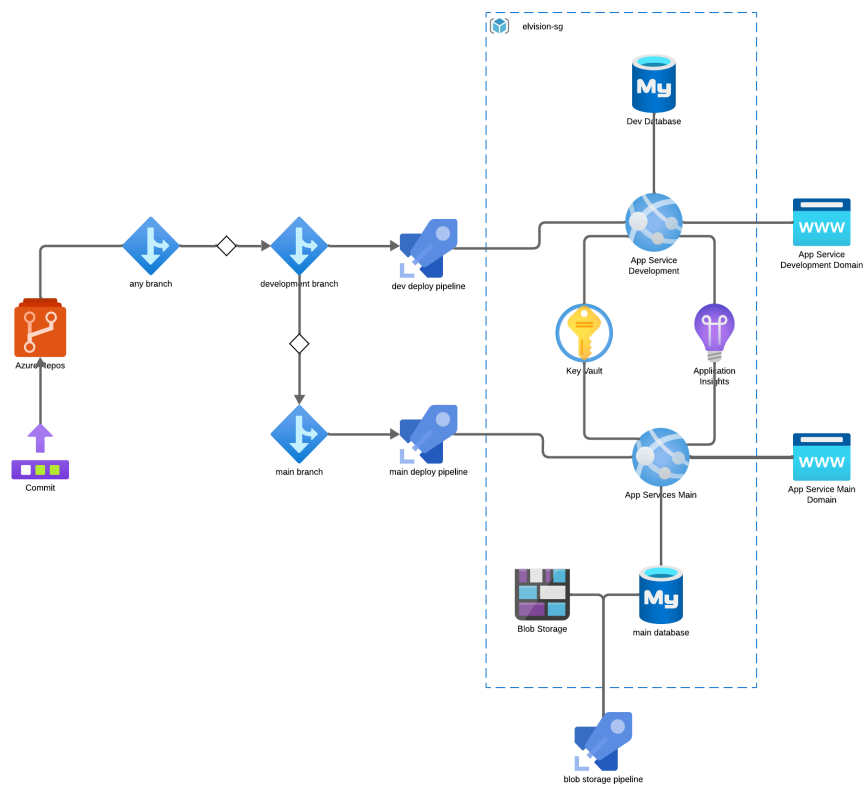
Passed

Test	Duration	Failing since	Failing build	Tags
<div>TestRun_ElVision_Build_&amp;_Analyze_20250102.8 (20/20)</div> <div>Tests GetDailyAverageTests GetDailyAverage ShouldRet</div>	0:00:24.596			

## Branch struktur



## Første uddrag af Azure Arkitektur



## Pipeline Run Tests Code

```

54  -- task: VSTest@3
55  -- inputs:
56  --   testSelector: 'testAssemblies'
57  --   testAssemblyVer2: |
58  --     **\Tests\bin\**\Tests.dll
59  --   searchFolder: '$(System.DefaultWorkingDirectory)'
60  --   codeCoverageEnabled: true
61  --   diagnosticsEnabled: true

```

## Kode Analyse Pipeline

```

14  -- task: DotNetCoreCLI@2
15  -- inputs:
16  --   command: 'build'
17  --   projects: '**/*.csproj'
18  --   arguments: '/p:RunAnalyzersDuringBuild=true /p:ReportAnalyzer=true /p:errorlog=$(Build.ArtifactStagingDirectory)\error.log'
19  -- displayName: 'Build solution with code analyzers'
20
21  Settings
22  -- task: WhiteSource@21
23  -- inputs:
24  --   cwd: '$(System.DefaultWorkingDirectory)'
25  --   projectName: 'ElVision'
26  -- displayName: Run Mend Bolt Scan
27
28  Settings
29  -- task: MicrosoftSecurityDevOps@1
30  -- displayName: 'Run Microsoft Security AntiMalware and BinSkim'
31  -- inputs:
32  --   tools: 'AntiMalware, BinSkim'
33  --   break: false
34  --   publish: true
35
36  Settings
37  -- task: Gitleaks@2
38  -- displayName: 'Run Gitleaks'
39  -- inputs:
40  --   scanlocation: '$(Build.SourcesDirectory)/ElVision'
41  --   configtype: 'predefined'
42  --   predefinedconfigfile: 'GitleaksUdmCombo.toml'
43  --   scanmode: 'nogit'
44  --   reportformat: 'sarif'
45  --   uploadresults: false
46  --   reportname: 'gitleaks-report'
47  --   reportfolder: '$(Build.ArtifactStagingDirectory)/CodeAnalysisLogs'
48  --   taskfail: false
49  --   redact: false

```



elovorblik-ai

Application Insights

Performance

Refresh

Code Optimizations

Profiler

View in Logs

Analyze with Workbooks

Copy link

Feedback

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Investigate

Application map

Smart detection

Live metrics

Transaction search

Availability

Failures

Performance

Monitoring

Alerts

Metrics

Diagnostic settings

Logs

Workbooks

Usage

Users

Sessions

Server

Browser

Local Time: Last 7 days

Roles = Cloud

Operations

Dependencies

Roles

Dependency response time

5.0 ms

50<sup>th</sup>

95<sup>th</sup>

99<sup>th</sup>

Dependency count

10

Thru 12

Fri 13

Sat 14

Sun 15

Mon 16

Tue 17

Wed 18

Thru 12

Fri 13

Sat 14

Sun 15

Mon 16

Tue 17

Wed 18

11:00 AM

11:00 PM

11:00 AM

11:00 PM

Select operation

Search to filter items...

DEPENDENCY NAME

DURATION ...

COUNT

PIN

Overall

98.1 ms

2.91k

HTTP: POST api.elovorblik.dk/customerapi/api/meterdata/gettimeseries/2024-03-01/2024-03-10...

639 sec

1

HTTP: GET api.energidataservice.dk/dataset/DeclarationTransmissionGridmix

1.94 sec

22

HTTP: POST api.elovorblik.dk/customerapi/api/meterdata/gettimeseries/2024-08-01/2024-12-18...

1.58 sec

1

HTTP: POST api.elovorblik.dk/customerapi/api/meterdata/gettimeseries/2024-12-15/2024-12-16...

1.12 sec

1

HTTP: POST api.elovorblik.dk/customerapi/api/meterdata/gettimeseries/2024-12-04/2024-12-11...

830 ms

4

HTTP: POST api.elovorblik.dk/customerapi/api/meterdata/gettimeseries/2023-12-12/2023-12-13...

823 ms

1

HTTP: POST api.elovorblik.dk/customerapi/api/meterdata/gettimeseries/2024-01-15/2024-12-18...

765 ms

1

HTTP: GET api.energidataservice.dk/dataset/DeclarationTransmissionGridmix

Distribution of durations: zoom into a range

Scale

5

4

3

2

1

Dependency count

1

Duration

1.0ms

14ms

78ms

260ms

570ms

1.1sec

2.0sec

1.0ms

14ms

78ms

260ms

570ms

1.1sec

2.0sec

Insights

No insights were found

Drill into...

22 Samples