VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Matti Soikkeli

POLAR FLOW -PALVELUN KÄYTTÖLIITTYMÄN KEHITTÄMINEN

Liiketalous
2022

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Matti Soikkeli
Opinnäytetyön nimi	Polar Flow -palvelun käyttöliittymän kehittäminen
Vuosi	2022
Kieli	suomi
Sivumäärä	29
Ohjaaja	Klaus Salonen

Tämän opinnäytetyön tavoitteena oli luoda uusi aloitussivu Polar Flow -palveluun. Tämän lisäksi tarkoituksena on dokumentoida Polar Flow -verkkopalvelun frontend teknologioita ja arkkitehtuuria sekä käydä läpi, miten käyttöliittymä on yleisellä tasolla rakentunut.

Polar Flow -palvelun web-käyttöliittymä on pidemmällä aikavälillä siirtymässä vaiheittain uuteen teknologiaan. Opinnäytetyössä käydään läpi vanhaa teknologiaa ja uudempaa, johon kyseinen käyttöliittymä on siirtymässä. Dokumentointiosuudessa käydään läpi mitä kaikkia vaiheita migraatio vaatii.

Aluksi työssä tutkitaan Polar Flow -palvelun toiminnallisuutta, tarvittavia työkaluja ja teknologioita aloitussivun toteuttamiseksi. Tämän jälkeen tutustutaan kokonaiskuvaan, miten käyttöliittymä on rakentunut yleisellä tasolla. Työssä tarkastellaan myös vanhaa teknologiaa ja mitä hyötyjä on uuteen teknologiaan siirtymisestä.

Opinnäytetyön lopputuloksena Polar Flow -palvelussa on käytössä uusi aloitussivu uusia asiakkaita varten.

Avainsanat frontend, käyttöliittymä, ohjelmistokehitys

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Matti Soikkeli
Title	Development of the Polar Flow web service UI
Year	2022
Language	Finnish
Pages	29
Name of Supervisor	Klaus Salonen

The goal of this thesis was to create a new start page for the Polar Flow service and document the technologies used. In addition to this, the objective was to document the frontend technologies and architecture of the Polar Flow web service and review how the user interface is structured in general.

In this thesis, the user interface of Polar Flow web service is gradually transitioning to a more modern technology. The thesis examines the old technology and studies where the user interface is moving. The documentation section examines all the steps this migration requires.

Initially, the work will review the functionality of the Polar Flow web service and the necessary tools and technologies to complete the implementation of the landing page. After this the overall picture of how the user interface is structured on a general level is introduced. The work also examines old technology and the benefits of new technology transition.

As the result of the thesis, the Polar Flow web service has a new landing page for new customers.

Keywords frontend, user interface, software development

SISÄLLYS

TIIVISTELMÄ

ABSTRACT

1 JOHDANTO 6

2 POLAR FLOW 7

3 FRONTEND TEKNOLOGIAT JA KÄSITTEET 8

3.1 JavaScript 8

3.2 TypeScript 8

3.3 React 9

3.4 React Testing Library 9

3.5 Monorepo 9

3.6 Less (Leaner Style Sheets) 10

3.7 Create React App 10

3.8 Lerna 11

3.9 Yarn Workspaces 11

3.10 Play Framework 11

3.11 Scala 11

4 TESTAUS 12

4.1 Yksikkötestaus 12

4.2 Integraatiotestaus 12

4.3 Hyväksyntätestaus 13

5 KÄYTTÖLIITTYMÄN KEHITTYMINEN 14

5.1 Vanha teknologia 14

5.2 Välivaihe (Flow UI 2.2) 14

5.3 Nykytilanne (monorepo) 16

6 ALOITUSSIVUN UUDISTAMINEN 18

6.1 Uudistuksen tausta 18

6.2 Projektin tavoite 19

6.3 Suunnittelu 19
6.4 Toteutus 20
6.5 Testaus 21
7 YHTEENVETO 24
7.1 Aloitussivun uudistus 24
7.2 Käytetyt teknologiat 24
7.3 Jatkokehitys 25
LÄHTEET 27

KUVIO- JA TAULUKKOLUETTELO

Kuva 1. Harjoituksen analyysi Polar Flow -palvelussa 7
Kuva 2. Create React App asennus 10
Kuva 3. Multirepon rakenne 15
Kuva 4. Monorepon rakenne 16
Kuva 5. Vanha aloitussivu 18
Kuva 6. Aloitussivun UX suunnitelma 19
Kuva 7. Uudistettu aloitussivu 20
Kuva 8. Otsikon ja informaatiolaatikoiden yksikkötestaus 21
Kuva 9. Informaatiolaatikon sisäisen otsikon testaus 21
Kuva 10. Informaatiolaatikoiden sisällä olevien ikonien testaus 22
Kuva 11. Informaatiolaatikoiden sisältämien painikkeiden testaus 23

1 JOHDANTO

Opinnäytetyön tavoitteena on luoda Polar Flow -palveluun uusi aloitussivu ja dokumentoida käyttöliittymässä käytetyistä teknologioista. Työn toimeksiantajana toimii suomalainen Polar Electro Oy, joka valmistaa langattomia sykkeenmittauslaitteita sekä erilaisia harjoitussuorituksia tukevia ohjelmistoja ja teknologioita.

Polar Flow on ilmainen työkalu urheilusuoritusten, aktiivisuuden ja unen suunnitteluun ja seurantaan.

Aloitussivu on sivu, jolle uusi asiakas ohjataan ensimmäisenä ostaessaan Polar tuotteen. Aloitussivulla ohjeistetaan asiakasta lataamaan Polar Flow -sovellus puhelimelle tai Polar FlowSync tietokoneelle. FlowSyncin avulla käyttäjä voi synkronoida laitteen tietokoneen välityksellä Polar Flow -palveluun. Kyseinen sivu on keskeisessä roolissa asiakkaan tyytyväisyyden takaamiseksi, koska se on paikka, jonne asiakas ensimmäisenä ohjataan.

Polar Flow -käyttöliittymää päivitetään vaiheittain uudempaan teknologiaan ja aloitussivun uudistus on osa tätä työtä. Työssä käydään läpi vanhan ja uuden teknologian eroavaisuudet ja syyt, miksi halutaan päivittää käyttöliittymää uuteen.

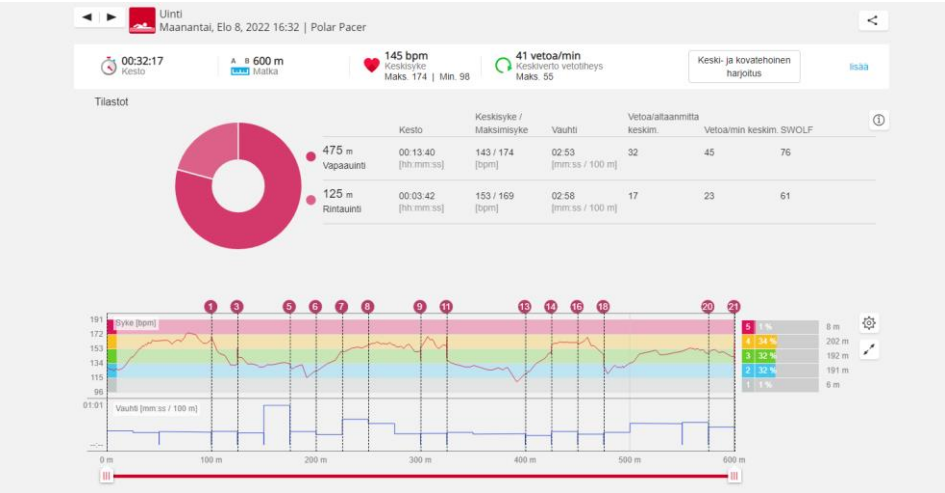
Työ on rajattu käsittelemään aloitussivun uudistamista eikä tässä opinnäytetyössä käydä läpi kaikkien käyttöliittymään kuuluvien moduulien toiminnallisuutta. Työssä ei myöskään käsitellä Polar Flow -puhelinsovellusta. Kyseessä on toiminnallinen opinnäytetyö.

2 POLAR FLOW

Polar Flow -palvelu on käyttäjän tietokoneella toimiva selainpohjainen sovellus. Polar Flow -palvelu mahdollistaa käyttäjälle laajaa seurantaa aktiivisuuden, treenien ja unen ylläpitämiseen. Palvelussa on käytettävissä päiväkirja, johon käyttäjä voi suunnitella tulevia urheilusuorituksia tai tarkastella aiempia harjoitteita, joita rannelaite on tallentanut. Käyttäjä synkronoi laitteen urheilusuorituksen jälkeen tietokoneelle, jonka jälkeen kaikki rannelaitteessa olleet tiedot siirtyvät näkyviin Polar Flow -palveluun.

Palvelussa voi seurata päivittäistä aktiivisuutta. Rannelaite mittaa käyttäjän toimintaa ympäri vuorokauden. Kun laite on synkronoitu Polar Flow -palveluun, voi käyttäjä seurata helposti omaa aktiivisuuttaan ja harjoitteita.

Laitteet ovat yhteensopivia laajasti eri urheilulajien kanssa. Esimerkiksi uintitreenistä on mahdollista analysoida monipuolisesti käyttäjän sykealueet, uitu matka, treenin kesto ja uintilajit, joita on treenin aikana suoritettu. Polar Flow -palvelu piirtää yksinkertaisia graafeja sykkeistä ja vauhdista, joita käyttäjän on helppo tulkita (kuva 1). Polar Flow -palvelu kertoo myös käyttäjälle harjoituksen vaikutuksesta, jonka perusteella käyttäjä pystyy tulkitsemaan omaa suoritusta ja muokkaamaan harjoitteluaan tarpeen mukaan. (Polar 2022).



KUVA 1. Harjoituksen analyysi Polar Flow -palvelussa

3 FRONTEND TEKNOLOGIAT JA KÄSITTEET

Tämän luvun tarkoituksena on esitellä erilaisia teknologioita mitä on hyödynnetty aloitussivun uudistuksessa ja muissa Polar Flow -palvelun ominaisuuksissa. Kyseisten teknologioiden valinta perustuu nopeaan ja ketterään ominaisuuksien kehittämiseen ja ylläpitoon Polar Flow -palvelun sisällä.

3.1 JavaScript

JavaScript on olio-ohjelmointikieli, joka mahdollistaa dynaamisen toiminnallisuuden lisäämisen verkkosivuille. Lähes kaikki verkkosivustot käyttävät sitä erilaisten käyttökokemusten parantamiseksi. JavaScriptiä tukevat suosituimmat käyttöjärjestelmät, kuten Windows ja macOS. Myös kaikki suositut verkkoselaimet tukevat JavaScriptiä. (Javatpoint 2021).

3.2 TypeScript

TypeScriptin on kehittänyt Microsoft 2010-luvun alussa, jonka jälkeen se julkaistiin avoimeen lähdekoodiin vuonna 2012.

TypeScript on JavaScriptin osajoukko ja se lisää JavaScript-kieleen vahvan tyypityksen ja sen avulla käännösaikaisen virheiden tarkastuksen. TypeScript on suunniteltu suurten sovellusten kehittämiseen ja TypeScript koodi kääntyy selainten ymmärtämäksi JavaScript koodiksi. (J. Goldberg 2022).

TypeScript suorittaa ohjelmassa staattisen tyyppitarkistuksen, mikä tarkoittaa, että TypeScript tarkistaa ohjelman läpi virheiden varalta ennen sen suorittamista editorissa. Tämä helpottaa kehittäjän työtä, koska mahdolliset virheet saadaan selville kehitysvaiheessa. (D. Choi 2020).

3.3 React

React on JavaScript-pohjainen kirjasto käyttöliittymien rakentamiseen. Sen on perustanut Meta (ent. Facebook) ja tällä hetkellä React on yksi suosituimmista JavaScript työkaluista.

Reactilla käyttäjä voi rakentaa yksittäisiä käyttöliittymäkomponentteja. Yksittäinen komponentti on JavaScript-luokka tai funktio, joka hyväksyy syötteitä ja tulostaa React elementin, jonka tarkoituksena on kuvata kuinka sen kuuluisi näkyä käyttöliittymässä. (H. Narayn 2022)

3.4 React Testing Library

React Testing Library on kirjasto React-sovellusten testaamiseen. React testing library antaa työkalut yksikkötestaamista varten.

React testing libraryn avulla kehittäjä pystyy kirjoittamaan ylläpidettäviä yksikkötestejä. Jotta testeistä saa ylläpidettäviä pitkässä juoksussa, ne eivät saa sisältää niinkään komponenttien toteutustietoja, vaan tarkoituksena on luoda testin avulla varmuus siitä, mihin kyseistä komponenttia käytetään. Tällä tavoin myös muut kehittäjät pystyvät tekemään komponenttiin muutoksia ilman, että yksikkötestit hajoavat ja työnteko hidastuu. (Testing-library 2022).

3.5 Monorepo

Monorepo on yksi arkisto, jonka sisällä on useita erillisiä projekteja. Tämä mahdollistaa nopean tavan tehdä yksittäisiä koodimuutoksia eri projekteihin.

Monorepossa koodin uudelleenkäytettävyys on vaivatonta, koska projektin koodikanta sijaitsee kokonaisuudessaan samassa arkistossa. Sama arkisto voi pitää sisällään myös useita ohjelmointikieliä.

Monorepo antaa mahdollisuuden työskennellä tehokkaasti. Se lisää johdonmukaisuutta ja vähentää kitkaa uusien projektien luomisessa helpottamalla koodin jakamista sekä tiimien välistä yhteistyötä. (Monorepo.tools 2022)

3.6 Less (Leaner Style Sheets)

Less on CSS-esiprosessori, joka mahdollistaa muokattavan ja uudelleenkäytettävän tyylisivun verkkosivuille. Less on dynaaminen tyylisivukieli, joka kykenee laajentamaan CSS:n ominaisuuksia. Se tarjoaa erilaisia toimintoja, kuten muuttujia, funktioita, mixinejä sekä operaatioita, joiden avulla voidaan rakentaa dynaamista CSS:ää.

Less:in avulla voidaan kirjoittaa puhtaampaa ja luettavampaa koodia hyödyntämällä sisäkkäisyyksiä (nesting). (Lesscss 2022).

3.7 Create React App

Uuden React-projektin aloittaminen on aiemmin ollut monimutkaista, koska siinä vaadittiin useita eri työkaluja ja se piti sisällään eri työvaiheita.

Vuonna 2016 Facebookin julkaisema Create React App komentorivityökalu poisti kaikki monimutkaiset vaiheet ja teki React sovelluksen perustamisesta yksinkertaisen. Asennus tapahtuu npm-ohjelman (Node package manager) avulla, jolla voit asentaa tarvittavan paketin terminaalissa. (Reactjs 2022).

```
npx create-react-app my-app
cd my-app
npm start
```

KUVA 2. Create React App asennus

3.8 Lerna

Lerna on nopea monorepo-työkalu JavaScript ja TypeScript pakettien hallinnoimiseen ja julkaisemiseen samasta arkistosta.

Lerna helpottaa kehittäjien työskentelyä hallitsemalla projektien välisiä riippuvuuksia, versiointia sekä koodin käyttöönottoa. Lernan hyödyt tulevat esiin varsinkin isommissa projekteissa, joissa on haasteena ylläpitää kaikkia prosesseja manuaalisesti. (Lerna.js 2022)

3.9 Yarn Workspaces

Yarn workspaces on pakettienhallinnan työkalu. Sitä käytetään pakettien ja riippuvuuksien hallinnassa ja sen avulla pystyy tallentamaan kaikki tiedostot yhteen monoliittiseen arkistoon. (M. Kocik 2020).

3.10 Play Framework

Play Frameworkin on luonut Guillaume Bort vuonna 2007. Se on avoimeen lähdekoodiin perustuva verkkosovelluskehys, joka on luotu tekemään verkkosovellusten kehittämisestä helpompaa ja tuottavampaa. Play tukee Scala ja Java-ohjelmointikieliä. (Object computing 2014).

Play Framework noudattaa MVC (Model-View-Controller) suunnittelumallia, jonka tehtävä on eriyttää sovellusten esitys- ja logiikkakerros toisistaan ja jakaa vastuita eri osa-alueille. (Hurja 2020).

3.11 Scala

Scala, lyhenne sanoista scalable language on moniparadigmainen ohjelmointikieli, joka on suunniteltu ilmaisemaan yleisimpiä ohjelmointimalleja ytimekkäästi ja tyyppiturvallisesti. Se integroi olio- ja toiminnallisten kielten ominaisuuksia. (Docs Scala-lang 2022).

4 TESTAUS

Testauksessa tarkastellaan ohjelman laatua ja toiminnallisuutta suorittamalla ohjelmaa, tarkoituksena löytää mahdolliset virheet eli bugit ohjelmistosta kehitystyön aikaisessa vaiheessa. Testauksella vähennetään riskejä, sillä kehittämisvaiheessa esiin nousevat virheet aiheuttavat vähemmän kustannuksia kuin tuotantokäytössä. (Cse.hut 2016).

4.1 Yksikkötestaus

Yksikkötestaus on uuden moduulin laadunvarmistamisen menetelmä, jossa käydään läpi kirjotetun koodin toiminnallisuus. Kyseessä on testaamisen ensimmäinen vaihe, jonka tarkoituksena on havaita mahdolliset virheet kirjoitetusta koodista. (Wikipedia 2022).

Yksikkötestauksen tarkoituksena on testata UI moduulin sisäisiä komponentteja. Tällä tasolla on tärkeää suorittaa yksikkötestejä mahdollisimman kattavasti sekä normaalia käyttöä, että erilaisia rajatapauksia kokeillen, jotta mahdolliset virheet huomataan jo kehitysvaiheessa.

Yksikkötestien luomisessa voidaan arvioida kattavuutta eri näkökulmista, kuten funktiokattavuus, jonka tarkoituksena on arvioida, onko testissä kutsuttu kaikkia käytettyjä metodeja. Toinen kattavuutta arvioiva mitta on lausekattavuus. Tarkoituksena huomioida onko testissä suoritettu ohjelman kaikki kirjoitetut rivit ja lauseet. (Cse.hut 2016).

4.2 Integraatiotestaus

Integraatiotestauksessa testataan järjestelmän osien välisiä yhtymäkohtia, eli integraatioita. Tällä tavoin varmistetaan, että data kulkee ohjelmistojen välillä oikeassa ajassa, oikeassa muodossa sekä oikeilla arvoilla.

Integraatiotesti käy läpi sivun toiminnallisuudet. Sillä tarkistetaan, että yksiköt toimivat yhdessä oikein sekä testataan backend rajapinnan kokonaisuuden toimivuus. (ProjectTop 2022).

4.3 Hyväksyntätestaus

Hyväksyntätestaus on testausvaiheen viimeinen prosessi, jossa testataan, että ohjelman eri prosessit toimivat alusta loppuun asti oikein ennen kuin ohjelmisto otetaan käyttöön tuotannossa.

Testaaja käy läpi ohjelman kaikki mahdolliset toiminnallisuudet ja tarkistaa, vastaako työ alkuperäistä suunnitelmaa.

5 KÄYTTÖLIITTYMÄN KEHITTYMINEN

Tässä luvussa käydään läpi Flow-palvelun käyttöliittymän rakentumista vanhasta teknologiasta uuteen ja minkälaisia eri vaiheita siirtymä sisältää.

5.1 Vanha teknologia

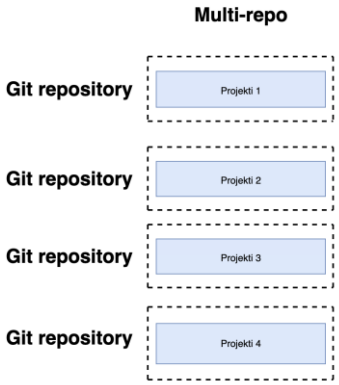
Vanhassa teknologiassa käytettiin Play Frameworkin tukemia malleja (templates) hyödyntäen Scala-ohjelmointikieltä. Malli toimii kuin funktio, joka hyväksyy mallin muuttujat parametreina ja palauttaa HTML-koodin. Play Frameworkin mallijärjestelmän avulla pystyy luomaan dynaamisesti esim. html, xml tai json tiedostoja. (Baeldung 2021).

Vaikka Play Framework on käytössä muutamissa suurissa yrityksissä, niin sen käytön opettelu on työlästä. Scala ei ole kovin tunnettu ohjelmointikieli, joten osaajia on haastava löytää.

Play Framework on toimiva ja tehokas verkkosovelluskehys, mutta jo lähtöpisteessä sitä ei onnistuttu käyttämään oikein, eikä tarvittavia korjausliikkeitä onnistuttu tekemään. Lisäksi käytetyn Play Frameworkin versiopäivitys osoittautui työlääksi, ja siten käytetty versio jäi usein jälkeen kehityksestä. Lopulta kehittäminen oli työlästä sen hitauden vuoksi.

5.2 Välivaihe (Flow UI 2.2)

Ensimmäinen vaihe Polar Flow -käyttöliittymän uudistamiseksi oli siirtää Polar Flow:n UI omaan kerrokseen. Teknologia vaihdettiin samalla nykyaikaisempaan React-pohjaiseen sovellukseen. Samalla versionhallinnan osalta otettiin käyttöön multirepo-malli, jossa jokaisella ominaisuudella oli oma git-arkisto.



Kuva 3. Multi-repon rakenne

Alkuperäinen idea oli jakaa arkistoja (Repository) tiimien vastuualueelle, mutta aina kun aloitettiin kehittämään uutta ominaisuutta, sille piti luoda uusi arkisto.

Välivaiheen aikana Polarilla oli enemmän tiimejä ja alkuperäinen idea oli jakaa ominaisuuksia tiimeille siten, että tiimit vastaavat oman vastuualueensa ominaisuuksista ja julkaisisivat ne tuotantoon itsenäisesti. Arkkitehtuuri ei kuitenkaan mahdollistanut tällaista julkaisutapaa vaan sitoi ominaisuudet riippuvuudeksi yhdelle käyttöliittymäsovellukselle.

Tämän myötä jokaiselle uudelle ominaisuudelle piti luoda uusi arkisto ja arkistojen määrä oli lopulta yli 15. Tämä hidasti ylläpitoa, koska jokaisen arkiston kirjastojen ja työkalujen ylläpidosta piti huolehtia ja katsoa, että versiot ovat keskenään yhteensopivia.

Tavoitteena oli, että kehittäjä pystyisi kehittämään käyttöliittymän ominaisuuksia nopeasti ja vaivattomasti. Usean arkiston kloonaaminen ja kehitysympäristön pystyttäminen sekä arkistojen vaihtaminen keskenään oli kuitenkin työlästä ja hankaloitti yhteisten komponenttien kehittämistä. Silloin haluttiin myös mahdollistaa koko Polar Flow-verkkopalvelun käyttöliittymän kehittäminen paikallisessa ympäristössä.

Välivaiheen aikana oli käytössä hot-reload toiminto, mutta se sijaitsi kehitettävän moduulin sisällä. Välivaiheen aikana tehtiin uusia ominaisuuksia käyttäen react ohjelmointikieltä.

Hot-reload toiminnolla kehittäjä näkee koodiin tehdyt muutokset ilman, että koko sovellus käynnistetään uudestaan. Aina kun tehdään muutoksia, kehittäjän tarvitsee vain tallentaa koodi, jonka jälkeen tehdyt muutokset ovat näkyvillä kehitysympäristössä, ja kehittämistyö on ketterämpää ja tehokkaampaa. (Geeks for geeks 2022).

5.3 Nykytilanne (monorepo)

Vanhemmat teknologiat toivat mukanaan liikaa hidastavia tekijöitä, jotka aiheuttivat kehittäjille ylimääräistä työtä. On taloudellisesti järkevää käyttää teknologiaa, joka mahdollistaa kehittäjille ketterän kehittämisen.

Polar Flow -palvelun nykytilanteessa käytetään monorepo-mallia, missä kaikki UI projektit sijaitsevat yhden arkiston alla. Tämä mahdollistaa käyttäjälle sekä ominaisuuden että yhteisten komponenttien nopean ja ketterän kehittämisen. Kehittäjä saa koko Polar Flow -verkkopalvelun auki omassa lokaalissa ympäristössä ja uusien ominaisuuksien lisääminen, päivitysten teko ja bugien korjaus on huomattavasti vaivattomampaa.

Kuva 4. Monorepon rakenne

Monorepo perustuu Create React App (CRA) ympäristöön, jossa yleisesti käytössä olevat työkalut on yhdistetty ja konfiguroitu toimimaan keskenään. Käyttöliittymän ominaisuudet ovat jaoteltu monorepossa omiksi CRA projekteikseen. Monorepo-mallissa on käytössä React ohjelmointikieli ja monorepo työkalut Lerna ja Yarn.

Yarnin avulla kehittäjä pystyy käynnistämään yksittäisen moduulin tai kaikki moduulit kehitystyötä varten vaivattomasti. Kehittäjälle aukeaa sama näkymä, joka näkyy tuotannossa.

Flow sisältää edelleen muutamia vanhan teknologian omaavia moduuleita, joiden päivittäminen nykyiseen on työlästä. Jokaisen vanhan moduulin päivittäminen uuteen vaatii uudelleenohjelmointia Reactilla.

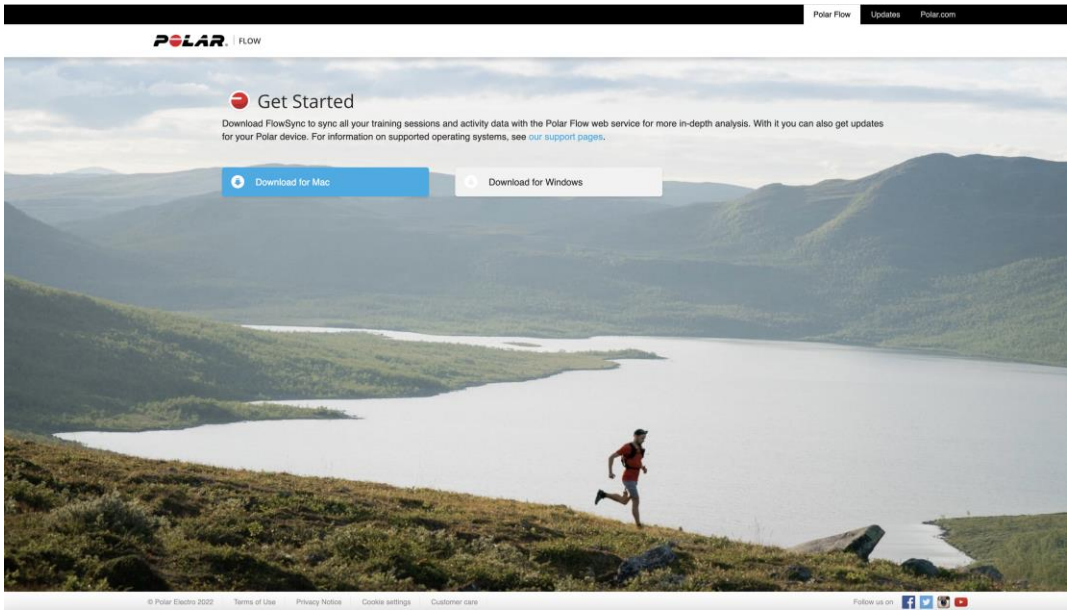
Nykytilanteessa tarkoituksena on kehittää Flow palvelua nykyaikaisilla teknologioilla. Monorepon ja siinä käytettyjen työkalujen Lernan ja Yarnin ohelle on tullut uudempia ja ketterämpiä teknologioita. Tarkoituksena on pysyä teknologian kehityksessä mukana ja päivittää vanhempia teknologioita uusiin.

6 ALOITUSSIVUN UUDISTAMINEN

Käyttäjä ostaa tuotteen ja tuotepaketti sisältää ohjeet, jotka ohjaavat hänet aloitussivulle. Aloitussivulla kerrotaan, miten käyttäjä pääsee käyttämään Polar Flow-palvelua. Sivuston sisältämissä informaatiolaatikoissa kerrotaan mitä käyttäjän kannattaa tehdä ja suositellaan lataamaan omaan käyttöön sopiva sovellus joko puhelimeen tai tietokoneelle. On myös mahdollista käyttää tuotetta ilman Polar Flow -palvelua, mutta silloin käyttäjä ei saa parasta käyttökokemusta ostetusta laitteesta.

6.1 Uudistuksen tausta

Aloitussivu on yksi Polar Flow -palvelun tärkeimmistä näkymistä. Vanha aloitussivu oli liian pelkistetty ja siihen haluttiin luoda uusi ulkoasu, jonka tarkoituksena on auttaa paremmin käyttäjää uuden laitteen käyttöönotossa.



KUVA 5. Vanha aloitussivu

6.2 Projektin tavoite

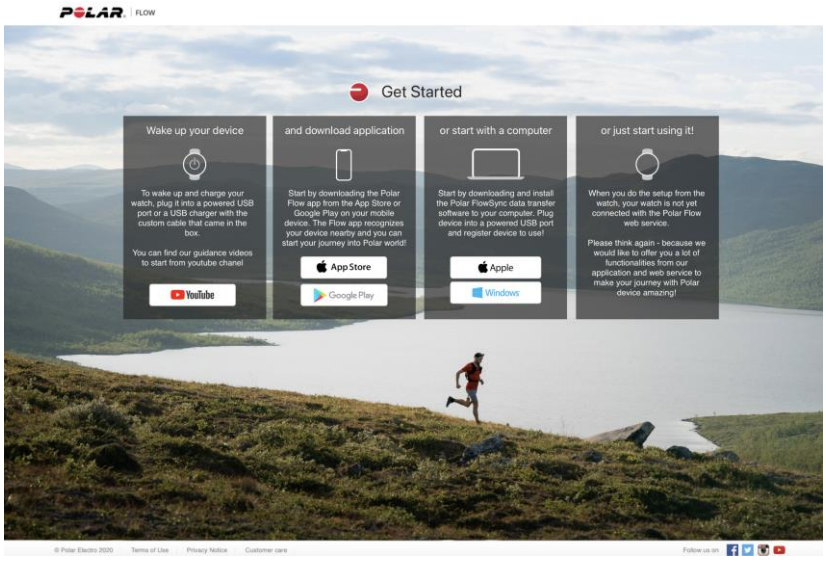
Projektin tavoitteena oli luoda nykyaikainen ja helppolukuinen aloitussivu uusia käyttäjiä varten, jotta uuden laitteen käyttöönotto olisi mahdollisimman vaivatonta. Ikonien, tekstisisällön ja painikkeiden avulla käyttäjä onnistuu navigoimaan helposti haluamaansa vaihtoehtoon.

6.3 Suunnittelu

Sivuun oli valmiina suunnitelmat, joita kävimme UX-suunnittelijoiden kanssa läpi ja teimme tarvittavia päivityksiä suunnitelmaan. Aloitussivulla on vähän toiminnallisuutta, joten suunnittelu painottui sivun ulkomuotoon.

Sivulle haluttiin neljä informaatiolaatikkoa, joissa kaikissa on oma ikoni ja tekstisisältö. Tekstit opastavat käyttäjää Polar Flow -palvelun käyttöönotossa. Kolmeen informaatiolaatikkoon suunniteltiin painikkeet, joista käyttäjä pääsee lataamaan sovelluksen halutulle alustalle. Sivulle määriteltiin myös oma taustakuva.

Sivulle suunniteltiin myös kielikäännökset, jotta informaatiolaatikoiden sisällä olevat tekstit näkyvät käyttäjän halutessa myös muilla kielillä. Sivun yläpalkkiin lisättiin kielivalikko, jossa käyttäjä pystyy valitsemaan haluamansa kielen.



KUVA 6. Aloitussivun UX suunnitelma

6.4 Toteutus

Monorepo pitää sisällään kaikki Polar Flow -käyttöliittymän moduulit ja myös uudelle aloitussivulle piti luoda oma moduuli. Olemassa olevasta template moduulista otettiin kopio pohjarungoksi, jota lähdettiin muokkaamaan.

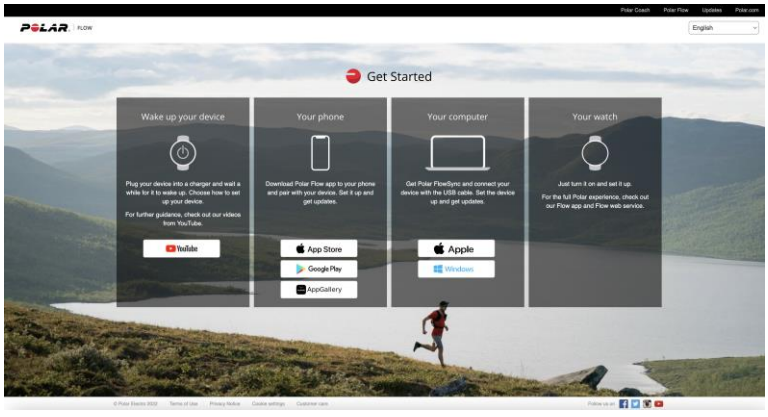
Seuraavaksi luotiin käynnistys- ja testiscriptit projektin juuresta löytyvään package.json tiedostoon, jotta aloitussivun pystyy käynnistämään kehittämistä varten ja sille pystytään suorittamaan yksikkötestauksia. Yksikkötestauksella tarkoitetaan sitä, että kaikki moduulin toiminnallisuudet testataan koodin sisällä buildin yhteydessä ennen kuin siitä suoritetaan oma vetopyyntö (Pull request) versionhallintajärjestelmään.

Moduulin luomisen jälkeen aloitussivu käynnistetään omassa paikallisessa kehitysympäristössä ja kehitystyö voi alkaa käyttäen VsCode-ohjelmointiympäristöä.

Aloitussivulle ja informaatiolaatikoille luotiin omat tyylitiedostot, joissa pyrittiin luomaan aloitussivun ulkoasusta suunnitelman mukainen.

Kokonaisuudessaan sivu sisältää neljä informaatiolaatikkoa, ikoneita, tekstiä, painikkeita ja taustakuvan.

Seuraavaksi piti luoda vsCode-ohjelmointiympäristöön oma tiedosto aloitussivulle. Aloitussivun tiedostonimi on ModuleContainer.tsx.



Kuva 7. Uudistettu aloitussivu

6.5 Testaus

Aloitussivulle luotiin omat tiedostot yksikkötestaamista varten. Tällä tavoin varmistettiin, että kaikki tehdyt näkymät renderöityvät sivulle oikein.

Sivun näkymää varten luotiin oma ModuleContainer.test.js tiedosto, jossa testattiin, että sivustolle renderöityy otsikko ja neljä informaatiolaatikkoa.

```
describe(ModuleContainer.displayName, () => {
    it('should render all infoboxes', () => {
        // Act
        render(<ModuleContainer />);

        // Assert
        expect(screen.getByRole('list', { name: 'information areas' })).toBeTruthy();
        expect(screen.getAllByRole('listitem').length).toEqual(4);
        expect(screen.getByText('flow_start:header.wake_up_device')).toBeInTheDocument();
        expect(screen.getByText('flow_start:header.download_app')).toBeInTheDocument();
        expect(screen.getByText('flow_start:header.flow_sync')).toBeInTheDocument();
        expect(screen.getByText('flow_start:header.setup_with_watch')).toBeInTheDocument();
    });
});
```

Kuva 8. Otsikon ja informaatiolaatikoiden yksikkötestaus

Informaatiolaatikoille luotiin myös oma InfoBox.test.js tiedosto, jossa testattiin, että jokainen informaatiolaatikko saa oikean otsikon.

```
    it('should render infotext', () => {
        // Act
        render(<InfoBox infoText="this is infotext" />);

        // Assert
        expect(screen.getByText('this is infotext')).toBeInTheDocument();
    });
```

Kuva 9. Informaatiolaatikon sisäisen otsikon testaus

Samassa tiedostossa testattiin myös informaatiolaatikoiden sisältö. Yksikkötestissä testattiin, että jokainen informaatiolaatikko saa oikean ikonin.

```
it('should render phone image correctly', () => {
    // Act
    render(<InfoBox imageType={ImageType.PHONE} />);

    // Assert
    expect(screen.getByAltText('phone')).toHaveAttribute('src', 'phone_icon.png');
});

it('should render watch image correctly', () => {
    // Act
    render(<InfoBox imageType={ImageType.ONLY_WATCH} />);

    // Assert
    expect(screen.getByAltText('watch')).toHaveAttribute('src', 'watch_icon.png');
});

it('should render computer image correctly', () => {
    // Act
    render(<InfoBox imageType={ImageType.COMPUTER} />);

    // Assert
    expect(screen.getByAltText('computer')).toHaveAttribute('src', 'computer_icon.png');
});

it('should render device image correctly', () => {
    // Act
    render(<InfoBox imageType={ImageType.WAKEUP_DEVICE} />);

    // Assert
    expect(screen.getByAltText('wakeup device')).toHaveAttribute('src', 'wakeup_device_icon.png');
});
```

Kuva 10. Informaatiolaatikoiden sisällä olevien ikonien testaus

Kolme informaatiolaatikkoa sisältää erilaisia painikkeita. Ensimmäisen laatikon painike ohjaa käyttäjän Polarin YouTube-sivustolle. Toisessa laatikossa ohjeistetaan käyttäjää lataamaan Polar Flow-puhelinsovellus ja sitä varten on olemassa kolme erilaista painiketta. Käyttäjä pystyy lataamaan sovelluksen Android, Apple sekä Huawei-laitteelle. Kolmannessa laatikossa ohjeistetaan käyttäjää lataamaan FlowSync tietokoneelle, jonka avulla käyttäjä pystyy synkronoimaan laitteen datat Polar Flow -palveluun.

Neljäs informaatiolaatikko pitää sisällään tietoa siitä, jos käyttäjä haluaa ottaa laitteen käyttöön ilman Polar Flow -palvelun asennusta. Tämä informaatiolaatikko ei pidä sisällään painikkeita.

Viimeisessä yksikkötestissä katsottiin, että painikkeet sijaitsevat niille asetetuilla paikoilla.

```
it('should render buttons if set in props', () => {
    // Act
    render(
        <InfoBox
            buttons={
                <>
                    <button aria-label="button1">Button 1</button>
                    <button aria-label="button2">Button 2</button>
                </>
            }
        />
    );

    // Assert
    expect(screen.getByRole('button', { name: 'button1' })).toBeInTheDocument();
    expect(screen.getByRole('button', { name: 'button2' })).toBeInTheDocument();
});
```

Kuva 11. Informaatiolaatikoiden sisältämien painikkeiden testaus

Integraatiotestauksessa testattiin sivun toimivuutta. Testissä haluttiin, että aloitussivu aukeaa oikein, vaikka käyttäjä ei ole kirjautunut sisään palveluun. Testattiin myös, että palveluun kirjautunut käyttäjä saa aloitussivun auki. Painikkeiden testauksessa haluttiin nähdä, että ne aukeavat käyttäjälle oikein ilman virheitä.

7 YHTEENVETO

Tässä kappaleessa käydään läpi yleisesti opinnäytetyön prosessia ja teknisen osuuden onnistumista sekä mahdollista jatkokehitystä toimeksiantajalle.

Työ tarjosi tekijälle mahdollisuuden kehittää omaa osaamista ja ymmärrystä Polarilla käytetyistä teknologioista ja työtavoista.

7.1 Aloitussivun uudistus

Aloitussivun uudistaminen onnistui lähes täysin suunnitelman mukaan. Sivulle jouduttiin jälkikäteen suunnittelemaan vielä kielivalikko, joka ei ollut mukana alkuperäisessä suunnitelmassa. Tämä onnistui vaivattomasti, sillä vastaavia kielivalikkoja oli käytetty myös muualla Polar Flow -palvelussa. Kyseessä oli myös ensimmäinen sivu, johon käyttäjän kuului päästä ilman sisäänkirjautumista Polar Flow -palveluun. Tämä vaihe oli haastava, sillä Polar Flow -palvelu ei sisällä muita sivuja mihin kuuluisi päästä ilman kirjautumista.

Ohjelmointikielten harjoittelu otti aikansa, koska tekijällä ei ollut aiempaa kokemusta mm. Reactista tai TypeScriptistä. Työn edetessä tekijä pääsi opettelemaan ohjelmointia näillä kielillä ja työtapoja toimeksiantojen mukaan.

7.2 Käytetyt teknologiat

Työn haastavin osuus oli kertoa käytetyistä teknologioista, koska tekijä ei ole ollut töissä Polarilla kovin pitkään. Vanhojen käytettyjen teknologioiden omaksuminen tuotti hieman vaikeuksia. Kuitenkin työkavereiden opastuksella tekijä pääsi hyvin perille siitä, miten Polarin käytetyt teknologiat ovat kehittyneet ajan myötä.

Kokonaisuudessaan kyseinen osuus antoi tekijälle arvokasta oppia tulevia haasteita varten.

7.3 Jatkokehitys

Nykypäivänä frontend teknologiat kehittyvät nopeasti ja uusia ohjelmistoja sekä työkaluja julkaistaan jatkuvasti. On tärkeää, ettei kaikista uusimpiin teknologioihin siirrytä liian nopeasti, koska alkuvaiheessa ne saattavat olla epästabiileja eikä ole varmuutta sopivatko ne käytössä olevien teknologioiden korvaajaksi. On järkevää odottaa ja seurata vierestä mihin uudet teknologiat kykenevät ja mitkä niistä yleistyvät. Tässä kappaleessa käydään läpi uudempia ja tehokkaampia teknologioita, joita mahdollisesti voidaan ottaa käyttöön vanhempien käytössä olevien teknologioiden tilalle.

Monorepo työkaluja on kehitetty paremmiksi ja niitä on nykypäivänä useita eri tarkoituksiin sopivia. Käytössä oleva Create React App pohjainen ratkaisu käyttää Webpack- ja Babel ohjelmistoa ja nämä ohjelmistot on luotu JavaScriptillä.

Webpack luo riippuvuuskaavion yhdestä tai useammasta aloituspisteestä ja yhdistää projektin tarvitsemat moduulit yhdeksi tai useammaksi nipuksi, jotka ovat staattisia resursseja. Babel toimii kääntäjänä. Kyseessä on ohjelma, joka muuntaa yhden tyyppisen lähdekoodin toisen tyyppiseksi lähdekoodiksi. Babel ottaa JavaScript koodit (React, ECMAScript) ja muuntaa ne selaimen ymmärrettäväksi JavaScript koodiksi. (P. Srivastava 2021)

Kyseiset ohjelmistot eivät ole kovin tehokkaita ja niiden tilalle on kehitetty vastaavia työkaluja, jotka on luotu eri ohjelmointikielillä tehokkuuden parantamiseksi.

Babelin tilalle on kehitetty tehokkaampia ohjelmistoja ja yksi niistä on Esbuild. Esbuild on kirjoitettu Go ohjelmointikielellä, joka kykenee jakamaan muistia säikeiden välillä minkä seurauksena sillä on huomattavasti parempi suorituskyky kuin JavaScript pohjaisilla kääntäjillä. (F. Almasi 2022)

Webpackin kehittäjät ovat luoneet tehokkaamman työkalun nimeltään Turbopack joka on luotu Rust ohjelmointikielellä. Turbopack on huomattavasti tehok-

kaampi kuin Webpack, ja sen suorituskyky perustuu optimoituun koodiin ja inkrementaaliseen laskentaan, joka mahdollistaa välimuistin tallentamisen yksittäisten toimintojen tasolle. Kun Turbopack suorittaa tehtävän, se ei tee samaa tehtävää uudelleen. (Turbo.build 2022).

LÄHTEET

Baeldung. 2021. Templating in Play Framework with Scala. Viitattu 10.11.2022. https://www.baeldung.com/scala/play-templating

Codeberry school. 2020. Mitä on JavaScript-ohjelmointi? Viitattu 22.9.2022. https://codeberryschool.com/blog/fi/javascript-ohjelmointi-opas/

Cse.hut. 2016. Ohjelmoinnin peruskurssi. Viitattu 15.11.2022. http://www.cse.hut.fi/fi/opinnot/CSE-A1121/2015/yleista/index.html

Choi, D. 2020. Full-Stack React, TypeScript, and Node. Viitattu 3.10.2022. https://learning.oreilly.com/library/view/full-stack-react-typescript/9781839219931/B15508_01_Final_JC_ePub.xhtml#_idParaDest-18

Codecademy. 2022. Learn React Testing. Viitattu 25.8.2022. https://www.codecademy.com/learn/learn-react-testing/modules/react-testing-library/cheatsheet

Docs Scala-lang. 2022. Tour of Scala. Viitattu 10.8.2022. https://docs.scala-lang.org/tour/tour-of-scala.html

Goldberg, J. 2022. Learning TypeScript. Viitattu 25.8.2022. https://learning.oreilly.com/library/view/learning-typescript/9781098110321/ch01.html#idm45415820400352

Geeks for geeks. 2022. Difference between hot reloading and live reloading in react native. Viitattu 1.11.2022. https://www.geeksforgeeks.org/difference-between-hot-reloading-and-live-reloading-in-react-native/

Hurja. 2020. MVC for dummies: malli, näkymä ja ohjain-arkkitehtuuri web-sovelluksissa. Viitattu 26.10.2022. https://www.hurja.fi/blogi/mvc-for-dummies-malli-nakyma-ja-ohjain-arkkitehtuuri-web-sovelluksissa/

Itnext. 2018. React app monorepo with Lerna. Viitattu 6.9.2022. https://itnext.io/guide-react-app-monorepo-with-lerna-d932afb2e875

Javatpoint. 2021. JavaScript Tutorial. Viitattu 15.8.2022. https://www.javatpoint.com/javascript-tutorial

Javatpoint. 2021. React create-react-app. Viitattu 16.10.2022. https://www.javatpoint.com/react-create-react-app

Kocik, M. 2020. Yarn workspaces – monorepo beginner's guide. Viitattu 25.10.2022. https://medium.com/swlh/yarn-workspaces-monorepo-beginners-guide-ed89de47aa25

Lesscss. 2022. Overview. Viitattu 15.10.2022. https://lesscss.org/

Lerna.js. 2022. Introduction. Viitattu 21.10.2022. https://lerna.js.org/docs/introduction

Mezzalira, L. 2021. Building Micro-Frontends. Viitattu 10.10.2022. https://learning.oreilly.com/library/view/building-micro-frontends/9781492082989/ix01.html

Monorepo tools. 2022. What is a monorepo. Viitattu 12.10.2022. https://monorepo.tools/#what-is-a-monorepo

Medium. 2021. Setup react with webpack and babel. https://medium.com/age-of-awareness/setup-react-with-webpack-and-babel-5114a14a47e9

Narayn, H. 2022. Just React!. Viitattu 13.9.2022. https://learning.oreilly.com/library/view/just-react-learn/9781484282946/html/521019_1_En_1_Chapter.xhtml

Object computing. 2014. Scala and the Play Framework. Viitattu 7.8.2022. https://objectcomputing.com/resources/publications/sett/january-2014-scala-and-the-play-framework

ProjectTop. 2022. Testaussanasto – Ohjelmistotestauksen tärkeimmät termit selitettynä. Viitattu 1.9.2022. https://projecttop.com/testaussanasto/

Play Framework. 2021. Documentation. Viitattu 10.11.2022. https://www.playframework.com/documentation/1.2.7/templates

Reactjs. 2022. Create a new react app. Viitattu 20.10.2022. https://reactjs.org/docs/create-a-new-react-app.html

TypeScriptLang. 2022. TypeScript from scratch. Viitattu 5.10.2022. https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

Testing-library. 2022. React Testing Library. Viitattu 20.8.2022. https://testing-library.com/docs/react-testing-library/intro/

Turbo.build. 2022. Getting started with Turbopack. https://turbo.build/pack/docs

Vardan, H. 2020. Why is lerna so popular among developers? Viitattu 22.10.2022. https://betterprogramming.pub/why-is-lerna-so-popular-among-developers-ef78d965d3ea

Webtips. 2022. Why ESBuild is so fast? https://www.webtips.dev/setup-esbuild

Wikipedia. 2022. Unit testing. Viitattu 15.8.2022. https://en.wikipedia.org/wiki/Unit_testing

Polar Electro Oy. Who We Are. 2022 Viitattu 10.8.2022. https://www.polar.com/fi/about_polar/who_we_are