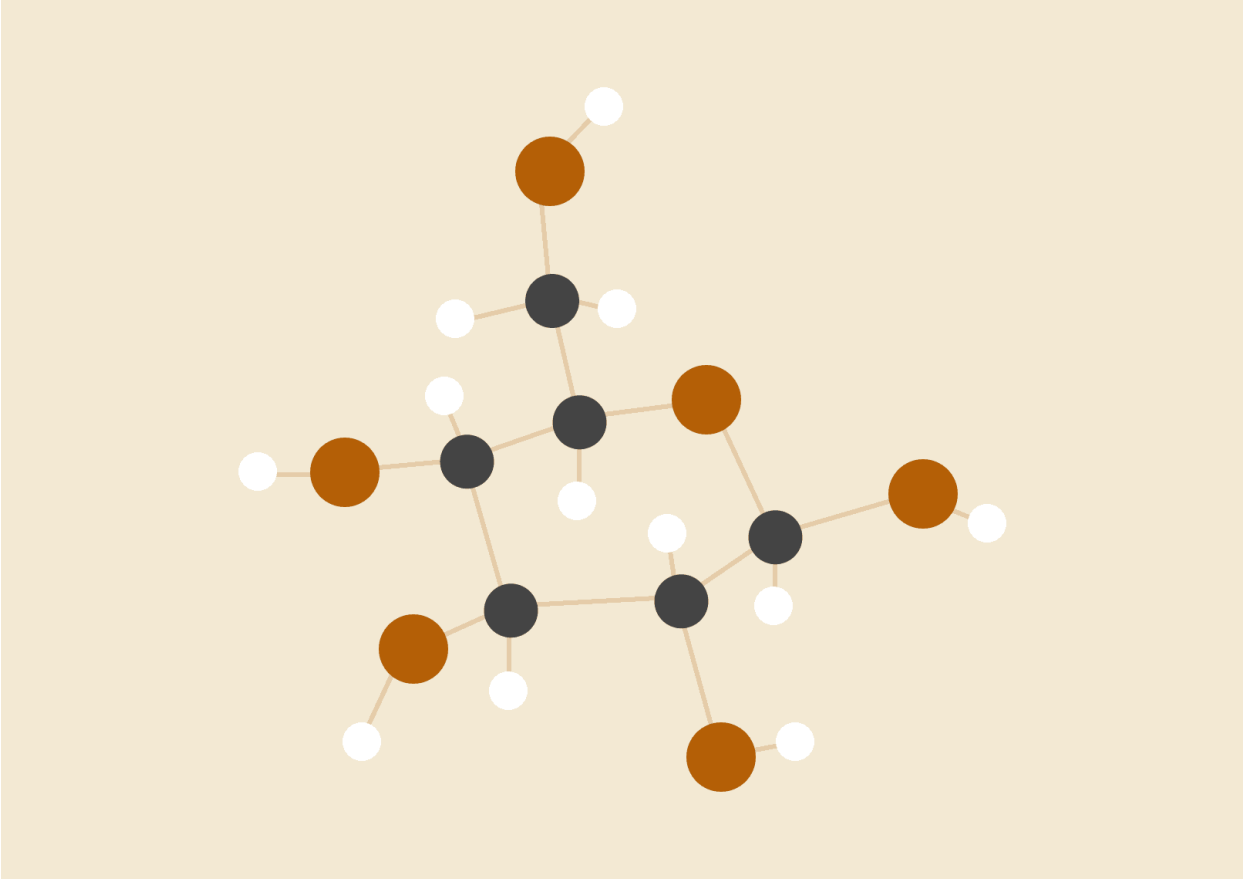
PRODUKTRAPPORT

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.

Læsevejledning

Dette er produktrapporten, som bør læses efter procesrapporten, da den beskriver det endelige produkt.
Produktrapporten indeholder teknisk dokumentation, herunder beskrivelser af produktets funktioner, specifikationer og tekniske detaljer, samt detaljerede instruktioner om, hvordan produktet skal installeres, konfigureres og bruges, så brugerne kan få mest muligt ud af produktet.

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.

Arkitektur

Produktets arkitektur består af tre separate platforme: en målingsenhed, et API og en app. Målingsenheden er ansvarlig for at indsamle og registrere sensor målinger, som sendes videre til API'et. API'et fungerer som en backend og håndterer data, herunder behandling af data fra målingsenheden og lagring af data i en database. Appen er brugerens tilgang til produktet og giver brugeren adgang til at se data, der er blevet indsamlet og behandlet af målingsenheden og API'et.

Arkitekturen er bygget på et distribueret system, hvor hver platform er adskilt fra hinanden og kan skiftes ud eller opdateres uafhængigt. Dette gør det muligt at opgradere en specifik platform uden at påvirke de andre platforme. API'et bruger standard kommunikationsprotokoller og RESTful API designprincipper for at gøre det let at integrere med andre systemer eller tjenester.

Arkitekturen er designet til at være fleksibel og skal kunne tilpasses til forskellige sensorer og målinger, der skal registreres. Dette er opnået ved at gøre målingsenheden udskiftelig og tilpasse API'et til at modtage data fra forskellige sensorer. Appen er også designet til at kunne opdateres og tilpasses herefter.

Teknisk produktdokumentation

Teknisk beskrivelse af de tre enheders udviklede struktur og miljø.

Embedded

Den embeddede løsning er i projektets tilfælde beskrevet som lavet på en specifik enhed med et specifikt sensormodul, det gør sig dog ikke gældende for funktionsdygtigheden af projektet, da projektet er udarbejdet til at kunne håndtere alle typer sensor data, fra alle typer enheder, som udgangspunkt behøver målingsdata slet ikke at komme fra en embedded enhed, så længe at det kan leveres til Api'et via netværks request, kan Api'et håndtere dataen i den relationelle database, knyttet til korrekte målingsdata som defineres ved opsætning / tilknytning af målingsenheden.

Web Api

backend delen som agere Api, er opbygget i PHP frameworket "Symfony" som benytter en ORM (Doctrine). koden er opsat samt struktureret til at overholde følgende krav:

- Moderne teknologier, herunder Symfony og Doctrine.
- Brug af Entities til håndtering af database objekter.
- Struktureret REST opsætning som følger best-praksis api struktur.
- Robust og skalerbar opsætning af backend.

Databasen er i dette tilfælde opsat i ekstern docker container, men er ikke afgørende for backend, da backend kan tilknytte enhver database uanset type.

I udviklingen af backend, er der anvendt "Repository Pattern" hvilket er et udviklingsmønster som benyttes til at strukturere koden på, for at adskille logik fra datahåndtering, på den måde kan hver del ændres uafhængigt af hinanden.
Det gør koden mere modulær samt lettere at vedligeholde på sigt, samt hvis en dag database teknologien skiftes, kan det ændres uden at ændre i selve applikations logikken.

App

Mobil applikationen er bygget i Javascript frameworket React-Native, dette understøtter cross-platform og giver brugere adgang til applikationen uanset valg af mobil OS.
Applikationen er bygget for sig selv, uden at tage højde for hvor data kommer fra, så længe data kan tilgås via netværk, er det muligt at implementere i App'en, det giver brugeren frihed til at implementere data fra andre enheder hvis ønsket på sigt.
Applikationen er lavet i minimalt stilrent design som følger almen hånd-øje taktiske designvalg som anbefalet af miljøstyrelsen, og derfor sikret fremadrettet ift. UI / UX og bør påskønnes af enhver.

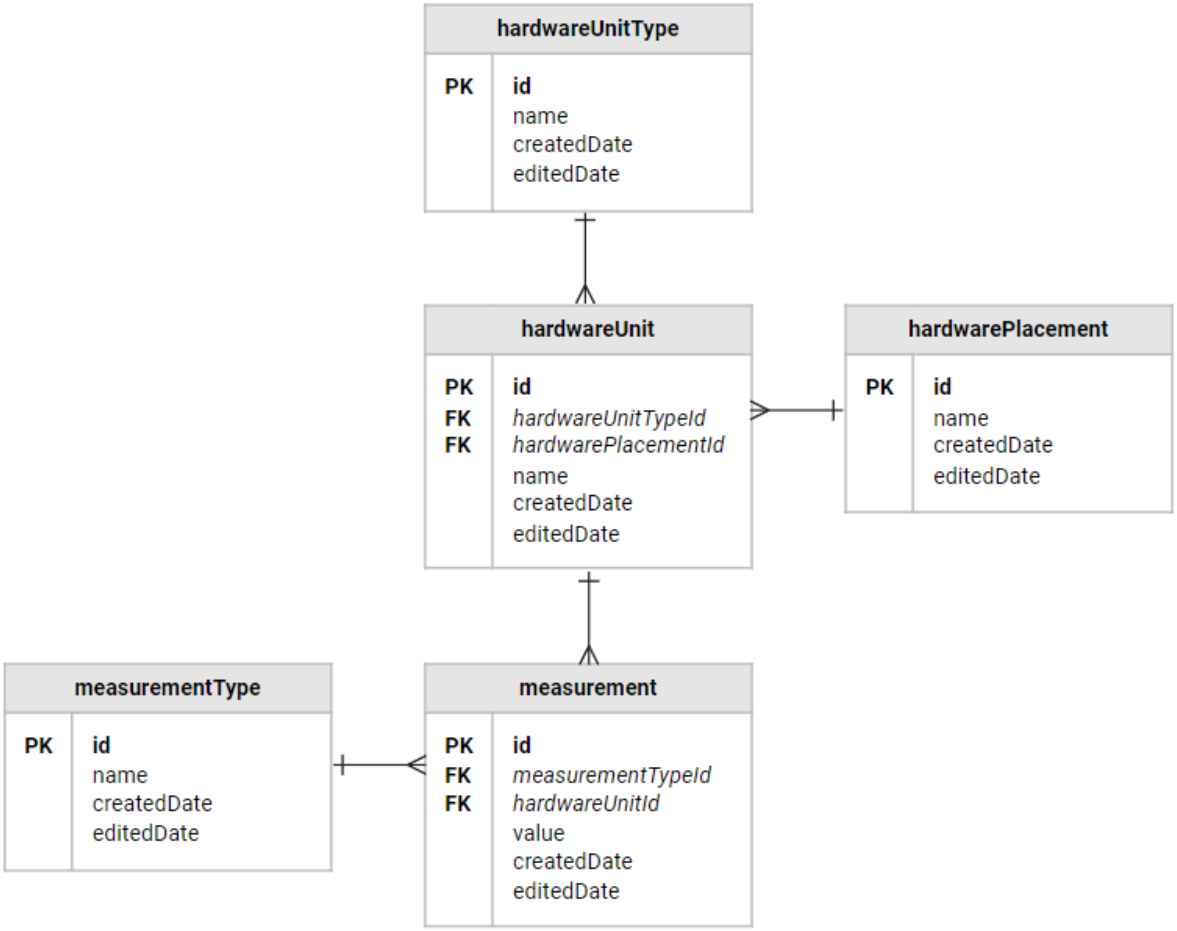
Under udviklingen af applikationen er der taget brug af komponenter, da det separerer koden og dermed giver en mere modulær og overskuelig kodebase.
Ved at opdele appen i mindre dele kan man fokusere på specifik funktionalitet, hvilket gør koden lettere at fejlsøge, teste og vedligeholde.
Ved at opdele koden i komponenter giver det en god genanvendelighed samt skaber en konsistent brugergrænseflade på tværs af appen.

Database

Backenden benytter en MySQL database.

Selve databasen og tabellerne i denne, er genereret med de migrations der er lavet i Symfony. Opsætningen er er lavet for at give bedst muligt mening i forhold til projektets formål.

Hvis løsningen eventuelt havde været lavet til salg hos flere virksomheder, kunne det give god mening at udvide databasen til også at indeholde hvilke virksomheder, rummene med sensorer var placeret i, samt hvilke land / by virksomheden befinder sig i. Med den givne tid til dette projekt, har jeg valgt at forholde mig til én enkelt virksomhed.



Versionsstyring

Projektet benytter GitHub som versions styringsteknologi, da det er en pålidelig og effektiv måde at administrere og opretholde en velorganiseret og pålidelig kodehistorik. GitHub's funktioner til versionshistorik og branch håndtering gør det let at spore og administrere ændringer i koden over tid.

Samlet set er GitHub en pålidelig og effektiv teknologi til versionsstyring, der hjælper med at opretholde en velorganiseret og pålidelig kodehistorik og på sigt giver kunden mulighed for at automatisere bygningsprocessen og frigivelse cyklussen (CI /CD).

Kravspecifikation

For at give overblik over kravspecifikationer, har jeg opdelt disse i mine tre kategorier, Embedded, Web API og App.

Embedded

Den embeddede del af projektet, skal sende sensor data, i dette tilfælde vil det være temperatur og fugtighed, via websockets til et web API.

ID	Krav	Prioritet	Type
E1	Opsætning af ESP8266 og DHT11	Høj	Funktionelt
E2	Mål temperatur og fugtighed fra DHT11 modul	Høj	Funktionelt
E3	Opret forbindelse til Wifi	Høj	Funktionel
E4	Send temperatur og fugtighed til URL som parametre	Høj	Funktionel

Web API

Web API delen skal modtage data via HTTP request, gemme disse i data i en relationel database og videregive dataen til react-native app ved forespørgsel.

Fra web API skal det være muligt at administrere enheder, enhedstyper, placeringer og typer af målinger.

ID	Krav	Prioritet	Type
W5	Database opsætning med tabel relationer.	Høj	Funktionelt
W6	GET, POST, PUT, DELETE funktionalitet af placeringer.	Høj	Funktionelt
W7	GET, POST, PUT, DELETE funktionalitet af enhedstyper.	Høj	Funktionelt
W8	GET, POST, PUT, DELETE funktionalitet af målingstyper.	Høj	Funktionelt
W9	GET, POST, PUT, DELETE funktionalitet af enheder.	Høj	Funktionelt
W10	GET, POST, PUT, DELETE funktionalitet af målinger.	Høj	Funktionelt

App

App delen skal kunne modtage data fra web API ved forespørgsel, vise denne data til brugeren i form af listevisning i et overskueligt UI, hvor sensor data opdateres i realtid.

ID	Krav	Prioritet	Type
A11	Modtag data fra web API i form a JSON.	Høj	Funktionel
A12	Listevisning af sidste 20 målinger pr. placering.	Mellem	Funktionel
A13	Oprettelse, redigering, sletning af placeringer.	Høj	Funktionel
A14	Oprettelse, redigering, sletning af enhedstyper.	Høj	Funktionel
A15	Oprettelse, redigering, sletning af målingstyper.	Høj	Funktionel
A16	Oprettelse, redigering, sletning af enheder.	Høj	Funktionel

Accepttest oversigt

For at bevare overblik over accepttest, har jeg opdelt disse, som gjort i kravspecifikationer, Embedded, Web API og App.

Embedded

Oversigt over test krav til den embeddede udviklingsproces.

ID	Acceptkrav	Kritisk	Godkendt	Afvist
E1	Efter opsættelse af ESP8266 skal det være muligt at få vist serial test data i Arduino IDE Serial Monitor.	Ja	Ja	
E2	Efter tilføjelse af DHT11 skal det være muligt at se temperatur samt fugtmåling i Serial Monitor.	Ja	Ja	
E3	Opret forbindelse til wifi via ESP8266 websocket library, her skal der opsættes en sigende fejlbesked hvis forbindelse ikke bliver oprettet, eller forbindelse mistes.	Ja	Ja	
E4	Når wifi forbindelse er oprettet, skal det være muligt at kalde en URL med parametre, der skal som feedback returneres HTTP-Response 200.	Ja	Ja	

Web API

Oversigt over testkrav samt status af Web API.

ID	Acceptkrav	Kritisk	Godkendt	Afvist
W5	Det skal være muligt at indsætte test data i tabeller med relationel forbindelse, det ikke skal være muligt at indsætte foreign key id's som ikke eksisterer.	Ja	Ja	
W6	Fra Postman skal det være muligt at foretage POST, GET, PUT og DELETE af placeringer.	Ja	Ja	
W7	Fra Postman skal det være muligt at foretage POST, GET, PUT og DELETE af enhedstyper.	Ja	Ja	
W8	Fra Postman skal det være muligt at foretage POST, GET, PUT og DELETE af målingstyper.	Ja	Ja	
W9	Fra Postman skal det være muligt at foretage POST, GET, PUT og DELETE af enheder.	Ja	Ja	
W10	Fra Postman skal det være muligt at foretage POST, GET, PUT og DELETE af målinger.	Ja	Ja	

App

Oversigt over testkrav samt status af cross-platform React-Native App.

ID	Acceptkrav	Kritisk	Godkendt	Afvist
A11	Der skal foretages GET kald til web API, ved modtagelse skal der være læsbart JSON data.	Ja	Ja	
A12	Det skal være muligt at se en liste af seneste 20 sensor målinger for hver specifik placering, som har aktive sensorer.	Ja	Ja	
A13	Det skal være muligt at oprette, redigere og slette placeringer fra app.	Ja	Ja	
A14	Det skal være muligt at oprette, redigere og slette enhedstyper fra app.	Ja	Ja	
A15	Det skal være muligt at oprette, redigere og slette målingstyper app.	Ja	Ja	
A16	Det skal være muligt at oprette, redigere og slette enheder fra app.	Ja	Ja	

Opsætning

For at opsætte produktet, er der en række kommandoer som skal udføres for at opsætte produktet med korrekt konfigurationer, dette anbefales gjort af en systemadministrator, da de følgende vejledninger vil foregå i en Powershell terminal, samt kræver at systemet har opsat nødvendige installationer.
De nødvendige installationer som kræves før opsætning af projektet er følgende:

- Git (https://git-scm.com/download/win)
- Docker (https://www.docker.com/products/docker-desktop/)
- Scoop (https://scoop.sh/)
- Composer (https://getcomposer.org/)
- PHP (https://www.php.net/manual/en/install.windows.php)

Herefter følges nedenstående vejledninger i Powershell terminal med administrator privilegier:

```
# Hent git repository
git clone https://github.com/MichaelAggerholm/Interdisciplinary_Project.git

# Naviger ind i mappen "Interdisciplinary_Project\measurementApi"
cd .\Interdisciplinary_Project\measurementApi

# Build docker container
docker-compose build

# Kør docker container
docker-compose up -d

# Installer Symfony-cli ved brug af scoop
scoop install symfony-cli

# Installer projekt dependencies med composer
composer install

# Start Symfony serveren
symfony server:start
```

Ved dette punkt, kører nu første del af projektet, som udgør projektets backend.

Næste skridt er at starte selve mobil applikationen op, denne del kræver også nogle system installationer før opsætning kan foretages:

- Node.js (https://nodejs.org/en/download/)
- Expo-cli (https://docs.expo.dev/workflow/expo-cli/)
- Npm (https://docs.npmjs.com/downloading-and-installing-node-js-and-npm)

Følg nu følgende vejledninger:

```
# Åben ny powershell vindue og gå til rodmappen “Interdisciplinary_Project”
cd .\Desktop\Interdisciplinary_Project

# Naviger ind i mappen “measurementApp”
cd .\measurementApp

# Installer dependencies med npm
npm install

# Start app med npx expo
npx expo start

# Vælg herefter enhed, “a” for Android, ”i” for IOS simulator.
a
```

Kørsel foretages nu på simulator, her anbefales det at en android enhed tilsluttes via usb, eller at en simulator enhed i forvejen er installeret på systemet (Se bilag: Opstartet app på Android enhed, s. 25)

Efterfølgende skal der tilsluttes en målingsenhed som sender data til backend api, i mappen “Interdisciplinary_Project”, er kodebasen til opsætning af en ESP8266 enhed med et DHT11 modul, i mappen “esp8266_dht11”.

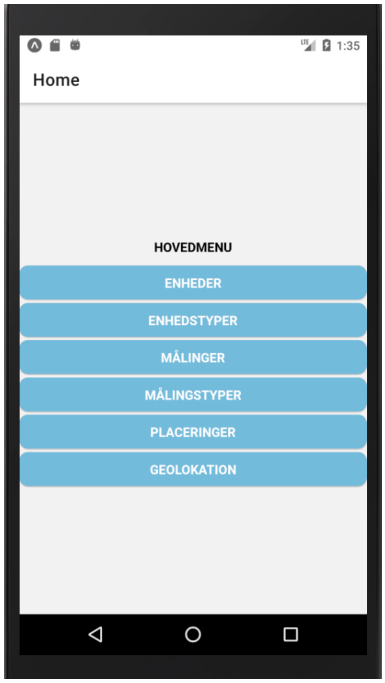
Nu kan brugervejledning tages i brug, til dokumenteret vejledning af applikationen samt håndtering af målinger, enheder, placeringer, enhedstyper og målingstyper, god fornøjelse.

Brugervejledning

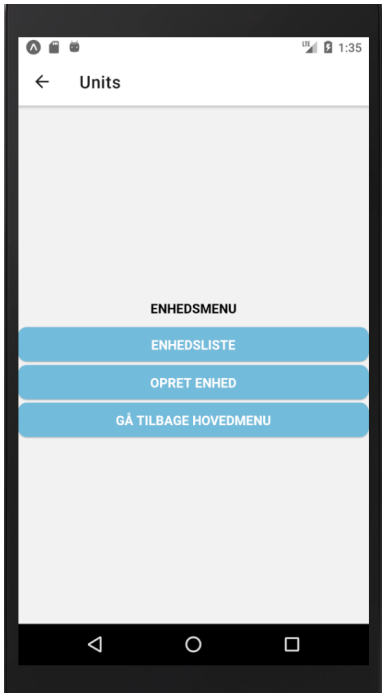
For brug af applikationen kan det være en hjælp at have en målrettet guide som beskriver hver funktionalitet i detaljerede vejledninger, følg med herunder for brug af applikationen samt funktionaliteter.

I denne vejledning er der taget udgangspunkt i en Android enhed, med brug og funktionalitet af enheder og målinger, det samme vil gøre sig gældende for funktionalitet af enhedstyper, placeringer og målingstyper.

1. Ved opstart af applikationen bliver brugeren mødt af en hovedmenu:

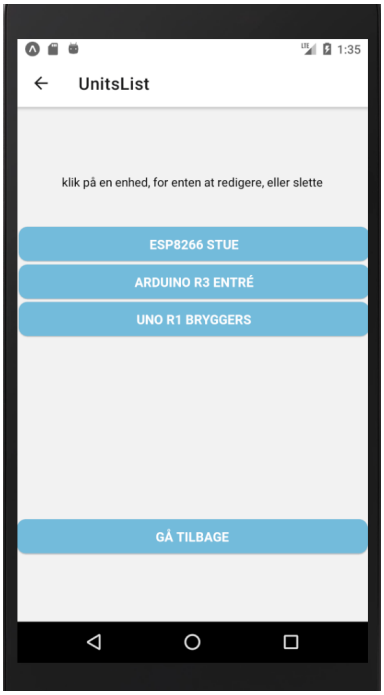


2. Her vælges punktet “Enheder”

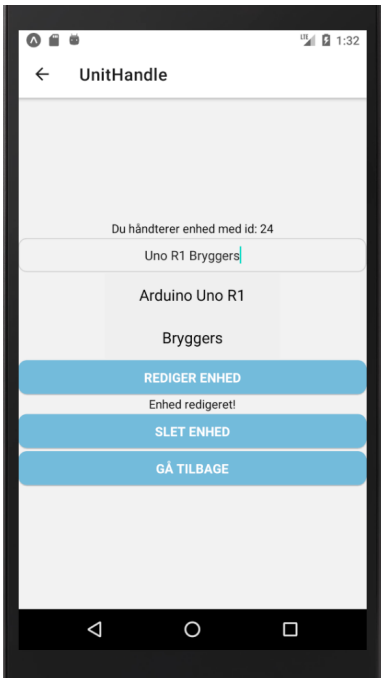


   a. For at oprette en ny enhed i systemet, vælges menupunktet “Opret Enhed”
      i. Udfyld enhedens navn.
      ii. Udfyld enhedens type.
      iii. Udfyld enhedens placering.
         1. Hvis ikke ønskede type eller placering er tilgængelig, oprettes disse under henholdsvis menupunktet Enhedstyper eller Placeringer.

b. For at få en liste af nuværende enheder, klik på menupunktet “Enhedsliste”

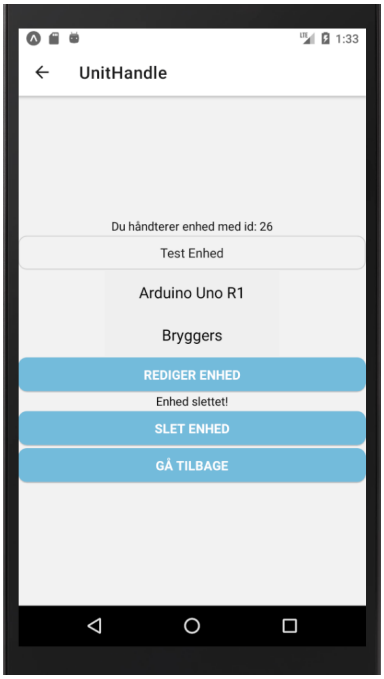


c. Under enhedslisten, kan der klikkes på en enhed, dette vil tage brugeren til enhedshåndteringen, her er det muligt at redigere enheden, eller slette enheden.

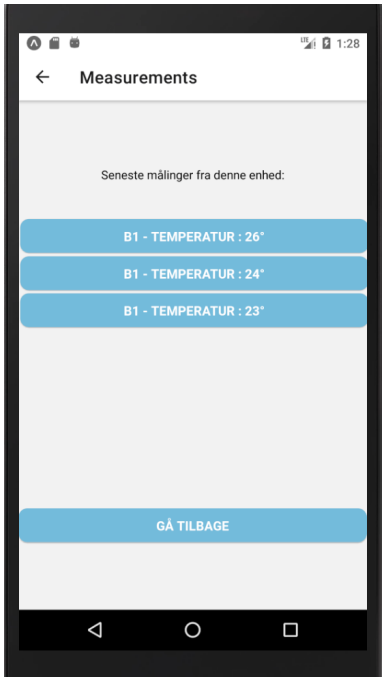


i. For at redigere enheden, udfyld nyt / tilpasset enhedsnavn, placering og enhedstype hvis ikke dette skal forblive det samme. Tryk derefter på menu knappen “Rediger Enhed”, en besked vil komme frem med bekræftelse på redigering, eller fejl hvis ikke enheden blev redigeret.

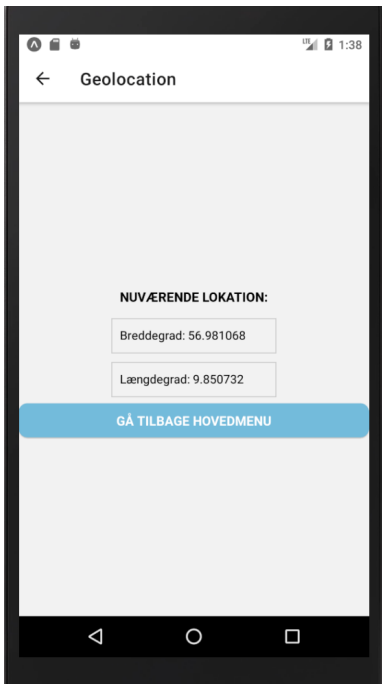
ii. For sletning af enhed, tryk på menu knappen “Slet Enhed” En besked vil komme frem med bekræftelse på sletning, eller fejl hvis ikke enheden blev slettet.



3. For at få et overblik over målinger for en enhed, klik på menupunktet “Målinger”, naviger herefter til punktet “Målingsliste”, her vil brugeren blive mødt af aktive enheder, ved klik på en enhed, vil brugeren få vist en liste af enhedens seneste 20 målinger.



Som ekstra opgave, skulle der hentes data fra ét modul i telefonen, her kan man under menupunktet se enhedens geolokation, det virker dog ikke på emulator ved mindre man har en tredjeparts token tilknyttet emulatoren, det har jeg her, på min hjemme emulator:



Konklusion

Slutproduktet står færdigt, de tre udviklede platforme lever uafhængigt af hinanden, det giver en enorm frihed samt skalerbarhed ift. den pågældende bruger, da projektet kan benyttes i helhed, eller separat i platforme, der kan nemt tilføjet platforme, målinger og data uanset hvorfra, eller hvilket framework det består fra.
Der kan også udvides eller tilpasses i én platform, uden nødvendig behov for tilpasning i de andre platforme.

Meningen med slutproduktet skulle lige præcis være dette, at det ikke er fastlåst til en bestemt enhed, eller måling, men giver brugeren frihed til at tilknytte stort set lige hvad der gør sig gældende.

Desværre har projektet den ulempe at det er en stejl opsætning, for at få systemet til at køre, som beskrevet i opsætningen skal brugeren nærmest agere systemadministrator for at starte systemet, udover det er der en del krav til systemets installationer for overhovedet at kunne benytte opsætningsguiden.
Det vil gavne projektet hvis der blev sat tid af til at udvikle et ordentligt miljø for koden, samt nogle automatiseringsprocesser.
Det kan blandt andet gøres ved at lave et docker miljø for henholdsvis app samt api, hvor hver docker container opsættes specifikt til formålet, gemmes i sit eget selvstændige image, og derefter let kan hentes ned og startes med den allerede opsatte konfiguration.

Grundet den korte udviklingsfase af mobil applikationen, bør denne på sigt gøres endnu mere modulær ved at opdele yderligere af koden i komponenter, den manglende tid har gjort kodebasen delvist opdelt i komponenter, hvor nogle filer desværre stadig står med alt for meget funktionalitet, som bør opdeles.

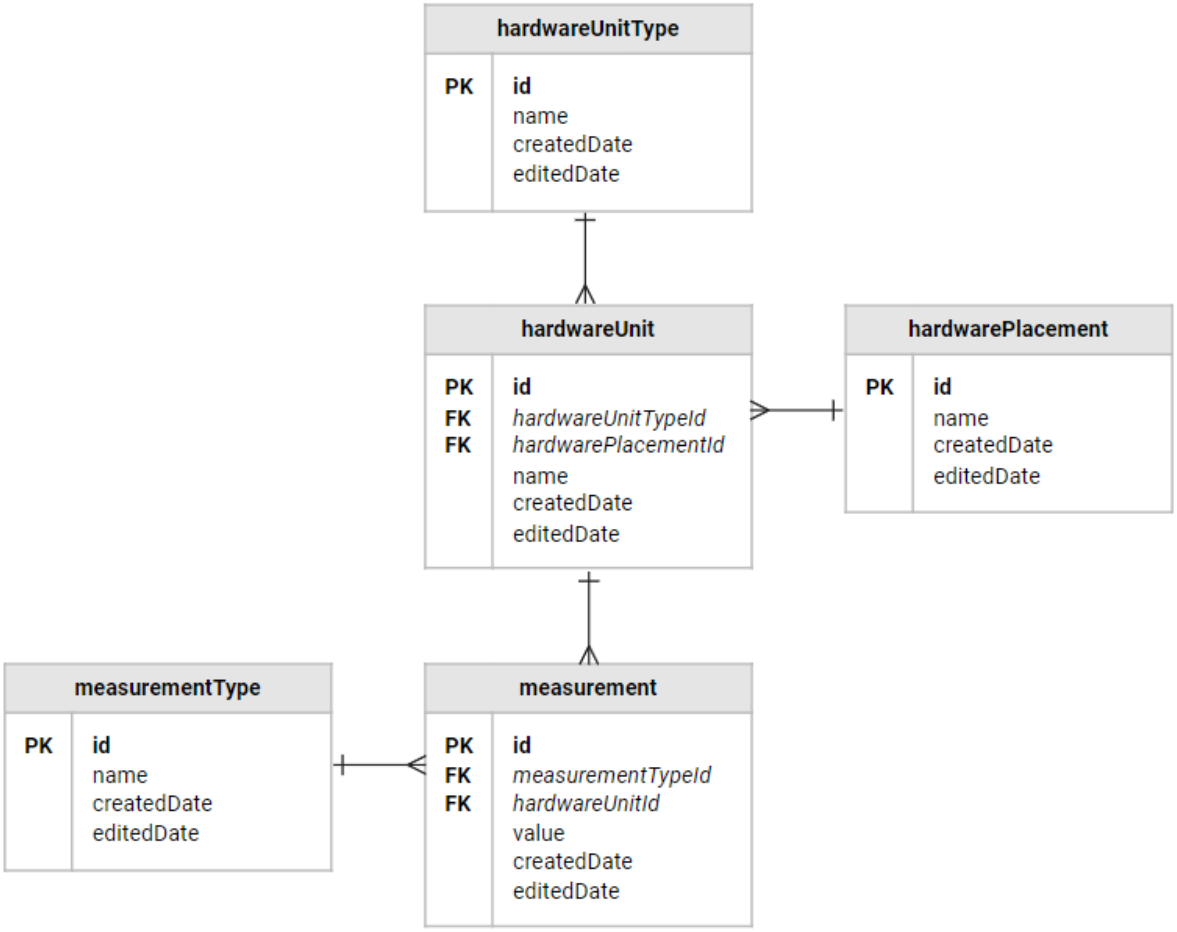
Som forbedring af projektet anbefales det at der implementeret tests af koden, i form af unit tests og en automatiseringsprocess af disse, det kan blandt andet være ved brug af Jenkins som hjælper med kørsel af diverse tests ved ændringer i koden, denne del kan let implementeres da projektet i forvejen gør brug af versionsstyring, som stort set er et af de eneste nødvendigheder for at kunne gøre brug af Jenkins automatiseringsproces.

Sidst skal det nævnes, for at afværge enhver undren, at grunden til at der er tilføjet Geolokation i App'en, uden en dokumenteret beskrivelse, er udelukkende for at overholde målpinde i App 3 faget.

Bilag

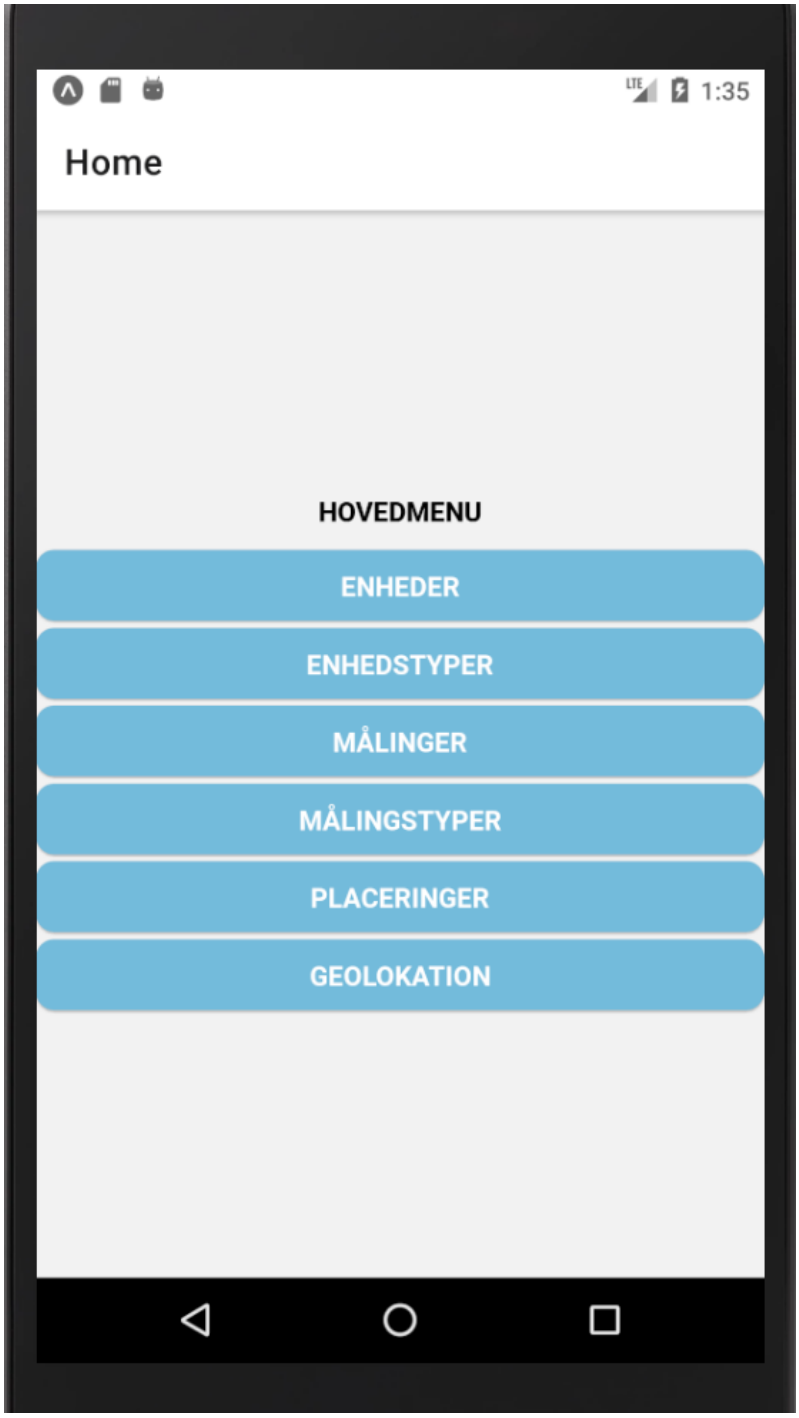
Database diagram

[Link til database diagram]



Opstartet app på Android enhed

App menuen som brugeren bliver mødt af ved app opstart.



Geolokation

Enhedens geolokation, fra rigtig emulator med google geolokation token tilføjet: