SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

HERO-JÄRJESTELMÄN KEHITYS JA TUOTTEISTAMINEN

TEKIJÄ: Reino Palviainen

SAVONIA-AMMATTIKORKEAKOULU OPINNÄYTETYÖ

Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä Reino Palviainen	
Työn nimi Hero-sovelluksen kehitys ja tuotteistaminen	
Päiväys 19 Joulukuuta 2021	Sivumäärä/Liitteet 23
Toimeksiantaja/Yhteistyökumppani(t) Preventos Informatics Oy	

Tiivistelmä

Opinnäytetyön tarkoituksena oli kehittää tilaajan Preventos Informaticsin Hero-järjestelmää lisäämällä siihen osio jossa käyttäjät voivat tarkastella heille kuuluvia mittapisteitä sekä valita mittaustuloksia ajanjaksoittain. Tarpeena oli myös erillinen sovellus jolla luoda ja muokata käyttäjäkonfiguraatioita.

Työ alkoi Heroon perehtymällä ja sen toimintaperiaatteiden selvityksellä. Tämän jälkeen alkoi Reactiin tutustuminen, ja tilaajan toiveiden selvitys sen suhteen minkälaisia tekniikoita sekä ratkaisuja tulisi käyttää. Tutkittiin mikä olisi paras tapa saada halutut ominaisuudet graafeihin, jonka täytyi tukea useaa yhtäaikaista datajoukkoa että saataisiin mittaukset visualisoitua käyttäjälle.

Lopputulos oli kaksiosainen: Ensimmäisessä osassa kehittiin Preventos Informaticsin web-sovelluksen toiminnallisuutta laajentaen sitä tukemaan tiedonkeruujärjestelmän mitatun ja analysoidun datan esittämiseen aikasarjoina. Tämän osion toiminnallisuus oli React-pohjainen, ja perustui osin jo olemassaolevaan vanhempaan toiminnallisuuteen joka piti siirtää Reactiin perustuvaan ratkaisuun.

Toisen osan tarkoitus oli helpottaa admineiden työtä rakentamalla sovellus jossa voidaan helposti luoda ja muokata konfiguraatioita jotka sisältävät pääsovelluksen tarvitsemat tiedot asiakkaasta. Sovellus rakennettiin .NET kehitysalustan ja C#-kielen avulla. Tähän osaan ei ollut olemassa aiempaa ratkaisua, ja se rakennettiin tyhjältä pöydältä.

Avainsanat
React, JavaScript, .NET, .NET Core, C#, Ohjelmistokehitys.

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author Reino Palviainen	
Title of Thesis Development and productization of Hero-application.	
Date 19 December 2021	Pages/Appendices 23
Client Organisation /Partners Preventos Informatics	

Abstract

The aim of this thesis was to further develop the client's Hero-application by implementing a section to it where users can view the measuring points they have access to, and select measurement data based on selected time series. The client also required a separate application for creating and editing user configurations.

First, the Hero-application and its basic functionalities were examined. The next step was to learn how to use React. Then, discussions with the client were carried out in order to find out the requirements and preferences concerning the implementation. Research was made to clarify the best way to implement the wanted properties to charts, that had to support multiple. This way the measurements could be visualized for the user.

The final result consists of two parts: In the first part Preventos Informatics's web application was further developed by implementing a section to it that supported the visualization of measured and analyzed data in time series. This part was built on React, and partly used existing functionalities from previous versions, that needed to be modified for the React-based solution.
In the second part the aim was to ease the workload of admins by building an application that could easily create and edit user configurations that hold information about the user required by the main application. This application was built on .NET development platform with C#. There were no previous solutions for this part, so it all had to be built from scratch.

Keywords
React, JavaScript, .NET, .NET Core, C#, Software development.

ESIPUHE

Opinnäytetyö kehitti osaamistani Reactin parissa erittäin paljon, ja myös .NET osaaminen karttui. Haluan kiittää Preventos Informaticsin asiantuntijaa joka osasi hyvin kertoa ja rajata tarvittavat toimenpiteet, sekä avustaa ongelmien aikana.

Kuopiossa 19 joulukuuta 2021

Reino Palviainen

SISÄLTÖ

KUVALUETTELO

1 JOHDANTO

Tiedonkeruujärjestelmien tuottama datamäärä kasvaa jatkuvasti, ja tarvitaan parempia tapoja suodattaa, analysoida ja esittää tätä dataa. Tästä seuraa se että yrityksien on pysyttävä ajan hermolla ja kehittettävä omia järjestelmiään, sekä mietittävä mitä tulevaisuudessa voi tapahtua, ja kuinka se voi vaikuttaa jo olemassaoleviin ja kehitteillä oleviin järjestelmiin ja sovelluksiin. Tämä työ keskittyi juuri tähän, kehittäen Hero-sovellusta.

Tilaajan toiveena oli kehittää heidän websovellusta siten että serveriltä haettu data voitaisiin esittää graafina, sekä karttapohjaisessa UI:ssä. Käyttäjä syöttää halutun aikavälin, valitsee halutun mittakohteen jonka jälkeen haetaan data joka piirtyy graafiin. Kartalle piirtyy jokainen mittauspiste mihin käyttäjällä on oikeus, ja kartalta klikattuna mittapisteestä näytetään kuva, yleisiä perustietoja sekä viimeisimmät mittauslukemat sensorikohtaisesti. Tämä osa työtä oli React-pohjainen.

Haasteena tässä oli se että kyettäisiin näyttämään luotettavasti eri aikaintervallilla mitattuja lukemia eri mittareilta samaan aikaan samassa graafissa, jossa onnistuttiinkin.

Tilaaja toivoi myös toista, pienempää sovellusta jolla voitaisiin luoda JSON-objekteja käyttäjien konfigurointia varten, ja tämä toteutettiin .NET pohjalle. Sovelluksessa haasteena oli tarve kyetä luomaan n-kappaletta uusia osioita perusobjektin sisälle, sekä muokkaamaan olemassaolevia osioita tarvittaessa.

1.1 Lyhenteet

API = Application Programming Interface, Ohjelmointirajapinta.

Frontend = Järjestelmän asiakkaan päädyn toimintoja kuvaava yleistermi.

Backend = Järjestelmän taustatoimintoja kuvaava yleistermi.

IoT = Internet of Things, Esineiden Internet.

JSON = JavaScript Object Notation.

MVC = Model-View-Controller.

UI = User Interface, käyttöliittymä.

CSS = Cascading Style Sheets, ulkoasun muokkauksen tyylikieli.

1.2 Työn tilaaja

Työn tilaaja Preventos Informatics on keskittynyt vesi- ja muiden nestevirtojen toimintaa seuraavien järjestelmien hallintaan ja valvontaan, sekä tarjoaa Vesikortti-koulutuksia. Yritys perustettiin kolmen vesi- ja informaatioteknologian osaajan toimesta 2018 (Preventos Informatics Oy, 2018).

2 IOT JA TIEDONKERUUJÄRJESTELMÄT

Nykymaailmassa data on tärkein valuutta, sitä kerätään kaikkialla ihmisten ostostaipumuksista ilmanpaineisiin, internetsivujen kävijämääristä asuntojen vedenkäyttöön ja tuhansista muista kohteista. Tiedon avulla voidaan kehittää tehokkaampia algoritmeja joilla houkutella uusia asiakkaita verkkosivuille, helpottaa vikojen havaitsemista huomaamalla poikkeavat lukemat muutoin vakaasta polttoainekulutuksen mittauskäyrästä, havaita erityisesti rasittuvat kohdat älykkäiden metsäkoneiden rungosta, ja vain mielikuvitus on rajana mihin tätä dataa voidaan käyttää.

2.1 IoT

IoT, eli Internet of Things, tarkoittaa käytännössä pieniä laitteita jotka keräävät jatkuvasti dataa (esim. sademäärä tai ilmankosteus) ja lähettävät sen langattomasti (tai langallisena joissain harvoissa tapauksissa) eteenpäin. Nämä laitteet voivat mitata mitä vain, ja ne tuottavat suuria määriä dataa joka päivä maailmanlaajuisesti (Wikipedia, 2021).

2.2 Tiedonkeruujärjestelmä

Tiedonkeruujärjestelmällä tarkoitetaan esimerkiksi vesiputkien veden virtausnopeutta seuraavien mittareiden datan keräystä luotettavasti yhteen paikkaan, jotta voitaisiin hyödyntää tätä dataa esim. vikojen nopeassa tunnistuksessa. Eri järjestelmiä käytetään erilaisissa paikoissa, sillä järjestelmä joka on luotu keräämään ja hyödyntämään dataa vedenkulutuksesta tai verkkosivujen kävijämäärästä, ei toimi mittareiden kanssa jotka keräävät dataa tuulivoiman tuottamasta energiasta.

3 HERO-SOVELLUKSEN KEHITYS

Jokaisen sovelluksen täytyy pysyä mukana nykyajassa sekä vastata kehittyviin tarpeisiin asiakasrintamalla. Kehitys on jatkuvaa, sillä uusien teknologioiden tai tarpeiden myötä tarvitaan aina uusia ominaisuuksia, tai täytyy päivittää olemassaolevia.

3.1 Kehityksen tarve

Tilaajan toive oli lisätä sovellukseen osio jossa käyttäjä näkee kartan päälle piirrettynä omat mittapisteet, ja jossa voidaan visualisoida valitun ajanjakson mittausdata graafissa, sekä yksittäisenä graafina että useamman mittapisteet lukemat yhtäaikaa.

3.2 Käytetyt kielet ja kirjastot

Tässä osiossa mainitaan oleellisimmat kielet ja kirjastot. Kielillä tarkoitaan ohjelmointikieliä, esimerkiksi Javascript tai C#, ja kirjastoilla tarkoitetaan ohjelmistokirjastoja/luokkakirjastoja jotka sisältävät luokkia, aliohjelmia ja vastaavia joita voidaan käyttää modulaarisesti.

Ohjelmointikielellä tehdään lähdekoodia, josta tuotetaan binääriä kääntäjällä/tulkilla jota tietokoneet ymmärtävät (Staff, 2021). Täten voidaan ohjeistaa tietokoneita toimimaan halutulla tavalla, ja tähän prosessiin perustuu kaikki tietotekniikka.

Kirjastot helpottavat sovelluskehitystä kun tarvitut ominaisuudet on helposti saatavilla pakettimuodossa, eikä ohjelmoijan tarvitse kehittää omia ratkaisuja ongelmaan tai rakentaa uutta toiminnallisuutta sovellukseen, mikäli se on jo olemassa kirjastona. Tässä säästetään paljon aikaa ja vaivannäköä, jonka seurauksena ohjelmistoala on kehittynyt huimasti lyhyessä ajassa: kerran kehitetty uusi sovellus/ohjelma/ominaisuus/luokka on usein rakennettu vapaasti käytettäväksi kirjastoksi, jonka päälle voi lähteä kehittämään uutta toimintoa tai yhdistelemään luokkia ja toimintoja uudella tavalla (Wikipedia, Kirjasto (tietotekniikka), 2021).

3.2.1 JavaScript

Dynaaminen ohjelmointikieli jota käytetään mm. Websivujen rikastamiseen interaktiivisuuden ja ulkoasun suhteen, ja esimerkiksi kolmannen osapuolen API-palveluiden integrointiin omaan websovellukseen. Se on helppo oppia, ja kokemuksen myötä lisääntyvät vaihtoehdot mihin kaikkeen JavaScriptiä voi käyttää: yksinkertaisen pop-up modaalin lisäämisestä verkkosivulle kokonaisen 3D-pelin rakentamiseen (Mozilla, 2021).

3.2.2 React

React on JavaScript kirjasto, käytetty käyttöliittymien rakennuksessa (frontend). Koodi jaetaan komponentteihin, joita yhdistellään saaden lopputulokseksi haluttu kokonaisuus. React on myös helppo yhdistää mihin tahansa ”backend”-tekniikkaan (Facebook, 2021).

3.2.3 Bootstrap

Reaktiivisen ja ammattimaisen käyttöliittymän luomisessa Bootstrap nopeuttaa prosessia valtavasti. Erittäin suosittu open-source CSS kirjasto front-end kehityksessä. Sai alkunsa vuonna 2011 kahden Twitterillä työskennelleen ohjelmistokehittäjän, Mark Otto ja Jacob Thornton, halusta luoda yhteinäisyyttä erillisten sisäisten työkalujen välille, vähentäen esimerkiksi sivustojen huoltamisen työkuormaa kun käytössä olisi yksi kirjasto useiden sijaan (Bootstrap, n.d.).

3.2.4 Aiemmat graafinpiirtokirjastot

Graafia lähdettiin toteuttamaan aluperin Recharts-kirjastolla, mutta ajan myötä tultiin siihen tulokseen että se ei ole tarpeeksi joustava tämän opinnäytetyön vaatimuksiin (Recharts, n.d.).

3.2.5 Chart.js

Kun päätös Recharts:sta luopumiseen oli tehty, alkoi vaihtoehtojen tutkiminen. Chart.js, open source-JavaScript kirjasto, oli se mihin päädyttiin. Chart.js sisältää paljon eri vaihtoehtoja ulkoasun suhteen, on responsiivinen oletuksena ja sisälsi vastauksen erääseen haasteeseen josta lisää myöhemmin (Chartjs, n.d.).

3.2.6 Leaflet

Leaflet on open-source JavaScript kirjasto interaktiivisten karttojen luomiseen, jotka ovat myös oletuksena mobiiliystävällisiä. Kevyt ja monipuolinen kirjasto, joka ei vaadi paljon tehoja toimiakseen. Tätä käytettiin kartan luomiseksi, sekä siihen piirrettävien mittapisteiden visualisointiin. Leafletin tekijä on Vladimir Agafonkin (Agafonkin, n.d.).

3.3 Olemassaolleen sovelluksen muokkaus

Hero-sovellus tarvitsi lisäominaisuuksia, jotka siihen lisättiin tämän opinnäytetyön aikana. Tilaajalla oli alusta asti tiedossa perusominaisuudet joita opinnäytetyössä oli tarkoitus lisätä Heroon. Nämä ominaisuudet kehittyivät hieman yksityiskohtien puolella työn aikana.

3.3.1 Alkuperäinen sovellus

Hero-sovellus oli jo käytössä ennen tämän opinnäytetyön alkamista. Sovellus perustui aiemmin olemassaolleeseen, osin Vue-pohjaiseen sovellukseen jonka pohjimmainen tarkoitus oli toimia asiakasrekisterinä. Tämä sovellus mahdollisti asiakkaiden lisäyksen karttapohjalle, sekä viestinnän asiakkaille joko tekstiviestillä tai sähköpostilla. Heron tarkoitus oli toteuttaa käyttäjäpohjainen kirjautuminen, selkeyttää toimintoja, sekä luoda pohja käyttöliittymälle johon saadaan konfiguraation avulla eri asiakkaille näkymään heille tarkoitetut toiminnot, näkymät, mittapisteet ja mittaustulokset joihin heillä on oikeus. Hero myös poisti tarpeen kustomoida jokaiselle asiakkaalle erikseen omaa sovellusta, kuten jouduttiin tekemään aiemmin.

3.3.2 Erityyppiset käyttäjät

Selvennyksenä täytyy määritellä erityyppiset käyttäjät hieman tarkemmin: On olemassa Preventoksen toimeksiantajia, esimerkiksi vesiosuuskunta, jolla on omat asiakkaansa (esim. taloudet jotka kuuluvat ko. vesiosuuskuntaan). Hero-sovellus oli rakennettu alunperin toimeksiantajien ja heidän asiakkaidensa väliseen viestintään, sekä toimeksiantajan asiakkuuksien hallintaan. Uusi osuus joka lisättiin tämän opinnäytetyön aikana mahdollistaa muuntyyppisten toimeksiantajien lisäämisen Heroon, esimerkiksi kaupunkien, jolloin asiakkuuksien hallinta-ominaisuus ei ole tarpeen, vaan on tarve vain käyttää mittausten visualisointi-ominaisuutta. Tässä opinnäytetyössä puhutaan siis pääasiassa toimeksiantajista joilla ei ole tarvetta käyttää asiakkuuksien hallintaa, koska se osio ei kuulu tähän opinnäytetyöhön.

3.3.3 Uusi osio mittausdatan tarkastelua varten

Tämä osio toteutettiin tässä opinnäytetyössä. Vaatimuksina oli sovelluksen yleiseen ilmeeseen sopiva osio jossa mittapisteistä joihin asiakkaalla on käyttöoikeus voidaan hakea mittalukemia syötetyltä ajanjaksolta sekä tarkastella niitä visuaalisesti. Sisältää myös kartan johon lisätty ”markereita” jotka esittävät mittapisteiden sijainnin kartalla, joita klikkaamalla nähdään viimeisimmät lukemat mittapisteen jokaiselta sensorilta, ajankohta siitä milloin mittaus on tehty sekä kuva mittapisteestä.

3.3.4 Edellinen versio

Uuden osion perustoiminnallisuus oli osin jo olemassa. Aiempi versio oli jokaiselle asiakkaalle erikseen kustomoitu sovellus, joka rakennettiin JavaScriptin, HTML ja CSS avulla. Tästä johtuen sitä ei voitu suoraan tuoda React-pohjaiseen Heroon, vaan siitä voitiin ottaa perustoimintaperiaate sekä yleinen logiikka esim. graafien piirtoon, ja toteuttaa näitä React-pohjaisesti. React käyttää JavaScriptiä, mutta komponenttipohjaisesti.

3.3.5 Haasteet

Suurin haaste oli se että graafikirjaston täytyi tukea useaa datajoukkoa yhtäaikaisesti ja kyetä piirtämään ne siten että x-akselin aikajanalle piirtyvät mittaukset olivat kukin oikealla kohdalla. Koska eri datajoukoissa ei aina löytynyt mittausta samalta ajanhetkeltä, tuotti tämä ongelmia.

Käyttöliittymässä oletetaan että API-kutsulla haetun datan aikaleimat ovat keskenään vertailukelpoisia, eli käytännössä siihen että eri sensoreiden sisäiset kellot ovat riittävän tarkasti samassa ajassa.

Esimerkki toiminnasta yksinkertaistettuna:

On kaksi datajoukkoa, J1 ja J2. Joukko J1 sisälsi mittaukset ajakohdille 1, 3, 5, 7, 9 ja J2 ajankohdat 2, 4, 6, 8, 10. Nämä täytyi saada piirtymään graafiin siten, että x-akseli sisälsi ajankohdat 1-10 ja kukin mittaus oli oikeassa kohdassa aikajanaa, eli piirtyivät järjestyksessä 1, 2, 3, 4 ... 10 ilman että se rikkoisi toimivuutta tai ulkoasua.

Ratkaisuna tähän oli muuttaa kaikki mittadatan aikaleimat samaan formaattiin, joka tässä tapauksessa oli 1.1.1970 klo 00:00:00 ja mittauksen ajakohdan välinen ero millisekunteina (unix-aika) (Mozilla, 2021), sekä löytää oikeanlainen kirjasto joka tässä tapauksessa oli Chart.js.

Testidata oli rajoitettu, eikä sis täysin vastannut oikeaa dataa, ja tästä seurasi pieniä ongelmia kun siirryimme oikean datan testaukseen.

React/JavaScript ei ollut entuudestaan tuttu, sekä sen komponenttipohjainen rakenne aiheutti omat ongelmansa kun taustalla oli enemmän .NET Core ja C# kokemusta, sekä näiden kahden alustan ja kielen erot rakenteissa ja syntakseissa.

4 CONFIG-SOVELLUS

Hero-sovelluksen kehityksen lisäksi tilaaja tarvitsi kevyen sovelluksen jolla olisi helppo luoda, poistaa ja muokata käyttäjäkonfiguraatioita. Tämän sovelluksen ulkoasu on hyvin yksinkertainen, sillä se on vain admineiden käytössä, joten tähän osaan ei panostettu.

4.1 Perustoiminta

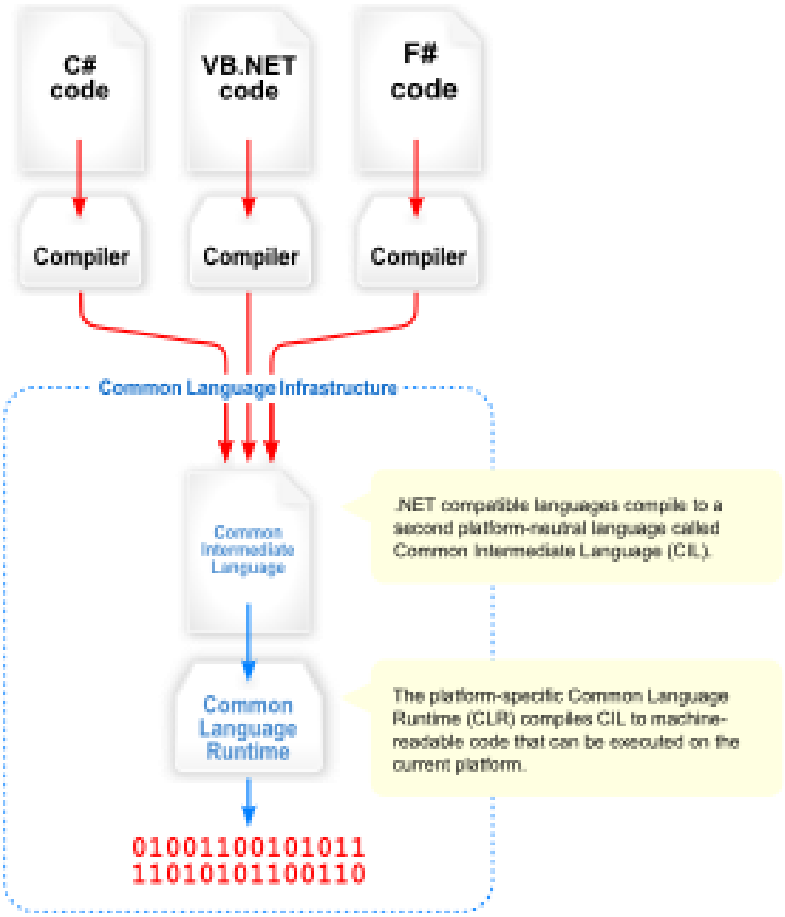
Sovelluksen perustoimintoina ovat käyttäjäkonfiguraatioiden luonti ja muokkaus. Admin voi lisätä, muokata ja poistaa tarpeen tullen JSON objektin osia, jonka pohjalta Hero-sovellus lukee seuraavat tiedot: Käyttäjäkohtaiset tiedot kuten id, pid, tyyppi, mittapisteet, mittapisteiden koordinaatit, mittapisteiden sensorit, aikavyöhyke, mittausintervalli, mittapisteen kuva, mahdolliset lisätiedot, viestiasetukset.

4.2 .NET Core ja C#

Tässä osiossa kerrotaan hieman lisää .NET kehitysalustasta sekä C# kielestä.

4.2.1 .NET Core

.NET on ilmainen vapaan lähdekoodin alusta kehittäjille, joka käyttää Yhteistä kieli-infrastruktuuria (Common Language Infrastructure, kuva 1).



Kuva 1. CLI (Wikipedia, .NET, 2021).

Tämä sovellus tehtiin .NET Corella jonka avulla voi kehittää järjestelmästä riippumattomia sovelluksia, joka tässä tapauksessa tarkoittaa että rakentamasi sovellus toimii yhtä hyvin windows käyttöjärjestelmässä, linuxilla kuin myös macOS järjestelmissä. Kehittäjä voi valita itselleen sopivimman kielen seuraavista vaihtoehdoista: C#, F# tai Visual Basic.

C# on vahvasti tyypitetty ohjelmointikieli.

.NET Core 1.0 julkaisiin kesäkuussa 2016, ja sen perustarkoitus oli helpottaa järjestelmästä riippumattomien sovellusten kehitystä sekä suosia ”basaari”-tyylistä vapaan lähdekoodin kehitysmallia (Wikipedia, .NET, 2021).

4.2.2 MVC

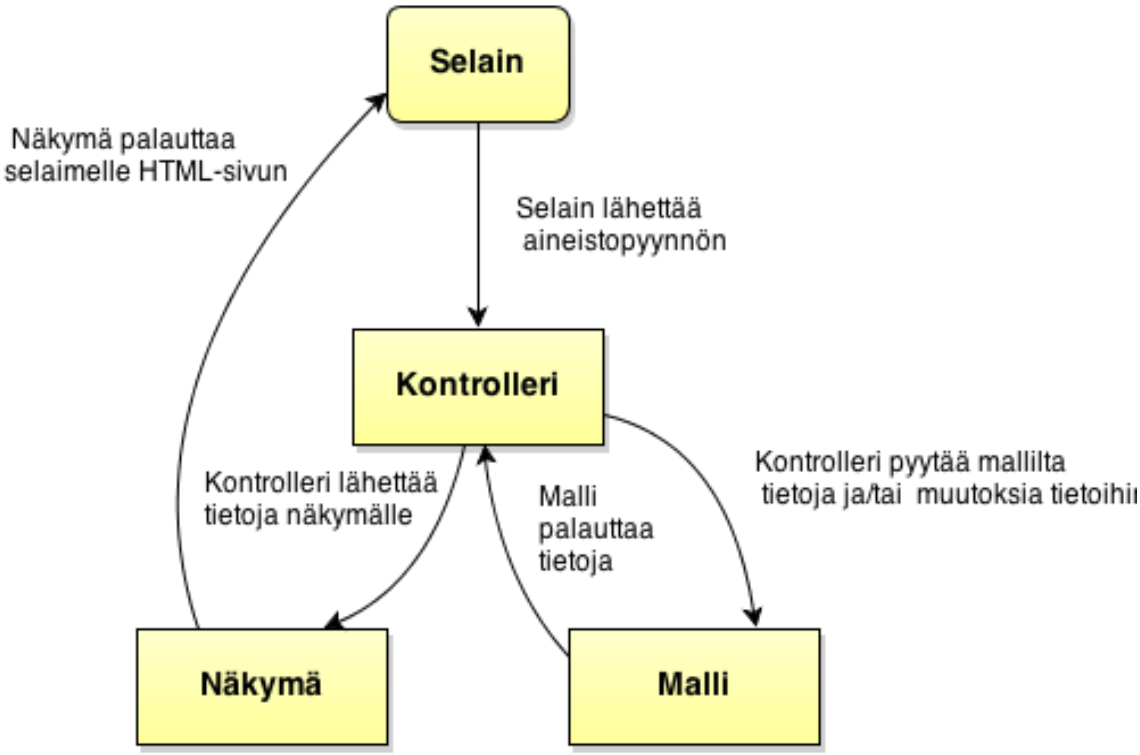
MVC tarkoittaa Model-View-Controller arkkitehtuuria (kuva 2).

Tässä tiedot on jaettu kolmeen osaan:

Model sisältää tiedot jota halutaan näyttää/käyttää, ns. tietokantakuvaus.

View:llä tarkoitetaan näkymää, tässä asetetaan käyttöliittymän ulkoasu ja muut siihen liittyvät.

Controller hoitaa käyttäjän käskyjen käsittelyt, sekä Model- ja View-osien muokkaukset tarpeen tullen.

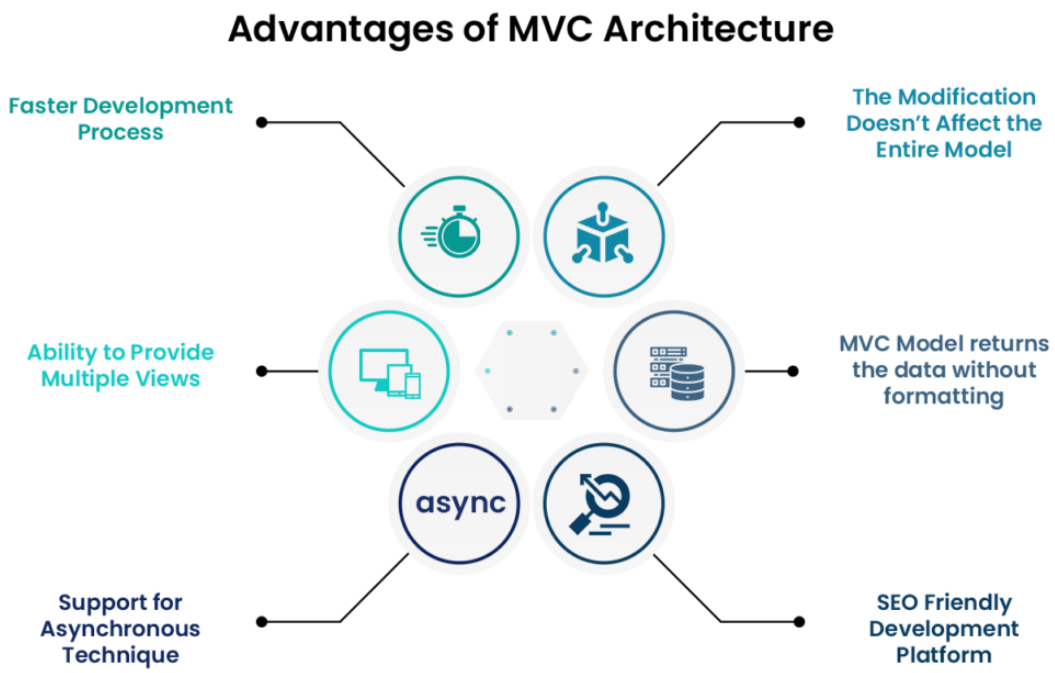


Kuva 2. MVC arkkitehtuurin periaatteet (Consuegra, n.d.).

MVC:n hyödyt ja haitat:

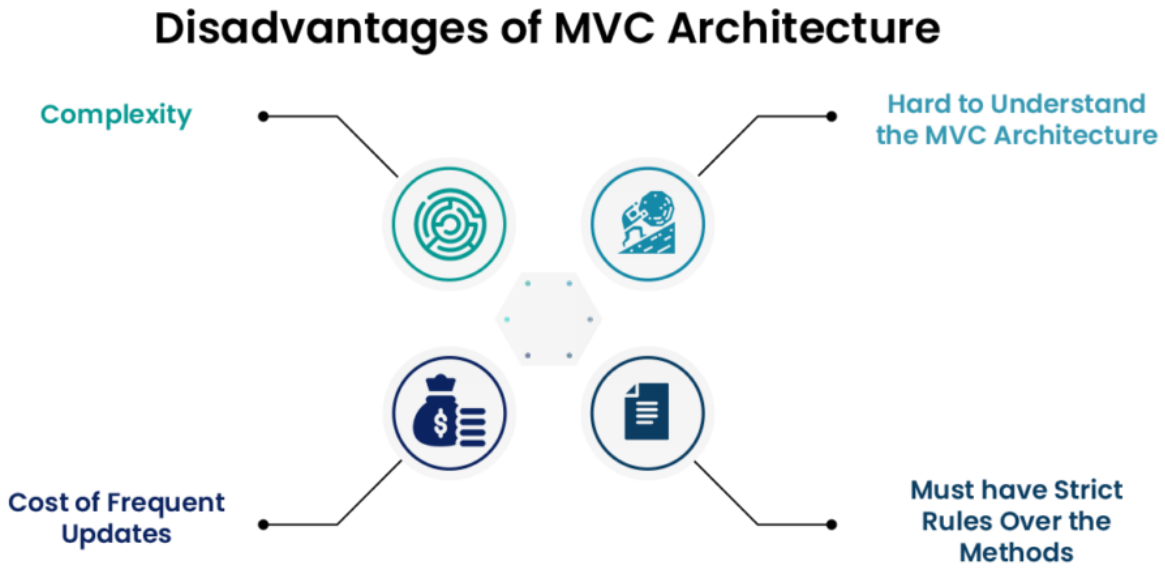
Hyötyinä voidaan mainita seuraavat (kuva 3):

- Samaa mallia (Model) voidaan käyttää monissa näkymissä (View) eri tavoin ilman että alkuperäistä mallia tarvitsee muokata.
- Jokaista osiota (Model, View, Controller) voidaan tietyin rajoittein muokata erikseen ilman että lopputulos vaikuttaa muihin osiin.
- UI:n asynkrointi on helppoa.



Kuva 3. MVC hyödyt (Sharma, n.d.).

Haittoina mainittakoon että kolmeen osaan jakautuva arkkitehtuuri voi vaikeuttaa sisällön ymmärtämistä kehittäjän näkökulmasta, sekä tämä arkkitehtuuri voi aiheuttaa tarpeettoman tiedon hakemista tietokannasta, sillä malli (Model) usein sisältää paljon muutakin tietoa kuin mitä tarvittaisiin näkymässä (View) tai kontrollerissa (Controller) (Kuva 4) .



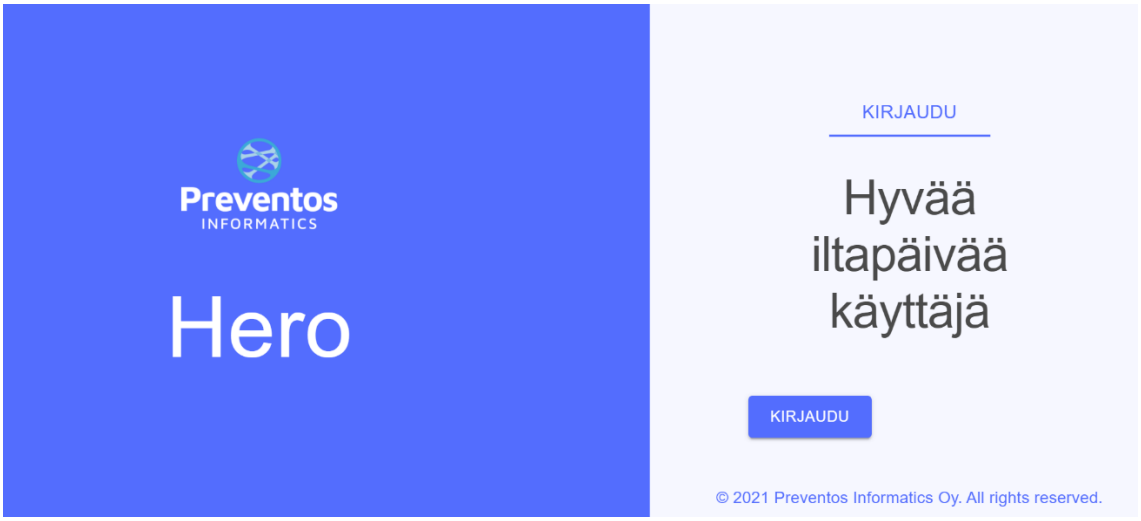
Kuva 4. MVC haitat (Sharma, n.d.).

5 LOPPUTULOS

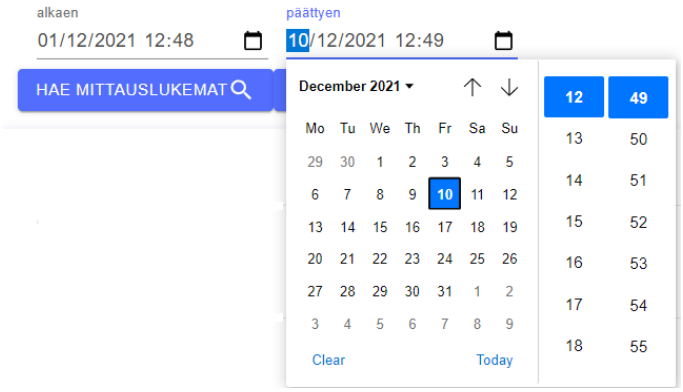
Opinnäytetyön lopputuloksena saatiin tilaajan toivomat kaksi erillistä osiota valmiiksi, ja asiakas oli tyytyväinen tuloksiin. Tässä kohtaa opinnäytetyötä tarkastelemme hieman tarkemmin kummankin osion lopputuloksia.

5.1 Hero

Olemassaolevaan sovellukseen oli tarve luoda osio jossa tapahtuisi mittadatan haku ja visualisointi käyttäjän omien valintojen pohjalta. Käyttäjä valitsee ajanjakson kalenteri-pohjaisesta valikosta (kuva 6) sekä haluamansa sensorit ja mittapisteet (kuva 7), jonka jälkeen lähetetään API kutsu serverille joka palauttaa kaikki valittujen mittapisteiden mittaukset valitulle ajanjaksolle ja nämä tiedot tallennetaan selaimen välimuistiin. Tämä siksi, että käyttäjä voi seuraavaksi haluta katsoa samasta mittapisteestä jonkin toisen, aiemmin valitsematta jätetyn sensorin mittaukset samalle ajanjaksolle, niin ei tarvitse tehdä uutta hakua tietokantaan asti erikseen ennen kuin paikallinen tieto on vanhentunut, tai halutaan hakea dataa jota ei löydy paikallisesta muistista.



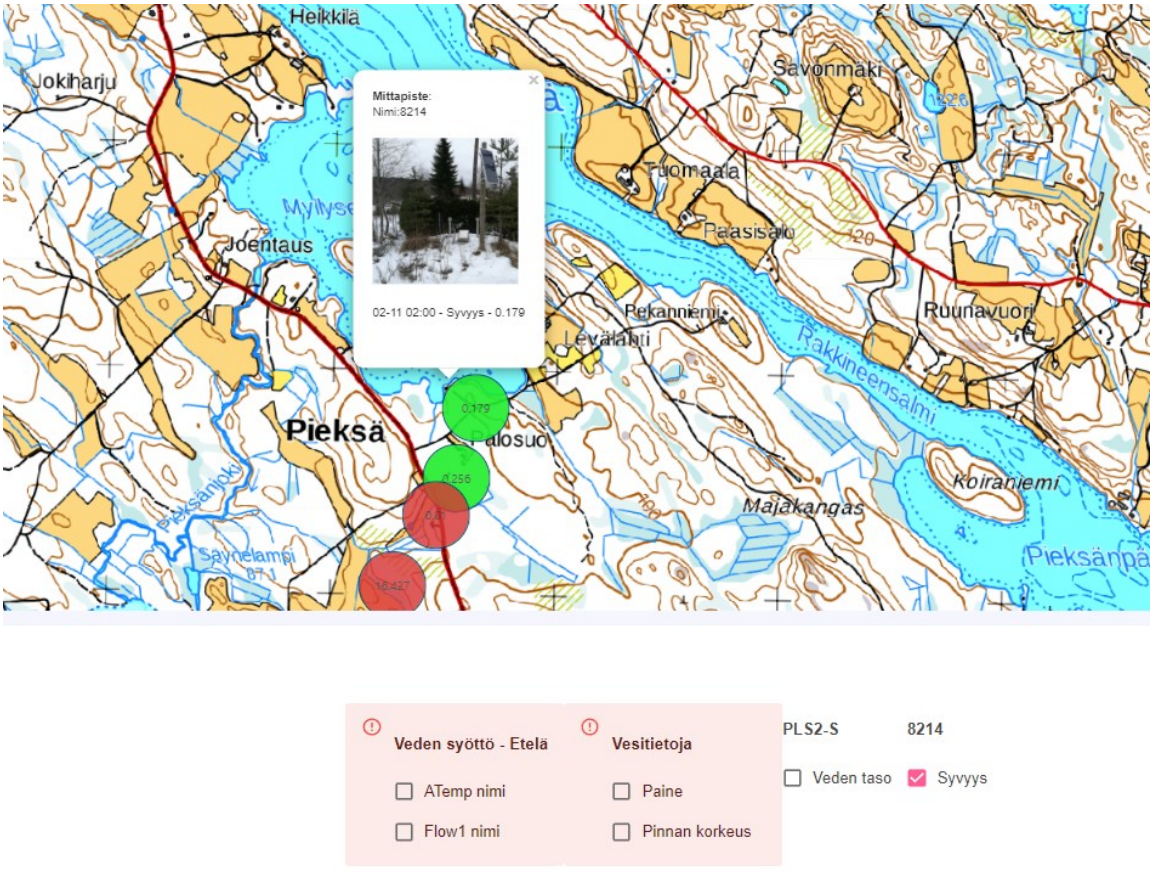
Kuva 5. Hero-järjestelmän kirjautumisikkuna (Preventos Informatics Oy, 2018).



Kuva 6. Aikavälin valinta (Palviainen, 2021).

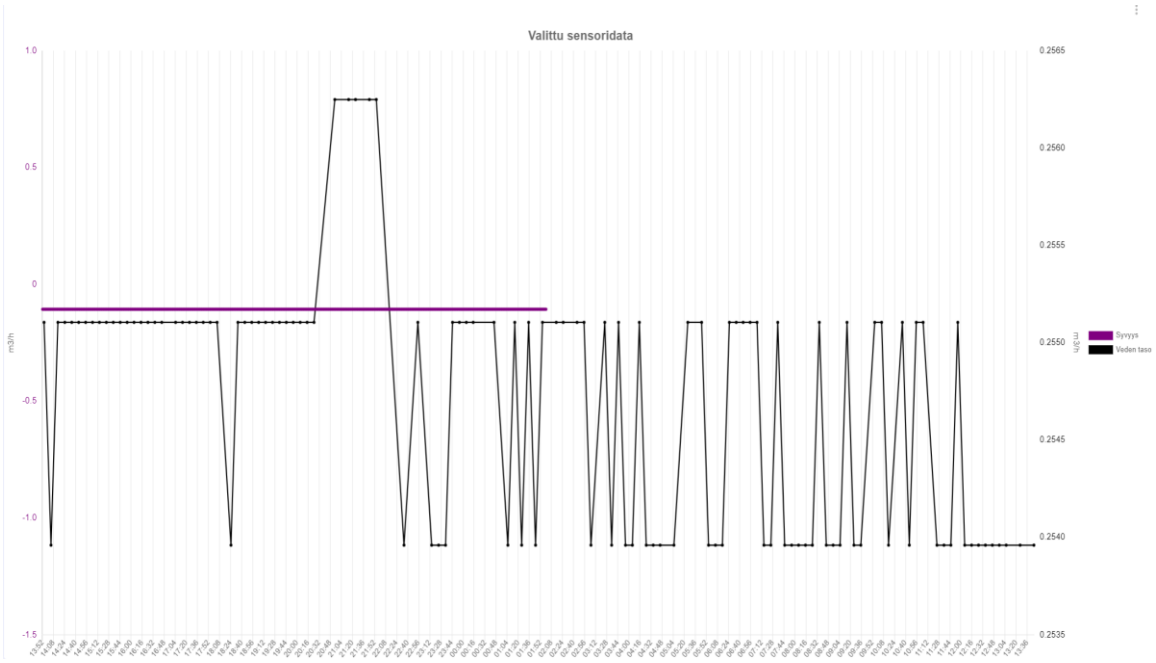
Aikajakso voidaan valita minuutin tarkkuudella mikäli tarpeen on. Usein tämä on liian tarkkaa, sillä kuten mainittu, mittapisteet lähettävät dataa tietyin intervallein tietokantaan (esimerkiksi 6 tunnin välein), mutta joissain tilanteissa se voi olla tarpeen.

Seuraavaksi välimuistista luetusta tiedosta suodatetaan muut sensorilukemat pois jotta saadaan haluttu tieto ja valittu data piirretään graafiin (Kuva 8). Huomioitavaa on että välimuistin dataa itsessään ei tässä kohtaa muuteta, vaan se säilyy saatavilla erikseen asetetun ajan (esim. 1 tunti) jonka jälkeen välimuistista ei enää lueta, vaan uusi tieto haetaan serveriltä ja se ylikirjoittaa välimuistin datan. Tämän tarkoituksena on vähentää serverin kuormitusta, ja koska mittapisteiden data ei päivity välittömästi tietokantaan uuden mittauksen tapahtuessa, vaan tietyn intervallin välein lähetetään uusi datasetti kantaan, seuraisi tästä se että liian usein tapahtuva API kutsu serverille palauttaisi täysin saman datan.



Kuva 7. Kartan popup ja käyttäjän mittapisteet sekä sensorit. Kaikki data ja asiakastiedot mitä näkyy opinnäytetyössä ovat pelkästään testausta varten luotu (Palviainen, 2021). Karttapalvelun tarjosi maanmittauslaitos (Maanmittauslaitos, n.d.).

Punaisella piirretyt mittapisteet sekä kartalla että sen alapuolella indikoivat että ko. pisteissä on ongelmia sensoreissa, koska valitulta ajanjaksolta (tai oletusajanjaksolta -24h) ei ole mittauksia.

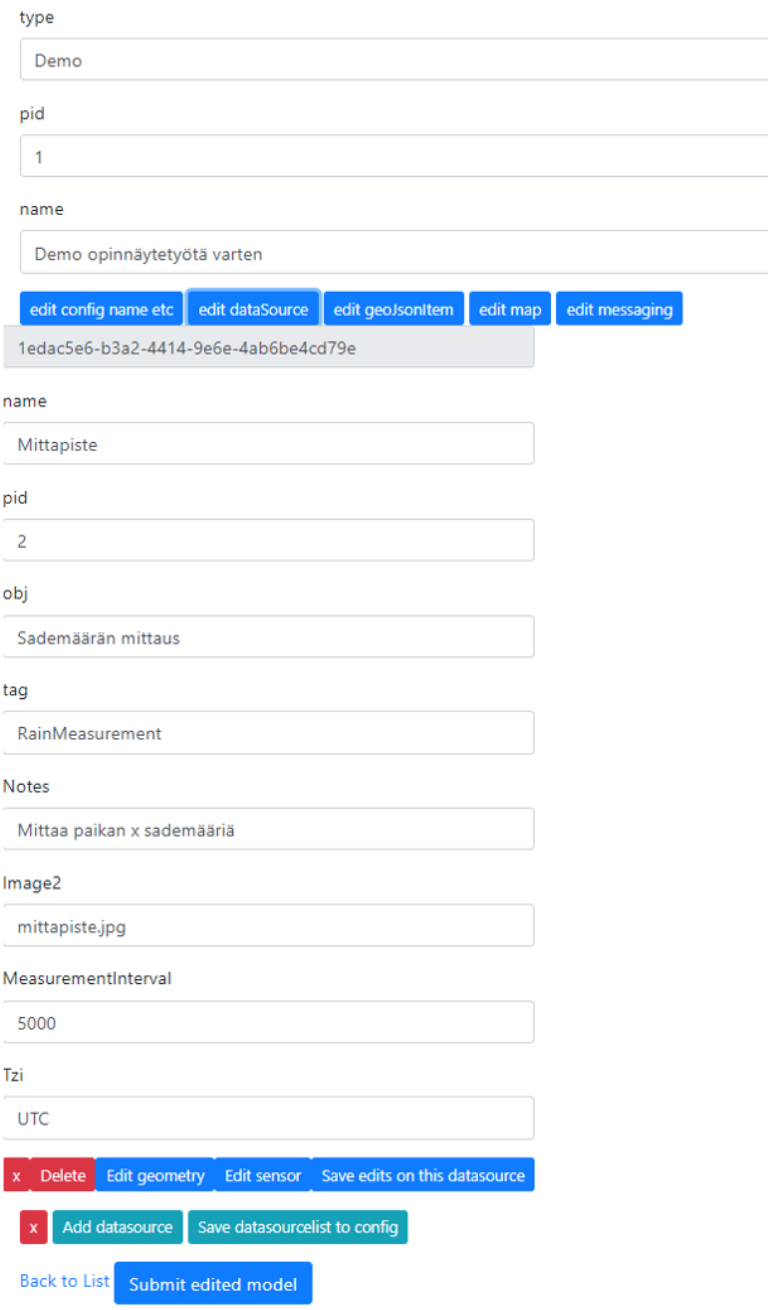


Kuva 8. Kahden sensorin yhtäaikainen piirto samalla aikajaksolle. Molemmille sensoreille piirretään oma y-akselin asteikko omalla värillä (vasen ja oikea reuna kuvassa), jotta asiakkaan on helpompi hahmottaa mittaukset ja lukemat (Palviainen, 2021).

Ylläolevassa kuvassa on haettu viimeisen 24h mittaukset. Kuten graafissa näkyy, on Syvyys-sensori lopettanut mittaukset noin klo. 02:00, jonka jälkeen ei ole mittauksia sille sensorille, ja tästä voi päätellä että sensoriin on tullut jokin vika. Aikajana, eli tässä tapauksessa x-akseli, on yhtenäinen molemmille sensoreille.

5.2 Config

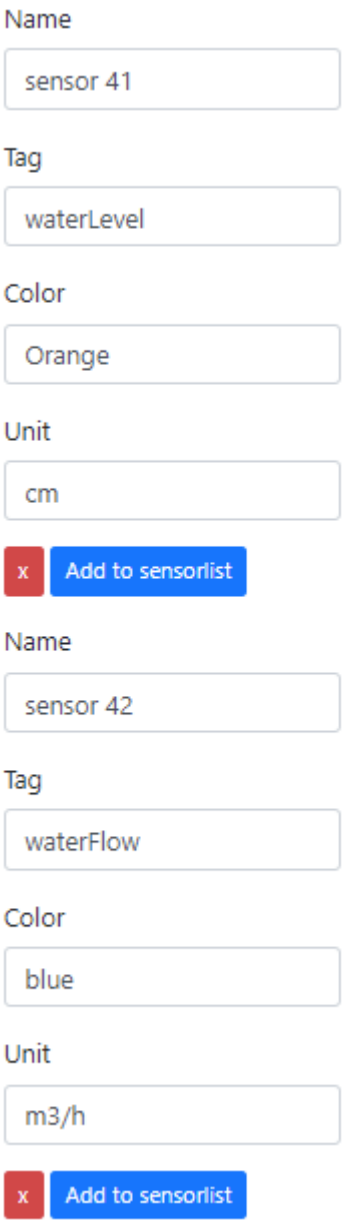
Hero-sovelluksen Adminit tarvitsivat kevyen sovelluksen jolla luoda käyttäjähallinta-konfiguraatioita, jotka sisältävät oleelliset tiedot siihen mihin käyttäjällä on lupa päästä käsiksi, mittapisteet, sensorien tiedot, mittapisteiden sijainnit jne. Erityisesti tarpeen oli sovellus, jossa perusolioon voi lisätä, muokata sekä poistaa eri osioita dynaamisesti. Tehtävä aloitettiin siis luomalla perusobjekti, joka sisälti yhden kappaleen jokaista tarvittavaa osiota, joita täytettiin tarpeen mukaan. Tarvittaessa käyttäjä lisää perusobjektiin arraytä/objekteja, esimerkiksi luodaan DataSources-objekti, se täytetään ja sen jälkeen syötetään DataSources-arrayseen (kuva 9). Jokainen DataSources-objekti sisältää taas omia arraytä sekä objekteja, esimerkiksi useita sensori-olioita arrayssä. Täten voidaan luoda ja täyttää kaikki tarvittavat tiedot, ja ohjelma tallentaa JavaScriptillä luodut lisäobjektit/arrayt oikeaan paikkaan perusobjektissa (Kuva 11).



Kuva 9. Config-olion luominen (Palviainen, 2021).

Alussa oli vaikeuksia saada varsinkin muokatut lisäoliot tallentumaan oikein mikäli käyttäjä ei tiennyt tarkalleen missä järjestyksessä luoda ja tallentaa, ja siitä seurasi että sovelluksella luotiin vain perusobjekti, joka kopioitiin ja täytettiin manuaalisesti. Luonnollisesti tämä ei riittänyt, joten lähdin muokkaamaan logiikkaa uusiksi. Onnekseni huomasin että jo aiemmin tehty logiikka perusobjektin luomisessa oli uudellenkäytettävissä pienten muutoksien jälkeen myös jo olemassaolevan objektin muokkauksessa, ja tämä vähensi työmäärää.

Sovellus rakennettiin .NET Corella käyttäen MVC-arkkitehtuuria, ja itse JSON-objektiin osioiden lisäys/poisto/muokkaus tehtiin JavaScriptin avulla: JavaScriptille annettiin tarvitun objektin (tai objektikokoelman, esimerkiksi Sensors-array) perustiedot (mm. tarvittujen muuttujien nimet sekä mahdolliset olemassaolevat arvot näille muuttujille) jonka pohjalta luodaan uusia muokattavia kohteita (kuva 10). Nämä JavaScriptillä luodut kohteet ovat dynaamisia, eikätkä niinkään kiinteitä osia sovellusta ulkoasullisesti: ne ovat näkyvillä vain tarpeen mukaan väliaikaisesti.



Kuva 10. JavaScriptillä täytettäväksi luodut Sensor-objektit (Palviainen, 2021).

Current config json object


```
{
  "id": "b3a35893-249b-44d7-9d4d-186d1eece253",
  "type": "Demo",
  "pid": 1,
  "name": "Demo opinnäytetyötä varten",
  "map": {
    "startPosition": {
      "lat": 27.63533,
      "lng": 62.91304,
      "zoom": 1
    }
  },
  "geoJsonItems": [
    "demo2.json"
  ],
  "dataSources": [
    {
      "id": "1edac5e6-b3a2-4414-9e6e-4ab6be4cd79e",
      "name": "Mittapiste",
      "pid": 2,
      "obj": "Sademäärän mittaus",
      "tag": "RainMeasurement",
      "notes": "Mittaa paikan x sademääriä",
      "image": "mittapiste.jpg",
      "measurementInterval": "5000",
      "tzi": "UTC",
      "sensors": [
        {
          "name": "Demosensori2",
          "tag": "Dsen2",
          "color": "red",
          "unit": "mm"
        },
        {
          "name": "Demosensori2",
          "tag": "Dsen2",
          "color": "red",
          "unit": "mm"
        }
      ],
      "geometry": {
        "type": "Point",
        "coordinates": [
          28.075624,
          63.075672
        ]
      }
    }
  ],
  "messaging": [
    {
      "method": "Sms",
      "sender": "Reino Palviainen"
    }
  ]
}
```


Kuva 11.Config sovelluksella luotu täytetty JSON-objekti (Palviainen, 2021).

6 POHDINTA

Opinnäytetyön tavoitteena olleet 2 eri osiota saatiin tehtyä. Hero-sovelluksen jatkokehityksessä haluttu lisäosa jossa asiakaalle voidaan helpommin visualisoida mitattua dataa graafien avulla, sekä jossa näkee kartalle piirtyneenä mittapisteet, ja erillinen Config-osio jossa adminit voivat luoda käyttäjäkonfiguraatioita dynaamisesti.

Hero-osion lopputulos on hyödyllinen datan visualisoinnissa, ja tätä konseptia voidaan käyttää muuallakin. Vaatimuksena ollut graafien sisällön toimivuus eri intervalleilla mitatulla mittatuloksilla yhtäaikaisesti helpottaa sovelluksen yleistä jatkokehitystä mikäli vastaavaa toimivuutta tarvitaan muilla sovelluksen osioilla. Osio myös yksinkertaistaa valitun datan haun aikajaksoittain sekä sisältää peruslogiikan datan käsittelylle paikallisesti. Jatkokehityksen puolella tätä osiota voitaisiin kehittää tiedon paikallisessa tallennuksessa: tallennuslogiikka jätettiin jokseenkin yksinkertaiseksi koska se riitti toistaiseksi, mutta siihen voisi panostaa tulevaisuudessa.

7 LÄHDELUETTELO

Agafonkin, V. (ei pvm). *Leaflet*. Noudettu osoitteesta Leafletjs: https://leafletjs.com

Bootstrap. (ei pvm). *Bootstrap*. Noudettu osoitteesta Bootstrap: https://getbootstrap.com

Chartjs. (ei pvm). Noudettu osoitteesta Chartjs: https://www.chartjs.org

Consuegra, D. (ei pvm). *Advancedkittenry*. Haettu 17. 12 2021 osoitteesta Arkkitehtuuri ja MVC: https://advancedkittenry.github.io/koodaaminen/arkkitehtuuri/index.html

Facebook. (2021). *React*. Noudettu osoitteesta React: https://reactjs.org

Maanmittauslaitos. (ei pvm). *Maanmittauslaitos*. Noudettu osoitteesta https://www.maanmittauslaitos.fi

Microsoft. (ei pvm). *.NET*. Noudettu osoitteesta Microsoft: https://dotnet.microsoft.com/en-us/

Mozilla. (20. 5 2021). *Developer Mozilla*. Haettu 17. 12 2021 osoitteesta Date: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/valueOf

Mozilla. (29. 5 2021). *JavaScript*. Noudettu osoitteesta Mozilla: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Palviainen, R. (20. 12 2021). *Hero-sovelluksen kehitys ja tuotteistaminen*. Kuopio, Finland.

Preventos Informatics Oy. (2018). Hero.

Recharts. (ei pvm). *Recharts*. Noudettu osoitteesta https://recharts.org

Sharma, S. (ei pvm). *Benefits and Drawbacks of MVC Architecture*. Noudettu osoitteesta sheysharma.com: https://shreysharma.com/benefits-and-drawbacks-of-mvc-architecture/

Staff. (2. 2 2021). *Computerscience*. Haettu 17. 12 2021 osoitteesta Computer Programming Languages: https://www.computerscience.org/resources/computer-programming-languages/

Wikipedia. (4. 3 2020). *MVC-Arkkitehtuuri*. Noudettu osoitteesta Wikipedia: https://fi.wikipedia.org/wiki/MVC-arkkitehtuuri

Wikipedia. (14. 12 2021). *.NET*. Noudettu osoitteesta Wikipedia: https://en.wikipedia.org/wiki/.NET

Wikipedia. (3. 11 2021). *Boostrap (front-end framework)*. Noudettu osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

Wikipedia. (13. 11 2021). *Kirjasto (tietotekniikka)*. Haettu 17. 12 2021 osoitteesta Wikipedia: https://fi.wikipedia.org/wiki/Kirjasto_(tietotekniikka)

Wikipedia. (9. 7 2021). *Wikipedia*. Noudettu osoitteesta Esineiden Internet: https://fi.wikipedia.org/wiki/Esineiden_internet