Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

ERIK NURKKALA

SPA Web-sovelluksen konvertointi SSR-sovellukseksi

SÄHKÖ- JA AUTOMAATIOTEKNIIKKA
2022

Tekijä(t)
Nurkkala, Erik

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
lokakuu 2022

Sivumäärä
19

Julkaisun kieli
Suomi

Julkaisun nimi
SPA Web-sovelluksen konvertointi SSR-sovellukseksi

Tutkinto-ohjelma
Sähkö- ja automaatiotekniikka

Tiivistelmä

Nykypäivän verkkosivuista suuri osa on toteutettu Single Page Application tekniikkaa käyttäen ja se aiheuttaa suuria rajoitteita esimerkiksi hakukoneiden hakutuloksissa.

Työ suoritettiin Dyme Solutions Oy:lle ja ratkaisua luotiin yhteen heidän tuotteistaan, josta kyseinen rajoite löytyy.

Työn tuloksena esitellään useampi ratkaisuvaihtoehto kyseiseen ongelmaan ja dokumentoidaan toteutukseen valikoitunut ratkaisu.

Avainsanat
verkko-ohjelmointi, konvertointi, ohjelmointi, ohjelmointikirjastot, ohjelmointiympäristö

Author(s) Nurkkala, Erik	Type of Publication Bachelor’s thesis	Date October2022
	Number of pages 19	Language of publication: Finnish

Title of publication
SPA Web Application conversion to SSR-application

Degree program
Electrical and Automation Engineering

Abstract

Nowadays a large part of the websites use Single Page Application technology, which creates some restrictions for example search engines.

The work was done for Dyme Solutions Oy, and solution was created for one of their products.

The work result introduces multiple solutions to the problem and documents selected implementation.

Keywords
web programming, converting, programming, programming environment

SISÄLLYS

1 JOHDANTO 6
1.1 Työn toimeksiantaja 6
1.2 Työn määrittely 6
2 WEB TEKNOLOGIAT LYHYESTI 7
2.1 Single Page Application 7
2.2 Static Site Generation 7
2.3 Server Side Rendering 7
3 SINGLE PAGE APP ONGELMAT 7
3.1 Hakukoneoptimointi 7
3.2 Sosiaalisen median jako 8
4 SPA SOVELLUKSEN KONVERTOINTI SSR SOVELLUKSEKSI 9
4.1 Mahdolliset toteutustavat 9
4.2 Valittu toteutustapa 10
4.3 Työn aloitus 10
4.4 Työn toteutus 12
4.4.1 Käyttöliittymäkomponenttien uudelleenkirjoitus 12
4.4.2 Sivujen staattinen generointi 13
4.4.3 Sivujen datan hallinta staattisten sivujen generoinnissa 14
4.4.4 Tapahtumasivujen regenerointi 15
4.4.5 Staattisten sivujen käännöstekstien käsittely 16
4.4.6 Sosiaalisen median jakamisen parantaminen 18
5 YHTEENVETO 18
LÄHTEET
LIITTEET

SYMBOLI- JA LYHENNELUETTELO

SPA	Single Page Application
SSR	Server Side Rendering
SSG	Static Site Generation
JavaScript	Pääasiassa verkkoympäristössä käytettävä ohjelmointikieli
SEO	Search Engine Optimization
HTML	Hyper Text Markup Language
API	Application Programming Interface
DOM	Document Object Model
Fallback	Varasuunnitelma
Revalidate	Uudelleenvalidoida
Regenerate	Uudelleenrakentaa

1 JOHDANTO

Työn toimeksiantaja on viime vuosina käyttänyt verkkosivujen toteutukseen single page application toteutustapaa. Niiden suosio saattaa selittyä sillä, että verkkopalveluilta kaivataan applikaatiomaista sulavuutta, ikään kuin sivu olisi laitteelle asennettu applikaatio.

SPA (Single Page Application):n vahvuuksien lisäksi sen puutteet ovat myös alkaneet tulla selkeämmin esille. Heikkouksiin sen sijaan kuuluu esimerkiksi hakukonenäkyvyys ja sivun esikatselunäkymä, kun sivu jaetaan esimerkiksi sosiaaliseen mediaan.

1.1 Työn toimeksiantaja

Työn toimeksiantaja on porilainen Dyme Solutions Oy. Yritys on perustettu 2014 ja työllistää noin 15 henkilöä, pääosin ohjelmistokehittäjiä. ” Dyme Solutions on ammattitaitoinen ohjelmistoalan moniosaaja Porista, joka rakentaa ohjelmistoja erilaisille yrityksille ja organisaatiolle toimialariippumattomasti.”(Dyme Solutions, n.d)

1.2 Työn määrittely

Työn tarkoituksena on parantaa toimeksiantajan olemassa olevaa verkkoapplikaatiota. Kehityskohteita ovat verkkoapplikaation näkyvyys hakukoneissa sekä sosiaalisen median esikatselun parantaminen sivuston linkkiä jaettaessa.

2 WEB TEKNOLOGIAT LYHYESTI

2.1 Single Page Application

SPA eli Single Page Application on webimplementaatio, jossa selain lataa vain yhden verkkosivun ja sitä päivitetään JavaScriptiä käyttäen. Tämä antaa käyttäjälle mahdollisuuden käyttää verkkosivua ilman, että aina ladataan uusi sivu. Tämä voi parantaa sivun suorituskykyä ja tarjota käyttäjälle dynaamisemman kokemuksen. (MDN, 2021)

2.2 Static Site Generation

Static Site Generationissa verkkosivu rakennetaan tietyn datan pohjalta. Näiden sivujen jakaminen käyttäjälle on todella nopeaa, koska sivu on koko ajan valmiina eikä niitä valmisteta tarpeen vaatiessa. Suosittuja Static Site Generaattoreita ovat esimerkiksi Gatsby ja Next.js. (Cloudflare, n.d.)

2.3 Server Side Rendering

Server Side Renderingissä sivun renderointi käyttövalmiiksi hoidetaan palvelimella. Palvelin hakee tarvittavat tiedot esimerkiksi tietokannasta ja renderöi valmiin HTML (Hyper Text Markup Language)-sivun. Palvelimella renderöidyt sivut tyypillisesti aukeavat nopeasti ja ovat heti käyttövalmiita, koska käyttäjän selaimelle ei lähetetä suurta määrää JavaScriptiä. (Google Developers, 2019)

3 SINGLE PAGE APP ONGELMAT

3.1 Hakukoneoptimointi

SPA sovelluksien näkyvyys hakukoneessa on usein huono johtuen tekniikasta, miten, hakukoneet indeksoivat verkkosivuja. Ne eivät suorita sivulla olevaa JavaScript koodia vaan arvioivat sivun HTML rakennetta (Stack Overflow Blog, 2021). Javascript

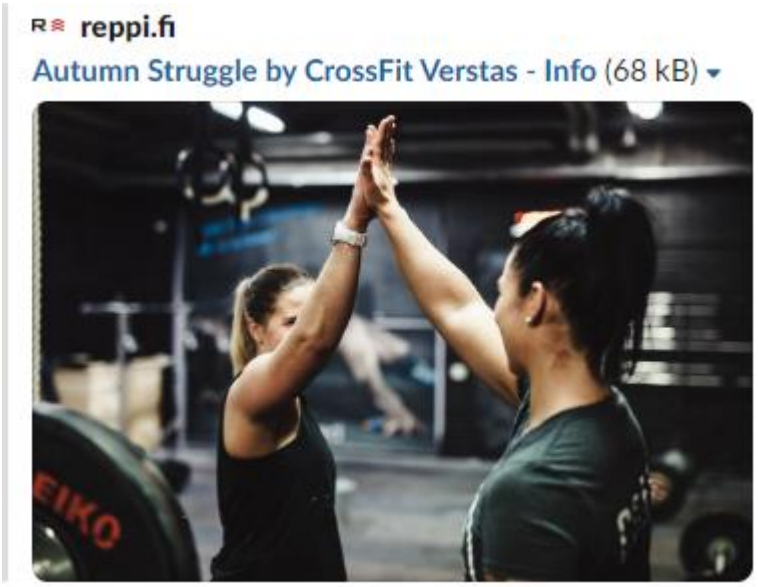
koodin suorittamatta jättäminen aiheuttaa sen, että suuri osa verkkosivun sisällöstä ei ole saatavilla tässä vaiheessa.

3.2 Sosiaalisen median jako

Sosiaalisen median esikatselut toimivat kuten hakukoneiden indeksoijat. Ne eivät usein suorita verkkosivun JavaScript-koodia. Esikatseluikkunat lukevat sivustojen metadataa ja sitä kautta näyttävät esikatselun.

Tässä käytän esimerkkinä sattumanvaraisesti valittua verkkosivua.
Linkki https://reppi.fi/competitions/info/65045fb1-0c82-11ec-8b35-c3eb783cdff9 osoittaa yksittäiseen kilpailuun reppi.fi sivustolla. Esikatselu on suoritettu Slack pikaviestintäsovelluksessa.

Kuvassa 1 nähdään miltä sosiaalisen median esikatselun pitäisi näyttää. Esikatselussa näkyy kilpailu nimi ”Autumn Struggle by CrossFit Verstas” sekä kilpailun valokuva. Kuvassa 2 on esimerkki miltä SPA sovelluksen esikatselu usein näyttää. Esikatselusta puuttuu aikaisemmin mainitut alisivukohtaiset tiedot ja niiden tilalla on sivuston yleiset tiedot.



Kuva 1. Sosiaalisen median esikatselu sivukohtaisella metadatalla.

Kuva 2. Sosiaalisen median esikatselu ilman sivukohtaista metadataa.

4 SPA SOVELLUKSEN KONVERTOINTI SSR SOVELLUKSEKSI

4.1 Mahdolliset toteutustavat

Yleisimmät verkkopalveluiden toteutukseen käytetyt kirjastot ovat Angular, React ja Vue.js (Hackr.io, 2021). Jokaiselle näistä on saatavilla ohjelmointikirjasto, joka lisää niihin ominaisuudet renderöidä sivu palvelimella. Esimerkiksi Reactille suosituin kirjasto on Next.js (Next.js, 2021).

Kuitenkin usein koko verkkoapplikaation migraatio toiselle alustalle on todella suuritöinen ja kallis operaatio. Lisäksi ohjelmointivirheiden mahdollisuus on suuri, kun applikaatio kirjoitetaan uudestaan. Tästä syystä järkevämpi vaihtoehto voisi olla muuttaa vain osa applikaatiosta palvelimella renderöidyksi. Esimerkiksi aikaisemmin esimerkinä mainitun Reppi.fi palvelun tapauksessa vain kilpailusivut voitaisiin renderöidä palvelimella ja muu applikaatio jatkaisi toimintaansa normaalisti. Käytännössä tämä toteutettaisiin siten, että luotaisiin uusi projekti jollakin palvelinrenderöintiä tukevalla

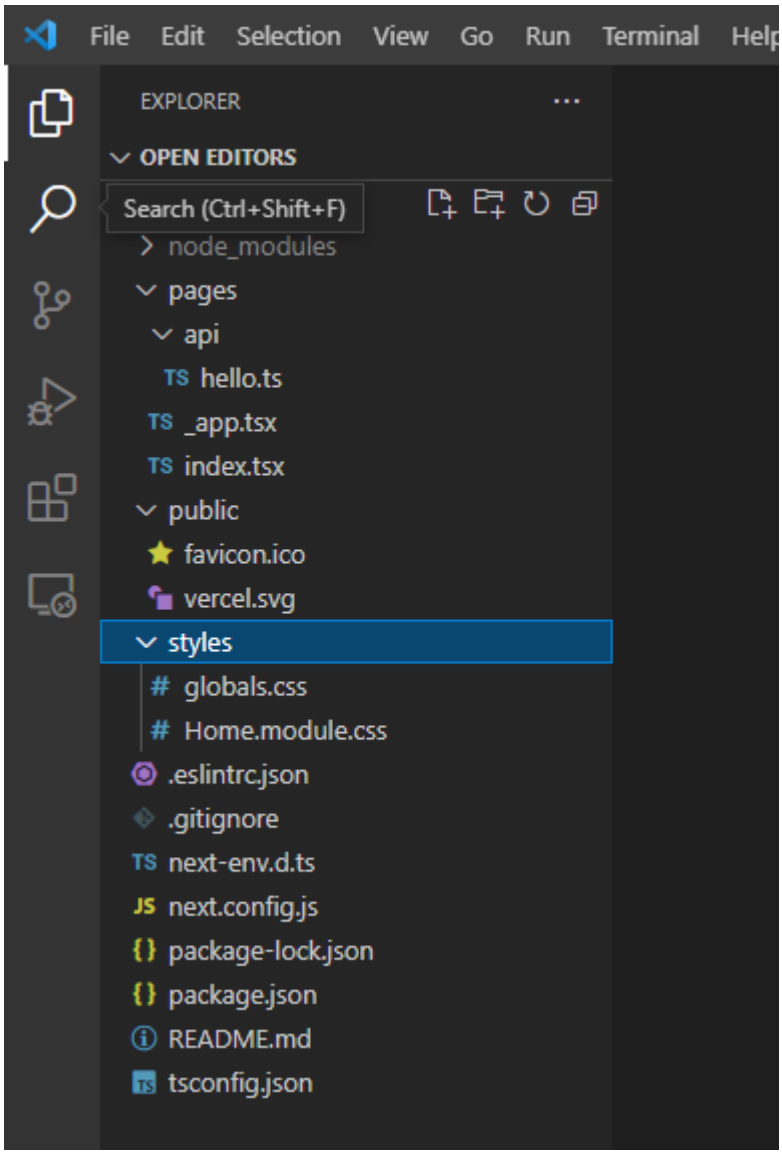
kirjastolla ja applikaation sisällä liikenne ohjattaisiin sinne, kun käyttäjän selain pyytää kilpailun sivua.

4.2 Valittu toteutustapa

Työn toteutukseen valittiin Next.js ohjelmointikirjasto. Toteutusteknologian valinta tässä tapauksessa oli melko vapaata, koska koodia joudutaan kirjoittamaan uudelleen. Työ tullaan toteuttamaan siten, että nykyisestä verkkopalvelusta ohjataan tietyn polun liikenne Next.js kirjastolla toteutettuun verkkosivustoon. Next.js applikaatio tarjoillaan Vercel palvelun kautta. Vercel on sivuston tarjoiluun melko ilmeinen valinta, sillä Vercel on Next.js:än takana. Myös hinnoittelu on todella kilpailukykyinen, 20 dollaria per käyttäjä (Vercel, n.d). Yhdellä käyttäjällä voi tarjoilla useampaa verkkosivua ja Vercel tarjoaa paljon hyödyllisiä ominaisuuksia, joita myöhemmin varmasti tarvitaan.

4.3 Työn aloitus

Next.js projekti perustetaan käyttämällä tarjottua komentorivityökalua. Ajamalla komento: **npx create-next-app@latest --typescript** työkalu luo kuvassa 3 näkyvän kansiorakenteen (Next.js, 2022).



Kuva 3. Työkalun luoma projektipohja.

Luodusta projektista löytyy tarvittavat konfiguraatiotiedostot, sekä hakemistot muun muassa tyylitiedostoille (styles) sekä verkkosivuille (pages). Api (Application Programming Interface) hakemistoa ei tämän projektin puitteissa tulla tarvitsemaan, sillä projekti tulee käyttämään julkista apia. Kuvassa 3 näkyy työkalun luoma projektipohja.

4.4 Työn toteutus

4.4.1 Käyttöliittymäkomponenttien uudelleenkirjoitus

Tarvittavat käyttöliittymäkomponentit oli aikaisemmin toteutettu käyttäen Svelte-ohjelmointikirjastoa (Svelte, n.d). Sveltellä luodut komponentin ovat onneksi todella lähellä, sitä miten käyttöliittymäkomponentit luotaisiin ilman mitään ohjelmointikirjastoa. Tästä syystä komponenttien uudelleenkirjoitus Next.js yheensopivaksi komponentiksi käy melko vaivattomasti.

Kuvista 4 ja 5 voi vertailla miltä osa footer-komponenttia näyttää ensin Sveltellä toteutettuna ja sitten Next.js:llä toteutettuna. Sisältö on täysin sama mutta eri ohjelmointikirjastoilla on erilainen tyyli kirjoittaa asioita.

```
<footer class="footer" lang="{$locale}">
  <div class="footer__logo">
    <img src="{$darkModeEnabled ? logo.dark.url : logo.light.url}" alt="">
  </div>
  <div class="footer__links">
    <span class="footer__links__copyright">
      Copyright &copy; <span class="footer__links__copyright-emphasized">{copyrightText}</span>
    </span>
```

Kuva 4. Komponentin toteutus Svelteä käyttäen.

```
const Footer: React.FC = () => {      You, 3 weeks ago • Create footer
  const siteConf = config.site;
  const [darkModeEnabled, setDarkModeEnabled] = useState<boolean>(false);
  const { t, i18n } = useTranslation();
  return (
    <footer className={styles['footer']} lang={i18n.language}>
      <div className={styles['footer__logo']}>
        <Image width={260} height={65} src={`${config.api.tempAssetUrl}${darkModeEnabled ? siteConf.footer.logo.dark.url : siteConf
      </div>
      <div className={styles['footer__links']}>
        <span className={styles['footer__links__copyright']}>
          Copyright &copy; <span className={styles['footer__links__copyright-emphasized']}>{siteConf.copyrightText}</span>
        </span>
```

Kuva 5. Komponentin toteutus Next.js:ää käyttäen.

Komponenttien uudelleenkirjoituksessa ei siis lähdetty täysin alusta, vaan käytettiin olemassa olevia komponentteja uuden toteutuksen pohjana. Myöskään komponenttien tyylitiedostoihin ei tehty muutoksia vaan ne toimivat suoraan, kunhan DOM (Document Object Model) on rakennettu kuten vanhassa toteutuksessa.

Komponenttien uudelleenkirjoitus kuitenkin vaatii osaavan Next.js ohjelmoijan, jotta komponentti saadaan toimimaan uudessa ympäristössä. Esimerkiksi työkalun rakentaminen, joka muuntaisi komponentit, olisi hankalaa rakentaa applikaation kokonaistoiminnan kannalta. Verkosta löytyy myös muutamia adaptereita, jotka lupaavat Svelte-komponenttien toimivan React ympäristössä, mutta nämä vaikuttivat yksittäisten henkilöiden harrasteprojekteilta, eikä niinkään varteenotettavilta vaihtoehdoilta. Lisäksi tämänkaltaisissa toteutuksissa usein törmätään siihen, että lopulta jotkin ominaisuudet eivät toimi ja uudelleenkirjoitus on väistämätön. Myös projektin ylläpito tulevaisuudessa voisi olla haastavampaa, jos toteutuksesta löytyy yhteensopivuus kerros.

4.4.2 Sivujen staattinen generointi

Työssä sivut generoidaan staattisesti julkisesta LinkedEvents tapahtuma api:sta saadun datan perusteella. Eli jokaista tapahtumaa kohden generoidaan staattinen web sivu. Apin palauttamien tapahtumien suuri määrä aiheutti päänvaivaa staattisten sivujen generoinnin kannalta. Kaikki api:n palauttamat tapahtumat voidaan toki käytännössä generoida, mutta tämän tekemiseen menisi suuri aika. Lisäksi voidaan miettiä, miten suurta liikenne vanhojen jo menneiden tapahtumien sivuilla on. Tuskin kovinkaan merkittävää.

Onneksi Vercelin tiimi on kehittänyt ratkaisun tähän ja sivulle voidaan antaa fallback niminen parametri. Tämä parametri kertoo Next.js kirjastolle, että jos pyydettyä sivua ei löydy valmiina staattisesti generoituna sivuna, generoidaan se ja tallennetaan mahdollisia tulevia sivun pyyntöjä varten palvelimelle. Generointi on nopea, jonka aikana käyttäjälle voidaan näyttää jokin verkkosivuilta tuttu sivun latausta indikoiva animaatio. Seuraava kyseistä sivua pyytävä käyttäjä saa sivun salamannopeasti selaimeensa, koska sivu on generoitu valmiiksi. Koodin ensimmäisessä kääntövaiheessa siis generoidaan vain tietty määrä sivuja valmiiksi ja sitä täydennetään uusien pyyntöjen mukaan.

Kuvassa 6 on koodi, joka hoitaa sivujen polkujen generoinnin. Sivujen polut generoidaan käyttämällä Next.js:n tarjoamaa getStaticPaths-funktiota. Paths-muuttuja sisältää

tapahtumien sivujen polut. GetEventSlug-funktio muokkaa polusta halutun muotoisen.

```
export async function getStaticPaths() {
  const apiUrl = `${config.api.baseUrl}/${config.modules.events.apiPath}`;
  const events = await fetch(`${apiUrl}/event/`).then(res => res.json());

  const paths = events.data.map((event: LinkedEvent) => {
    return {
      params: { id: getEventSlug(event.id, event.name.fi ?? '', event.start_time) }
    };
  });

  // We'll pre-render only these paths at build time.
  return { paths, fallback: true };
}
```

Kuva 6. Työssä implementoitu staattisten polkujen luonti.

Tässä tapauksessa polkuun halutaan tapahtuman id, nimi ja alkamisaika. Näistä muodostetaan merkkijono esimerkiksi, /1232143423_testitapahtuma-01-05-2022. Tämä pääte osoittaa yksittäiseen tapahtumasivuun ja se lisätään verkko-osoitteen perään. Koko verkko-osoite kyseiselle tapahtumalle voisi siis olla https://testi-tapahtuma-sivu.fi/tapahtuma/1232143423_testitapahtuma-01-05-2022. Tämän tyylinen polku luodaan jokaiselle apin palauttamalle tapahtuman id:lle.

4.4.3 Sivujen datan hallinta staattisten sivujen generoinnissa

Kun polut tarvittaville sivuille on luotu, tarvitsee sivujen data toimittaa staattisen sivun generointia varten. Jokaisella tapahtumasivulla on luonnollisesti eri data, mutta sivun ulkoasu on sama, eli data joka voi olla vaikka tapahtuman nimi ja kuvaus, pitää syöttää dynaamisesti. Tähän käytetään Next.js:n tarjoamaa getStaticProps-funktiota. GetStaticProps-funktio hakee koodin kääntövaiheessa apista tarvittavan datan per tapahtuma. Tällöin sivun rakenteen määrittelyssä käytetään muuttujia, joihin data sijoitetaan. Otetaan esimerkiksi tapahtuman nimi. Sivun määrittelyssä tapahtuman nimi korvataan esimerkiksi muuttujalla nimeltä eventName ja sivun luonti vaiheessa siihen dynaamisesti syötetään kunkin tapahtuman nimi.

Kuvassa 7 getStaticProps-funktiolla haetaan api:sta yksittäisen tapahtuman data ja tarjotaan se tapahtuma sivulle. Funktio saa sisäänsä sivun polun, joka luotiin

aikaisemmassa vaiheessa getStaticPaths-funktion avulla. Kuten aikaisemmin mainittiin tämä polku sisältää tapahtuman id:n. Id:tä hyödyntämällä saamme api:sta haettua yksittäisen tapahtuman datan. Tapahtuman data tarjoillaan props-objektiin, joka päätyy myöhemmin tapahtumasivun generointiin käytettäväksi. Tapahtumasivulla datasta voidaan hyödyntää sivun luomiseen esimerkiksi aikaisemmin mainittua tapahtuman nimeä tai muuta tarpeellista dataa.

```
export const getStaticProps: GetStaticProps = async context => {
  const apiUrl = `${config.api.baseUrl}/${config.modules.events.apiPath}`;
  if (context?.params?.id) {
    const eventId = (context.params.id as string).split('_')[0].replace('$', ':');
    const event = await fetch(`${apiUrl}/event/${eventId}`).then(res => res.json());
    return {
      props: {
        event,
        ...(await serverSideTranslations('fi'))
      },
      revalidate: 60
    };
  }
  return {
    props: {}
  };
};
```

Kuva 7. Työssä implementoitu staattisen datan generointi.

4.4.4 Tapahtumasivujen regenerointi

Tapahtumasivut generoidaan koodin kääntö vaiheessa valmiiksi, sillä datalla, jota api sillä hetkellä palauttaa. Pitää ottaa huomioon, että api:n palauttama data voi muuttua. Joku saattaa käydä muokkaamassa tapahtuman nimeä tai sen ajankohtaa. Generoidut sivut ovat etukäteen staattisesti generoituja ja eivät muuttuneeseen dataan reagoi, ellei sille lisätä erikseen käsittelyä.

Tähän Next.js tarjoaa aikaisemmin esiintyneen revalidate (Uudelleenvalidointi) parametrin. Parametriin asetetaan sekuntimäärä, kuinka usein Vercel generoi sivun uudestaan. Tässä projektissa arvoksi valittiin 60 sekuntia. Eli kun joku editoi tapahtuman tietoja, kestää niiden näkyminen sivulla maksimissaan 60 sekuntia. Tämä projekti pyörii Vercel palvelussa, jossa hinnoittelu ei ota kantaa siihen, kuinka usein sivuja generoidaan uudestaan, jolloin kyseinen generointisykli on järkevä.

Kuitenkin jos resurssien käytöstä koituisi kustannuksia generointisyklin tiheyden mukaan, voitaisiin parametriin asettaa suurempikin arvo. Ottamatta kantaa siihen mikä arvo generointisykliin valitaan, on tärkeää kommunikoida tämä tapahtumia editoivalle taholle, jotta heille ei tule tunnetta, että järjestelmä ei toimi.

4.4.5 Staattisten sivujen käännöstekstien käsittely

Verkkosivut, jotka tukevat useampaa eri kieltä vaativat listauksen käännösteksteistä. Ratkaisu on käyttää sivun lähdekoodissa muuttujia. Esimerkiksi teksti ”tervetuloa”, voisi olla muuttuja nimeltä ”welcome”. Sitten vain tehdään jokaiselle tuetulle kielelle kielitiedosto, jossa kerrotaan ”welcome”-muuttujan arvo. Kun käyttäjällä on valittuna suomen kieli, haetaan suomen kielen käännöstiedostosta welcome muuttujan arvo, joka on tervetuloa ja sama prosessi toteutetaan muille tuetuille kielille.

Staattisesti generoitujen sivujen kohdalla tilanne on hieman haastavampi. Sivut on rakennettu etukäteen eli kieltä ei siis voi kesken kaiken muuttaa. Käytännössä siis samasta sivusta täytyy generoida staattinen sivu kaikilla tuetuilla kielillä.

Tässä projektissa valitsin käännöstekstien hallintaan Next.js:lle suunnatun next-i18next. Tämän kirjaston valintaperusteet tähän projektiin olivat melko kevyet. Valintaperusteena oli oikeastaan korkea viikoittaisten latausten määrä npmjs verkkosivulla sekä se, että kirjasto on räätälöity juuri Next.js:n käyttöön. Tällöin voidaan olettaa, että sen tekijät ovat ottaneet huomioon juuri aikaisemmin mainitsemani ongelmat staattisten verkkosivujen kanssa, jotka ovat Next.js:n ydinasioita. Toki kirjastoa valitessa tein nopean katsauksen, ja varmistin, että näitä asioita on oikeastikin huomioitu.

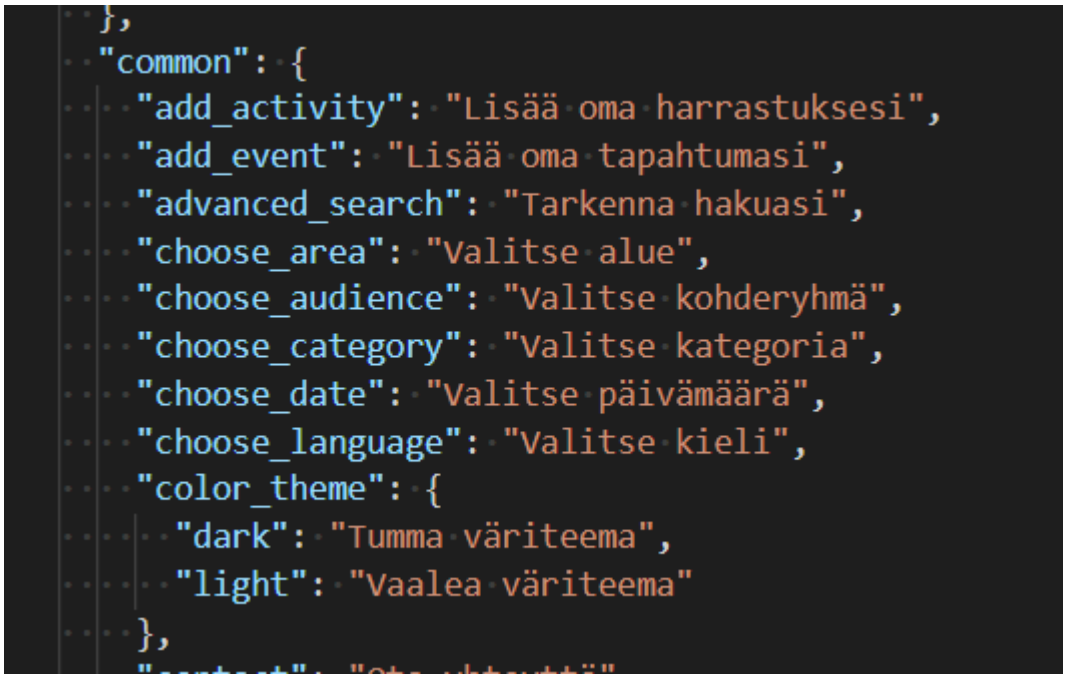
Kuvassa 8 kutsutaan kirjaston tarjoamaa serverSideTranslations-funktiota, joka hakee koodin kääntövaiheessa käännöstiedostosta käännöstekstit. Funktio ottaa parametrina initialLocale:n joka tässä tapauksessa on Suomi eli fi. Kirjaston käyttäminen myös lisää verkkosivun URL kenttään käytössä olevan kielivalinnan. Eli jos sivu on normaalisti https://testi.fi/tapahtuma on se käännöskirjaston käyttämisen jälkeen

https://testi.fi/fi/tapahtuma tai esimerkiksi https://testi.fi/en/tapahtuma. Osoite riippuu siitä millä kielellä sivu on generoitu.

```
export const getStaticProps: GetStaticProps = async context => {
  const apiUrl = `${config.api.baseUrl}/${config.modules.events.apiPath}`;
  if (context?.params?.id) {
    const eventId = (context.params.id as string).split('_')[0].replace('$', ':');
    const event = await fetch(`${apiUrl}/event/${eventId}`).then(res => res.json());
    return {
      props: {
        event,
        ...(await serverSideTranslations('fi'))      You, 4 weeks ago • Create fallback fo
      },
      revalidate: 60
    };
  }
  return {
    props: {}
  };
};
```

Kuva 8. Käännöstekstien hallinta.

Kuvassa 9 on esimerkki käännöstiedostosta. Käännöskirjasto lataa tämänkaltaisen JSON tiedoston ja tarjoaa sen komponentille käytettäväksi.



```
  },
  "common": {
    "add_activity": "Lisää oma harrastuksesi",
    "add_event": "Lisää oma tapahtumasi",
    "advanced_search": "Tarkenna hakuasi",
    "choose_area": "Valitse alue",
    "choose_audience": "Valitse kohderyhmä",
    "choose_category": "Valitse kategoria",
    "choose_date": "Valitse päivämäärä",
    "choose_language": "Valitse kieli",
    "color_theme": {
      "dark": "Tumma väriteema",
      "light": "Vaalea väriteema"
    },
```

Kuva 9. Käännöstiedosto.

4.4.6 Sosiaalisen median jakamisen parantaminen

Jotta sosiaalisessa mediassa jaetun linkin esikatselulaatikkoon saadaan oikeat tiedot pitää ne applikaation koodissa eksplisiittisesti asettaa. Sosiaalisen median palvelut eivät siis osaa katsella verkkosivua ja ottaa sieltä automaattisesti vaikkapa sivun otsikkoa ja valokuvaa esikatselulaatikkoon näkyville.

Kuvassa 10 käytetään Next.js:n tarjoamaa Head komponenttia tagien asettamiseen. Title tagi on normaali HTML tagi ja se asettaa tapahtuman nimen näkymään esimerkiksi selaimen tabissa. Meta tageihin on asetettu OpenGraph protokollan mukaista dataa (OPG, n.d). Ensimmäinen meta tagi og:title asettaa tapahtuman nimen sosiaalisen median esikatselun otsikoksi. Toinen meta tagi og:image asettaa esikatselun valokuvaksi tapahtuman valokuvan.

```
<Head>
  <title>{event.name.fi}</title>      You, 1 second ago • Uncommitted char
  <meta property='og:title' content={event.name.fi ?? ''} key='title'/>
  <meta property='og:image' content={event?.images[0]?.url ?? ''}/>
</Head>
```

Kuva 10. Open Graph tagien asettaminen.

Meta tageihin voi asettaa paljon muutakin. Tässä tapauksessa tapahtuman nimen ja valokuvan lisäksi tullaan todennäköisesti lisäämän myös og:description tagi johon otetaan pieni osa tapahtuman kuvauksesta tai se kokonaisuudessaan jos se on riittävän lyhyt. Eri alustat käsittelevät tageja hieman eri tavalla, mutta turvallinen descriptionin pituus on noin 200 merkkiä. Myös og:titlessä pitää huomioida sen pituus.

5 YHTEENVETO

Tämän opinnäytetyön aikana tutustuttiin verkkoapplikaatioiden erilaisiin teknisiin toteutuksiin. Eri teknisten totutusten vahvuuksiin sekä heikkouksiin. Tämä opinnäytetyö

antaa vastauksia, kun mietitään uuden verkkoapplikaation teknistä toteutusta tai halutaan ratkaista olemassa olevan applikaation ongelmia.

Työssä esiteltiin web-teknologioita yleisesti, jonka jälkeen siirryttiin ratkaisemaan toimeksiantajan verkkoapplikaation ongelmia. Toimeksiantajan ongelma ratkaistiin käyttämällä Next.js ohjelmointikirjastoa, jolla osa applikaatiota kirjoitettiin uudestaan.

Työ on tarkoitus tulla toimeksiantajalle tuotantokäyttöön, joten vaikka Next.js:n käyttäminen ratkaisuna saattoi olla ratkaisuista työläin, on se paras ratkaisu tulevaisuuden kannalta. Vercelin tiimi kehittää kirjastoa aktiivisesti ja siihen lisätään uusia ominaisuuksia jatkuvasti. On siis epätodennäköistä, että uutta ratkaisua verkkoapplikaation tekniseen toteuttamiseen tarvitsee lähiaikoina etsiä.

Työn tekemisen aikana sain oppia paljon moderneista web teknologioista sekä erityisesti staattisten verkkosivujen generoinnista ja niihin liittyvistä haasteista.

LÄHTEET

Hackr.io. (2021). 10 Best JavaScript Frameworks to Use in 2021. Haettu 31.10.2021 osoitteesta https://hackr.io/blog/best-javascript-frameworks

Next.js. (2022). Getting Started. Haettu 13.2.2022 osoitteesta https://nextjs.org/docs/getting-started

Next.js. (2021). The React Framework. Haettu (31.10.2021) osoitteesta https://nextjs.org/

Vercel (2022). Pricing. Haettu (13.2.2022) osoitteesta https://vercel.com/pricing

Google Developers. (2019). Rendering on the Web. Haettu (31.10.2021) osoitteesta https://developers.google.com/web/updates/2019/02/rendering-on-the-web

MDN. (2021). SPA (Single-page application). Haettu 31.10.2021 osoitteesta https://developer.mozilla.org/en-US/docs/Glossary/SPA

Svelte. (n.d) Cybernetically enhanced web apps. Haettu 20.2.2022 osoitteesta https://svelte.dev/

OPG. (n.d). The Open Graph protocol. Haettu 18.4.2022 osoitteesta https://ogp.me/

Stack Overflow Blog. (2021). What I wish I had known about single page applications. Haettu 31.10.2021 osoitteesta https://stackoverflow.blog/2021/02/24/what-i-wish-i-had-known-about-single-page-applications/

Cloudflare. (n.d). What is a static site generator? Haettu 31.10.2021 osoitteesta https://www.cloudflare.com/learning/performance/static-site-generator/.

Dyme Solutions. (n.d). Yritys. Haettu 16.10.2021 osoitteesta https://dymesolutions.com/yritys/