OULUN AMMATTIKORKEAKOULU

Antte Alatalo

Responsiivisen käyttöliittymän toteuttaminen Ympäristökioski-sovellukselle

Responsiivisen käyttöliittymän toteuttaminen Ympäristökioski-sovellukselle

Antte Alatalo
Opinnäytetyö
Kevät 2022
Tradenomi, tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tradenomi (AMK), Tietojenkäsittely

Tekijä: Antte Alatalo
Opinnäytetyön nimi: Responsiivisen käyttöliittymän toteuttaminen Ympäristökioski-sovellukselle
Työn ohjaaja: Jouni Juntunen
Työn valmistumislukukausi ja -vuosi: Kevät 2022 Sivumäärä: 39

Tämän opinnäytetyön tarkoituksena oli toteuttaa responsiivinen käyttöliittymä Ympäristökioski-sovellukselle. Se on maanviljelijöille suunniteltu sähköinen työkalu, joka ehdottaa ympäristötoimenpiteitä maatilan lohkoille niiden ominaisuuksien perusteella. Sovelluksesta vastaavan hankkeen tavoitteena on edistää ympäristöystävällisempää viljelyä, ja lisätä maatilojen välistä, kestäviin tuotantotapoihin liittyvää yhteistyötä.

Opinnäytetyön teoria osuudessa vertaillaan eri toteutustapoja millä käyttöliittymä on mahdollista kehittää, jonka jälkeen seurataan käyttöliittymän toteutusta valituilla työkaluilla. Tavoitteena oli luoda intuitiivinen ja miellyttävän näköinen käyttöliittymä, joka on helppokäyttöinen myös puhelimella ja tabletilla. Tietoperustana oli aiempi kokemus verkkosivujen kehityksestä, ja käyttöliittymää rakennettiin käytettyjen teknologioiden dokumentaatioiden avulla. Kehityksessä myös hyödynnettiin lukuisia netistä löytyviä kirjoituksia, kommentteja, ja työkaluja.

Käyttöliittymästä tuli visuaalisesti odotusten vastainen, ja sen käytettävyys on hyvä myös pienemmillä näytönko'oilla. Sovelluksen käyttöliittymän, sovelluslogiikan ja tietokannan kehitystä jatketaan opinnäytetyön päättymisen jälkeen.

Asiasanat: Käyttöliittymä, responsiivisuus, verkkokehitys

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Antte Alatalo
Title of thesis: Responsive user interface for a web application
Supervisor: Jouni Juntunen
Term and year when the thesis was submitted: Spring 2022
Number of pages: 39

The purpose of the thesis was to develop a responsive user interface for a web application, called Ympäristökioski. Its goal was to propose environmentally friendly farming solutions to farmers, based on their farm's properties and interests. It's part of a bigger project, aiming for sustainable farming practices and increased collaboration between farms.

In the theory section different frameworks for frontend development are compared, and after that the development of the user interface with the chosen technologies is presented. The goal was to develop an aesthetically pleasing user interface, that is easy to use with all screen sizes.

The vision of the user interfaces visuality was achieved, and the application retains its usability even in smaller screen sizes. The development of the applications user interface, logic, and database continues after thesis' completion.

Keywords: User interface, responsiveness, web development

SISÄLLYS

1 JOHDANTO

Viljelijöillä on haastava työ. Heidän on osattava maataloustuotannon parhaat käytännöt, mutta viime vuosina myös ruoantuotannon ympäristövaikutukset ovat tulleet entistä olennaisemmaksi aiheeksi. On nyt entistä tärkeämpää, että viljelijät tuntevat toimintansa ympäristövaikutukset sekä mahdollisuudet kuormitusriskien minimointiin. Internetistä löytyy lähes rajattomasti tietoa, mutta ongelma on se, että juuri itselle relevantin tiedon löytäminen ja sen soveltaminen on rajallisilla resursseilla haastavaa.

Ympäristökioski on sähköinen työkalu, jonka avulla viljelijät voivat yhteistyössä neuvojien kanssa suunnitella nykyistä paremmin tilalleen ja peltolohkoilleen sopivia ympäristötoimenpiteitä. Työkalun avulla viljelijä voi selvittää ja seurata erilaisia ympäristötunnuslukuja. Se on osana Ympäristöviisasviljelijä hanketta, jonka tavoitteena on lisätä maatilojen välistä, erityisesti kestäviin tuotantotapoihin liittyvää yhteistyötä ja verkostoitumista. Hankkeessa kehitetään maatiloille ympäristöviisaita toimintatapoja hyödyntäen uusinta tutkimustietoa sekä laskenta- ja mallinnusmenetelmiä. (Rahkila 2021).

Opinnäytetyössä toteutetaan ProAgrialle responsiivinen verkkosovellus yhdessä lehtori Pekka Ojalan kanssa. Hän toteuttaa sovellukselle backendin ja sovelluslogiikan, ja minun tehtäväni on käyttöliittymän suunnittelu ja toteuttaminen. Hankkeessa on myös mukana monta eri tahoa, jotka antavat palautetta ja suuntaa sovelluksen etenemiselle.

1.1 Tavoitteet ja kehitystehtävä

Sovelluksen tarkoituksena on auttaa tilaa hahmottamaan, minkälainen kokonaisuus tilasta muodostuu ympäristönäkökulmasta katsottuna. Näin tullaan tietoisemmiksi siitä, ovatko nykytoimet riittävät vai voisiko kuvaa täydentää vielä jollain tilalle sopivalla ja taloudellisella ratkaisulla. Sovellus perustuu aiemmin kehitettyyn pdf-työkaluun, jolla viljelijä tekee maatilan alkukartoituksen, valitsee ominaisuudet tilan lohkoille ja valitsee toimenpiteet valitsemiensa ominaisuuksien perusteella. Sovellus kehitettiin alustavasti pöytäkoneiden ja kannettavien leveämmille näytöille, koska pdf-työkalu sisältää sellaisia nelikenttiä ja matriiseja, joiden käytön arveltiin olevan hankalaa mobiililaitteiden pienemmän näyttökoon vuoksi. (Baltic Sea Action Group 2021).

Hankkeen edetessä pidemmälle, sovelluksesta päätettiin kehittää mobiiliystävällinen versio, koska nähtiin tärkeänä, että viljelytilaa hahmotettaessa sovellusta voisi käyttää paikan päällä älypuhelimella tai tabletilla. Myös hankkeen aikataulu ja resurssit mahdollistivat mobiiliystävällisen version kehittämisen.

Suunniteltuja näkymiä sovelluksessa on alun perin kahdeksantoista kappaletta. Niistä ensimmäisenä ovat rekisteröinti ja kirjautuminen, jotta käyttäjä voi tallentaa oman etenemisen ja ehdotetut toimenpiteet. Itse sovelluksen ensimmäisessä näkymässä käyttäjä voi syöttää tai tuoda oman tilansa tiedot, tai tuoda sovelluksen oman testitilan tiedot halutessaan kokeilla sovellusta. Lohkotnäkymässä listautuvat tilan lohkojen tiedot kuten tunnus, nimi ja pinta-ala. Lohkoja voi myös yksittäisesti lisätä, muokata, ja poistaa. Ryhmät-näkymässä on mahdollista luoda ryhmiä lohkojen ryhmittelyä varten. Tilatason alkukartoitukseen käytettävissä näkymissä käyttäjä voi mm. kartoittaa tilan käytössä olevia mahdollisuuksia ja resursseja, lisätä mielenkiinnon kohteita ympäristönhoitoon ja viljelyn kehittämiseen, kartoittaa lohkojen sijainteja vesistöihin nähden matriisin avulla, ja lisätä lohkokohtaisia ominaisuuksia. Toimenpiteet-näkymässä käyttäjä voi valita lohkoille toimenpiteitä lohkojen ominaisuuksien perusteella. Lopuksi kooste-näkymässä käyttäjälle näkyy valitsemat toimenpiteet lohkokohtaisesti.

Kehityksessä on myös ennalta-arvattavia haasteita mobiili-käyttöliittymän suhteen, sillä moni käytettävä termi on kirjaimellisesti pitkä ja asiat pitää listata monessa kohtaa vierekkäin, joten haasteena tulee kaiken tarvittavan asian mahduttaminen näyttöön selkeästi. Myös matriiseja on käytetty viljelytilan hahmottamiseen, joiden tulee olla myös mobiilinäytöllä luettavia ja käytettäviä.

1.2 Työkalut ja menetelmät

Toimeksiannon vaatimuksena oli, että frontend toteutetaan Reactilla, joka on tämän hetken käytetyin SPA (Single Page Application) ratkaisu. Reactiin on myös saatavilla monia kirjastoja ja frameworkkejä, jotka helpottavat työtä valmiiksi rakennettujen komponenttien avulla.

Käyttöliittymän muotoilun ja responsiivisuuden toteuttamiseen on eri tapoja. Yksi tapa tyylien määrittelyyn on kirjoittamalla ne suoraan CSS-tiedostoon, hyödyntäen sen sisäänrakennettuja ominaisuuksia kuten CSS Grid ja CSS Flexbox. Vaihtoehtoisesti frameworkit, kuten Bootstrap, Tailwind, tai React-bootstrap mahdollistavat tyylien määrittelyn suoraan HTML-tiedostoon. Tutustun alkuun

kyseisiin toteutustapoihin vertaillen niiden soveltuvuutta sovellukseen, ottaen huomioon helppokäyttöisyyden, oppimisen hyödyn, ja oman kokemukseni.

1.3 Rajaus

Tehtäväni oli toteuttaa sovellukselle responsiivinen käyttöliittymä valitsemallani toteutustyylillä. Toteutuksessani keskityn puhtaasti käyttöliittymän tekemiseen, eli itse toiminnallisuuden toteuttaminen jää vastuualueeni ulkopuolelle. Käytän käyttöliittymissä kovakoodattua tietoa, eli toteutan näkymät niin että niissä on ulkonäöllisesti kaikki asiat mitä valmiissa sovelluksessa tulee olemaan, mutta toiminnallisuus, esim. asioiden lisääminen listaan toteutetaan vasta myöhemmässä vaiheessa.

Toteutan myös sovelluksen navigoinnin edellinen/seuraava tyylillä. Sovellusta käytetään yksi näkymä kerrallaan, jolloin käyttäjän täytettyä tarvittavat tiedot hän voi edetä seuraavaan osioon, tai palata edelliseen.

2 KÄYTTÖLIITTYMÄN TOTEUTUSMAHDOLLISUUDET

Käyttöliittymää suunnitellessa on hyvin tärkeää ottaa huomioon lukuisien eri laitteiden käytettävyys ja koko. Internetin alkuvuosina käytännössä kaikki verkkoselaukseen kykenevät päätelaitteet olivat pöytäkoneita, mutta teknologian ja mobiililaitteiden kehittyessä nykypäivän verkkoliikenteestä globaalisti vain kolmasosa on peräisin pöytäkoneelta (Enge 2021).

”Mobile first” on nykyään hyvin yleinen toimintamalli, jossa käyttöliittymä suunnitellaan ensin mobiililaitteille, ja vasta sen jälkeen laajempiin näyttöihin. Perinteisesti kun käyttöliittymä on ensin suunniteltu laajemmille näytöille, jossa on huomattavasti enemmän tilaa eri ominaisuuksille, käy helposti niin että mobiili-versiosta tulee vain riisuttu versio koko tuotteesta jossa osa ominaisuuksista jää pois, tai on vaikeammin löydettävissä. Mobile first-filosofialla käyttöliittymän suunnittelussa sille priorisoidaan tärkeimmät elementit parhaan käyttökokemuksen kannalta, jonka jälkeen ominaisuuksia ja sisältöä lisätään laajemmilla näytöillä. Näin tuotetta voidaan vahvistaa käyttämällä hyödyksi laajemman näytön mahdollisuudet, sen sijaan että ominaisuuksia joutuisi vähentämään skaalatessa pienemmälle näytölle. (Xia 2017).

Käyttöliittymän responsiivisuudella tarkoitetaan sivun sisällön mukautumista päätelaitteen koosta riippuen. Käytännössä kaikki uudemmat verkkosivut ovat responsiivisiä. Pöytäkoneen, tabletin, tai mobiililaitteen käyttäjän huomiotta jättäminen ei ole kannallista käyttökokemuksen tai käyttäjämäärän maksimoimiseksi. Ennen oli yleistä luoda verkkosivuille oma mobiiliversio erilliseen URL:iin, mutta nykyään se ei ole kannattavaa mm. hakukoneoptimisoinnin kannalta, koska Google arvioi sivustot mobiilisisällön perusteella (Google 2022).

Responsiivinen sivu voidaan toteuttaa määrittelemällä elementtien koko näytön koosta riippuen CSS-tiedostoon, ja erilaisten frameworkien kuten Bootstrapin avulla osan määrityksistä voi tehdä suoraan HTML-tiedostoon. Elementit, jota halutaan muuttaa näytön koon vaihtuessa, määritellään uudestaan ”breakpointtien” sisään. Breakpoint on yleensä pikseleillä määritelty näytön maksimi tai minimi leveys, jossa siihen määritetyt tyylit tulevat voimaan, kun sivu havaitsee sitä vastaavan näytön koon (kuva 1). Yleisesti niihin määritellään sisällön, kuten fonttien ja kuvien koko. (Mozilla 2022).

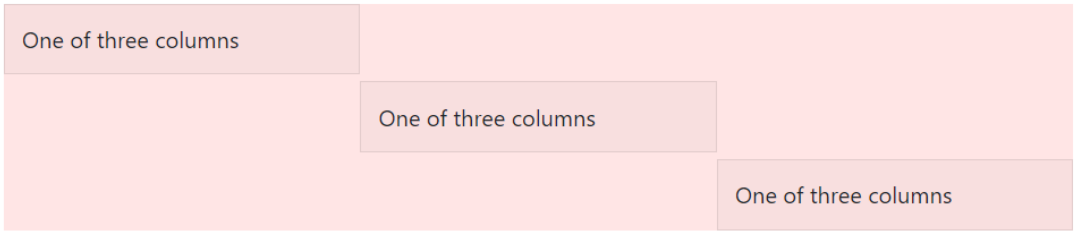
```
@media (min-width:768px) {
  .grid-container {
    display: grid;
    grid-template-columns:  1fr 1.5fr 1fr 1fr 1fr 0.3fr 0.2fr;
  }
  .grid-item {
    padding: 0.1em;
    font-size: 1em;
    align-items: flex-start;
    font-weight: 600;
  }
  .grid-header-item {
```

Kuva 1. Media queryllä määritelty breakpoint

2.1 Bootstrap

Bootstrap on ollut jo pitkään yksi käytetyimmistä CSS-frameworkeistä. Pitkään jatkuneen suosion myötä käyttäjäyhteisö on laaja, ja tukea on helposti saatavilla. Bootstrapillä responsiivisen käyttöliittymän saa rakennettua vaivattomammin sen sisäisen Flexboxilla rakennetun grid-systeemin avulla. Grid-systeemi jakaa käyttöliittymän osiin; riveihin ja sarakkeisiin, jolloin siitä saa helposti järjestetyn ja symmetrisen. Suoraan HTML-tiedostosta voi päättää kuinka ison osan mikäkin alue ottaa näytöstä riippuen käyttäjän näytön koosta. Perinteisesti käyttöliittymän osioiden koko on hoidettu itse tyyli-tiedostoja kirjoittamalla, mutta Bootstrapin valmiiksi määritetyt luokat hoitavat sen ilman ylimääräistä CSS-määrittelyä. Bootstrapissä käyttöliittymän osioiden leveys on jaettu kahteentoista (12) saman levyiseen sarakkeeseen, joista myös itse sarakkeet voi jakaa pienempiin osiin. (Bootstrap 2022).

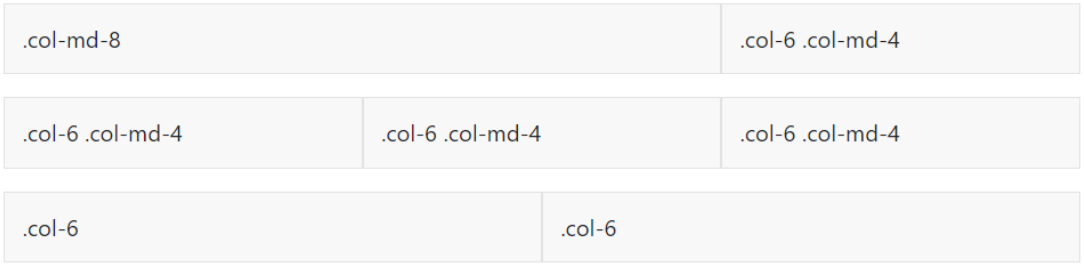
Luokalle ei ole pakko määrittää sarakkeen leveyttä, vaan Bootstrap määrittelee sen automaattisesti. Kun yhdellä rivillä on monta saraketta, jolle ei ole määritetty leveyttä, vievät ne oletuksena saman verran tilaa, hyödyntäen koko rivin leveyden (kuva 2). Diveille voi myös määrittää muita ominaisuuksia, kuten elementin paikan pysty- tai rivisuunnassa. (Bootstrap 2022).




```
<div class="container">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
</div>
```

Kuva 2. Bootstrap esimerkki

Responsiivisuus on helppo toteuttaa Bootstrapillä. Sarakkeille voidaan määrittää niiden leveys riippuen käyttäjän näytön koosta Bootstrappiin rakennettujen breakpointtien avulla. Sarakkeille voidaan määrittää monta eri luokkaa, jotka vastaavat sarakkeen leveyttä tietyllä näytön koolla. Sarake, jonka luokat ovat ”col-6 col-md-4”, vie kolmasosan rivin leveydestä, kun käyttäjän näytön koko on vähintään 768px, mutta sitä pienemmällä näytöllä se puolet leveydestä (kuva 3). Kun yhdessä rivissä olevien sarakkeiden mitta on enemmän kuin 12, siirtyy sen ylimenevät sarakkeet uudelle riville. (Bootstrap 2022).



Kuva 3. Responsiiviset Bootstrap-sarakkeet

Bootstrapissä on myös laaja valikoima valmiiksi rakennettuja komponentteja helposti käytettävissä. Muun muassa napit, modaalit, dropdown-menut, ja lomakkeet tuovat sivuun toiminnallisuutta ilman ylimääräistä koodausta.

2.2 React-bootstrap

React-Bootstrap on kirjasto, joka yhdistää Bootstrap komponentit suoraan React komponentteihin. Sillä ei ole riippuvuutta natiivi Bootstrapistä, vaan kirjaston käyttäminen vaatii ainoastaan Reactin ja React-bootstrapin asentamisen. React-bootstrapissä syntaksi on kiistämättä helpommin luettavampaa. Sen sijaan että DOM:sta tulee div-viidakko, saman komponentin voi toteuttaa selkeämmällä ja lyhyemmällä syntaksilla (kuva 4). (React-bootstrap 2022).

```
<Form>
  <Form.Group className="mb-3" controlId="formBasicEmail">
    <Form.Label>Email address</Form.Label>
    <Form.Control type="email" placeholder="Enter email" />
    <Form.Text className="text-muted">
      We'll never share your email with anyone else.
    </Form.Text>
  </Form.Group>

  <Form.Group className="mb-3" controlId="formBasicPassword">
    <Form.Label>Password</Form.Label>
    <Form.Control type="password" placeholder="Password" />
  </Form.Group>
  <Form.Group className="mb-3" controlId="formBasicCheckbox">
    <Form.Check type="checkbox" label="Check me out" />
  </Form.Group>
  <Button variant="primary" type="submit">
    Submit
  </Button>
</Form>
```

Kuva 4. React-bootstrapillä tehty lomake

Komponentteihin on valmiiksi yhdistetty Reactin ja Boostrapin toiminallisuutta, jolloin sitä ei tarvitse erikseen kirjoittaa. Yksi haitta React-bootstrapissä on se, että jos tulevaisuudessa rakennetun käyttöliittymän pohjana on jokin muu framework kuin React, opitusta tulee hyödytöntä. Huomioonotettavaa on myös se, että kun Bootstrap on sidottu yhteen Reactin kanssa, jos syystä tai toisesta Reactin haluaa myöhemmin vaihtaa johonkin muuhun frameworkiin, pitää käyttöliittymän koodi toteuttaa pitkälti uusiksi (Cohen 2021).

2.3 Tailwind

Tailwind on kasvavassa suosiossa oleva ”utility-first” CSS-framework, jossa valmiiksi määritetyt tyylit lisätään suoraan HTML-elementteihin. Sen sijaan että elementtien tyylit määritellään niille annettujen class-attribuuttien perusteella CSS-tiedostossa, Tailwindissä tyylit määritellään suoraan class-attribuutteihin. (Tailwind Labs Inc. 2022).

```
<div class="p-6 max-w-sm mx-auto bg-white rounded-xl shadow-lg flex items-center space-x-4">
  <div class="shrink-0">
    <img class="h-12 w-12" src="/img/logo.svg" alt="ChitChat Logo">
  </div>
  <div>
    <div class="text-xl font-medium text-black">ChitChat</div>
    <p class="text-slate-500">You have a new message!</p>
  </div>
</div>
```

Kuva 5. Tailwind esimerkki

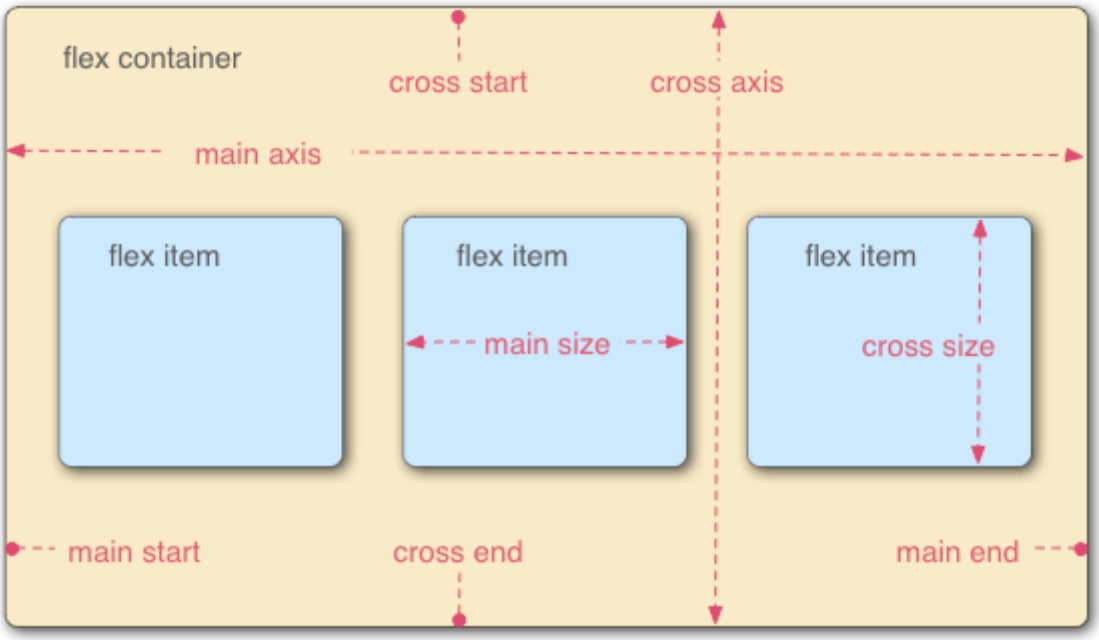
Tailwindin suosio perustuu sen nopeaan ja suoraviivaiseen toteutukseen. Elementeille on helppo määrittää niille yksilölliset ominaisuudet. Se nopeuttaa tuotantoa, koska tyylitiedoston ja HTML-tiedoston välinen pomppiminen vähentyy huomattavasti. Yleinen ongelma perinteisellä CSS-määrittelyllä on se, että moni eri elementti jakaa saman class-attribuutin, jolloin pienikin tyylimuutos saattaa huomaamatta vaikuttaa muihin elementteihin. Koska Tailwindissä tyylit määritetään yksilöllisesti suoraan elementtiin, tarkoittamattomat tyylimuutokset eivät ole ongelma (kuva 5). (Tailwind Labs Inc 2022).

Tailwindin tyylimäärittelytavalla myös CSS-tiedoston koko pysyy paljon pienempänä ja selkeämpänä. Se tuo selkeyttä tyylitiedostoon kumminkin epäselkeämmän HTML-tiedoston hinnalla. Elementeistä tulee helposti hyvin pitkiä ja hämmentäviä attribuutti-janoja. Tailwindissä tuntuu myös vaikeammalta tehdä jälkikäteen tyylimuutoksia elementtiin, joka toistuu monessa eri näkymässä.

2.4 CSS Grid ja Flexbox

CSS Grid ja Flexbox on CSS:n sisään rakennettuja ominaisuuksia, joilla käyttöliittymän muotoilun ja responsiivisuuden voi toteuttaa ilman ulkoisia kirjastoja tai frameworkkejä. Flexbox on yksiulotteinen, eli elementit voi järjestää joko sarakkeena tai rivinä. Grid on puolestaan kaksiulotteinen, eli

elementtien järjestyksen voi päättää sarakkeina ja riveinä. Flexbox on vähän yksinkertaisempi systeemi, jota pääosin käytetään, kun sivun sisältö halutaan helposti mahduttaa sopivaan muotoon. (Maldonado 2020).



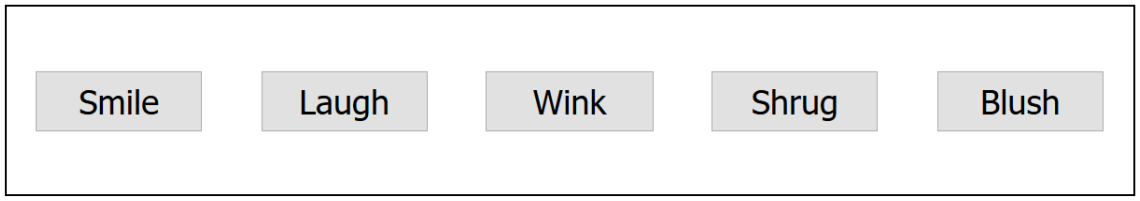
Kuva 6. Flexbox malli

Oletuksena containerin elementit järjestyvät riveittäin, jolloin uusi elementti mahdutetaan samalle riville. Elementeille voidaan määrittää niiden leveys pikseleinä, tai suhteuttamalla ne muihin elementteihin, jolloin kaikista elementeistä saadaan helposti mm. saman levyisiä. Elementin sisälle sijoitettu teksti ei ylety pitemmälle kuin sen leveys, vaan sisältö järjestyy allekkain (kuva 6). Elementeille myös annetaan flex-wrap-attribuutti, joka oletuksena on nowrap, tarkoittaen että uusi elementti mahdutetaan samalle riville, mahdollisesti kutistaen muita rivin elementtejä. ”Flex-wrap: wrap” puolestaan määrittää uuden elementin paikan uudelle riville, jos ylemmän rivin määritetty kokonaisleveys ylittyy. Elementeille voidaan myös määritellä muita attribuutteja, jotka järjestävät elementtejä toivottuun muotoon. (Mozilla 2022).

```
div {
  display: flex;
  align-items: center;
  justify-content: space-around;
}
```

Kuva 7. Flexbox-attribuutteja

Ensin diville määritetään, että sen tulee käyttää flexboxia (kuva 7). ”Align-items” määrittää elementtien kohdan poikkisuunnassa pääakselista. Sen oletusarvona on ”stretch”, joka venyttää elementin sen isäluokalle määritetyn luokan pituiseksi, tai jos pituutta ei ole määritetty, siitä tulee yhtä pitkä kuin saman rivin pisimmästä elementistä. Arvolla ”center” elementti järjestyy korkeussuunnassa keskelle, jos pääakselin suunta on oletus, eli sivusuunnassa. ”Justify-content” määrittää elementtien sijainnin pääakselin suunnassa. Oletuksena se on ”flex-start”, jolloin elementit järjestyvät rivin alkuun, mutta arvolla ”space-around” ne järjestyvät tasaisesti, antaen hieman tilaa elementtien väliin (kuva 8). (Mozilla 2022).



Kuva 8. Keskitetyt elementit kuvan 7. attribuuteilla

CSS Grid on Flexboxia voimakkaampi systeemi, jolla ulkoasu voidaan määritellä kaksiulotteisesti. Sarakkeiden ja rivien koko voidaan määrittää pikseleinä, prosentteina, tai suhteutettuna isäluokan kokoon. Käyttöliittymä voidaan toteuttaa halutessaan hyvin tarkasti, mutta Gridiä on myös mahdollista käyttää vähän yksinkertaisemmalla tyylillä. (Mozilla 2022).

```
.wrapper {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```

Kuva 9. CSS Grid esimerkki

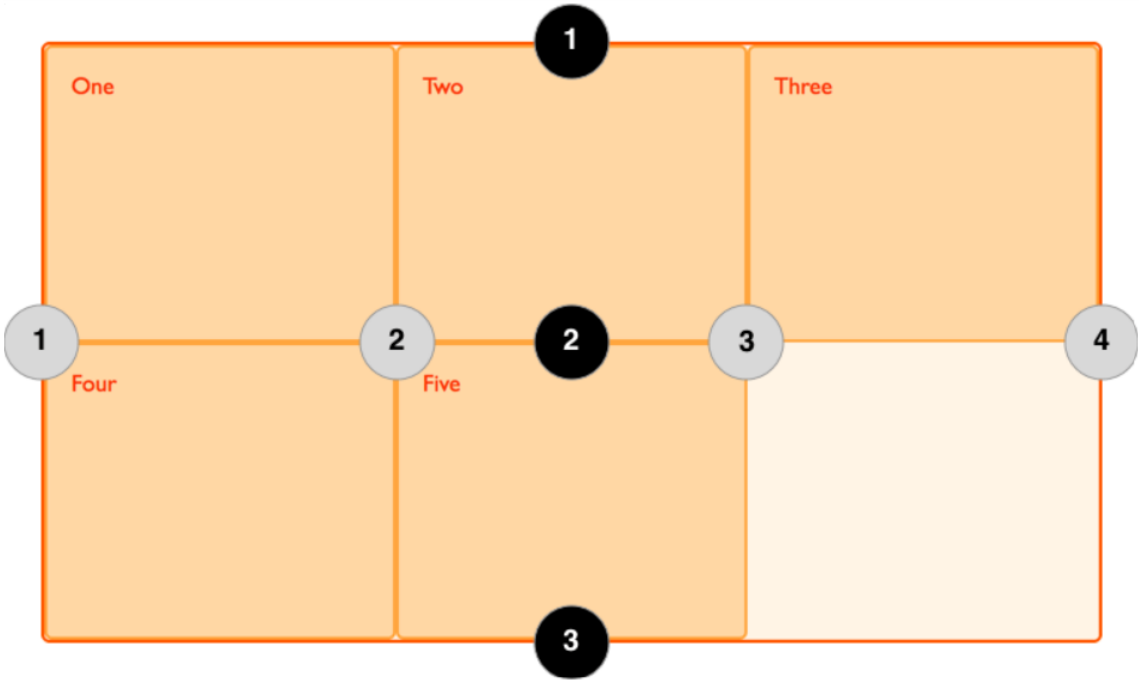
Luokalle määritetään, että se käyttää gridiä, jolle määritetään kolme 200px levyistä saraketta (kuva 9). Jos luokassa on enemmän kuin kolme elementtiä, ne järjestyvät uudelle riville. Yleisempi tapa määrittää sarakkeiden leveys, on hyödyntää ”fr”-arvoja (fraction). Sen avulla leveys määritellään suhteessa luokalle annettuun tilaan, jolloin koko tila saadaan hyödynnettyä ja esimerkiksi ja sarakkeista saa helposti esimerkiksi samankokoisia (kuva 10). (Mozilla 2022).

```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Kuva 10. Sarakkeet fraction-arvoilla

CSS Gridistä voimakkaan tekee kumminkin sen mahdollisuus määrittää ulkoasu kaksiulotteisesti. Jos elementtejä lisätään gridiin enemmän kuin sille on määritetty rivejä tai sarakkeita, niille automaattisesti määritetään paikka ja koko perustuen muiden elementtien kokoon. Tarkemmin käyttöliittymä voidaan kumminkin toteuttaa hyödyntämällä mahdollisuutta luoda kaksiulotteinen kehikko.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: 100px 100px;
}
```



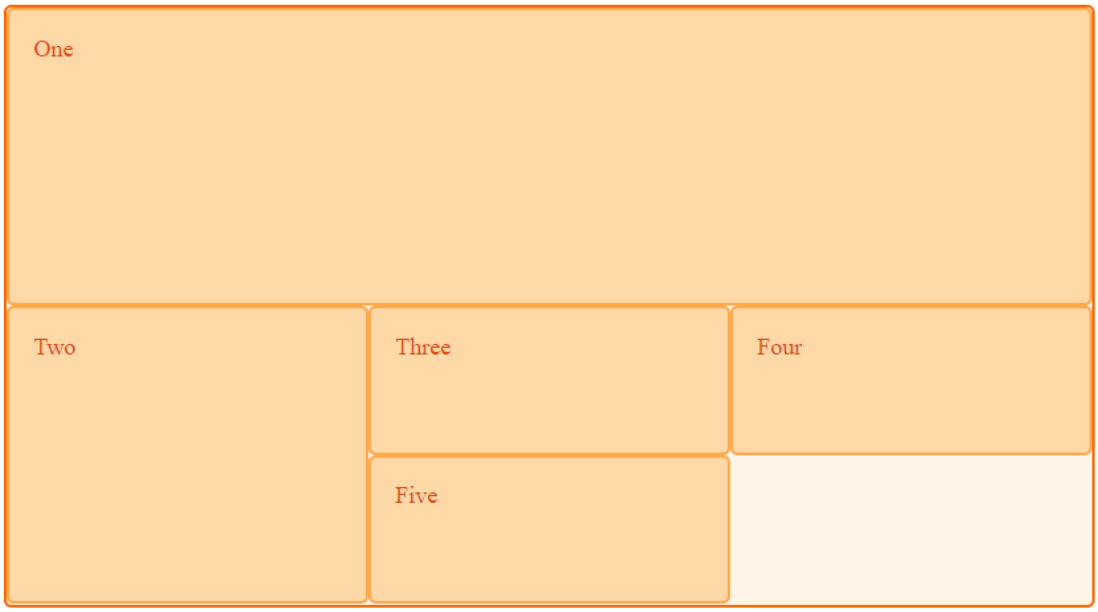
Kuva 11. CSS Grid-kehikko

Luokalle luodaan kehikko, jossa on kolme samanmittaista saraketta, ja kaksi 100 pikselin korkuista riviä (kuva 11). CSS Grid luo luokan lohkojen välille linjat, jonka avulla sisältö voidaan sijoittaa haluttuun kohtaan. Jokaiselle lohkolle voidaan määrittää haluttu koko antamalla sille arvot, jotka määrittävät lohkon alku- ja loppukohdan.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}

.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}

.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}
```

Kuva 12. Tarkemmin määritetyt lohkot

Luokalle määritetään kolme samankokoista saraketta, ja rajaton määrä rivejä, jotka ovat oletuksena 100px korkuisia (kuva 12). Tämä luo vastaavanlaisen kehikon (kuva 11), jonka ensimmäinen sarakelinja alkaa vasemmalta linjasta 1, ja päättyy linjaan 4. Rivit ovat 100px korkeita, mutta lohko voidaan määrittää viemään esimerkiksi kahden rivin verran tilaa, jolloin rivin korkeudeksi tulee 200px.



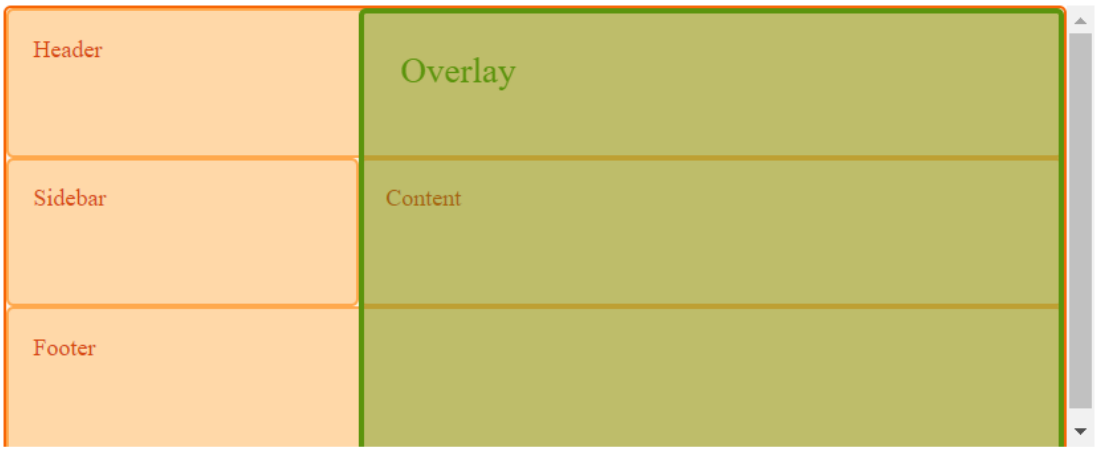
Kuva 13. Ulkoasun muoto kuvan 12 määrittelyillä

Lohko yksi on määritetty alkamaan sarakelinjasta 1, ja päättymään linjaan 4, käyttäen kaiken sen luokalle määritetyn leveyden (kuva 13). Sen korkeus alkaa rivilinjalta 1, ja päättyy linjalla 3, jolloin se vie kahden 100px korkuisen rivin tilan. Lohkolle kaksi on määritetty sarakelinjan alkukohta, mutta ei loppukohtaa, jolloin se oletuksena vie sarakkeille määritetyn tilan, eli kolmasosan leveydestä. Lohkoille kolme, neljä ja viisi ei ole erikseen määritetty ominaisuuksia, jolloin jokainen niistä on 100px korkuinen, ja kolmasosan luokan leveydestä. (Mozilla 2022).

```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-auto-rows: minmax(100px, auto);
  grid-template-areas:
    "hd hd hd hd   hd   hd   hd   hd   hd"
    "sd sd sd main main main main main main"
    "ft ft ft ft   ft   ft   ft   ft   ft";
}

.header {
  grid-area: hd;
}

.footer {
  grid-area: ft;
}

.content {
  grid-area: main;
}

.sidebar {
  grid-area: sd;
}

.wrapper > div.overlay {
  z-index: 10;
  grid-column: main-start / main-end;
  grid-row: hd-start / ft-end;
  border: 4px solid rgb(92,148,13);
  background-color: rgba(92,148,13,.4);
  color: rgb(92,148,13);
  font-size: 150%;
}
```

Kuva 14. Nimetyt lohkot

Yksi hyvä ominaisuus CSS Gridissä on sen mahdollisuus nimetä lohkot, jonka avulla sisällön voi sijoittaa sivuun tarkasti. Isäluokalle määritetään kehikko, jonka lohkoille annetaan nimet (kuva 14). Lohkot ovat määritetyn kokoisia, ja lapsiluokkien koko voidaan määrittää viittaamalla kehikolle määritettyjen lohkojen nimiä. Luokka overlay on määritetty alkamaan ensimmäisestä main-nimisestä lohkosta, ja päättymään viimeiseen. Sen korkeus alkaa hd-lohkosta, ja päättyy ft-lohkoon. (Kuva 15.)



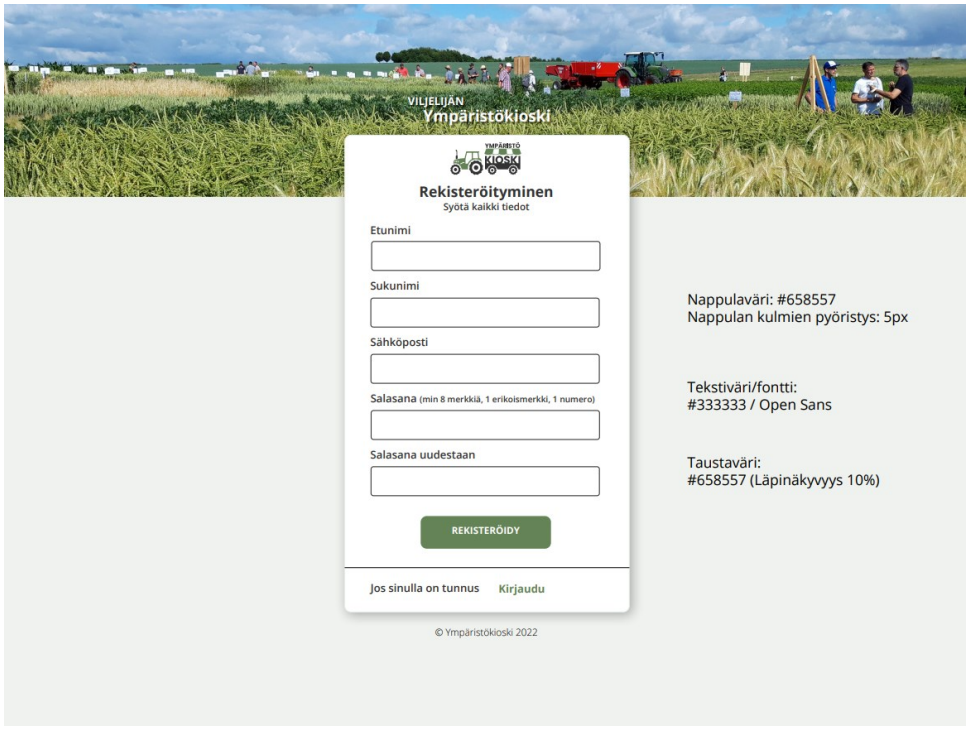
Kuva 15. Ulkoasu kuvan 14. määrityksillä

CSS Grid on hyvä vaihtoehto, jos haluaa rakentaa tarkkoja ja komplekseja ulkoasuja (Maldonado 2020). Yksinkertaisin toteutusmenetelmä se ei ole, ja alussa CSS Grid saattaa tuntua päälle-käyvältä. Pienen oppimiskynnyksen yli päästessä se on kumminkin ehkä voimakkain tapa toteuttaa haluttu ulkoasu, ja tarpeeksi osaavalla on lähes rajattomat mahdollisuudet käyttöliittymän muotoiluun. Hyvä ominaisuus CSS Gridissä on myös se, että uusimmilla Mozilla-selaimilla gridiä voi tutkia suoraan sivustolla.

3 YMPÄRISTÖKIOSKIN KÄYTTÖLIITTYMÄN TOTEUTUS

Käyttöliittymän toteuttamiseen vaihtoehtoja ja ratkaisuja on monia. Intuitiivisen käyttökokemuksen takaamiseksi iso osa ajasta menikin käyttöliittymän käytettävyyden ja ulkonäön suunnitteluun. Puhtaan visuaalisen ulkonäön lisäksi värimaailma myös vaikuttaa siihen, miten helposti käyttäjä osaa suorittaa haluamansa toiminnon. Värimaailman tulee olla myös hillitty. Toisinaan web-sivuilla voi olla hyödyksi käyttää moniakin eri värejä ja kontrasteja huomion keskittämiseen, mutta tämän sivun käyttäjän päämäärä on hyödyntää sovellusta ympäristöystävällisemmän viljelyn suunnitteluun, joten käytettävyyden helppous on prioriteettina huomiota herättävän ulkonäön sijasta.

Sovellukselle oli suunniteltu valmiiksi alihankkijan toimesta rekisteröinti näkymä, joka oli mielestäni hyvä, ja lähdin sen pohjalta toteuttamaan myös tabletti ja pöytäkoneen koneen kokoiset näkymät (kuva 16). Sovelluksesta oli versio, jonka pohjalle sovelluksen toiminallisuutta oli rakennettu, mutta ulkonäkö vanhanaikainen, niin otin sen näkymistä viitteitä mitä elementtejä käyttöliittymässä tulisi olla.

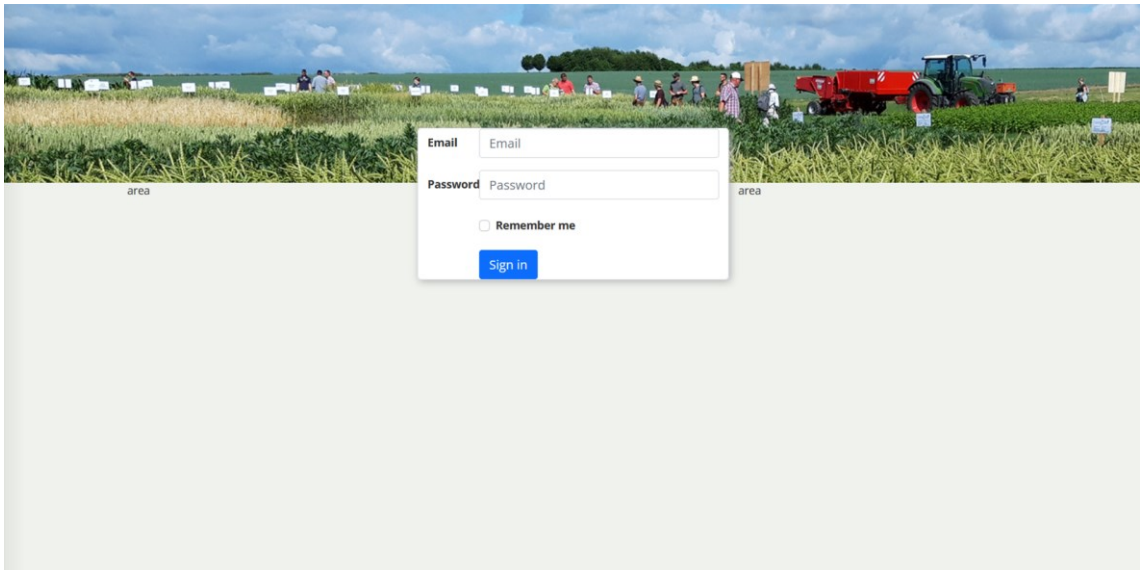


Kuva 16. Alihankkijan layout-esimerkki

Visiona oli pitää jokaisessa näkymässä keskellä staattinen alue, jossa rekisteröinti-näkymästä viimeiseen näkymään asti logo pysyy kokoajan samassa paikassa, ja leveämmällä näytöllä keskialueeseen tulee lisää leveyttä. Sovelluksen värimaailma painottuu luontoteeman mukaisesti vihreän eri sävyihin.

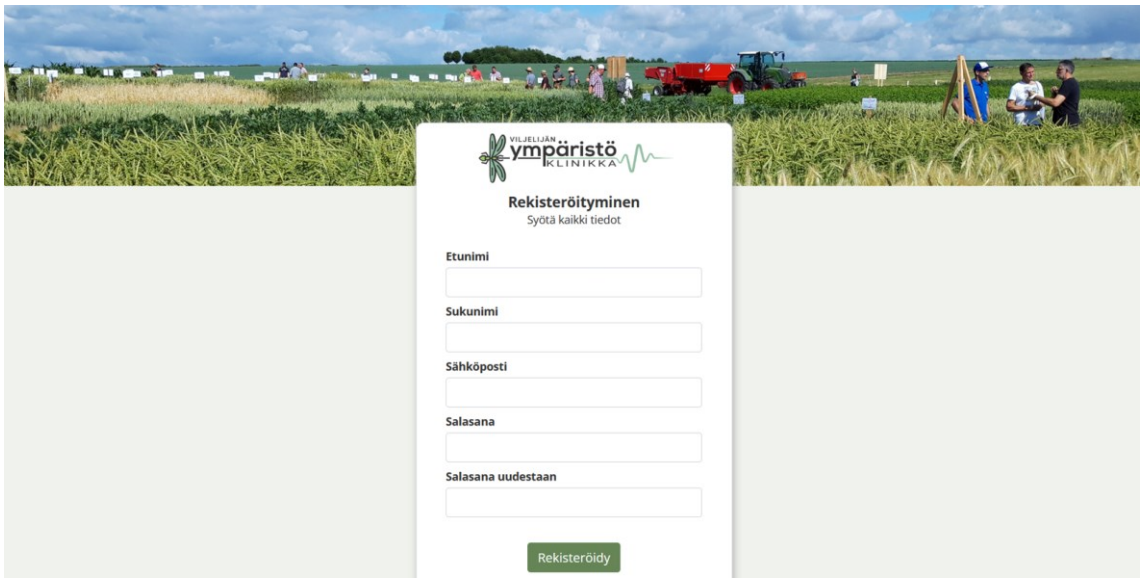
Toteutustapaa miettiessä päädyin lopulta Bootstrappiin, joka oli minulle jo entuudestaan jonkun verran tuttu. Sen helppokäyttöisyys oli ratkaiseva tekijä. Näkymät sovelluksessa on hyvin samanlaisia ja yksinkertaisia, joten esim. Tailwindin yksilöllisempiä määrittelyitä en kokenut hyödylliseksi. Myös responsiivisuuden toteutus on Bootstrapilla minun mielestäni kaikkien helpointa. Suoraan elementteihin määritetään koot, jolla elementin leveys määrittyy näytön koon mukaan. React-bootstrappiä en valinnut, koska koin hyödyllisemmäksi pelkän Bootstrapin osaamisen, enkä muutenkaan toteuta toiminnallisuutta sovellukseen.

Lähdin toteuttamaan käyttöliittymää alihankkijan suunnitellusta rekisteröinti-näkymästä. Ensimmäiseksi tehtäväksi otin alueen tekemisen keskelle, jossa rekisteröintilomake on, joka onnistui helposti Bootstrapillä pienen perehtymisen jälkeen. Keskialueen tuli olla myös osaksi bannerin päällä, jonka tein negatiivisella marginilla (kuva 17).



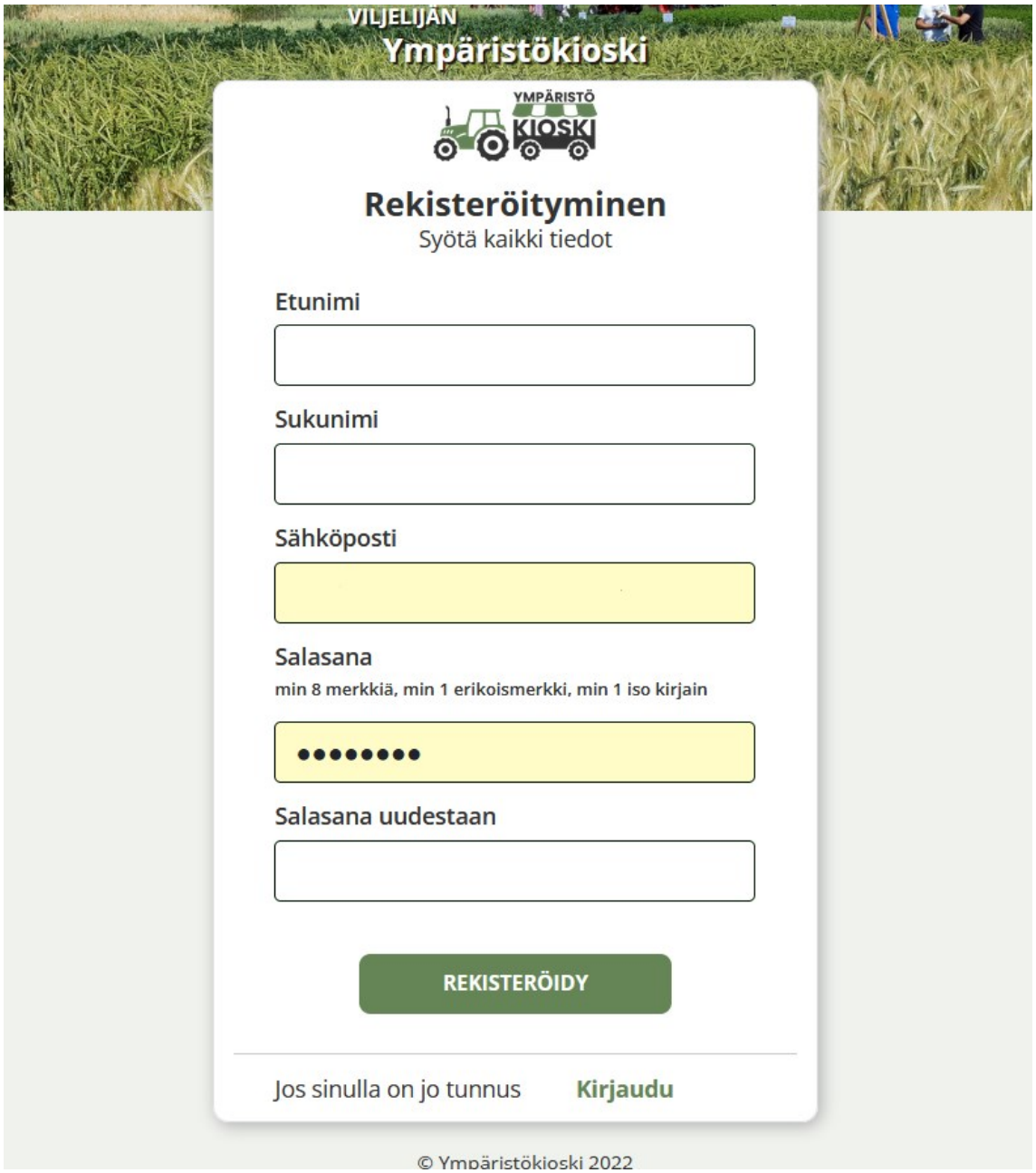
Kuva 17. Käyttöliittymän muotoilun alkuvaihe

Seuraavaksi aloin toteuttamaan kunnollisen rekisteröintilomakkeen lisäämistä Bootstrapin komponenteilla. Lisäsin keskialueeseen lomaketiedot, logon, ja rekisteröintipainikkeen, ja pyöristin lomakkeen kulmat. Käytin ympäristökioskin entistä logoa placeholderina, koska uudempaa logoa en ollut vielä saanut. (Kuva 18).



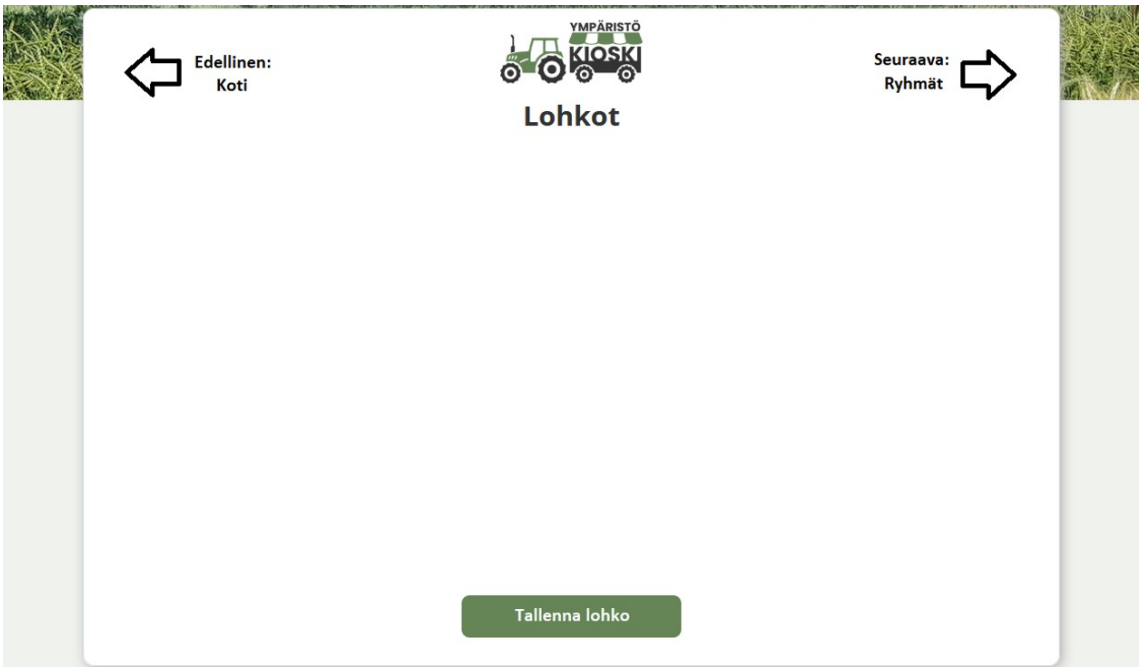
Kuva 18. Keskeneräinen rekisteröinti-näkymä

Kun sisältö oli lisätty, aloin määrittelemään tyylejä. Tein lomakekentän ympäriviivauksesta tummat, ja muotoilin rekisteröintipainikkeen vastamaan alihankkijan suunnittelemaa tyyliä. Vaihdoin painikkeen fontin paksummaksi, ja tein siitä leveämmän lisäämällä tyhjää tilaa (padding). Vaihdoin myös logon uudempaan versioon, ja lisäsin sovelluksen nimen bannerin päälle. Toteutin tyylimuotoilut bannerin päällä olevaan tekstiin, ja keskitin sen lomakkeen ylle. Lisäsin rekisteröintilomakkeen alalaitaan linkin kirjautumis-näkymään. Toteutin myös kirjautumis-näkymän samalle pohjalle, vaihtaen vain lomakkeen kentät sopiviksi. (Kuva 19).



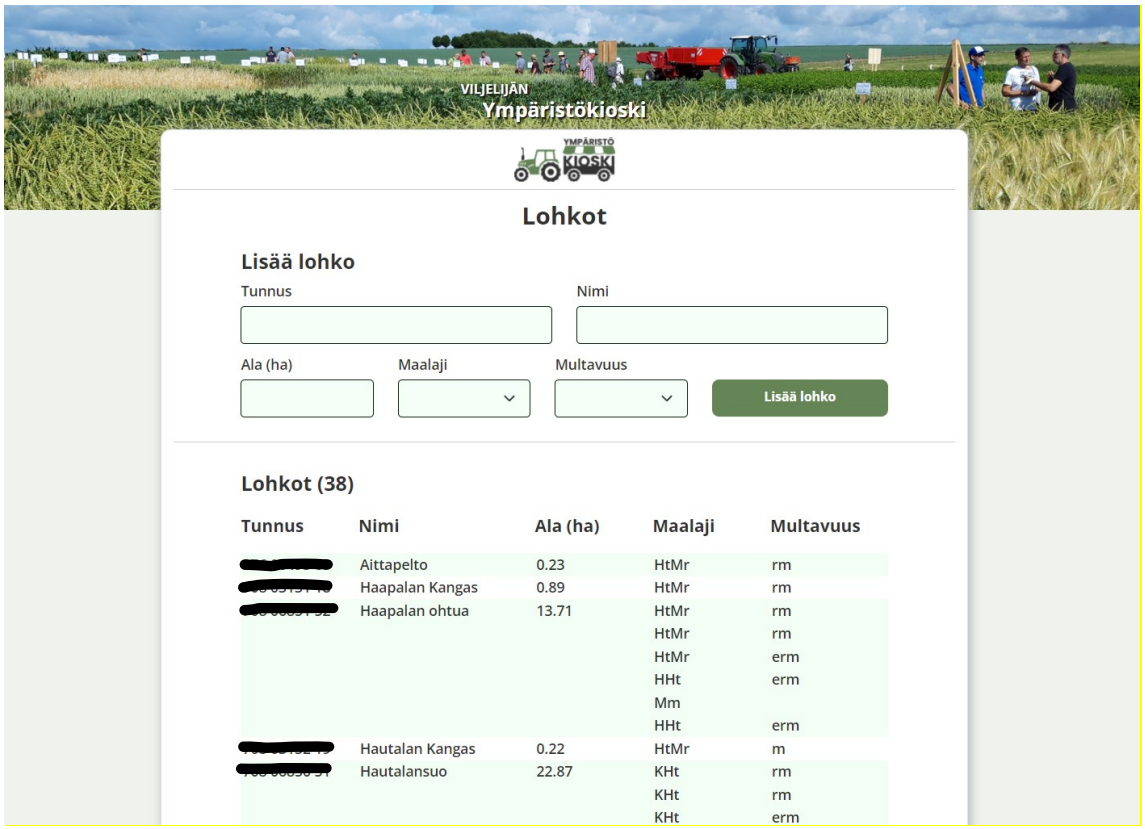
Kuva 19. Valmis rekisteröinti-näkymä

Koska näkymät ovat muodoltaan samanlaisia, tein template-tiedoston, jota käytettäisiin jokaisessa komponentissa, vaihtaen vain sisällön näkymästä riippuen. Näin näkymän perusmuotoa ei tarvitse määritellä jokaisessa komponentissa uudelleen. Kun kirjautumis- ja rekisteröinti-näkymät oli tehty, aloin suunnittelemaan itse sovelluksen ensimmäistä näkymää. Ideanani oli leventää keskialuetta, ja lisätä navigointipainikkeet ylänurkkiin. (Kuva 20).



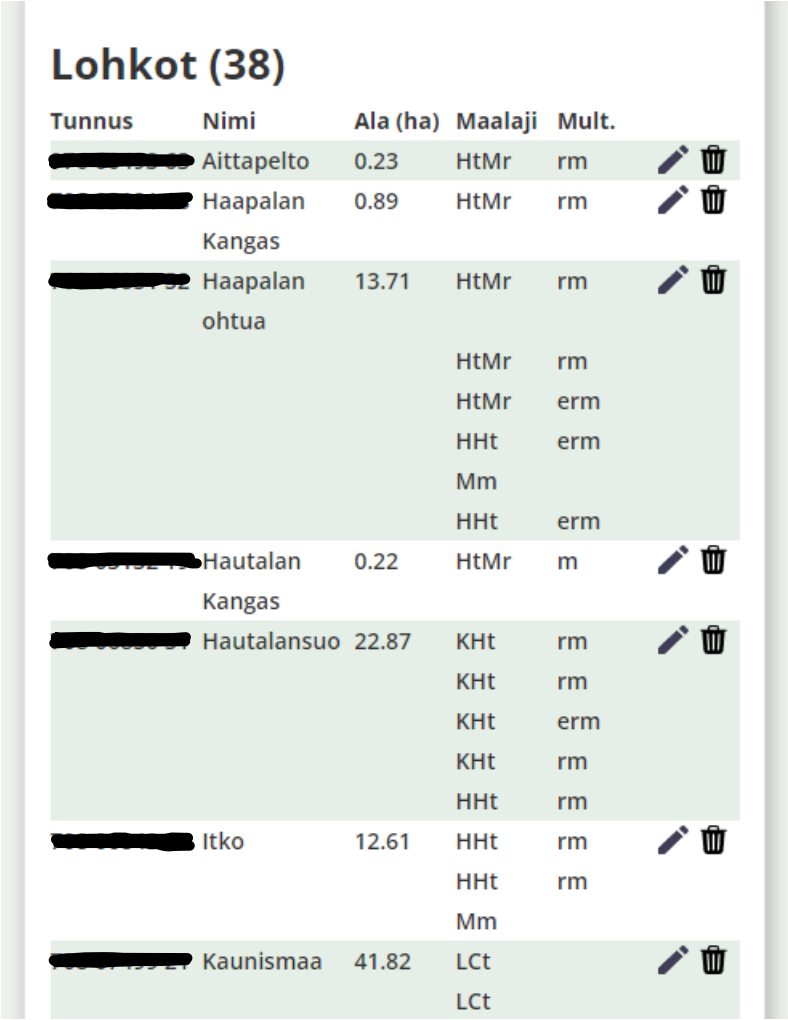
Kuva 20. Lohkot-näkymä kuvitus

Ensimmäistä vaihetta eli koti-näkymää en heti alussa tehnyt, sillä ei ollut vielä varmuutta siitä, mitä se lopulta pitää sisällään, joten aloitin lohkot-näkymästä. Siinä on tarkoitus havainnollistaa tilan lohkot käyttäjälle, ja halutessa lisätä, poistaa, tai muokata niitä. Lisäsin lomakkeen lohkon lisäämistä varten, ja listasin lohkot tauluun käyttäen CSS Gridiä. Kyseessä oli myös ensimmäinen responsiivinen näkymä, joten toteutin näkymästä pöytäkone, tabletti, ja mobiiliversion. Alun perin oli huolia, miten lohkojen tiedot saataisiin listattua niin, että ne näkyisivät selvästi myös mobiililaitteilla, mutta sen toteutettua selvisi, että ne näkyvät hyvin, eikä erilaista listaa tarvitse toteuttaa erikseen mobiilille. Listasin lohkot niin, että joka toisella olisi eri taustaväri selkeyden vuoksi. Kuvista on peitetty lohkojen tunnukset. (Kuva 21).



Kuva 21. Lohkot-näkymä pöytäkoneella

Toteutin pöytäkoneen näkymän niin, että näkymän reunoilla olisi pieni alue tyhjää tilaa. Tabletilla keskialue olisi hieman kapeampi, ja tyhjä alue poistuisi. Mobiililla näkymä taas pienentyisi entisestään, ja rekisteröintikentät menisivät allekkain. Lisäsin lohkoihin muokkaus- ja poistopainikkeet, jolla lohkot voitaisiin yksitellen muokata tai poistaa. Multavuus sarake piti myös lyhentää muotoon ”Mult.”, jotta mobiililla listan sarakkeilla olisi tarpeeksi tilaa (kuva 22).



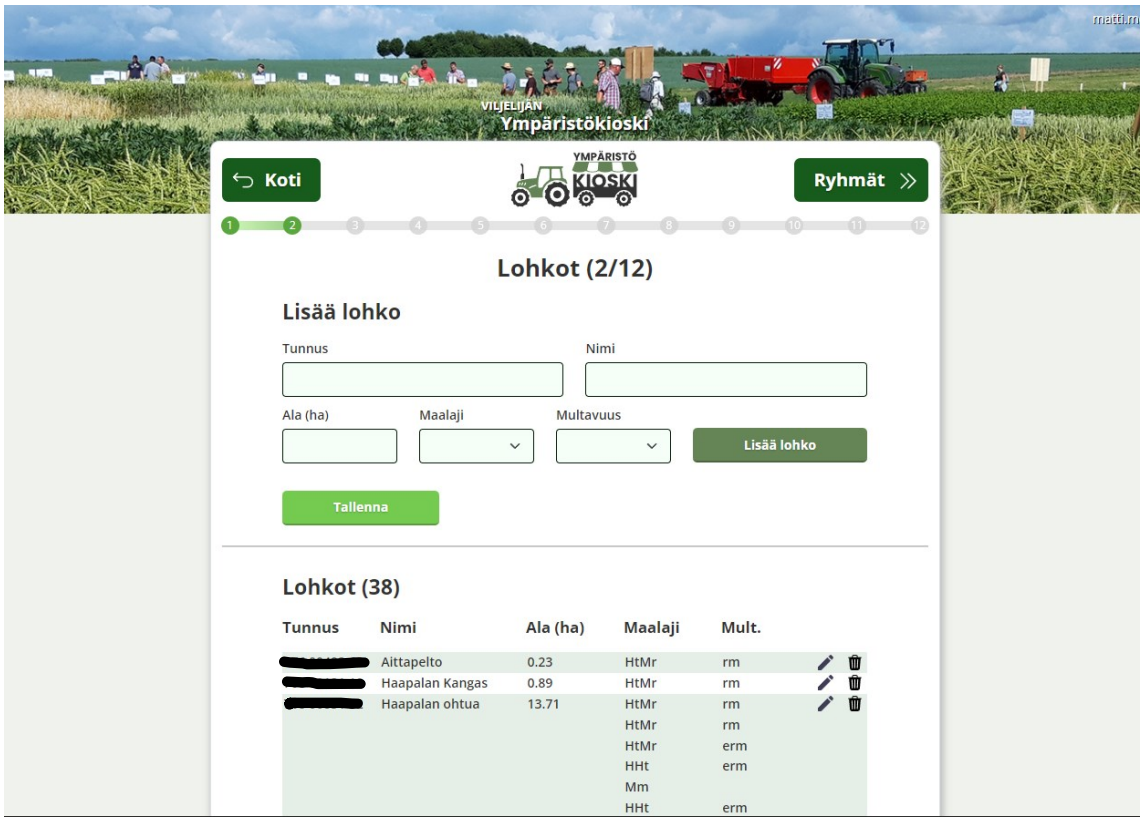
Kuva 22. Lohkojen listaus mobiililla

Toteutin navigoinnin yläkulmissa olevilla painikkeilla, joissa lukee edellisen ja seuraavan näkymän nimet. Pöytäkoneella ja tabletilla näissä ei ollut mitään ongelmaa, mutta mobiilinäytöillä pitemmät näkymien nimet, kuten ”Mahdollisuudet”, oli hankala mahduttaa niin että fontin koko olisi tarpeeksi iso ja painikkeet olisivat sopusuhtaisia. Päätin toteuttaa navigoinnin niin, että mobiililaitteilla painikkeissa lukisi näkymien nimien sijasta edellinen ja seuraava. Tähän tarvitsin uuden template-tiedoston, jonka tulisi tulla käyttöön, kun näytönkooksi havaitaan mobiili. Toteutin sen funktiolla, joka seuraa käyttäjän näytön kokoa, ja lopuksi palautan sopivan templaten, riippuen onko käyttäjän näytön koko yli vai alle määritetyn breakpointin (kuva 23).

```
const [width, setWidth] = React.useState(window.innerWidth);

React.useEffect(() => {
  const handleWindowResize = () => setWidth(window.innerWidth);
  window.addEventListener("resize", handleWindowResize);
  return () => window.removeEventListener("resize", handleWindowResize);
}, []);

const breakpoint = 767.98;

const content = (
  <>···
  </>
)

return (
  width < breakpoint ?
  <TemplateMobile content={content} currentView={"Groups"} />
  : <Template content={content} currentView={"Groups"} />
)
```

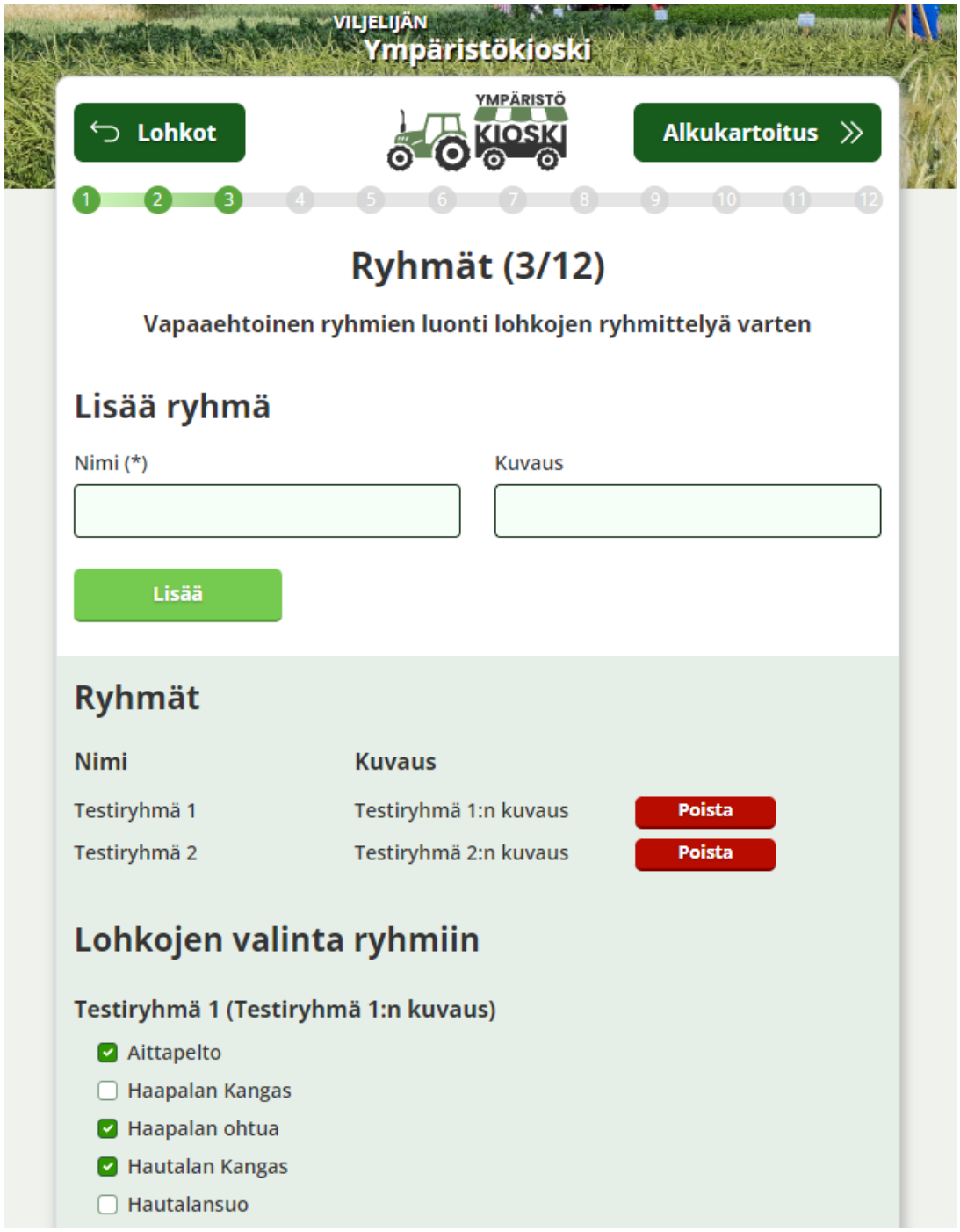
Kuva 23. Näytön koon tarkistajafunktio ja sisällön palautus

Lisäsin ”progress-baarin”, joka havainnollistaa käyttäjälle kuinka pitkällä sovelluksessa käyttäjä on, ja kuinka paljon jäljellä (kuva 24). Sovelluksen nappeihin lisäsin pienen tummennetun alueen nappien alalaitaan, jotta ne eivät näyttäisi niin tylsiltä. Yläkulmaan lisäsin myös käyttäjän tunnuksen ja kirjaudu ulos-painikkeen, sekä sivun alalaitaan hanketekijöiden nimet.



Kuva 24. Valmis lohkot-näkymä

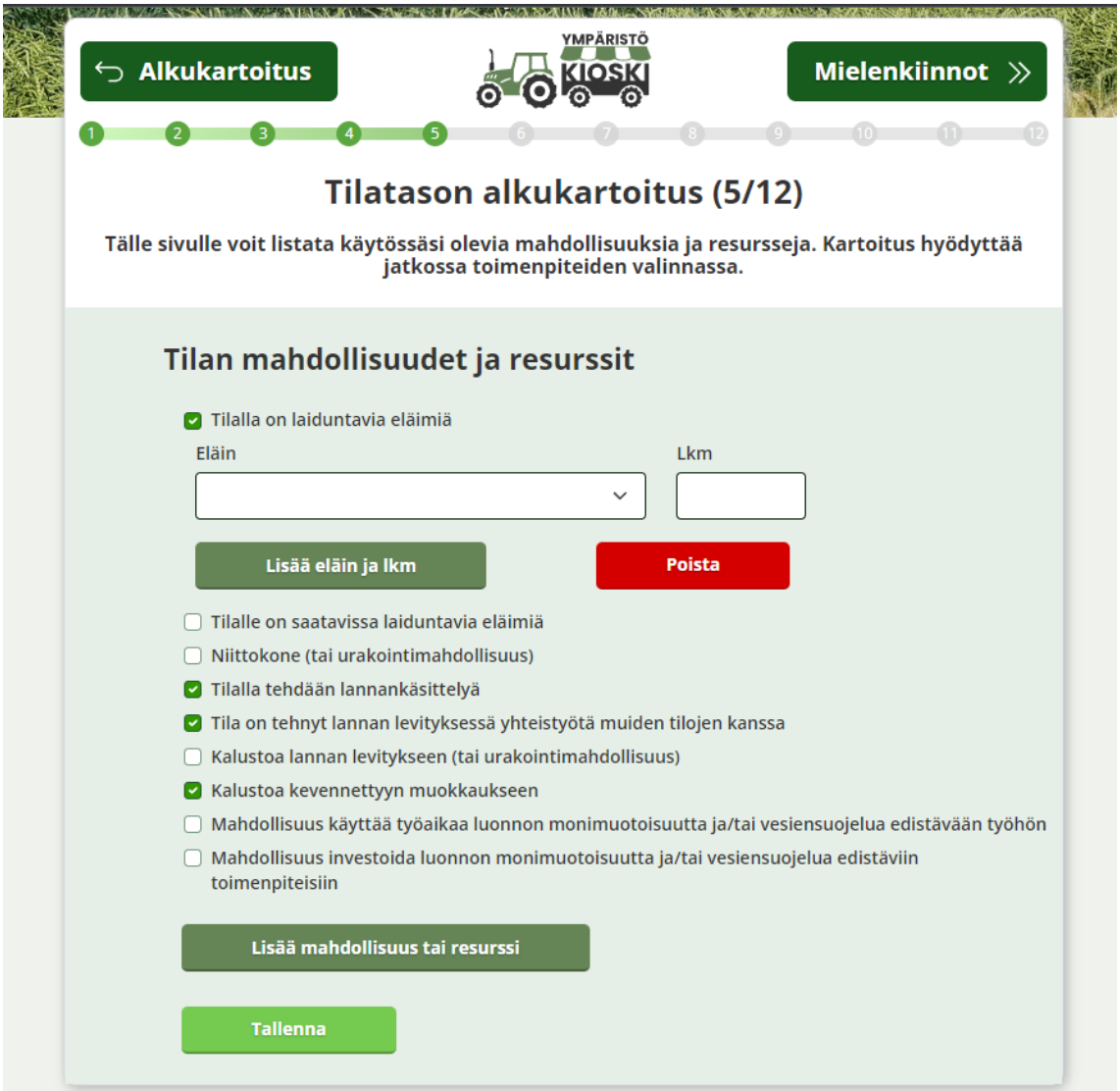
Seuraavaksi lähdin toteuttamaan ryhmät-näkymää. Ryhmät-näkymässä käyttäjä voi vapaavalintaisesti luoda eri ryhmiä lohkoille selkeyden vuoksi. Sisällöltään se oli samantyyppinen kuin lohkot-näkymä, lisäsin alkuun lomakekentät, ja sivun alaosaan listasin ryhmiin valittavat lohkot. Palavereissa oltiin samaa mieltä, että taustaväriä voisi lisätä myös muihinkin näkymiin, että ne eivät olisi vain valkoisia. Tässäkin testasin värien vaihtamista vuorotellen lohkoa kohti, mutta se ei ollut toimiva ratkaisu. (Kuva 25).



Kuva 25. Ryhmät-näkymä tabletti koolla

Seuraavaksi tein tilatason alkukartoitus-näkymän, jossa kerrotaan tilan perustiedot ja listataan sen maalajit. Käyttäjä voi myös kirjoittaa mm. tilan erityisistä luonnonvaroista, tai vesistöalueiden tilasta. Seuraavassa näkymässä tilaa kartoitetaan valitsemalla tilan mahdollisuudet ja resurssit, jotka hel-

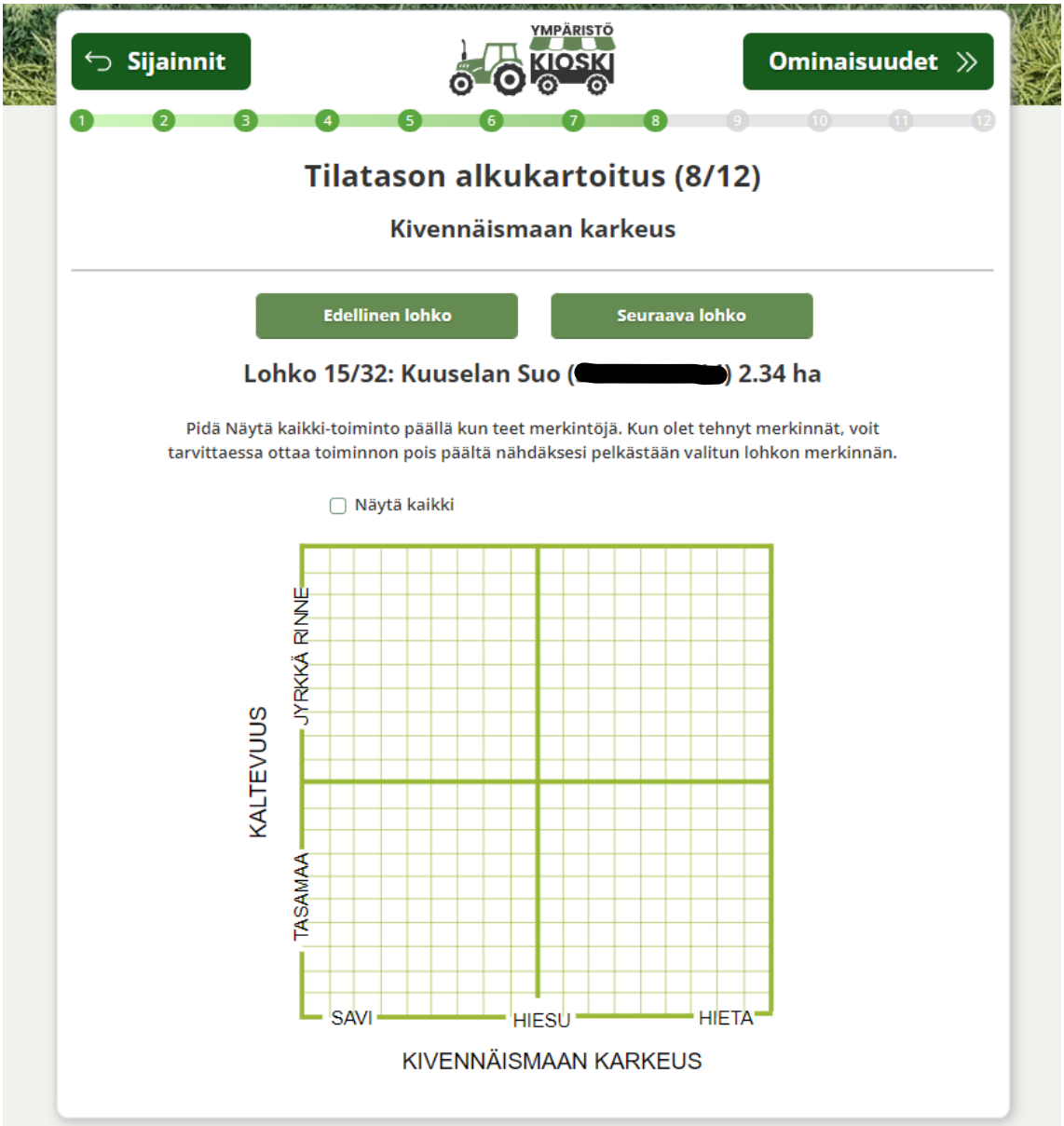
pottavat jatkossa tilan toimenpiteiden valinnassa. Listasin mahdollisuudet ja resurssit, ja jos käyttäjä valitsee ”tilalla on laiduntavia eläimiä”, tulee myös kentät, jossa eläimet voidaan lisätä (kuva 26).



Kuva 26. Tilan mahdollisuudet ja resurssit.

Seuraavassa näkymässä kartoitetaan tilan mielenkiinnon kohteita ympäristöhoitoon ja viljelyn kehittämiseen. Tämä näkymä oli yksinkertainen, tarvitsi vain listata käyttäjää kiinnostavat aiheet samalla tyylillä kuin edellisessä näkymässä. Seuraavissa näkymissä kartoitetaan tilan lohkoja sijaintia vesistöihin nähden, ja kivennäismaan karkeutta. Kummatkin näkymät ovat identtisiä, sisältäen matriisin, joissa vain matriisin ominaisuudet vaihtuvat. Näissäkin näkymissä oli aluksi huolena, onko

matriisi mobiililaitteella käytettävä, mutta niiden tehtyä selvisi, että on. Näissä näkymissä poikkeuksellisesti sijoitin sisällön keskelle, jotta näkymä olisi symmetrinen ja miellyttävämmän näköinen. (Kuva 27).



Kuva 27. Kivennäismaan karkeus

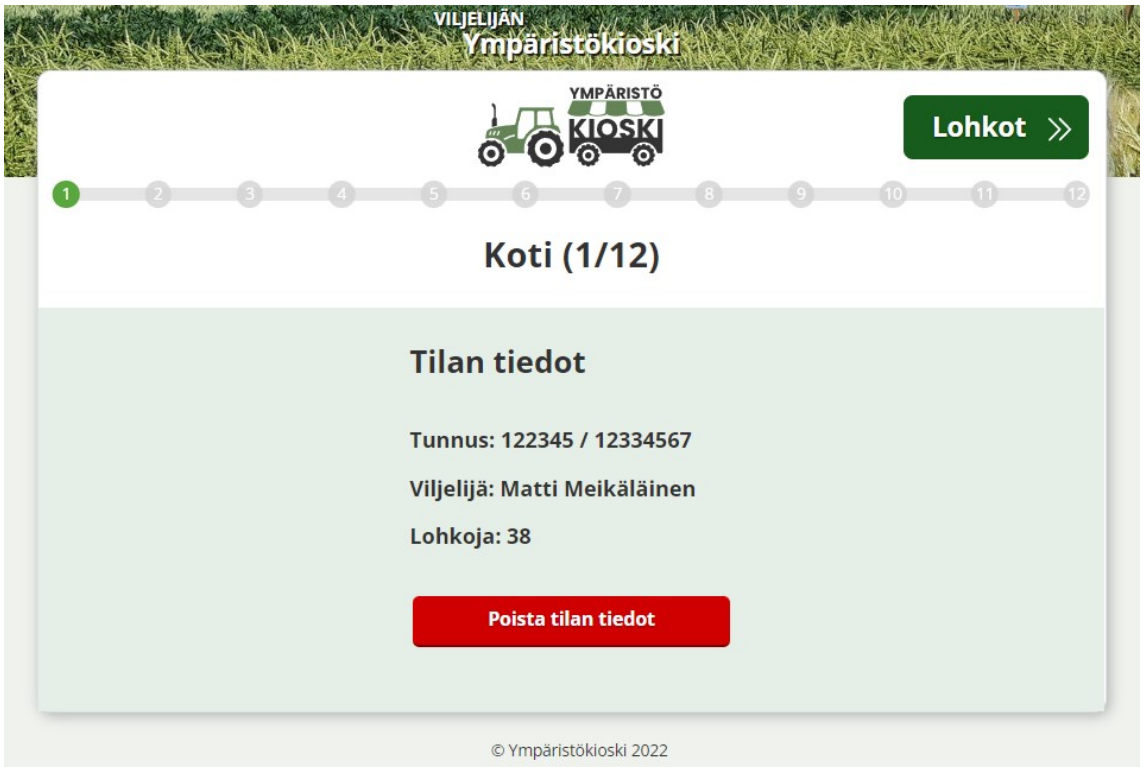
Seuraavassa näkymässä käyttäjä valitsee lohkoille niiden sisältämät ominaisuudet. Tässä käytin hyödyksi Bootstrapin haitarikomponenttia. Lohkot listataan, ja painamalla lohkoa se laajenee, listaten siihen valittavat ominaisuudet. Määritin myös haitarit pysymään auki, vaikka muitakin lohkoja avattaisiin. (Kuva 28).

Kuva 28. Ominaisuudet

Seuraava näkymä on toimenpiteet, jossa käyttäjä voi valita lohkoilleen toimenpiteitä, jotka havainnollistetaan käyttäjälle seuraavassa näkymässä matriisilla. Valinta piti tehdä painikkeilla, joka tuotti aluksi minulle ongelmia muotoilun kannalta. Painikkeiden tekstien pituus vaihtelee hyvin paljon, joten painikkeiden määrittäminen tekstin määrän mukaan ei näyttänyt hyvältä. Toteutin painikkeet tekemällä niistä saman levyisiä, ja keskittämällä tekstin. (Kuva 29).

Kuva 29. Toimenpiteet mobiililla

Lopuksi toteutin kooste-näkymän, jossa tilan lohkoille ehdotetut toimenpiteet listataan, ja palasin koti-näkymään, jossa käyttäjä voi tuoda sovellukseen oman tilansa lohkot, tai käyttää testilohkoja sovelluksen testaamiseen. Poistin turhat navigointipainikkeet ensimmäisestä ja viimeisestä näkymästä selkeyden vuoksi (kuva 30).



Kuva 30. Koti-näkymä

4 POHDINTA

Kehityksen alkuvaiheessa oli ylimääräistä monimutkaisuutta, koska lähdin toteuttamaan käyttöliittymää kloonaamalla repositorion, johon sovelluksen alkuperäinen käyttöliittymä oli tehty Webflow-työkalulla. Se hankaloitti prosessia tarpeettomasti, kun projektikansio oli täynnä tiedostoja ja määrityksiä, joita en tarvinnut. Suhteellisen aikaisessa vaiheessa kumminkin huomattiin, että puhtaasta kansiosta aloittaminen olisi huomattavasti helpompaa käyttöliittymän kehityksen kannalta. Kun tehtäväalueeni oli ainoastaan käyttöliittymä, näkymien rakentaminen puhtaalta pöydältä teki tehtävästä paljon selkeämpää. Myös käyttöliittymän integrointi toimivuudesta vastaavaan koodiin helpottui, kun mitään ylimääräistä ja tarpeetonta koodia ei ollut.

Tavoitteena oli luoda miellyttävän näköinen responsiivinen käyttöliittymä, jossa mielestäni onnistuin hyvin. Sovelluksen mobiiliversion toteuttaminen oli odotettua helpompaa, kun monet ennustetut ongelmat olivatkin turhia huolia. Mobiilikoolla eri huolenaiheet, kuten lohkojen listaus ja matriisit eivät tuottaneet ongelmia, ja teksti pysyi selkeästi luettavissa. Tabletilla ja pöytäkonekoolla ei oikeastaan missään vaiheessa kehitystä tullut kummempia aivopähkinöitä esille, ja kehitys sujui suhteellisen suoraviivaisesti.

Yksi seikka, joka vaivasi päätäni jonkin verran, oli bannerin taustakuvan sijoittaminen oikeaan kohtaan jokaisella näytönkoolla. Taustakuvassa oli valkoisia alueita, jotka bannerin valkoisen tekstin alle sijoittuessaan tekivät tekstistä epäselvää luettavaa. Kun näytönkoko levenee, taustakuva levenee, jolloin myös kuvan korkeus kasvaa suhdekoon säilyttämiseksi, mutta bannerille annetun tilan korkeus pysyy samana. Näin bannerin tekstin yläpuolella olevat valkoiset alueet väistämättömästi sijoittuvat tekstin alle näytönkoon leventyessä. Ideaali ratkaisu olisi varmaan ollut tehdä bannerin tekstille funktio, joka määrittää sen sijainnin suhteutettuna näytönkokoon. Toteutin sen kumminkin lukuisilla breakpointeilla, jotka minun piti selvittää tarkentamalla sivua ulospäin, koska läppärini näyttö ei ollut tarpeeksi iso.

Bootstrap oli käyttöliittymän toteutustapana mielestäni hyvä valinta. Sovelluksessa ei ollut kauheasti eroa näkymien välillä, eikä mitään monimutkaisia näkymiä tarvinnut rakentaa. Näkymien sisällöt sai helposti mahtumaan pienellekin alueelle, ja mobiilikoon aluemäärittelyt olivat Bootstrapillä helposti toteutettavissa. CSS Gridiä käytin lohkojen listauksessa, kun tilaa oli hyvin rajallinen määrä

ja sitä piti hyödyntää tarkasti. Bootstrap ja CSS Grid ovat hyvä yhdistelmä, jolla saadaan yksinkertaisten näkymien toteutuksesta nopeaa, mutta tarvittaessa luotua myös yksityiskohtaiset aluemäärittelyt.

Tyylimäärittelyissä oli jonkun verran toistuvaa koodia, pääosin painikkeiden kannalta. Tulevaisuudessa aion perehtyä enemmän Sassiin, jolla toistuvat tyylimäärittelyt saadaan tallennettua muuttujiin, ja luokkakohtaiset tyylimäärittelyt on helppo toteuttaa muuttujien päälle. Näin koodia on helpompi kierrättää sen sijaan että sitä toistaisi, jolloin tyylitiedostoistakin tulee paljon selkeämpiä ja lyhyempiä.

Sovelluksen kehitys jatkuu myös minun osaltani ainakin syksyyn asti. Hanketekijöiden keskustelun ja itse viljelijöiden sovellustestauksen kautta sovellukseen on saatu kehitysideoita, ja hyviä parannusmahdollisuuksia on tuotu esille. Käyttöliittymään tulee vielä muutoksia ainakin koosteen osalta, johon on tarkoitus lisätä diagrammi maalajien listauksen sijaan, ja toimintojen valinta-näkymään, josta on tarkoitus luoda intuitiivisempi käyttää. Uusia näkymiä pitää vielä luoda, jossa käyttäjä voi mm. muokata omia tietojansa, vaihtaa salasanansa, ja antaa palautetta. Toiminnallisuudesta vastaavaa koodia pitää refraktoroida, ja koodin laatua parantaa eri työkaluilla. Tietokantaoperaatioihin pitää toteuttaa virheidenkäsittelyt, ja tietokannasta tulevien virheilmoitusten dialogi pitää kustomoida.

Hanketekijöiden ja lehtori Pekka Ojalan kanssa työskennellessä pääsin mukaan sovelluksen kehitysprosessiin, ja opin kuinka projektikehitys toimii kulissien takana. Huomioonotettavia asioita on enemmän kuin ulkoapäin saattaa vaikuttaa. Kehityshaasteet, palaute, jatkuvat kehitysideat, ja byrokratia tekivät projektista paljon moniulotteisempaa kuin odotin. Käyttöliittymän kehityksessä luonnollisesti kehityin, ja opin erilaisia toteutusmenetelmiä. Bootstrap, HTML ja CSS ovat nyt hyvin hallussa, ja tästä on hyvä jatkaa mm. sovelluslogiikan pariin.

LÄHTEET

Baltic Sea Action Group. LumoVesi-työkalu. Hakupäivä 13.05.2022. https://carbonaction.org/fi/materials/lumovesi-tyokalu/

Bootstrap. Grid system. Hakupäivä 23.3.2022. https://getbootstrap.com/docs/5.1/layout/grid/

Cohen, Nitsan 2021. React Bootstrap vs Bootstrap – Comparison. Hakupäivä 31.3.2022. https://www.linkedin.com/pulse/react-bootstrap-vs-bootsrap-comparison-nitsan-cohen/

Enge, Eric 2021. Mobile vs. Desktop Usage in 2020. Hakupäivä 12.4.2022. https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage#:~:text=Globally%2C%2068.1%25%20of%20all%20website,total%20time%20on%20site%20globally.

Google 2022. Mobile-first indexing best practices. Hakupäivä 12.4.2022. https://developers.google.com/search/mobile-sites/mobile-first-indexing

Maldonado, Leonardo 2020. When to use Flexbox and when to use CSS grid. Hakupäivä 17.4.2022. https://blog.logrocket.com/flexbox-vs-css-grid/

Mozilla 2022a. Responsive design. Hakupäivä 11.4.2022. https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design

Mozilla 2022b. Flexbox. Hakupäivä 17.4.2022. https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

Mozilla 2022c. Basic Concepts of grid layout. Hakupäivä 18.4.2022. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout

Mozilla 2022d. Layout using named grid lines. Hakupäivä 18.4.2022. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Layout_using_Named_Grid_Lines

Rahkila, Riina 2021. Viljelijän arvokkaan työn tukena. Hakupäivä 31.3.2022. https://www.proagriaoulu.fi/fi/viljelijan-arvokkaan-tyon-tukena/

Tailwind Labs Inc 2022. Utility-First Fundamentals. Hakupäivä 11.4.2022. https://tailwindcss.com/docs/utility-first

Why React-Bootstrap? Hakupäivä 30.3.2022. https://react-bootstrap.github.io/getting-started/why-react-bootstrap/

Xia, Vincent 2017. What is Mobile First Design? Why It's Important & How To Make It? Hakupäivä 11.4.2022. https://medium.com/@Vincentxia77/what-is-mobile-first-design-why-its-important-how-to-make-it-7d3cf2e29d00