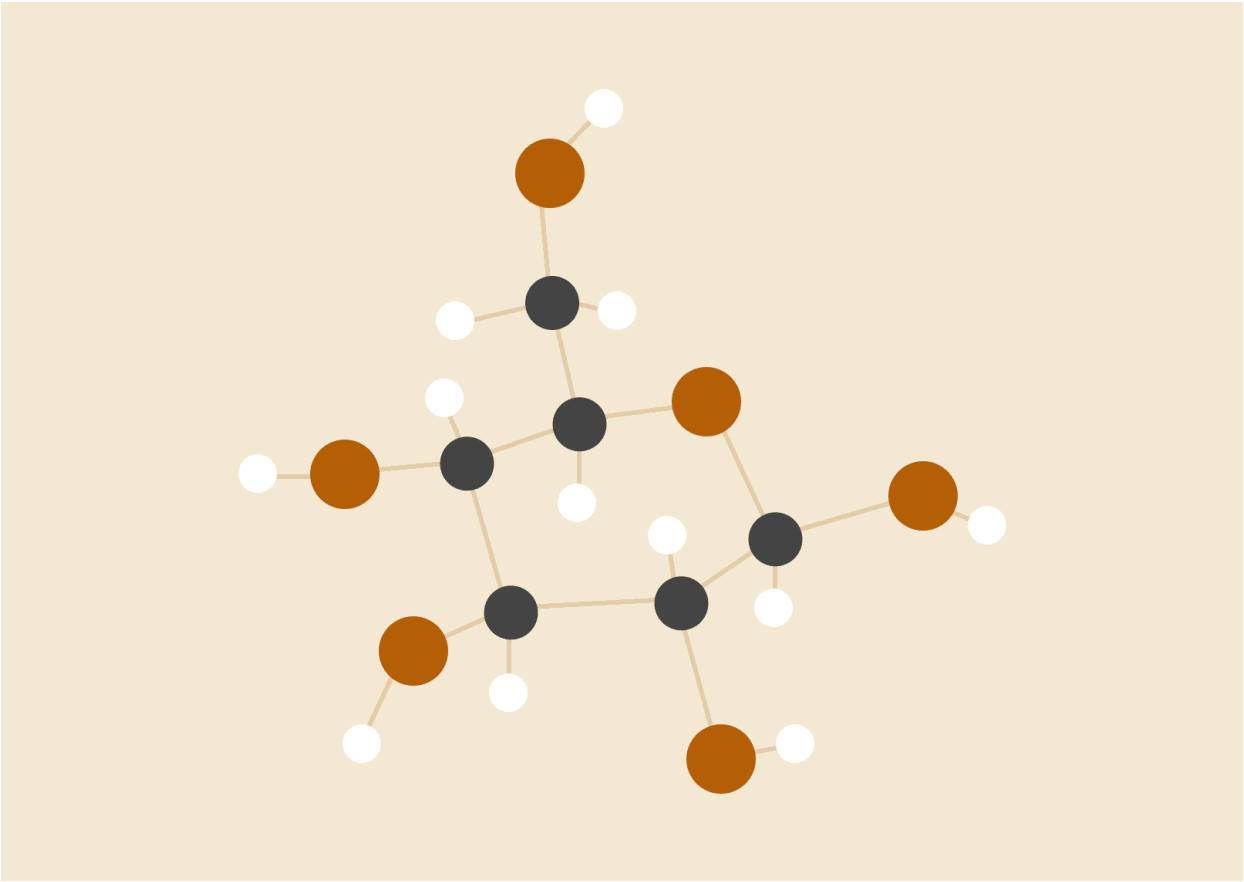
PROCESRAPPORT

Slagter Karlsen - Overvågning af sensor data



Michael Aggerholm

17. Marts 2023

Tværfagligt projekt - Datatekniker med spec. i programmering

Indholdsfortegnelse

Indledning

Projektet er lavet som tværfaglig opgave på Data og kommunikationsuddannelsen, på Tech College i Aalborg.

I rapporten vil jeg beskrive projektets udvikling, fra research og idegenerering til implementering og til sidst evaluering.
Jeg vil gennemgå de teknologier og metoder, jeg har brugt, samt de udfordringer jeg er stødt på, og hvordan jeg har løst dem.

Formålet med rapporten er at give indblik i min proces og tankegang, samt at dokumentere projektet for fremtidig reference.

Hvis ønsket, kan det være en fordel at have adgang til projekt filerne under læsning, de findes frit tilgængeligt her på Github:
https://github.com/MichaelAggerholm/Interdisciplinary_Project

Læsevejledning

I mit projekt har jeg valgt at opdele min dokumentation i to halvdele, procesrapport og produktrapport, for at give en mere detaljeret gennemgang samt beskrivelse af projektet, som omfatter både det endelige produkt, samt processen der er gennemført for at opnå det. Jeg håber at det vil give læseren mere overblik, samt en bedre forståelse projektet.

Dette er procesrapporten, som bør læses først, da den beskriver den proces, det har været for at nå frem til det endelige produkt. Den indeholder information om projektets planlægning, design, udvikling, test og implementering. Procesrapporten giver også en detaljeret beskrivelse af de udfordringer, der er stødt på undervejs, samt hvordan jeg har løst dem.

Forord

Overvågning af sensor data for Slagter Karlsen, er en opgave der er bygget på baggrund af, at der i flere mindre virksomheder som håndterer fødevarer, stadig håndterer temperatur og fugtighedsmålinger i opbevarings og kølerum manuelt, altså ved at aflæse et hygro- termometer og notere dato samt målinger i et excel ark eller lignende.

I projektet er målet, at der løbende skal måles temperatur og fugtighed fra en embedded enhed med et sensor modul.
Den målte data, skal via websocket sendes som POST request parametre til et web API som ved modtagelse gemmer målingerne i en MySQL database med dato, tidspunkt og hvilken enhed / placering der er tilknyttet den enheden som har foretaget den pågældende måling.

Der opbygges en cross-platform app med listevisning af sensor data fra de tilkoblede sensorer, opdelt i de givne placeringer.
Dataen fra de tilkoblede sensorer skal komme fra GET- request til web API'et i form af JSON og vises i en overskuelig liste, som indeholder de seneste 20 målinger.

Projektet er opbygget til at det skal være let for brugeren at tilføje yderligere enheder, sensorer, placeringer og typer af målinger, derfor er databasen opbygget med hver del i hver sin databasetabel som via relationer kædes sammen til en komplet sensor måling.

Case

Slagter Karlsen ønsker en mobil app, hvor temperatur og fugtighed fra forskellige rum i virksomheden kan overvåges i realtid.

App'en skal være hurtig og have en overskuelig oversigt over nuværende samt en del af tidligere målinger. Det skal desuden fremgå hvilke enheder og sensorer, der er i hvert enkelt rum.
Det skal være muligt at få den nuværende temperaturmåling fra den mobile enhed, hvis en medarbejder ønsker en tjek måling uden at skulle flytte eller opsætte ny enhed.

Slagter Karlsen vil gerne have mulighed for at kunne tilføje yderligere sensorer eller enheder til systemet på sigt hvis nødvendigt. Målinger skal gemmes og fremgå med dato og tidspunkt, hvis det skulle blive nødvendigt at dokumentere målingerne i fremtiden.

App'en skal fungere på både Android og IOS, og administreringen af enheder skal være tilgængelig via web browser.

Problemformulering

Det skal vi en app, være muligt at se en listevisning af sensor målinger fra tilknyttede enheder, det skal være muligt at tilføje flere enheder, sensorer og målings typer til systemet.

Projektplanlægning

til projektudarbejdelsen er der valgt at benytte kanban som agil metode, da det er en metode der er nem at forstå og anvende, det giver mulighed for at få et overblik over arbejdet og dermed sikre at der ikke er for mange opgaver i gang på en gang.
Der er valgt at benytte et kanban board i moqups, som er et online værktøj til at lave wireframes og mockups.
På moqups er der lavet tre forskellige kanban boards:
Et som visualiserer opstarten af projektet (Se bilag: Kanban board, projektstart, s. 15).
Ét som visualisere midtvejs i projektforløbet (Se bilag: Kanban board, midtvejs, s. 16).
Ét som visualiserer sidst i forløbet (Se bilag: Kanban board, færdiggørelse, s. 17).

Inden udviklingsprocessen påbegyndes er der taget nogle valg, for at komme frem til de valg og elementer som benyttes i projektet er der ved opstart udarbejdet et mindmap (Se bilag: Mindmap, s. 14), i mindmap er der noteret diverse muligheder samt elementer der kunne gøre sig gældende for at nå det ønskede udgangspunkt.
Efterfølgende er der sorteret og plukket elementer som tilsammen er relevante for opbygning af et solidt projekt, samt mængde af opgaver som forventes at kunne udføres i den givne tidsramme.

Herefter er hver opgave opdelt, defineret og beskrevet i kravspecifikation (Se dokumenteret kravspecifikation i produktrapport s 10), i kravspecifikationen er der valgt at kombinere opgaver som CRUD funktionaliteter, det er valgt for at gøre kravspecifikationerne mere overskuelige.
Efterfølgende er der defineret tests pr. opgave, i en accepttest oversigt (Se dokumenteret accepttest i produktrapport s. 13).

ved projektets opstart er der lavet en estimeret tidsplan (Se bilag: Estimeret tidsplan, s. 12), denne skal så vidt muligt følges, dog noteres der en realiseret tidsplan løbende, for til sidst i projektets forløb at kunne sammenligne og bruge resultatet til forbedring fremadrettet. (Se bilag: Realiseret tidsplan, s. 13)

Som start i udviklingsprocessen udarbejdes der et database diagram, hvor det sikres at logikken i relationer samt opbyggelse giver fundamentalt mening (Se bilag: Database diagram, s. 18).

Herefter opsættes et klassediagram som giver overblik over de controllers og klasser som der skal opsættes i backend. (Se bilag: Klasse diagram, s. 19).

Efter fastsættelse af database samt klasse logik, starter udviklingsfasen som beskrevet i kanban board og tidsplan, den er opdelt i tre faser:

- Web API
- Embedded
- App

Under hele projektforløbet noteres der løbende tilbageblik på opgaver, udfordringer og genovervejelser, disse noter bliver til sidst brugt til reflektering samt læring fra de forskellige opgaver og udfordringer som har opstået undervejs.

Under hele projektets forløb, bliver der noteret daglige opgaver samt eventuelle udfordringer der har været de specifikke dage, i logbog som findes fra side 22.

Metodevalg og teknologi

Til projektet er der foretaget nogle valg, metoder og teknologier, de er herunder beskrevet, samt hvilke tanker der er taget i den proces.

Moqups

Moqups er et onlineværktøj, hvor det er muligt at oprette diverse visualiseringer, som tidsplan, kanban board osv. Det er valgt til projektet da det giver mulighed for at have alle disse et samlet sted, og med et bredt udvalgt i værktøjer til at levere et professionelt stykke arbejde i form af visualisering af planlægning, dokumentation og procesforløb.

ESP8266 NodeMCU

Under den embeddede udvikling, er det en nødvendighed at have netværksadgang for at kunne sende sensor data, på en let og overskuelig måde til et web API, i den forbindelse er der valgt et ESP8266 board, da det er udstyret med indbygget WiFi.
Prisen er forholdsvis lav sammenlignet med andre udviklingsboards, da det specielt er egnet til mindre IoT-opgaver.
En mulig forbedring ift. stabilitet, hastighed og sikkerhed, vil være at opgradere udviklingsboard til en ESP32, den har en højere kostpris men kommer med en stor række fordele til mere avancerede IoT-projekter. (Se bilag: Embedded opsætning, s. 21).

DHT11

En DHT11 understøtter målinger af temperatur og fugt, som er to sensor elementer der er lette at anvende og arbejde med i et projekt som dette.
præcisionen er høj taget prisen i betragtning. En mulig forbedring til projektet vil være at opgradere til en DHT22, som dog også kun måler temperatur og fugt, men med en højere præcision. (Se bilag: Embedded opsætning, s. 21).

Symfony

Valget af framework til Web API faldte på Symfony 6, da det er et hurtigt PHP baseret framework, som ikke er overfyldt af funktionalitet som i projektets omfang ikke er relevant.
Symfony er komponentbaseret, hvilket vil sige at der er mulighed for kun at inkludere relevante komponenter til projektet.
Havde projektet haft et større omfang, eller en tungere backend ift. funktionalitet, kunne der havde været benyttet Laravel, dog er det en personligt beslutning at bevare en hastighedsmæssig fordel fremfor yderligere inkluderede biblioteker.

Doctrine

Doctrine er en PHP-baseret datamapper, som gør det muligt at integrere med relationelle databaser ved at arbejde med objekter frem for SQL forespørgsler.
Det gøres ved hjælp af entity klasser som repræsenterer tabeller i databasen, og objekter, som repræsenterer rækker i tabellerne.
Doctrine sørger på den måde for at håndtere kommunikationen mellem objekter og databasen.
Der findes et bredt udvalg af ORM til PHP udvikling, mine grunde er listet herunder:

- Integration: Doctrine ORM er standard ORM-løsning i Symfony-frameworket, og det er designet til at arbejde sømløst sammen med Symfony-komponenterne.
- Objekt-orienteret tilgang: Doctrine gør det muligt at arbejde med databasen på en objekt-orienteret måde, hvilket gør koden mere intuitiv og lettere at vedligeholde.
- Database-abstraktion: Abstraherer database adgangen og giver en mere sikker måde at kommunikere med databasen på. Det betyder også, at man kan arbejde med flere forskellige databaser uden at skulle ændre på sin kode.
- Dokumentation: Doctrine er en populær ORM-løsning inden for PHP-fællesskabet og har en stor og aktiv brugerbase, der gør det let at finde hjælp.

React-Native

App frameworket React-Native er blot ét af mange store frameworks til opbygning af App's som giver mulighed for cross-platform udvikling.

Min begrundelse for valg af React-Native er stort set baseret på en nysgerrig interesse indenfor React som læring af et Javascript baseret framework.

Udover det giver det som udvikler et behageligt samt hurtigt framework at arbejde i. React-Native er komponentbaseret hvilket gør det nemt at genanvende kode samt logik.

udover at arbejde i React-Native er der i projektet opsat nogle teknologier som er bygget til at arbejde sammen med React udvikling:

Expo

Expo er et open-source værktøjssæt som gør det nemmere at udvikle mobil apps, det giver nemlig mulighed for at reducere den tid og kompleksibilitet der kræves for at oprette projektet samt installere tredjeparts pakker.
Hvis app'en en dag skal offentliggøres eller lægges live, er det via Expo muligt at gøre dette, stort set helt automatiseret.

Metro

Metro benyttes til at køre React-Native app på diverse enheder, det vil sige at hvis en kode-snippet, pakke eller lignende ikke er éns mellem eksempelvis IOS og Android, går Metro ind og sørger for at enheden får den korrekt pakke, samt at det virker efter hensigten.

Det giver også let mulighed for udvikling på egen enhed som emulator, eller at benytte en virtuel emulator af enten IOS eller Android styresystem. Det giver også mulighed for udvikling samt debugging i en webbrowser, hvilket gør fejlfinding processen mere overskuelig.

Fetch / Distribuering af data mellem IoT / Api / App

Som udgangspunkt bør der nok anvendes en websocket eller noget lignende SignalR. Dog grundet projektets tidsramme og omfang, er det valgt at der benyttes loops..

1. Det vil sige at den embeddede enhed sender målingsdata til api hvert sekund.
2. Målingsdata Postes direkte ind i Api via HTTP-Request.
3. Api data hentes fra App hvert tredje sekund for konstant at holde brugeren opdateret med aktuel data.

Versionsstyring

Som version styringsværktøj benyttes der Github, der er mange andre valg af platforme til versionsstyring, men her er nogle af mine argumenter for valget:

- Personlig kendskab.
- Mulighed for at gemme og følge historikken af kodeændringer.
- let at integrere direkte fra terminal.
- Let tilgængelig og veldokumenteret dokumentation

Postman

Der var stor personlig uenighed i valg af teknologi til test af API, valget stod mellem Postman og Swagger. Dog giver Postman mulighed for opsætning af automatiseret API testkald samt automatisk generering af test data ved bulk-testing.

Her er yderligere punkter som jeg fandt interessante, og som gjorde mit valg:

- Postman understøtter en bred vifte af API-protokoller, såsom HTTP, HTTPS, REST, SOAP, GraphQL og andre, hvilket gør det muligt at teste forskellige typer API'er.
- Automatisering: Postman understøtter også automatisering af API-tests gennem scripting og integration med Continuous Integration/Continuous Deployment (CI/CD)
- Adgang til mange ressourcer og vejledninger, der kan hjælpe med at løse eventuelle problemer eller spørgsmål.

Jeg har løbende brugt Postman collections til at teste diverse request metoder i koden (Se bilag: Postman Request Collection, s. 20)

Konklusion

Som udgangspunkt, er projektet forløbet rigtigt godt. Der er blevet undersøgt teknologier, værktøjer, metoder samt arbejdet med den komplette stak af et projekt som indeholder alt fra tankeproces, planlægningsproces og udviklingsproces.

Projektets forløb har givet nogle erfaringer som helt sikkert har gavnet ift. fremtidige projekter samt valg af teknologier.
Fremadrettet vil der fra min side, blive lagt stort fokus på at arbejde videre med komponentbaseret udvikling, da det giver rigtig god mening at opdele kode i specifikke komponenter, på denne måde bliver koden langt mere genanvendelig, læsbar og lettere at vedligeholde på sigt.

Dog er det sidste gang at mit fokus bliver lagt så lavt på selve installationsfasen, projektet her, er svært at installere for brugeren. Fremadrettet skal der prioriteres mere tid og energi på at få lavet et ordentligt setup i evt. docker miljø, og heri kan det lettes endnu mere med bash scripts som automatisere opstartsprocessen.

Set i bakspejlet, vil jeg personligt gerne have haft en hel del mere docker samt automatisering og test med i projektet.
Til fremtidige projekter vil jeg undersøge teknologier som github, CI/CD, automatisering af unit tests ved upload, scripts til projektopstart, for selvom at jeg her beskriver det som en ulempe for brugeren, har det også været tidskrævende i udviklingsprocessen at først skulle starte embedded, derefter docker, web api. emulator, app og forskellige udviklingsprogrammer til hver del.

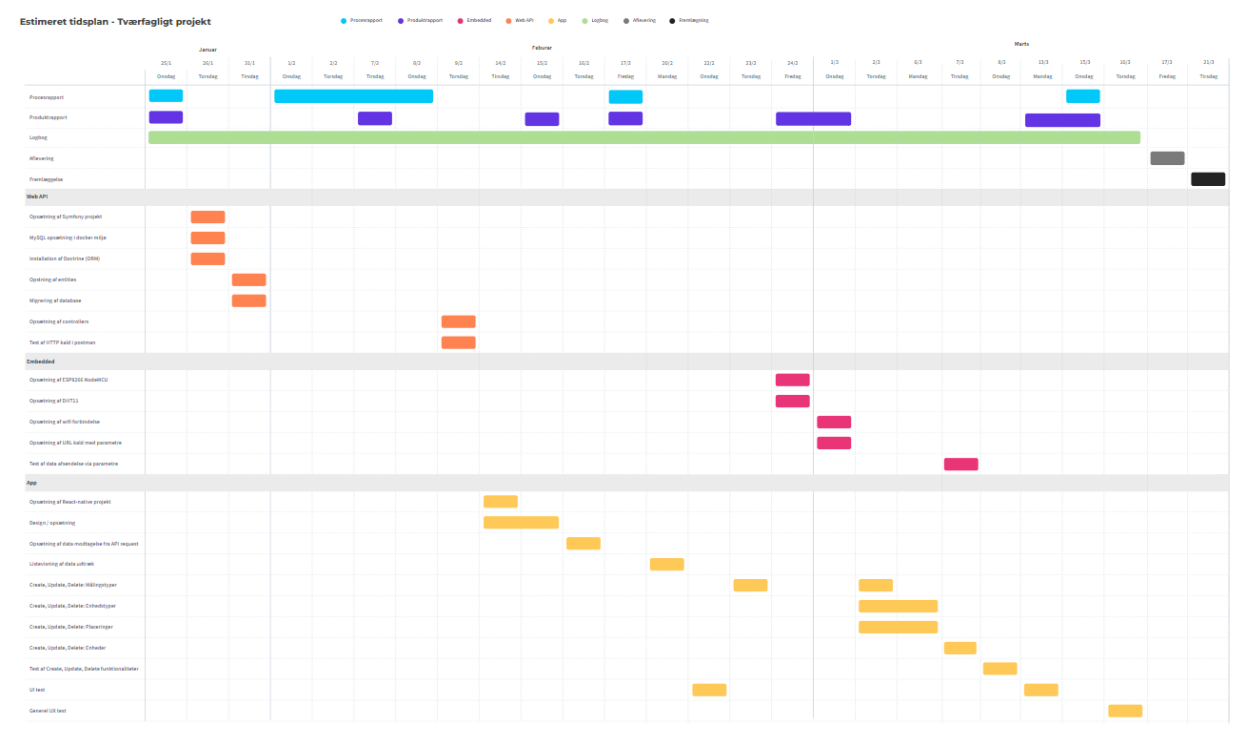
I forhold til udvikling af app, var det forventet at størstedelen af tiden skulle benyttes her på, da jeg ved projektets start stod med absolut ingen erfaring herom, valget på React-native som framework var udelukkende baseret på en interesse for læring af javascript.
Det havde sikkert ikke lettet processen at vælge ethvert andet framework, og jeg er helt sikkert min erfaring rigere, da jeg nu kan sige at dette ikke bliver mit sidste projekt med React inkluderet.
Det er dog sikkert at jeg ikke vil på lige så dybt vand, når det hedder svendeprøve frem for tværfagligt projekt.

Bilag

Herunder er links samt billeder til mine ressourcer samt tredjeparts applikationer, det er blandt andet tidsplaner, kanban board, mindmap og klassediagram.

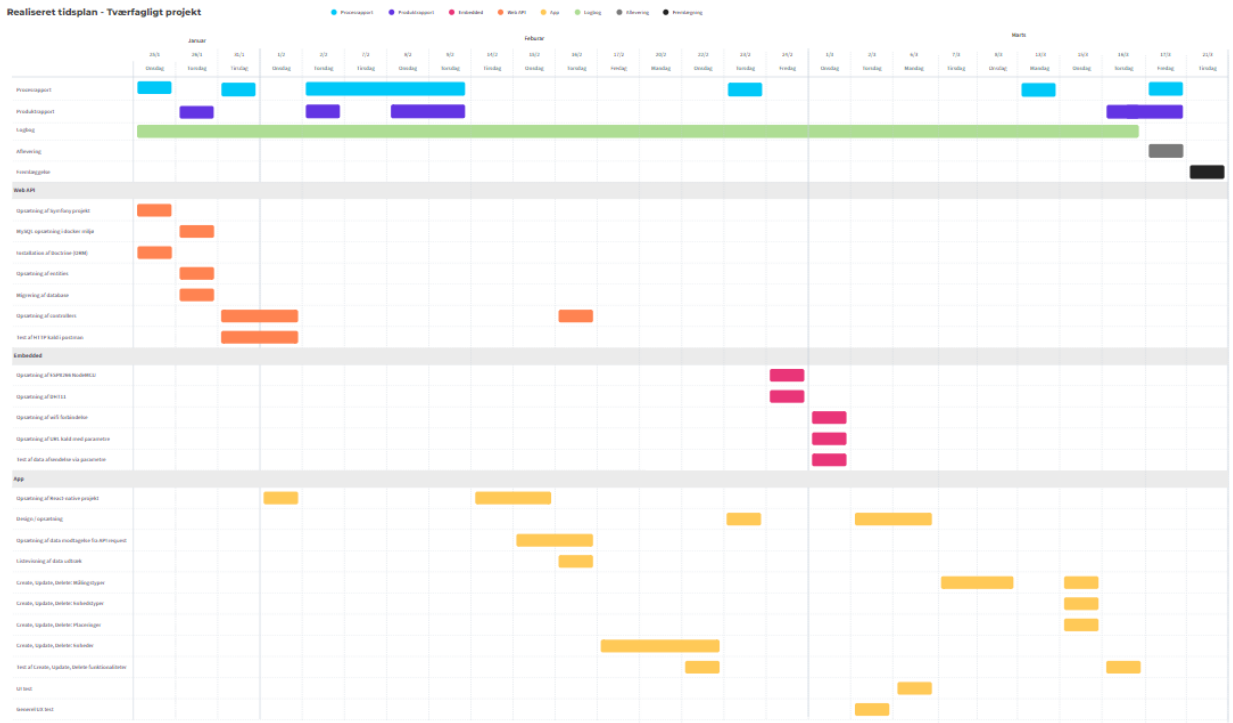
Estimeret tidsplan

[Link til estimeret tidsplan]



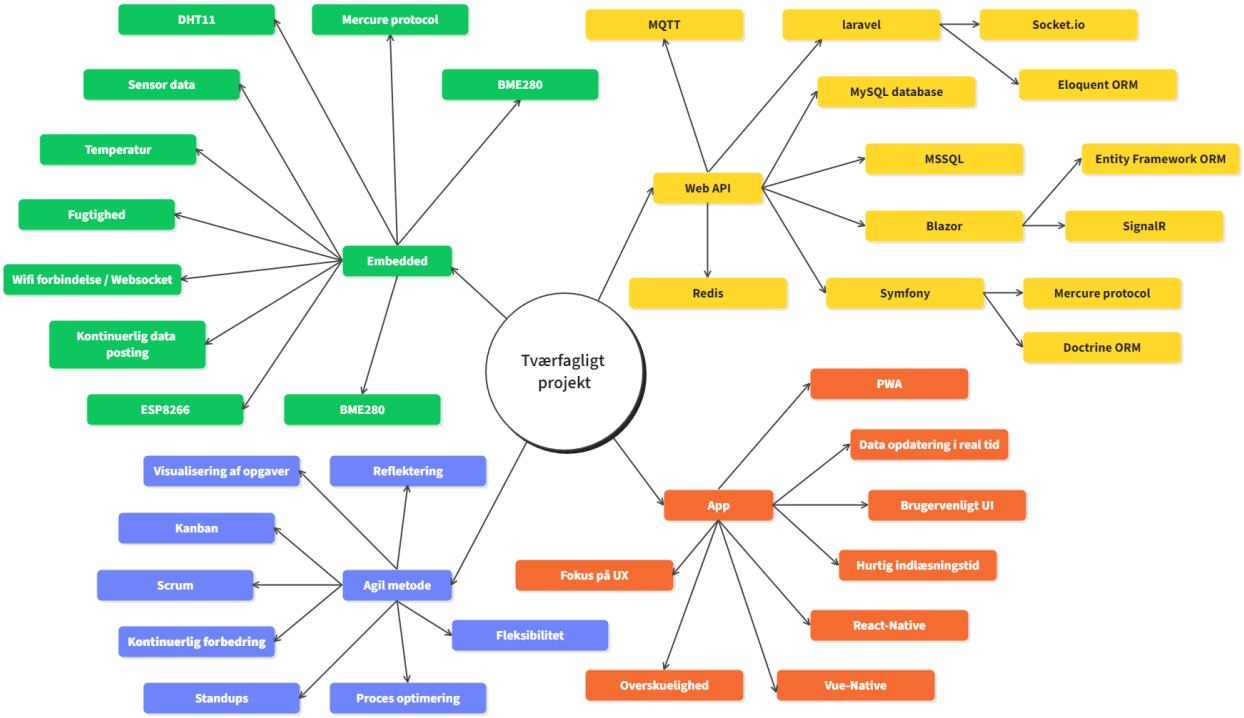
Realiseret tidsplan

[Link til realiseret tidsplan]



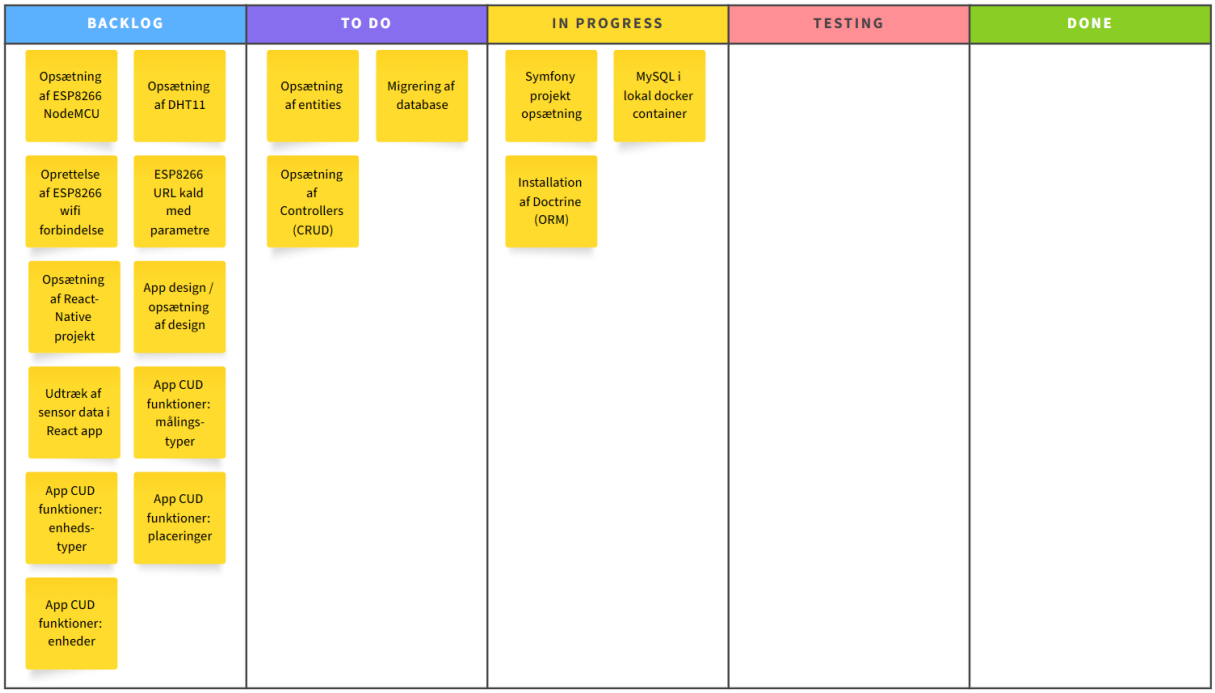
Mindmap

[Link til mindmap]



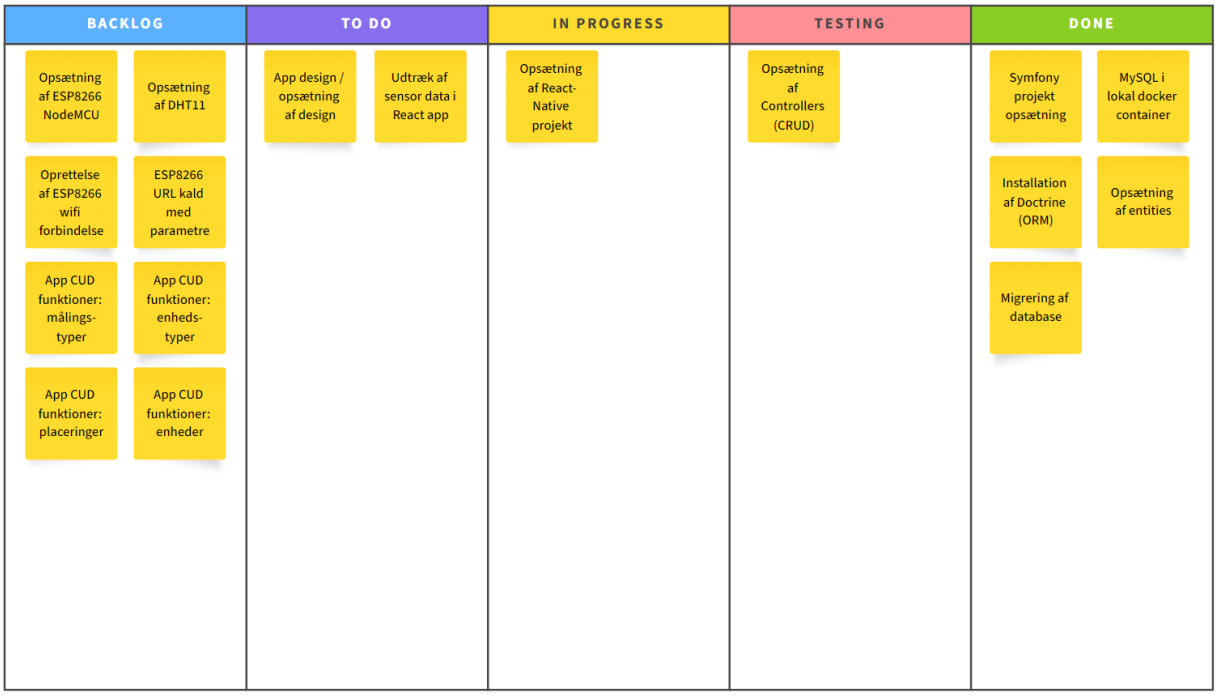
Kanban board, projektstart

[Link til Kanban Board]



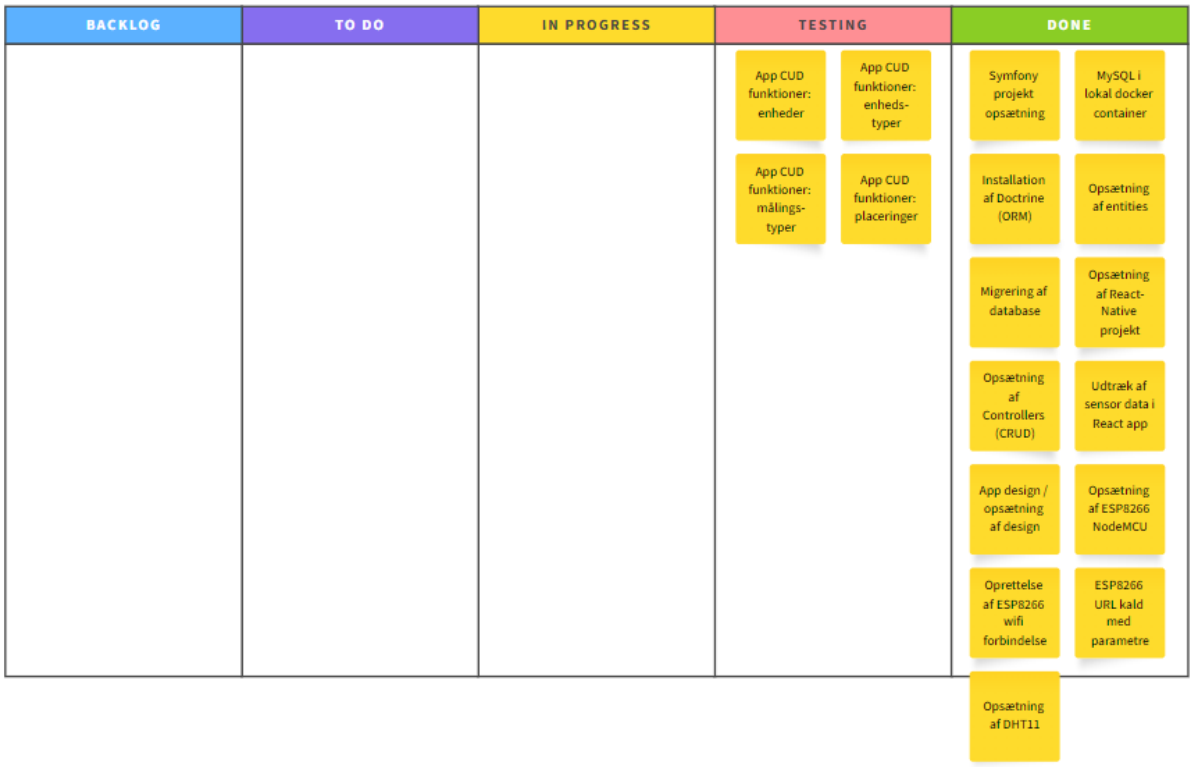
Kanban board, midtvejs

[Link til Kanban Board]



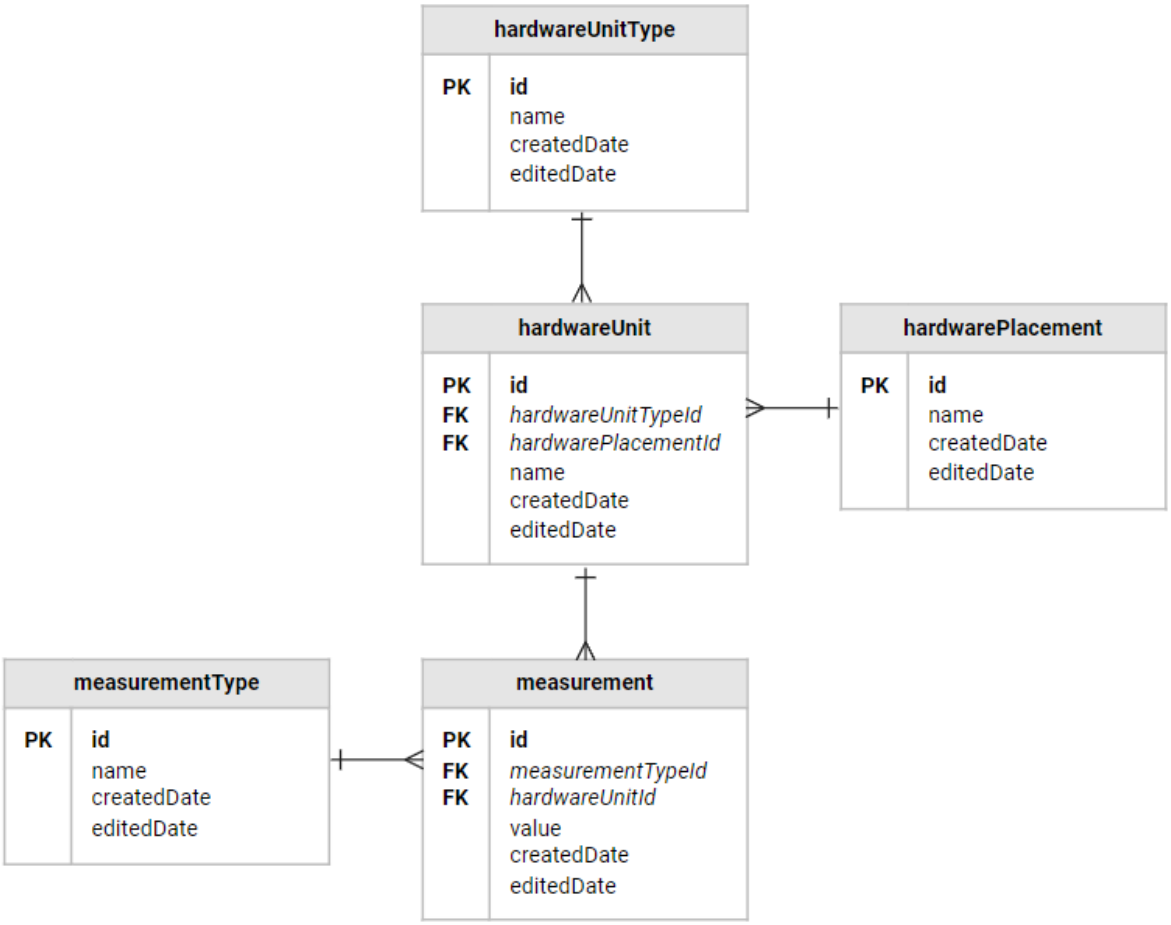
Kanban board, færdiggørelse

[Link til Kanban Board]



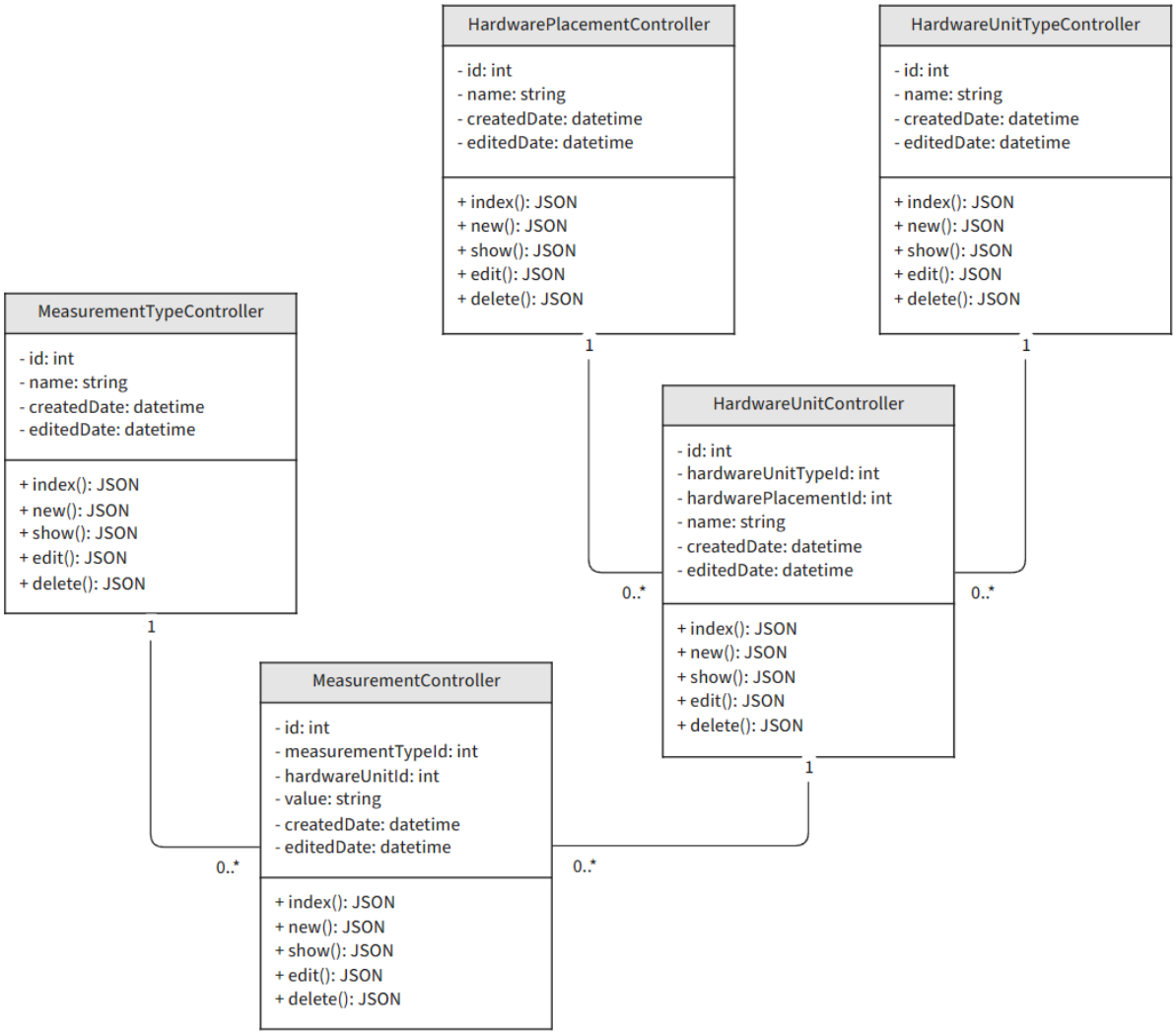
Database diagram

[Link til diagram]



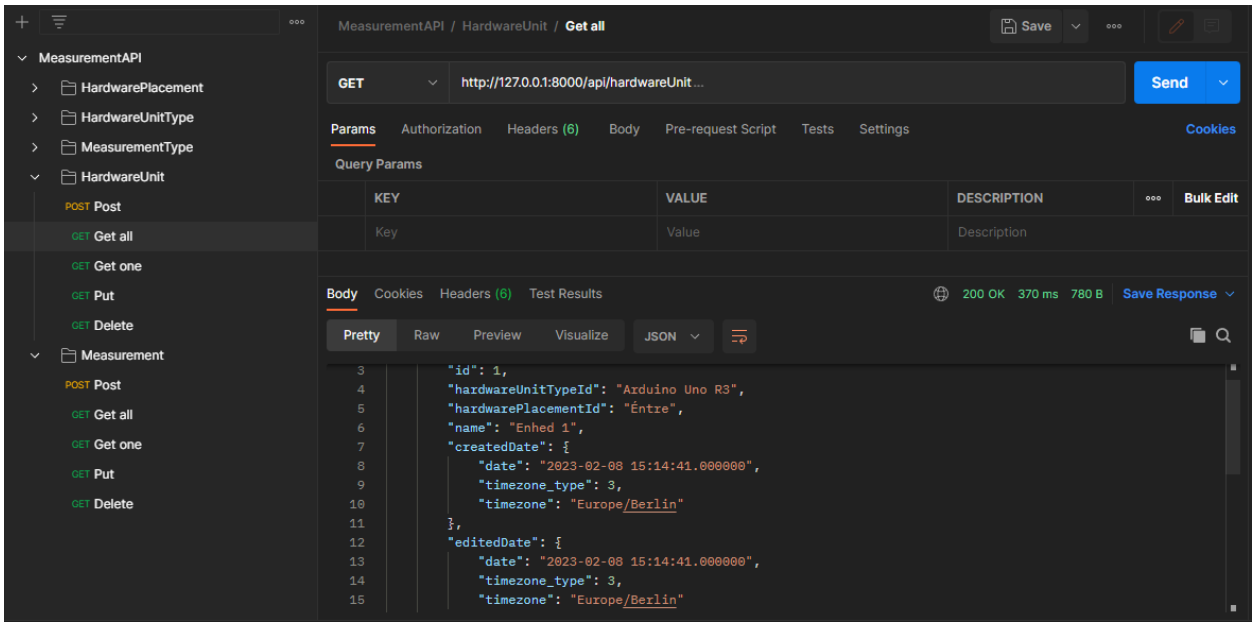
Klassediagram

[Link til diagram]



Postman Request Collection

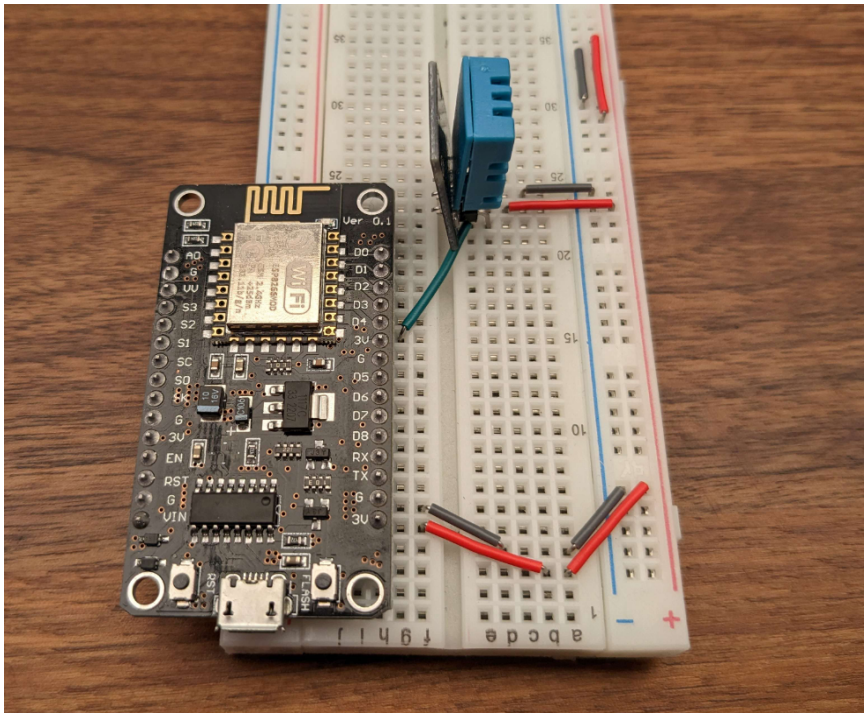
I Postman er der tilføjet test af alle HTTP request methods, i en collection, herefter køres en test af den samlede collection, for at bekræfte succes af CRUD metoder.

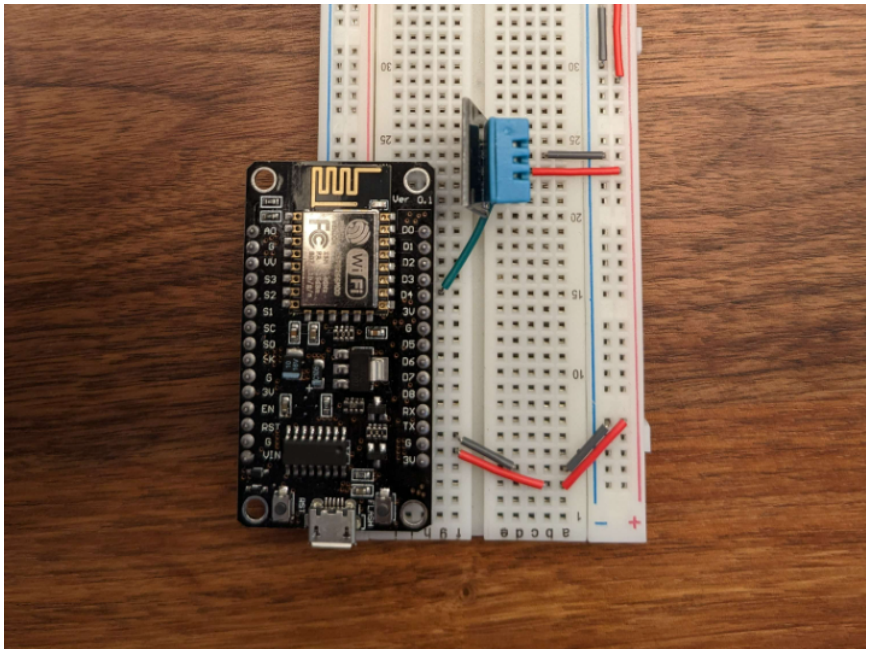


I projektmappen “Project_assets” ligger der en fil til import af fuld http request collection som let kan importeres til Postman, hvis ønsket.

Embedded opsætning

Opsætningen af NodeMCU ESP8266 med DHT11 på breadboard.





Logbog

25/01-2023 - Onsdag

Opstart af tværfagligt projekt.
Opbyggelse af mindmap samt valg af metoder.
Indledning og case beskrivelse i procesrapport.
Påbegyndt opsætning af web API.
Opsætning af database diagram.

26/01-2023 - Torsdag

Fortsættelse på web API, i dag med MySQL database i docker miljø.
Påbegyndt produktrapport samt beskrevet database opsætning heri.

31/01-2023 - Tirsdag

Opsætning af klassediagram.
Opbyggelse af Controllers samt tests i Postman.

01/02-2023 - Onsdag

Påbegyndelse på App del, valg af framework samt opstart af projekt.
Status samt fremvisning af web API.

02/02-2023 - Torsdag

Udarbejdelse af problemformulering.
Opsætning af Estimeret tidsplan samt kravspecifikation.
Opbygning af forord i procesrapport.
Færdiggørelse af indledning i procesrapport.

07/02-2023 - Tirsdag

Skrevet læsevejledning i procesrapport.
Påbegyndt dokumentation for projektplanlægning i procesrapport.
Påbegyndt beskrivelse af metodevalg og teknologi i procesrapport.

08/02-2023 - Onsdag

Aflevering af estimeret tidsplan, kravspecifikation, case og problemformulering.
Opbyggelse af realiseret tidsplan.
Yderligere opsætning af procesrapport samt rettelser heri.
Opsætning af Accepttest oversigt.

09/02-2023 - Torsdag

Tilføjelser samt rettelser til projektplanlægning og metodevalg.
Fortsættelse på diagrammer: Kanban boards og flowchart.

14/02-2023 - Tirsdag

Omstrukturering af hele react-native applikationen, nuværende setup er for tungt, der er ikke nok fejlbeskeder og debug metoder i android studio.. jeg mangler også en måde at teste IOS, samt et hurtigere udviklingsmiljø.

15/02-2023 - Onsdag

Ny opsætning af react-native app, denne gang med emulator direkte i CLI, der er opsat iphone / android emulator enheder som startes og køres direkte fra powershell, og med real-time opdatering af react kode i visual code, derudover er det lykkedes at opsætte en cli konsol log hvor jeg kan overvåge mine egne debug logs.!

16/02-2023 - Torsdag

Fetch af data fra API med succes.
Indblik i react-native syntax, begyndelse af data fremvisning i listeform samt indblik i design opsætning.

17/02-2023 - Fredag

Påbegyndt opsætning af CRUD funktionalitet for enheder i App.
Lidt besværligt.. Refactoring af React-Native app da min oprindelige tilgang ikke har været korrekt.

20/02-2023 - Mandag

Refactoring af App da det kan gøres lettere ved opsætning af komponenter.
Success med oprettelse af enheder fra App

22/02-2023 - Onsdag

Opsætning af Edit og Delete funktionalitet for enheder fra App.
Test af alle CRUD funktionaliteter for enheder fra App.

23/02-2023 - Torsdag

Påbegyndt UI opsætning, nu hvor funktionalitet begynder at være på plads.
Opfølgning og fortsættelse på procesrapport.

24/02-2023 - Fredag

Opsætning af ESP8266 NodeMCU
Opsætning af DHT11 sensor på ESP8266

01/03-2023 - Onsdag

Wifi forbindelse fra ESP8266
Afsendelse af sensor data som parametre fra ESP8266.
Test af data modtagelse i web api, virker med stor success og før end forventet.

02/03-2023 - Torsdag

UI opsætning af app, udtræk af kode til komponenter for mere genanvendelig kode.
Kort test af om det giver mening rent UX-wise med nuværende opsætning og navigation.

06/03-2023 - Mandag

Færdiggørelse af UI, samt test af nuværende UI på forskellige enheder.

07/03-2023 - Tirsdag

Påbegyndt opsætning af Målingsenheder i app, Fetch af data samt CUD funktioner.

08/03-2023 - Onsdag

Fortsættelse op opsætning af CUD funktionalitet på målingsenheder i App.

13/03-2023 - Mandag

Udelukkende en dag til procesrapport gennemgang samt “næsten” færdiggørelse.
Lidt indblik i hvad yderligere en produkt rapport skal indeholde, dog intet reélt arbejde heri.

15/03-2023 - Onsdag

I dag er der fart på!!
Færdiggørelse af målingstyper opsætning i App!
Opsætning og funktionalitet for enhedstyper i App!
Opsætning og funktionalitet for placeringer i App!

16/03-2023 - Torsdag

Produkt rapport arbejde. Mere produkt rapport arbejde. Mest produkt rapport arbejde.
Test af alle CRUD funktionaliteter i App.

17/03-2023 - Fredag

Færdiggørelse af Projektrapport.
Gennemgang samt smårettelser i Procesrapport.
Aflevering!