jamk.fi

# Frontend-lähtötasoprojektin suunnittelu ja mallitoteutus

Sami Honkasalo

Opinnäytetyö
Elokuu 2020
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

**Kuvailulehti**

| Tekijä(t) | Julkaisun laji | Päivämäärä |
|---|---|---|
| Honkasalo, Sami | Opinnäytetyö, AMK | Elokuu 2020 |
| | Sivumäärä<br>44 | Julkaisun kieli<br>Suomi |
| | | Verkkojulkaisulupa myönnetty: x |

Työn nimi

**Frontend-lähtötasoprojektin suunnittelu ja mallitoteutus**

Tutkinto-ohjelma

Insinööri (AMK), Tieto- ja viestintätekniikka

Työn ohjaaja(t)

Jani Immonen, Marko Rintamäki

Toimeksiantaja(t)

Devecto Oy


Tiivistelmä

Opinnäytetyön tavoitteena oli perehtyä frontend-kehittäjän tärkeisiin ominaisuuksiin ja ammatillisiin taitoihin. Näiden avulla suunniteltiin lähtötasoprojekti, jota toimeksiantaja pystyy hyödyntämään uusien työntekijöiden rekrytointiprosessissa, sekä perehdytyksessä. Toinen tavoite oli lähtötasoprojektin tehtävänannon mukaisen esimerkkiprojektin suunnittelu ja toteutus.

Web-kehityksen alalla työskentelevän kehittäjän tärkeitä ominaisuuksia ja taitoja arvioitiin yhdessä projektiryhmän ja toimeksiantajan web-kehitykseen erikoistuneen tiimin kanssa. Mallitoteutus projektista tehtiin hyödyntäen nykyaikaista erittäin laajassa käytössä olevaa React-kirjastoa.

Työn tuloksena toimeksiantaja sai määrittelydokumentaation ja tehtävänannon, jonka avulla lähtötasoprojektia voidaan hyödyntää uuden työntekijän rekrytointi- ja perehdytysprosesseissa. Näiden lisäksi toimeksiantaja sai käyttöönsä mallisuorituksen täydestä lähtötasoprojektista, jota voidaan hyödyntää tulevien toteutusten arvioinnissa.

Lähtötasoprojektin avulla toimeksiantaja pystyy kehittämään ja tehostamaan frontend- ja web-kehittäjien rekrytointia. Perehdytyskäytössä lähtötasoprojekti antaa uudelle työntekijälle hyvän keinon tutustua tulevien projektien tyyliin ja sitä kautta kehittymään heti uransa alkuvaiheilla.



Avainsanat (asiasanat)

Frontend, Web-kehitys, React


Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

**Description**

| Author(s) | Type of publication | Date |
|---|---|---|
| Honkasalo, Sami | Bachelor's thesis | August 2020 |
| | | Language of publication: Finnish |
| | Number of pages<br>44 | Permission for web publication: x |

Title of publication

**Designing and implementing a front-end starting project**

Degree programme

Information and Communications Technology engineering

Supervisor(s)

Jani Immonen, Marko Rintamäki

Assigned by

Devecto Oy


Abstract

The purpose of the thesis was to get familiar with the important qualities and work life skills of a front-end developer. These were used to design a starting project that could be used by the assignee Devecto Oy during the recruitment and introduction of a new employee. Another purpose for the thesis was to design and implement an example of the project, using the designed specification documents.

The qualities and skills required from a developer working on the field of web development were evaluated together with the project group and a team of web developers working for the assignee. The example implementation was developed using the modern widely used React-library.

As a result of the work the assignee received the specification documentation and the assignment, which can be used to utilize the starting project during a recruitment process or an introduction process of a new employee. The assignee also received an example implementation of the whole starting project, which can be used when evaluating future implementations of the starting project.

With the starting project the assignee can improve and enhance the recruitment of future front-end and web developers. In introduction use, the starting project can give the new employee a good chance to familiarize themselves with the style of their future projects and through that they can improve themselves right at the start of their career.



Keywords/tags (subjects)

Front-end, Web development, React


Miscellaneous (Confidential information)

# Sisältö

**1 Johdanto ..... 7**

1.1 Opinnäytetyön tarkoitus ja tavoitteet ..... 7

1.2 Opinnäytetyön toimeksiantaja ..... 7

**2 Lähtötasoprojekti ..... 8**

2.1 Rekrytointiprojekti ..... 9

2.2 Perehdytysprojekti ..... 11

**3 Frontend ..... 12**

3.1 Suosituimmat frontend-teknologiat ..... 13

**4 Lähtötasoprojektin tehtävänanto ..... 18**

4.1 Perehdytysprojektiin valitut osaamisalueet ..... 19

4.1.1 Toimeksiantajan projektitoiminnassa vastaan tulleet osaamisalueet ..... 20

4.1.2 Yleisiä frontend-kehittäjän tärkeitä ominaisuuksia ..... 20

**5 Lähtötasoprojektin mallitoteutus ..... 21**

5.1 Sovelluksen rakenne ja asettelu ..... 23

5.2 Karttanäkymä ..... 29

5.3 Listanäkymä ..... 32

5.4 Yksittäisen metsäkoneen näkymä ..... 35

5.5 Tunnistautumisnäkymä ..... 37

**6 Pohdinta ..... 41**

**Lähteet ..... 44**

**Kuviot**

Kuvio 1. Devecto Oy:n logo ..... 8
Kuvio 2. FizzBuzz-tehtävän ratkaisu ..... 10
Kuvio 3. FizzBuzz-tehtävän esimerkkituloste ..... 10
Kuvio 4. Palindromitehtävän malliratkaisu ..... 11
Kuvio 5. Palindromitehtävän esimerkkituloste ..... 11
Kuvio 6. Fullstack-kaavio ..... 13
Kuvio 7. Suosituimpien frontend-teknologioiden Google-hakukerrat ..... 15
Kuvio 8. Suosituimpien frontend-teknologioiden maantieteellinen suosio ..... 15
Kuvio 9. Suosituimpien frontend-teknologioiden suosio Suomessa ..... 16
Kuvio 10. Suosituimpien frontend-teknologioiden latauskerrat ..... 16
Kuvio 11. Angular-syntaksi ..... 17
Kuvio 12. React-syntaksi ..... 17
Kuvio 13. Vue-syntaksi ..... 18
Kuvio 14. Esimerkki käyttäjätarinasta ..... 19
Kuvio 15. Material Design -kirjaston esimerkkikomponentteja ..... 22
Kuvio 16. Sovelluksen työpöytänäkymä ..... 24
Kuvio 17. Sovelluksen mobiilinäkymän navigointivalikko ..... 25
Kuvio 18. Sovelluksen mobiilinäkymä suljetulla navigointivalikolla ..... 26
Kuvio 19. Sovelluksen tumma teema ..... 27
Kuvio 20. Sovelluksen työpöytänäkymä pienennetyllä navigointivalikolla ..... 27
Kuvio 21. Käyttäjän tunnistautumisen React Context ..... 28
Kuvio 22. Sovelluksen karttanäkymä ..... 29
Kuvio 23. Metsäkoneiden virhetilataso ..... 30
Kuvio 24. Mapbox-kartan lisäys sovellukseen ..... 30
Kuvio 25. Karttanäkymä seurantatilassa ..... 31
Kuvio 26. Metsäkoneen reittiviivan päivitys karttaan ..... 32
Kuvio 27. Sovelluksen metsäkonelista ..... 33
Kuvio 28. Metsäkonelistan ohjelmakoodi ..... 34
Kuvio 29. Metsäkonelistan mobiilinäkymä ..... 35
Kuvio 30. Yksittäisen metsäkoneen näkymä ..... 36
Kuvio 31. Kuvaajakirjaston käyttö sovelluksessa ..... 37

Kuvio 32. Sovelluksen kirjautumisnäkymä ..... 38
Kuvio 33. Kirjautumisnäkymän toteutus ..... 39
Kuvio 34. Kirjautumisnäkymän virheilmoitus ..... 39
Kuvio 35. Metsäkoneen yksittäisnäkymä tunnistautuneelle käyttäjälle ..... 40
Kuvio 36. Kirjautumislomakkeen yksikkötesti ..... 41

# 1 Johdanto

## 1.1 Opinnäytetyön tarkoitus ja tavoitteet

Web-kehitys on suuressa, jatkuvasti kasvavassa osassa ohjelmistoalan yrityksen liiketoimintaa, jonka seurauksena kyseisiin työtehtäviin rekrytoidaan jatkuvasti lisää henkilöstöä. Rekrytointiprosessin tuomien haasteiden jälkeen kohdataan usein lisähaasteita uuden työntekijän perehdytyksessä ja koulutuksessa.

Opinnäytetyön tavoitteena on tarjota toimeksiantajalle lähtötasoprojekti, jota voidaan hyödyntää tilanteen mukaan osana rekrytointiprosessia ja/tai uuden työntekijän perehdytyksessä ja lähtötason selvittämisessä, jolloin perehdytyksessä ja koulutuksessa voidaan keskittyä olennaisiin asioihin. Tavoitteena on siis tarjota ratkaisua uuden työntekijän rekrytoinnin ja työsuhteen alkuvaiheen tuomiin haasteisiin.

Opinnäytetyön toteutuksen pääpisteinä ovat frontend-suunnittelijan tärkeimpien ammatillisten ominaisuuksien tunnistaminen, niiden perusteella suunniteltu lähtötasoprojekti ja valmis esimerkkitoteutus kyseisestä projektista. Ammattitaidon osalta kriittisten ominaisuuksien tunnistaminen auttaa toimeksiantajaa jo rekrytointiprosessissa ja lähtötasoprojekti tarjoaa mahdollisuuden uudelle työntekijälle testata ja näyttää taitojaan työsuhteen alkuvaiheessa. Uuden työntekijän tekemää lähtötasoprojektia käydään yhdessä läpi sopivan tiimin kanssa, jolloin uuden tekijän vahvuudet ja mahdolliset heikkoudet voidaan tunnistaa, sekä tiimin resurssit voidaan keskittää tarpeelliseksi katsotulla tavalla, esimerkiksi koulutuksen osalta.

## 1.2 Opinnäytetyön toimeksiantaja

Devecto Oy on erityisesti älykkäiden laitteiden ja koneiden ohjelmistosuunnitteluun ja testausjärjestelmiin keskittynyt tuotekehitystalo. Sulautettujen laitteiden ja liikkuvan kaluston järjestelmäkehityksen lisäksi Devecto on erikoistunut web-palvelujen kehittämiseen, sekä koneoppimiseen. (Devecto n.d.)


Kuvio 1. Devecto Oy:n logo

Devecto on perustettu vuonna 2014 ja yrityksen liikevaihto vuonna 2019 oli 6 miljoonaa euroa. Devectolla on töissä 80 henkilöä ja toimipisteitä on Espoossa, Tampereella, Jyväskylässä ja Kajaanissa. Yritys on laajasti henkilöstön omistuksessa. (Devecto Oy rekisteritiedot n.d.) (Devecto n.d.)

# 2 Lähtötasoprojekti

Lähtötasoprojektin pääideana on se, että sitä voidaan hyödyntää tarpeiden mukaan rekrytointivaiheessa, sekä myös rekrytoinnin jälkeen uuden työntekijän lähtötason selvittämisessä ja perehdyttämisessä. (Mähönen 2020.)

Lähtötasoprojekti on laaja kokonaisuus, josta voidaan ottaa pienempi osa rekrytointia varten, jotta haaste ei ole liian kuormittava mahdolliselle työntekijälle. Perehdytystarkoituksessa käytettävää haastetta voidaan pitää laajempana kokonaisuutena, josta tekijä voi halutessaan tehdä kaikki mahdolliset osa-alueet. Rekrytointiprojektin/haasteen arvioitu työmäärä on noin 2–4 tuntia, kun taas perehdytysprojektin kesto voi olla jopa useita työpäiviä, tekijän halukkuudesta ja taitotasosta riippuen. (Mähönen 2020.)

Lähtötasoprojektin teeman ja siinä tarvittavien taitojen valinta tehdään yleensä yrityksessä tehtyjen projektien ja käytettyjen teknologioiden perusteella, jolloin projekti antaa pientä esimakua mahdollisista työtehtävistä, sekä auttaa yritystä tarkemmin tunnistamaan työtehtäviin soveltuvia henkilöitä. (Mähönen 2020.)

Projektin osa-alueet ja haasteet pisteytetään vaikeustason ja arvioidun ajankäytön mukaan, jolloin lopputulosta voidaan arvioida alustavasti pelkästään tekijän omien merkintöjen perusteella, ilman tarkempaa ohjelmakoodin tutkimista. (Mähönen 2020.)

Lähtötasoprojektin suunnittelutyössä on erityisen tärkeää huomioida teknologiarajoitusten minimoiminen, jotta projekti antaisi tekijälle vapauden valita itselleen mieluisimmat teknologiat ja toteutustavat. Teknologiarajoitusten pois jättäminen auttaa myös projektin päivittämisessä ja pitkäikäisyydessä. Ajatuksena on suunnitella aiherajaukseen sopiva geneerinen tehtävänanto, joka tarjoaa tekijälleen alan työtehtäviin sopivia haasteita, ilman turhan tarkkoja rajauksia. (Mähönen 2020.)

## 2.1 Rekrytointiprojekti

Rekrytointiprojekteja käytetään usein ohjelmistoalan työhaastatteluissa potentiaalisen työntekijän osaamisen kartoittamisessa. Niitä käytetään usein vastavalmistuneiden ja harjoittelijoiden palkkauksessa. Yleensä rekrytointiprosessissa käytettävät projektit/haasteet ovat loogista ajattelua testaavia pieniä tehtävänantoja, joiden avulla voidaan tarkastella mahdollisen työntekijän ajattelumallia ja ongelmanratkaisukykyä. (What To Expect From A Pre-Interview Coding Challenge 2017.)

Yksi yleisimmistä rekrytointiprosessissa käytettävistä haasteista on FizzBuzz. FizzBuzz on tehtävä, jossa ohjelma tulostaa näkyville numerot 1-X, mutta numeron ollessa jaollinen kolmella, ohjelma tulostaa tekstin fizz. Numeron ollessa jaollinen viidellä, ohjelma tulostaa buzz. Mikäli numero on jaollinen kolmella ja viidellä, ohjelma tulostaa näytölle fizzbuzz. Muussa tapauksessa ohjelma tulostaa näytölle kyseisen numeron. Kuviossa 2 näkyy FizzBuzz-tehtävän ratkaisu. Kuviossa 3 näkyy ratkaisun esimerkkituloste, kun numerot pyydettiin välillä 0-20. (Perna, 2019)

```
const fizzBuzz = (count: number): void => {
  for(let i = 1; i <= count; i++) {
    // Check if the number is a multiple of 3 and 5
    if(i % 3 === 0 && i % 5 === 0) {
      console.log('fizzbuzz')
    } // Check if the number is a multiple of 3
      else if(i % 3 === 0) {
      console.log('fizz')
    } // Check if the number is a multiple of 5
      else if(i % 5 === 0) {
      console.log('buzz')
    } else {
      console.log(i)
    }
  }
}

// Call the program
fizzBuzz(20);
```

Kuvio 2. FizzBuzz-tehtävän ratkaisu

```
1
2
"fizz"
4
"buzz"
"fizz"
7
8
"fizz"
"buzz"
11
"fizz"
13
14
"fizzbuzz"
16
17
"fizz"
19
"buzz"
```

Kuvio 3. FizzBuzz-tehtävän esimerkkituloste

Muita suosittuja rekrytointihaasteita ovat esimerkiksi pienimuotoinen ohjelma, joka ilmoittaa, onko annettu sana palindromi. Kuviossa 4 näkyy palindromihaasteen esimerkkiratkaisu ja kuviossa 5 näkyy ratkaisun esimerkkituloste. (Perna, 2019)

```
const palindrome = (str: string): string => {
  // Turn the string to lowercase
  str = str.toLowerCase();
  // Reverse input string and return the result
  // Check if the string is the same when reversed
  const result = str === str.split('').reverse().join('');
  return str + (result ? " is a palindrome" : " is not a palindrome");
}

console.log(palindrome("saippuakauppias"));
console.log(palindrome("racecar"));
console.log(palindrome("test"));
```

Kuvio 4. Palindromitehtävän malliratkaisu

```
"saippuakauppias is a palindrome"
"racecar is a palindrome"
"test is not a palindrome"
```

Kuvio 5. Palindromitehtävän esimerkkituloste

## 2.2 Perehdytysprojekti

Perehdytysprojekti on rekrytointiprojektia laajempi kokonaisuus, jonka pääideana on tarjota uudelle harjoittelijalle/työntekijälle sopiva tutustuminen uusien työtehtävien maailmaan ja niiden tuomiin haasteisiin. Perehdytysprojektin toinen tärkeä tarkoitus on näyttää työntekijän uusille tiimiläisille, että mitä uusi tiimiläinen osaa hyvin, sekä

missä on parannettavaa. Perehdytysprojektia voidaan soveltaa esimerkiksi harjoitteluprosessin alkuvaiheessa, ennen varsinaisen projektityöskentelyn aloitusta. (Mähönen, 2020.)

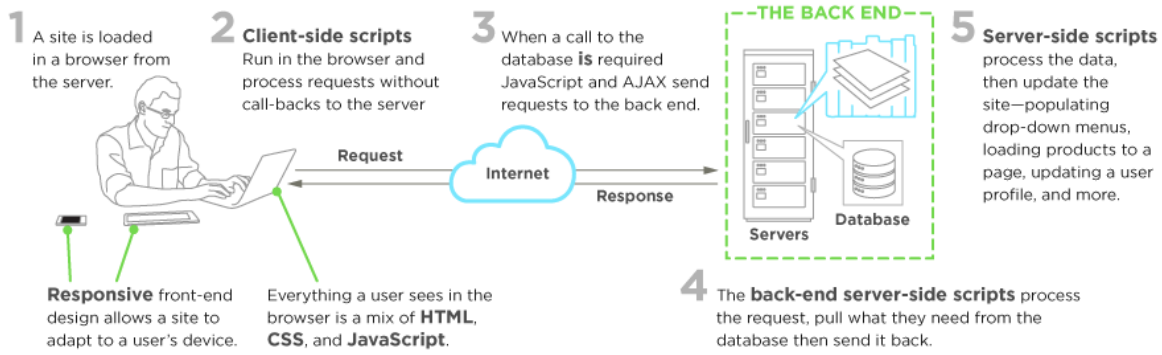
# 3 Frontend

Frontend-kehityksellä tarkoitetaan asiakkaalle/käyttäjälle verkkosivuston tai web-sovelluksen kehittämistä. Kehitys tehdään yleisesti käyttäen HTML-, CSS- ja JavaScript -ohjelmointikieliä. Kehityksen päämääränä on valmistaa sivusto/sovellus, joka tarjoaa loppukäyttäjälle helposti ymmärrettävää ja tärkeää tietoa ja dataa. (What is a front-end developer n.d.)

Kehityksen pääasiallisina haasteina ovat jatkuvasti kehittyvät teknologiat, sekä käyttäjien päivittyvä ja laaja laitekanta. Kehittäjän/suunnittelijan tulee varmistaa, että tuote on käytettävissä kaikilla nykyaikaisilla laitteilla, selaimilla ja käyttöjärjestelmillä. (What is a front-end developer n.d.)

Kehityksen lopputuloksena syntyy yleensä verkkosivusto tai web-sovellus, joka on JavaScriptin avulla yhteydessä palvelimella toimivaan backendiin, jota kautta käyttäjä voi nähdä esimerkiksi palvelun tietokantaan tallennettua tietoa ja esimerkiksi muokata kyseistä dataa. (What is a front-end developer n.d.)

Frontend on siis käyttäjälle näkyvä osuus verkkosovelluksessa. Frontend keskittyy enemmän visuaaliseen puoleen, käyttäjäkokemukseen ja näyttää backendistä saatavilla olevaa dataa käyttäjälle, sekä lähettää käyttäjän antamia komentoja backendiin. Kuviossa 6 on esitelty niin sanottu fullstack-kaavio, jossa erotellaan frontend ja backend. (What is a front-end developer n.d.)




Kuvio 6. Fullstack-kaavio

## 3.1 Suosituimmat frontend-teknologiat

Nykyisin frontend-projekteissa käytettävä teknologia valitaan yleensä kolmen suurimman JavaScript-kirjaston/ohjelmistokehyksen väliltä. Kaikki kolme kehystä ovat laajassa suosiossa ja käytössä haastavien, monimutkaisten ja modernien käyttöliittymien rakentamisessa. (Schwarzmüller, 2020)

**Angular**

Angular on Googlen kehittämä ohjelmistokehys JavaScriptille, jota Google käyttää sisäisesti omissa palveluissaan ja sivustoissaan. Angular on kolmesta suosituimmasta paketista laajin ja sisältää kehittäjälle laajimman kirjon käytettäviä työkaluja, ominaisuuksia ja palveluita. Angularin ydintarkoituksena on rakentaa uudelleenkäytettäviä käyttöliittymäkomponentteja, joita yhdistelemällä voidaan rakentaa laajoja käyttöliittymäkokonaisuuksia. (Schwarzmüller, 2020)

**React**

React on Facebookin kehittämä ja käyttämä JavaScript-kirjasto, joka Angulariin verrattuna sisältää itsessään ainoastaan keinon sisällön näyttämiseen ja manipulointiin. Angularin tapaan myös React perustuu komponenttien kasaukseen ja niiden yhdistelyyn. Suurin osa yleisistä/tärkeistä ominaisuuksista, kuten esimeriksi lomakkeiden

hallinta, hoidetaan yhteisön/muiden kehittäjien tekemillä työkaluilla ja kirjastoilla. Esimerkiksi Facebook-sivusto on toteutettu Reactilla. (Schwarzmüller, 2020)

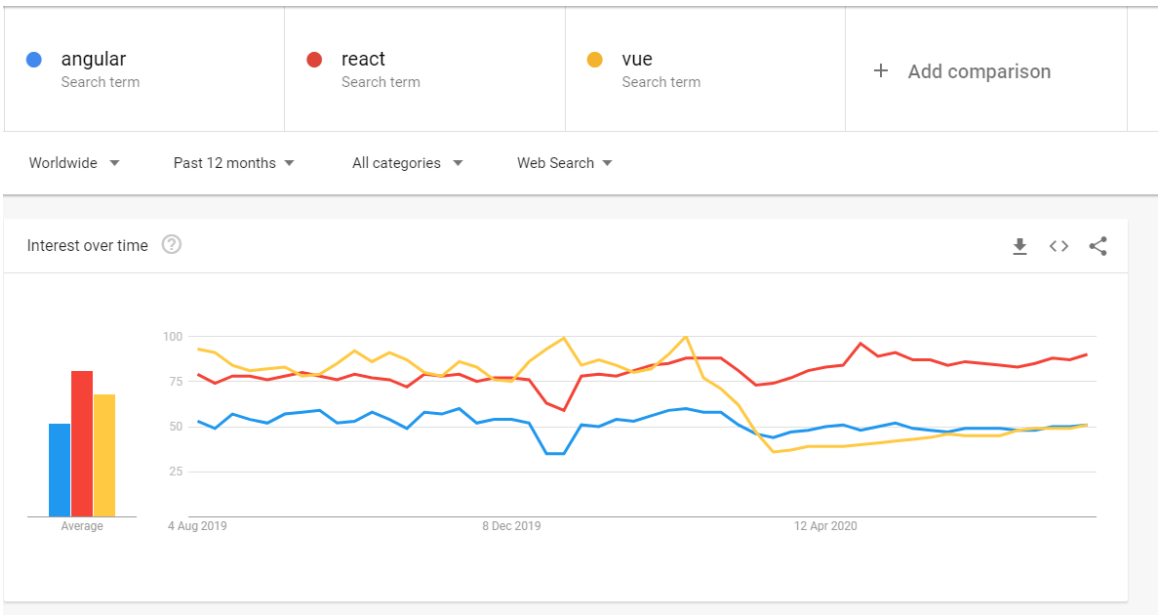
**Vue**

Vue on yhteisöpohjainen ohjelmistokehys, joka asettuu laajuudeltaan Reactin ja Angularin väliin. Vue ei ole perussisällöltään yhtä laaja kuin Angular, mutta se sisältää kuitenkin Reactia enemmän ominaisuuksia ja toimintoja. Myös Vue on ohjelmointityyliltään komponenttipohjainen. Vue on käytössä esimerkiksi suuren Alibaba-verkkokaupan sivuilla. (Schwarzmüller, 2020)

**Angular vs. React vs. Vue**

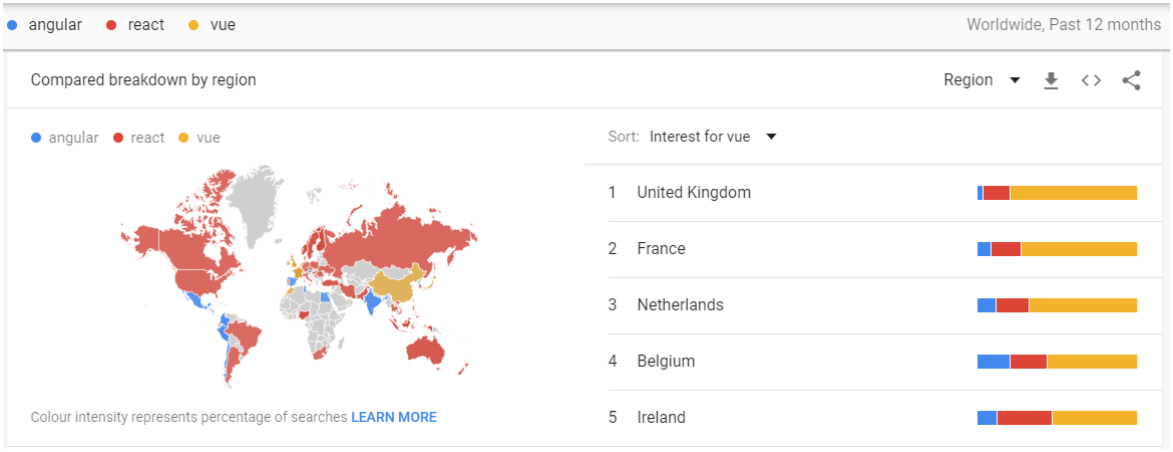
Kaikissa esitellyissä kirjastoissa/kehyksissä on omia hyviä ja huonoja puolia ja valinta tehdään yleensä tapauskohtaisesti tarvittavien ominaisuuksien/toimintojen perusteella. Angularin laajuus on joissakin tapauksissa hyvä, mutta pienemmissä sovelluksissa/sivustoissa niin laajaa kokonaisuutta ei välttämättä tarvita. Reactin suppea koko ei välttämättä toimi laajemmissa sovelluksissa, jos tarvittavia ominaisuuksia joudutaan rakentamaan kehitystyön aikana itse. Mikäli kolmansien osapuolien tekemiä paketteja/kirjastoja joudutaan käyttämään suuria määriä, ylläpidettävien osien lukumäärä kasvaa turhan paljon ja mahdollisia ongelmakohtia saattaa syntyä esimerkiksi yhteensopivuusasioihin liittyen. (Schwarzmüller, 2020)

Pakettien suosion vertailu suoritettiin käyttäen kahta eri keinoa; Google Trends –palvelua, sekä pakettien latausmäärää suositulla npm-pakettihallintasivustolla.

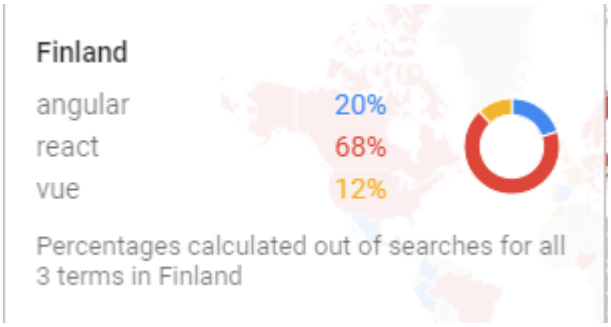



Kuvio 7. Suosituimpien frontend-teknologioiden Google-hakukerrat

Kuviossa 7 näkyy Googlen hakukerrat maailmanlaajuisesti ajalta elokuu 2019 – elokuu 2020. Vuen suosio kuvaajassa selittyy osin kuviossa 8 ilmenevässä aluekohtaisessa suosiossa, josta näkyy, että Vue on erityisen suosittu Kiinassa, jonka suuri asukasmäärä vaikuttaa myös maailmanlaajuiseen suosioon vahvalla painoarvolla.

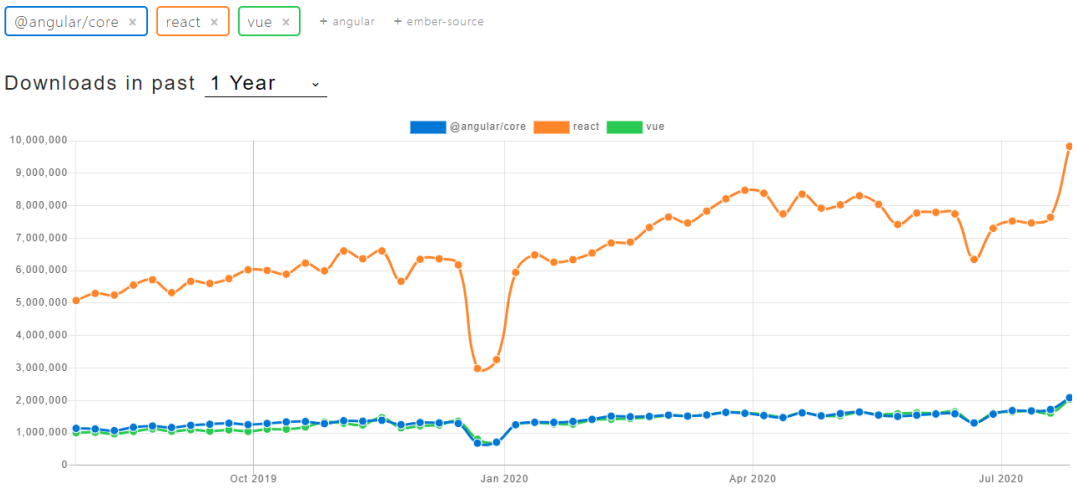



Kuvio 8. Suosituimpien frontend-teknologioiden maantieteellinen suosio




Kuvio 9. Suosituimpien frontend-teknologioiden suosio Suomessa

Kuviossa 9 ilmenee Suomen tilastot Google Trends –palvelussa. Suomessa Reactin osuus on merkittävästi muita suurempi.




Kuvio 10. Suosituimpien frontend-teknologioiden latauskerrat

Pakettien viikoittaiset latauskerrat aikavälillä elokuu 2019 – elokuu 2020 npm-sivustolla näkyvät kuviossa 10, josta näkee selkeästi Reactin suuren suosion kehittäjien keskuudessa.

Kuviossa 11 näkyy tyypillisen Angular-komponentin syntaksi. Kuviossa 12 näkyy vastaavan React-komponentin ohjelmakoodi, ja kuviossa 13 näkyy Vuen vastaava komponentti. Esimerkkikomponentti on lista käyttäjistä, joissa on klikkauksen tunnistava tapahtumakäsittelijä.

```
import { Component, Input, Output, EventEmitter } from '@angular/core'

@Component({
  selector: 'app-user-list',
  template: `
    <ul>
      <li *ngFor="let user of users" (click)="onSelectUser(user.id)">
        {{ user.name }}
      </li>
    </ul>
  `,
})
export class UserListComponent {
  @Output() selectUser = new EventEmitter<{ id: string; name: string }>()
  @Inout() users: { id: string; name: string }[]

  onSelectUser(id: string) {
    this.selectUser.emit(this.users.find(u => u.id === id))
  }
}
```

Kuvio 11. Angular-syntaksi

```
import React from 'react'

export function UserList(props) {
  function userSelectHandler(userId) {
    props.onSelectUser(props.users.find(u => u.id === userId))
  }

  return (
    <ul>
      {props.users.map(user => (
        <li key={user.id} onClick={userSelectHandler.bind(null, user.id)}>
          {user.name}
        </li>
      ))}
    </ul>
  )
}
```

Kuvio 12. React-syntaksi

```
<template>
  <ul>
    <li v-for="user in users" :key="user.id" @click="selectUser(user.id)">
      {{ user.name }}
    </li>
  </ul>
</template>
<script>
  export default {
    props: ['users'],
    methods: {
      selectUser: function(userId) {
        this.$emit(
          'selectUser',
          this.users.find(u => u.id === userId)
        )
      },
    },
  }
</script>
```

Kuvio 13. Vue-syntaksi

Merkittävin ero pakettien syntaksin välillä on se, että React ei käytä perinteistä HTML-koodia, vaan se käyttää erityistä JSX-syntaksia, joka on JavaScriptillä kirjoitettavaa HTML:n perustuvaa ohjelmakoodia. (Schwarzmüller, 2020)

Loppujen lopuksi paketeista on mahdotonta todeta parasta, sillä kaikilla paketeilla on omat vahvuutensa ja heikkoutensa, ja kaikki ovat käytössä web-kehittäjien keskuudessa, sekä suurien yritysten palveluissa, eli tukea ja kirjastoja löytyy kaikkiin tarpeisiin. Paras tapa valita käytettävä kirjasto/paketti, on päättää se projektikohtaisesti, projektin laajuudesta ja kehittäjien omista kokemuksista riippuen. (Shaumik, 2020)

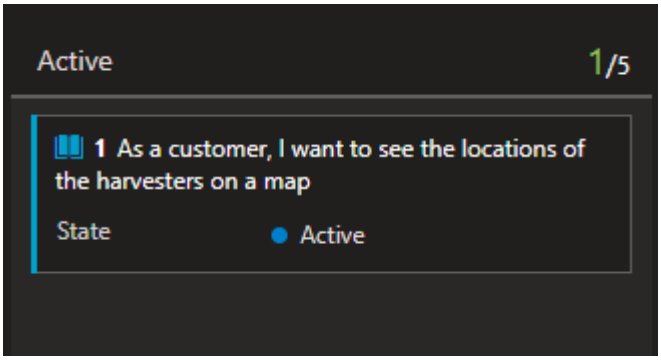
# 4 Lähtötasoprojektin tehtävänanto

Lähtötasoprojektin vaatimusmäärittelyä ja tehtävänantoa lähdettiin muotoilemaan käyttäjätarinapohjaisena. Käyttäjätarinat kirjoitettiin kuvitteellisten loppukäyttäjien ja asiakkaan/tilaajan näkökulmasta. Käyttäjätarinoiden avulla tehtävänantoa jaettiin

pienempiin osakokonaisuuksiin, joista tekijä voi valita omiin taitoihinsa sopivat haasteet.

Lähtötasoprojektin teemaan sopivia haasteita ja käyttäjätarinoita mietittiin ja pisteytettiin frontend-kehittäjille tärkeiden ominaisuuksien/taitojen pohjalta. Lähtötasoprojektin teemaksi/aiheeksi valittiin projektiryhmän ja toimeksiantajan yhteispäätöksenä kuvitteellisen metsäkoneyrityksen laitteiston seuranta- ja hallintasovellus.

Sovellukselta vaadittuja ominaisuuksia ja toiminnallisuuksia mietittiin yhdessä toimeksiantajan kanssa. Valinnat perustuivat aiemmissa projekteissa vastaan tulleisiin toiminnallisuuksiin ja toimeksiantajan edustajan kokemuksiin vastaavien projektien osalta. Sovellusta käytettäisiin ns. hallintasivustona, jossa käyttäjä voisi nähdä metsäkoneiden sijainnit reaaliajassa, sekä muuta koneesta saatavaa mittausdataa. Datan luennan lisäksi sovelluksen kautta tulisi pystyä muuttamaan esimerkiksi backendillä generoitavien hälytysten raja-arvoja. Kuviossa 14 näkyy esimerkki projektiin suunnitellusta käyttäjätarinasta.




Kuvio 14. Esimerkki käyttäjätarinasta

## 4.1 Perehdytysprojektiin valitut osaamisalueet

Frontend-kehittäjälle tärkeitä ominaisuuksia pohdittiin yhdessä toimeksiantajan ja projektiryhmän keskuudessa. Projektiin liittyvät osaamisalueet ja haasteet jaettiin kahteen eri osioon. Ensimmäiseen osioon mietittiin ensisijaisesti toimeksiantajan

yleisen projektitoiminnan kannalta tärkeitä asioita, eli pääasiassa muissa projekteissa ilmenneitä haasteita. Toiseen osioon mietittiin yleisesti web-kehittäjälle tärkeitä osaamisalueita ja ominaisuuksia.

### 4.1.1 Toimeksiantajan projektitoiminnassa vastaan tulleet osaamisalueet

Toimeksiantajan projektitoiminnan kannalta tärkeitä osaamisalueita ovat esimerkiksi yleisten/suosittujen valmiiden kirjastojen käyttö. Kirjastojen käyttäminen osoittaa muun muassa tekijän kykyä sisäistää kolmannen osapuolen tekemää dokumentaatiota ja niiden soveltamista.

Ensimmäiseksi käyttäjätarinaksi valittiin kuvitteellisten metsäkoneiden näyttäminen kartalla. Tehtävänanto kirjoitettiin käyttäjätarinamaisesti muotoon: “Käyttäjänä haluan nähdä metsäkoneiden sijainnit kartalla reaaliajassa”. Käyttäjätarinaan liittyy kaksi osaamisaluetta; reaaliaikaisen datan käyttö, sekä karttakirjastojen käyttö. Karttakirjastojen käyttöön liitettiin myös muutama vähemmän laaja käyttäjätarina.

Toinen tärkeä osaamisalue on graafi/kuvaajakirjastojen käyttö. Tähän aiheeseen kirjoitettiin aluksi yksi käyttäjätarina; “Käyttäjänä haluan nähdä metsäkoneen öljyn pinnankorkeusmittauksen kuvaajamuodossa”. Kuvaajakirjastojen käytössä vaaditaan lisätaitona myös datakäsittelyä, jotta käytettävä data saadaan kuvaajalle sopivaan muotoon.

### 4.1.2 Yleisiä frontend-kehittäjän tärkeitä ominaisuuksia

Yleisempiä frontend-kehitykseen liittyviä osaamisalueita pohdittiin pääasiassa projektitiimin ja web-kehitykseen erikoistuneen osaston keskuudessa. Tärkeimpänä taitona esiin nousi backendistä saatavilla olevan datan hakeminen ja saadun datan näyttäminen sille sopivalla tavalla.

Muut tärkeät ominaisuudet liittyivät pääasiassa sovelluksen visuaaliseen puoleen ja käytettävyyteen. Tärkeitä aiheeseen liittyviä käyttäjätarinoita olivat esimerkiksi täysi käytettävyys mobiililaitteilla, sekä käyttäjävalinta vaalean ja tumman väriteeman välillä.

Näiden lisäksi tehtävänantoon lisättiin myös käyttäjätarinoita datan muokkaukseen, käyttäjän tunnistautumiseen, valitun väriteeman tallentamiseen, sekä sovelluksen yksikkötestaukseen liittyen.

Myös kolmannen osapuolen avoimen datan käyttöön liittyvä käyttäjätarina lisättiin tehtävänantoon. Pääideana on se, että tekijä osaisi hakea valitsemastaan ulkoisesta palvelusta nimetyn sijainnin, eli kaupungin tai alueen, jokaiselle metsäkoneelle, koneen koordinaatteja hyödyntäen.

Ominaisuuksiin ja osaamisalueisiin liittyvien käyttäjätarinoiden lisäksi tekijälle annetaan vapaus esitellä omaa osaamistaan ja ammattitaitoaan parhaaksi katsomallaan tavalla.

Käyttäjätarinoiden lisäksi lopputuloksesta arvioidaan myös ohjelmakoodin ulkoasua, loogisia ratkaisuja, sekä tekijän suunnitteleman käyttöliittymän ulkoasua ja käytettävyyttä.

# 5 Lähtötasoprojektin mallitoteutus

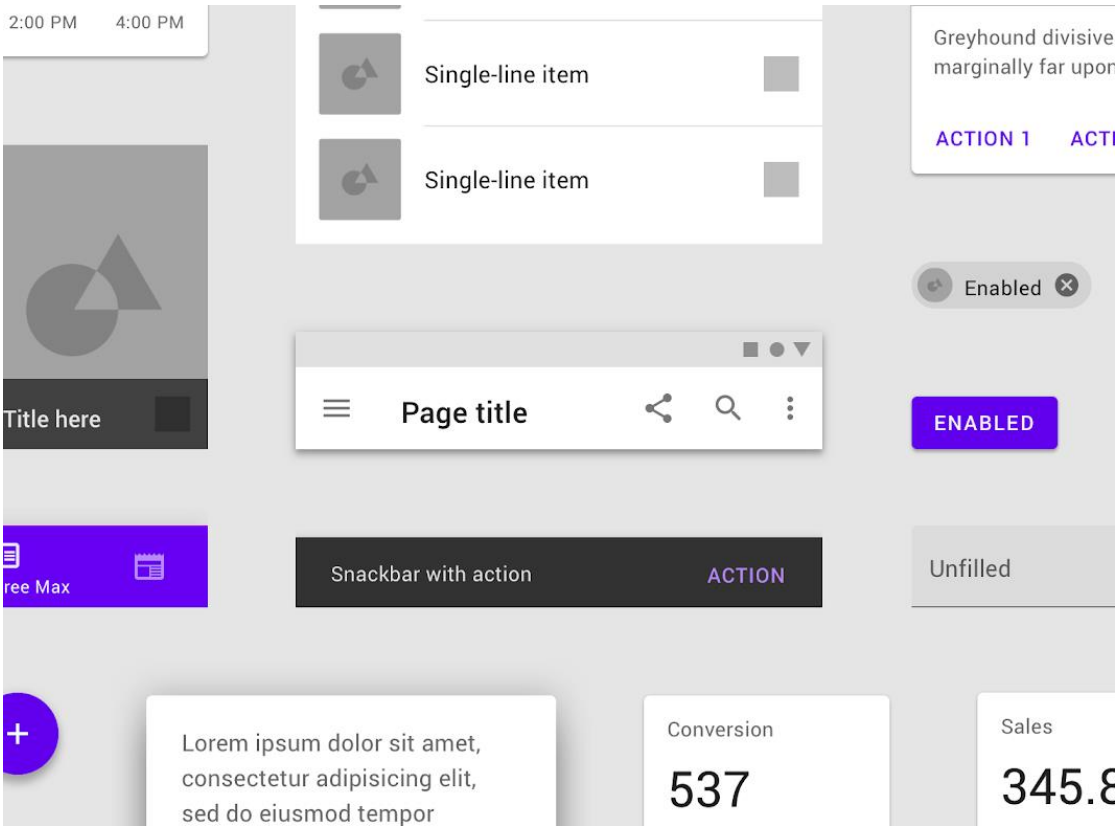
Lähtötasoprojektin mallitoteutusta lähdettiin toteuttamaan tehtävänannossa olevien käyttäjätarinoiden pohjalta. Mallitoteutuksen tavoitteena oli tarjota toimeksiantajalle valmis esimerkki projektista, joka sisältää kaikki tehtävänannossa määritellyt vaatimukset käyttäen sopivia nykyaikaisia teknologioita ja ohjelmointinäkökulmasta katsottuna siistejä ratkaisuja.

Mallisuoritukseen valittiin lisäominaisuutena ja osaamisnäytteenä käytettäväksi Googlen kehittämä Material Design –tyyliohjeistus, joka sisältää Googlen määrittelemiä ulkoasuun ja käytettävyyteen liittyviä ohjeistuksia, sekä malleja.

**Material Design**

Material Design on Googlen kehittämä ulkoasuun ja käytettävyyteen kantaa ottava

tyylikirjasto. Material-järjestelmän tarkoituksena on auttaa korkealaatuisten digitaalisten kokemusten rakentamisessa Android-, iOS-, Flutter- ja web-alustoilla. Material-järjestelmä on saanut inspiraatiota fyysisestä maailmasta ja sen tekstuureista, erityisesti heijastavuuden ja varjostuksen osalta. Material on komponenttipohjaisesti suunniteltu. Kuviossa 15 näkyy esimerkkejä Material-järjestelmän komponenteista. Komponenttien ulkoasun ja toiminnallisuuksien lisäksi Material tarjoaa ohjeistuksia sovelluksen teemaan, käytettyihin väreihin, fontteihin, varjostukseen ja muuhun. (Material Design, n.d.)




Kuvio 15. Material Design -kirjaston esimerkkikomponentteja

Projektissa käytettävät teknologiavalinnat tehtiin ennen varsinaisen käytännön toteutuksen aloitusta. Teknologiavalintoja tehdessä huomioon otettiin projektin kokoluokka, teknologian soveltuvuus projektin tarpeisiin, sekä teknologian yleinen suosio nykyaikaisissa ohjelmistoprojekteissa.

Teknologiavalintakriteerien perusteella projektin pääkirjastoksi ja ohjelmistokehykseksi valittiin JavaScript-kirjasto React. Angular koettiin hieman liian laajaksi projektin kokoluokkaa ajatellessa, ja Vue tuntui markkinaosuudeltaan liian vähän käytetyltä. Reactin lisäksi ohjelmointikieleksi valittiin JavaScriptin sijaan JavaScriptiin pohjautuva TypeScript, joka on jatkuvasti yleistyvä vaihtoehto.

Teknologiavalintojen jälkeen, tehtävänanto käytiin huolella läpi ja sieltä eriteltiin käyttäjätarinat, joiden huomioiminen heti kehitystyön alussa olisi sujuvan toteutuksen kannalta tärkeää. Esimerkiksi teemavalinta tumman ja vaalean teeman välillä oli helpompi toteuttaa, jos sen otti huomioon jo kehitystyön alkuvaiheessa. Myös mobiilikäytettävyys kannattaa ottaa huomioon heti sovelluskehityksen alussa.

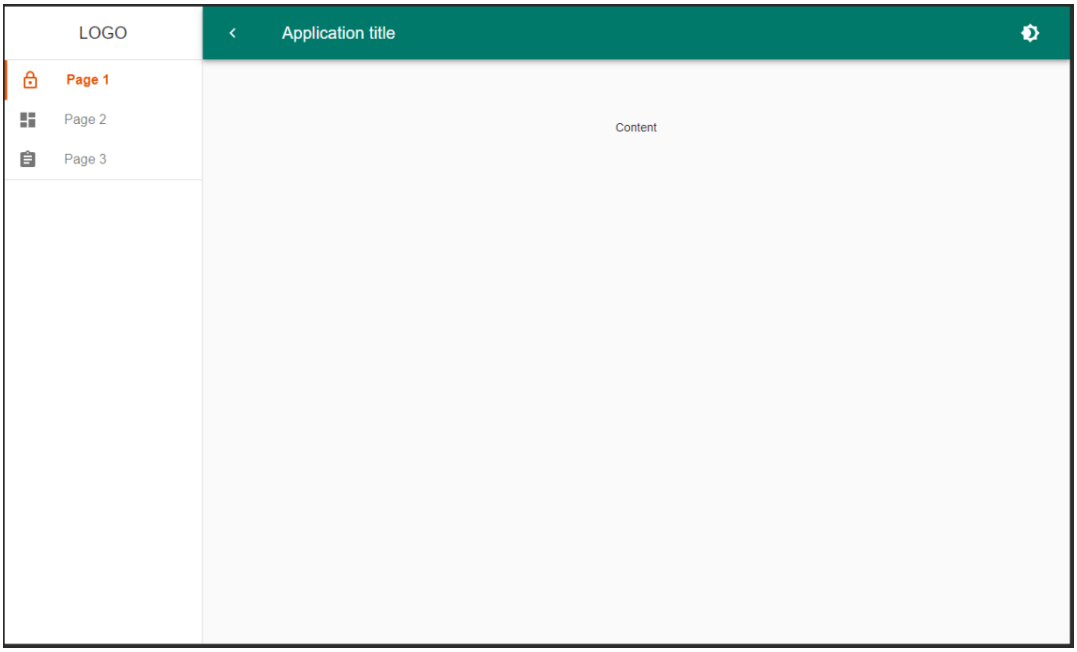
Sovelluksen kehitys toteutettiin suositulla Visual Studio Code –tekstieditorilla, ja versionhallintana käytettiin laajasti käytössä olevaa Git-versionhallintaa. Sovelluksen repository sijaitsi Microsoftin DevOps -ympäristössä. Projektia kehitettiin ketterän kehityksen menetelmiä noudattaen ja etenemistä seurattiin pääasiassa DevOps-ympäristössä sijaitsevan Kanban-taulun avulla.

Sovelluksen versiohallinnan rakenteena käytettiin niin sanottua feature-mallia, jossa sovellukseen luodaan erillinen haara jokaiselle ominaisuudelle, joka yhdistetään pääkehityshaaraan ominaisuuden valmistuttua. Tämä on yleisesti projektitoiminnassa käytössä oleva malli, sillä se mahdollistaa usean kehittäjän samanaikaisen työskentelemisen.

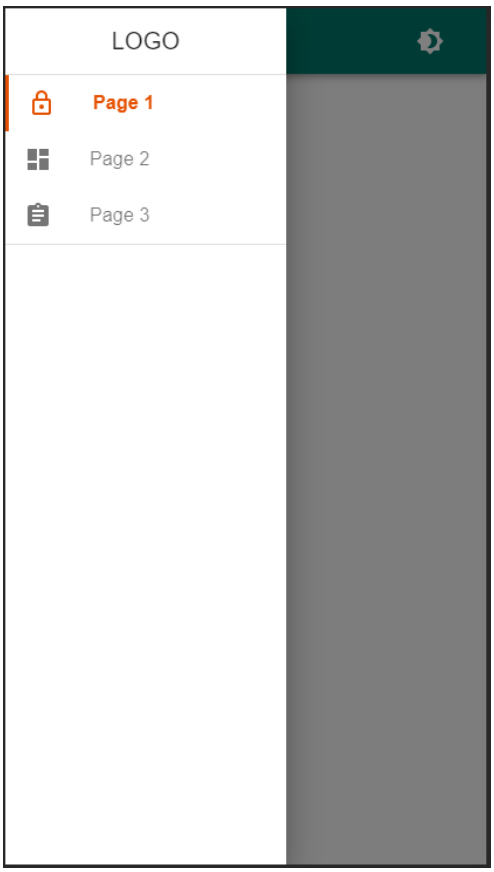
## 5.1 Sovelluksen rakenne ja asettelu

Sovelluksen suunnittelu aloitettiin hyödyntämällä Googlen Material Design –suunnittelukirjaston mallitoteutuksia ja esimerkkejä. Sovellusta lähdettiin suunnittelemaan ns. “hallintapaneelin” tyylillä, jossa pääfokus olisi esitettävällä datalla ja tarvittava informaatio olisi esillä mahdollisimman selkeästi.

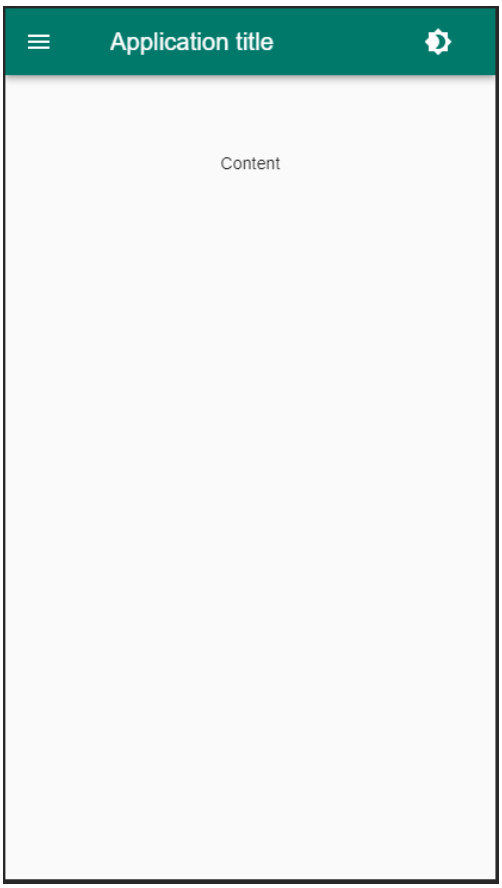
Sovelluksen navigointivalikko asetettiin työpöytäkäytössä kiinteästi vasempaan laitaan (kuvio 16), ja mobiilikäytössä valikko voidaan avata (kuvio 17) ja sulkea tarvittaessa (kuvio 18).




Kuvio 16. Sovelluksen työpöytänäkymä

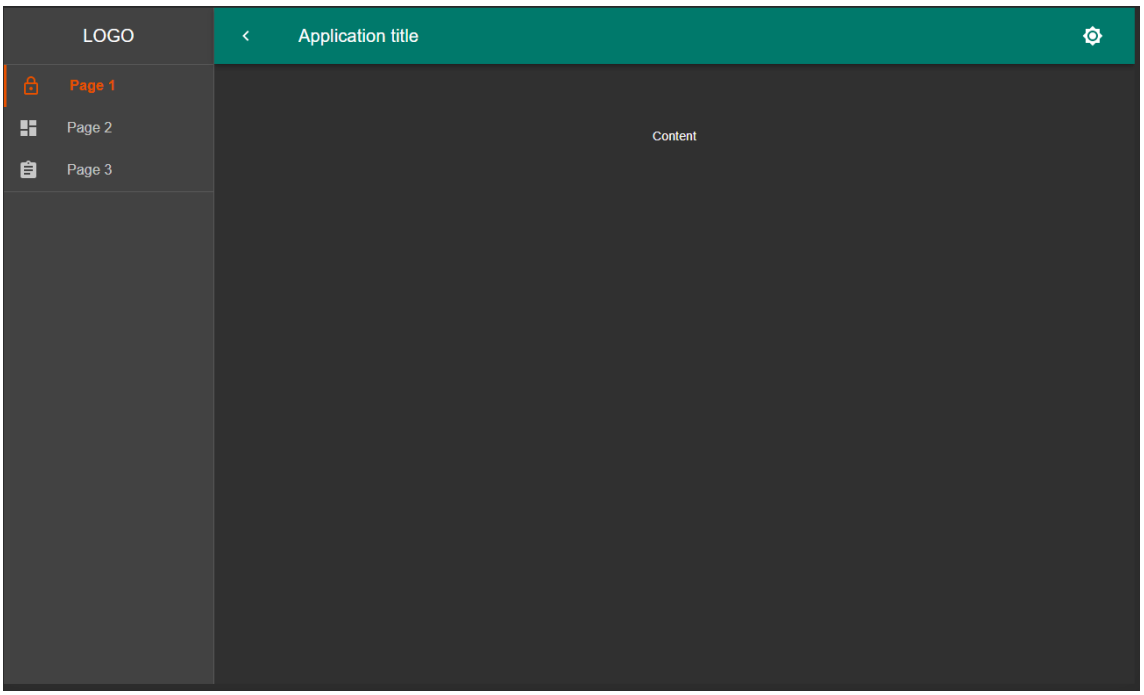



Kuvio 17. Sovelluksen mobiilinäkymän navigointivalikko



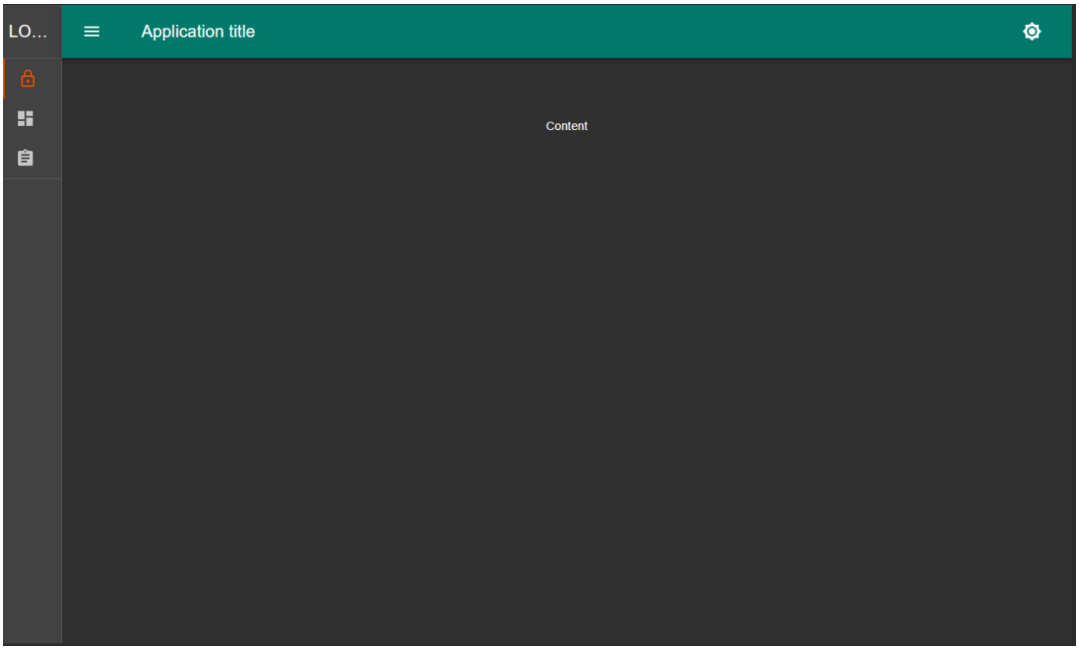

Kuvio 18. Sovelluksen mobiilinäkymä suljetulla navigointivalikolla

Sovelluksen perusrakenteen mukana tehtiin toiminnallisuus myös teeman vaihtamiselle. Teeman vaihto voidaan tehdä yläpalkin oikeasta laidasta. Kuviossa 19 näkyy sovelluksen tumma teema.




Kuvio 19. Sovelluksen tumma teema

Työpöytänäkymään lisättiin vielä erillinen sivupalkin pienennystoiminto, jolloin tärkeän sisällön kokoa voidaan kasvattaa myös isommilla näytöillä. Kuviossa 20 on esitelty työpöytänäkymän navigointipalkki pienennetyssä muodossa.




Kuvio 20. Sovelluksen työpöytänäkymä pienennetyllä navigointivalikolla

Sovelluksen perusrakenteen/pohjan toteutuksen ja manuaalitestauksen jälkeen sovelluksen toiminnallisuuksia alettiin toteuttamaan käyttäjätarinoiden pohjalta.

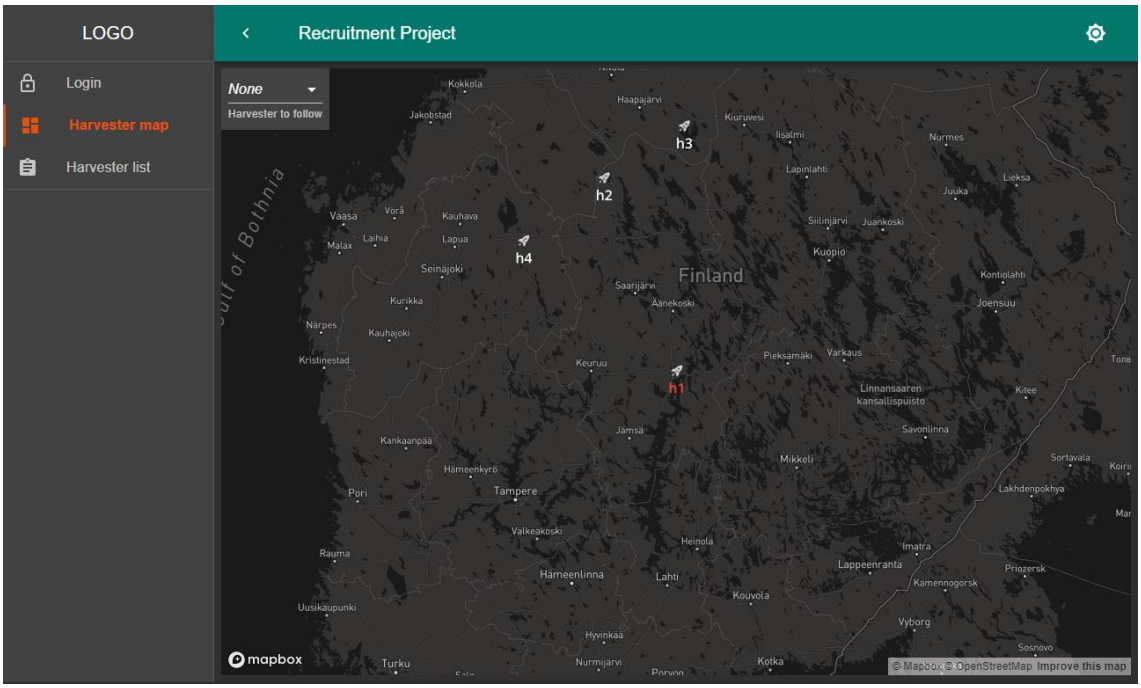
Teeman vaihto ja sivupalkin tilan muuttaminen toteutettiin käyttäen Reactin uutta Context-ominaisuutta, joka mahdollistaa sovelluksessa laajasti käytössä olevien tilojen ja toimintojen jakamisen alakomponentteihin helposti. Context on eräänlainen keskitetty tilahallinta, jonka avulla voidaan myös suorittaa sinne määriteltyjä toiminnallisuuksia, kuten esimerkiksi sivupalkin pienentäminen/suurentaminen. Kuviossa 21 näkyy esimerkki käyttäjän tunnistautumiseen liittyvästä Context-tilasta, joka sisältää tiedot käyttäjän tunnistautumisesta, sekä funktiot, joiden avulla sovellus voi kirjata käyttäjän sisään ja ulos. Kyseisen tunnistautumis-Contextin ideana on se, että käyttäjän kirjautumisen tilaa voidaan tarkastella eri puolilla sovellusta, jonka perusteella voidaan valita, näytetäänkö käyttäjälle tiettyjä osia sovelluksesta.

```
17  const AuthProvider: React.FC = ({ children }) => {
18    const [isLoggedIn, setIsLoggedIn] = useState(false);
19    const [username, setUsername] = useState('');
20
21    // On mount, automatically login if data in localstorage
22    useEffect(() => {
23      const lsData = localStorage.getItem('userData');
24      const lsUserData = lsData && JSON.parse(lsData);
25      if (lsUserData && lsUserData.username) {
26        setIsLoggedIn(true);
27        setUsername(lsUserData.username);
28      }
29    }, []);
30
31    const login = useCallback((user: string) => {
32      setIsLoggedIn(true);
33      setUsername(user);
34      // Set data to localstraoge on login
35      localStorage.setItem('userData', JSON.stringify({ username: user }));
36    }, []);
37
38    const logout = useCallback(() => {
39      setIsLoggedIn(false);
40      setUsername('');
41      // Remove data from localstorage on logout
42      localStorage.removeItem('userData');
43    }, []);
44
```

Kuvio 21. Käyttäjän tunnistautumisen React Context

## 5.2 Karttanäkymä

Sovelluksen pääsivuna toimii karttanäkymä (kuvio 22), josta käyttäjä näkee kaikki käytössä olevat metsäkoneet ja niiden sijainnit. Kartalla olevat raketti-ikonit kuvastavat metsäkoneita ja niiden sijainti päivittyy kartalla reaaliajassa aina, kun metsäkoneiden sijaintitieto muuttuu. Kartalla olevien metsäkoneiden tunnistetieto näkyy kartalla punaisena, jos metsäkoneessa on virhe, esimerkiksi alhainen öljyn pinnankorkeus tai jokin muu mahdollinen vikatila. Virhetila toteutettiin tekemällä karttaan kaksi erillistä kerrosta, joiden välillä metsäkoneita voidaan vaihdella, riippuen virhetilasta. Päätasolla ja virhetasolla erona on ainoastaan metsäkoneen tunnistetiedon tekstin väri. Kuviossa 23 esitellään virhetilatason (layer) lisääminen karttasovellukseen.




Kuvio 22. Sovelluksen karttanäkymä

```
// Also add a layer for error harvesters
if (!m.getLayer('harvesters-error')) {
  m.addLayer({
    id: 'harvesters-error',
    type: 'symbol',
    source: 'harvesters',
    paint: {
      'text-color': '#f44336',
    },
    layout: {
      // get the icon name from the source's "icon" property
      // concatenate the name to get an icon from the style's sprite sheet
      'icon-image': ['concat', ['get', 'icon'], '-15'],
      // get the title name from the source's "title" property
      'text-field': ['get', 'title'],
      'text-font': ['Open Sans Semibold', 'Arial Unicode MS Bold'],
      'text-offset': [0, 0.6],
      'text-anchor': 'top',
    },
    filter: ['==', ['get', 'error'], true],
  });
}
```

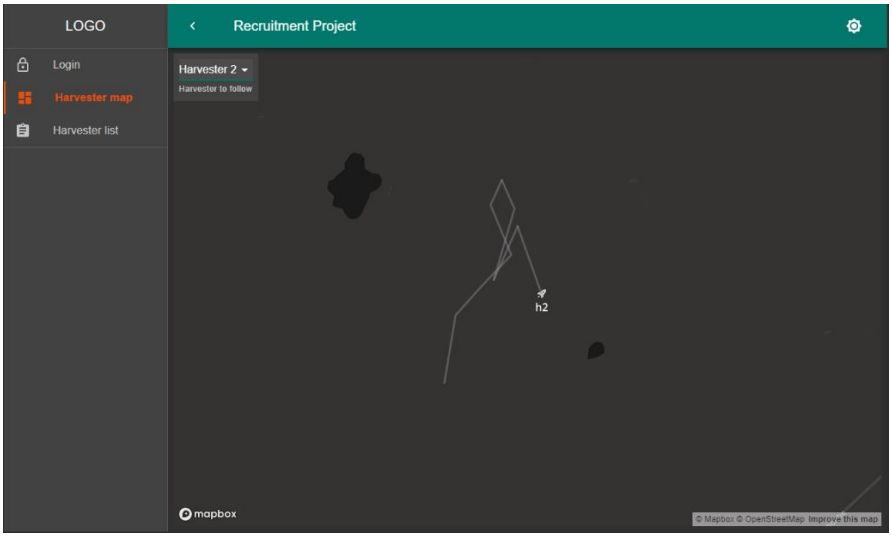
Kuvio 23. Metsäkoneiden virhetilataso

Sovelluksen kartta toteutettiin käyttäen toimeksiantajan muissa projekteissa käytössä olevaa Mapbox-karttakirjastoa, joka hyödyntää OpenStreeMap-karttoja. Kuviossa 24 näkyy ohjelmakoodi, jolla kartta renderöidään päänäkymän latautuessa.

```
// Create map on mount
useEffect(() => {
  mapboxgl.accessToken = process.env.REACT_APP_MAP_API_KEY || '';
  if (mapContainer && mapContainer.current) {
    const m = new mapboxgl.Map({
      container: mapContainer.current,
      style: 'mapbox://styles/mapbox/outdoors-v11',
      zoom: 6,
      center: { lat: 62.2518079, lng: 25.7671327 },
    });
    setMap(m);
  }
}, []);
```

Kuvio 24. Mapbox-kartan lisäys sovellukseen

Metsäkonekartan voi asettaa myös tarvittaessa seurantatilaan (kuvio 25), jolloin kartta seuraa valittua metsäkonetta. Metsäkoneista piirretään myös reittiviiva, viimeisimpien sijaintitietojen perusteella.




Kuvio 25. Karttanäkymä seurantatilassa

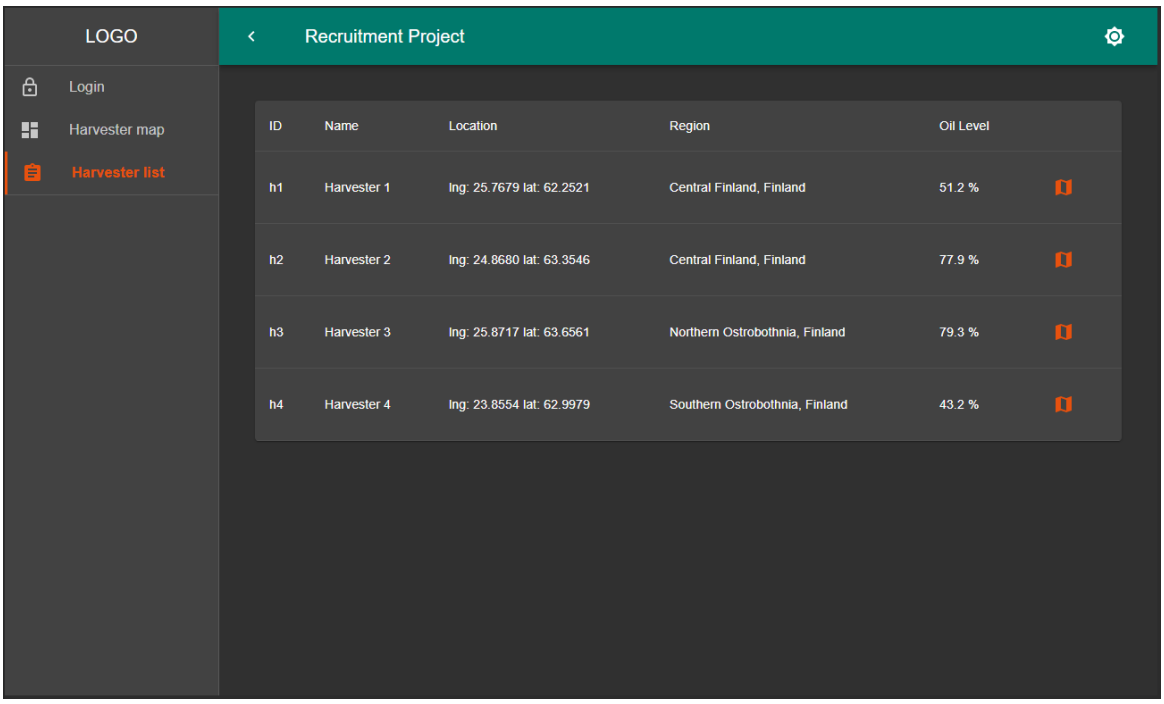
Kuviossa 26 näkyy ohjelmakoodi, jolla metsäkoneen reittiviiva päivitetään metsäkoneen backend-datan päivittyessä. Reittidatasta luodaan ns. GeoJSON-tyyppinen tietue, joka sisältää metsäkoneen aiempia sijaintitietoja.

```
66  const updateHarvesterRoutes = useCallback(
67    ({ map, harvesters }: UpdateRoutes) => {
68      const routeData: GeoJSON.FeatureCollection<GeoJSON.LineString> = {
69        type: 'FeatureCollection',
70        features: [],
71      };
72      // Get the routes source
73      if (map && map.getSource('routes')) {
74        const source: mapboxgl.GeoJSONSource = map.getSource(
75          'routes'
76        ) as mapboxgl.GeoJSONSource;
77        if (source) {
78          // Loop through all harvester routes and update them
79          harvesters.forEach((h) => {
80            const coords = h.route.map((r) => [r.lng, r.lat]);
81            const feature: GeoJSON.Feature<
82              GeoJSON.LineString,
83              GeoJSON.GeoJsonProperties
84            > = {
85              type: 'Feature',
86              properties: {},
87              geometry: {
88                type: 'LineString',
89                coordinates: coords,
90              },
91            };
92            routeData.features.push(feature);
93          });
94          // Set new data
95          source.setData(routeData);
96        }
97      }
98    },
99    []
100 );
```

Kuvio 26. Metsäkoneen reittiviivan päivitys karttaan

## 5.3 Listanäkymä

Kartan lisäksi metsäkoneiden tietoja ja tiloja voi tarkkailla metsäkonelistasta (kuvio 27), josta näkee nopeasti ja helposti kaikkien metsäkoneiden tiedot. Listaan haetaan metsäkoneen sen hetkinen kaupunki/aluesijainti erillisestä kolmannen osapuolen sijaintitietopalvelusta. Metsäkonelistan mobiilinäkymä tehtiin kokonaan erillisenä (kuvio 29), koska työpöytäversio ei skaalaudu taulukkomaisuutensa takia hyvin kapealle näytölle. Metsäkoneen tietojen kohdalla oleva oranssi painike siirtää käyttäjän karttanäkymään ja valitsee automaattisesti kyseisen metsäkoneen automaattisesti seurattavaksi.



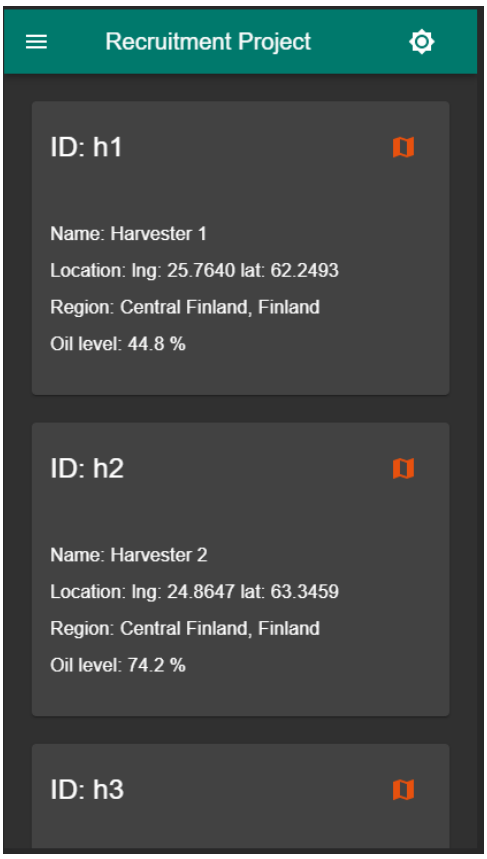

Kuvio 27. Sovelluksen metsäkonelista

Kuviossa 28 näkyy metsäkonelistan ohjelmakoodi. Lista toteutettiin hyödyntäen Material UI –kirjaston taulukkokomponentteja. Taulukkoon luodaan haluttu määrä otsikoita, joiden rinnalle määritellään taulukon soluja.

```
const HarvesterTable = ({
  harvesters,
  loading,
  handleRowClick,
  handleButtonClick,
}: Props) => {
  const classes = useStyles();
  return (
    <TableContainer component={Paper}>
      <Table aria-label="harvester table">
        <TableHead>
          <TableRow>
            <TableCell>ID</TableCell>
            <TableCell>Name</TableCell>
            <TableCell>Location</TableCell>
            <TableCell>Region</TableCell>
            <TableCell>Oil Level</TableCell>
            <TableCell />
          </TableRow>
        </TableHead>
        <TableBody>
          {harvesters.map((h) => {
            return (
              <TableRow
                key={h.id}
                hover
                onClick={(event) => handleRowClick(event, h.id)}
                className={classes.tableRow}
              >
                <TableCell component="th" scope="row">
                  {h.id}
                </TableCell>
                <TableCell>{h.name}</TableCell>
                <TableCell>
                  lng: {h.location.lng.toFixed(4)} lat:{' '}
                  {h.location.lat.toFixed(4)}
                </TableCell>
                <TableCell>{loading ? <LinearProgress /> : h.region}</TableCell>
                <TableCell>{h.oilLevel} %</TableCell>
                <TableCell>
                  <IconButton
                    title="Show on map"
                    aria-label="show on map"
                    onClick={() => handleButtonClick(h.id)}
                    color="secondary"
                  >
                    <MapIcon />
                  </IconButton>
                </TableCell>
              </TableRow>
            );
          })}
        </TableBody>
      </Table>
    </TableContainer>
  );
};

export default HarvesterTable;
```

Kuvio 28. Metsäkonelistan ohjelmakoodi

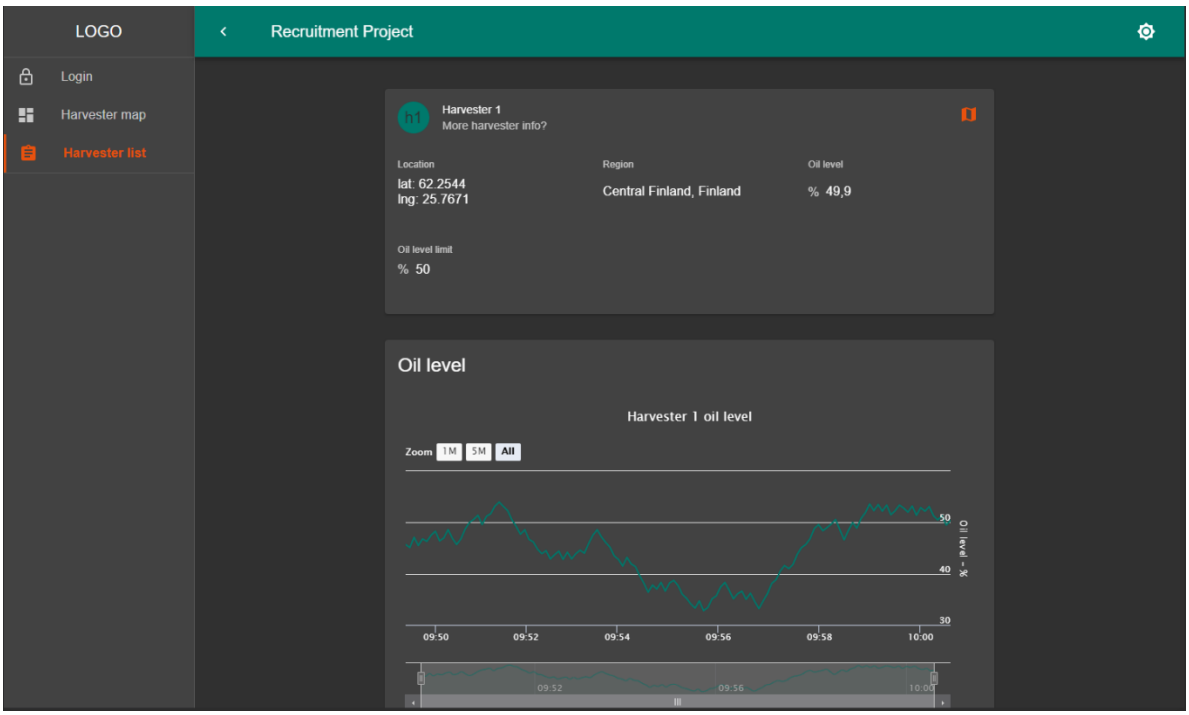



Kuvio 29. Metsäkonelistan mobiilinäkymä

Metsäkoneiden listanäkymästä voi valita yksittäisen metsäkoneen, jolloin koneen tietoja pääsee tarkastelemaan yksityiskohtaisemmin.

## 5.4 Yksittäisen metsäkoneen näkymä

Yksittäisen metsäkoneen näkymässä käyttäjä voi tarkastella metsäkoneen perustietojen lisäksi myös tarkempia yksityiskohtia. Esimerkkiprojektissa yksittäisen metsäkoneen näkymään on lisätty esimerkiksi öljyn pinnankorkeuden hälytysraja, sekä öljyn pinnankorkeuden kuvaaja.




Kuvio 30. Yksittäisen metsäkoneen näkymä

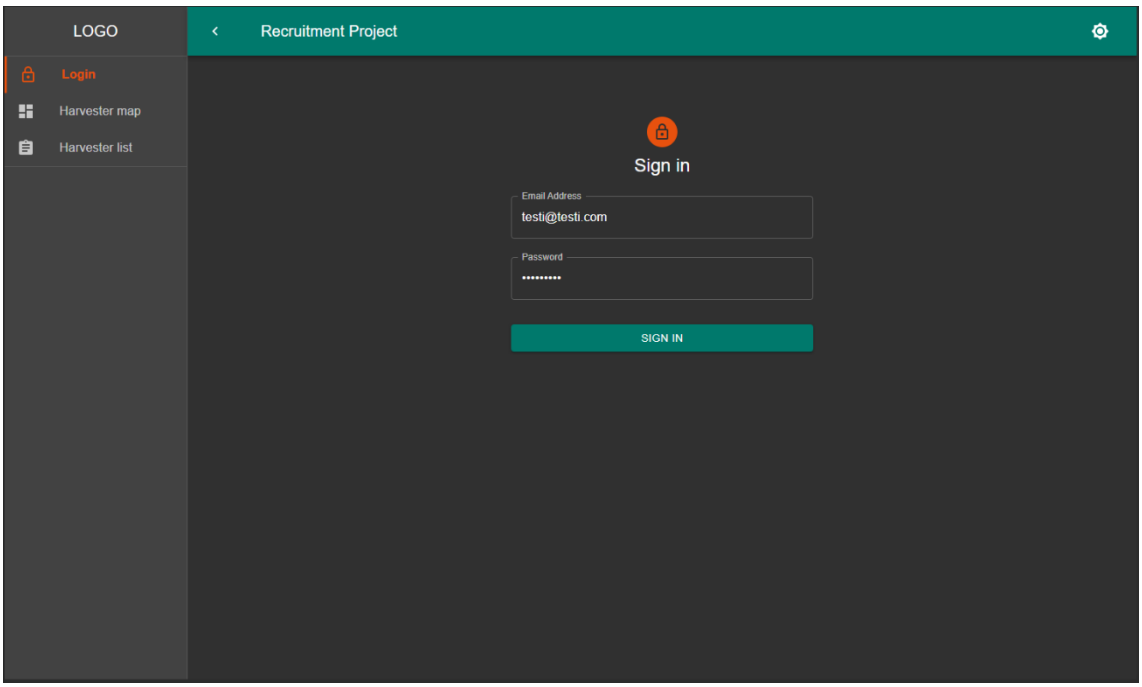
Kuviossa 30 on esitelty metsäkoneen yksittäisnäkymä, sekä öljyn pinnankorkeuden mittauksen viivakuvaaja. Viivakuvaaja toteutettiin laajasti käytössä olevalla Highcharts-kirjastolla. Kuviossa 31 on esitelty viivakuvaajan toteutus ohjelmakoodin puolella. Kuvaaja on sijoitettu Material Card -komponentin sisälle, ja kuvaajan dataa päivitetään metsäkoneiden backend-datan päivittyessä.

```
76    // Update graph data when oil level history changes by adding a point to the
      series
77    // Adding a single point is faster and better for performance, compared to
      updating the whole series
78    useEffect(() => {
79      if (initialized && chart && chart.series[0] && oilLevelHistory[0]) {
80        const newestData = oilLevelHistory[0];
81        const point = { x: newestData.time.getTime(), y: newestData.value };
82        chart.series[0].addPoint(point);
83      }
84    }, [chart, initialized, oilLevelHistory]);
85
86    // Change chart theme when app theme changes
87    useEffect(() => {
88      if (chart) {
89        chart.update(getGraphStyles(themeMode));
90      }
91    }, [themeMode, chart, getGraphStyles]);
92
93    return initialized ? (
94      <Card>
95        <CardHeader title="Oil level" />
96        <CardContent>
97          <HighchartsReact
98            constructorType="stockChart"
99            highcharts={Highcharts}
100           options={options}
101           callback={chartCreated}
102           allowChartUpdate={false}
103         />
104       </CardContent>
105     </Card>
106   ) : (
107     <p>No data</p>
108   );
109 };
```

Kuvio 31. Kuvaajakirjaston käyttö sovelluksessa

## 5.5 Tunnistautumisnäkymä

Sovellukseen lisättiin myös tunnistautumisnäkymä (kuvio 32), jota kautta käyttäjä pääsee kirjautumaan järjestelmään, ja siten suorittamaan mahdollisia hallintatoimintoja ja muita tunnistautumisen taakse lukittuja toiminnallisuuksia. Projektin käyttäjätarinoiden mukaisesti öljynpinnan hälytysrajan muokkaaminen on piilotettu tavallisilta käyttäjiltä. Tunnistautumisnäkymässä on yksinkertainen kirjautumislomake, johon käyttäjä voi syöttää sähköpostinsa ja salasanansa.

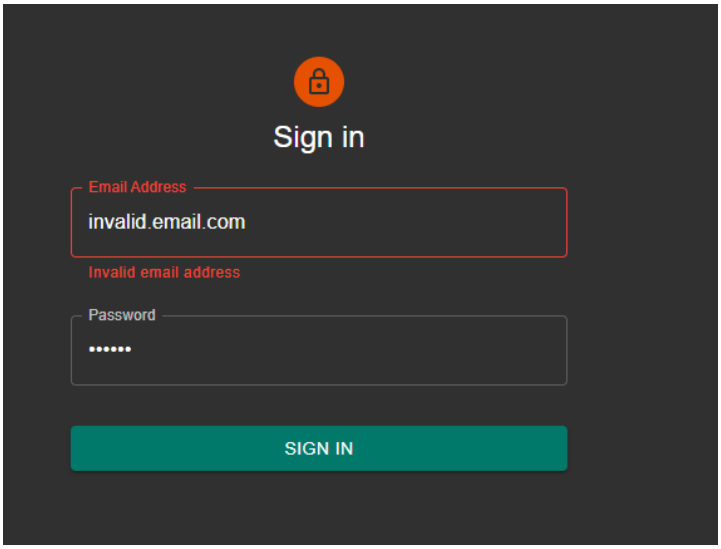



Kuvio 32. Sovelluksen kirjautumisnäkymä

Kuviossa 33 on esitelty kirjautumisnäkymän käytännön toteutus. Toteutus on tehty hyödyntäen Material UI –kirjaston tekstikenttiä ja muita sopivia komponentteja. Kirjautumislomakkeen varmennuksessa on hyödynnetty kolmannen osapuolen kirjastoa, jonka avulla lomakkeen kentille voidaan asettaa haluttuja varmennusehtoja. Kuvion 33 riveillä 68-72 näkyy sähköpostikentän varmennukseen liittyvä koodi. Sähköpostikenttä on pakollinen ja sen tarkistetaan regex-määrittelyllä. Kuviossa 34 on esitelty viallisesta syötteestä tuleva virheilmoitus.

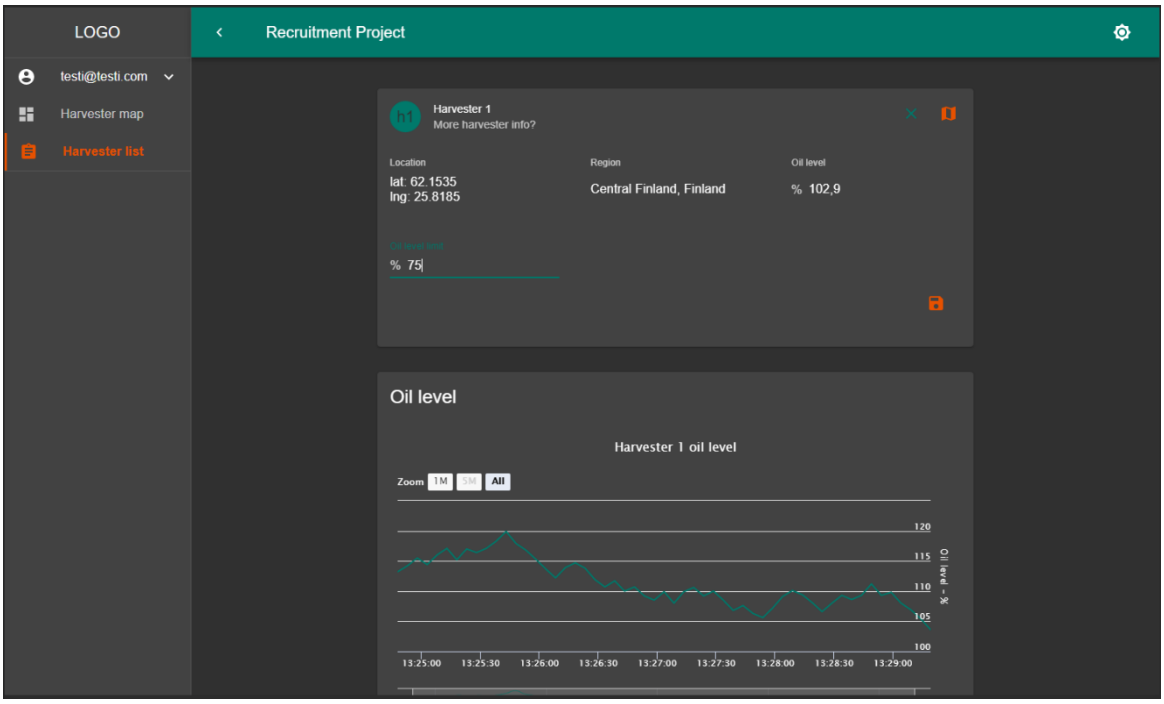
```
return (
  <Container component="main" maxWidth="xs">
    <div className={classes.paper}>
      <Avatar className={classes.avatar}>
        <LockOutlinedIcon />
      </Avatar>
      <Typography component="h1" variant="h5">
        Sign in
      </Typography>
      <form className={classes.form} onSubmit={handleSubmit(onSubmit)}>
        <TextField
          inputRef={register({
            required: 'Required',
            pattern: {
              value: /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i,
              message: 'Invalid email address',
            },
          })}
          variant="outlined"
          margin="normal"
          fullWidth
          id="email"
          label="Email Address"
          name="email"
          autoComplete="email"
          autoFocus
          error={!!errors.email}
          helperText={errors.email?.message}
        />
        <TextField
          inputRef={register({ required: 'Required' })}
          variant="outlined"
          margin="normal"
          fullWidth
          name="password"
          label="Password"
          type="password"
          id="password"
          autoComplete="current-password"
          error={!!errors.password}
          helperText={errors.password?.message}
        />
        <Button
          type="submit"
          fullWidth
          variant="contained"
          color="primary"
          className={classes.submit}
        >
          Sign In
        </Button>
      </form>
    </div>
  </Container>
);
```

Kuvio 33. Kirjautumisnäkymän toteutus




Kuvio 34. Kirjautumisnäkymän virheilmoitus

Kuviossa 35 näkyy metsäkoneen yksittäisnäkymä käyttäjän ollessa kirjautuneena. Vasemmassa reunassa on kirjautumispainikkeen sijaan käyttäjän sähköpostitunnus, jota kautta käyttäjä pääsee kirjautumaan ulos. Koneen yksittäisnäkymä tarjoaa nyt mahdollisuuden muokata öljyn pinnankorkeuden hälytysrajaa. Ylemmän kortin oikeassa yläkulmassa on rasti-ikoni, jonka avulla muokkaustila voidaan sulkea. Muokatut tiedot voidaan tallentaa oikean alakulman tallenna-painikkeella.




Kuvio 35. Metsäkoneen yksittäisnäkymä tunnistautuneelle käyttäjälle

Mallitoteutukseen tehtiin toiminnallisuuksien lisäksi vielä yksikkötestejä projektissa käytetyille komponenteille, käyttäjätarinoiden vaatimusten mukaisesti. Kuviossa 36 on esitelty esimerkki kirjautumislomakkeen yksikkötestauksesta. Yksinkertaisessa yksikkötestissä varmistetaan, että näkymän tekstikentät ja painike näkyvät näkymässä oikein, sekä testataan tekstikenttien tekstinsyötön toiminnallisuus.

```
import React from 'react';
import '@testing-library/jest-dom/extend-expect';
import { render, fireEvent } from '@testing-library/react';
// eslint-disable-next-line import/no-unresolved
import 'mutationobserver-shim';

import Login from './Login';

global.MutationObserver = window.MutationObserver;

test('renders correct form fields and button', () => {
  const component = render(<Login />);

  const email = component.container.querySelector('#email');
  expect(email).not.toBeNull();

  const password = component.container.querySelector('#password');
  expect(password).not.toBeNull();

  const button = component.container.querySelector('button[type="submit"]');
  expect(button).not.toBeNull();
});

test('allows entering text', () => {
  const component = render(<Login />);

  const email = component.container.querySelector('#email');
  expect(email).not.toBeNull();
  expect(email).toBeEmpty();
  if (email) {
    fireEvent.change(email, { target: { value: 'test@test.com' } });
    expect(email).toHaveValue('test@test.com');
  }

  const password = component.container.querySelector('#password');
  expect(password).not.toBeNull();
  expect(password).toBeEmpty();
  if (password) {
    fireEvent.change(password, { target: { value: '123' } });
    expect(password).toHaveValue('123');
  }
});
```

Kuvio 36. Kirjautumislomakkeen yksikkötesti

# 6 Pohdinta

Opinnäytetyön tavoitteiksi määriteltiin selvitystyö frontend-kehittäjän tärkeistä ominaisuuksista, ominaisuuksien pohjalta suunniteltu lähtötasoprojekti, jota voidaan tarvittaessa hyödyntää rekrytointivaiheessa tai uuden työntekijän työsuhteen alkuportailla. Tavoitteena oli myös valmistaa mallitoteutus suunnitellusta lähtötasoprojektista, käyttäen projektin lopullista määrittelydokumentaatiota.

Lopputuloksena työstä valmistui määrittelydokumentaatio, jonka avulla lähtötasoprojekti voidaan teettää rekrytointi- tai perehdytystarkoituksessa, sekä malliprojekti, joka sisältää kaikki vaatimuksissa määritellyt toiminnallisuudet.

Työn tulokset vastaavat alussa määriteltyjä vaatimuksia, ja työ saatiin pidettyä määritellyissä rajoissa, vaikka suunnittelupalaverien aikana toteutuksen laajentaminen nousi välillä esille. Mahdolliset laajentamiset ja tarkennukset jätettiin kuitenkin myöhemmälle jatkokehitykselle.

Tavoitteissa suuressa roolissa ollut osaamisalueiden ja ominaisuuksien tunnistaminen toteutettiin pääosin toimeksiantajan tiimin keskuudessa, jolloin niitä voitiin pohjustaa konkreettisiin työelämän taitoihin, sekä toimeksiantajan projektitoiminnan kannalta hyödyllisiin asioihin. Lähtötasoprojektin aihevalinta mietittiin yhdessä toimeksiantajan kanssa, jotta aihe olisi projektitoiminnan kannalta realistinen, ja jotta se tarjoaisi mahdolliselle työntekijälle tilaisuuden tutustua yhteen toimeksiantajan toimialoista.

Opinnäytetyön tuloksina kehittynyttä lähtötasoprojektin vaatimusmäärittelyä voidaan hyödyntää myös rekrytointi/perehdytysprojektin ulkopuolella, esimerkiksi uuden työntekijän työhaastattelussa haastattelukysymyksinä.

Opinnäytetyön tuloksina valmistuneiden määrittelydokumenttien ja malliprojektin avulla toimeksiantaja voi tehostaa rekrytointiprosessiaan, sekä nopeuttaa ja kehittää uuden työntekijän perehdytysjaksoa.

Opinnäytetyöprosessi tarjosi erinomaisen mahdollisuuden tutustua oman alan tärkeisiin osaamisalueisiin, sekä perehtyä projektidokumentaation valmistamiseen ja ketterän kehityksen projektitoimintaan. Nämä kaikki ovat tärkeitä työkaluja nykyaikaisen insinöörin työkalupakissa ja niiden kehittäminen on jatkuva prosessi, joka alkaa heti työuran ensivaiheilla.

Perehdytysprojektin vaatimusmäärittelyn ja käyttäjätarinoiden pohtiminen ja tekeminen tarjosi myös mahdollisuuden itsearviointiin ja oman osaamisen miettimiseen. Tärkeiksi koettuja ominaisuuksia ja osaamisalueita oli hyvä verrata omiin taitoihin ja sitä kautta tunnistaa kehitettäviä asioita.

Malliprojektin käytännön toteutus tarjosi mahdollisuuden tutustua alan uusimpiin teknologioihin ja kokonaisen projektityön suunnitteluprosessiin. Mallitoteutus tehtiin noudattaen ketterän kehityksen mukaisia suunnittelutapoja ja muita projektiryhmän keskuudessa katsottuja hyväksi havaittuja toimintatapoja. Käytännön toteutuksen ohjelmakoodia, versionhallintaa ja muita käytännöllisiä asioita käytiin läpi yhdessä projektiryhmän kanssa, jolloin työn laatua ja ohjelmakoodin selkeyttä pystyttiin valvomaan.

Ohjelmointityö tarjosi paljon hyviä haasteita. Esimerkiksi karttakirjastojen käyttö oli entuudestaan täysin tuntematonta, kuten myös React-kirjaston ja TypeScript-ohjelmointikielen yhteiskäyttö. Erityisen suuressa roolissa olivat erillisten kirjastojen dokumentaation sisäistäminen, sekä muiden avustavien materiaalien tulkitseminen. Näitä asioita pidetään yleisesti erittäin tärkeinä ohjelmointialalla, erityisesti frontend-kehityksessä, joka kehittyy ja muuttuu suurella vauhdilla.

Kokonaisuudessaan opinnäytetyö tarjosi erinomaisen mahdollisuuden haastaa omaa osaamistani, ja sitä kautta kehittyä alan ammattilaisena. Pääsin tutustumaan entistä tarkemmin tieto- ja viestintätekniikan insinöörin toimenkuvaan ja työelämän arkeen nykyaikaisen projektityön osalta. Opinnäytetyöprosessissa koetut onnistumiset ja haasteet tuovat paljon tärkeitä asioita ja opetuksia, joista on varmasti hyötyä vakituista työelämää ajatellen.

# Lähteet

Shaumik, D. 2020. Angular vs React vs Vue: Which Framework to Choose in 2020. Artikkeli codeinwp-sivustolla. Viitattu 25.8.2020. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

Devecto. N.d. Yritysesittely Devecto Oy:n verkkosivuilla. Viitattu 24.7.2020. https://devecto.com/devecto/

Devecto Oy rekisteritiedot. N.d. Devecto Oy:n rekisteritiedot asiakastieto-verkkosivulla. Viitattu 24.7.2020. https://www.asiakastieto.fi/yritykset/fi/devecto-oy/26161849/rekisteritiedot

Schwarzmüller, M. 2020. Angular vs React vs Vue. Artikkeli academind-sivustolla. Viitattu 3.8.2020. https://academind.com/learn/angular/angular-vs-react-vs-vue-my-thoughts/

Material Design. N.d. Material Design esittely. Viitattu 25.8.2020. https://material.io/design/introduction

Mähönen, P. 2020. Opinnäytetyön toimeksianto. Haastattelu 21.4.2020.

Perna, M. 2019. How to Beat 5 common JavaScript Interview Challenges. Artikkeli sitepoint-sivustolla. Viitattu 30.7.2020. https://www.sitepoint.com/5-common-coding-interview-challenges/

What To Expect From A Pre-Interview Coding Challenge. 7.11.2017. Viitattu 30.7.2020. Artikkeli Forbes-sivustolla. https://www.forbes.com/sites/quora/2017/11/07/what-to-expect-from-a-pre-interview-coding-challenge/#51768845115a

What is a front-end developer? N.d. Artikkeli Frontend Masters –sivustolla. Viitattu 3.8.2020. https://frontendmasters.com/books/front-end-handbook/2018/what-is-a-FD.html