Prosjektdokumentasjon Software Engineering og Testing

Gruppe 4:
Stian Lindseth, Erling N. Arnesen, Stefan Stefanovic, Jørgen A. Fjellstad

21. november 2024

Innhold

1 Innledning og problemstilling

Smarthjem-teknologi har i dag potensialet til å forenkle hverdagen, med muligheter for å kontrollere blant annet lys, lyd, varme og sikkerhet i hjemmet. Likevel ser vi at mange av dagens løsninger er rettet mot teknologientusiaster og krever bruk av mobilapper eller teknisk kompetanse, noe som gjør dem utilgjengelige for mange. Oppdragsgiver ønsker å endre dette ved å gjøre smarthjem-teknologi mer tilgjengelig og enklere å ta i bruk for alle, uavhengig av teknisk bakgrunn.

Domenet for dette prosjektet er vanlige hjem, der målet er å utvikle en løsning som kan brukes av alle, inkludert personer som ikke er komfortable med mobilapper eller teknologi generelt. Brukere av FullKontroll er personer som ønsker enkel og intuitiv styring av smarthjem-enheter, uten å måtte åpne en app eller gå gjennom kompliserte oppsett.

Løsningsforslag: FullKontroll
FullKontroll er vår foreslåtte løsning for dere, utviklet for å gi sluttbrukerne enkel og intuitiv styring av smarthjem-funksjoner uten behov for en app eller teknisk opplæring. Dette produktet er en kompakt fjernkontroll, og denne skal la brukeren styre smarthjem-enheter i hjemmet på en enkel og umiddelbar måte, noe som støtter deres visjon om å gjøre smarthjem-teknologi tilgjengelig for alle.

FullKontroll tilbyr følgende nøkkelfunksjoner som møter deres målsetninger:

- **Direkte kontroll uten app:** FullKontroll vil være en fysisk fjernkontroll som kobles direkte til smarthjem-enheter i hjemmet. Brukeren vil enkelt kunne styre funksjoner ved å trykke på knapper, uten å måtte navigere i en app eller digitale menyer.
- **Brukervennlig design:** Fjernkontrollen vil være utstyrt med justeringshjul, tydelige knapper og symboler som er tydelige og enkle å forstå. Dette vil sikre at brukerne raskt kan lære seg funksjonene uten opplæring.
- **Enkel tilkobling og kompatibilitet:** FullKontroll skal fungere med en mange smarthjem-enheter og kobles raskt til hjemmets nettverk uten kompliserte prosesser.
- **Sikkerhet og personvern:** FullKontroll vil bruke internett for å kommunisere med smarte enheter, men all dataoverføring vil være kryptert. Krypteringen beskytter brukernes data og sørger for at ingen uvedkommende kan få tilgang til informasjonen som sendes mellom fjernkontrollen og enhetene. Dette sikrer personvern og forhindrer uvedkommende fra å få tilgang til systemet.
- **Lang batterilevetid:** Takket være energieffektiv teknologi, vil batteriene vare lenge. Dette er ideelt for dem som ønsker en løsning som ikke krever mye vedlikehold. **Fleksibilitet og fremtidssikring:** Produktet kan utvides med nye funksjoner og støtte for flere enheter. Dette gjør det til en løsning som vokser med brukerens behov.

FullKontroll representerer en praktisk og brukervennlig løsning som gjør smarthjem-teknologi tilgjengelig for et større publikum. Fjernkontrollen skal være et enkelt verktøy som passer for alle, fra små familier til eldre brukere. Vi tror at dette produktet kan bidra

til å bringe smarthjem-teknologi inn i flere hjem, samtidig som det gir oppdragsgiver en konkurransefordel i markedet.

2 Kravspesifikasjon

2.1 Introduksjon til kravspesifikasjonen

Denne kravspesifikasjonen for FullKontroll beskriver hovedfunksjonene til systemet, brukerne og deres behov, og hvordan hver funksjon er prioritert og estimert.

2.2 Personas og brukerhistorier

For å sørge for at FullKontroll blir et nyttig produkt for sluttbrukerne, har vi utviklet personas som representerer ulike typer brukere. Disse viser hvem en bruker kan være, og hvilke utfordringer de kan stå overfor. Hver persona er knyttet til en brukerhistorie, som illustrerer et konkret behov.

Persona 1: Gudrun, 62 år - Teknologisk uerfaren, ønsker en enklere hverdag
Gudrun bor alene, men har sett på smarte hjem-løsninger. Hun synes løsningene virker kompliserte å forstå seg på. Hun vil bare ha en enkel måte å styre lyset og varmen i leiligheten uten å måtte reise seg. FullKontroll gir henne tilgang til smarte funksjoner i hjemmet sitt, noe som forenkler hverdagen hennes og gir henne mer tid til å slappe av.

- Brukerhistorie 1: "Som en teknologisk uerfaren bruker vil jeg kunne slå av alle lysene i leiligheten min med ett tastetrykk når jeg legger meg, så jeg slipper å gå innom hvert eneste rom."
- Brukerhistorie 2: "Som en bruker med behov for enkel tilgang vil jeg kunne justere varmen i ulike rom med én enhet, så jeg slipper å bøye meg ned til hver ovn."

Persona 2: Jens og Kristin, 34 og 33 år - Travle småbarnsforeldre
Jens og Kristin har to små barn og en hektisk hverdag. De ønsker en enkel måte å kontrollere lys, varme og sikkerhet på, uten å måtte gå innom flere forskjellige applikasjoner. FullKontroll gir dem muligheten til å slå av lys, redusere varme eller aktivere alarmen før de går for dagen, uten å måtte bruke flere apper.

- Brukerhistorie 3: "Som småbarnsforelder vil jeg kunne slå av alle lysene i huset samtidig når alle drar hjemmefra om morgenen, slik at vi får en så effektiv avgang som mulig."
- Brukerhistorie 4: "Som en person med en travel hverdag vil jeg kunne aktivere alarmen før vi forlater huset, uten å måtte ta opp telefonen og gjøre det via en app."

Persona 3: Svein, 44 år - Ønsker en effektiv og komfortabel hverdag
Svein er en businessmann med et travelt liv. Han liker smarthjem-teknologien, men har ikke så høy interesse for teknisk oppsett. Han ønsker et smart hjem som ser stilig ut, og

er funksjonelt. Han vil gjerne koble til så mange enheter som mulig, for å kunne justere på alt fra ett sted. FullKontroll gir han mulighet til å tilpasse hjemmet sitt raskt, med alle teknologier samlet i én fjernkontroll.

- Brukerhistorie 5: "Som musikkelsker ønsker jeg å kunne lagre innstillingene for "Musikk-kveld" på fjernkontrollen, slik at jeg kan gjenopprette dem neste gang jeg ønsker den samme følelsen i hjemmet."
- Brukerhistorie 6: "Som en person som verdsetter enkelhet vil jeg kunne styre både lyd og lys, døralarmer og hvitevarer, alt fra ett sted. Jeg har ikke tid til å sette meg inn i kontooppretelse på 10 forskjellige apper."

2.3 Estimeringsmetode

Hvert krav er identifiserbart og vurdert med en estimering av utviklingsomfang og forretningsnytte, basert på T-shirt Sizing-metoden. T-shirt Sizing ble valgt fordi den gir en enkel og fleksibel oversikt over omfanget på kravene. Dette gjør det mulig å gi kunden en tidlig indikasjon på ressursbruk uten å gå i detalj om nøyaktige tidsestimater, noe som er hensiktsmessig for dette prosjektets tidlige fase. Estimeringsnivåene er definert slik:

- Small (S): 1-3 timer
- Medium (M): 4-6 timer
- Large (L): 1 dag
- Extra Large (XL): Mer enn 1 dag

2.4 Oversikt over krav

Nedenfor følger en oversikt over FullKontrolls funksjonelle og ikke-funksjonelle krav, inkludert estimater for utviklingsomfang og forretningsnytte. Disse kravene viser tydelig hvordan systemet vil møte behovene til målbrukerne som beskrevet i personas og brukerhistorier.

2.4.1 Funksjonelle krav

- **KR-01 Direkte kontroll over enheter**
  Beskrivelse: FullKontroll skal gi brukeren muligheten til å styre smarthjem-enheter som lys, lyd, alarmer og termostat direkte med knapper på fjernkontrollen.
  Estimat – Utviklingsomfang: Extra Large (XL)
  Estimat – Forretningsnytte: Høy
  Dekkes av prototypen: Ja

- **KR-02 Enkel tilkobling og oppsett**
  Beskrivelse: FullKontroll skal støtte en enkel tilkoblingsprosess der nye enheter kan legges til fjernkontrollen med noen få tastetrykk, uten behov for teknisk veiledning.
  Estimat – Utviklingsomfang: Large (L)
  Estimat – Forretningsnytte: Høy
  Dekkes av prototypen: Ja

- **KR-03 Tilpasning av preferanser**
  Beskrivelse: Brukeren skal kunne lagre preferanser for f.eks. lys og varmeinnstillinger, slik at FullKontroll kan sette enhetene i ønsket tilstand med ett tastetrykk (f.eks.,"nattmodus").
  Estimat – Utviklingsomfang: Large (L)
  Estimat – Forretningsnytte: Medium
  Dekkes av prototypen: Nei

2.4.2 Ikke-funksjonelle krav

- **KR-04 Brukervennlig design**
  Beskrivelse: Fjernkontrollen skal ha et intuitivt design med lettforståelige symboler og knapper, inkludert et justeringshjul for å regulere lysstyrke og varme.
  Estimat – Utviklingsomfang: Medium (M)
  Estimat – Forretningsnytte: Høy
  Dekkes av prototypen: Nei

- **KR-05 Sikkerhet og personvern**
  Beskrivelse: All kommunikasjon mellom fjernkontrollen og smarthjem-enhetene skal være kryptert for å beskytte brukernes data og personvern.
  Estimat – Utviklingsomfang: Large (L)
  Estimat – Forretningsnytte: Høy
  Dekkes av prototypen: Nei

For å gi en grunnleggende demonstrasjon av FullKontrolls funksjonalitet vil prototypen inkludere følgende krav:

- KR-01 Direkte kontroll over enheter
- KR-02 Enkel tilkobling og oppsett

Disse funksjonene gir et solid grunnlag for å teste FullKontrolls kjernefunksjonalitet og gir et godt innblikk i brukeropplevelsen og produktets enkelhet i bruk.

Krav-ID	Beskrivelse	Omfang	Forretningsnytte	Dekkes av prototype
KR-01	Direkte kontroll over enheter	XL	Høy	Ja
KR-02	Enkel tilkobling og oppsett	L	Høy	Ja
KR-03	Tilpasning av preferanser	L	Medium	Nei
KR-04	Brukervennlig design	M	Høy	Nei
KR-05	Sikkerhet og personvern	L	Høy	Nei

3 Systemarkitektur

3.1 Valg av programmeringsspråk

For FullKontroll-prosjektet vurderes flere programmeringsspråk og rammeverk for å identifisere en teknologi som både kan oppfylle prosjektkravene og utvide teknisk kompetanse. Kotlin ble vurdert på grunn av sin moderne syntaks og effektivitet, men ettersom Kotlin primært brukes til mobilutvikling, anses det som lite egnet for dette webbaserte prosjektet. Java i kombinasjon med Spring Boot, et populært rammeverk for webapplikasjoner, ble også vurdert. Men etter å ha sett på hvor komplekst det kan være å jobbe med Spring Boot, fant vi at det ikke passet for oss med tanke på prosjektets tidsramme.

Node.js, som gjør det mulig å kjøre JavaScript på serveren og utvikle raske applikasjoner, ble også vurdert. Men på grunn av JavaScript sin begrensede støtte for full objektorientering, vurderes C# i kombinasjon med ASP.NET som et mer passende valg. ASP.NET tilbyr en robust plattform for webutvikling og gir muligheten til å utvikle en ryddig og strukturert prototype, samtidig som prosjektgruppen får verdifull erfaring med teknologier som er relevante i industrien.

3.2 Valg av rammeverk

Et rammeverk er et sett med verktøy, retningslinjer og kodebiblioteker som hjelper utviklere med å bygge programvare. ASP.NET er valgt som rammeverk fordi det gir oss de verktøyene vi trenger for å lage en skikkelig webapplikasjon. Det tilbyr støtte for flere designmønstre, inkludert MVC (Model-View-Controller) og Web API, og har gode funksjoner for å håndtere kommunikasjon mellom ulike deler av systemet. Med ASP.NET kan vi dele opp applikasjonen i tydelige moduler som gjør den lettere å vedlikeholde og utvikle. Rammeverket har også funksjoner for håndtering av nettverkskommunikasjon, dataflyt og autentisering.

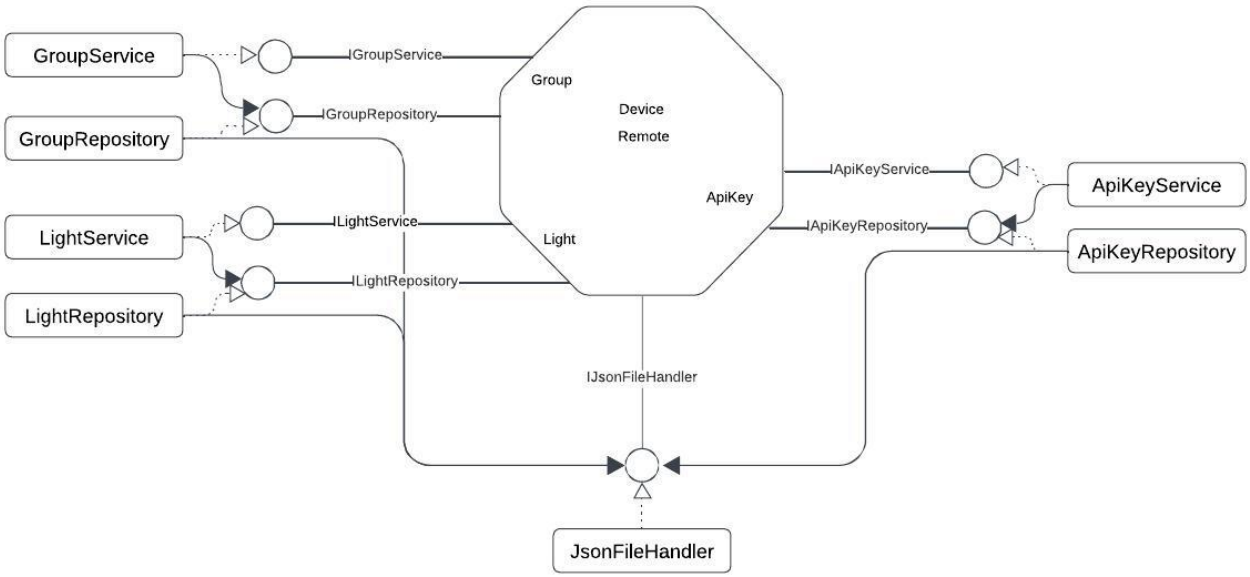
3.3 Arkitekturvalg

Arkitekturen som vil bli benyttet i FullKontroll, skal sikre at systemet blir robust, skalerbart og lett å vedlikeholde. Arkitekturen organiserer systemet og fastsetter hvordan ulike deler av systemet samhandler og hvilke roller hver komponent har.

Arkitekturen i FullKontroll er laget for å være fleksibel, slik at vi enkelt kan gjøre endringer i fremtiden. Den kombinerer heksagonal arkitektur i backend med MVC i frontend. Heksagonal arkitektur, også kjent som “Ports and Adapters“, hjelper oss med å skille forretningslogikken fra teknologi. Det betyr at vi kan bytte ut en del av systemet uten å endre kjernen. I frontend bruker vi MVC, som gir oss en ryddig måte å strukturere grensesnittet på.

3.4 Komponentdiagram

Dette diagrammet viser hvordan systemet er organisert med hovedkomponenter som GroupService, LightService, og ApiKeyService. Det illustrerer relasjonene mellom service-lag, repository-lag, og hvordan data lagres og håndteres gjennom JsonFileHandler. Diagrammet gir et visuelt overblikk over systemets tekniske struktur og samspill mellom komponentene.



3.5 Backend

Backenden, som håndterer forretningslogikk og dataflyt, vil bli utviklet med ASP.NET Web API. Dette API-et vil fungere som en bro mellom klientapplikasjonen (frontend) og data eller tjenester som backenden håndterer. Backenden vil bruke en heksagonal arkitektur for å gjøre systemet fleksibelt og enkelt å vedlikeholde. Denne arkitekturen deler systemet inn i separate deler som kan kommunisere med hverandre gjennom veldefinerte grensesnitt. Kjernen vil inneholde forretningslogikken, mens adaptere rundt kjernen vil håndtere kommunikasjonen med eksterne systemer som databaser eller API-er.

Fordelen med heksagonal arkitektur er at det er enkelt å bytte ut komponenter uten å påvirke kjernen. Hvis vi for eksempel vil legge til et nytt API senere, kan vi gjøre det uten å måtte endre på hele systemet. Dette gjør systemet lettere å vedlikeholde og gir fleksibilitet for fremtidige utvidelser.

Backendens struktur og relasjoner mellom komponenter som Controllers, Services, Repositories, og modeller er visualisert i Vedlegg 9: Klassediagram_Backend. Diagrammet gir en detaljert oversikt over hvordan data og funksjonalitet håndteres i backend.

3.6 Endepunkter og HTTP-forespørsler

Backend vil definere spesifikke URL-er, kalt endepunkter, som vil håndtere ulike handlinger. Endepunkter vil fungere som mål for HTTP-forespørsler, som brukes til å sende og motta data fra serveren. Prosjektet vil benytte ASP.NET Web API til å utvikle disse endepunktene, som skal håndtere handlinger som å hente, legge til eller oppdatere informasjon om smarthjem-enheter. Når frontend sender en forespørsel, for eksempel en GET- eller POST-forespørsel, vil den bli sendt til et bestemt endepunkt på backend, som bearbeider forespørselen og returnerer et svar.

3.7 Frontend

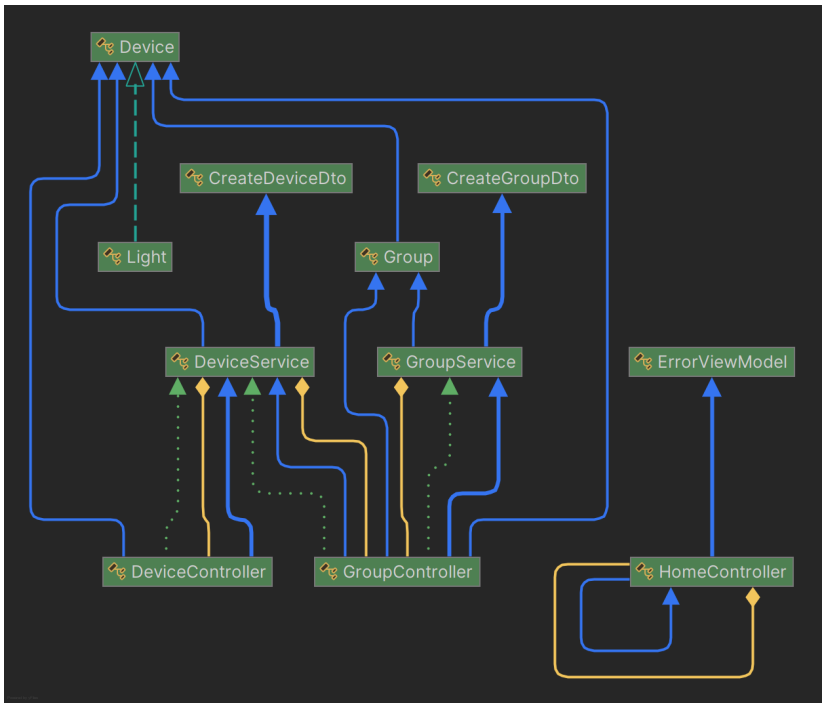
Frontend-delen, som brukeren ser og interagerer med, vil være basert på ASP.NET MVC-arkitektur. MVC er et mønster som deler applikasjonen inn i tre hoveddeler:

1. **Model:** Representerer data og logikken som trengs for å vise informasjon til brukeren eller behandle inndata fra brukeren. Den kan inneholde valideringslogikk, forretningslogikk relatert til visningen, eller data som hentes fra backend via tjenester.

2. **View:** Viser data til brukeren. Dette grensesnittet bygges med HTML og C# (i form av Razor-sider) og lar brukeren interagere med applikasjonen.

3. **Controller:** Håndterer brukerens input og oppdaterer visningen. Når brukeren gjør en handling, som å trykke på en knapp, fanger Controlleren opp handlingen og viser nødvendig informasjon i View.

Ved å bruke MVC kan frontend-koden struktureres på en måte som gjør systemet enklere å vedlikeholde og utvide. Under følger et klassediagram for frontend, der relasjonene mellom klasser som DeviceController, DeviceService, og Device blir visualisert.



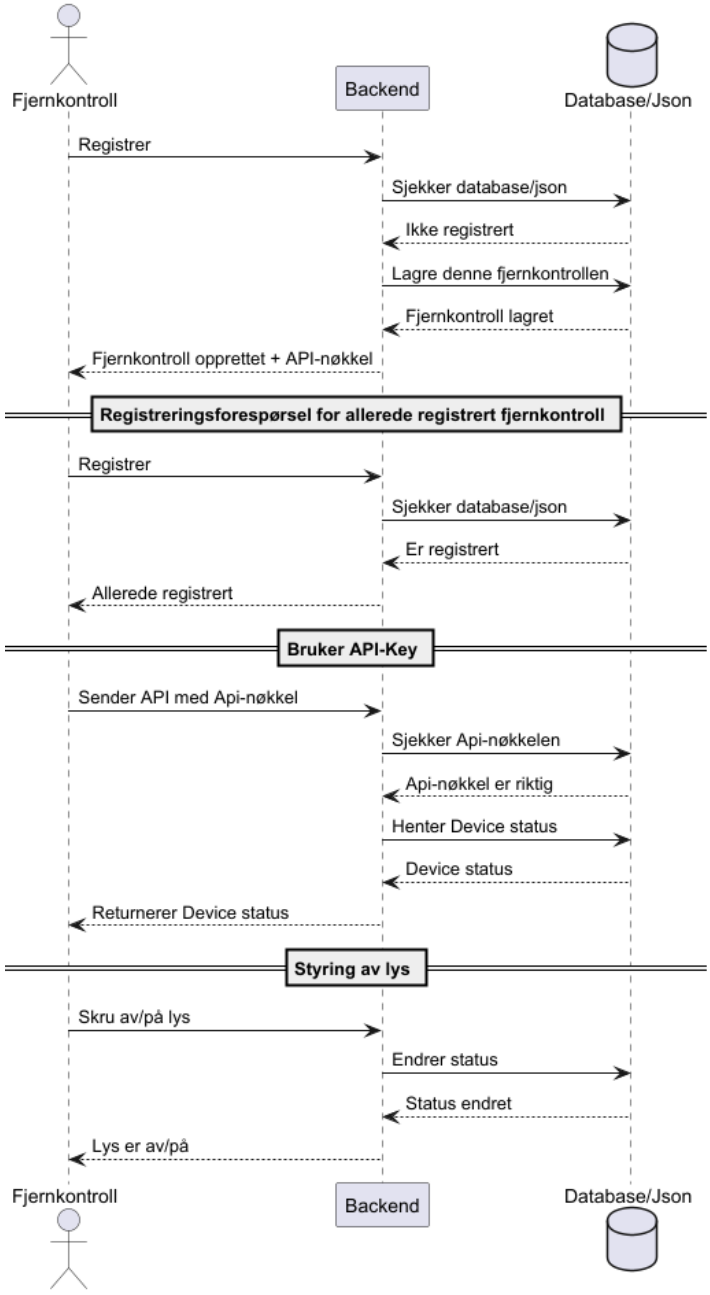
3.8 Service-lag i frontend

For å gjøre koden ryddig og modulær vil frontend inkludere et Service-lag, som består av klasser som håndterer HTTP-forespørsler til backend. For eksempel vil en 'DeviceService'-klasse utføre oppgaver som å hente og oppdatere informasjon om enheter. Dette samler API-kommunikasjonen i ett lag, noe som gjør koden enklere å vedlikeholde og teste. Ved å bruke DeviceService i Controlleren vil HTTP-forespørsler kunne utføres på en organisert måte, der logikken blir oversiktlig og lett å videreutvikle.

Denne strukturen gir applikasjonen en fleksibel og modulær oppbygging som er enkel å vedlikeholde. Arkitekturvalget og bruken av ASP.NET vil gjøre det mulig å bygge et skalerbart system som kan håndtere en rekke forskjellige smarthjem-enheter og tilpasse seg fremtidige utvidelser. Kombinasjonen av heksagonal arkitektur i backend og MVC-arkitektur i frontend gir klare skiller mellom ulike deler av applikasjonen, samtidig som den opprettholder god kommunikasjon mellom komponentene.

3.9 Sekvensdiagram

Sekvensdiagrammet illustrerer arbeidsflyten fra brukerens input i fjernkontrollen til back-end og videre til smarthjem-enhetene. Det viser hvordan systemet behandler forespørsler og returnerer enheter til ønsket tilstand, som når lysene slås av.



4 Teknisk dokumentasjon

4.1 Oversikt over systemet

FullKontroll-systemet vil være en smarthjem-løsning som gir brukeren enkel tilgang til å styre smarthjem-enheter. Når brukeren sender en kommando via fjernkontrollen, vil systemet sende forespørselen fra brukergrensesnittet til backenden, som behandler den og videresender instruksjonene til de tilkoblede IoT-enhetene. Systemet vil benytte ASP.NET-rammeverket, som gir funksjonalitet til å håndtere kommandoer effektivt og sikkert. Systemet vil gi brukeren mulighet til å styre for eksempel lys, varme og lydanlegg fra ett sted, uten behov for å bruke flere apper eller teknisk oppsett.

4.2 Valg av verktøy og redskaper

For å bygge FullKontroll-systemet vil vi bruke flere verktøy som støtter utviklingsprosessen, gir god kontroll på versjoner av prosjektet, og gjør det enklere å samarbeide om utviklingen.

- **GitHub og GitHub Desktop:**
  GitHub vil brukes til å lagre og dele kode, samt til å holde oversikt over oppgaver i en backlog. GitHub gir god versjonskontroll, noe som gjør at teamet enkelt kan oppdatere og kommentere hverandres kode. GitHub Desktop, en visuell klient for GitHub, vil brukes for å håndtere prosjektet både lokalt og i skyen. Vi velger disse verktøyene på grunn av deres brukervennlighet og utbredelse i utviklingsmiljøer.

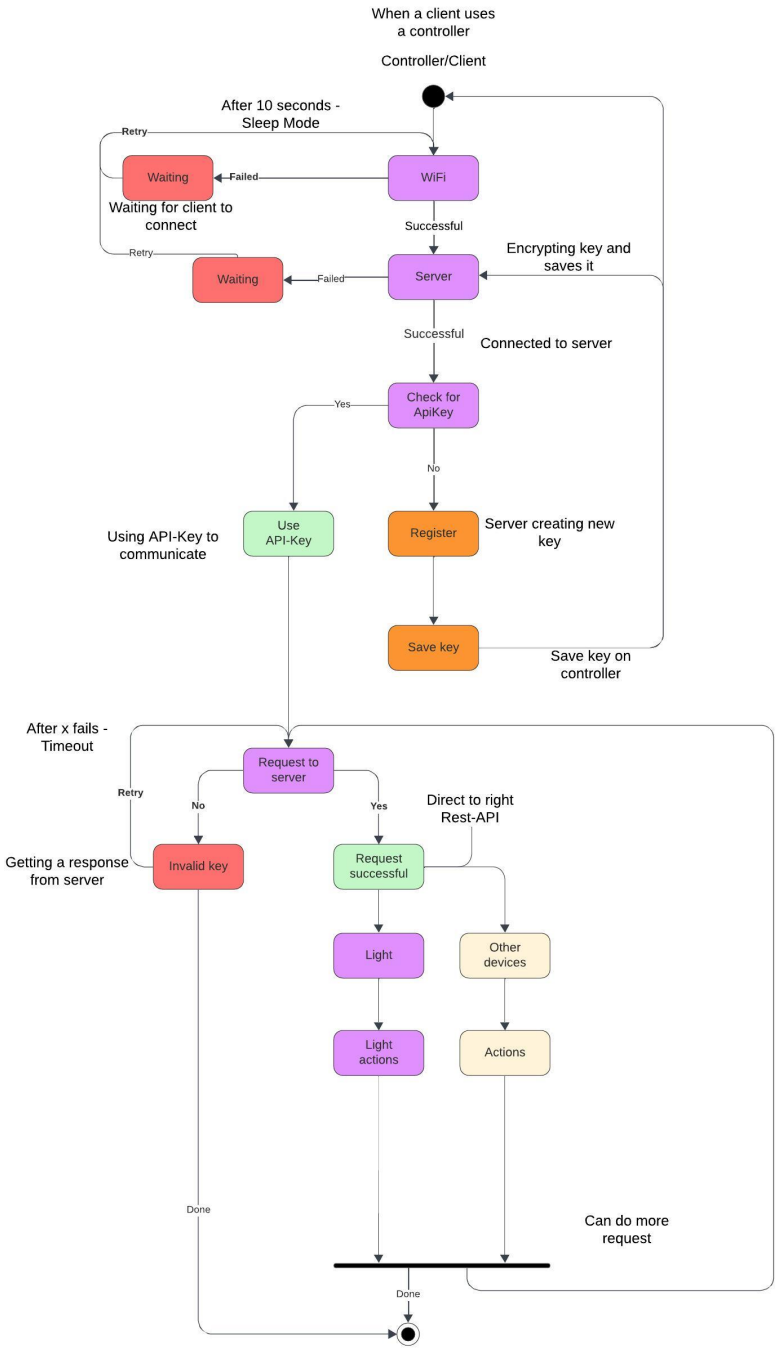
- **Visual Studio Code (VS Code):**
  VS Code vil være hovedverktøyet for å skrive og teste koden i prosjektet. VS Code støtter mange programmeringsspråk og Git-kommandoer som push, pull og commit. Koderedigeringsverktøyet er valgt på grunn av sin fleksibilitet og gode integrasjon med GitHub. Det gir en smidig arbeidsflyt og støtter rask testing og justering av koden

- **ESP32 mikrokontroller:**
  For å bygge prototypen av fjernkontrollen vil vi bruke ESP32, en mikrokontroller med støtte for både Wi-Fi og Bluetooth. Denne mikrokontrolleren er valgt på grunn av sitt lave energiforbruk og rimelige pris, noe som gjør den godt egnet for prototyper og testing av funksjonalitet.

- **Arduino IDE:**
  Arduino IDE vil brukes til å programmere ESP32-enheten. Dette utviklingsmiljøet gir tilgang til mange biblioteker som forenkler utviklingsprosessen. Arduino IDE lar teamet raskt justere og laste opp kode, noe som er nyttig for å teste endringer i prototypen.

4.3 Aktivitetsdiagram

Aktivitetsdiagrammet beskriver hvordan systemet opererer gjennom forskjellige scenarier, som tilkobling til server, validering av API-nøkler, og utførelse av kommandoer. Det gir en detaljert forklaring på hvordan systemet reagerer på ulike brukerhandlinger og beslutninger.


When a client uses a controller
Controller/Client
After 10 seconds - Sleep Mode
Retry
Waiting
Failed
WiFi
Waiting for client to connect
Successful
Retry
Waiting
Failed
Server
Encrypting key and saves it
Successful
Connected to server
Check for ApiKey
Yes
No
Use API-Key
Register
Server creating new key
Using API-Key to communicate
Save key
Save key on controller
After x fails - Timeout
Request to server
Retry
No
Yes
Direct to right Rest-API
Getting a response from server
Invalid key
Request successful
Light
Other devices
Light actions
Actions
Done
Can do more request
Done

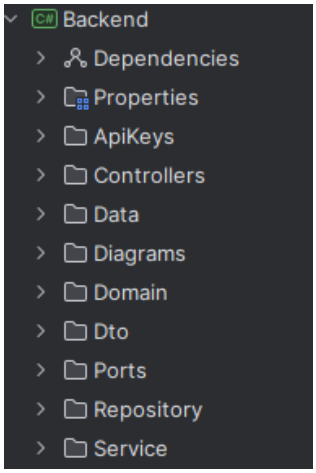
4.4 Filstruktur og moduloppbygning

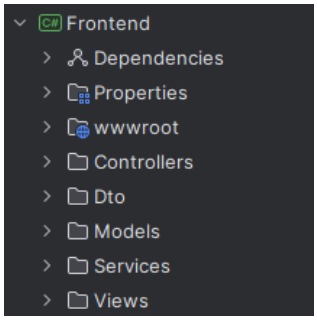
For å sikre en organisert og lett vedlikeholdbar kodebase vil prosjektet struktureres i flere mapper og moduler. Dette gjør det enklere for utviklere å navigere i koden og forstå systemets oppbygning. Her er en oversikt over hovedmappene og hva de vil inneholde:

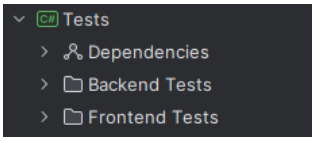
Backend: Backend-delen håndterer forretningslogikken og databehandling i prosjektet. Den bygges med heksagonal arkitektur (Ports and Adapters). Mappestrukturen inkluderer:

- **Dependencies:** Eksterne biblioteker og rammeverk som backend er avhengig av.
- **ApiKeys:** Der API-nøklene blir lagret.
- **Controllers:** Endepunkter for HTTP-forespørsler fra frontend.
- **Data:** Der dataene om enheter og grupper blir lagret.
- **Diagrams:** Diagrammer for dokumentasjon av systemet.
- **Domain:** Domenemodeller som representerer forretningslogikken.
- **Dto:** Data Transfer Objects for å utveksle data.
- **Ports:** Grensesnitt for kommunikasjon med eksterne systemer.
- **Repository:** Lag for datahåndtering og lagring.
- **Service:** Forretningslogikk som kommuniserer med repository-laget.

Figurene under viser hvordan mappene struktureres i prosjektet:







Frontend: Frontend-delen inneholder koden som håndterer brukergrensesnittet. Strukturen følger Model-View-Controller (MVC)-mønsteret og består av:

- **Dependencies:** Eksterne biblioteker og pakker som frontend er avhengig av.

- **wwwroot:** Vil lagre statiske filer som CSS, JavaScript, og bilder.
- **Controllers:** Logikk for håndtering av brukerinteraksjon og dataflyt til views.
- **Dto:** Data Transfer Objects som brukes for å utveksle data mellom lag.
- **Models:** Representasjon av applikasjonsdata.
- **Services:** Klasser for håndtering av HTTP-forespørsler til backend.
- **Views:** Brukergrensesnitt for eventuelle visualiseringer i web-applikasjonen.

Tests: Vil inneholde tester som sikrer funksjonaliteten i ulike deler av systemet. Testene dekker både forretningslogikk og spesifikke moduler, som API-er og Core-funksjoner. Den er delt inn i:

- **Dependencies:** Eksterne biblioteker og rammeverk brukt i testing.
- **Backend Tests:** Tester for backend-funksjonalitet.
- **Frontend Tests:** Tester for frontend-funksjonalitet.

Denne strukturen gir et klart skille mellom de ulike ansvarsområdene, og gjør det enkelt å finne og oppdatere spesifikke funksjoner uten å påvirke resten av systemet.

4.5 Avhengigheter

4.5.1 Avhengigheter brukt i backend

FullKontroll-systemet vil benytte flere avhengigheter (biblioteker og rammeverk) i backend for å muliggjøre effektiv utvikling og funksjonalitet. Disse avhengighetene blir integrert for å støtte ASP.NET Core MVC-arkitekturen og håndtere kommunikasjon, datahåndtering, og logikk.

- **ASP.NET Core MVC**: Rammeverket som vil gi struktur til backenden. Det støtter oppbyggingen av API-er, slik at frontend og smarte enheter kan kommunisere effektivt med backend.
- **Entity Framework Core**: Vil brukes for å administrere data og interaksjon med databasen. Entity Framework forenkler lagring og henting av data som bruker-, enhets- og statusinformasjon.
- **System.Net.Http**: Vil håndtere HTTP-forespørsler som kommer fra GUI og andre systemkomponenter.
- **System.Text.Json**: Vil støtte serialisering og deserialisering av JSON-data, noe som er viktig for dataoverføring mellom frontend og backend.
- **System.Threading.Tasks**: Vil støtte asynkrone operasjoner, som muliggjør effektiv håndtering av API-forespørsler og behandling av data i bakgrunnen uten å blokkere hovedtråden.

4.5.2 Avhengigheter brukt i ESP32 mikrokontroller

For å sikre kommunikasjon og funksjonalitet på ESP32-enheten bruker vi flere spesialiserte biblioteker:

- **WiFi.h**: Gir ESP32-en muligheten til å koble seg til trådløst internett, slik at den kan kommunisere med andre enheter på nettverket.
- **HTTPClient.h**: Tillater ESP32-en å sende og motta meldinger over nettverket, noe som muliggjør kommunikasjon med serveren for å styre smarte enheter.
- **Wire.h**: Brukes for å koble ESP32-en til andre komponenter via I2C-protokollen, som OLED-skjermen for visning av tekst.
- **Adafruit_GFX.h og Adafruit_SSD1306.h**: Kombinerte biblioteker som gir ESP32-en muligheten til å vise tekst og enkel grafikk på OLED-skjermen, noe som gjør informasjonen tydelig for brukeren.
- **ArduinoJson.h**: Gjør det mulig for ESP32-en å tolke JSON-data, slik at den kan forstå meldinger fra serveren og hente ut viktig informasjon om enhetens status og grupper.

4.5.3 Avhengigheter brukt i grafisk brukergrensesnitt (GUI)

FullKontroll vil ha et grafisk brukergrensesnitt (GUI) som gir brukeren enkel tilgang til smarthjem-funksjonene. GUI-et vil bygges med 'System.Windows.Forms' i C#, som gjør det mulig å lage vinduer og knapper i Windows-miljøet.

For å lage brukergrensesnittet vil vi bruke flere verktøy, kalt avhengigheter. Avhengigheter inkluderer både biblioteker og rammeverk, som gir ferdige funksjoner og strukturer som gjør det enklere å programmere bestemte oppgaver. I FullKontroll-prosjektet vil vi bruke følgende avhengigheter i GUI-et:

- **System**: Dette biblioteket gir grunnleggende funksjoner som datatyper og funksjoner for å lagre informasjon og kommunisere med andre deler av systemet.
- **System.Threading**: Dette biblioteket gjør det mulig å bruke tråder. En tråd kan utføre flere oppgaver samtidig, noe som gjør GUI-et responsivt, selv når det utfører tidkrevende handlinger i bakgrunnen.
- **System.Windows.Forms**: Dette biblioteket gjør det mulig å bygge visuelle komponenter, som knapper og vinduer, som brukeren ser og interagerer med. Det brukes til å lage hoveddelen av brukergrensesnittet.
- **System.Net.Http**: Dette biblioteket brukes til å sende og motta informasjon via internett. Når brukeren gir en kommando, sender systemet en forespørsel til backenden for å utføre handlingen.
- **System.Threading.Tasks**: Dette biblioteket sikrer at brukergrensesnittet holder seg responsivt, selv når systemet utfører oppgaver som tar tid, som for eksempel å hente data fra smarthjem-enheter.

- **System.IO**: Dette biblioteket brukes til å håndtere filer og mapper. I systemet brukes det til å lese, skrive og lagre informasjon, slik at viktige data lagres mellom økter.
- **System.Text.Json** og **System.Text.Json.Serialization**: Disse bibliotekene brukes til å jobbe med JSON (JavaScript Object Notation), et dataformat som gjør det enkelt å sende og motta informasjon mellom systemets ulike deler. JSON brukes ofte til å strukturere og organisere data på en måte som både mennesker og maskiner kan lese.

4.6 GUI-funksjonalitet

Brukergrensesnittet i FullKontroll vil bygges med System.Windows.Forms, som gir et enkelt kontrollpanel for smarthjem-funksjoner. Brukeren vil ha tilgang til knapper og kontrollere for funksjoner som å slå av/på lys, justere varmeinnstillinger, og aktivere sikkerhetssystemer. Når brukeren trykker på en knapp, sender systemet en forespørsel til backend, som behandler kommandoen og sender instruksjonene til de smarte enhetene. Dette gjør at alle kommandoene kan utføres raskt og direkte fra ett sted, uten at brukeren må bruke flere applikasjoner eller avanserte tekniske innstillinger.

4.7 Praktisk bruk av API-er

Systemet kommuniserer med smarthjem-enheter via et sett med "adresser"(API-er) som gjør det mulig å sende instruksjoner til enhetene og motta oppdatert informasjon. Dette betyr at systemet raskt kan oppdatere enhetens status, slik at brukeren ser resultatene direkte i grensesnittet.

Eksempler på API-endepunkter:

- `/devices/turnOffLights`: Dette endepunktet brukes for å slå av alle lysene i systemet.
- `/devices/status`: Henter status for alle tilkoblede enheter.

Når brukeren sender en kommando via GUI-et, vil systemet sende en forespørsel til et API-endepunkt som tar seg av kommandoen. Et eksempel på en forespørsel for å slå av lysene kan se slik ut:

```
POST /devices/turnOffLights
Response: { "status": "success", "message": "Lights turned off" }
```

4.8 Begrensninger og antagelser

For at systemet skal fungere optimalt, har vi definert enkelte tekniske begrensninger og antagelser som gjelder for FullKontroll.

Tekniske begrensninger

- **Støttede protokoller:**
  Systemet forutsetter at smarte enheter støtter de vanlige kommunikasjonsmåtene (HTTP eller MQTT) og kan utføre grunnleggende kommandoer, slik at brukerne får en sømløs opplevelse ved å kunne styre alle enhetene sine fra ett sted.

- **Internett-tilgang for kommunikasjon:**
  FullKontroll krever internett- eller lokalt nettverkstilgang for å kunne kommunisere med smarthjem-enhetene. Hvis forbindelsen ikke er stabil, kan systemet oppleve forsinkelser eller midlertidig tap av funksjonalitet.

Funksjonelle begrensninger

- **Enkelhet i brukergrensesnittet:**
  Brukergrensesnittet er designet for enkelhet, og tilbyr kun grunnleggende funksjoner som å slå av/på lys, justere lysstyrke og varmeinnstillinger. Mer avanserte funksjoner for tilpasning er ikke inkludert for å sikre et intuitivt oppsett.

- **Begrenset sikkerhetsnivå:**
  Selv om systemet bruker grunnleggende kryptering for kommunikasjon, vil det ikke inkludere avanserte sikkerhetstiltak som tofaktorautentisering eller dyp kryptering på alle nivåer.

Antagelser om bruker og miljø

- **Brukeren har begrenset teknisk kunnskap:**
  FullKontroll er utviklet for brukere uten spesialisert teknisk kunnskap, og det er en antagelse at brukeren har behov for enkle og intuitive kontroller.

- **Kompatibilitet med smarthjem-enheter:**
  Systemet antar at alle tilkoblede smarthjem-enheter støtter de valgte kommunikasjonsprotokollene (HTTP eller MQTT) og er i stand til å utføre grunnleggende kommandoer.

- **Stabil strømtilgang:**
  Systemet forutsetter at alle tilkoblede enheter og fjernkontrollen har en stabil strømforsyning, slik at funksjonaliteten ikke påvirkes av strømmangel eller avbrudd.

5 Testing og kvalitetssikring

5.1 Introduksjon til testing

Testing er en kritisk del av systemutviklingen for å sikre at de utviklede funksjonene oppfyller kravene og fungerer som forventet. Testene fokuserer på å identifisere feil tidlig og på å sikre at systemet leverer pålitelig ytelse i alle relevante bruksscenarier. Gjennom en strukturert tilnærming til testing legges det vekt på både funksjonalitet og brukeropplevelse.

5.2 Planlagt teststrategi

Teststrategien er designet med følgende fokusområder:

- **Omfang:** Testene fokuserer på alle kritiske aspekter av systemet, med hovedvekt på funksjonalitet, stabilitet og korrekt datahåndtering.

- **Testnivåer:** Testene inkluderer en kombinasjon av enhetstester, integrasjonstester og funksjonelle tester for å sikre helhetlig dekning.

- **Verktøy:** Testene implementeres ved hjelp av C#-baserte testverktøy som XUnit og Moq, og automatiserte kjøringer skal brukes for å sikre effektivitet og rask tilbakemelding.

Testresultatene vil dokumenteres for å gi en oversikt over identifiserte feil, samt for å tydelig vise hvilke deler av systemet som krever videre forbedring.

5.3 Testing og fokusområder

Testene fokuserer på de mest kritiske funksjonene i systemet og organiseres i følgende kategorier:

Funksjonelle Tester: De funksjonelle testene fokuserer på å validere at systemet leverer riktig funksjonalitet. Spesifikke testområder inkluderer:

- **Kontroll av enheter:** Testene sikrer at systemet korrekt kan endre tilstanden til tilkoblede enheter, som å slå dem av eller på.
- **Håndtering av grupper:** Testene validerer at grupper kan opprettes, redigeres og slettes uten feil, samtidig som brukeren mottar tilbakemeldinger ved ugyldige handlinger.
- **Datahåndtering:** Testene skal bekrefte at informasjon om enheter og grupper lagres korrekt i JSON-filer og hentes uten datafeil eller tap.

Enhetstester: Enhetstestene fokuserer på små, isolerte deler av koden for å validere kritiske funksjoner, som logikken for gruppering av enheter og håndtering av brukerinput. Dette sikrer:

- Tidlig oppdagelse av feil i individuelle komponenter.
- Stabilitet og pålitelighet i grunnleggende systemfunksjoner.

Integrasjonstester: Integrasjonstestene fokuserer på å validere samspillet mellom ulike moduler, som samhandlingen mellom enhetskontroll, gruppehåndtering og datalagring, for å sikre at systemet fungerer sømløst som en helhet.

5.4 Kvalitetssikring

For å sikre høy kvalitet gjennom hele utviklingsprosessen implementeres følgende tiltak:

Kodelesing og teamgjennomgang: Alle nye funksjoner gjennomgås av teammedlemmer før de integreres i hovedkodebasen. Dette reduserer risikoen for skjulte feil og øker kvaliteten på koden.

Automatisert testing: Automatiserte testskript brukes for å forenkle testkjøringer og sikre rask tilbakemelding på systemets tilstand.

Brukervennlig feilhåndtering: Systemet skal gi klare og forståelige feilmeldinger ved ugyldige handlinger, for eksempel:

- Når brukeren forsøker å legge til en enhet som allerede finnes.
- Ved forsøk på å opprette en gruppe med et navn som allerede eksisterer.

5.5 Dokumentasjon av testresultater

Testresultatene loggføres og organiseres i en testrapport som gir oversikt over gjennomførte tester, identifiserte feil, og status for systemets stabilitet. Dette sikrer transparens og gir utviklere og interessenter tydelig informasjon om systemets tilstand.

Instruksjoner for å kjøre tester:

1. Åpne prosjektmappen: `IoT-Prosjekt`.
2. Kjør `Run tests.bat` for å gjennomføre alle testene.
3. Les testrapporten for detaljer om resultatene.

5.6 Begrensninger og antagelser

Testene fokuserer på funksjonalitet og stabilitet, og ytelsestesting har vært utenfor prosjektets omfang på dette stadiet. Dette skyldes tidsbegrensninger og behovet for rask levering av en fungerende prototype. Videre testing, som stresstester, sikkerhetstester og skalerbarhetstester, anbefales før systemet settes i produksjon for å sikre robusthet og pålitelighet i større skala.

6 Prototype

6.1 Formål med prototypen

Prototypen er utformet som en minimum viable product (MVP) for å gi verdi til kunden ved å vise hvordan FullKontroll kan fungere i praksis. Den viser hvordan løsningen kan håndtere smarthjem-funksjoner, og vi har fokusert på å bygge noe som gir verdi og gir et realistisk inntrykk av løsningen.

6.2 Instruksjoner for å kjøre og teste prototypen

For å kjøre prototypen, kreves .NET 6.0 og 8.0. Prototypen startes ved å kjøre en fil rett fra hovedmappen. Se Vedlegg for full brukermanual. Dette vedlegget inneholder all nødvendig informasjon for å komme i gang.

6.3 Funksjonalitet og brukergrensesnitt

Prototypen fokuserer på et funksjonelt brukergrensesnitt som demonstrerer kjernefunksjoner uten unødvendig visuell kompleksitet. Brukergrensesnittet består av knapper for enkel tilgang til funksjoner som per nå består av å slå av/på lys, uten behov for innlogging. Dette gir brukeren rask tilgang til de viktigste funksjonene og reduserer kompleksiteten.

6.4 Mulige begrensninger og forbedringsområder

Det kan være noen begrensninger i prototypen, inkludert:

- Enkelte funksjoner vil være grunnleggende og fungerer som demonstrasjoner av systemets potensial.
- Grensesnittet vil være fokusert på funksjonalitet, ikke visuell design, ettersom hovedmålet er å vise systemets kjernefunksjoner.

Disse potensielle begrensningene forklares for å gi kunden realistiske forventninger til hva prototypen kan vise, samt for å peke på forbedringspotensialet i en endelig implementasjon.

6.5 Kort oppsummering

Prototypen er planlagt å være kjørbar, enkel å teste, og vil demonstrere de viktigste funksjonene som foreslått i kravspesifikasjonen. Dokumentasjonen er holdt kort og funksjonsorientert for å sikre at sensor og oppdragsgiver enkelt kan forstå verdien og begrensningene i systemet.

7 Evalueringer og konklusjon

7.1 Planlagt fremgangsmåte

I utviklingen er det planlagt å bruke utprøvende utviklingsmodell, også kjent som eksperimentell utvikling eller prototyping. Denne modellen vil gi oss frihet til å teste ulike løsninger og justere dem underveis, uten å være låst til en fast plan. Modellen gir fleksibilitet, som passer dette prosjektet godt. Hvis kravene skulle endre seg underveis, kan vi tilpasse oss uten store omkostninger, noe som er verdifullt for prosjektets utvikling.

For å holde orden på koden og unngå konflikter bruker vi Git. Vi oppdaterer kodebasen ofte og lager separate branches for nye funksjoner. Dette lar oss teste og finjustere hver del for seg, uten å påvirke hovedversjonen som alltid skal være stabil.

Vi vil ha faste statusmøter der vi sjekker fremgangen, løser tekniske utfordringer, og justerer planen hvis det trengs. Disse møtene er viktige for å sikre at alle som arbeider med prosjektet holder seg oppdatert og kan komme med innspill. I tillegg vil vi bruke møtene til å bestemme hvilke oppgaver som er viktigst å fokusere på fremover.

Arbeidet blir delt opp i oppgaver til hver person, som dekker spesifikke deler av systemet, som dokumentasjon, brukergrensesnitt og tilkobling av smart-enheter. På denne måten kan vi arbeide fokusert med ett område av gangen, men likevel ha rom for justeringer underveis når vi lærer mer om hva som fungerer best.

Med denne måten å arbeide på, tror vi at vi kan lage en løsning som både er enkel å bruke og pålitelig, og som møter oppdragsgivers krav og forventninger.

7.2 Forventet måloppnåelse

FullKontroll skal gjøre det enkelt for brukerne å styre smarte enheter i hjemmet fra én fjernkontroll. Målet er å lage et system som dekker alle kravene som er beskrevet, samtidig som det er enkelt å bruke og gir en god opplevelse for alle typer brukere. Løsningen vil tilby:

- En praktisk fjernkontroll som gir tilgang til funksjoner som å slå av og på lys, justere varmen, styre lyd og aktivere sikkerhetssystemer, uten at brukeren trenger teknisk kunnskap.
- Trygg og stabil kommunikasjon med smarte enheter gjennom kryptering, slik at brukernes data og personvern blir beskyttet.
- Et system som er lett å utvide med nye funksjoner og enheter, enten det brukes i et lite hjem eller i et større bygg.
- Lavt strømforbruk og god batterilevetid, slik at fjernkontrollen kan brukes over lengre tid uten hyppig lading.
- Enkelt oppsett som gjør det raskt å koble fjernkontrollen til smarthjemmet uten behov for avanserte prosesser.

Vi tror FullKontroll vil dekke både brukernes behov og oppdragsgivers forventninger. Løsningen skal være fleksibel, kostnadseffektiv og enkel å ta i bruk. Målet er å skape et produkt som gjør hverdagen lettere og mer komfortabel for brukerne, samtidig som det viser hvordan teknologien kan brukes til å gjøre smarthjem enkelt for alle.

7.3 Forventede styrker og svakheter

FullKontroll-prosjektet har klare fordeler og utfordringer som vi forventer vil påvirke hvordan løsningen blir mottatt og brukt.

Styrker
På styrkesiden forventer vi at systemet vil være:

- **Enkelt å bruke:** Den fysiske fjernkontrollen og det intuitive brukergrensesnittet er designet for å minimere tekniske barrierer. Vi tror dette vil appellere til brukere som ikke ønsker å sette seg inn i komplekse apper eller systemer.
- **Sikkert:** Med kryptert kommunikasjon og ingen avhengighet av tredjeparts skybaserte tjenester, gir systemet brukerne trygghet om at deres data er godt beskyttet.
- **Fleksibelt:** Arkitekturen gjør det mulig å legge til flere funksjoner og enheter i fremtiden, noe som gir produktet et langt livsløp.
- **Energieffektivt:** ESP32-komponentens lave strømforbruk sikrer at fjernkontrollen kan brukes lenge mellom hver lading.

Svakheter
På svakhetssiden er det noen områder vi forventer kan by på utfordringer:

- **Begrenset funksjonalitet i starten:** Som et MVP vil ikke alle ønskede funksjoner være inkludert fra første dag. Dette kan være en skuffelse for noen brukere.
- **Kompatibilitet:** Selv om produktet støtter vanlige smarthjem-protokoller, kan enkelte brukere oppleve problemer med å koble til eldre eller mindre vanlige enheter.
- **Tilpasning til brukeres hjem:** Hvert hjem er unikt, og vi vet ikke sikkert hvordan produktet vil fungere i alle mulige konfigurasjoner.
- **Teknisk støtte:** Det kan bli utfordrende å sikre god teknisk støtte i oppstartsfasen, spesielt hvis mange brukere tar produktet i bruk samtidig.

Disse styrkene og svakhetene gir et realistisk bilde av hva vi tror produktet vil tilby ved lansering, og hvilke områder vi kan forbedre i fremtiden.

7.4 Mulige utfordringer og tiltak

FullKontroll er et ambisiøst prosjekt, men det kommer ikke uten utfordringer. Noen av disse kan være tekniske, mens andre er knyttet til brukeropplevelsen. Det viktigste er å være forberedt og løsningsorientert.

Oppkobling av smarte enheter er ett område vi vet kan bli krevende. Brukere som ikke er vant til teknologiske løsninger, kan oppleve oppsettet som komplisert. Kanskje spesielt hvis ting ikke fungerer som forventet ved første forsøk. For å unngå frustrasjon planlegger vi å legge inn visuelle instruksjoner som guider brukeren gjennom hele prosessen, trinn for trinn. Enkle feilmeldinger kan også hjelpe, slik at det blir tydelig hva som gikk galt og hva som må gjøres.

I tilfeller der nettverksforbindelsen mellom fjernkontrollen og smarthjem-enhetene blir midlertidig borte, kan systemet lagre kommandoer lokalt på fjernkontrollen. Dette betyr at kommandoer som "slå av lys" eller "juster temperaturen" blir husket og sendt til enhetene så snart forbindelsen er gjenopprettet. Dette sikrer at systemet oppleves som pålitelig, selv i situasjoner med varierende nettverksforhold.

Sikkerhet er også en av de største prioriteringene. Vi bruker kryptering for å beskytte kommunikasjonen, men risikoen for sikkerhetsbrudd kan aldri fjernes helt. Regelmessige oppdateringer av protokoller og overvåking av potensielle sårbarheter vil derfor være avgjørende. Brukernes data må alltid være trygge.

Når det gjelder brukeropplevelsen, vet vi at noen funksjoner kan virke avanserte. Gruppeadministrasjon og tilpasning av innstillinger er eksempler på dette. Her vil vi legge inn små forklaringer i grensesnittet. Det er en enkel løsning som kan gjøre en stor forskjell.

Til slutt må vi ta høyde for konkurransen i markedet. Smarthjem-produkter finnes i mange former, og det kan være vanskelig å skille seg ut. Derfor vil vi fokusere på enkelhet og brukervennlighet som våre viktigste styrker. Det skal ikke føles komplisert å bruke FullKontroll. Snarere tvert imot.

Med disse tiltakene håper vi å møte utfordringene på en måte som skaper verdi for både oppdragsgiver og brukerne.

7.5 Videreutviklingsmuligheter

FullKontroll har et sterkt utgangspunkt, men det er mange muligheter for videreutvikling:

- **Flere enheter:** Systemet kan støtte flere typer smarte enheter og funksjoner. Her kan det settes opp porter (interface) til forskjellige typer smarthjem-enheter. Det vil si at når fjernkontrollen kobles til en enhet så skal fjernkontrollen kunne hente riktig adapter og få en bruksanvisning til hvordan den kan styre den type enhet. Dette kan vi gjøre siden vi har valgt å bruke heksagonal arkitektur som bruker Adapter og Port.
- **Sentral HUB:** En sentral løsning kan forenkle kommunikasjonen mellom enhetene. Dette vil i praksis bli en HUB til fjernkontrollen som kan erstatte bridges (broer) som for eksempel Philips Hue Bridge. Det vil si at smarthjem-enheter kan ha et samlepunkt, og fjernkontrollen kan da oppdatere alle enhetene i HUBen.
- **Database:** Lagring av brukermønstre kan gi forslag til energieffektivitet og automatisering. Det kan settes opp en database som vil gjøre det enklere å samle inn data og lagre dette. Det vil også gjøre det mer sikkert for dataene som blir lagret i tilfelle noen klarer å bryte seg inn i backenden. Det skal fortsatt krypteres data før lagring til sensitiv informasjon.
- **Bedre kommunikasjon:** WebSocket-teknologi kan gjøre systemet raskere og mer responsivt. Vi har nå tatt i bruk Rest API som har fungert bra for prototypen, men vi kan vurdere om websocket er en bedre løsning med tanke på live response til det endelige produktet.
- **ESP32-funksjoner:** Fjernkontrollen kan gjøres mer avansert, med flere valgmuligheter direkte på enheten. For eksempel muligheten for stemmestyring.
- **Energisparing:** Ved å integrere automatiserte løsninger kan systemet redusere unødvendig strømbruk. Fjernkontrollen kan bidra til en mer bærekraftig hverdag ved å redusere energiforbruket i hjemmet. For eksempel kan den integreres med smarte energisystemer som automatisk slår av enheter når de ikke er i bruk.
- **Personalisering:** Brukere kan få flere valg for å tilpasse systemet til egne behov. Med flere valgmuligheter kan fjernkontrollen tilpasses hver enkelt bruker, noe som er begrenset i den nåværende prototypen.
- **Logger:** I den videre utviklingen av prototypen vurderer vi å implementere en logger i backend for å forbedre både sikkerhet og feilsøking i systemet vårt. En logger er et verktøy eller en komponent som automatisk registrerer hendelser og handlinger i systemet, og lagrer informasjonen i loggfiler. Disse loggene kan brukes til å spore systemaktivitet, identifisere problemer og analysere uventet oppførsel.

Disse mulighetene gjør systemet fleksibelt og i stand til å møte både dagens og fremtidens krav. Videreutvikling kan gjøre FullKontroll til en komplett løsning for både private og kommersielle smarthjem.

7.6 Avsluttende konklusjon

FullKontroll er planlagt som en løsning som gjør smarthjem-teknologi enkel og tilgjengelig for alle. Fjernkontrollen er valgt som hovedkomponent fordi den gir en intuitiv og ukomplisert brukeropplevelse. Dette designet gjør at brukerne slipper å forholde seg til apper eller komplekse oppsett, noe som senker terskelen for å ta i bruk teknologien.

Samtidig ser vi at valget av en fysisk fjernkontroll kan ha enkelte begrensninger. For noen brukere kan det være behov for mer fleksibilitet, som flere tilpasningsmuligheter eller en annen måte å administrere systemet på. Vi mener likevel at dette designet er et godt utgangspunkt som treffer de fleste i målgruppen.

Gjennom prosessen vil testing og evaluering være avgjørende. Dette sikrer at produktet er brukervennlig og oppfyller målene vi har satt for prosjektet. Ved å fokusere på enkelhet og tilgjengelighet, tror vi at FullKontroll kan bli en løsning som gjør smarthjem-teknologi relevant og praktisk for flere.

8 Vedlegg

Vedlegg 1: Brukermanual for prototypen
Instruksjoner for å sette opp og bruke prototypen. Dokumentet finnes som en separat fil, Brukermanual_prototypen.pdf".

Vedlegg 2: Komponentdiagram
Viser systemets oppbygning og hvordan hovedkomponentene kommuniserer.

Vedlegg 3: Klassediagram backend
Viser backendens struktur og relasjoner mellom Controllers, Services, Repositories, DTO-er, og modeller. Diagrammet illustrerer hvordan data og funksjonalitet håndteres i back-end.

Vedlegg 4: Klassediagram frontend
Viser relasjonene mellom klasser i frontend, inkludert Controllers, Services, DTOs, og modeller som Device og Group.

Vedlegg 5: Sekvensdiagram
Forklarer steg-for-steg-prosessen fra brukerforespørsel til utførelse.

Vedlegg 6: Aktivitetsdiagram
Beskriver flyt og beslutninger i systemet, som API-nøkkelvalidering og kommandohåndtering.

Vedlegg 7: Backend mappestruktur
Viser backendens struktur med mapper som Controllers, Repository, og Service.

Vedlegg 8: Frontend mappestruktur
Oversikt over mappene i frontend, inkludert Controllers, Views, og wwwroot.

Vedlegg 9: Tests mappestruktur
Strukturen for testmodulen, delt inn i backend- og frontend-tester.

Vedlegg 10: Video av prototype
En video som viser prototypens funksjonalitet, i tilfelle feil forekommer ved kjøring.