SAVONIA
ammattikorkeakoulu

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

PIPEDRIVE DATASTA VISUALI-SOITU KÄYTTÖLIITTYMÄ

TEKIJÄ Iiro Välimaa

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Iiro Välimaa	
Työn nimi Pipedrive datasta visualisoitu käyttöliittymä	
Päiväys 21.1.2024	Sivumäärä/Liitteet 24
Toimeksiantaja/Yhteistyökumppani(t) Metatavu Oy	

Tiivistelmä

Tässä opinnäytetyössä kehitettiin toimeksiantajan CRM-dataa visualisoiva työkalu, jonka tarkoituksena on tuoda projekteja esille aikaisemmassa vaiheessa. Haettu data lajitellaan projektien vaiheen mukaan. Työkalun tarkoituksena on myös antaa työntekijöille mahdollisuus ilmoittaa kiinnostuksensa tulevia projekteja kohtaan.

Opinnäytetyön sovellus kehitettiin suoraan toimeksiantajan käynnissä olevaan projektiin ja se tehtiin vastaamaan yrityksen sovelluskehitysstandardeja- ja käytäntöjä. Toteutettu työ hyväksytettiin yrityksellä ja siihen tehtiin vaadittavia korjauksia, kunnes se läpäisi standardit. Sovelluksen kehittämisessä käytettiin toimeksiantajan käyttämiä moderneja teknologioita kuten React, Typescript, Swagger, sekä Serverless framework. Näistä vain React oli entuudestaan tuttu. Työn tavoitteena oli luoda mahdollisimman lähelle tuotantokelpoista oleva sovellus, jotta yritys voi itse mukauttaa sovelluksen omiin tarpeisiinsa.

Tässä raportissa käydään läpi opinnäytetyössä käytetyt tekniikat, esitellään sovelluksessa käytettävä CRM työkalu ja sen ominaisuudet, sekä käydään läpi toteutettu sovellus ja sen ominaisuudet. Raportin lopussa kerrotaan, miten kehitystyö sujui, mitä ongelmia oli, mitä saatiin aikaiseksi, sekä mitä jäi tekemättä.

Avainsanat

React, Typescript, Serverless, Swagger, REST, Ohjelmistokehitys

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Iiro Välimaa	
Title of Thesis User Interface Visualized from Pipedrive Data	
Date 21 January 2024	Pages/Appendices 24
Client Organisation /Partners Metatavu Oy	

Abstract

The purpose of this thesis was to develop a web application that visualizes the client's CRM data with a purpose of bringing up projects in the early stages. The data retrieved from the CRM will be sorted based on the stage of the project. The application also provides employees with a way to express their interest in the projects.

The application created in this thesis was developed directly to the client's project and it was made to meet the client's standards and best practices. The implemented work was approved by the company, and necessary fixes were made until the application met the needed quality. Modern technologies were used in the development of the application, such as React, Typescript, Swagger and the Serverless framework. The goal of the thesis was to develop the software to be as close to production readiness as possible, allowing the company to make changes to the application to meet the company's needs.

This report covers the technologies used in the thesis, introduces the CRM tool and its features used in the application and introduces the application and its features created in this thesis. At the end of the report, the development process and the arisen problems are reviewed. The accomplishments and the yet to be completed tasks are presented.

Keywords
React, Typescript, Serverless, Swagger, REST, Software Development

SISÄLTÖ

KUVALUETTELO

KÄSITTEET

API	Ohjelmointi rajapinta
Pipedrive	CRM-ohjelmiston tarjoava yritys
CRM	Asiakkuudenhallinta-järjestelmä
Sandbox	Testausympäristö, joka antaa autenttisen kokemuksen testaamiseen
Framework	Valmiiksi kasattu setti uudelleenkäytettäviä kirjastoja ja komponentteja.
UI	Käyttöliittymä
Syntaksi	Varattujen sanojen ja lauseiden tunnistus
Endpoint	Rajapinnan päätepiste
Postman	Rajapintojen testaamiseen käytettävä työkalu
Serveless	Palvelin, jonka käytöstä laskutetaan vain suoritusajan verran
Github	Versionhallinta työkalu
Repository	Githubiin tallentuva tietovarasto

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on kehittää opinnäytetyön toimeksiantajalle, Metatavulle työkalu, joka visualisoi heidän tulevia projektejaan. Työkalu olisi tulossa heidän sisäiseen käyttöönsä ja tarkoituksena on tuoda tulevia projekteja aikaisemmassa vaiheessa näkyville. Työkalu lajittelee projekteja statuksen ja julkaisuajan mukaan. Projekteissa tulisi näkyä myös niissä käytetyt teknologiat, sekä nappi, jolla yrityksen työntekijät pystyvät ilmoittamaan olevansa kiinnostuneita työskentelemään projektissa.

Opinnäytetyön toimeksiantaja on Mikkelissä sijaitseva Metatavu Oy. Metatavu on 2016 perustettu ohjelmistotalo. Nykyään yritys tarjoaa ohjelmistokehityksen lisäksi palvelumuotoilua, konseptointia, sekä teknistä määrittelyä. Yritys on keskittynyt avoimen lähdekoodin ratkaisujen kehittämiseen. Myös opinnäytetyö on toteutettu avoimella lähdekoodilla.

Opinnäytetyö kehitetään selaimessa käytettäväksi sovellukseksi ja se liitetään yrityksen aikaisempaan projektiin. Sovelluksen käyttöliittymä toteutetaan React.js kirjastolla, joka käyttää elementteihin Material UI -elementtikirjastoa. Sovelluksen koodissa käytetään Typescriptiä normaalin JavaScriptin sijasta. Rajapinnan toteuttamiseen käytetään Serverless framework:ia ja sen apuna Swagger työkalusarjaa. Projektia hallinnoidaan GitHub versionhallintajärjestelmässä.

2 KÄYTETYT TEKNIIKAT

2.1 HTML

HTML (HyperText Markup Language) on verkkosivujen rakentamiseen tarkoitettu merkintäkieli. Kieli sai alkunsa vuonna 1989 Tim Berners-Lee:n konseptista liittää tutkimuspapereita hypertekstiin. Tarkoituksena oli mahdollistaa tutkimuspapereiden helppo jakaminen verkonvälityksellä (W3C julkaisupäivä tuntematon).

HTML verkkosivut koostuvat elementeistä. Elementit erotetaan toisistaan tageja käyttämällä, jotka kertovat, mitä kukin elementti sisältää, sekä kuinka se tulisi näyttää (Mozilla 2023).

2.2 CSS

CSS (Cascading Style Sheet) on verkkosivujen ulkoista olemusta määrittelevä kieli. CSS avulla käyttäjä voi määritellä, miten ja minkä näköisinä esimerkiksi verkkosivujen elementtejä tulisi näyttää (Mozilla 2023).

2.3 JavaScript

JavaScript (alun perin ECMAScript) on vuonna 1995 Brendan Eich:in kehittämä, verkkosivujen ohjelmointi- ja skriptikieli (Peltomäki 2017, 1). Nykyään JavaScriptiä käytetään muihinkin tarkoituksiin, kuin vain verkkosivuihin. Tällainen käyttötarkoitus on esimerkiksi Node.js, jolla voidaan rakentaa palvelinpuolen sovelluksia (Peltomäki 2017, 7).

JavaScript on yksisäikeinen, synkronisesti tai asynkronisesti ajettava kieli. Synkronisessa ajossa koodia ajetaan sitä mukaan, miten se on kirjoitettu. Asynkronisessa ajossa JavaScript voi suorittaa jo seuraavan asian sillä välin, kun se odottaa vielä vastausta edeltävältä vaiheelta (Mozilla 2023).

Ensimmäisissä versioissa JavaScript oli tulkattava, joka teki siitä hitaan. Modernit selaimet käyttävät JIT-kääntäjiä (Just in-time compilation), joista ehkä tunnetuin on Googlen V8-selainmoottori. Nämä JIT-kääntäjät ovat tehostaneet JavaScriptiä ja sen toimintaa selaimissa. JIT-käännös muuntaa JavaScriptin konekieleksi sitä mukaa, kun sitä ajetaan (Peltomäki 2017, 5).

2.4 TypeScript

Typescript on Microsoftin kehittämä, avoimeen lähdekoodiin perustuva JavaScript kirjasto, jonka tarkoituksena on tehostaa JavaScript ohjelmointia, tarjoamalla paremmat virheiden käsittelyt, sekä laajemmat virheilmoitukset (Typescript 2023).

2.5 React.js

React.js on Metan, eli entisen Facebookin kehittämä avoimeen lähdekoodiin perustuva JavaScript kirjasto. React.js on tarkoitettu rakentamaan verkkosivujen käyttöliittymiä komponenttipohjaisesti. Tämä tarkoittaa sitä, että esimerkiksi normaalin napin HTML-koodi tarvitsee luoda vain kerran, ja tätä komponenttia voidaan käyttää aina uudestaan ja uudestaan ilman, että samaa koodia joutuu kirjoittamaan monta kertaa (Meta julkaisuaika tuntematon). Alla kuvassa 1 esimerkki React.js-syntaksista, jossa JavaScript funktio palauttaa HTML-elementin.

```
function App() {
  return (
    <div className="App">
      <h1>Hello World!</h1>
    </div>
  );
}
```

KUVA 1. Esimerkki React.js syntaksista.

2.6 Node.js

Node.js on Ryan Dahlin vuonna 2009 JavaScriptillä kehittämä tapahtumapohjainen palvelinpuolen ajoympäristö. Node oli alkujaan tarjolla vain UNIX pohjaisille käyttöjärjestelmille, kuten Linux ja Mac OS X, mutta Microsoft julkaisi siitä Windows-version vuonna 2011 (Peltomäki 2017, 7). Nykyään Node on alustariippumaton (GitHub 2023).

Noden toiminta perustuu asynkroniseen tapahtumapohjaiseen ajomalliin. Tämä tarkoittaa, että Node pyörii jatkuvassa silmukassa, jossa jokainen tapahtuma tulee olla estämätön (non-blocking), jotta silmukka ei pysähdy. I/O-toiminta suoritetaankin siis takaisinkutsuvilla funktioilla eli callback-funktioilla. Callback-funktiot ovat muille funktioille parametreinä annettuja funktioita (Peltomäki 2020, 14–17).

I/O (Input/Output) eli siirräntä tarkoittaa asioita, jotka liikkuvat systeemin ulkopuolelta sisäänpäin tai sisäpuolelta ulos. I/O tapahtumia ovat esimerkiksi palvelimelta tiedon hakeminen tai sinne kirjoittaminen.

2.7 Npm

Npm on Node.js ajoympäristölle vuonna 2009 kehitetty pakettien hallintaohjelma (Npm julkaisuaika tuntematon). Vuoden 2019 kesäkuuhun mennessä Npm:llä oli 12 miljoonaa käyttäjää ja sisälsi yli miljoona pakettia (Nassri 2020).

2.8 JSON

JSON (JavaScript Object Notation) on datan tallennukseen ja siirtoon kehitetty yksinkertainen ja kevyt tiedostomuoto. Nimensä mukaisesti se on alun perin kehitetty JavaScriptiä varteen, mutta sitä voidaan käyttää nykyään lähes kaikissa moderneissa C-sukuisissa kielissä (JSON julkaisuaika tuntematon). JSON koostuu kahdesta osasta, avaimesta ja arvosta (Peltomäki 2017, 114).

2.9 CORS

CORS (Cross-Origin Resource Sharing) on protokolla, joka auttaa palvelimia päättämään mitkä osoitteet ovat hyväksyttäviä ja mitkä mahdollisesti kiellettyjä tai haitallisia (Mozilla 2023). CORS ei ole

kuitenkaan täydellinen suoja Cross-Origin hyökkäyksiä vastaan, mikäli se on huonosti tai väärin konfiguroitu (PorstSwigger julkaisuaika tuntematon).

2.10 Rajapinta

API (Application programming interface) eli sovelluksen ohjelmointi rajapinta on joukko ominaisuuksia ja sääntöjä, jotka toimivat ohjelman sisällä, mahdollistaen kanssakäymisen sovelluksen kanssa. Rajapintaa voidaan pitää yksinkertaisena sopimuksena sitä tarjoavan sovelluksen ja muiden esineiden, kuten kolmansien osapuolten välillä (Mozilla 2023).

2.11 REST-arkkitehtuuri

REST (Representional State Transfare) on arkkitehtuurityyli, joka luotiin ohjeistamaan World Wide Webin kehitys- ja suunnitteluarkkitehtuuria. REST arkkitehtuurin ansiosta on mahdollista luoda skaalautuvia systeemejä, jotka pysyvät isoinakin joukkoina tehokkaina (Mozilla 2023). Rajapintoja, jotka käyttävät REST arkkitehtuuria, kutsutaan RESTful API:ksi (Red Hat 2020).

2.12 Dotenv

Dotenv on moduuli, johon käyttäjä voi tallentaa ympäristömuuttujia, kuten palvelimen osoitteita, käyttäjätunnuksia tai salasanoja. Näitä muuttujia säilytetään env-tiedostopäätteisessä tiedostossa, eikä niitä viedä versionhallintaan, vaan ne säilytetään palvelimella (Dotenv julkaisupäivä tuntematon).

2.13 Serverless

Serverless palvelimella tarkoitetaan palvelua, jossa palvelimen tarjoaja veloittaa käyttäjältä vain prosessin keston mukaisesti. Serverless ei siis tarkoita palvelimetonta, vaan sen tarkoituksena on vähentää tarvetta vuokrata kokonaisia palvelimia. Tarkoituksena on tarjota vain tarvittuun tehtävään vaadittava määrä prosessointia, josta veloitetaan vain prosessiin käytetyn ajan (Cloudflare julkaisuaika tuntematon). Serverless palvelun tarjoajia ovat esimerkiksi Amazonen AWS, jota myös opinnäytetyön toimeksiantaja käyttää.

2.14 Swagger

Swagger tarjoaa kehittäjille mahdollisuuden luoda REST-rajapintoihin tiukempia rakenteita. Kehittäjä kuvailee rajapintansa, sekä siinä käytettävät oliot. Kuvailun perusteella rakennetaan rajapinta ja lopuksi Swagger-tiedostosta voidaan luoda koodi frontendiin. Myös palvelinpuolen koodin luominen on mahdollista. Tällä pyritään vähentämään suunnittelun ja kehityksen aikana syntyviä virheitä. Swagger tiedostoa pystytään käyttämään myös dokumentointiin (Swagger julkaisuaika tuntematon).

2.15 Material UI

Material UI on avoimeen lähdekoodiin perustuva React.js komponenttikirjasto, joka tarjoaa laajan valikoiman valmiita elementtejä. Kirjasto käyttää Googlen Material tyyliä pohjanaan CSS tyyleille.

Kirjastot pystytään asentamaan käyttämällä, joko npm, yarn tai pnpm pakettienhallinta ohjelmaa. Lisäksi fontteja pystytään liittämään suoraan tiedostoihin käyttämällä link-elementtiä (Material UI julkaisuaika tuntematon).

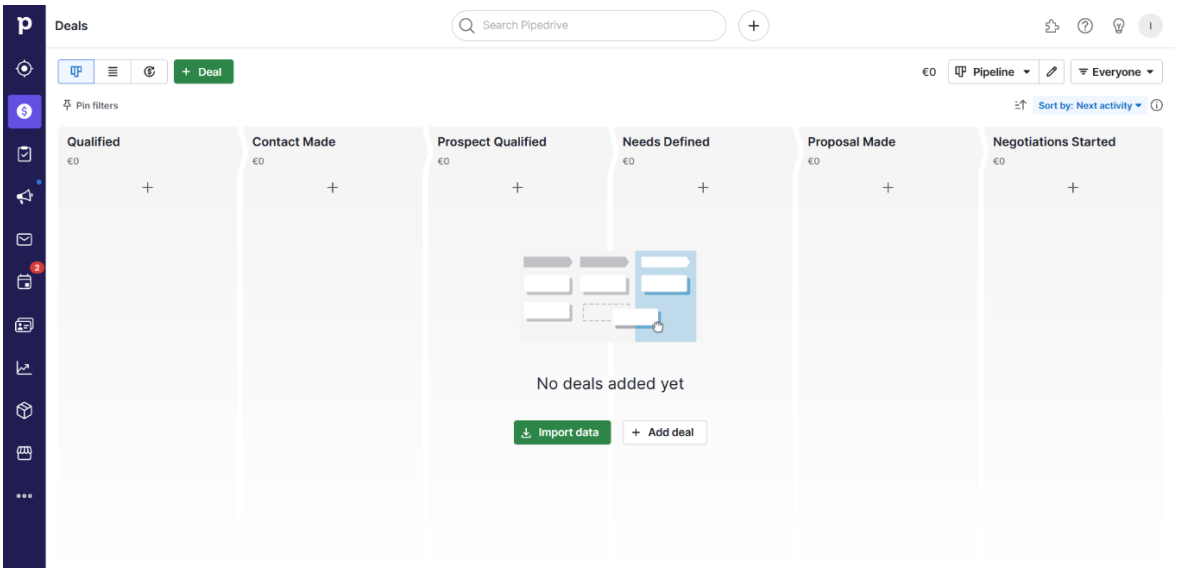
3 PIPEDRIVE

3.1 Yleistä Pipedrivestä

Pipedrive on vuonna 2010 alkunsa saanut yritys, jonka perustamisen lähtökohtana oli tarjota monipuolisempi CRM-järjestelmä, kuin markkinoilla oli siihen aikaan tarjota. CRM-järjestelmä on asiakassuhteiden hallinnointiin luoto työkalu, jolla voidaan tallentaa, hallita ja seurata myyntidataa. Tätä dataa voi olla esimerkiksi mahdolliset myynnit, toteutuneet kaupat, sekä tulevat projektit. Pipedrive tarjoaa kehittäjille laajasti tietoa ja ohjeita verkkosivuillaan (Pipedrive julkaisuaika tuntematon).

3.2 Pipedrive Sandbox

Pipedrive tarjoaa kehittäjille Sandboxin eli hiekkalaatikon, jossa ohjelmoijat voivat kehittää ja testata sovellustaan ilman pelkoa yrityksen oikean datan vahingoittamisesta tai hävittämisestä. Hiekkalaatikko vastaa oikeaa Pipedrive-tiliä ja sisältää kaikki samat ominaisuudet kuin oikea tili (Pipedrive julkaisuaika tuntematon). Kuvassa 2 on esitetty Pipedrive Sandbox:in oletusnäkymä.



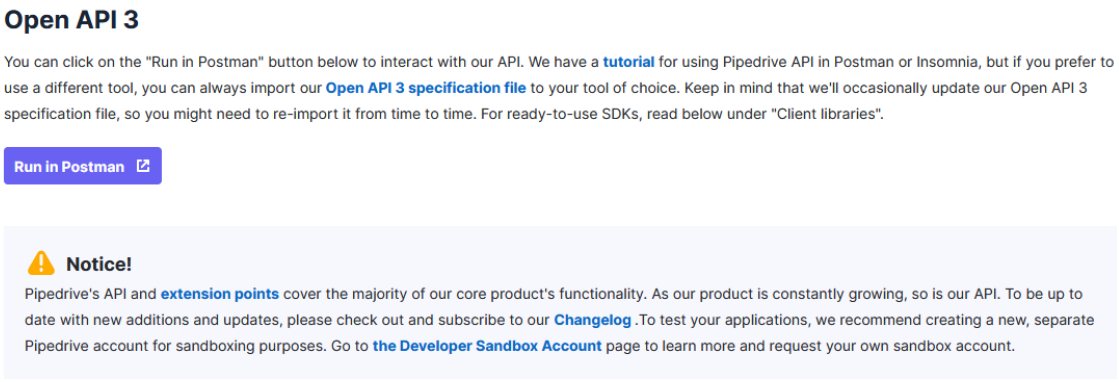
KUVA 2. Oletusnäkymä Pipedrive Sandbox:in etusivulla (Pipedrive julkaisuaika tuntematon).

Hiekkalaatikon tarkoituksena on suojata yrityksiä ja niiden dataa sovelluksen kehityksen ja sen testauksen aikana tapahtuvien virheiden aiheuttamilta vahingoilta. Tällainen vahinko voi olla esimerkiksi datan tahaton poistaminen, mikä voi vahingoittaa yritystä niin rahallisesti, kuin myös imagollisesti.

3.3 Pipedrive API

Pipedrive API on REST arkkitehtuuria käyttävä rajapinta, jonka läpi Pipedriven asiakkaat pääsevät käyttämään Pipedriven dataa. Rajapinta palauttaa käyttäjälle datan JSON muodossa. Rajapinta kykenee käsittelemään CORS pyyntöjä (Pipedrive julkaisuaika tuntematon).

Kuten kuvassa 3 näkyy, Pipedrive tarjoaa mahdollisuuden kopioida heidän rajapintansa päätepisteiden käyttämät ohjeet suoraan Postmaniin.



KUVA 3. Pipedriven ohjeistus rajapinnan päätepisteiden kopioimisesta Postmaniin (Pipedrive julkaisuaika tuntematon).

3.4 Custom Fields

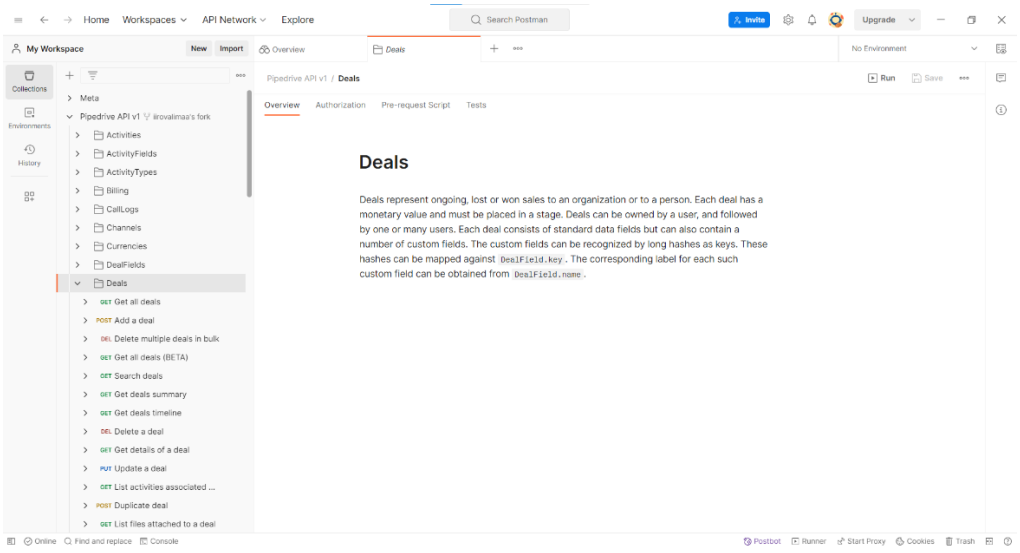
Pipedrive sisältää ominaisuuden luoda uusia datakenttiä käyttäjien tarpeisiin. Näitä mukautettuja kenttiä pystytään luomaan jokaiseen oliotyyppiin Pipedrivessä. Tämä kattaa diilit, organisaatiot, henkilöt sekä tuotteet (Pipedrive julkaisuaika tuntematon).

Mukautettu kenttä voi olla tyypiltään mikä tahansa 16 kenttätyypistä, jotka Pipedrive tarjoaa. Näitä tyyppejä ovat esimerkiksi erilaiset tekstit, oliot, numerot, sekä monivalinnat. Huomioitavaa on, että taulukko ei ole yksi valittavista tyypeistä.

3.5 Postman

Postman on Abhinav Asthanan, rajapintojen kehittämistä ja niiden testaamista varten kehitetty työkalu, jolla pystyy tekemään API-kutsuja ja vastaanottamaan niiden vastauksia. Myöhemmin Asthnana perusti sovelluksen nimeä kantavan yrityksen Ankit Sobtin sekä Abhijiy Kanen kanssa (Asthana 2021).

Postmanissa pääsee käsiksi Pipedrive-rajapinnan päätepisteisiin. Nämä päätepisteet kopioitiin tilille edeltävässä kappaleessa. Kuvassa 4 on näkymä, jossa käyttäjä voi tutkia tarkemmin rajapinnan päätepisteiden metodeja, niiden vaatimia parametrejä, sekä testata niitä.



KUVA 4. Pipedrive rajapinnan kopio Postmanissa.

4 SUUNNITTELU

Suunnitteluvaihe on tärkeä osa ohjelmistokehityksen prosessia. Vaikka toteutankin projektin pääsääntöisesti itsenäisesti, on hyvä kirjata ylös edes pääpiirteinen suunnitelma. Suunnitelma on myös hyvä tehdä siltä varalta, että jotain unohtuu. Suunnitelmaa seuraamalla tietää, mitä kaikkea projektissa tulee tehdä. Kun sovelluksen vaatimat rakennuspalikat on saatu listattua, on hyvä käydä ne läpi ja miettiä, missä järjestyksessä kehityksen vaatima tehtävä kannattaa suorittaa.

Ennen kehitysvaihetta kirjoitin vapaamuotoisen suunnitelman, kuinka toteutan käyttöliittymän sekä rajapinnan. Piirsin sovelluksen pääpiirteet ja listasin ne. Suunnittelin myös tekeväni sovelluksen elementeistä monikäyttöisiä, joka saattaa viedä enemmän aikaa suunnitteluvaiheessa, mutta joka säästää kehitysvaiheessa aikaa.

4.1 Yrityksen vaatimukset

Toimeksiantajan antama tehtävä oli selkeä. Toteutettavan sovelluksen tulee hakea yrityksen projektidataa heidän Pipedrive-tililtään ja se tulee esittää verkkosivulla visuaalisesti selkeinä elementteinä. Lisäksi sovelluksen käyttäjällä tulee olla mahdollisuus esittää kiinnostuksensa projektia kohtaan.

Vaikka toimeksiantaja oli antanut minulle suhteellisen vapaat kädet, oli minulle asetettu pari vaatimusta toteutukseen. Sovelluksen käyttöliittymän tulee olla kirjoitettu Reactilla ja sen tulee olla käyttää Typescriptiä normaalin Javascriptin sijasta.

Minulla oli heti ensimmäisestä palaverista lähtien päässä kuva, minkä näköinen käyttöliittymä tulisi olemaan.

4.2 Käyttöliittymä

Käyttöliittymä on oleellinen osa tätä opinnäytetyötä, koska projekti vaatii datan visualisointia ja lajittelua. Datan visualisoinnilla pyritään auttamaan ihmistä käsittelemään tietoa tehokkaammin, jotta tehtävän suorittaminen olisi tehokkaampaa. Lisäksi visualisointia tehdessä tulee ottaa huomioon kohdeyleisö.

Sovelluksen ulkoasuun minulle oli annettu vapaat kädet, mutta koska Metatavu-home käyttää valmista UI-kirjastoa oli viisasta lähteä tekemään sen kanssa yhtenäistä käyttöliittymää.

4.3 Rajapinnan suunnittelu

Projektissa käytetään kahta rajapintaa. Ensimmäinen on Metatavun AWS Lambda -palvelin, jonka tarkoituksena on piilottaa Pipedrive rajapinnan käyttöön vaadittava avain. Toinen käytetty rajapinta on Pipedrive API, josta itse tiedot haetaan.

Teknistä toteutusta varten tuli miettiä rajapinnan toiminnot, mihin rajapintaa käytetään sekä mitä sen tulee tehdä. Pääasiassa sovellus hakee tietoa HTTP GET -metodilla, mutta myös PUT- ja PATCH-metodia tarvitaan, jotta henkilöiden kiinnostus tulevia projekteja kohtaan saadaan lisättyä kyseisiin projekteihin. Samat metodit pätevät myös kiinnostuksen poistamiseen projekteista.

4.4 Mukautettujen kenttien suunnittelu ja käyttö

Opinnäytetyön yksi osa oli antaa työntekijöille mahdollisuus ilmoittaa kiinnostuksensa tuleviin projekteihin. Tämän toteuttaminen vaati eniten mietintää, koska Pipedrive ei tarjoa mahdollisuutta tallentaa dataa taulukko- tai listamuodossa. Siksi päädyin ratkaisuun, jossa käyttäjien id tallennetaan merkkijonona Pipedriveen, sille luotuun mukautettuun kenttään. Tässä kentässä id:t on eroteltu toisistaan puolipisteellä. Lisäksi projekteissa käytetyt teknologiat käyttävät mukautettua kenttää. Tässäkin tapauksessa asiat erotellaan toisistaan puolipisteellä.

5 TOTEUTUS

5.1 Kehitys ja toteutus

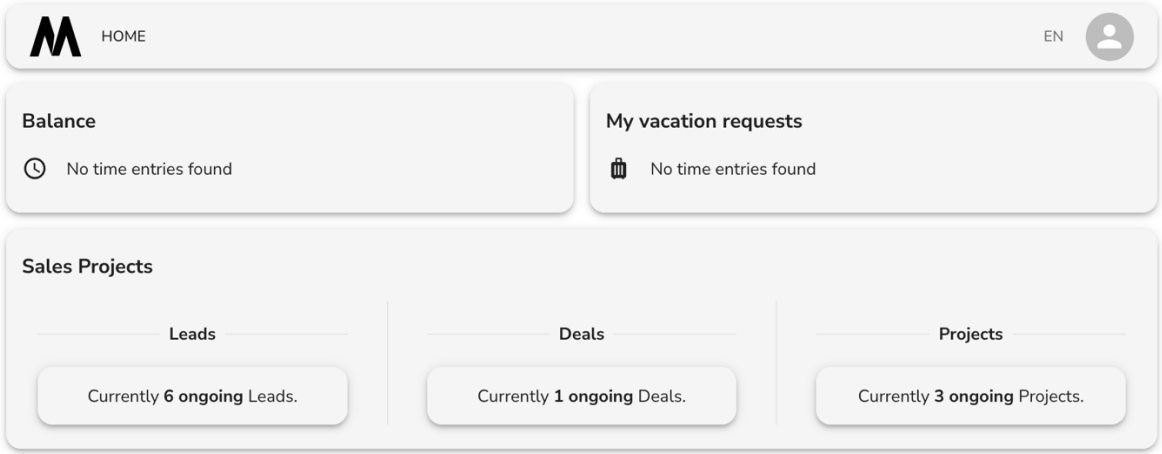
Sovelluksen kehitys lähti käyntiin paikallisena projektina ja vapaalla aikataululla. Aluksi tarkoitukseni oli testata projektiin käytettäviä perustoimintoja, kuten hakufunktioita erillisessä projektissa, jota käytin prototyyppinä ja testausalustana koko projektin ajan. Käytin kehityksessä apuna GitHubia, johon tallensin projektin lähdekoodin.

Sovellustani varten tarvitsin Metatavun autentikoinnin, joka oli alun perin tarkoitus lisätä sovellukseeni. Päädyimme kuitenkin Metatavun kanssa nopeasti siihen päätökseen, että sovellusta olisi helpompi kehittää suoraan heidän ”Metatavu-home”-projektiinsa, joka sisältäisi jo valmiiksi vaadittavan toimintaympäristön. Lisäksi tämä helpottaisi huomattavasti sovelluksen integrointia, joka tulisi ennemmin tai myöhemmin vastaan.

Metatavu-home projekti on yrityksen sisäinen työkalu, josta työntekijät pystyvät tarkkailemaan työaikaa sekä tekemään että muokkaamaan loma-anomuksia.

5.2 Etusivu

Metatavu-home on rakennettu Material UI CSS-kirjastolla. Etusivulla jokainen elementti on Card elementti. Tein siis etusivulle jo olemassa olevaan ympäristöön sopivan elementin, jossa näkyy vain vähän dataa, josta kaikki on oleellista. Kuvan 5 Sales projects komponentin pylväät hakevat kaikki kyseiseen luokkaan sisältyvät projektit ja laskevat niiden määrän.

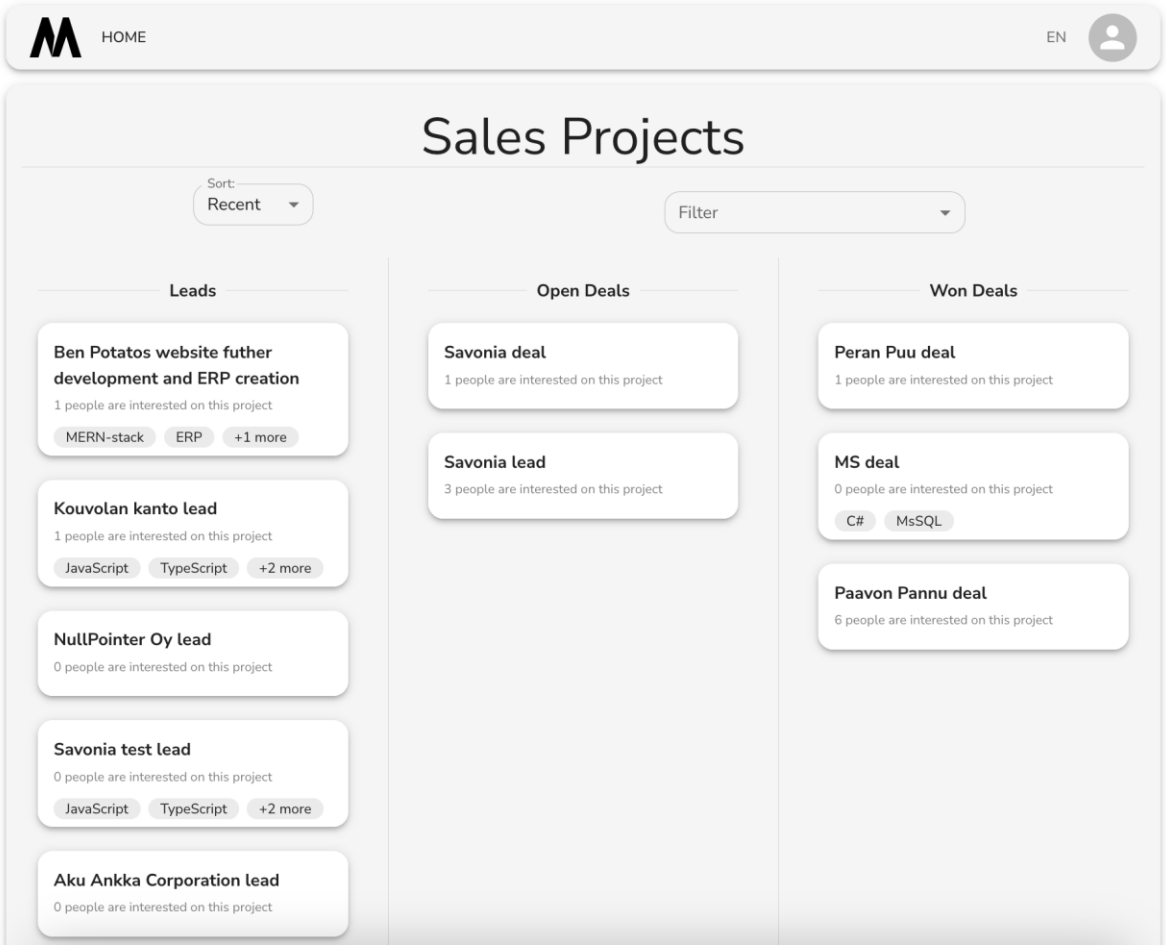


KUVA 5. Sales Project elementti Metatavu-home projektin etusivulla.

5.3 Pää-elementti

Kuvassa 6 näkyvä sovelluksen pääelementti on linkitetty Metatavu-home:n alahakemistoksi, jossa sivu kokoaa koodista ”Pipedrive-screen” -elementin. Elementti itsessään on jaettu useaan osaan, eli otsikkoon, hakuasetuksiin sekä pylväisiin, jotka käyttävät samaa koodia listan rakentamiseen. Jotta sovellus osaa hakea oikean datan oikeaan pylvääseen, elementille annetaan parametrinä otsikko-

teksti, sekä hakuparametriteksti. Hakuparametrin ei tarvitse vastata Pipedrive-rajapinnan hakuparametrejä, sillä kyseistä hakuparametriä käytetään aktivoimaan ensimmäisen rajapinnan lambda-funktio.



KUVA 6. Sovelluksen pääelementti.

Leads:it tarkoittavat mahdollisia kauppoihin johtavia kyselyitä ja tarjouksia. Deals:illä tarkoitetaan tarjouksia, jotka on jo käsitelty. Ne voivat olla joko voitettuja tai hävittyjä tarjouksia. Pipedriven tarjoama projects-erittely on maksullinen, eikä Metatavu ole nähnyt tarpeelliseksi maksaa siitä. Projektien tilalla käytetäänkin siksi voitettuja diilejä ja diilien tilalla avoimia diilejä.

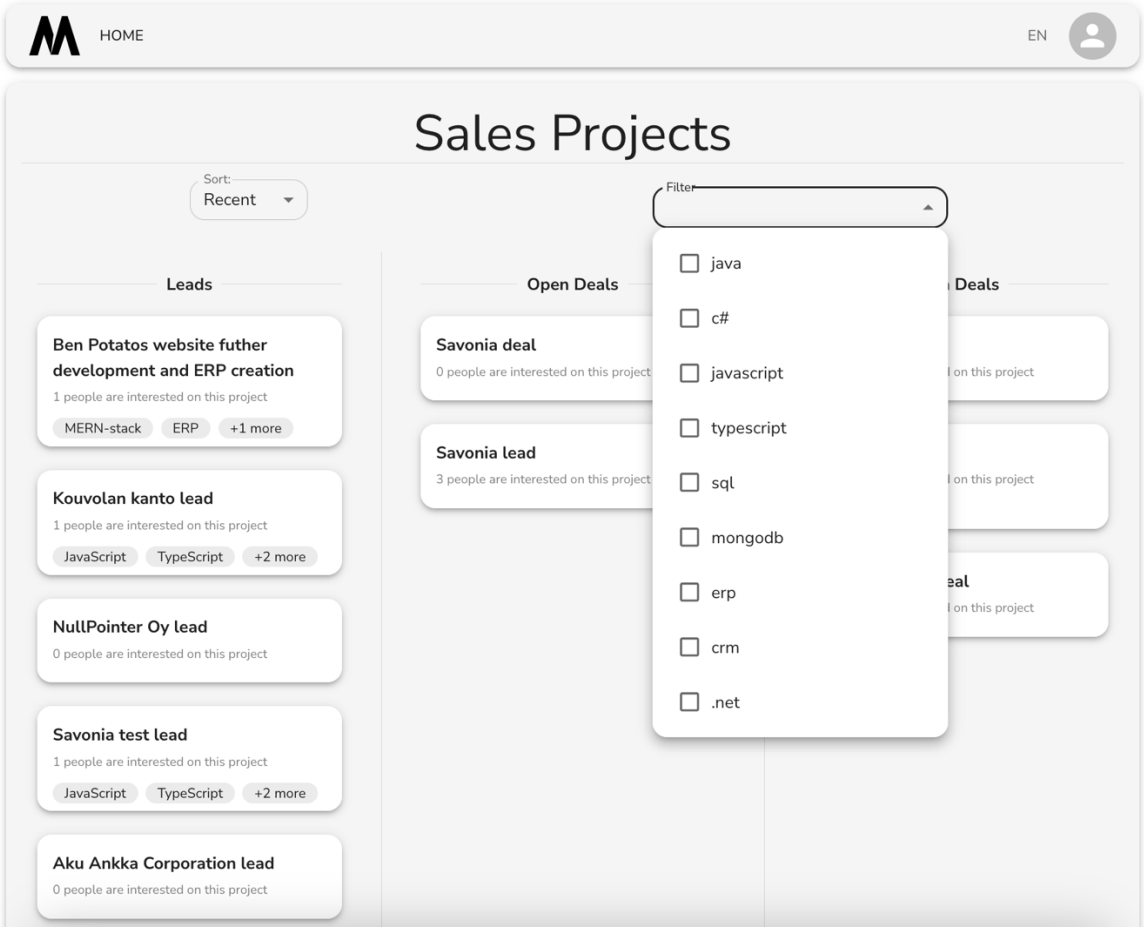
Nämä kolme eri ryhmää (Leads, Open Deals, Won Deals) käyttivät kaikki samaa koodia. Ainoana eroavaisuutena oli niille annettavat parametrit. Title on visuaalisesti näytettävä otsikko ja rowtype määrittää haettavan datan. Rowtype-parametri ei ole suoraan verrattavissa Pipedriven rajapinnan parametriin, vaan se kertoo Metatavun rajapinnassa suoritettavan lambda-funktion.

Alla olevan kuvan 7 elementti palauttaa dataa, joka haetaan ”Won Deals”-otsikon alle ja sen data haetaan rajapinnasta diileistä, jonka status on won.

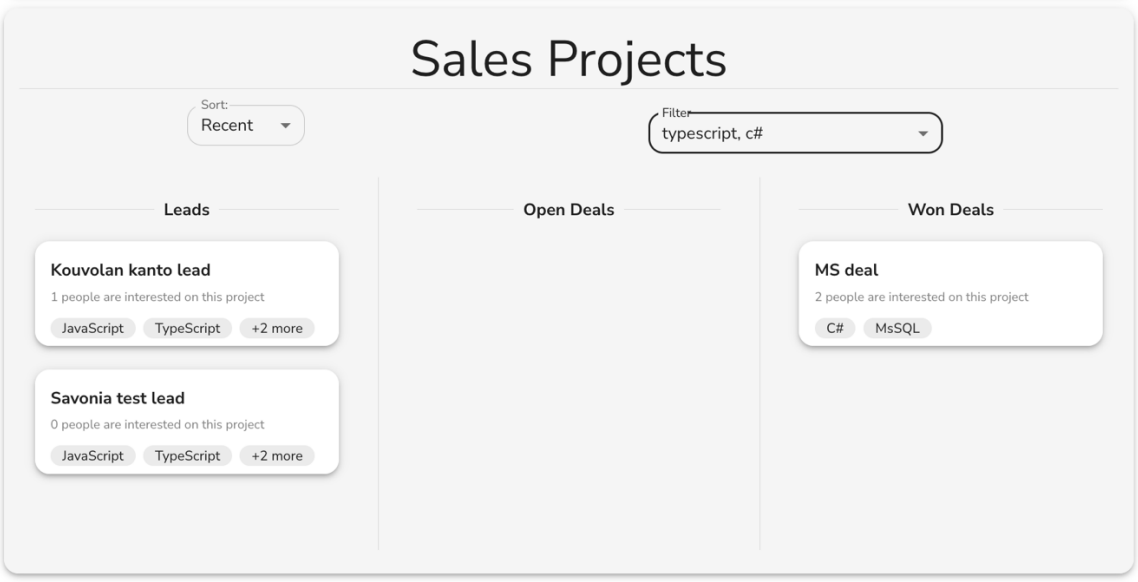
```
<Grid item xs={12} sm={12} md={3.5}>
    <ProjectColumn title='Won Deals' rowtype='dealswon' sorting={sort} usedTech={usedTechList} />
</Grid>
```

KUVA 7. Sovelluksen pääelementin pylväiden uudelleenkäytettävä koodi.

Hakuasetukset koostuvat lajittelusta sekä seulonnasta. Käyttäjä pystyy lajittelemaan projekteja uusimman tai vanhimman mukaan. Seulonnassa käyttäjä voi valita haluamansa kielen tai teknologian, joita vastaavat projektit näkyvät listoissa. Seulassa voidaan käyttää useita kieliä. Tällöin se hakee ne projektit, joissa on käytetty vähintään yhtä valituista teknologioista. Halutun teknologian klikkaaminen päivittää luettelon automaattisesti. Kuvassa 8 esitetään seulontavalikko ja kuvassa 9 seulonnan kriteerien täyttävät projektit.



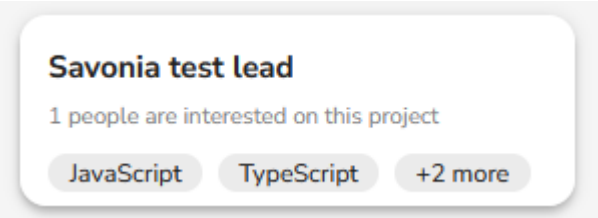
KUVA 8. Projektien seulominen käytettyjen tekniikoiden mukaan.



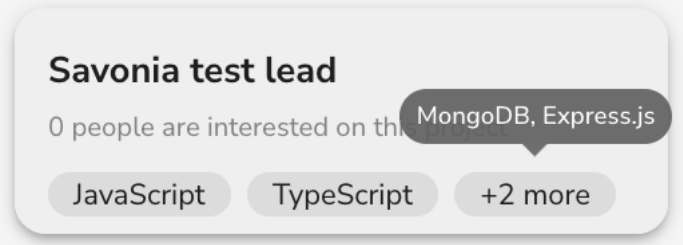
KUVA 9. Projektit, joissa käytetty joko TypeScript tai C# kieltä.

5.4 Yksittäinen projekti sen tarkempi data

Sovelluksessa yksittäinen projekti kasataan kuvan 10 mukaiseen Card-elementtiin. Elementti koostuu projektin otsikosta, kiinnostuneiden lukumäärästä sekä siinä käytetyistä teknologioista. Mikäli projektissa käytetään useampaa kuin kahta teknologiaa, niiden nimet näkee viemällä kursorin ”+n more” elementin päälle, kuten kuvassa 11 havainnollistetaan. Tähän Card-elementtiin haetaan data kyseisen projektin id:tä käyttämällä. Projekti-id liitetään korttiin linkkinä sen luonnin yhteydessä.

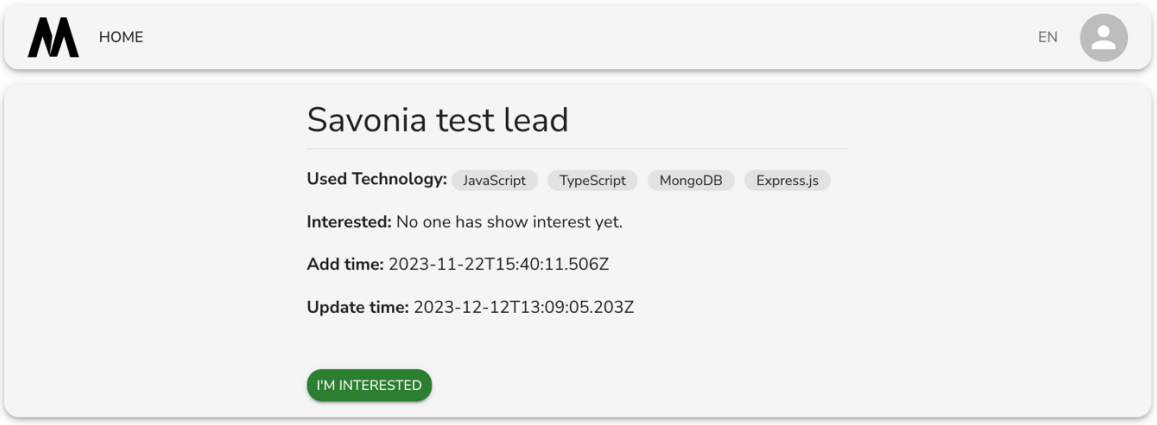


KUVA 10. Esimerkki projektikortista.



KUVA 11. Piilotetut teknologiat näkyvät "+2 more" elementin pälle mentäessä.

Projektin korttia painamalla avautuu uusi sivu, jossa näkyy enemmän dataa kyseisestä projektista kuvan 12 mukaisesti. Sivulla on myös uutena elementtinä nappi, jota painamalla käyttäjä pystyy ilmoittamaan kiinnostuksensa kyseiseen projektiin. Napin painaminen lähettää käyttäjän id:n rajapintaan, jossa se lisätään Pipedriveen luotuun mukautettuun-kenttään. Kuten kuvassa 13 näkyy, nappi muuttaa ulkomuotoaan painalluksen jälkeen.



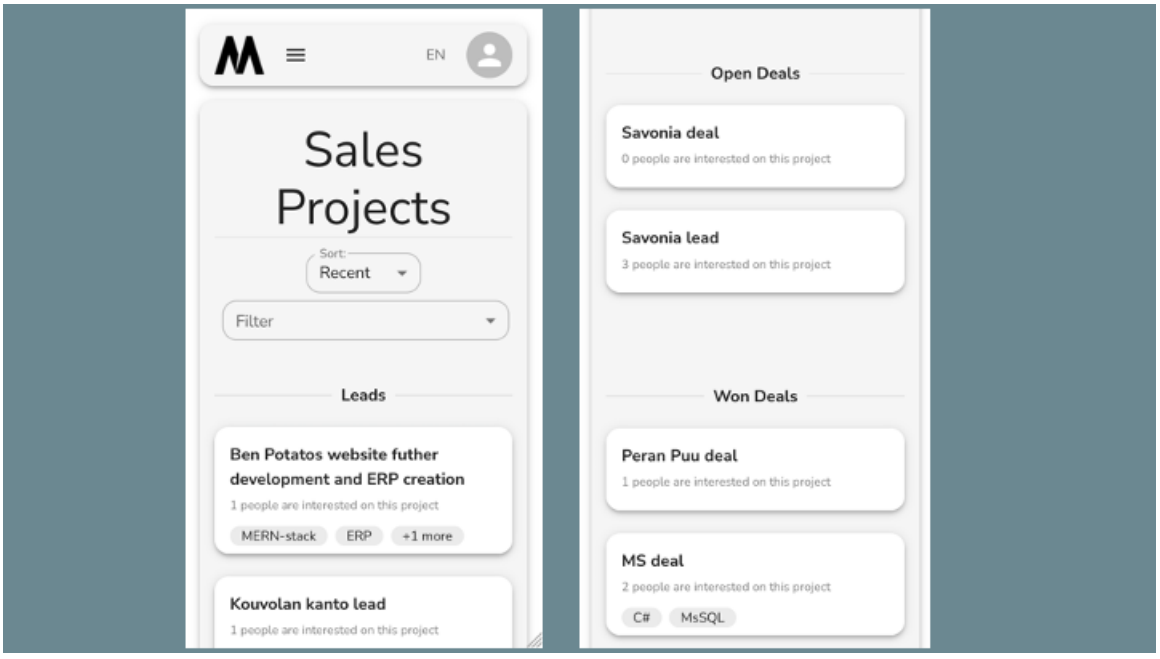
KUVA 12. Projekti tarkemman datan ikkuna ennen ilmoittautumista.

KUVA 13. Projekti tarkemman datan ikkuna ilmoittautumisen jälkeen.

Projektin latautuessa sovellus käy läpi id:t ja hakee niihin liitetyt nimet yrityksen tiedoista. Nämä nimet näytetään projektin tarkemmissa tiedoissa, kuten alla olevasta kuvasta nähdään.

5.5 Sovelluksen käyttöliittymä mobiililaitteilla

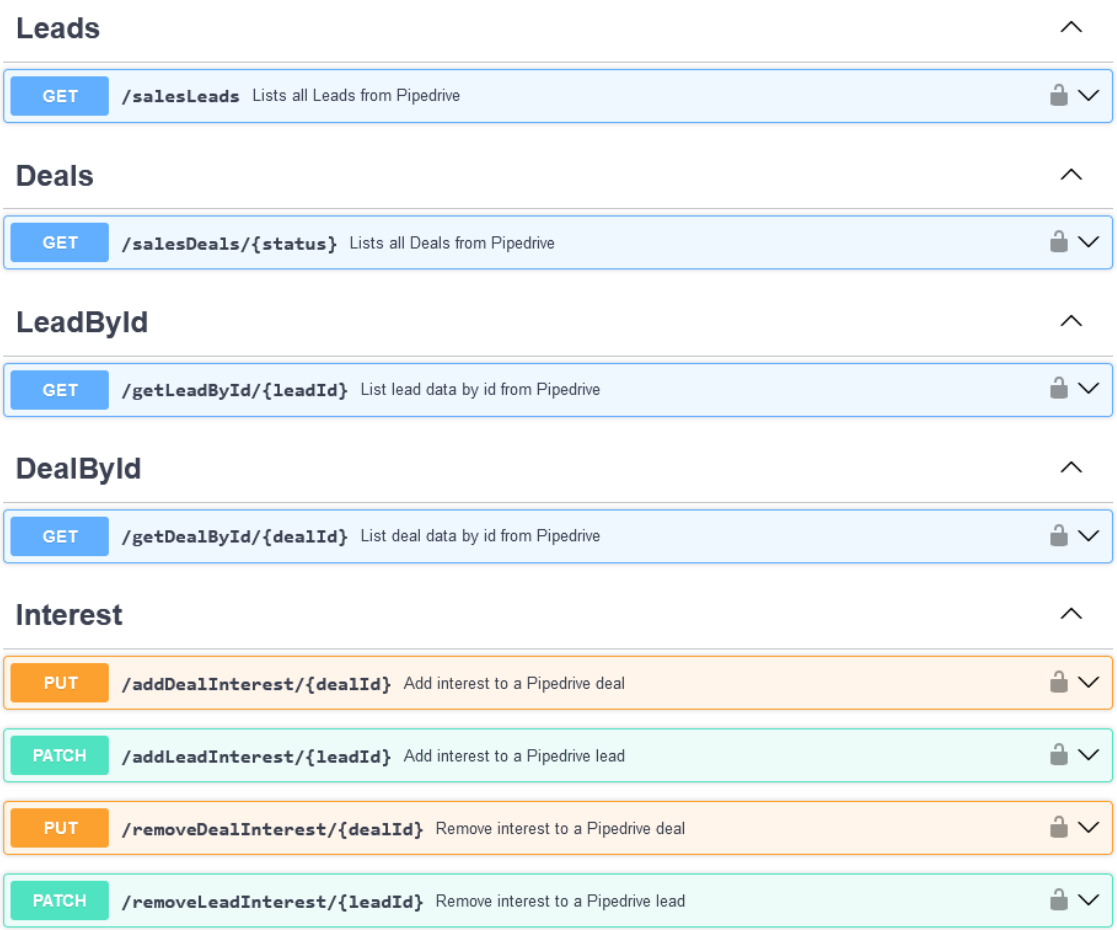
Opinnäytetyön toimeksiantaja ei vaatinut, että käyttöliittymään täytyisi olla mobiiliystävällinen. Yritin silti tehdä käyttöliittymästä mahdollisimman hyvän käyttää kaikilla laitteilla. Kuten kuvassa 14 on havainnollistettu, mobiilikäyttöliittymässä kategoriat ovat aseteltu päällekkäin. Sen lisäksi elementit skaalautuvat näytön leveyden mukaisesti.



KUVA 14. Sovelluksen päänäkymä iPhone 12 näytöllä.

5.6 Sovelluksen rajapinta

Sovelluksen rajapinnan toteuttamiseen käytin Serverless framework:ia. Rajapinta käyttää mallinaan swagger.yaml tiedostoon kirjoitettua rajapinnan kuvausta. Rajapintaa ajetaan Amazonin AWS Lambda -palvelimelta. Lopuksi frontendiin generoitiin Swaggerin generate-ominaisuutta käyttämällä olioiden rakenteet sekä hakufunktiot, joita käytetään kutsumaan serverless palvelinta. Alla olevassa kuvassa 15 näkyy Swagger:in luoma dokumentti rajapinnan päätepisteistä.



KUVA 15. Rajapinnan päätepisteet Swagger editorissa.

5.7 Opinnäytetyön integrointi

Käytin opinnäytetyön koodin tallentamiseen ja hallinnointiin Githubia, johon tallensin sovelluksen eri versioita. Päätimme yrityksen kanssa siirtyä ensimmäisen viikon jälkeen kehittämään sovellusta suoraan heidän olemassa oleviin GitHub repositoreihin, mikä helpotti sovelluksen kehittämistä ja testaamista. Tämä poisti lopusta myös vaivalloisen työn koodin integroimiseen.

6 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli kehittää yritykselle verkkosovellus, joka hakee dataa yrityksen Pipedrive tililtä ja näyttää sitä heidän verkkosovelluksessansa. Lisäksi sovelluksessa tuli olla ominaisuus, jolla yksittäinen työntekijä pystyy osoittamaan kiinnostuksena haluamaansa projektiin. Projektin edetessä alkuperäiset suunnitelmat muovautuivat nykyisille paikoilleen. Tällainen oli esimerkiksi käytetyt teknologiat, joka oli vain yhtenä ehdotuksena projektin alussa.

Sovelluksen UI-puoli liitettiin Metatavun ”metatavu-home” -projektiin, joka on tarkoitus ottaa heillä sisäiseen käyttöön tulevaisuudessa. Rajapinta liitettiin home-lambda -projektiin ja sitä kuvailemaan luotu swagger.yaml -tiedosto home-lambdas-api-spec projektiin. Sovelluksen integrointi suoraan tuleviin toimintaympäristöihin poistaa yritykseltä paljon päänvaivaa.

Projekti sujui hyvin ja suurimmilta ongelmilta vältyttiin. Pieniä ongelmakohtia olivat projekteista kiinnostuneiden käyttäjä-id:n tallentaminen, sillä Pipedrive ei tarjonnut mahdollisuutta tallentaa dataa listaan tai taulukkoon. Tästäkin ongelmasta selvittiin. Jos olisin tehnyt jotakin toisin, olisin aloittanut ohjelmistokehitysvaiheen aiemmin, jotta olisin mahdollisesti voinut saada opinnäytetyöni valmiiksi jo ennen vuodenvaihdetta.

Projekti oli minulle haastava, sillä siinä käytettiin paljon sellaista teknologiaa, mitä en ollut aikaisemmin käyttänyt. Näitä olivat esimerkiksi TypeScript, Serverless, sekä Swagger. Nämä hyvin keskeisissä rooleissa olevat tekniikat vaikuttivat alkuun pelottavilta ja vaikeilta oppia. Jälkikäteen katsottuna kaikki vaikuttavat nyt hyvin yksinkertaisilta ja helpoilta. Sain siis paljon lisää kokemusta ohjelmistokehityksestä ja opin uusia suuressakin käytössä olevia teknologioita.

Sovelluksessa näytettävän datan mukauttaminen toimeksiantajan tarpeisiin jää jatkokehitykseen. Toinen jatkokehitys mahdollisuus on voitettujen diilien pylvään muuttaminen, jossa Pipedrive datan sijasta näytettäisiin projektidataa toimeksiantajan projektienhallinta-sovelluksesta.

LÄHTEET

Asthanan, Abhinav 2021. How we built Postman - the product and the company. Postmanin blogi. 5.4.2021. https://blog.postman.com/how-we-built-postman-product-and-company/. Viitattu 17.10.2023.

Cloudflare julkaisuaika tuntematon. Verkkojulkaisu. What is serverless computing? https://www.cloudflare.com/learning/serverless/what-is-serverless/ . Viitattu 17.11.2023.

Dotenv.org julkaisuaika tuntematon. Verkkojulkaisu. https://www.dotenv.org/docs/security/env. Viitattu 30.10.2023.

Github 2023. Verkkojulkaisu. Building Node.js. Päivitetty 22.10.2023. https://github.com/nodejs/node/blob/main/BUILDING.md Viitattu 31.10.2023.

JSON julkaisuaika tuntematon. Verkkojulkaisu. Introducing JSON. https://www.json.org/json-en.html. Viitattu 11.10.2023.

Material UI julkaisuaika tuntematon. Material UI – Overview. Verkkojulkaisu. https://mui.com/material-ui/getting-started/. Viitattu 17.11.2023.

Material UI julkaisuaika tuntematon. Installation. Verkkojulkaisu. https://mui.com/material-ui/getting-started/installation/. Viitattu 17.11.2023.

Meta julkaisuaika tuntematon. Verkkojulkaisu. Meta Open Source. https://opensource.fb.com/projects/react/. Viitattu 17.10.2023.

Mozilla 2023. Verkkojulkaisu. Mdn web docs – HTML: HyperText Markup Language. Päivitetty 18.7.2023. https://developer.mozilla.org/en-US/docs/Web/HTML. Viitattu 30.10.2023.

Mozilla 2023. Verkkojulkaisu. Mdn web docs. Päivitetty 22.06.2023. https://developer.mozilla.org/en-US/docs/Web/CSS. Viitattu 29.10.2023.

Mozilla 2023. Verkkojulkaisu. Mdn web docs. Päivitetty 7.7.2023. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing. Viitattu 11.10.2023.

Mozilla 2023. Verkkojulkaisu. Mdn web docs. Päivitetty 25.9.2023. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Viitattu 04.10.2023.

Mozilla 2023. Verkkojulkaisu. Mdn web docs – API- Päivitetty 8.6.2023. https://developer.mozilla.org/en-US/docs/Glossary/API. Viitattu 17.10.2023.

Mozilla 2023. Verkkojulkaisu Mdn web docs – REST. Päivitetty 8.6.2023. https://developer.mozilla.org/en-US/docs/Glossary/REST. Viitattu 30.10.2023.

Mozilla 2023. Verkkojulkaisu. Mdn web docs – Cross-Origin Resource Sharing. Päivitetty 10.8.2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. Viitattu 17.10.2023.

Nassri, Ahmad 2020. So long, and thanks for all the packages. Npm blogi. 14.4.2020. https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages. Viitattu 17.10.2023.

Npm julkaisuaika tuntematon. About npm. Verkkojulkaisu. https://www.npmjs.com/about. Viitattu 17.10.2023.

Peltomäki, Juha 2020. Node.js, Web palveluiden ohjelmointi. Helsinki BoD – Books on Demand.

Peltomäki, Juha 2017. JavaScript-kieli, Uudet ominaisuudet. Helsinki BoD – Books on Demand.

Peltomäki, Juha 2017. JavaScript-kieli, Uudet ominaisuudet. Helsinki BoD – Books on Demand.

Pipedrive julkaisuaika tuntematon. Verkkojulkaisu. Myyjien kehittämä ja myyjiä varten kehitetty CRM-järjestelmä. Pipedrive. https://www.pipedrive.com/fi/about. Viitattu 04.10.2023.

Pipedrive julkaisuaika tuntematon. Verkkojulkaisu. Custom fields. https://pipedrive.readme.io/docs/core-api-concepts-custom-fields. Viitattu 8.12.2023.

Pipedrive julkaisuaika tuntematon. Verkkojulkaisu. API Reference. https://developers.pipedrive.com/docs/api/v1. Viitattu 4.10.2023.

PortSwigger julkaisuaika tuntematon. Verkkojulkaisu. What is CORS (cross-origin rescource sharing)? https://portswigger.net/web-security/cors. Viitattu 17.10.2023.

Red Hat 2020. Verkkojulkaisu. What is a REST API? Julkaistu 8.4.2020. https://www.redhat.com/en/topics/api/what-is-a-rest-api. Viitattu 30.10.2023.

Swagger julkaisuaika tuntematon. Verkkojulkaisu. Documentation you're your API design. https://swagger.io/solutions/api-documentation/. Viitattu 13.12.2023.

Typescriptlang.org 2023. Verkkojulkaisu. Microsoft. Päivitetty 7.7.2023. https://www.typescriptlang.org/. Viitattu 09.10.2023.

W3C julkaisuaika tuntematon. Verkkojulkaisu. World Wide Web Consortium. https://www.w3.org/People/Raggett/book4/ch02.html. Viitattu 29.10.2023.