OULUN AMMATTIKORKEAKOULU

VERKKOSIVUJEN KÄYTTÖLIITTYMÄSUUNNITTELU JA FRONTEND-KEHITYS TAMATEUR RY:LLE

Santeri Ronkainen
Opinnäytetyö
Syksy 2024
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Laite- ja tuotesuunnittelun suuntautumisvaihtoehto

Tekijä: Santeri Ronkainen
Opinnäytetyön otsikko: Verkkosivujen käyttöliittymäsuunnittelu ja frontend-kehitys tamateur ry:lle
Työn ohjaaja: Meija Lohiniva
Työn valmistumislukukausi ja -vuosi Syksy 2024
Sivumäärä: 31

Tässä opinnäytetyössä toteutettiin verkkosivujen frontend-kehitys Tamateur ry:lle hyödyntäen React-kirjastoa ja JavaScript-ohjelmointikieltä.

Teoriaosuudessa perehdytään käyttöliittymäsuunnittelun (UI) ja käyttökokemussuunnittelun (UX) perusteisiin sekä verkkosivujen peruselementteihin. Lisäksi tarkastellaan mobiiliystävällisen suunnittelun merkitystä ja sen toteuttamista osana projektia. Näitä teoreettisia lähtökohtia sovellettiin käytännössä verkkosivujen kehityksessä.

Työn tuloksena Tamateur ry sai toimivat, selkeästi navigoitavat ja mobiiliystävälliset verkkosivut, jotka vastaavat toimeksiantajan tarpeita.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information Technology
Option of Device and Product design

Author: Santeri Ronkainen
Title of thesis: Website user interface design and frontend development for tamateur ry
Supervisor: Meija Lohiniva
Term and year when the thesis was submitted: Autumn 2024
Number of pages: 31

This thesis focused on developing the frontend of a website for Tamateur ry using the React library and JavaScript programming language.

The theoretical part delves into the fundamentals of user interface (UI) and user experience (UX) design, as well as the essential elements of web development. Additionally, it examines the importance of mobile-friendly design and how it was implemented in the project. These theoretical principles were applied throughout the website development process.

As a result, Tamateur ry received a functional, easy-to-navigate, and mobile-friendly website.

SISÄLLYS

SANASTO

API - (Application Programming Interface) sallii useiden sovellusten kommunikoinnin toistensa kanssa. (1.)

Backend – Sovelluksen taustajärjestelmä, joka käsittelee ja tallentaa tietoja sekä tukee frontendin toimintaa.

CSS - Cascading Style Sheets on kieli, jota käytetään verkkosivujen ulkoasun muotoiluun.

Docker – Mahdollistaa sovellusten ja niiden riippuvuuksien paketoinnin ja ajon eristetyissä ympäristöissä, eli niin sanotuissa konteissa.

Footer - Verkkosivujen alapalkki, jossa yleensä on yhteystietoja tai linkkejä.

Frontend - Verkkosivujen näkyvä osa, jonka käyttäjä näkee ja jonka kanssa on vuorovaikutuksessa.

Header - Verkkosivujen yläpalkki, jossa on yleensä navigointilinkkejä.

HTML - HyperText Markup Language on merkintäkieli, jota käytetään verkkosivustojen luomiseen ja muotoiluun. (2.)

JavaScript - Ohjelmointikieli, jota käytetään verkkosivujen toiminnallisuuden kehityksessä.

JSX - JavaScript syntaksi, jota käytetään komponenttien luontiin Reactissa, muistuttaa hieman HTML-kieltä.

React - JavaScript – kirjasto, jota käytetään käyttöliittymien ja komponenttipohjaisten sovellusten rakentamiseen.

UI - User Interface. Käyttöliittymä, eli kaikki käyttäjän näkemät ja käytettävät elementit, kuten painikkeet ja valikot.

UX – User Experience Design. Käyttäjäkokemussuunnittelu, viittaa yleensä siihen, että miten käyttäjien käyttökokemusta voidaan kehittää sekä parantaa.

1 JOHDANTO

Tämä verkkosivuprojekti tehdään Tamateur ry:lle. Tamateur on Tampereelta lähtöisin oleva tapahtuma, jota on järjestetty vuodesta 2021 lähtien. Tapahtuman ideana on luoda amatööripelaajille helposti lähestyttävä LAN-turnauskokemus. LAN (Local Area Network) -turnauksessa osallistujat kokoontuvat samaan paikkaan kilpailemaan verkkopeleissä, mikä mahdollistaa erilaisten ja intensiivisten pelikokemuksien jakamisen. Verkkosivut halutaan osaksi tapahtumaa helpottamaan järjestäjien toimintaa ja kasvattamaan tapahtuman kuulijakuntaa. Verkkosivuilla tuodaan esille tapahtumaa ja niiden kautta ry:llä on mahdollisuus löytää esimerkiksi uusia yhteistyökumppaneja.

Työssä hyödynnetään nykyaikaisia web-kehitysteknologioita, kuten Reactia, joka mahdollistaa komponenttipohjaisen kehityksen. Verkkosivujen suunnittelussa keskeisiä asioita ovat käyttöliittymäsuunnittelu (UI) ja käyttökokemussuunnittelu (UX), joiden tavoitteena on luoda selkeä, visuaalisesti miellyttävä ja helppokäyttöinen verkkosivusto. Teoriaosuudessa käsitellään tarkemmin käyttöliittymä- ja käyttökokemussuunnittelun periaatteita.

1.1 Työn tavoite

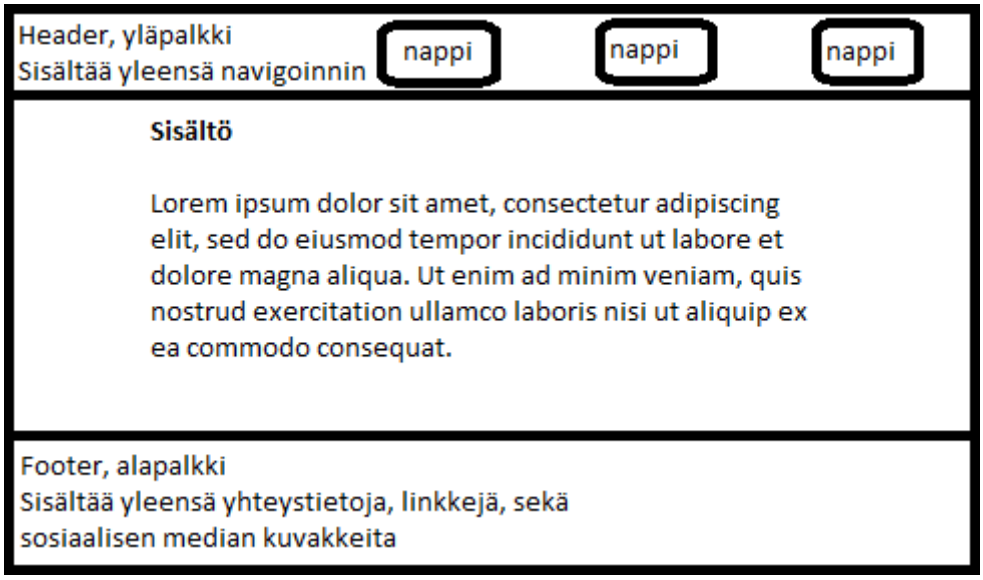
Työn tavoitteena on suunnitella ja toteuttaa verkkosivuston Frontend-kehitys käyttäen ReactJS:n kirjastoja. Tässä käytettävät teknologiat ovat HTML, CSS, sekä JavaScript. Frontend-kehityksen tarkoituksena on tarjota käyttäjille sulava sekä selkeä käyttökokemus. Tärkeimpänä tavoitteena on automatisoida ilmoittautumisprosessi, jotta järjestäjien ei tarvitse pyörittää ilmoitusprosessia manuaalisesti, sekä osallistujat varmistavat paikkansa heti turnaukseen, kun ovat ilmoittautuneet.

2 KÄYTTÖLIITTYMÄSUUNNITTELU

Käyttöliittymäsuunnittelu (UI-suunnittelu) on prosessi, jossa suunnittelijat luovat käyttöliittymiä ohjelmistoihin tai tietokoneistettuihin laitteisiin keskittyen niiden ulkoasuun ja tyyliin. Tavoitteena on suunnitella käyttöliittymiä, jotka ovat käyttäjille helppokäyttöisiä ja miellyttävinä. (3.)

2.1 Verkkosivujen käyttöliittymä

Suurin osa verkkosivuista muistuttaa ulkoasultaan toisiaan, sillä lähes jokaisella verkkosivulla, on samat peruselementit: yläpalkki (header) ja alapalkki, (footer). Kuvassa 1 on esitelty verkkosivumalli, jota moni suunnittelija noudattaa. (4.)



Kuva 1. Verkkosivujen yleinen rakenne

Kuvassa 1 näkyy verkkosivujen tyypillinen rakenne. Yläpalkissa (header) on navigointivalikko, keskellä pääsisältöalue ja alapalkissa (footer) on yhteystietoja, linkkejä sekä sosiaalisen median kuvakkeita. Sisältöalueessa on esimerkkitekstinä ”Lorem ipsum” havainnollistamassa tekstisisällön paikkaa.

2.2 Käyttökokemussuunnittelu

UX-suunnittelu (UX, *user experience*) tai käyttäjäkokemuksen suunnittelu on kokonaisuus, jossa keskitytään suunnittelemaan käyttäjien vuorovaikutusta tuotteiden, palvelujen ja järjestelmien kanssa. Se käsittää käyttäjien tarpeiden, motiivien ja käyttäytymisen ymmärtämisen ja sellaisten ratkaisujen luomisen, jotka ovat sekä toimivia että miellyttäviä käyttää. Lyhyesti sanottuna UX-suunnittelua voidaan soveltaa käytännössä mihin tahansa kokonaisuuteen, johon liittyy ihmisen ja tuotteen tai palvelun vuorovaikutusta. Käyttäjäkokemuksen suunnittelua voidaan soveltaa esimerkiksi alla listattuihin kokonaisuuksiin:

Verkkosivustot: UX-suunnittelun avulla voidaan luoda verkkosivustoja, jotka ovat helppokäyttöisiä ja intuitiivisia. Verkkosivujen vierailijoille tulisi tarjota mahdollisimman helppo kulku heidän tahtomaansa päämäärään. Esimerkiksi verkkokaupoissa korostuu navigoinnin helppous, jotta kävijä löytää nopeasti haluamansa tuotteen.

Mobiilisovellukset: UX-suunnittelua hyödyntäen luodaan mobiilisovelluksia, jotka ovat optimoitu pienille näytöille ja kosketukseen perustuville toiminnoille. Usein mobiilisovellusten käyttäjät haluavat, että sovellusta olisi vaivatonta käyttää yhdellä kädellä ja että sovelluksen päätoiminnot olisivat helposti saatavilla peukalon kohdalla. Tästä syystä esimerkiksi mobiilisovelluksen navigointi on usein mobiilisovelluksissa näytön alaosassa.

Yksi tärkeimmistä osa-alueista on käytettävyys, jolla tarkoitetaan sitä, miten helposti käyttäjät voivat löytää haluamansa tuotteen/palvelun tai muun ratkaisun ongelmaansa. Tähän kuuluvat esimerkiksi intuitiivinen navigointi, selkeät ohjeistukset ja responsiivinen suunnittelu (5.)

3 REACT

React on suosittu avoimen lähdekoodin JavaScript-kirjasto, jota käytetään käyttöliittymien rakentamiseen. Reactin avulla kehittäjät voivat luoda komponentteja, mikä tekee kehitysprosessista tehokkaan ja nopean. Reactin avulla voidaan rakentaa monimutkaisia käyttöliittymiä yhdistämällä itsenäisiä koodiyksiköitä, joita kutsutaan komponenteiksi. Komponentit voivat olla esimerkiksi nappeja, lomakkeita tai koko sivuston osia. Reactin tavoitteena on tehdä käyttöliittymien kehittämisestä modulaarista ja selkeää. (6.)

3.1 Mikä on React

React on JavaScript-pohjainen käyttöliittymien (UI) rakentamiseen tarkoitettu avoimen lähdekoodin kirjasto, jonka kehitti alun perin Jordan Walke, Facebookilla (nykyisin Meta) työskentelevä ohjelmoija. (7.) React otettiin ensimmäistä kertaa käyttöön vuonna 2011 Facebookin uutissyötteessä, ja vuonna 2013 siitä tehtiin avoimen lähdekoodin projekti. Tämän jälkeen React on kasvanut yhdeksi suosituimmista kirjastoista web-kehityksessä, ja sitä käytetään laajasti dynaamisten ja nopeasti päivittyvien käyttöliittymien rakentamiseen. Sen modulaarinen rakenne ja tehokkuus tekevät siitä erinomaisen valinnan nykyaikaisessa web-kehityksessä. (8.)

3.2 Komponenttipohjainen kehitys

React-komponentit ovat ReactJS-sovelluksen rakennuspalikoita. Ne auttavat jakamaan käyttöliittymän pienempiin, uudelleenkäytettäviin osiin, mikä tekee koodin hallinnasta ja ylläpidosta helpompaa. Komponentit voivat olla luokka- tai toimintopohjaisia, ja molemmilla tyypeillä on tärkeä rooli dynaamisten ja interaktiivisten verkkosovellusten rakentamisessa. (9.)

Esimerkkinä komponenttipohjaisesta kehityksestä voidaan käyttää projektissa olevaa Home.jsx komponenttia (kuva 2). Tämä komponentti on vastuussa verkkosivujen etusivun ulkoasusta ja toiminnasta. Kun käyttäjä siirtyy etusivulle, Home.jsx -komponentti renderöidään, jolloin se näyttää sille suunnitellun sisällön.

src > pages > Home.jsx > ...

```jsx
// src/pages/Home.jsx
import './Home.css';
import logo from '../assets/images/tamateur-logo.svg';

const Home = () => {
    return (
        <div className="home-page">
            <p className="welcome-text">
                Seuraava Tamateur CS2 turnaus<br />
                järjestetään tutussa ja turvallisessa Bar & Cafe Lategamessa<br />
                Tampereella 29.11 - 30.11.2024
            </p>
            <img src={logo} alt="Logo" />
        </div>
    );
};

export default Home;

```

Kuva 2. Verkkosivujen etusivu, Home.jsx

Kuva 3 esittää App.jsx -komponentin rakenteen, joka sisältää React Router DOM linkityksen. App.jsx -komponentissa määritellään, miten eri näkymät ja komponentit yhdistyvät ja millaisia reittejä sovelluksella on. Kuvassa näkyy, kuinka React Router mahdollistaa eri sivujen näyttämisen ja navigoinnin verkkosivulla. App.jsx sisältää reittejä, jotka ohjaavat käyttäjän eri sivuille, kuten yhteistyökumppanit, meistä ja ilmoittautumiset.

Kuva 3. App.jsx sisältää projektissa käytettäviä komponentteja

3.3 JSX

JSX (JavaScript XML) on Reactin käyttämä syntaksi, joka yhdistää JavaScriptin ja HTML merkintäkielen. Tämän avulla kehittäjät voivat kuvata käyttöliittymäkomponenttien rakennetta käyttämällä HTML:ää muistuttavaa syntaksia.

Kuva 4. JavaScriptin ja JSX:n eroavaisuudet, ’Hello, world!’

JSX käyttää HTML:n kaltaista syntaksia, mikä tekee siitä helpommin luettavaa. Esimerkiksi kuvassa 4 rivillä 3 on esitelty JSX:n syntaksi ’Hello world’ -ohjelmasta. Kuten kuvasta huomaa, niin syntaksi on selkeä ja visuaalisesti ymmärrettävä, kuten taas rivillä 6 on esitetty JavaScript syntaksi, joka käyttää React.createElement metodia, mikä tekee samasta ohjelmasta hieman pidemmän ja hankalampaa luettavaa.

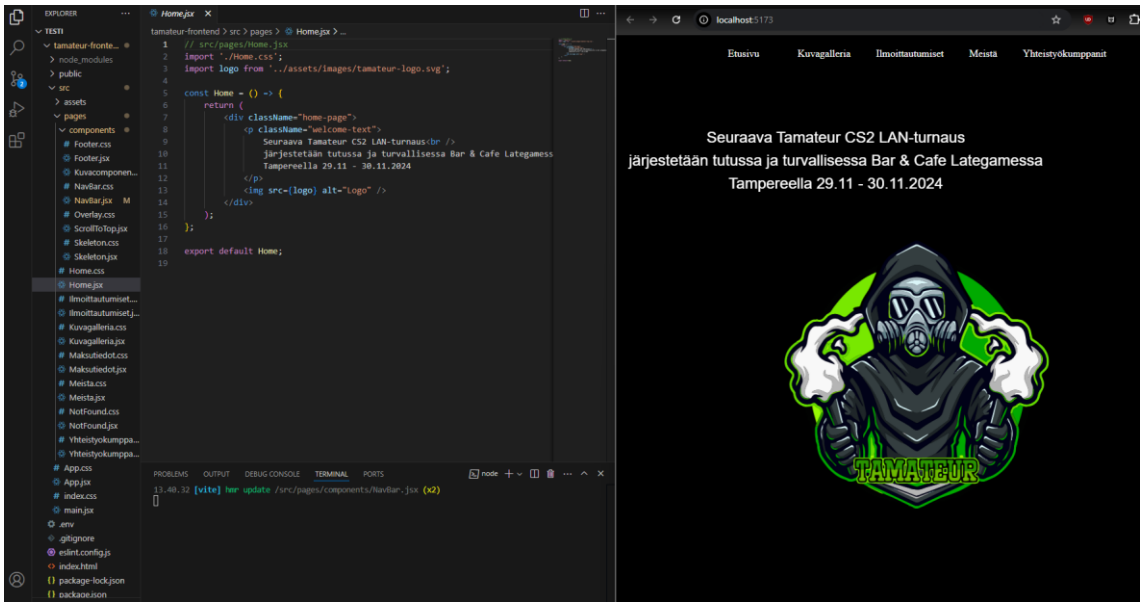
3.4 Visual Studio Code

Visual Studio Code (VS Code) on suosittu Microsoftin kehittämä avoimen lähdekoodin koodieditori, jota käytetään laajasti verkkosivujen kehityksessä, mukaan lukien React-projekteissa. Se tarjoaa laajan valikoiman ominaisuuksia ja laajennuksia, jotka helpottavat koodin kirjoittamista, virheiden etsintää ja projektin hallintaa.

Hyviä laajennuksia verkkosivujen kehityksessä ovat mm:

- React snippets: Tarjoaa valmiita koodinpätkiä React-komponenttien luomiseen. (10).
- Vite: Työkalu, jonka tavoitteena on tarjota nopeampi ja kevyempi kehityskokemus nykyaikaisiin verkkosivuprojekteihin. (11.)
- Npm (Node Package Manager): Javascriptin pakettien hallintatyökalu, joka mahdollistaa työkalujen asentamisen ja hallinnan projekteissa. (12.)

Käytettäessä Viteä voit käynnistää sen kehitysympäristön komennolla: npm run dev. Tämä komento käynnistää paikallisen palvelimen, jossa voi testata verkkosivujen toimintaa, kuten kuvasta 5 näkee.



Kuva 5. Paikallinen kehitysympäristö on käynnistetty komennolla npm run dev.

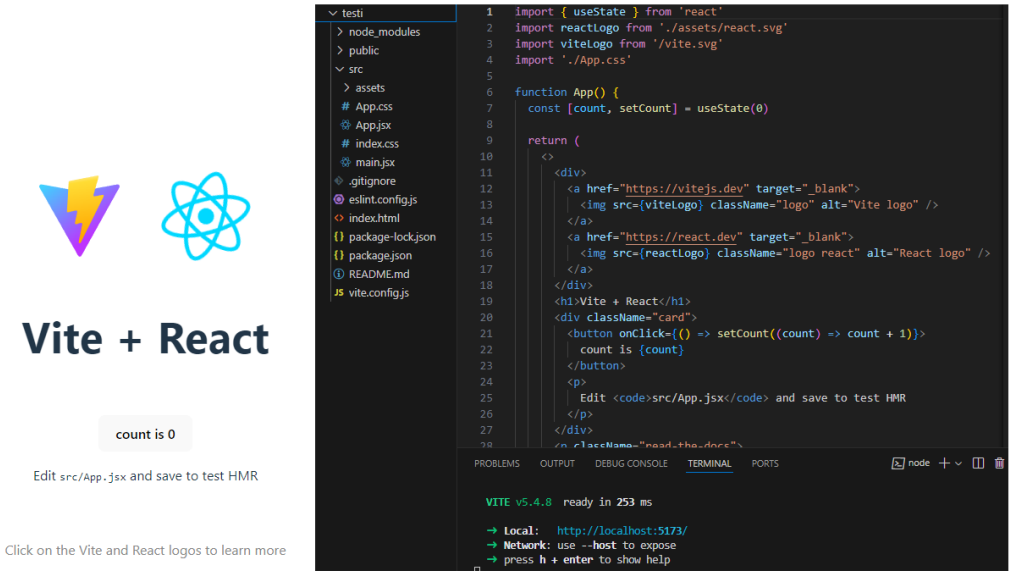
Localhostin ollessa päällä, verkkosivuja on helppo testata sekä kehittää reaaliajassa.

4 VERKKOSIVUJEN TOTEUTUS

Toimeksiantaja määritteli verkkosivuille sisällön, mutta toteutuksessa on muuten vapaat kädet. Verkkosivuilta pitää löytyä seuraavat sivut: etusivut, kuvagalleria, ilmoittautumiset, maksutiedot, meistä sekä yhteistyökumppanit. Lisäksi verkkosivut suunnitellaan siten, että navigointi on käyttäjille mahdollisimman helppoa, jolloin verkkosivujen sisältö on helposti löydettävissä. Verkkosivut optimoidaan myös pienille näytöille, mikä tekee niistä mobiiliystävällisiä ja käytettäviä kaikilla laitteilla. Jokaiselle sivulle tulee omat JSX- ja CSS-komponentit, jotka sisältävät kaikki tarvittavat koodit. Verkkosivujen teossa hyödynnetään teoriaosuudessa esiteltyä asioita (4).

Projekti aloitettiin VS Codessa komennolla npm run vite@latest (13), joka asentaa uusimman version Vitestä sekä kaikki tarvittavat työkalut. Tämä komento luo projektin perusrakenteen ja lisää kaikki riippuvuudet, kuten Reactin ja React DOMin, sekä myös määrittelee kehitysympäristön ja tuotantoon tarkoitetut komennot (npm run dev, npm run build ja npm run preview). Näiden avulla projektin kehittäminen ja testaaminen onnistuu helposti.

Kuvassa 6 kyseinen komento on syötetty VS Coden terminaaliin ja npm run dev komennolla saadaan projektin pohja auki, mikä vahvistaa asennuksen onnistuneen.

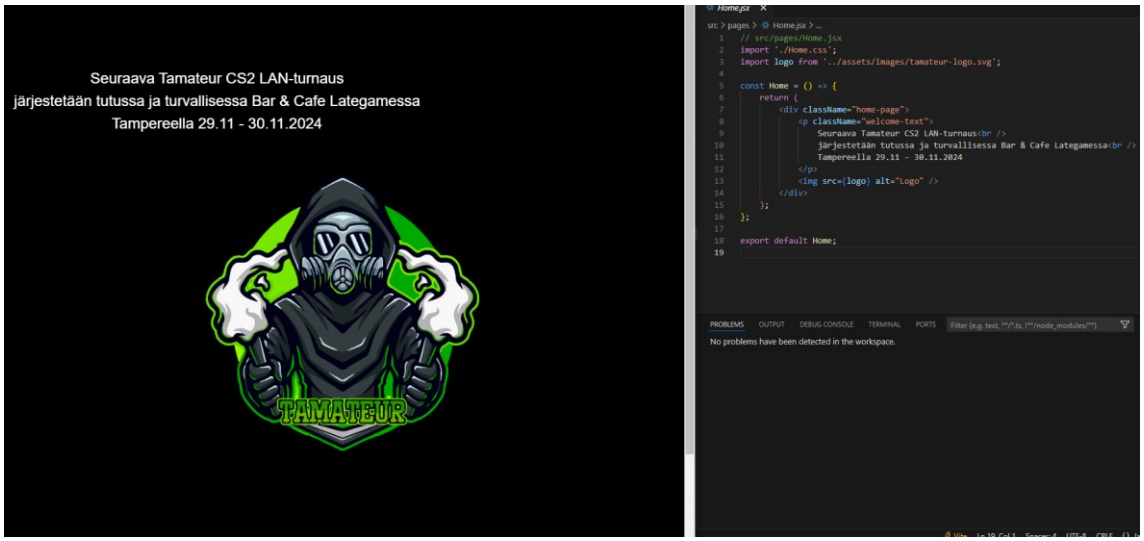


Kuva 6. npm run vite@latest komento loi valmiin projektipohjan ja avasi sen localhost-ympäristössä.

Projektin tekeminen aloitettiin tyhjentämällä valmiina olevia komponentteja, sekä tekemällä pages-kansio, johon kaikki verkkosivuilla tarvittavat sivut sijoitetaan. Varsinainen tekeminen alkaa verkkosivujen etusivusta, jonka komponentti on nimeltään Home.jsx.

4.1 Etusivu

Etusivulle ideana on saada seuraavan tapahtuman ajankohta, sekä ry:n logo näkyville, eli aika yksinkertainen etusivu kyseessä. Itse koodissa ei ole kovin ihmeellisiä toimintoja, joka on esitelty kuvassa 7.



Kuva 7. Home.jsx komponentin koodi, muotoilu tulee Home.css komponentista, joka on tuotu komennolla import './Home.css';

Jotta logolle saadaan hieman eloa, voidaan Home.css komponenttiin lisätä **@keyframes liike**, jolla saadaan zoomausefekti logolle. Tämä efekti on esitelty alla olevassa kuvassa.

```
/* Keyframes for zoom effect */
@keyframes liike {
  0% {
    transform: scale(1); /* Initial size */
  }
  100% {
    transform: scale(1.1); /* Zoom in effect */
  }
}
```

Kuva 8. Home.css zoomausefekti

Animaatio toistuu ikuisesti (loop) koska sille on määritelty **animation: liike 5s infinite;** jolloin efekti toistuu viiden sekunnin jaksoissa logon vaihdellessa koon 1 ja 1.1 välillä käyttäjän ollessa etusivulla.

4.2 Header, yläpalkki

Seuraavaksi aloitin yläpalkin, eli navigointivalikon (navbar) tekemisen. Tähän sijoitetaan linkit kaikille sivuille, joita verkkosivustolla tarvitaan. Navigointipalkista saadaan kiinteä käyttämällä CSS-ominaisuuksia **position: sticky;** ja **top 0;** näiden avulla navigointipalkki pysyy koko ajan näkyvissä, kun käyttäjä vierittää sivua alaspäin. Tämä on hyvä olla, sillä se parantaa käyttökokemusta (14) ja helpottaa toisille sivuille siirtymistä, kun palkki on aina näkyvillä. Koodiin on myös tehty hampurilaisvalikko, joka mahdollistaa valikon supistamisen mobiililaitteilla. Tästä lisää kappaleessa 5.

Kuva 9. Navigointivalikon toteutus ja ulkoasu

4.3 Footer, alapalkki

Kun yläpalkista tuli valmista, niin seuraavaksi aloin tekemään alapalkkia, eli footeria. Toimeksiantajan toiveena oli saada y-tunnus, yhteystiedot sekä sosiaalisen median kanavat näkyville verkkosivuille, joten ne ovat omasta mielestäni parhain laittaa tähän, sillä ne ovat näkyvillä jatkuvasti käyttäjille.

Komponenttia lähdin rakentamaan teorian mukaisesti (4) (kuva 1) eli alapalkki on aina näkyvissä sivujen alhaalla. Lähdin tutkimaan Reactin lisäosia, sillä pelkkä teksti sosiaalisen median kanavalla on aika tylsähkö. Löysin React-icons (15) lisäosan, josta löytyy valmiit ikonit sosiaalisen median alustoihin. Tamateur Ry:llä

on tällä hetkellä Twitch- ja discord-kanavat käytössä ja näille molemmille löytyy valmiit ikonit React-iconien kautta. Tämä lisäosa saadaan asennettua komennolla **npm install react-icons**, sekä asennuksen jälkeen molemmat saadaan tuotua komponentille käyttäen import { FaTwitch, FaDiscord } from 'react-icons/fa';

Footer.jsx

tamateur-frontend > src > pages > components > Footer.jsx > ...

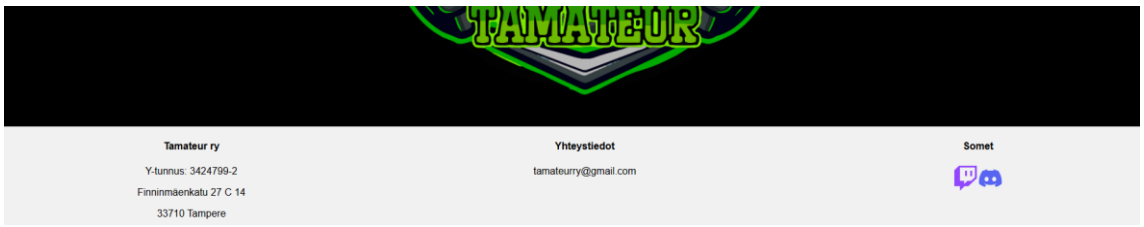
```jsx
import './Footer.css';
import { FaTwitch, FaDiscord } from 'react-icons/fa'; // react-icons

const Footer = () => (
  <footer className="footer">
    <div className="footer-section">
      <h4>Tamateur ry</h4>
      <p>Y-tunnus: 3424799-2</p>
      <p>Finninmäenkatu 27 C 14</p>
      <p>33710 Tampere</p>
    </div>
    <div className="footer-section">
      <h4>Yhteystiedot</h4>
      <p>tamateurry@gmail.com</p>
      <p></p>
    </div>
    <div className="footer-section">
      <h4>Somet</h4>
      <div className="social-media-icons">
    <a href="https://www.twitch.tv/tamateurtv" target="_blank" rel="noopener noreferrer">
      <FaTwitch size="2.5em" style={{ color: '#9146FF' }} />
    </a>
    <a href="https://discord.gg/UHRk5hsyPH" target="_blank" rel="noopener noreferrer">
      <FaDiscord size="2.5em" style={{ color: '#5865F2' }} />
    </a>
  </div>
      </div>
    </footer>
  );

export default Footer;
```

Kuva 10. Twitch ja Discord ikonit on tuotu komponentille riveiltä 20–24

Kuva 10 pitää sisällään koko JSX-koodin, josta näkee ry:n yhteystiedot sekä sosiaalisen median kanavat. Linkit on upotettu ikonien sisälle, mikä tekee niistä käyttäjäystävällisiä ja helposti käytettäviä.



Kuva 11. Valmis alapalkki

Alapalkki on mielestäni yksinkertainen (kuva 11) mikä tekee siitä miellyttävän näköisen. Verkkosivujen tausta on kokonaan musta, joten valkoinen väritys sopii tälle hyvin. Ry:n yhteystiedot ovat selkeästi näkyvillä, sekä sähköpostin löytää samasta paikkaa kätevästi. Sosiaalisen median kanaville pääsee klikkaamalla ikoneita.

4.4 Kuvagalleria

Seuraavaksi aloitin kuvagallerian tekemisen, joka on keskeinen osa verkkosivustoa, sillä se tarjoaa käyttäjille mahdollisuuden tutustua tapahtuman visuaalisiin hetkiin ja tunnelmiin. Kuvagallerian toteutuksessa mietittiin toimeksiantajan kanssa sitä, että minne tapahtuman kuvat laitettaisiin. Päädyimme siihen tulokseen, että kolmannen osapuolen palvelut ovat hyvä idea tähän, sillä tämä säästää oman palvelimen kaistaa, tallennustilaa, eikä vaadi toisilta ry:n jäseniltä teknistä osaamista, sillä kuvat voidaan vain ladata pilvipalveluun ja ne tuodaan verkkosivulle automaattisesti API:n avulla.

Lähdin tutkimaan erilaisia ilmaisien kuvien pilvipalveluita ja ensimmäisenä mieleen tuli Imgur (16) mutta tämä jäi vain tutkinnan tasolle, sillä hieman lueskeltuani Imgurin API:n toimintaa ja jostain syystä Imgurissa ei ole mahdollista localhostin kautta hakea kuvia, joka tekee sitten koodin testaamisesta hankalaa.

Pilvipalveluiden toimintoja tutkiessani löysin Flickr (17), jossa on todella hyvät ohjeet API:n käyttöönottoon. Kuvasta 5 huomaa sen, että navigointipalkkiin on tehty linkki jo valmiiksi, mutta varsinaiset komponentit uupuvat, eli täytyi luoda komponentit kuvagalleriaa varten. Nyt kun lisäosat ovat tulleet hieman tutuiksi, niin etsin tälle valmiin komponenttipohjan, jota käyttää kuvagalleriassa. Tähän löytyi Yet Another React Lightbox komponentti (18) jonka pohjalta toteutin tämän.

Koodissa keskeisessä roolissa on fetchImages-funktio, jonka avulla haetaan kuvat Flickr API:n kautta. Tämä asynkroninen funktio käyttää Axios-kirjastoa HTTP-pyyntöjen tekemiseen, mikä helpottaa tietojen hakemista ulkoisista lähteistä. Kuvassa 12 riveillä 16–18 käytetään API-avaimia, jotka ovat env-tiedostossa tallessa. Tämä pitää huolen siitä, että avaimet ei vuoda julkisiksi.

```
const fetchImages = async (page) => {
  try {
    const response = await axios.get('https://api.flickr.com/services/rest/', {
      params: {
        method: 'flickr.photosets.getPhotos',
        api_key: import.meta.env.VITE_FLICKR_API_KEY, // env variable
        photoset_id: import.meta.env.VITE_FLICKR_PHOTOSET_ID, // env variable
        user_id: import.meta.env.VITE_FLICKR_USER_ID, // env variable
        format: 'json',
        nojsoncallback: 1,
        page: page,
      },
    });

    const photos = response.data.photoset.photo;
    const urls = photos.map(photo => {
      return `https://live.staticflickr.com/${photo.server}/${photo.id}_${photo.secret}_b.jpg`;
    });

    setImageUrls(prevUrls => {
      const newUrls = urls.filter(url => !prevUrls.includes(url));
      return [...prevUrls, ...newUrls];
    });
    setLoading(false); // Set loading to false after images are fetched
  } catch (error) {
    console.error('Error fetching images:', error);
    setLoading(false); // Set loading to false even if there's an error
  }
};
```

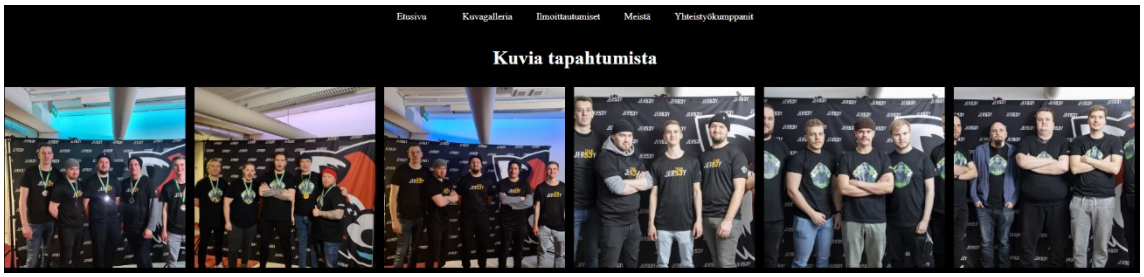
Kuva 12. FetchImages funktio

Kuvagalleria komponentin ollessa valmis, toimeksiantajalta tuli muutama kommentti koodin. Tärkeimpänä kehitysehdotuksena oli se, että kuville voisi tehdä jonkinlaisen latausanimaation, ettei ne vain ilmestyisi yhtäkkiä sivulle. Tämä tuo lisää käyttömukavuutta ja tekee gallerian visuaalisesti miellyttävämmän.

Latausanimaatiota varten valitsin React Loading Skeleton -lisäosan (19), jonka avulla kuville saa latausanimaation ennen niiden latautumista. Kuvassa 13 on React Loading Skeleton toiminnassa, jolloin animaatio on näkyvissä kuvien latautuessa.

Kuva 13. React loading skeleton, kuvat latautumassa

Kun kuvat on ladattu, ne näkyvät kuvagalleriassa kuvassa 14. Kuvia klikkaamalla avautuu lightbox, jossa voidaan tarkastella kuvia suurempina, mikä parantaa gallerian käytettävyyttä. Kuvassa 15 on esimerkki avautuneesta kuvasta suuremmassa koossa.



Kuva 14. Kuvat latautuneet kuvagalleriassa

Kuva 15. Lightbox avattuna ja kuva näkyy suurempana

4.5 Ilmoittautumiset

Toimeksiantaja antoi valmiiksi backend-tiedostot, jotka mahdollistavat ilmoittautumislomakkeen ja sen kehittämisen sekä testaamisen paikallisesti. Backendistä löytyy tärkeimpinä asioina email, real_name ja player_name, sekä virheilmoitukset, jotka voidaan palauttaa frontendille virhetilanteissa.

Ilmoittautumislomaketta ja backendin toimintoja voidaan testata Dockerin avulla, joka käynnistää paikallisen palvelimen ja luo automaattisesti tarvittavat asiat, kuten tietokannan. Tähän tuli hyvät ohjeet toimeksiantajalta, sillä ei tarvitse tehdä

mitään muuta kuin käynnistää docker ja käynnistää paikallinen backend komennolla ”**docker compose up -d**”. Backendiin on tehty FastAPI ja se toimii palvelimena, joka välittää viestit frontendin ja PostgresSQL-tietokannan välillä, jonka paikallisena osoitteena toimii http://localhost:8080/registrations/. Tätä osoitetta voidaan kutsua API:lla ilmoittautumislomakkeessa, joka vie sitten tiedot tietokantaan, jos ilmoittautuminen hyväksytään.

Docker on avoin alusta sovellusten kehittämiseen, jakeluun ja ajamiseen. Docker mahdollistaa sovellusten erottamisen infrastruktuurista, jolloin ohjelmiston toimittaminen on nopeampaa. Dockerin avulla voit hallita infrastruktuuria samalla tavalla kuin sovelluksia. Hyödyntämällä Dockerin menetelmiä koodin toimittamiseen, testaamiseen ja käyttöönottoon voit merkittävästi lyhentää koodin kirjoittamisen ja sen tuotantoon viemisen välistä viivettä. (20.)

Teoriaosuudessa käsiteltiin käyttöliittymäsuunnittelun ja käytettävyyden tärkeyttä. Käyttöliittymäsuunnittelu (UI-suunnittelu) on prosessi, jossa suunnittelijat luovat käyttöliittymiä ohjelmistoihin tai tietokoneistettuihin laitteisiin keskittyen niiden ulkoasuun ja tyyliin. Tavoitteena on suunnitella käyttöliittymiä, jotka ovat käyttäjille helppokäyttöisiä ja miellyttäviä. Käyttökokemussuunnittelu (UX-suunnittelu) keskittyy käyttäjien tarpeiden, motiivien ja käyttäytymisen ymmärtämiseen, ja luo ratkaisuja, jotka ovat sekä toimivia että miellyttäviä käyttää (4).

Näitä periaatteita sovellettiin ilmoittautumislomakkeen suunnittelussa siten, että lomakkeen kentät on aseteltu loogisesti ja helposti ymmärrettävästi. Kuvassa 16 esitetty ilmoittautumislomake on suunniteltu siten, että jokaisella sarakkeella on omat kenttänsä: sähköposti, etu- ja sukunimi ja pelinick. Tämä selkeä ja looginen asemointi auttaa käyttäjiä täyttämään lomakkeen nopeasti ja virheettömästi. Käytettävyys huomioitiin myös virheilmoitusten selkeydessä ja informatiivisuudessa, jotta käyttäjät voivat helposti korjata mahdolliset virheet ja jatkaa ilmoittautumista sujuvasti.

Lähdin rakentamaan ilmoittautumislomaketta backendissä mainittujen kohtien mukaan, eli kaavakkeessa pitää olla kentät sähköpostille, etu- ja sukunimelle sekä pelissä käytettävä nimelle, eli pelinickille.

Kuva 16. Turnauksen ilmoittautumislomake

Ilmoittautumislomake (kuva 16) on tehty siten, että käyttäjän on täytettävä kaikki kentät ennen lomakkeen lähettämistä. Tässä on käytetty HTML5 required attribuuttia (21), joka varmistaa sen, että kaikkien kenttien pitää olla täynnä, jotta turnaukseen voidaan ilmoittautua. Lisäksi ilmoittautumislomakkeella on backendissä määritelty virheidenkäsittelyä, jotka ilmoittavat käyttäjille mahdollisista virheistä lomakkeella. Esimerkiksi jos käyttäjä koittaa ilmoittautua turnaukseen samalla sähköpostiosoitteella useasti, lomake antaa virheilmoituksen (409) ”already exists!”, joka kertoo, että kyseinen sähköposti on jo rekisteröity turnaukseen.

Kun ilmoittautuminen on hyväksytty, käyttäjä ohjataan automaattisesti maksutiedot sivulle (kuva 17), josta käyttäjä löytää tarvittavat tiedot maksun suorittamista varten. Ilmoittautumislomake tyhjentyy automaattisesti, kun käyttäjä on ilmoittautunut, jotta käyttäjän tiedot ei jää lomakkeelle.

```
34          // Redirect to the Maksutiedot page if registration is successful
35          navigate('/maksutiedot');
36          // Reset the form fields
37          setEmail('');
38          setRealName('');
39          setPlayerName('');
```

Kuva 17. Ilmoittautumiset.jsx komponentin koodinpätkä, rivillä 35 viedään käyttäjä maksusivulle, jos ilmoittautuminen hyväksytään.

5 TAMATEUR

Kuten kappaleen 4.2 lopussa on mainittu, että yläpalkin koodiin on tehty hampurilaisvalikko, joka mahdollistaa sen, että verkkosivuja on miellyttävä käyttää puhelimella. Toimeksiantaja vinkkasi tästä hampurilaistyylisestä valikosta, joten lisäsin sen navbar.jsx (yläpalkki) komponenttiin (kuva 18).

```jsx
const NavBar = () => {
  const [isOpen, setIsOpen] = useState(false);

  const toggleMenu = () => {
    setIsOpen(!isOpen);
  };

  const closeMenu = () => {
    setIsOpen(false);
  };

  return (
    <nav className="navbar">
      {/* burger Menu icon */}
      <div className={`menu-icon ${isOpen ? 'open' : ''}`} onClick={toggleMenu}> {/* burger menu animation */}
        <div className="bar"></div>
        <div className="bar"></div>
        <div className="bar"></div>
      </div>

      {/* Navbar Menu */}
      <ul className={`navbar-menu ${isOpen ? 'open' : ''}`}>
        <li className="navbar-item">
          <Link to="/" className="navbar-button" onClick={closeMenu}>Etusivu</Link>
        </li>
        <li className="navbar-item">
        </li>
        <li className="navbar-item">
          <Link to="/kuvagalleria" className="navbar-button" onClick={closeMenu}>Kuvagalleria</Link>
        </li>
        <li className="navbar-item">
          <Link to="/Ilmoittautumiset" className="navbar-button" onClick={closeMenu}>Ilmoittautumiset</Link>
        </li>
        <li className="navbar-item">
          <Link to="/meista" className="navbar-button" onClick={closeMenu}>Meistä</Link>
        </li>
        <li className="navbar-item">
          <Link to="/yhteistyokumppanit" className="navbar-button" onClick={closeMenu}>Yhteistyökumppanit</Link>
        </li>
      </ul>
      {isOpen && <div className="overlay open" onClick={closeMenu}></div>}
    </nav>
```

Kuva 18. Navbar.jsx komponentin toteutus

Navigointibaarin komponentti käyttää useState-hookia, joka määrittää sen, että onko valikko auki tai kiinni. ToggleMenu funktio aukaisee valikon, kun taas closeMenu varmistaa sen, että valikko menee kiinni, kun painaa linkkiä tai valikon ulkopuolelta. Demonstroidakseen tämän toiminnon, kuva 19 näyttää mobiilinäkymän etusivulla, oikeassa yläkulmassa näkyy hampurilaisvalikon painike, joka avaa ja sulkee navigointivalikon käyttäjän toimintojen mukaisesti.

Kuva 19. Verkkosivun näkymä puhelimella, valikko kiinni.

Valikon ollessa kiinni, navigointilinkit eivät ole näkyvillä, mutta kun valikkoa painaa (kuva 20) valikko aktivoituu, jolloin kaikkiin linkkeihin pääsee käsiksi.

Kuva 20. Verkkosivujen näkymä puhelimella, valikko aukaistu.

5.1 Verkkosivujen aukaisu ja palaute

Verkkosivut saatiin otettua käyttöön ja tärkeimpänä, eli ilmoittautumiset saatiin tämän myötä myös aukaistua. Ilmoittautumisten ollessa auki, laitoin osallistujille viestiä, että kertokaa jos ilmenee jotain ongelmia tai on antaa palautetta verkkosivuista. Navigoinnista tuli viestiä, että toimii puhelimella ja tietokoneilla hyvin, samoten etusivulla olevassa animaatio on hyvä. Ainoat pienet viat tulivat vain puhelimilla ilmi, jos oikein bugeja etsi. Kun androidilla vetää sivuja alaspäin niin näkyy valkoinen viiva yläpalkin pohjalla, se tulee siitä, kun HTML tausta on valkoinen eikä musta. Myös pientä ongelmaa ilmaantui backendissä maksutietojen kanssa, backend generoi QR:n sekä virtuaaliviivakoodit kun käyttäjä ilmoittautuu,

niin esimerkiksi virtuaaliviivakoodin viestikenttää tulee automaattisesti pelinick, josta ry sitten tunnistaa, että kuka on maksanut osallistumismaksun. Mutta jostain syystä QR-koodit taikka virtuaaliviivakoodit eivät toimineet kaikilla, vaikka testauksessa toimi hyvin X pankin kanssa. Joillain osallistujilla virtuaaliviivakoodit toimivat, mutta sitten toisella ei toiminut, vaikka oli sama pankki käytössä. Maksutietoihin tuli tehtyä myös sarakkeet mistä voi kopioida maksutiedot, niin ei jäänyt maksaminen siitä kiinni.

6 POHDINTA

Opinnäytetyön aihe tuli puheeksi 2024 kesän lopulla ry:n kanssa. Ilmoitin heille olevani halukas ainakin kokeilemaan verkkosivujen tekemistä, vaikka aikaisempaa kokemusta Reactin käytöstä ei ollut. Tutkin Reactin toimintoja ennen opinnäytetyön aloittamista ja tulin siihen tulokseen, että voin ottaa haasteen vastaan ja lähteä tekemään verkkosivuja. Ry:ssä on kuitenkin useita koodareita mukana, joten apua ongelmiin oli saatavilla, niin ei tarvinnut yksin miettiä, että kun asiat eivät toimineet.

Ongelmia oli aika paljonkin verkkosivuja tehtäessä, mutta kaikkiin niihin löytyi lopulta vastaukset, onhan React kuitenkin suosittu ja johon löytyy todella paljon tutoriaaleja sekä valmiita komponentteja, joka helpotti paljon tekemistä. Lisäosia oli jotenkin hankala löytää, taikka enemmänkin ajatella, että mitä kaikkea on jo tehty valmiiksi, joita voisi hyödyntää.

Teoriaosuudessa puhuttiin käyttöliittymäsuunnittelun ja käytettävyyden tärkeydestä. Käyttöliittymäsuunnittelu (UI-suunnittelu) keskittyy siihen, että käyttöliittymät ovat käyttäjille helppokäyttöisiä ja miellyttäviä. Verkkosivujen käyttöliittymässä on usein samat peruselementit, jotka auttavat käyttäjiä navigoimaan sivustolla helposti. Käyttökokemussuunnittelu (UX-suunnittelu) taas keskittyy siihen, että ymmärretään käyttäjien tarpeet ja käyttäytyminen, ja luodaan ratkaisuja, jotka ovat sekä toimivia että miellyttäviä käyttää.

Näitä periaatteita sovellettiin projektin aikana siten, että verkkosivujen käyttöliittymä suunniteltiin loogisesti ja käyttäjäystävällisesti. Esimerkiksi sivujen ulkoasua ja navigointia suunniteltiin siten, että käyttäjät voivat helposti löytää tarvittavat tiedot. Lisäksi sivustolle tehtiin hampurilaisvalikko, joka parantaa navigointia erityisesti mobiililaitteilla. Mobiiliystävällinen suunnittelu varmistaa, että sivusto toimii hyvin eri laitteilla ja näytön ko'oilla, mikä parantaa käyttäjäkokemusta merkittävästi.

Jatkokehitystä on aina hyvä miettiä ja tähän olen jo miettinyt muutaman asian frontend puolelle. Sivut, jossa on tekstiä pelkästään, ovat aika tylsää luettavaa,

kun tausta on musta ja teksti on valkoinen, eli sivujen ulkoasua voisi muokkailla sekä tehdä hieman interaktiivisemman käyttäjille. Esimerkiksi etusivulle voisi lisätä aikaisemmista tapahtumista pieniä kuvakkeita, joista pääsisi katsomaan aikaisempia live-lähetyksiä. Kuvagalleriaan voisi myös lisätä zoom efektin, jotta kuvia voisi hieman lähempää katsoa.

Opinnäytetyön aikana opin paljon verkkosivujen kehityksestä sekä ongelmanratkaisutaitoni kehittyivät paljon loppua kohti mentäessä. Lopputulos oli kuitenkin mielestäni hyvä, kun vertaa siihen, että kyseessä oli ensimmäinen React-projektini.

LÄHTEET

1. Passeli 2024. Kirjanpidon-sanakirja. Hakupäivä 5.9.2024. https://www.passeli.fi/kirjanpidon-sanakirja/a/api/

2. Bautomo 2024. Sanastoa. Hakupäivä 5.9.2024. https://bautomo.com/sanastoa/html/

3. Interaction-design 2024. User Interface (UI) Design. Hakupäivä 11.10.2024. https://www.interaction-design.org/literature/topics/ui-design

4. Developer mozilla 2024. What do common web layouts contain? Hakupäivä 22.10.2024. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Design_and_accessibility/Common_web_layouts

5. Haltu 2023. Ux-suunnittelu – mitä se on ja mistä asioista se koostuu? Hakupäivä 22.10.2024. https://www.haltu.fi/blogi/ux-suunnittelu

6. React 2013. Introduction to React.js. Hakupäivä 15.9.2024. https://youtu.be/XxVg_s8xAms?si=zobOaoWuilQ0qCHW

7. React 2024. Team. Hakupäivä 15.9.2024. https://legacy.reactjs.org/community/team.html

8. KOMODO 2024. Why react is still popular. Hakupäivä 16.9.2024. https://www.komododigital.co.uk/insights/9-reasons-why-react-is-still-popular-in-2021/

9. geeksforgeeks 2024. React Components. Hakupäivä 10.11.2024. https://www.geeksforgeeks.org/reactjs-components/

10. Microsoft 2024. ES7+ React/Redux/React-Native snippets. Hakupäivä 20.9.2024. https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets&ssr=false#overview

11. Vite 2024. guide. Hakupäivä 20.9.2024. https://vitejs.dev/guide/

12. npmjs 2024. npm – a JavaScript package manager. Hakupäivä 21.9.2024. https://www.npmjs.com/package/npm

13. React Router 2024. Tutorial. Hakupäivä 13.10.2024. https://reactrouter.com/en/main/start/tutorial

14. Smashingmagazine 2012. Hyrum Denney. A Sticky Menu Is Quicker To navigate. Hakupäivä 10.11.2024. https://www.smashingmagazine.com/2012/09/sticky-menus-are-quicker-to-navigate/

15. React-icons 2024. Hakupäivä 24.10.2024. https://react-icons.github.io/react-icons/

16. Imgur 2024. Imgur API. Hakupäivä 1.11.2024. https://apidocs.imgur.com/

17. Flickr 2024. Developer guide: API. Hakupäivä 2.11.2024. https://www.flickr.com/services/developer/api/

18. Yet Another React Lightbox 2022. Hakupäivä 2.11.2024. https://yet-another-react-lightbox.com/

19. npmjs 2024. React Loading Skeleton. Hakupäivä 2.11.2024. https://www.npmjs.com/package/react-loading-skeleton

20. Docker 2024. What is Docker? Hakupäivä 10.11.2024. https://docs.docker.com/get-started/docker-overview/

21. w3schools 2024. HTML <input> required Attribute. Hakupäivä 11.11.2024. https://www.w3schools.com/tags/att_input_required.asp