TYÖNOHJAUSSOVELLUKSEN WEB-KÄYTTÖLIITTYMÄPROTOTYYPIN KEHITTÄMINEN

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, Riihimäki

kevät 2021

Laura Haro

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka Tiivistelmä

Riihimäki

Tekijä	Laura Haro	Vuosi 2021
Työn nimi	Työnohjaussovelluksen web-käyttöliittymäprototyypin kehittäminen	
Ohjaajat	Petri Kuittinen	

TIIVISTELMÄ

Tämän opinnäytetyön tavoitteena oli toteuttaa prototyyppi Geometrix Oy:n työnohjaussovelluksen uudelle web-käyttöliittymälle. Prototyyppivaiheen tärkeimmät toteutettavat käyttötapaukset olivat työtehtävien hakeminen taulukkoon, työntekijäresurssien hakeminen puunäkymään sekä työtehtävien osoittaminen resursseille.

Web-sovellukset ovat monimutkaisia kokonaisuuksia, jotka koostuvat monista komponenteista. Ne voivat olla joko monen sivun sovelluksia tai yhden sivun sovelluksia. Työnohjaussovelluksen uusi käyttöliittymä toteutetaan React-sovelluksena. React on JavaScript kirjasto, jota käytetään front-end-, eli käyttöliittymäkehityksessä. Reactilla toteutetaan yhden sivun sovelluksia, jotka ajetaan selaimessa.

Kehitettävät ominaisuudet tunnistettiin haastatteluiden sekä Geometrixilla käydyn pohdinnan tuloksena. Haastattelut toivat sovelluksen loppukäyttäjiä myös lähemmäs kehittäjiä ja lisäsivät kehittäjien ymmärrystä loppukäyttäjien arjesta. Työnohjauksen web-käyttöliittymän arkkitehtuuri voidaan jakaa komponenttihierarkiaan ja tiedonvälitysarkkitehtuuriin. Komponenttihierarkia kuvaa komponenttien välisiä suhteita ja tiedonvälitysarkkitehtuuri tiedon liikkumista sovelluksessa. Työnohjauksen web-käyttöliittymä jakautuu kolmeen päänäkymään: tehtävänäkymään, karttanäkymään ja resurssinäkymään.

Avainsanat Front-end, käyttöliittymä, ohjelmistokehitys, web-sovellus

Sivut 57 sivua, joista liitteitä 2 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology
Riihimäki

Abstract

Author Laura Haro
Year 2021

Subject Developing a web user interface prototype for work management application

Supervisors Petri Kuittinen

ABSTRACT

The goal of this thesis project was to implement a prototype for the new web user interface of Geometrix Ltd's work management application. Retrieving tasks and showing them in a table, retrieving resources, and showing them in a tree view and pointing tasks to resources were the features that were implemented during the prototype phase.

Web applications are complex combinations of different components. They can be either multi page applications or single page applications. The new web user interface for work management is implemented as a React application. React is a JavaScript library which is used for building user interfaces as single page applications that run in a browser.

Two supervisors of Geometrix's client companies, who are end users of work management application, were interviewed to survey development targets for the new user interface. Features to be developed were defined based on the interviews and discussions within Geometrix's team. The interviews also brought the end users closer to the developers and increased the developers' understanding about the workdays of the end users.

The front-end architecture can be divided into a component hierarchy and a data transmit architecture. The component hierarchy describes the relations between components, and the data transmit architecture shows how the data travels in the application. The user interface of work management is divided into three main views: tasks view, map view, and resource view.

Keywords Front-end, software development, user interface, web-application
Pages 57 pages including appendices 2 pages

Sisällys

Kuvat, taulukot ja kaavat

Liitteet

1 Johdanto

Tämän opinnäytetyön tavoitteena oli toteuttaa prototyyppi Geometrix Oy:n (myöhemmin Geometrix) työnohjaussovelluksen uudelle web-käyttöliittymälle. Työnohjaussovellus on työtehtävien ja työntekijäresurssien hallintaan tarkoitettu web-sovellus. Sovellukselle kehitetään uusi käyttöliittymä nykyaikaisilla web-tekniikoilla.

Opinnäytetyön teoriaosuudessa käsitellään web-sovelluksia ja web-tekniikoita yleisesti, jonka jälkeen syvennytään Reactiin, moderniin JavaScript-kirjastoon, millä Työnohjaussovelluksen uusi käyttöliittymä toteutetaan. Näiden lisäksi teoriaosuudessa tarkastellaan JavaScript-kirjastoja ja React-moduuleja, joita käytetään käyttöliittymän kehittämisessä.

Toteutusosa jakaantuu kahteen osaan: käyttäjien ja käyttötapausten selvitysosaan sekä käyttöliittymäprototyypin toteutusosaan. Käyttäjien ja käyttötapausten selvitysosassa kuvataan, ketkä ovat työnohjaussovelluksen käyttäjiä ja mitkä ovat sovelluksen käyttötapaukset. Ohjelmistojen käyttöliittymäkehityksessä on tärkeää huomioida ne ihmiset, joille kyseistä sovellusta kehitetään eli ohjelmiston loppukäyttäjät. Kehityksessä pitää muistaa, että sovellusta ei tehdä itselle - se mikä kehittäjästä vaikuttaa loogiselta ja tärkeältä toiminnolta ei välttämättä ole sitä sovelluksen loppukäyttäjälle. Näin ollen, käyttöliittymäkehityksessä ei pidä tehdä oletuksiin perustuvia johtopäätöksiä, vaan todellisia kehitystarpeita tulee kartoittaa sovelluksen käyttäjiltä.

Käyttötapausten hahmottamisen tueksi järjestettiin kaksi käyttäjähaastattelua, joiden tarkoituksena oli tutustua sovelluksen käyttäjien arkeen ja niihin ongelmiin, joita työnohjaussovelluksella pyritään ratkaisemaan. Haastatteluilla selvitettiin myös, mitä puutteita nykyisessä työnohjaussovelluksessa on käyttäjien näkökulmasta ja miten niitä voitaisiin kehittää. Haastattelujen ja Geometrixin tiimin oman pohdinnan tuloksena koostettiin lista kehitettävistä ominaisuuksista.

Käyttöliittymäprototyypin toteutusosassa kuvataan työnohjaussovelluksen web-käyttöliittymän arkkitehtuuri kahdesta eri näkökulmasta: käyttöliittymäkomponenttien

hierarkian näkökulmasta sekä sovelluksen tiedonkulun näkökulmasta. Arkkitehtuurikuvausten lisäksi prototyypin toteutusosassa on käsitelty prototyyppivaiheen päätoiminnallisuudet yksityiskohtaisesti. Prototyyppivaiheessa toteutettiin sovelluksen layout, eli käyttöliittymän perusnäkymät sekä navigointi niiden välillä, tehtävien hakeminen listalle eli rajapintavastauksena saadun JSON-muotoisen tiedon näyttäminen taulukossa, resurssitiedon hakeminen ja näyttäminen puurakenteena sekä tehtävien osoittaminen resursseille.

Opinnäytetyön käsitteistö löytyy liitteestä 1 ja listaus kuvissa käytetyistä ikoneista löytyy liitteestä 2.

1.1 Työnohjaussovellus

Työnohjaussovellus on yksi Geometrixin Mobilenote-ohjelmiston moduuleista. Sovelluksen tarkoituksena on täsmällisen tiedon tarjoaminen kentällä suoritetuista töistä työnjohdolle. Työnjohto osoittaa päivän tehtävät valituille työryhmille karttapohjaisella käyttöliittymällä. Töiden etenemistä ja tilaa voidaan seurata reaaliaikaisesti, jolloin työkuorman jakaminen työntekijöiden välillä helpottuu ja resursoinnista tulee tehokkaampaa. (Geometrix Oy, 2020.)

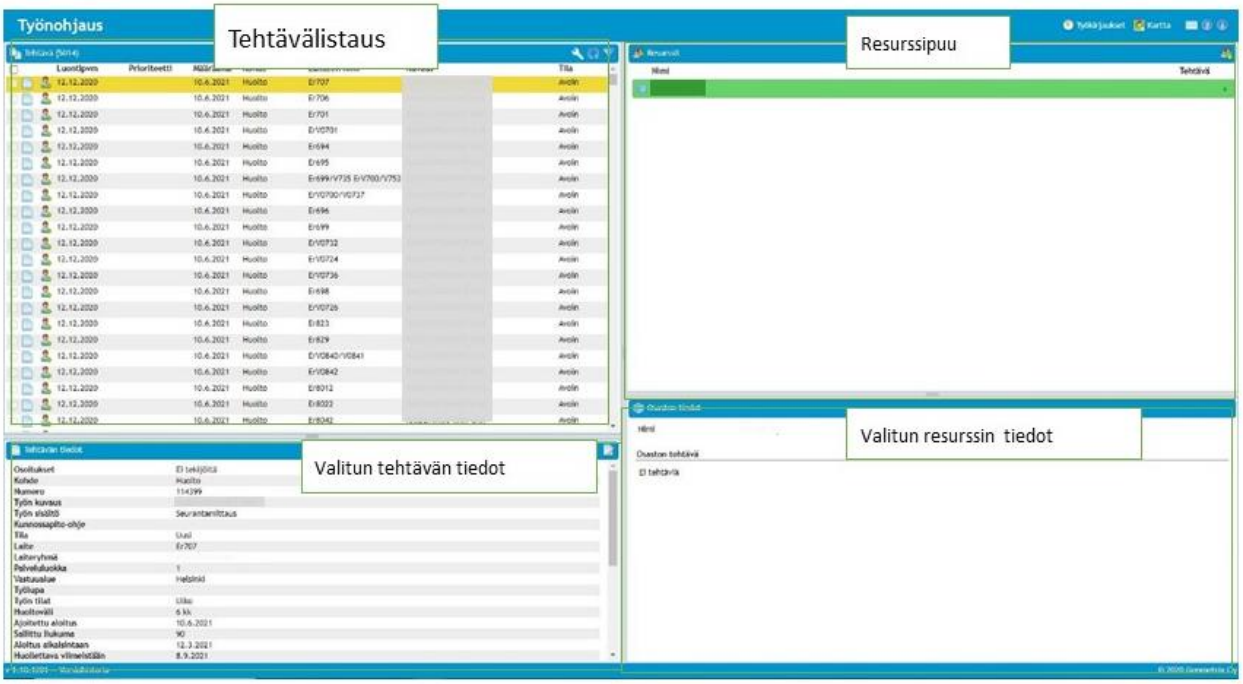
Kentällä työkohteet löytyvät karttapohjaisen mobiilisovelluksen avulla. Työntekijä voi merkitä työtehtävän tehdyksi työkohteessa sekä liittää raportille kuvia tekemistään huomioista. Tiedonkulku työntekijöiden ja työnjohdon välillä on sovelluksen avulla vaivatonta ja reaaliaikaista. Työnohjauksen kokonaisuus palvelee työnjohtajia tarjoamalla täsmällistä tietoa suoritetuista töistä ja töihin käytetystä ajasta sekä materiaaleista. (Geometrix Oy, 2020.)

1.2 Työnohjauksen nykyinen web-käyttöliittymä

Työnohjauksen nykyinen käyttöliittymä on toteutettu usean sivun MPA-sovelluksena ja se on samassa sovelluspaketissa palvelinpuolen (back-end) lähdekoodin kanssa. Työnohjauksen tämänhetkinen web-käyttöliittymä (front-end) on kehitetty JSP-tekniikalla ja se on suoraan yhteydessä palvelinpuolen Java-luokkiin. Työnohjauksen web-käyttöliittymä (Kuva 1) on

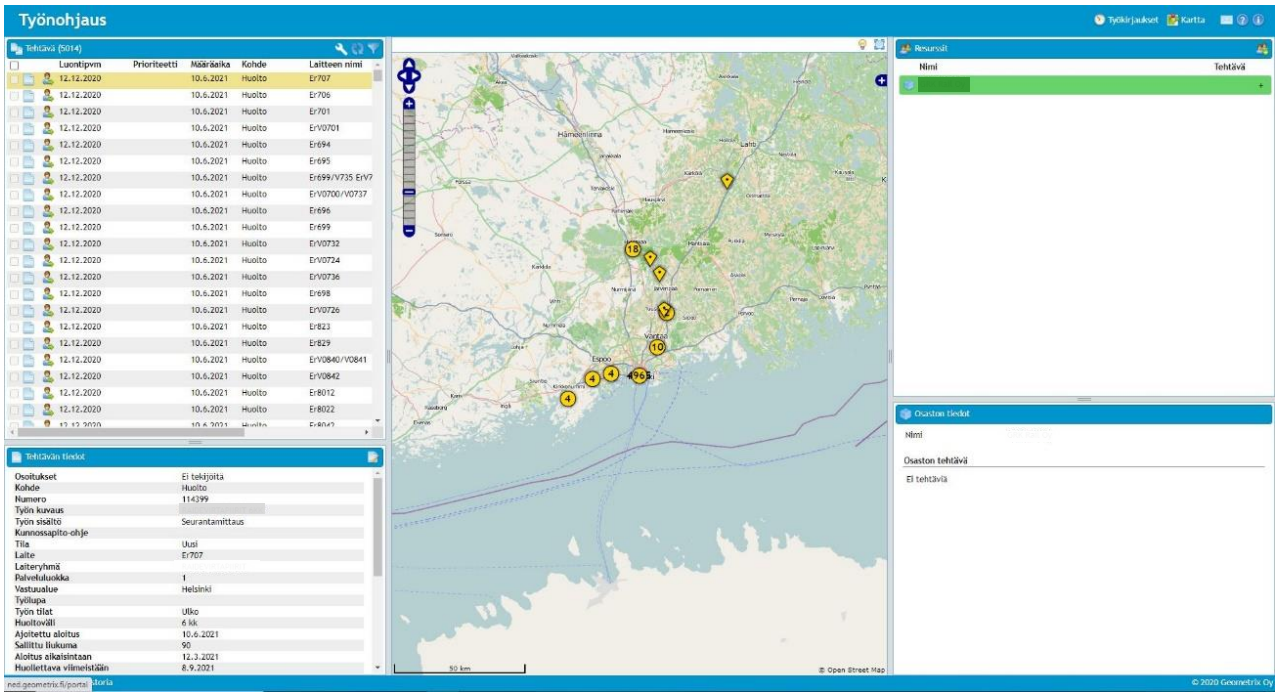
jäänyt teknisesti nykyajasta jälkeen ja nyt käyttöliittymän uusiminen on tarpeellista. Työnohjauksen visuaalisuutta kehitetään myös modernimpaan suuntaan.

Kuva 1. Työnohjauksen nykyinen käyttöliittymä



Jos käyttäjä haluaa nähdä kohteet kartalla, kartta avautuu nykyisessä käyttöliittymässä keskelle. Tällä toteutuksella se vie paljon tilaa tehtäviltä ja resursseilta. (Kuva 2.)

Kuva 2. Työnohjauksen nykyinen käyttöliittymä avatulla kartalla



Työnohjauksen käyttöliittymän uusimisessa on tarkoitus säilyttää kaikki olemassa olevat toiminnallisuudet. Nykyiset toiminnallisuudet on kuitenkin tarkoitus toteuttaa siten että niitä mahdollisesti täydennetään kehitysehdotusten pohjalta tai niitä kehitetään käytettävämmiksi. Kehitysehdotuksia kartoitettiin haastattelemalla kahden Geometrixin asiakasyrityksen työnjohtajia ja haastattelujen jälkeen tuloksia verrattiin Geometrixilla jo havaittuihin kehitettäviin ominaisuuksiin. Näiden havaintojen pohjalta laadittiin suuntaviivat sovelluksen uusimiselle ja tunnistettiin kehitettävät käyttötapaukset.

Työnohjauksen uusi käyttöliittymä toteutetaan yhden sivun SPA-sovelluksena React-kirjastolla. Tämän seurauksena sovelluksen web-käyttöliittymä toimii itsenäisenä sovelluskomponenttina eikä se ole enää samassa sovelluspaketissa palvelinpuolen sovelluksen kanssa. Näin ollen työnohjauksen kokonaisarkkitehtuuriin tulee myös muutos.

2 Web-sovellukset ja -tekniikat

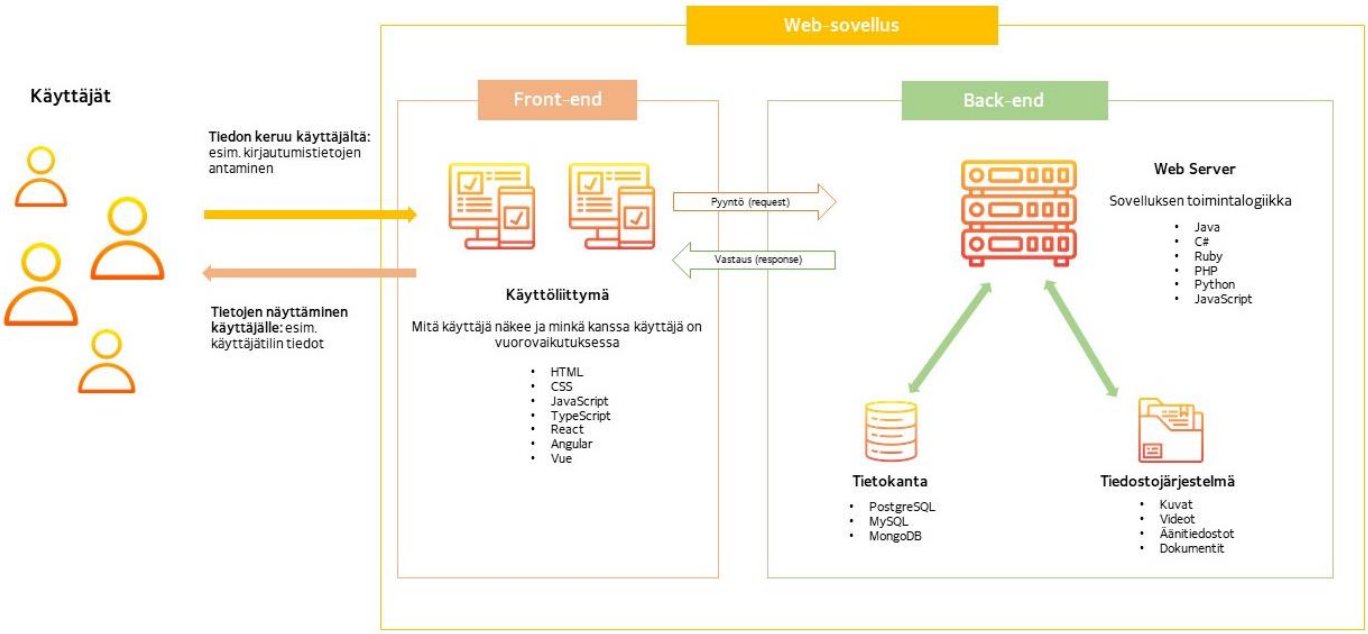
Web-sovellukset ovat mutkikkaita ohjelmistokokonaisuuksia. Niihin liittyy monia ohjelmistokomponentteja kuten käyttöliittymä, tietokanta, rajapinnat ja palvelimet. (Dabbs, 2019.) Web-sovellukset koostuvat kahdesta osasta: käyttöliittymästä (front-end) ja palvelinpuolen ohjelmistosta (back-end). Tieto liikkuu käyttöliittymän ja palvelinpuolen välillä rajapintojen (API) kautta. Web-sovelluksia käytetään selainohjelmien kautta ja niitä kehitetään web-tekniikoilla kuten HTML:llä, CSS:llä ja JavaScriptillä.

2.1 Web-sovellusten arkkitehtuuri

Kaikki mitä käyttäjä näkee ja minkä kanssa hän on vuorovaikutuksessa, kuuluu käyttöliittymään. Käyttöliittymän päätarkoitus on kerätä tietoa käyttäjältä. (Dabbs, 2019.) Kerättävä tieto voi olla esimerkiksi käyttäjän kirjautumistiedot tai painikkeen painallus, jolloin sovellus saa tiedon siitä, mikä toiminto sen täytyy suorittaa. Käyttöliittymän toteuttamiseen käytetään HTML:ää, CSS:ää ja JavaScriptia. Moderniin web-sovelluskehitykseen kuuluu jonkin sovelluskehyksen tai kirjaston, kuten Reactin, Angularin tai Vuen käyttäminen.

Palvelinpuolen ohjelmisto suoritetaan erillisellä palvelintietokoneella. (DigidWorks 2020.) Se on vastuussa tiedon käsittelystä ja säilyttämisestä. Palvelinpuoli prosessoi käyttöliittymän kautta saadut HTTP-pyynnöt, jolloin se esimerkiksi tallentaa tietoa tietokantaan tai hakee tietoa tietokannasta. Web-sovelluksen palvelinpuolen ohjelmointiin voidaan käyttää esimerkiksi Javaa, Pythonia, C#:ia, Rubya tai JavaScriptia. (Dabbs, 2019.) Kuvassa 3 on havainnollistettu web-sovellusten arkkitehtuuria (Kuva 3).

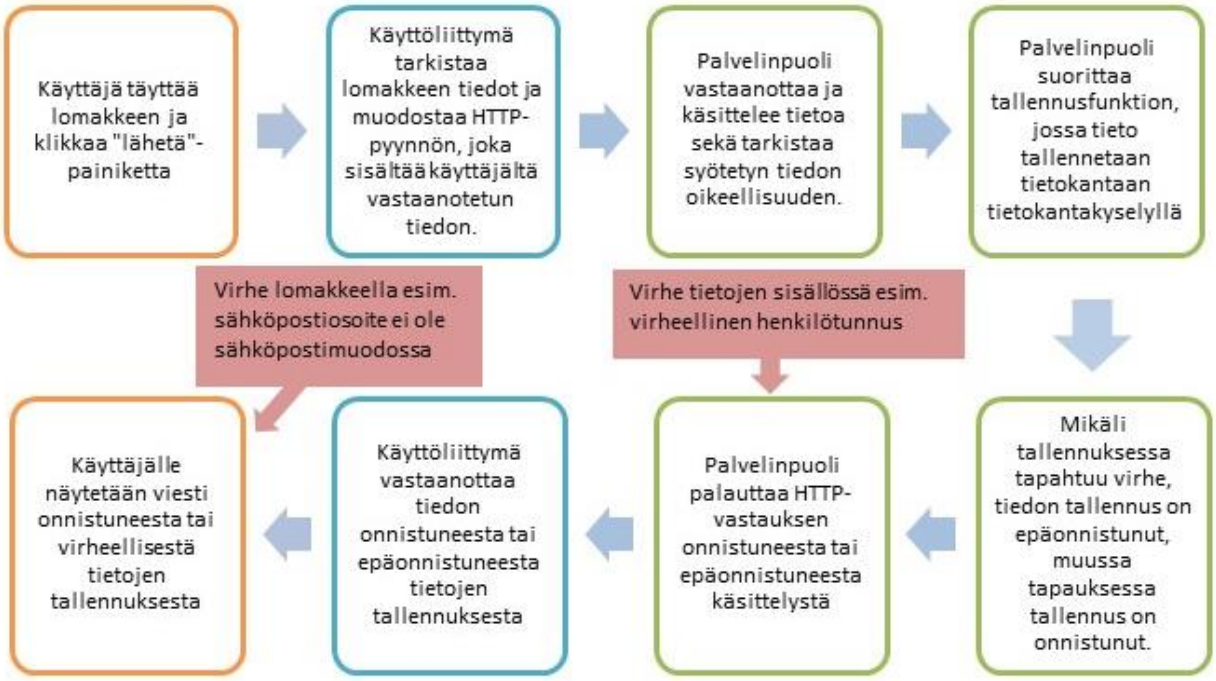
Kuva 3. Web-sovellusten arkkitehtuuri (Dabbs, 2019)



2.2 Tiedon kulkeminen web-sovelluksessa

Tiedon kulkeminen web-sovelluksessa lähtee liikkeelle käyttäjästä ja päätyy lopulta takaisin käyttäjälle (Kuva 4). Käyttäjä syöttää tietoa käyttöliittymän kautta, minkä seurauksena tiedonkäsittelyketju käynnistyy, jonka päätteeksi käyttäjä saa tiedon joko onnistuneesta tai epäonnistuneesta tapahtumaketjusta.

Kuva 4. Web-sovellusten tiedonkulku



Kuvan 4 prosessikuvaus on yleisestä skenaariosta, jossa käyttäjä täyttää lomakkeen web-sovelluksessa tallentaakseen tietoja sovellukseen. Kyseessä voi olla esimerkiksi rekisteröityminen jonkin verkkopalvelun käyttäjäksi tai henkilötietojen täyttäminen verkkokauppaostosten yhteydessä.

Ensimmäisessä vaiheessa käyttäjä täyttää lomakkeelle kysytyt tiedot ja lähettää lomakkeen klikkaamalla ”lähetä”-painiketta. Tämän jälkeen suoritetaan mahdollinen käyttöliittymäpuolen virheentarkistus, jossa varmistetaan, ovatko lomakkeen kentät täytetty oikein. Kyse voi olla esimerkiksi sähköpostin muodon tarkistamisesta tai pakollisten kenttien täyttämisen varmistamisesta. Nykyään lomakkeen tarkistus tehdään usein ennen painikkeen painamista, jolloin käyttäjä voi korjata tietoja heti kun lomake havaitsee, että kyseinen kenttä on täytetty väärin. Virhe voidaan tässä tilanteessa informoida punaisella värillä tai virheviestillä. Virheiden näyttäminen mahdollisimman aikaisessa vaiheessa parantaa sovelluksen käytettävyyttä ja käyttäjäkokemusta.

Jos sovelluksen käyttöliittymä ei havaitse virheitä syötetyssä tiedossa, muodostetaan HTTP-pyyntö, jolla tieto välitetään sovelluksen palvelinpuolelle. Palvelinpuolella lomakkeen tiedot

käsitellään sellaiseen muotoon, että tiedot voidaan tallentaa tietokantaan. Tiedon käsittely voi olla esimerkiksi eri objektien muodostamista, päivämäärien käsittelyä string-tietotyypistä date-tietotyypiksi tai laskentaa. Jos kyseessä olisi pyyntö, jolla haetaan tietoa näytettäväksi käyttäjälle, tiedon käsittely tapahtuisi toiseen suuntaan. Tällöin palautettavan tiedon pitää olla sellaisessa muodossa, että sen käsittely käyttöliittymässä on mahdollista.

Palvelinpuolella tehdään myös virheentarkistusta. Virheentarkistus suoritetaan sen takia, että palvelinpuoli toimii portinvartijana käyttöliittymän ja tietokannan välillä. Se on näin ollen vastuussa tietokantaan tallennettavan tiedon oikeellisuudesta. Palvelinpuolen virheentarkistus toimii myös varmistuksena sille, jos käyttöliittymäpuolen virheentarkistus on puutteellista. Jos palvelinpuolen virheentarkistus havaitsee virheen tiedon sisällössä, HTTP-pyynnön vastauksena palautetaan virhekoodi ja virheilmoitus ja käyttöliittymässä näytetään ilmoitus käyttäjälle.

Tiedon käsittelyn ja virheentarkistuksen jälkeen palvelinpuoli suorittaa tallennuksen tietokantaan. Mikäli tallennuksessa ei tapahdu virhettä, tallennus onnistuu ja tietokantaan syntyy uusi rivi. Tallennuksessa voi tapahtua esimerkiksi verkkovirhe, jolloin sovellus ei saa yhteyttä tietokantaan. Tallennuksen jälkeen palvelinpuoli palauttaa tiedon onnistuneesta tallennuksesta tai virheilmoituksen, jonka käyttöliittymä näyttää käyttäjälle.

2.3 Yhden ja usean sivun sovellukset

Web-sovelluksen toteuttamiselle on kaksi lähtökohtaista toteutustapaa: yhden sivun sovellus (single page application eli SPA) tai usean sivun sovellus (multi page application eli MPA). Merkittävä ero SPA-sovellusten ja MPA-sovellusten välillä on se, että MPA-sovellus päivittää HTTP-pyynnön seurauksena koko HTML-sivun, kun taas SPA-sovellus päivittää vain tiedon, joka näkyy sivulla. (Valuyi, 2020).

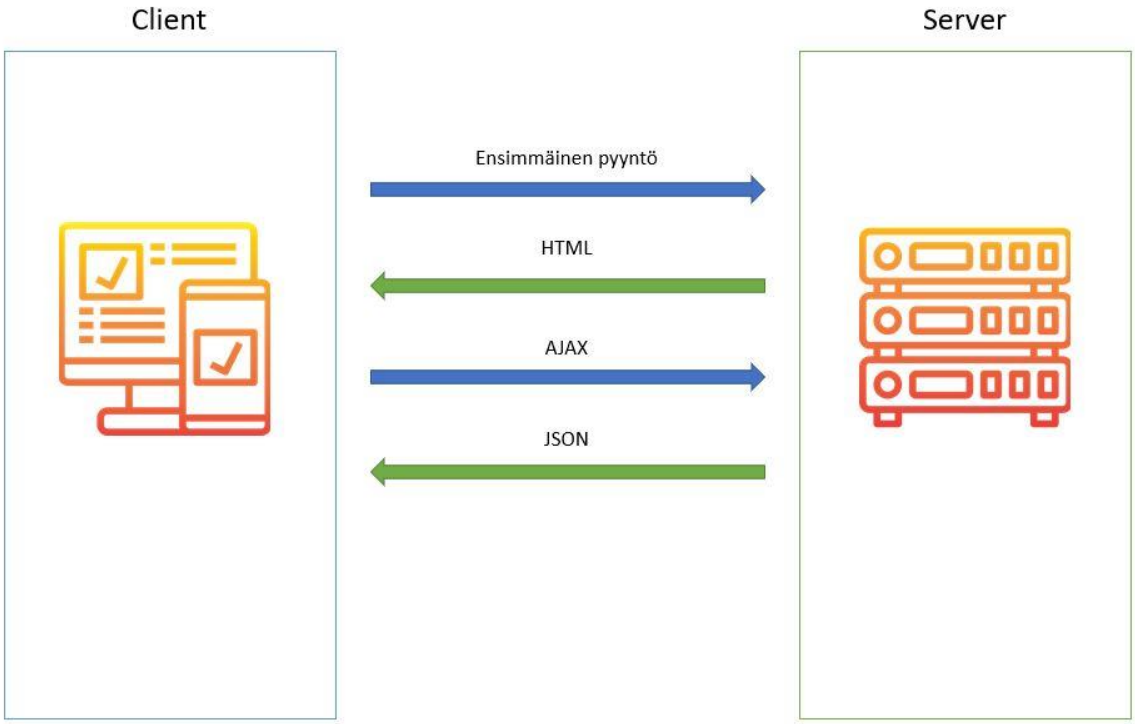
2.3.1 SPA-sovellukset

SPA-sovellukset ovat muihin sovellustyyppeihin verrattuna kevytrakenteisempia. Ne eivät vaadi HTML-sivujen uudelleenlataamista eikä käyttäjien tarvitse odottaa web-sivuille pääsyä pitkiä aikoja. Responsiivisen, eli erikokoisille näytöille skaalautuvan, käyttöliittymän

luominen on myös helpompaa yhden sivun sovelluksissa. Monet maailmanlaajuisesti tunnetut sovellukset kuten Gmail, Twitter, Facebook, Google Maps, Airbnb ja Pinterest ovat SPA-sovelluksia. (Valuyi, 2020).

SPA-sovellus hakee alkuperäisen HTML-sivun vain kerran, jonka jälkeen ainoastaan sivulla näytettävä tieto päivittyy rajapintapyyntöjen perusteella (Kuva 5).

Kuva 5. SPA-sovelluksen päivittymisen elinkaari (Valuyi, 2020.)



2.3.2 SPA-sovellusten edut

SPA-sovellusten sisällön latausajat voivat olla usean sivun sovelluksiin verrattuna moninkertaisesti nopeampia. Tämä johtuu siitä, että ne sisältävät vähemmän erillisiä sivuja ja informaatiota kuin MPA-sovellukset sekä siitä, että tiedostot ladataan vain kerran. Sovelluksen nopeus on merkittävä tekijä käyttäjäkokemuksen kannalta. (Valuyi, 2020).

SPA-sovellusten kehittäminen on helpompaa ja nopeampaa, koska sivuja tarvitaan vähemmän, toteutettavia ja testattavia toiminnallisuuksia on vähemmän ja näytettävää sisältöä on vähemmän. Näin ollen laadukkaan sovelluksen julkaisemiseen tarvitaan vähemmän resursseja. Sovelluksen ylläpito ja monitorointi julkaisun jälkeen on myös yksinkertaisempaa. (Valuyi, 2020).

SPA-sovelluksissa käytetään tehokkaasti välimuistia, mikä mahdollistaa sovelluksen käyttämisen ilman internet-yhteyttä silloin kun käyttäjä on jo kerran vieraillut sovelluksessa. Offline-tilassa välimuistiin tallennettu tieto synkronoidaan sovelluksen kanssa, kun internet-yhteys on palautettu. (Valuyi, 2020).

SPA-sovellus toimii myös pohjana mobiilisovellukselle. Sovelluksen palvelinpuolen koodia voidaan hyödyntää mobiilisovelluksessa käyttämällä samoja rajapintoja kuin mitä web-sovelluksen web-käyttöliittymä käyttää. (Valuyi, 2020).

2.3.3 SPA-sovellusten haitat

SPA-sovellusten kehittäjät kohtaavat toistuvasti haasteita sivuston indeksoinnissa ja sijoittumisessa hakukonetuloksissa korkealle. Tämä on kuitenkin tunnistettu ongelma ja sen ratkaisemisen eteen on nähty paljon vaivaa. Esimerkiksi Google on esitellyt uuden skeeman, joka sallii SPA- sovelluksille yhtäläiset mahdollisuudet hakukoneoptimointiin kuin MPA-sovelluksilla. (Valuyi, 2020).

SPA-sovellukset ovat alttiimpia XSS-hyökkäyksille. XSS-hyökkäyksessä hyökkääjä syöttää haitallisia skriptejä sovellukseen, jonka seurauksena voi olla käyttäjien arkaluontoisen datan vuoto. Nämä haavoittuvuudet johtuvat siitä, että joskus kokemattomat web-kehittäjät siirtävät joitain ominaisuuksia ja logiikkaa käyttöliittymän puolelle palvelinpuolelta. (Valuyi, 2020).

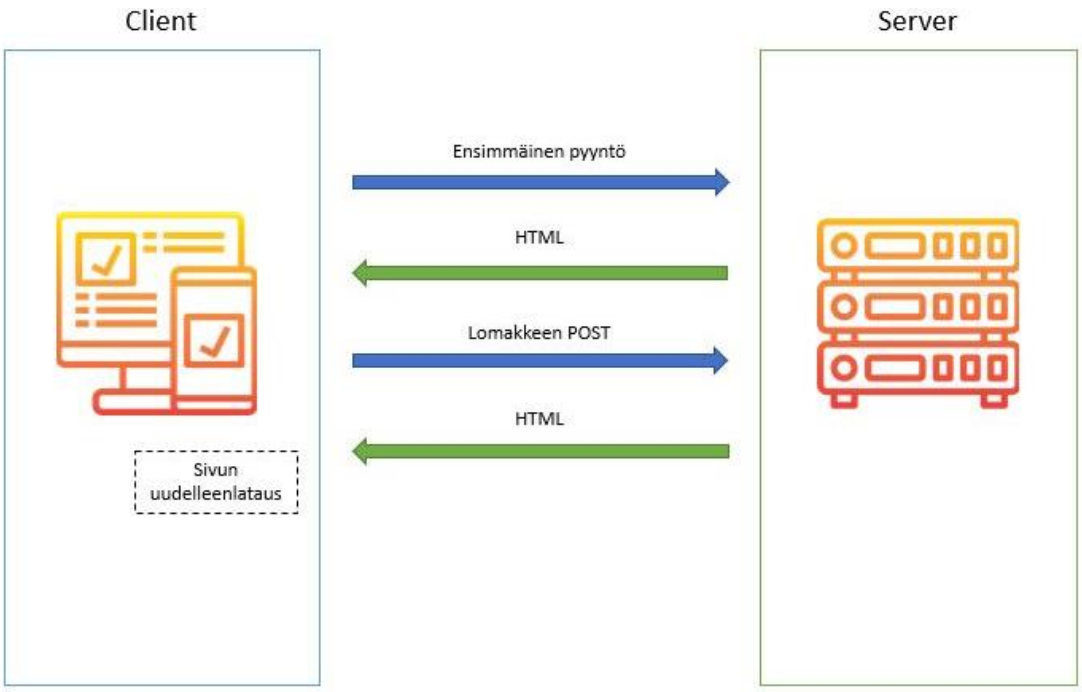
2.3.4 MPA-sovellukset

MPA-sovellukset ovat tyypillisesti hyvin laajoja; niissä on paljon eri sivuja ja kerroksia, mikä tekee niistä usein raskaita ja arkkitehtuuriltaan monimutkaisia. Monet verkkokaupat, kuten

Amazon ja verkko-oppimisympäristöt kuten Udemy ovat hyviä esimerkkejä MPA-sovelluksista. MPA-sovellusten laajuus aiheuttaa haasteita järjestelmän skaalattavuuteen ja ylläpidettävyyteen. Näiden ongelmien välttämiseen on kuitenkin nykyään olemassa runsaasti edistyneitä teknisiä ratkaisuja. Esimerkiksi mikropalveluarkkitehtuurilla tai serverless-arkkitehtuurilla on mahdollista käsitellä ja siirtää suuria määriä tietoa tehokkaasti. Arkkitehtuurin monimutkaisuus johtaa siihen, että sovellusten ylläpito ja tarvittaessa modernisointi on kallista ja hidasta. (Valuyi, 2020.)

MPA-sovellukset hakevat jokaisella pyynnöllä uuden HTML-sivun palvelimelta (Kuva 6).

Kuva 6. MPA-sovelluksen päivittymisen elinkaari (Valuyi, 2020.)



2.3.5 MPA-sovellusten edut

MPA-sovelluksissa sisältö päivittyy jatkuvasti, ja niiden sisältämät useat HTML-sivut tarjoavat monipuolisen alustan, jolle voi lisätä useita avainsanoja, kuvia ja metatietoja. Tämän seurauksena MPA-sovellukset ovat hakukoneystävällisempiä kuin SPA-sovellukset. (Valuyi, 2020). Mikäli kyseessä oleva sovellus on esimerkiksi verkkokauppa, joka tavoittelee korkeaa

sijoitusta Googlen luonnollisissa hakutuloksissa, niin sovellus kannattaa todennäköisemmin toteuttaa MPA-sovelluksena. (Valuyi, 2020).

MPA-sovelluksiin on saatavilla työkaluja tiedon analysointiin, jotka voivat tarjota tietoa esimerkiksi käyttäjän käyttäytymisestä ja järjestelmän toimivuudesta. Tietoa on saatavilla jokaisen sivun ja ominaisuuden suosiosta ja suorituskyvystä, siitä, miten paljon aikaa käytetään mihinkin ominaisuuteen, päivittäisten, viikoittaisten ja kuukausittaisten käyttäjien määristä, sekä yleisön segmentoinnista muun muassa iän tai sijainnin perusteella. (Valuyi, 2020).

2.3.6 MPA-sovellusten haitat

Koska MPA-sovellukset ovat laajempia kokonaisuuksia, niiden luominen vaatii enemmän resursseja. Kehittämiseen käytetty aika kasva suhteessa koostettavien sivujen sekä toteutettavien toiminnallisuuksien määrään. Näin ollen valmiin sovelluksen julkaisemiseen tarvitaan enemmän aikaa ja rahaa kuin yhden sivun sovelluksen julkaisuun. Tästä syystä myös julkaisun jälkeinen ylläpito ja jatkokehittäminen on vaikeaa ja kallista. (Valuyi, 2020).

MPA- sovellusten sisältämän sisällön määrä on usein valtava ja koska sovellus ladataan jatkuvasti uudelleen, palvelimet kuormittuvat enemmän kuin yhden sivun sovelluksissa. Tämä voi vaikuttaa web-sivujen nopeuteen ja järjestelmän suorituskykyyn negatiivisesti. (Valuyi, 2020).

2.3.7 SPA- ja MPA-sovellusten vertailu

Kun ohjelmistoprojektille tehdään päätöstä SPA-sovelluksen ja MPA-sovelluksen toteuttamisen välillä, toiminnan tarpeet ja tekniset vaatimukset täytyy ensin analysoida. Halutaanko sovellukseen sisällyttää paljon tietoa? Miten paljon ominaisuuksia ja sivuja tarvitaan? (Valuyi, 2020).

Mikäli kyseessä on monimutkainen sovellus, kuten verkkopankki tai lentojen varausjärjestelmä, jossa on runsaasti eri toiminnallisuuksia, MPA-sovellus on todennäköisemmin oikea ratkaisu. Useimmiten MPA-sovellus on kannattavampi

lähestymistapa isoille yrityksille. Myös silloin, kun päätavoitteena on hakukoneoptimointi, MPA-sovellus on parempi valinta. (Valuyi, 2020).

Jos taas tavoitteena on julkaista dynaaminen sovellus suhteellisen pienellä tietomäärällä, niin SPA-sovellus on oikea ratkaisu. Yhden sivun sovellukset soveltuvat erityisesti verkkoyhteisöjen kehittämiseen ja software-as-a-service-pavelujen (SaaS) kehittämiseen. (Valuyi, 2020).

Työnohjaussovelluksen kehittämisessä päädyttiin toteuttamaan uusi web-käyttöliittymä SPA-sovelluksena. SPA-sovellus on parempi valinta, koska sovellus on dynaaminen SaaS-palvelu Geometrixin asiakasyrityksille.

2.4 Selaimet

Web-sovelluksia käytetään selainohjelmien kautta. Selaimet hakevat tietoa ja näyttävät sitä käyttäjille ymmärrettävällä tavalla, kuten tekstinä ja kuvina. Tietoa siirretään HTTP-protokollalla, mikä määrittelee miten teksti, kuvat ja videot välitetään internetissä. Kaikki selaimet eivät kuitenkaan tulkitse formaatteja täysin samalla tavalla. Käyttäjille tämä tarkoittaa sitä, että sivusto saattaa näyttää erilaiselta ja toimi eri tavalla eri selaimilla. Web-standardeilla pyritään luomaan yhdenmukaisuutta selaimien välille. (Mozilla, n.d). Taulukossa 1 on esitetty suosituimmat selaimet vuonna 2020 (Taulukko 1).

Taulukko 1. Suosituimmat selaimet 2020 (GlobalStats, 2020.)

Selain	Markkinaosuus	Omistaja
Chrome	66,3 %	Google
Safari	16,76 %	Apple
Firefox	4,08 %	Mozilla-projekti (avoimen lähdekoodin ohjelmisto)
Samsung Internet	3,24 %	Samsung
Edge	2,61 %	Microsoft
Opera	2,06 %	Opera Software

Taulukosta 1 nähdään, että Googlen Chrome-selaimen markkinaosuus on huomattavasti muiden selaimien osuuksia suurempi. Taulukon selaimista Samsung Internet, Edge ja Opera käyttävät samaa Googlen kehittämää JavaScript-moottoria (Chromiumia ja V8:a) kuin Chrome (Biro, 2019.). Näin ollen Chromen markkinaosuus on teknisestä näkökulmasta

taulukossa 1 esitettyä lukua suurempi ja esimerkiksi Firefox jää Chromen ja Safarin rinnalla marginaaliselaimeksi. Firefoxin osalla 4,08 % markkinaosuus tarkoittaa kuitenkin yli 195 miljoonaa ihmistä, kun puhutaan kaikista internetin käyttäjistä, joten Firefoxia ei pidä unohtaa web-sovelluksia ja sivustoja kehitettäessä. Internetin käyttäjiä on noin 4,8 miljardia maailmanlaajuisesti (Worldometer, 2021). Suosituimpien selaimien rajautuminen kolmeen vaihtoehtoon tekee käyttöliittymätestauksesta helpompaa. Käyttöliittymän toimivuutta ei tarvitse enää testata useilla eri selaimilla ja joissain tapauksissa ainoastaan Chromella testaaminen saattaa riittää. Tähän vaikuttaa se, millä selaimella sovellusta todellisuudessa käytetään.

Selain noutaa verkkoon liitetyltä palvelimelta tietoa, jonka jälkeen se käyttää renderöintimoottoriksi kutsuttua ohjelmaa, joka kääntää tiedon tekstiksi ja kuviksi. Tämä tieto on kirjotettu HTML-kuvauksielellä (Hypertext Markup Language). (Mozilla, n.d.).

Hyperlinkkien kautta käyttäjät voivat siirtyä verkossa web-sivulta toiselle. Jokaisella sivulla, kuvalla ja videolla on oma, uniikki URL (Uniform Resource Locator), eli web-osoite. Kun selain vierailee palvelimella hakeakseen tietoa, web-osoite kertoo mistä mikäkin HTML:ssä kuvattu tieto löytyy. HTML-kertoo selaimelle missä ja miten tieto esitetään sivulla. (Mozilla, n.d.).

2.4.1 Evästeet

Verkkosivut tallentavat tietoa käyttäjistä evästeiksi kutsuttuihin tiedostoihin. Ne tallennetaan käyttäjän tietokoneelle seuraavaa vierailua varten. Kun käyttäjä vierailee sivustolla uudestaan, sivuston koodi lukee tiedoston. Evästeet mahdollistavat muun muassa käyttäjän kirjautumistietojen muistamisen. (Mozilla, n.d.).

Jotkin evästeet keräävät käyttäjistä yksityiskohtaisempaa tietoa. Näitä voivat olla esimerkiksi kiinnostuksen kohteet tai selaustavat. Tämän seurauksena sivustot voivat tarjota kohdennettua sisältöä, usein mainoksina. On myös olemassa kolmannen osapuolen evästeitä, jotka tulevat sivuilta, joilla käyttäjä ei vieraile kyseisellä hetkellä, ja jotka seuraavat käyttäjää sivulta sivulle keräten tietoa käyttäjästä. Tämäntyyppisten evästeiden estäminen on yleensä mahdollista, mutta kaikki selaimet eivät salli evästeiden estämistä. (Mozilla, n.d.).

2.4.2 Yksityisyys

Suurin osa isoimmista selaimista tarjoaa mahdollisuuden käyttää verkkoa yksityisesti. Yksityisen selaamisen tila piilottaa selaushistorian muilta saman tietokoneen käyttäjiltä. Yksityisenä tai tuntemattomana (incognito) selaaminen ei kuitenkaan piilota identiteettiä ja historiaa verkkopalvelujen tarjoajilta, mainostajilta tai valtiolta. Yksityisen salaamisen tila ainoastaan poistaa historian omalta tietokoneelta, mikä on hyödyllistä, jos yleisellä tai jaetulla koneella käsitellään sensitiivistä tietoa. (Mozilla, n.d.).

2.5 Ohjelmointikielet

Ohjelmiston ohjelmakoodi kirjotetaan ohjelmointikielillä. Niillä luodaan ohjelmiston toimintalogiikka. Ohjelmakoodi on ihmisen antama ohjeistus tietokoneelle siitä mitä halutaan tapahtuvan. Tietokoneet ymmärtävät binäärejä ja ohjelmointikielet ovat keino kääntää nollat ja ykköset sellaiseen muotoon, mitä ihminen voi lukea ja kirjoittaa. (Codecademy, 2020).

2.5.1 Matalan ja korkean tason ohjelmointikielet

Ohjelmointikielet voidaan jakaa kahteen kategoriaan: matalan ja korkean tason ohjelmointikieliin. Matalan tason kielet ovat laiteläheisempiä. Näin ollen ihmisen on vaikeampi lukea niitä. Matalan tason ohjelmointikielten etu on niiden nopeus sekä se että tietokoneen toimintaa pystytään hallitsemaan tarkemmalla tasolla. (Codecademy, 2020).

Korkean tason ohjelmointikielet ovat lähempänä ihmisten tapaa kommunikoida. Korkean tason kielissä käytetään sanoja kuten ”object”, ”class”, ”order” ja ”add”, mitkä ovat lähellä niitä sanoja mitä käytetään jokapäiväisessä kommunikoinnissa. Täten ohjelmointi korkean tason ohjelmointikielillä on helpompaa, mutta niiden kääntyminen konekieleksi vie enemmän aikaa. (Codecademy, 2020.)

Tietokoneiden kehittyminen yhä tehokkaimmiksi on mahdollistanut sen, että ero matalan tason ja korkean tason kielten suoritusajassa on usein vain millisekunteja. (Codecademy, 2020.)

2.5.2 Suosituimmat ohjelmointikielet vuonna 2020

Ohjelmointikielten suosion tarkasteluun voidaan hyödyntää indeksejä ja tilastoja, jotka laskevat ohjelmointikielten suosiota erilaisten indikaattoreiden perusteella.

TIOBE-indeksin sijoitukset perustuvat siihen, miten monella web-sivulla kyseinen kieli mainitaan (PYPL Index, 2020). Sijoitusten laskentaan käytetään kaikkia suosituimpia hakukoneita kuten Googlea, Bingiä, Yahoo!:ta, Wikipediaa ja Baidua. (TIOBE Index, 2020). Taulukossa 2 on esitetty 10 suosituinta ohjelmointikieltä marraskuussa 2020 TIOBE-indeksin mukaan (Taulukko 2).

Taulukko 2. TIOBE-indeksi: 10 suosituinta ohjelmointikieltä marraskuussa 2020 (TIOBE Index, 2020.)

Sijoitus 2020	Ohjelmointikieli	Osuus (%)	Muutos (%)
1	C	16,21	+0,17
2	Python	12,12	+2,27
3	Java	11,68	-4,57
4	C++	7,60	+1,99
5	C#	4,67	+0,36
6	Visual Basic	4,01	-0,22
7	JavaScript	2,03	+0,10
8	PHP	1,79	+0,07
9	R	1,64	+0,66
10	SQL	1,57	-0,15

PYPL-indeksin sijoitukset lasketaan sen perusteella, miten usein tietyn ohjelmointikielen tutoriaalia haetaan Googlesta. Mitä enemmän tutoriaaleja haetaan, sen suositumpi kielen oletetaan olevan. (PYPL Index, 2020). Taulukossa 3 on esitetty 10 suosituinta ohjelmointikieltä PYPL-indeksin mukaan (Taulukko 3).

Taulukko 3. PYPL-indeksi: 10 suosituinta ohjelmointikieltä marraskuussa 2020 (PYPL Index, 2020.)

Sijoitus 2020	Ohjelmointikieli	Osuus (%)	Muutos (%)
1	Python	30,80	+1,8
2	Java	16,79	-2,3
3	JavaScript	8,37	+0,3
4	C#	6,42	-0,9
5	PHP	5,92	-0,2
6	C/C++	5,78	-0,2
7	R	4,16	+0,4
8	Objective-C	3,57	+1,0
9	Swift	2,29	-0,2
10	TypeScript	1,84	-0,0

Githubin ohjelmointikielten tilasto perustuu siihen, miten paljon kyseiselä kielellä toteutettuihin ohjelmistoprojekteihin (repository) on tehty lisäyspyyntöjä (pull request). Taulukossa 4 on esitetty 10 suosituinta ohjelmointikieltä Githubin ohjelmointikielten tilastoon (Taulukko 4).

Taulukko 4. Githubin ohjelmointikielien tilasto: 10 suosituinta ohjelmointikieltä vuoden 2020 viimeisellä kvartaalilla (Github Language Statistics, 2020.)

Sijoitus 2020	Ohjelmointikieli	Osuus (%)	Muutos (%)
1	JavaScript	18,77	-1,5
2	Python	16,49	-1,1
3	Java	11,55	+1,3
4	Go	8,13	-0,2
5	C++	7,00	+0,1
6	Ruby	6,95	+0,1
7	TypeScript	6,66	+0,4
8	PHP	5,57	+0,2
9	C#	3,67	0,0
10	C	3,13	+0,2

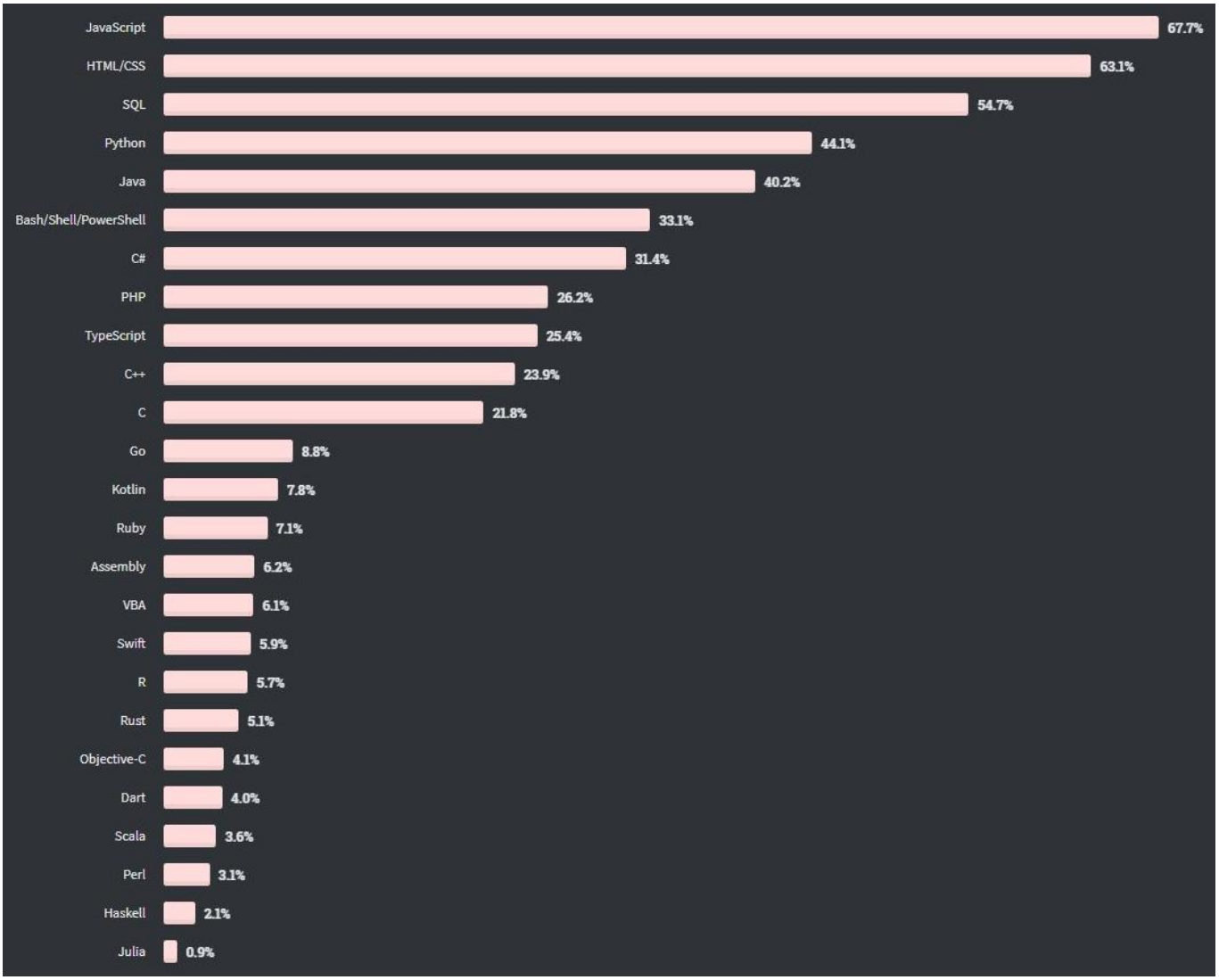
TIOBE-indeksin, PYPL-indeksin ja Githubin ohjelmointikielten tilaston perusteella voidaan päätellä, että käytetyimpiä ohjelmointikieliä vuonna 2020 ovat olleet Python, JavaScript ja Java. Verrattuna kahteen muuhun tilastoon, TIOBE-indeksin tarjoama tieto on harhaanjohtavaa. TIOBE-indeksi perustuu siihen, miten monta kertaa ohjelmointikieli mainitaan internetissä. Tämä ei kerro kovin paljon siitä, miten paljon kyseistä kieltä tänä päivänä käytetään.

PYPL-indeksi on parempi mittari kuin TIOBE-indeksi, koska se perustuu siihen, miten paljon kyseisen kielen tutoriaaleja haetaan Googlesta. Tieto on kuitenkin vain suuntaa antavaa. Tutoriaalien hakumäärä ei esimerkiksi kerro sitä, miten monta kertaa tutoriaaleja on tehty tai sitä, miten paljon kyseistä kieltä käytetään. Tutoriaalien hakumäärästä voidaan kuitenkin päätellä, mistä kielestä ollaan milloinkin kiinnostuneita.

Githubin ohjelmointikielten tilasto on näistä kolmesta tilastosta luotettavin, koska se kertoo eniten ohjelmointikielten todellisesta käytöstä. Taulukoista 3 ja 4 huomataan myös, että kolme suosituinta ohjelmointikieltä ovat samat. Tästä voidaan päätellä, että silloin kun ohjelmointikieltä käytetään paljon eli Githubiin tulee paljon lisäyspyyntöjä, on myös todennäköistä, että kyseiseen ohjelmointikieleen liittyviä tutoriaaleja haetaan Googlesta. Mielenkiintoista taulukoiden tiedoissa on JavaScriptin huomattavasti pienempi osuus PYPL-indeksissä, vaikka se on Githubin tilaston mukaan suosituin. Tämä selittyy ainakin osittain sillä, että Github tulkitsee JavaScriptilla tehdyt React-projektit pääosin JavaScriptiksi ja

Reactiin liittyvät tutoriaalit eivät näy JavaScriptin osuudessa PYPL-indeksissä. Indeksien lisäksi esimerkiksi ohjelmointiyhteisö Stack Overflow toteuttaa vuosittain ohjelmistokehittäjille suunnatun kyselytutkimuksen. Yhtenä kyselyn osiona selvitetään eri teknologioiden suosiota. Kuvassa 7 on esitetty suosituimmat ohjelmointi-, skripti- ja merkintäkielet vuoden 2020 kyselyssä (Kuva 7). Kysymykseen vastasi yli 57 000 henkilöä ja vastauksessa tuli valita useampia vaihtoehtoja. Kyselyn tuloksista huomataan, että JavaScript, Python ja Java ovat kehittäjiltä kysyttäessä viiden suosituimman ohjelmointikielen joukossa. Vaikka tulokset eroavat hieman toisistaan, voidaan yhteenvetona todeta, että tällä hetkellä kolme suosituinta ohjelmointikieltä ovat JavaScript, Python ja Java.

Kuva 7. Suosituimmat ohjelmointikielet Stack Overflown vuoden 2020 ohjelmistokehittäjäkyselyn mukaan (Stack Overflow, 2020.)



2.6 Web-tekniikat

Web-tekniikoita käytetään web-sovelluksen käyttöliittymän toteuttamiseen. (Schwartzmüller a, 2020). Web-sovellusten käyttöliittymien kehityksen selkärangan muodostavat HTML, CSS ja JavaScript (Morris n.d.). Kuvassa 8 on esitetty nämä kolme web-kehityksen pääelementtiä ja ne osa-alueet mitkä ne toteuttavat sivustolle (Kuva 8). HTML määrittelee mitä sivustolla on, CSS sen miltä sivustolla olevat asiat näyttävät ja JavaScript mitä sivustolla olevat asiat tekevät.

Kuva 8. Web-kehityksen kolme pääelementtiä (Morris, n.d.)

2.6.1 HTML

HTML (Hypertext Markup Language) on merkintäkieli, millä web-sivun rakenne ja sisältö jäsennetään. Sisältö voidaan jäsentää käyttämällä esimerkiksi tekstikappaleita, listoja, kuvia tai taulukoita. (MDN web docs a, 2020). Merkintäkielellä tekstiä merkitään siten että tietokone voi käsitellä sitä. Web-kehityksessä tyypillisimmin käytettyjä kuvauskieliä HTML:n lisäksi ovat XML ja XHTML. (Kyrnin, 2020).

HTML määrittelee web-sivun sisällön rakenteen. HTML-dokumentti koostuu elementeistä, joilla sisällön eri osat ympäröidään, jotta sisältö saadaan näkymään tai käyttäytymään tietyllä tavalla. (MDN web docs a, 2020).

2.6.2 CSS

CSS eli Cascading Style Sheets on tyylitiedostokieli, mitä käytetään merkintäkielillä kirjoitettujen dokumenttien esittämistavan määrittämiseen. CSS:llä kuvataan, miten elementit esitetään näytöllä, paperilla, puheessa tai muulla mediatyypillä. CSS kuuluu web-kehityksen ydintekniikoihin ja se on standardoitu W3C-vaatimusten mukaisesti selaimille. (MDN web docs b, 2020).

CSS-tyylejä määritellään tyylisäännöillä, joilla valitaan tietty elementti (MDN web docs c, 2020). CSS:llä kerrotaan esimerkiksi, minkä värinen elementin pitää olla, kuinka leveä ja korkea sen pitäisi olla, miten leveä marginaali sillä pitäisi olla ja millä elementin sivuilla tarvitaan marginaalia jne. Erilaisia CSS-sääntöjä on valtavasti ja CSS:llä voidaan toteuttaa vaativiakin visuaalisia tehosteita, kuten animaatioita.

2.6.3 JavaScript

JavaScript on korkean tason ohjelmointikieli, joka mahdollistaa monimutkaisten ominaisuuksien lisäämisen web-sivustoille. Aina kun web-sivu tekee jotain muuta kuin ainoastaan näyttää tietoa, on hyvin todennäköistä, että sivuston toteutuksessa on käytetty JavaScriptiä. (MDN web docs d, 2020). JavaScript on keskeinen osa web-kehitystä, joten kaikissa suosituimmissa selaimissa on sisäänrakennettu moottori sen kääntämiseen. Tämä tarkoittaa sitä, että JavaScriptiä voidaan kirjoittaa suoraan HTML-dokumenttiin ja selain ymmärtää nämä komennot eli JavaScriptin kääntämistä varten ei tarvita erillistä tulkkia. (Morris, n.d.)

JavaScriptiä käytetään yleisesti interaktiivisuuden lisäämiseen web-sivulle. Mikäli sivuston tarkoituksena on tehdä jotain muuta kuin näyttää staattista tietoa, nämä toiminnot toteutetaan yleensä JavaScriptillä. JavaScriptillä voidaan toteuttaa myös mobiilisovelluksia ja selainpohjaisia pelejä. Vaikka JavaScriptiä käytetään ensisijaisesti front-end-kehitykseen, sitä voidaan käyttää myös back-end-toteutuksiin. (Morris, n.d.)

2.7 Edistynyt JavaScript - kirjastot ja sovelluskehykset

Kun web-sovellusta kehitetään JavaScriptillä, vastaan tulee tilanteita, joissa sivulle lisättävä ominaisuus on sellainen, joka toistuu säännöllisesti web-sivuilla. Näitä ovat esimerkiksi valikoiden avaamisen ja sulkemisen animaatiot, erilaiset lomakkeet ja kirjautuminen sivustolle. Tällaiset ominaisuudet voidaan toteuttaa alusta lähtien itse, mutta nopeampi ja useimmiten kannattavampi tapa on käyttää hyvin testattua ja laajasti käytettyä sovelluskehystä tai ohjelmointikirjastoa. (Morris, n.d.)

Sovelluskehys tarjoaa kehitettävälle sovellukselle alustan, jonka päälle rakennetaan varsinainen toimiva ohjelma. Sovelluskehys voi esimerkiksi sisältää sisäänrakennettuja luokkia ja funktioita, joita voidaan käyttää esimerkiksi käyttäjältä saadun syötteen prosessointiin. (TechTerms, 2013.)

Kirjastojen ja sovelluskehysten ydinajatus on, että ohjelmistokehittäjien ei tarvitse keksiä pyörää uudestaan vaan toistuvien ongelmien ratkomiseen voidaan käyttää jo olemassa olevaa ratkaisua. (TechTerms, 2013.)

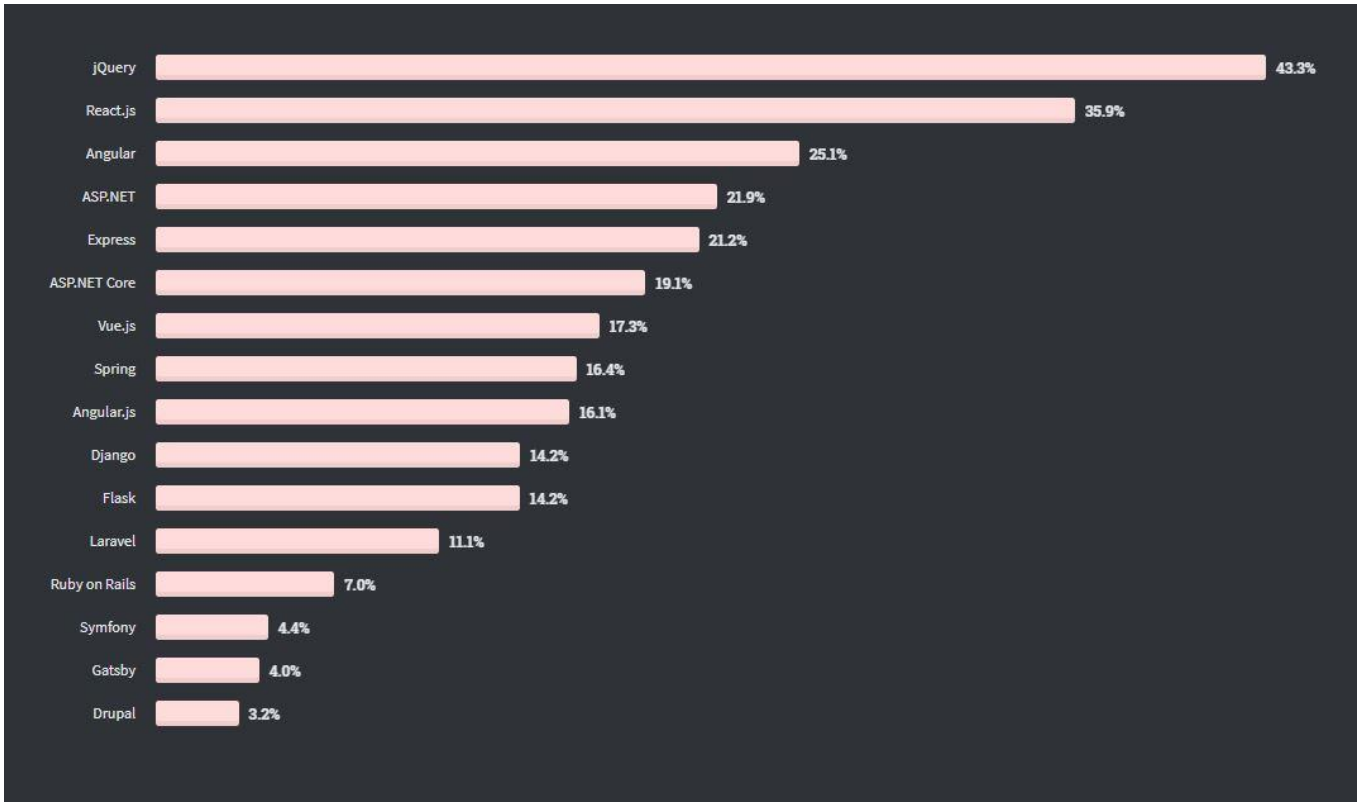
2.8 Suosituimmat JavaScript-sovelluskehykset ja kirjastot

JavaScript-sovelluskehykset ovat merkittävä osa modernia web-kehitystä. Ne tarjoavat kehittäjille testattuja työkaluja skaalattavien ja interaktiivisten sovellusten kehittämiseen. (MDN web docs, 2020-e). Sovelluskehykset tekevät dynaamisten sovellusten toteuttamisesta helpompaa. Uusia JavaScript-sovelluskehyksiä kehitetään nopealla tahdilla ja kehittäjille on tarjolla useita eri vaihtoehtoja.

Tämän hetken suosituimpiin moderneihin sovelluskehyksiin kuuluvat Vue, React ja Angular. Vue on kehittyvä sovelluskehys, joka lisää suosiota nopean oppimiskäyrän ja DOM-käsittelyn ansiosta. React on erittäin suosittu JavaScript-kirjasto, jolla on laaja osaajayhteisö. Reactin oppimiskäyrä on Vue:a jyrkempi. (Cardoso, 2020.) Tämän opinnäytetyön tuloksena toteutettavan Työnohjaussovelluksen uuden käyttöliittymän prototyypin kehittämiseen käytetään Reactia, joten sitä käsitellään tarkemmin kappaleessa 3. Angular on Googlen kehittämä raskaampi web-sovelluskehys, jossa ohjelmointiin käytetään TypeScript-kieltä. Angularin ensimmäinen versio AngularJS julkaistin ensimmäisen kerran vuonna 2010, mutta se poikkeaa hyvin paljon Angularin toisesta versiosta Angular 2:sta. Angularista puhuttaessa tarkoitetaan yleisesti Angular 2:sta ja sen jälkeen julkaistuja versioita. Muita saatavilla olevia JavaScript-sovelluskehyksiä ovat muun muassa Ember, Vueen perustuva Nuxt, Svelte, Vueen perustuva Gridsome, Backbone, Reactiin perustuva Gatsby ja Reactiin perustuva Next. (Cardoso, 2020.)

Kuvassa 9 on esitetty Stack Overflown tutkimuksen tulos, kun kehittäjiltä kysyttiin suosituimpia sovelluskehyksiä (Kuva 9). Tässä yhteydessä esimerkiksi React ja jQuery on luokiteltu sovelluskehyksiksi, vaikka ne ovat JavaScript kirjastoja.

Kuva 9. Suosituimmat web-sovelluskehykset Stack Overflown vuoden 2020 ohjelmistokehittäjäkyselyn mukaan (Stack Overflow, 2020.)



Kuvasta 9 nähdään, että perinteinen jQuery on edelleen kehittäjiltä kysyttäessä suosituin sovelluskehys. React on suosituin moderneista sovelluskehyksistä. Angularin osalta huomataan, että AngularJS ja Angularin muut versiot poikkeavat toisistaan sen verran, että ne on käsitelty omina sovelluskehyksinään.

3 React

React on käyttöliittymien toteuttamiseen tarkoitettu JavaScript-kirjasto. Se ei ole laaja sovelluskehys, vaikka Reactista usein puhutaan sovelluskehyksenä. Yleisesti voidaan sanoa, että React toteuttaa sovelluksen ”view”-kerroksen. (Boduch & Derks, 2020, s. 10-11.)

React on Facebookin kehittämä ja se julkaistiin toukokuussa 2013. React on suosittu tekniikka modernissa web-käyttöliittymäkehityksessä joustavuutensa ja yksinkertaisuutensa ansiosta. Koska React on suosittu ja vakiintunut JavaScript-kirjasto, siihen on saatavilla paljon hyvin dokumentoituja avoimen lähdekoodin kirjastoja. Reactilla toteutetaan JavaScript-pohjaisia SPA-sovelluksia. (Schwartzmüller, 2020 b.)

3.1 Komponentit

React perustuu komponenttiajatteluun ja Reactissa käyttöliittymiä koostetaan yhdistelemällä erilaisia komponentteja erilaisiksi kokonaisuuksiksi. Mikä tahansa web-sivu voidaan jakaa komponenteiksi. Esimerkkikomponentteja voisivat olla navigointipalkki, sivupalkki ja pääsisältö, johon sisältyvät otsikkokomponentti ja leipätekstikomponentti. (Schwartzmüller, 2020 b.)

Kun sovellus jaetaan komponenteiksi, näitä sovelluksen rakennuspalikoita voidaan toteuttaa yksittäisinä osina, eikä koko sovellusta tarvitse kehittää yhtenä isona kokonaisuutena. Tämä tekee kehitystiimin työskentelystä helpompaa ja tehokkaampaa, kun kehitettäviä ominaisuuksia voidaan jakaa työntekijöiden kesken sopivankokoisina tehtävinä. Komponentit tekevät sovelluksen koodista hallittavampaa ja ylläpidettävämpää. (Schwartzmüller, 2020 b.)

Komponentteja voidaan myös käyttää uudelleen helposti. Jos sovelluksessa on esimerkiksi tehtäväkomponentti ja sovelluksen pitäisi näyttää tehtävälista, samaa tehtäväkomponentin koodia voidaan käyttää useita kertoja ja varsinainen koodi ohjelmoidaan vain kerran. (Schwartzmüller, 2020 b.)

React-komponenttien voidaan ajatella olevan kustomoituja HTML-elementtejä. Komponentit ratkaisevat ongelman monimutkaisten käyttöliittymien toteuttamisesta vain niillä keinoilla, joita HTML ja JavaScript tarjoavat. (Schwartzmüller, 2020 b.)

3.1.1 JSX

React-komponentteja kirjoitetaan JSX-syntaksilla. JSX näyttää HTML:ltä, mutta se kääntyy JavaScriptiksi. Jokaisen komponentin pitää palauttaa JSX-koodia, koska se määrittelee mitä HTML:ää lopulta näytetään käyttäjälle. JSX ei ole HTML:ää, vaikka se näyttää hyvin samanlaiselta. Käytetään esimerkkinä TopBar-komponentin lähdekoodia (Kaava 1). Esimerkissä JSX-koodia on funktiosta palautettava osa. Tästä huomataan, että samanlaista HTML:n kanssa on puurakenne ja div-elementti. Eroavuutena nähdään, että esimerkiksi CSS-luokan asettaminen toimii eri tavalla: HTML-elementissä käytetään attribuuttia ”class” ja JSX:ssä attribuuttia ”className”. Lisäksi esimerkin ohjelmakoodi palauttaa muita komponentteja: AppBar, Toolbar ja Typography, joiden yhdistelmästä muodostuu TopBar-komponentti.

Kaava 1. TopBar-komponentti

```
import React from 'react';
import AppBar from '@material-ui/core/AppBar';
import Toolbar from '@material-ui/core/Toolbar';
import Typography from '@material-ui/core/Typography';


export const TopBar = ({classes, leftIcons, label, rightItemButtons}) =>
 {

return(
    <div>
        <AppBar position="static" className={classes.appBar}>
                <Toolbar className={classes.toolbar}>
                    {leftIcons}
                    <Typography variant="h6" className={classes.title}>
                        {label}
                    </Typography>
                    {rightItemButtons}
                </Toolbar>
        </AppBar>
    </div>
)
}
```

Kuten aikaisemmin mainittiin, JSX-koodi kääntyy JavaScriptiksi. Näin ollen React-komponentteja voidaan kirjoittaa myös ilman JSX:ää. React-komponentteja kirjoitetaan JavaScript-tiedostoihin, joihin React liitetään seuraavasti: ”*import React from ’react’;”*. Reactia voidaan käyttää elementtien luomiseen myös näin: React.createElement(’h1’, ’Tämä on otsikko’). Esimerkki palauttaa h1-elementin tekstillä ”Tämä on otsikko”. Esimerkki vaikuttaa yksinkertaiselta, mutta jos esimerkiksi kaavan 1 esimerkissä esitetty komponentti haluttaisiin kirjoittaa tällä tyylillä, koodista tulisi hyvin monimutkainen. React.createElement()-funktiota pitäisi kutua sisäkkäin useita kertoja ja esimerkkikomponentin koodi on vielä lyhyt. JSX mahdollistaa komponenttien kirjoittamisen HTML-tyylisellä syntaksilla, joka lopulta kääntyy React.createElement()-rakenteeksi. (Schwartzmüller, 2020 c.)

3.1.2 Funktionaaliset komponentit ja luokkakomponentit

Reactissa komponentteja voidaan luoda kahdella eri tyylillä: funktionaalisina komponentteina tai luokkakomponentteina. Kaavassa 1 luodaan funktionaalinen komponentti. Kyseessä on JavaScript-funktio, joka palauttaa JSX-koodia seuraavasti:

```
export const ExampleComponent = () => {
    return <div>JSX koodia</div>
}
```

Luokkakomponentit periytetään Component-luokasta ja luokan render()-metodi palauttaa JSX-koodia seuraavasti:

```
export class ExampleComponent extends Component {
     render () {
         return <div>JSX koodia </div>
     }
 }
```

Funktionaalisten komponenttien käyttö on yleisesti suositellumpi tapa kirjoittaa komponentteja, koska funktionaaliset komponentit ovat React-komponenttien yksinkertaisin muoto. (Schwarzmüller, 2020 d.) Työnohjaussovelluksen web-käyttöliittymän toteuttamisessa käytetään ainoastaan funktionaalisia komponentteja, joten tässä opinnäytetyössä keskitytään funktionaalisten komponenttien ominaisuuksiin.

3.2 Props-olio

Props on lyhenne sanasta ”properties” ja termiä käytetään yleisesti Reactin yhteydessä. Props-olio on keino välittää tietoa ”Parent”-komponentista ”Child”-komponenttiin. Komponentti, joka sisältää toisen komponentin, antaa sisältyvälle komponentille sen tarvitsemat parametrit props-oliona. (Ström, 2019.)

Aikaisemmassa kaavan 1 esimerkissä TopBar-komponentti saa sen ”Parent”-komponentilta props-oliossa parametrit: classes, leftIcons, label ja rightItemButtons. TopBar-komponentin ”Child”-komponentteja ovat AppBar, Toolbar ja Typography. Esimerkiksi Typography-komponentti saa TopBar-komponentilta variant-parametrin arvoksi ”h6”. Typography-komponentin renderöimä teksti on TopBar-komponentin label-parametrin arvo.

3.3 Tila

Joissain tilanteissa tietoa ei tarvita komponentin ulkopuolelta, vaan tietoa tarvitaan komponentin sisällä. Komponenttia voidaan muuttaa sisältä päin muuttamalla komponentin tilaa (state). Kaavassa 2 on esitetty, miten tilaa voidaan käyttää luokkakomponentissa (Kaava 2). Luokkakomponentilla on käytössä ”state”-objekti, jonka avulla komponentin tilaa voidaan hallita. Kaavan 2 tilan ”counter”-kenttä alustetaan arvoksi 1. Komponentti palauttaa ”div”-elementissä tämän arvon. Tila on tässä muodossa saatavilla ainoastaan luokkakomponenteissa. (Scwartzmüller, 2020 e.)

Kaava 2. Tila luokkakomponentissa (Schwarzmüller, 2020 f.)

```
export class Counter extends Component {
    state = {
        counter: 1
    };

    render () {
        return (
            <div>{this.state.counter}</div>
        );
    }
}
```

Reactin versiossa 16.8 julkaistiin uusi ominaisuus, hook-kutsut, minkä ansiosta tilan hallinta funktionaalisissa komponenteissa on myös mahdollista. Funktionaalisissa komponenteissa voidaan käyttää tilaa useState()-hook-kutsun avulla. Kaavassa 3 on esitetty tilaobjektin luominen tällä tavalla (Kaava 3). Esimerkistä nähdään, että useState()-hook-kutsulla luotu tilaobjekti on taulukko, jossa on itse tila (”pointResourcesDialog”) ja tilan muuttamiseen tarkoitettu funktio ”setPointResourcesDialog”. ”pointResourcesDialog”-tilan arvo on tässä tapauksessa objekti, jonka ”isOpen”-kentän arvo on lähtökohtaisesti false.

Kaava 3. Tilaobjektin luominen funktionaalisessa komponentissa

```
const [pointResourcesDialog, setPointResourcesDialog] = useState({
      isOpen: false
    });
```

3.4 Hook-funktiot

Hook-funktiot ovat suhteellisen uusi ominaisuus Reactissa. Hook- funktiot ansiosta komponenttien kanssa työskentely on kehittäjälle tehokkaampaa. Hook-funktiot korvaavat sellaisia toiminnallisuuksia, jotka aikaisemmin oli mahdollista toteuttaa vain luokkakomponenteissa, kuten tilan hallinnan. Näin ollen hook- funktiot mahdollistavat React-sovellusten luomisen ilman luokkakomponentteja. (Schwarzmüller, 2020 g.)

Hook- funktiot ovat JavaScript funktioita, joita voidaan käyttää joko funktionaalisissa komponenteissa tai muissa hook-funktioissa. Hook-funktio nimetään seuraavasti: useHookFunktionNimi. Nimen alussa on pienellä kirjoitettu ”use”, jonka jälkeen tulee hook-funktion nimi, esimerkiksi ”useState()”. Hook-funktioiden tarkoitus on paljastaa toiminnallisuuksia funktionaalisille komponenteille. Ne toimivat itsenäisesti jokaisessa komponentissa ja ne ovat uudelleenkäytettäviä. Hook-funktiot mahdollistavat tilan lisäämisen funktionaalisiin komponentteihin sekä toiminnallisuuksien jakamisen komponenttien välillä. Hook-funktioita voidaan kirjoittaa myös itse. (Schwarzmüller, 2020 g.) Taulukossa on listattu yleisimmin käytettyjä hook-funktioita (Taulukko 5).

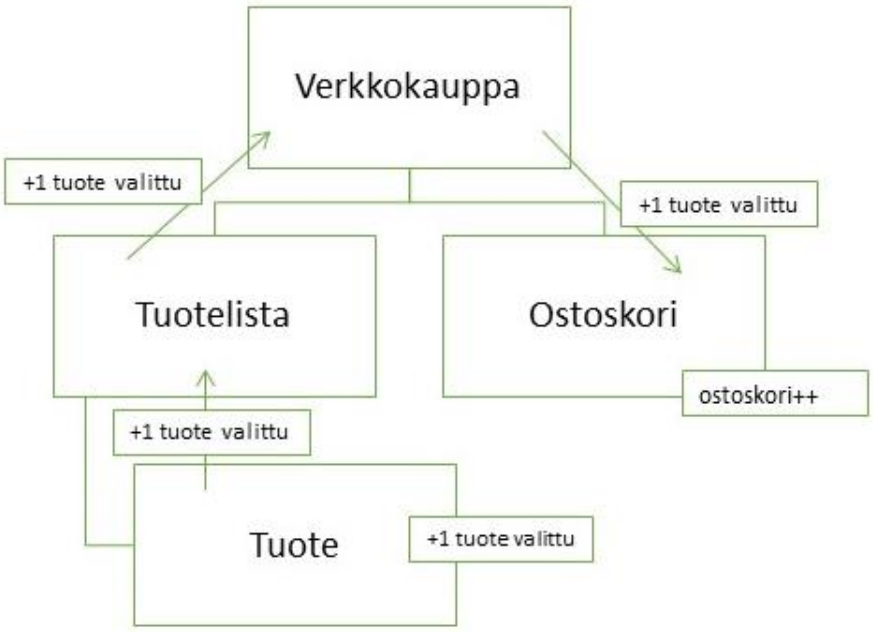
Taulukko 5. Yleisimmät hook-funktiot (React, 2020.)

Hook-funktio	Toiminto
useState()	Tilaobjektin luominen
useEffect()	”useEffect()”-funktion sisällä oleva koodi suoritetaan aina komponentin renderöimisen jälkeen. Tätä käytetään esimerkiksi tiedon hakemiseen sivun latautuessa.
useContext()	Context-objektin lisääminen komponenttiin muuttujana.

3.5 Context API

Context API on Reactin ominaisuus, jonka avulla voidaan välttää pitkiä props-ketjuja. Props-ketjuja syntyy silloin kun tietoa pitää välittää useiden komponenttien yli. Esimerkiksi yhdessä komponentissa voidaan käynnistää tapahtuma, jonka tulos näytetään toisessa komponentissa. Tällainen tilanne voisi olla esimerkiksi verkkokaupassa, kun tuote halutaan siirtää ostoskoriin. Jos käytetäisiin props-olioita tavallisella tavalla, kyseinen parametri pitäisi siirtää ensin siihen komponenttiin, jonka alla tuotekomponentti ja ostoskorikomponentti ovat ja sen kautta tieto lisäyksestä siirtyisi ostoskorikomponentille (Kuva 10).

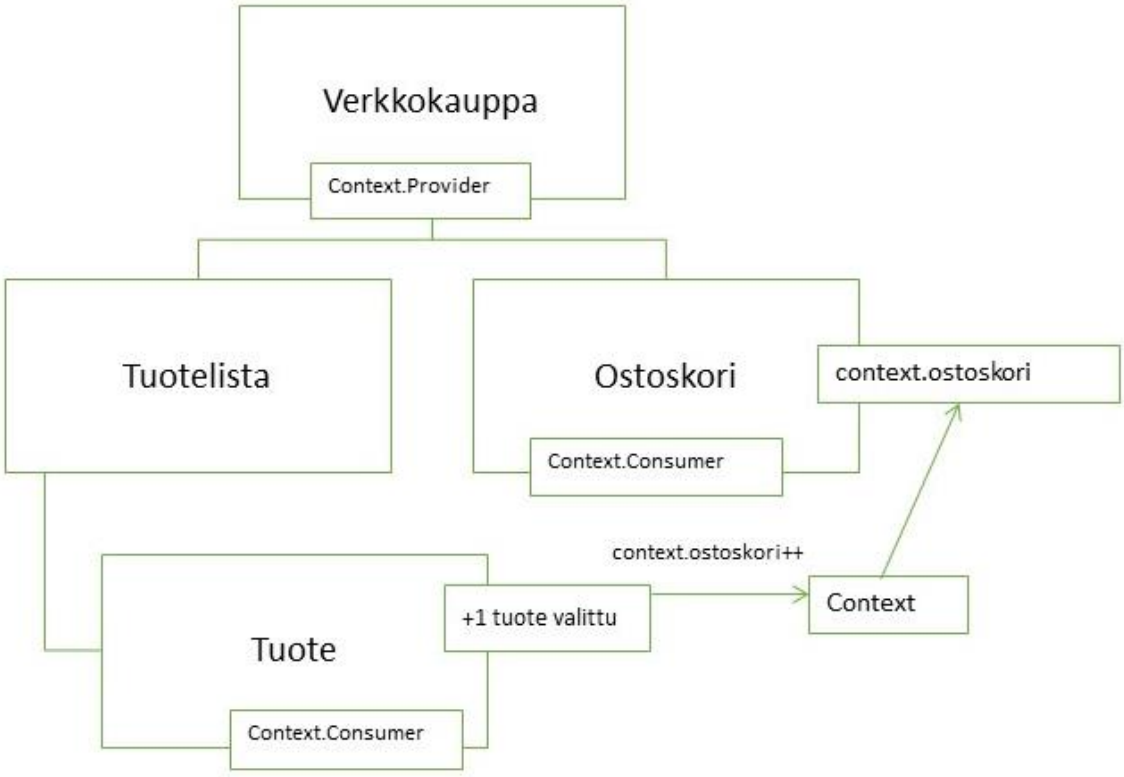
Kuva 10. Esimerkki: tuotteen lisääminen ostoskoriin props-ketjun kautta



Kuvan 10 esimerkissä tuotelista- ja verkkokauppakomponentit vain välittävät tietoa edelleen ostoskorikomponentille. Näiden komponenttien yli voidaan kuitenkin hypätä käyttämällä Context API:ta. Contextia voidaan käyttää missä tahansa sovelluksen komponentissa.

Luotua context-objektia käytetään siten että se komponentti, joka on hierarkiassa ylimpänä, toimii context-objektin tarjoajana (provider). Ne komponentit, joille halutaan antaa pääsy context-objektiin ovat objektin käyttäjiä (consumer). Kuvan 10 verkkokauppaesimerkissä verkkokauppakomponentti olisi context-objektin tarjoaja ja tuotekomponentti ja ostoskorikomponentti olisivat käyttäjiä. Tuotelista ei olisi tässä tapauksessa käyttäjä, koska se ei tarvitse context-objektin tarjoamaa tietoa tässä tapauksessa. Kuvassa 11 on esitetty verkkokauppaesimerkki siten että tiedonvälitykseen käytetään context-objektia (Kuva 11).

Kuva 11. Esimerkki: tuotteen lisääminen ostoskoriin context-objektin avulla



Context-objekti otetaan sovelluksessa käyttöön antamalla sille arvot tarjoajakomponentissa, jonka jälkeen palautettava JSX-koodi ympäröidään context.provider-tageilla. Context.provider-komponentille annetaan ”value”-parametrin arvoksi tarjoajakomponentissa asetettu context-objektin arvo seuraavasti:

```
<AppContext.Provider value={contextStateValue}>
```

Käyttäjäkomponentissa JSX-koodi ympäröidään context.consumer-tageilla. Tagien sisälle lisätään aaltosulut, koska JSX:n sisälle kirjoitetaan JavaScriptiä, ja tehdään funktiokutsu, jonka parametri on context-objekti, joka saadaan näin käyttöön. Kaavassa 4 on esitetty esimerkki context-objektin käyttäjäkomponentin palauttamasta JSX-koodista (Kaava 4).

Kaava 4. Context-objektin käyttäjäkomponentin palauttama JSX-koodi

```
return (
      <AppContext.Consumer>
        {(context) => <div>
          <Dialog
            open={context.pointResourcesDialogIsOpen}
            onClose={() => context.togglePointResourcesDialog(false)}
            aria-labelledby="form-dialog-title">
            <DialogTitle id="form-dialog-title"
                         className={classes.title}
             >{fi.tasks.pointTasks}
             </DialogTitle>
            <DialogContent>
            </DialogContent>
          </Dialog>
        </div>}
      </AppContext.Consumer>
    );
```

3.6 Työnohjauksen käyttöliittymän toteuttamisessa käytetyt kirjastot

Työnohjauksen käyttöliittymän toteutuksessa hyödynnetään monia React- ja JavaScript-kirjastoja. Tässä kappaleessa käsitellään niitä kirjastoja, jotka ovat sovelluksen kannalta perustavanlaatuisia. Näiden lisäksi sovelluksessa käytetään muita kirjastoja pienemmässä mittakaavassa, esimerkiksi päivämäärien formatointiin.

Kirjastojen käyttämisen tarkoituksena on nopeuttaa käyttöliittymäkehitystä siten että yleistä ongelmaa ei yritetä ratkaista itse, vaan hyödynnetään jo olemassa olevaa ratkaisua samaan ongelmaan. Tarkoitus on käyttää hyvin dokumentoituja ja aktiivisia kirjastoja.

3.6.1 Material-UI

Material-UI on komponenttikirjasto, jonka tarkoitus on tehdä web-sovelluskehityksestä Reactilla entistä nopeampaa. Se tarjoaa monipuolisen valikoiman erilaisia komponentteja yleisimpiin käyttötarkoituksiin. Komponenttien lisäksi Material-UI:n kautta voi hankkia valmiin käyttöliittymäpohjan edullisesti. Material-UI tarjoaa komponentteja muun muassa layoutin kokoamiseen, navigoinnin toteuttamiseen, käyttäjän syötteen vastaanottamiseen ja erilaisten prosessien etenemisen näyttämiseen. (Material-UI, 2020.)

Material-UI on kattavasti dokumentoitu avoimen lähdekoodin kirjasto, jota käyttävät muun muassa Spotify, Netflix, Unity ja Amazon (Material-UI, 2020). Sillä on lähes 390 000 käyttäjää ja viimeisin versio on julkaistu 3.12.2020 (Github a, 2020).

3.6.2 MUI-Datatables

MUI-DataTables on Material-UI:n pohjalta luotu kirjasto dynaamisten taulukoiden luomiseen. Se tarjoaa monipuoliset suodatus-, järjestämis- ja hakutoiminnot sisäänrakennettuna. Näiden toimintojen muokkaaminen omiin tarpeisiin sopivaksi on myös tehty suhteellisen helpoksi. Kirjasto on dokumentoitu hyvin ja se päivittyy aktiivisesti, siitä julkaistiin versio 3 kesällä 2020 (Github b, 2020).

3.6.3 Axios

Axios on Promise-pohjainen JavaScript-kirjasto Ajax-kutsujen lähettämiseen palvelimelle. Axios tekee asynkroonisten HTTP-kutsujen lähettämisestä REST-päätepisteille ja CRUD-operaatioiden tekemisestä helppoa. Kun kutsu lähetetään palvelimelle, se palauttaa vastauksen. Axiosin vastausobjekti sisältää seuraavat parametrit:

- Data: serveriltä palautuva haettu tieto
- Status: serveriltä palautuva HTTP-koodi

- StatusText: serveriltä palautuva HTTP-statusviesti
- Headers: serverin lähettämät headerit
- Config: Alkuperäisen kutsun konfiguraatio
- Request: Kutsuobjekti
  (ZetCode, 2020.)

Ajax-kutsut voitaisiin kirjoittaa myös käyttämällä JavaScriptin XML HTTP request -objektia, mutta Axiosin avulla kutsujen kirjoittaminen on helpompaa ja koodi luettavampaa.

3.6.4 MobX

MobX on yksinkertainen, skaalautuva ja hyvin testattu sovelluksen tilanhallintaan tarkoitettu kirjasto, jota käytetään useimmiten Reactin yhteydessä. Se tekee tilanhallinnasta yksinkertaista siten että se tekee epäjohdonmukaisen tilan luomisesta mahdotonta. MobX:n ydinajatus on, että kaikki tieto, joka voidaan johtaa sovelluksen tilasta, johdetaan siitä. (MobX, 2020.)

Työnohjaussovelluksen globaali tila toteutetaan MobX:llä. Tämä globaali tila toimii sovelluksen sisäisenä tietovarastona ja se on arkkitehtuurillisesti rajapintakutsujen ja komponenttien välissä. Komponentit saavat tarvitsemansa datan joko suoraan globaalista tilasta tai johdettua siitä.

4 Työnohjauksen käyttäjät ja käyttötapaukset

Työnohjaussovellusta käyttävät sellaiset Geometrixin asiakasyrityksissä toimivat henkilöt, jotka hallinnoivat ja koordinoivat työtehtäviä ja työntekijöitä, esimerkiksi työnjohtajat. Kun sovellusta kehitetään tietylle käyttäjäryhmälle, on tärkeää ymmärtää käyttäjien tarpeita ja toimintatapoja. Työnohjaussovelluksen käyttäjien näkökulman ymmärtämistä varten haastateltiin kahden Geometrixin asiakasyrityksen edustajia. Haastateltavia henkilöitä oli kaksi ja he työskentelevät työnjohtajina. Haastattelujen tarkoitus oli lisätä kehitystiimin ymmärrystä työnjohtajien työstä, kartoittaa nykyisen Työnohjauksen käyttöä sekä havaita kehityskohteita uuteen sovellukseen.

4.1 Käyttäjähaastattelut

Yksi tehokkaimmista tavoista selvittää, mitä ihmiset haluavat ja mitä ongelmia heillä on, on keskustella heidän kanssaan. Haastattelut eri sidosryhmien kanssa on tärkeä tapa kerätä tietoa. Haastattelutavat voivat vaihdella jäsennellyn haastattelun ja yleisen keskustelun välillä. Jäsennelty haastattelu toteutetaan siten että haastattelukysymykset suunnitellaan etukäteen ja haastattelussa noudatetaan suunniteltua linjaa täysin. (Benyon, 2019, s. 151.)

Usein haastattelut toteutetaan osittain jäsennellysti. Tällöin haastattelijalla on etukäteen valmisteltuja kysymyksiä käytössä, mutta muita aiheita voidaan käsitellä, kun niitä tulee haastattelun myötä esille. (Benyon, 2019, s. 151.)

Työnohjaussovelluksen käyttäjähaastatteluissa käytettiin osittain jäsenneltyä linjaa. Etukäteen oli valmisteltu muutamia haastattelukysymyksiä, mutta tavoitteena oli pitää haastattelutilanne keskustelevana.

Käyttäjähaastattelut toteutettiin kummallekin haastateltavalle omana Teams-palaverina. Haastattelujen tavoitteena oli saada vastaukset valmisteltuihin kysymyksiin ja samalla pitää keskustelu mahdollisimman avoimena.

Haastateltavilta kysyttiin seuraavat kysymykset:

- Minkälainen on tyypillinen työpäiväsi?
- Miten paljon sinulla on hallittavia tehtäviä ja työntekijöitä?
- Mikä työnohjauksen tarjoama tieto on sinulle tärkeintä?
- Missä työnohjauksen toimintojen käyttämisessä tapahtuu virheitä ja miten virheet on saatu ratkaistua?
- Jos saisit muuttaa kolmea asiaa työnohjauksessa, mitkä ne olisivat ja miten muuttaisit niitä?
- Minkälaisia työkaluja työnohjaukseen on aikaisemmin käytetty?

Haastattelukysymyksissä käsiteltyjen aiheiden lisäksi molemmissa keskusteluissa käsiteltiin haastateltaville mieleen tulleita kehitysehdotuksia. Asiakasyrityksiin viitataan termeillä ”Asiakasyritys 1” ja ”Asiakasyritys 2”. Asiakasyritys 1 toimii raideliikenteen tekniikkaosa-alueiden palveluntarjoajana ja Asiakasyritys 2 on kuntasektorilla toimiva rakentamispalveluja tuottava toimija.

4.1.1 Työnjohtajan työpäivä

Asiakasyrityksessä 1 haastatellun työnjohtajan työpäivä koostuu työnsuunnittelusta sekä palavereista ja sähköposteihin vastaamisesta. Tavoitteena olisi, että hän pääsisi käymään vähintään kerran viikossa työmailla. Työnsuunnittelussa on tarkoitus tehdä työtehtävien toteuttamisen suunnittelua mahdollisimman pitkällä aikajänteellä, noin kuukauden päähän suunnitteluhetkestä. Suunnitelmat saattavat muuttua, jos eteen tulee kiireellisiä töitä esimerkiksi seuraavalle päivälle.

Asiakasyrityksessä 2 työpäivät vaihtelevat laidasta laitaan. Työnjohtajan työtehtävät sisältävät tarjouslaskentaa, työmaakäyntejä, työn opastusta, materiaalien hankintaa ja palavereja. Ennakkoon on yleensä tiedossa seuraavan päivän työt tai osa saman viikon töistä. Työnjohtaja voi vaikuttaa itse paljon työpäivän ja työviikon rakenteeseen, esimerkiksi siten että toimistopäivät voi määritellä itse. Liikkuminen on Asiakasyrityksen 2 haastatellulle työnjohtajalle tärkeä osa työpäivää ja työmailla käydään paljon. Näin ollen työnohjaussovelluksen mobiilikäyttö koetaan tärkeänä ominaisuutena.

Vastauksista havaittiin, että vaikka asiakkaat toimivat samankaltaisilla toimialoilla, työpäivien dynamiikka on työnjohtajilla hyvin erilainen. Merkittävin ero huomattiin suunnittelussa. Toisessa yrityksessä pyritään tekemään suunnitelmat mahdollisimman pitkällä aikavälillä, kun taas toisessa tehtävät vaihtuvat nopeammalla syklillä. Myös mobiilikäytön havaittiin olevan tarpeellinen ominaisuus vain toiselle haastatelluista työnjohtajista. Työnohjaussovelluksen käyttöliittymän uusimisessa pitää ottaa huomioon, että vaikka sovelluksen käyttäjät työskentelevät samankaltaisissa tehtävissä, heidän työnsä on silti keskenään hyvin erilaista.

4.1.2 Työnohjaussovelluksen käyttöön käytetty aika

Asiakasyrityksen 1 haastateltu työnjohtaja käyttää työnohjaussovellusta tällä hetkellä noin tunnin viikossa. Tarkoitus olisi, että työnohjaus olisi päivittäisessä käytössä, mutta sovelluksen käyttöönotto on vielä alkuvaiheessa. Asiakasyrityksen 2 työnjohtaja käyttää työnohjausta puolesta tunnista tuntiin päivässä. Työtehtävien siirtoihin ja vaiheistukseen

kuluu hieman yli puoli tuntia aikaa ja työnohjauksen tuntiraporttia hyödynnetään tehtävien laskutuksessa.

4.1.3 Tärkein tieto työnohjauksessa

Asiakasyrityksen 1 työnjohtajalle tärkeintä työnohjaussovelluksen tarjoamaa tietoa on ajoitetut huollot, tehdyt työt, laitteiden sijainnit ja laitetiedot (esimerkiksi huolto-ohjeet). Asiakasyrityksessä 2 tärkeintä on ketterä kokonaisnäkymä ilman turhia klikkauksia. Tärkeimpinä esiin nousivat arviohinta, tehdyt työtunnit, arvioidut työtunnit ja kokonaiskuva. Asiakasyrityksen 2 keskeiseksi tarpeeksi työnohjaussovellukselle tunnistettiin se, että saatavilla olisi mahdollisimman paljon tietoa mahdollisimman helposti ja nopeasti saatavilla. Tämä johtuu pääasiassa siitä, että työnohjaussovelluksesta haetaan tietoja muiden työtehtävien kuten laskutuksen ja tarjouslaskennan tekemiseen, joita ei voi tehdä työnohjauksessa.

4.1.4 Työnohjauksen käyttövirheet

Asiakasyrityksessä 1 oli alkuvaiheessa epäselvää pitääkö työnjohtajan kuitata työt vielä erikseen tehdyksi sen jälkeen, kun asentaja on merkinnyt työn valmiiksi. Tällä hetkellä työnohjauksessa työnjohtajan pitää vielä käydä vaihtamassa työn tila valmiiksi. Tämä on havaittu Asiakasyrityksessä 1 työnjohtajia kuormittavana tekijänä, koska monesti asentajalla on parempi tietämys tehdystä työstä, kuin työnjohtajalla. Kehitysideana keskusteltiin mahdollisesta monitasoisesta tehtävien hyväksymisestä, jolloin työtehtävän tyypin perusteella asentajalla voisi olla valtuudet hyväksyä työtehtäviä.

Asiakasyrityksessä 2 ongelmana ovat olleet samannimisten paikkojen suuri määrä, jolloin tehtävien erottuminen on ollut vaikeaa. Nyt ongelma on ratkaistu siten että tehtävän kuvaus -kenttään lisätään manuaalisesti päivämäärä, jotta tehtävät erottuvat päivän mukaan toisistaan. Päivämäärän perusteella työntekijä hahmottaa, mikä tehtävä on uudempi ja mikä vanhempi.

Käyttövirheistä keskustelun tavoitteena oli hahmottaa käyttäjille haastavia tilanteita, joita voidaan parantaa tai kumota kehitysvaiheessa uuteen työnohjauksen käyttöliittymään.

Keskustelujen perusteella huomattiin, että tehtävien erottumista toisistaan pitää selkiyttää ja töiden hyväksyntää voisi kehittää monitasoiseksi.

4.1.5 Muut työtehtävien hallintaan käytettävät työkalut

Asiakasyrityksessä 1 muita työnohjaukseen käytettäviä työkaluja ovat Rautatieohjeet ja Excel. Joitain työnohjaukseen liittyviä asioita on myös muistin ja työkokemuksen varassa. Ohjeistuksissa on selkeät määrittelyt tietyille tarkastuksille muun muassa siitä, miten tietyt tarkastukset tehdään ja miten usein. Työtehtävien ja työntekijöiden ohjaamiseen ei ole ollut aikaisemmin käytössä IT-järjestelmää.

Asiakasyrityksessä 2 käytetään eri toimintoihin eri järjestelmiä ja ohjelmia. Työtehtävien ja työntekijöiden hallintaan käytetään Mobilenoten nykyistä työnohjausta, kulujen seurantaan käytetään SAP:ia, tarjouslaskentaan käytetään Exceliä, laskutukseen käytetään Exceliä ja SAP:ia ja dokumentinhallintaan on käytössä oma projektikansiotyyppinen ratkaisu. Asiakasyrityksen 2 työnjohtaja mainitsi, että mikäli kaikki nämä toiminnot saataisiin yhden järjestelmän alle, se säästäisi työnjohtajien työaikaa 5-8 tuntia viikossa eli noin 15-20 % viikoittaisesta työajasta.

4.1.6 Kehitysehdotukset

Molemmissa haastatteluissa keskusteltiin paljon kehitysehdotuksista. Asiakasyrityksen 1 haastattelussa esiin nousivat eritasoinen tehtävien kuittaus, työtehtävien massasyöttö, määräaikaishuoltojen aikalaskuri sekä tähän liittyvä muistutustoiminto, dokumenttien ja huolto-ohjeiden liittäminen työtehtävälle ja resurssien hallinta. Eritasoisella kuittauksella tarkoitetaan sitä, että kun asentaja merkitsee työn valmiiksi, esihenkilön ei tarvitse kuitata työtä erikseen järjestelmästä, paitsi silloin kun se nähdään tarpeelliseksi. Asentajalla olisi joissain tapauksissa valtuus kirjata tehtävä tehdyksi ja joissain tapauksissa tehtävä voi vaatia ylemmän tason hyväksynnän. Työtehtävien massasyötöllä tarkoitetaan sitä, että useamman työtehtävän voi osoittaa yhdelle tai useammalle työntekijälle samalla kerralla. Määräaikaishuoltojen aikalaskuri helpottaisi työnsuunnittelua, kun työtehtävien ennakointi olisi esitetty visuaalisessa muodossa. Tähän ehdotettiin kalenterinäkymää, mistä näkisi tulevat työtehtävät hyvissä ajoin. Ennakointiin liittyisi myös muistutustoiminto tietyn ajan

päästä tapahtuvasta työstä. Huolto-ohjeiden liittäminen työtehtävälle olisi myös tärkeä ominaisuus työnohjauksessa. Asiakasyrityksessä 1 on tarve laajemmalle resurssienhallinnalle ja mahdollisimman monen toiminnon saaminen saman järjestelmän alle nähtäisiin etuna.

Asiakasyrityksessä 2 esiin nousi, että nykyisessä työnohjauksessa on paljon klikkauksia siihen, että työn saa kuitatuksi, joten klikkausten vähentäminen tästä prosessista olisi tärkeää. Tehtävälistauksessa voisi olla vapaasanahaku, jolla taulukkoa saisi nopeasti suodatettua. Tehtävien massaosoitus, joka nähtiin asiakasyrityksessä 1 tarpeelliseksi, voisi olla hyvä lisäominaisuus, mutta sitä ei nähty asiakasyrityksessä 2 yhtä kriittiseksi. Työnohjauksen karttatoiminto toisi lisäarvoa, jos siihen saisi integroitua rakennusten tiedot karttatasona. Karttaan olisi hyvä saada kaikki osastot näkyviin, jotta kohteen tietoja voidaan tiedustella etukäteen, jos nähdään että siellä o jo toisen osaston tiimi paikalla. Työvaiheet ja työtunnit pitäisi saada paremmin näkyviin, koska se helpottaisi työnjohtajien muiden työtehtävien, kuten laskutuksen tekemistä. Nykyisessä työnohjauksessa työvaiheet ja työtunnit ovat kolmen klikkauksen takana. Tehtävälistaukseen olisi hyvä saada työvaiheet paremmin näkyviin, esimerkiksi riviltä aukeavalle alavalikkoon. Resurssinäkymässä voisi olla käytössä käyttäjähierarkia, josta näkyy, kuka käyttäjistä on työnjohtaja ja kuka työntekijä. Nykyisessä työnohjauksessa resurssit näkyvät resurssipuuna, jossa ovat osastot, ryhmät ja käyttäjät, mutta erityyppisillä käyttäjillä ei ole erottavaa tekijää. Tämä voitaisiin toteuttaa visuaalisella keinolla esimerkiksi erilaisella ikonilla.

Keskusteluissa havaittiin monia toteutuskelpoisia kehitysehdotuksia. Osa keskustelluista kehitysehdotuksista oli osattu ottaa jo huomioon käyttötapausten alustavassa suunnittelussa, mutta yksityiskohdat tarkentuivat haastattelujen myötä. Työnohjauksen tarkoitus on palvella sen käyttäjien tarpeita, joten käyttäjien konsultoiminen uuden käyttöliittymän kehitysvaiheessa on lopullisen tuotteen onnistumisen kannalta tärkeää. Geometrixin asiakkaat ovat itse omien toimintojensa asiantuntijoita, joten jos tuotekehityksen pohjalle halutaan saada mahdollisimman oikeaa tietoa nykyisen järjestelmän puutteista ja tarpeellisista uusista ominaisuuksista, niitä täytyy kysyä käyttäjiltä itseltään.

4.2 Käyttötapaukset

Käyttötapauksilla tarkoitetaan työnohjaussovelluksen ominaisuuksia. Jotta uuden sovelluksen kehittäminen olisi tehokasta, iso kokonaisuus kannattaa pilkkoa pienemmiksi osiksi. Tämä auttaa hahmottamaan kokonaisuutta ja töiden edistymistä on helpompi seurata. Työnohjaussovellus jaettiin käyttöliittymän uusimisprojektin alkuvaiheessa useampaan käyttötapaukseen ja näitä tunnistettuja käyttötapauksia täydennettiin käyttäjähaastatteluista tunnistetuilla kehitystarpeilla. Käyttötapaukset sisältävät nykyisen työnohjauksen ominaisuudet sekä uudet suunnitellut ominaisuudet.

Työnohjauksen uuteen web-käyttöliittymään toteutettavat ominaisuudet ovat:

- Kirjautuminen
- Tehtävien hakeminen listaan
- Tehtävien suodattaminen listassa
- Resurssilistan näyttäminen
- Tehtävän osoittaminen resurssille
- Tehtävän kommentointi
- Tehtävän tietojen katselu
- Tehtävän luominen
- Tehtävän muokkaaminen
- Roolin karttatasot karttakomponentissa
- Tehtävän tietojen massapäivitys
- Tehtävän osoittaminen usealle resurssille samalla kerralla
- Tehtävän liitedokumentin käsittelytoiminnot
- Resurssinäkymä
- Ryhmän luominen
- Ryhmän tietojen muokkaaminen
- Käyttäjän liittäminen ja irrottaminen ryhmästä
- Resurssin tehtävätilanteen katselu
- Tehtävätilanne Gantt-kaaviona
- Tehtävien näyttäminen kartalla
- Tehtävän tiedot puhekuplassa kartalla
- Tehtävän siirtäminen resurssilta toiselle
- Resurssien sijainnin katselu kartalla
- Tehtävien niputtaminen
- Tehtävien ennakointi

(Geometrix Oy: Työnohjausprojektin roadmapin suunnittelu, henkilökohtainen tiedonanto, 12.11.2020.)

5 Työnohjauksen käyttöliittymäprototyypin toteuttaminen

Työnohjaussovelluksen käyttöliittymän uusiminen on iso projekti, jossa toteutetaan monta ominaisuutta. Tässä opinnäytetyössä keskitytään käyttöliittymäprojektin alkuvaiheen toteutuksiin. Projektissa on tarkoitus toteuttaa ensin nykyisen työnohjauksen ominaisuudet, kuten tehtävien hakeminen ja osoittaminen ja sen jälkeen uudet ominaisuudet, kuten ryhmän luominen ja ryhmän tietojen muokkaaminen. Tässä kappaleessa käsitellään, minkälainen uuden React-sovelluksen arkkitehtuuri on ja miten prototyyppivaiheen ominaisuudet toteutettiin.

5.1 Prototyyppivaiheen ominaisuudet

Käyttöliittymän prototyyppivaiheessa toteutettavat ominaisuudet ovat Tehtävien hakeminen listaan, Resurssilistan näyttäminen ja Tehtävän osoittaminen resurssille. Ominaisuudet toteutettiin niiltä osin, että osoitustoimintoa pystyttiin demonstroimaan asiakkaalle joulukuussa 2020. Osoitustoiminnon toteuttamista varten ensiksi piti toteuttaa tehtävien hakeminen listalle ja resurssien hakeminen listalle, koska molempien tietojen täytyi olla olemassa ennen kuin tietoa näiden välillä voidaan siirtää.

5.2 Työnohjaussovelluksen web-käyttöliittymän arkkitehtuuri

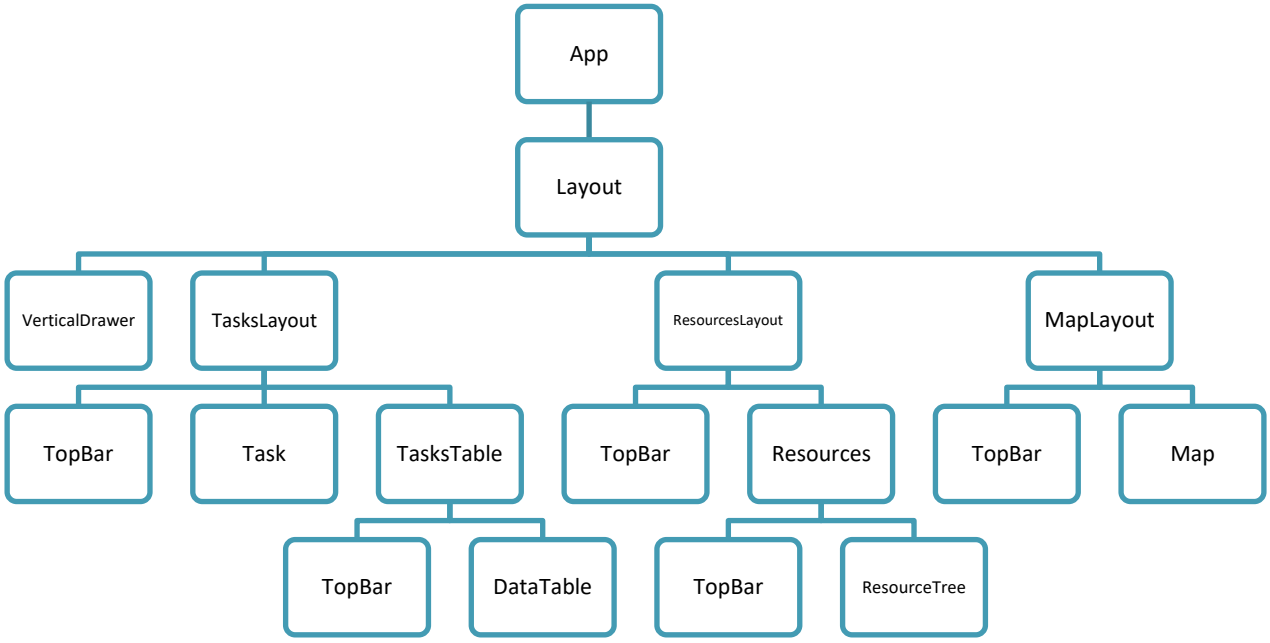
Työnohjaussovelluksen käyttöliittymän toteuttamisessa pyritään välttämään kovakoodattujen arvojen käyttämistä. Esimerkiksi string-tietotyypin arvot on koottu omaan JSON-tiedostoon. Pitkiä funktioita ja toistoa pyritään myös välttämään ja funktiot sekä muuttujat nimetään kuvaavasti. Tänä tekee koodista luettavampaa ja helpommin ylläpidettävää. Pitkiä funktioita voidaan välttää tekemällä jokaisesta koodin suorittamasta toiminnosta oma funktionsa. Näin koodista saadaan myös uudelleenkäytettävämpää. Koska kyseessä on React-sovellus, komponentit pyritään myös pitämään maltillisen kokoisina.

Työnohjaussovelluksen arkkitehtuuri voidaan jakaa kahteen osaan: komponenttihierarkiaan ja tiedonvälitysarkkitehtuuriin. Komponenttihierarkian tarkoitus on kuvata sovelluksen eri komponenttien välisiä suhteita. Tiedonvälitysarkkitehtuuri kuvaa sitä, miten tieto liikkuu sovellukseen, sovelluksen sisällä ja sovelluksesta ulospäin.

5.2.1 Komponenttihierarkia

Komponenttihierarkian ylimmällä tasolla on App-komponentti, joka renderöidään React-sovelluksen juuren index.js-tiedostossa. Se on sovelluksen juurikomponentti. Sovelluksen kehityksessä tavoitteena on pitää ”App”-komponentti mahdollisimman kevyenä, joten se sisältää ainoastaan ”Layout”-komponentin. ”Layout”-komponentti sisältää kolmen päänäkymän layout komponentit: ”TasksLayout”-komponentin, ”ResourcesLayout”-komponentin ja ”MapLayout”-komponentin sekä ”VerticalDrawer”-komponentin, joka on vasemman laidan navigointipalkki. Kaikki päänäkymäkomponentit sisältävät sisältökomponentin sekä ”TopBar”-komponentin. Sisältökomponentti, kuten Resources, sisältää kaikki siihen näkymään sisältyvät komponentit sekä ”TopBar”-komponentin. (Kuva 12.)

Kuva 12. Komponenttihierarkia



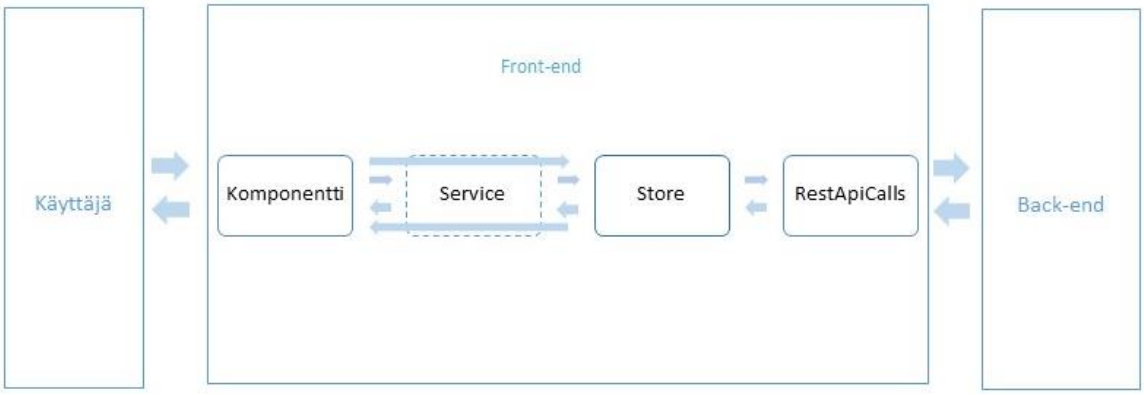
”TopBar”-komponentti on otsikkokomponentti, joka renderöidään sille annettujen parametrien mukaisesti. ”TopBar”-komponentin toteutuksessa konkretisoituu koodin uudelleenkäytettävyys, kun samaa komponenttia käytetään monessa sovelluksen osassa.

Komponenttihierarkiassa on esitetty tähän mennessä toteutetut komponentit. Hierarkian alimmalle tasolle lisätään uusia komponentteja sitä mukaan, kun työnohjauksen käyttöliittymäprojekti edistyy.

5.2.2 Tiedonvälitysarkkitehtuuri

Tiedonvälitysarkkitehtuurissa kuvataan, miten tieto liikkuu työnohjauussovelluksen web-käyttöliittymässä (Kuva 13).

Kuva 13. Tiedonvälitysarkkitehtuuri



Komponentit ovat selaimen kautta käyttäjälle näkyvä osa. Komponenttien kautta käyttäjä on vuorovaikutuksessa sovelluksen kanssa. Komponentti on yhteydessä ”Store”-luokkaan joko suoraan tai ”Service”-luokan kautta.

”Service”-luokat sisältävät tiedon käsittelyyn liittyvät funktiot, esimerkiksi tehtävän osoittamiseen tarvittavat funktiot sisällytetään ”PointingService”-luokkaan. Toiminnallisuudet voidaan kirjoittaa myös komponentteihin suoraan, mutta jako ”Service”-luokkiin tekee komponenteista kevyempiä ja helpommin luettavia.

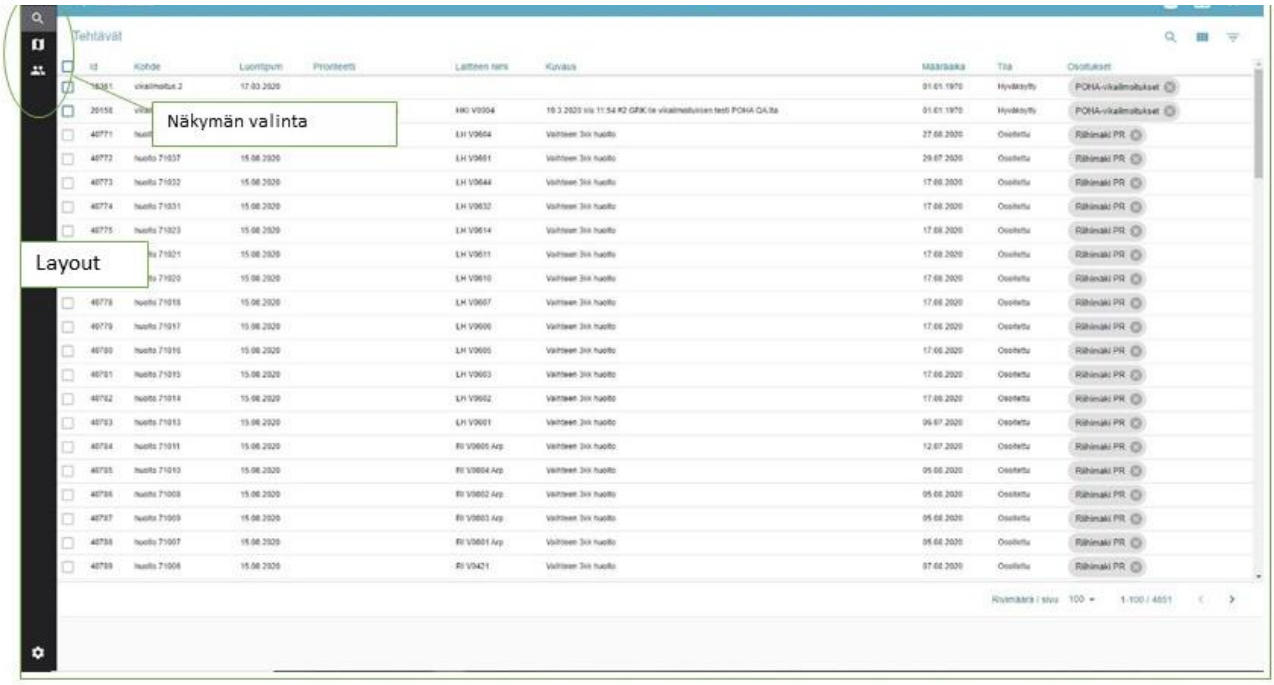
”Store” toimii sovelluksen tietovarastona ja tilana, mistä komponentit voivat käyttää tarvitsemaansa tietoa. ”Store” toimii myös siltana komponenttien ja ”RestApiCalls”-luokan REST-kutsujen välillä. Komponentista kutsutaan ”Store”-luokan funktiota, joka palauttaa tai välittää komponentin tarvitsemaa tai lähettämää tietoa. Tässä ”Store”-luokan funktiossa

kutsutaan ”RestApiCalls”-luokan asynkronista funktiota, joka tekee varsinaisen rajapintakutsun palvelinpuolen tarjoamalle rajapinnalle.

5.3 Työnohjauksen layoutin rakenne

Työnohjaussovelluksen uusi käyttöliittymä jakautuu kolmeen päänäkymään: tehtävänäkymään, resurssinäkymään ja karttanäkymään. Layout on sovelluksen pohjakomponentti ja siihen sisältyvät päänäkymät sekä navigointi päänäkymien välillä. Layout on ainoa Reactin ”App”-komponenttiin sisältyvä komponentti, joten se sisältää päänäkymien kautta kaikki sovelluksen komponentit. Tämän takia Layout-komponenttia käytetään sovelluksessa näkymien tilan hallintaan. Käyttäjä valitsee sovelluksen sivupalkista, mikä näkymä on milloinkin auki. Oletusnäkymä on tehtävänäkymä. ”Layout”-komponentti näyttää sivupalkin valinnan perusteella joko tehtävä-, kartta- tai resurssinäkymän. (Kuva 14.)

Kuva 14. Työnohjauksen käyttöliittymän layoutin rakenne



5.4 Tehtävälista

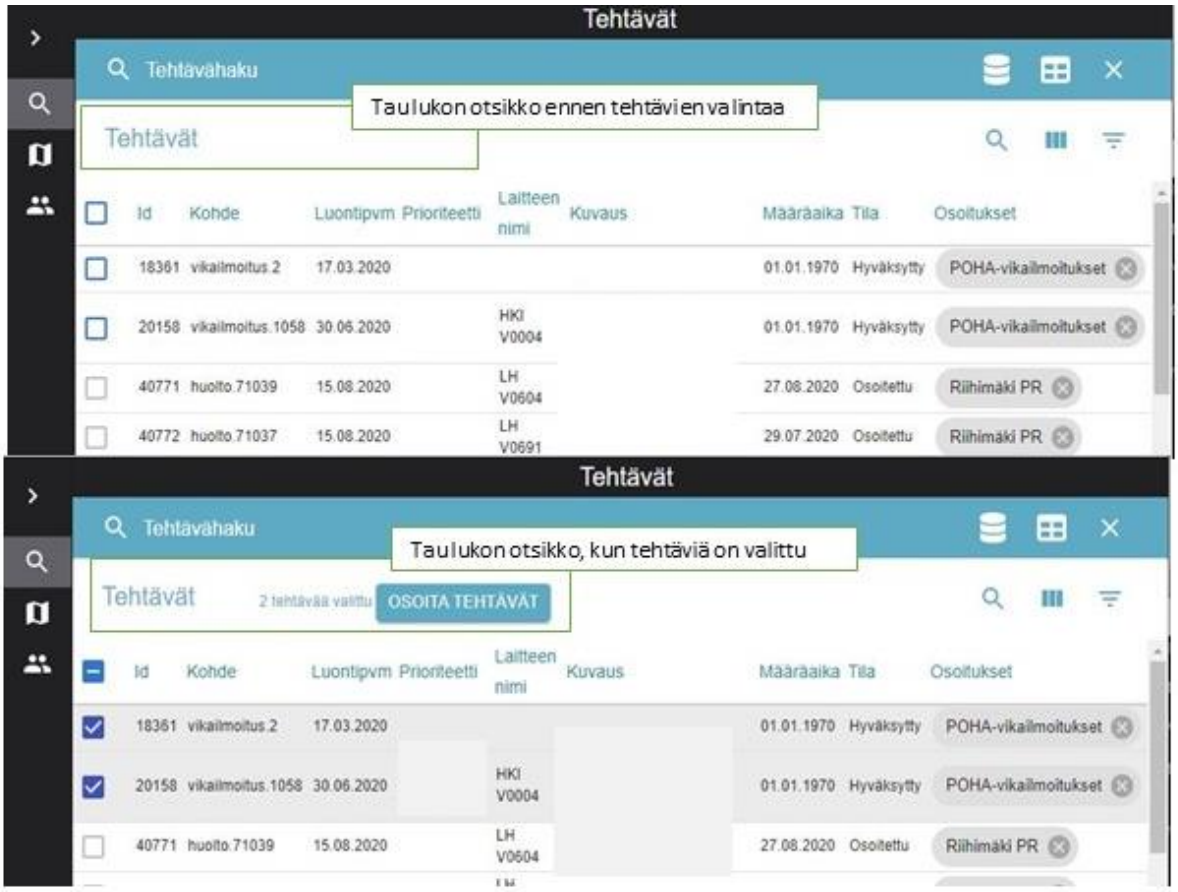
Työtehtävien näyttäminen listalla on työnohjauksen tärkeimpiä tehtäviä. Tehtävälista on se osa sovelluksesta mistä kaikki työnohjauksen ydintoiminnot lähtevät liikkeelle. Uudessa käyttöliittymässä tehtävälistan toteutukseen käytetään MUI-DataTables-kirjastoa, joka tarjoaa sellaisia ominaisuuksia, joita on tullut esiin kehitystarpeissa. Yksi näistä on dynaaminen vapaasanahaku. Kirjaston ”MUIDataTable”-komponentti saa props-olion kautta seuraavat parametrit: otsikko (title), taulukon sisältö (data), kolumnit (columns) ja asetukset (options). Nämä tiedot määritellään ”MUIDataTable”-komponentin palauttavassa komponentissa.

5.4.1 Otsikko

”MUIDataTable”-komponentin otsikkoparametrina palautetaan itse kehitetty ”DataTableTitle”-komponentti. Otsikkona voitaisiin palauttaa myös tekstiä, mutta taulukon otsikkoa on tarkoitus hyödyntää tehtävien osoittamisessa, joten sitä tarvitaan monipuolisempaan käyttöön.

”DataTableTitle”-komponentti tutkii, onko taulukon rivejä valittu. Jos tehtäviä ei ole valittu näytetään ainoastaan otsikkoteksti. Jos tehtäviä on valittu, näytetään valittujen tehtävien lukumäärä ja ”Osoita tehtävät”-painike, jonka kautta tehtävien osoittaminen resursseille aukeaa. Taulukon otsikko muuttuu, kun tehtävärivejä valitaan valintaruuduista (Kuva 15).

Kuva 15. Taulukon otsikon dynaamiikka rivivalinnan mukaan



5.4.2 Sisältö

Taulukon sisältö, eli listalla näytettävät työtehtävät, haetaan työnohjauksen palvelinpuolelta REST-rajapinnan kautta. Rajapintakutsun muodostamiseen käytetään Axios-kirjastoa. Tehtävien haku käynnistyy, kun käyttäjä avaa sovelluksen ja tehtävälistakomponentti renderöidään. Komponentti tarkistaa ”useEffect()”-hook-funktiossa onko ”Store”-luokan ”tasks”-taulukko tyhjä vai ei eli onko tehtäviä jo haettu. Mikäli tehtäviä ei ole, käynnistyy tehtävien haku rajapinnasta ”Store”-luokan kautta. Kun tehtävät palautuvat palvelinpuolelta, ne lisätään ”Store”-luokan ”tasks”-taulukkoon, minkä kautta ne ovat tehtävälistakomponentin käytettävissä.

Taulukon sisältö palautetaan suoraan ”MUIDataTable”-komponentin sisältöparametrina ja mahdolliset muotoilut suoritetaan joko yksittäisen sarakkeen asetusten kautta tai koko taulukon asetusten kautta.

5.4.3 Sarakkeet

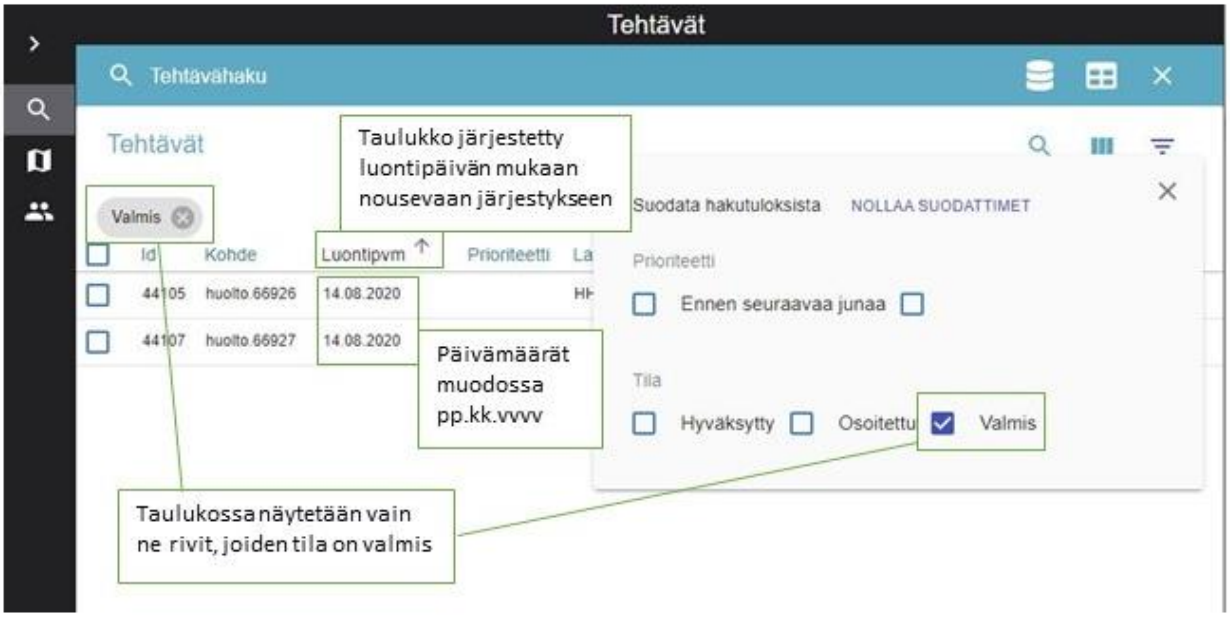
Seuraava parametri, jonka ”MUIDataTable”-komponentti saa on sarakkeet (columns). Se on tietotyypiltään taulukko, joka sisältää kaikkien taulukossa näytettävien sarakeobjektien tiedot. Sarakeobjekti saa parametrina nimen (name), tunnisteen (label) ja asetukset (options). Nimi on string-tietotyypin arvo, joka osoittaa siihen taulukon tietoaineiston ominaisuuteen, joka kyseisessä sarakkeessa halutaan näyttää, esimerkiksi ”createdDate”. Tässä sarakkeessa näytetään sisältötaulukon objektien ”createdDate”-kentän arvo. Tunniste on myös string-tietotyypin arvo ja se määrittelee, mitä taulukon sarakeotsikossa lukee. ”CreatedDate”-sarakkeen otsikoksi on määritelty ”Luontipvm”.

Sarakkeen asetukset -parametri on tietotyypiltään objekti. Asetukset-objektilla on monta eri vaihtoehtoa, miten sarakkeen tietoja voidaan muokata. Tässä vaiheessa Työnohjauksen tehtävälistan sarakkeissa yleisimpiä ovat ”sort”- ja ”filter”-toiminnot, jotka saavat boolean-arvon. ”Sort” määrittelee, voiko taulukon järjestää kolumnin arvojen mukaan ja ”filter” määrittelee voiko taulukkoa suodattaa sarakkeen arvojen mukaan.

Sarakkeen asetuksissa on myös parametreja, jotka vastaanottavat funktion. Esimerkiksi ”luontipvm”-sarekkeen sisältö on kustomoitava, koska arvo, joka palautuu rajapinnasta, on epoch-timestamp-muodossa. Sarakkeen sisältöä voidaan muokata ”customBodyRenderer”-parametrin kautta ja sille voidaan määrittää oma funktio. Tässä tapauksessa funktio formatoi sarakkeen päivämäärät muotoon pp.kk.vvvv ja palauttaa sen arvot tässä muodossa taulukkoon.

Sarakeasetukset objekti tarjoaa myös monia muita keinoja muokata taulukkoa sarakkeittain muun muassa suodattimien ja taulukon järjestämisen kustomointiin (Kuva 16).

Kuva 16. Sarakeasetukset käyttöliittymässä



5.4.4 Asetukset

Viimeinen ”MUIDataTable”-komponentin saama parametri on taulukon asetukset (options). Taulukon asetukset toimivat samalla logiikalla kuin saraketason asetukset eli parametri on tietotyypiltään objekti.

Taulukkotason asetuksissa määritellään esimerkiksi taulukon korkeus, mitä tekstikentissä kuten ”tooltip”-teksteissä ja alareunan sivutuksessa lukee ja miten paljon tuloksia näytetään sivua kohden.

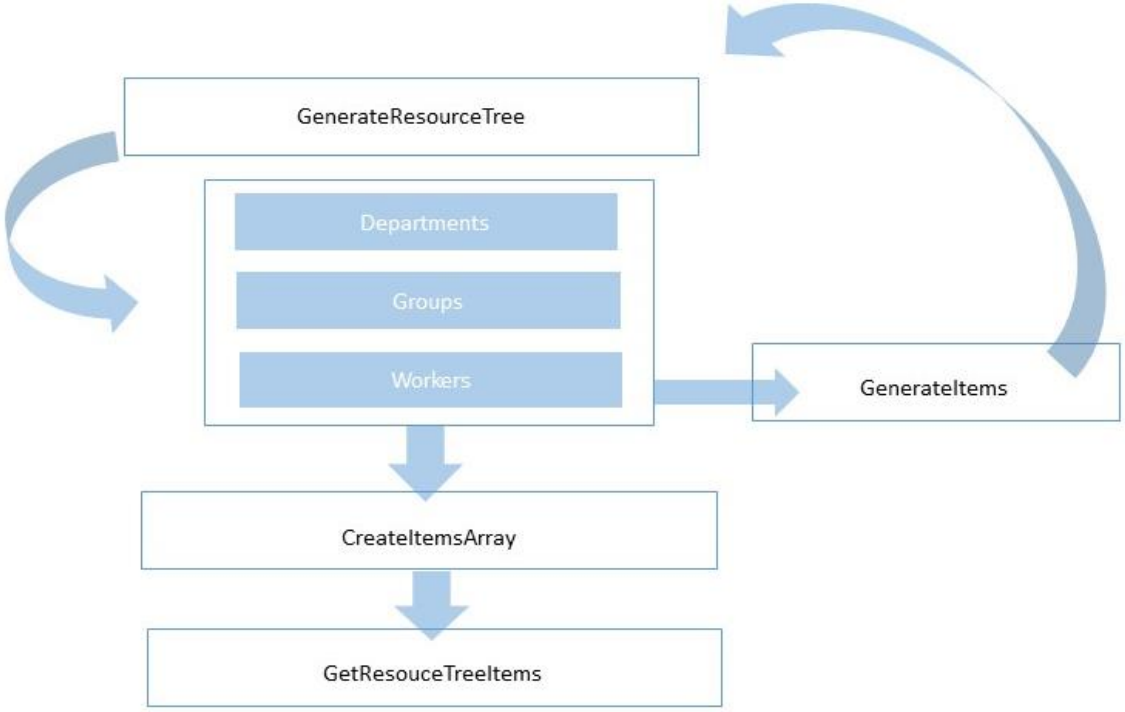
Taulukon asetukset tarjoavat myös funktioita eritarkoituksiin. Asetuksissa määritellään mitä tapahtuu, kun riviä klikataan tai mitä tapahtuu, kun rivejä valitaan valintaruuduista. Esimerkiksi rivien valintaan tarkoitettu ”onRowsSwlectionChange”-parametrille annetulla funktiolla lisätään valitut rivit ”Store”-luokan ”selectedTasks”-taulukkoon, jota käytetään tehtävien osoittamiseen resurssille ja jonka mukaan aikaisemmin mainittu taulukon otsikkokenttä päivittyy.

5.5 Resurssit

Tehtävälistauksen lisäksi Työnohjauksen kannalta tärkeä ominaisuus on resurssilistaus. Resurssitieto on rakenteeltaan puumainen, joten tiedon käsittely ei ole yhtä suoraviivaista verrattuna tehtävien listaukseen taulukkomuodossa. Resurssit näytetään resurssinäkymässä puurakenteena ja se on toteutettu Material-UI-kirjaston Tree-komponentilla.

Resurssitiedon haku rajapinnan kautta toimii samalla tavalla kuin tehtävälistan tietojen haku. Resurssitiedossa palautuu yksi pääosasto, johon kaikki muut resurssit sisältyvät. Resurssi voi olla osasto, ryhmä tai työntekijä. Osasto voi sisältää aliosastoja, ryhmiä ja työntekijöitä. Ryhmä sisältää työntekijöitä. Tiedon käsittely ja sen näyttäminen käyttöliittymässä toimii kuvan 17 prosessin mukaisesti (Kuva 17).

Kuva 17. Resurssipuun generointi



Resurssipuun generointi alkaa “generateResourceTree”-funktion kutsumisesta. Funktiossa tarkistetaan ensin, onko parametrina saatu resursseja. Funktiossa luodaan omat muuttujat kaikille resurssityypeille: ”departments”, ”groups” ja ”workers”.

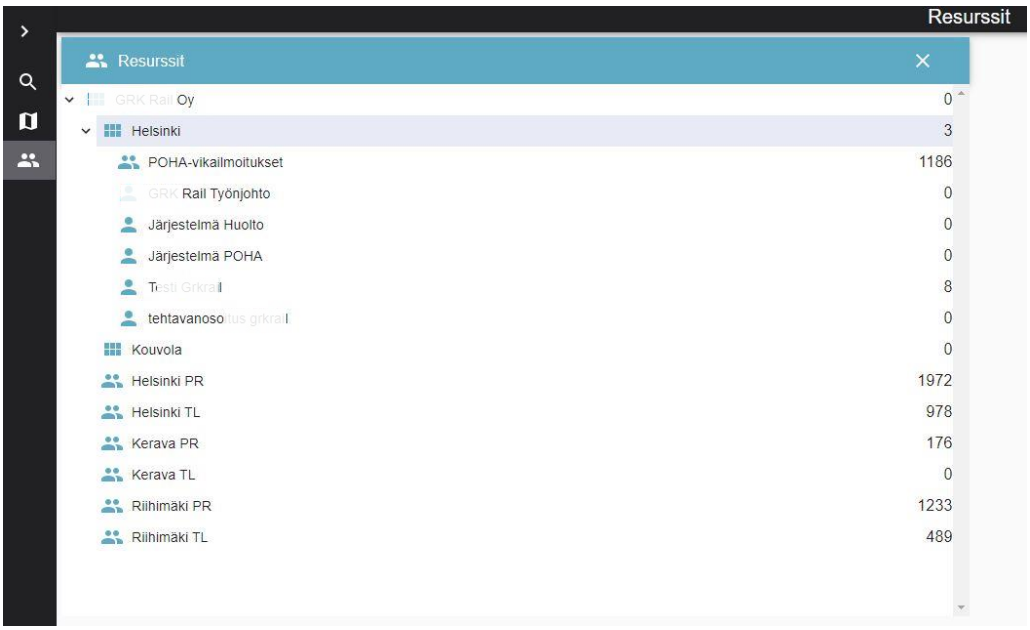
Tämän jälkeen kutsutaan ”generateItems”-funktiota kaikille pääresurssin aliresurssityypeille. Mikäli ”generateItems”-funktiolle välitetyllä aliresurssilla on aliresursseja, kutsutaan uudelleen ”generateResourceTree”-funktiota. Näin jatketaan kaikilla resurssityypeillä, kunnes aliresursseja ei enää ole.

Kun resurssilla ei ole aliresursseja, resurssi on kyseisellä kierroksella palautettava resurssi. Palautettavia resursseja voi olla yksi tai useampia. Kierroksen palautettavista resursseista muodostetaan oma taulukko. (departmentChildren). Kun ”departmentChildren”-taulukko on saatu, kutsutaan ”getResourceTreeItems”-funktiota, joka palauttaa sillä kierroksella palautettavan varsinaisen ”TreeItem”-komponentin.

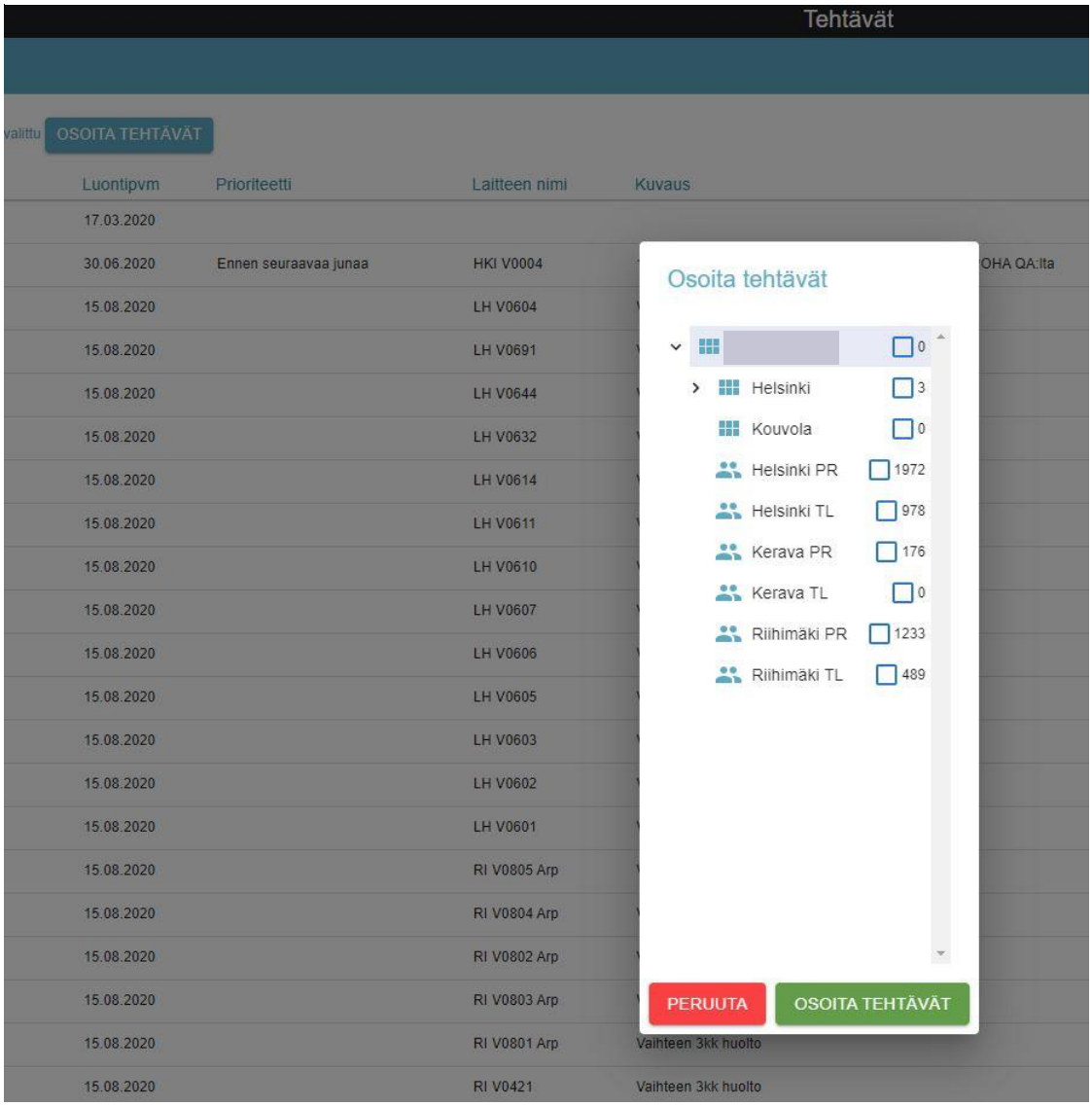
Käytännössä resurssipuun muodostaminen tapahtuu siten että ensimmäinen muodostettava komponentti on puurakenteen sisimmäinen komponentti ja koko iterointikierros on valmis vasta kun prosessin aloittanut komponentti palautetaan. Tämä johtuu siitä, että lopulta prosessin aloittanut komponentti ympäröi kaikki muut puurakenteen komponentit.

Resurssinäkymän (Kuva 18) lisäksi ”ResourceTree”-komponenttia käytetään tehtävien osoitusprosessissa, jossa resurssirivi sisältää myös valintaruudun (Kuva 19). Resurssi-rivillä näytetään tällä hetkellä resurssin nimi ja sille osoitettujen tehtävien lukumäärä.

Kuva 18. Resurssipuu käyttöliittymän resurssinäkymässä



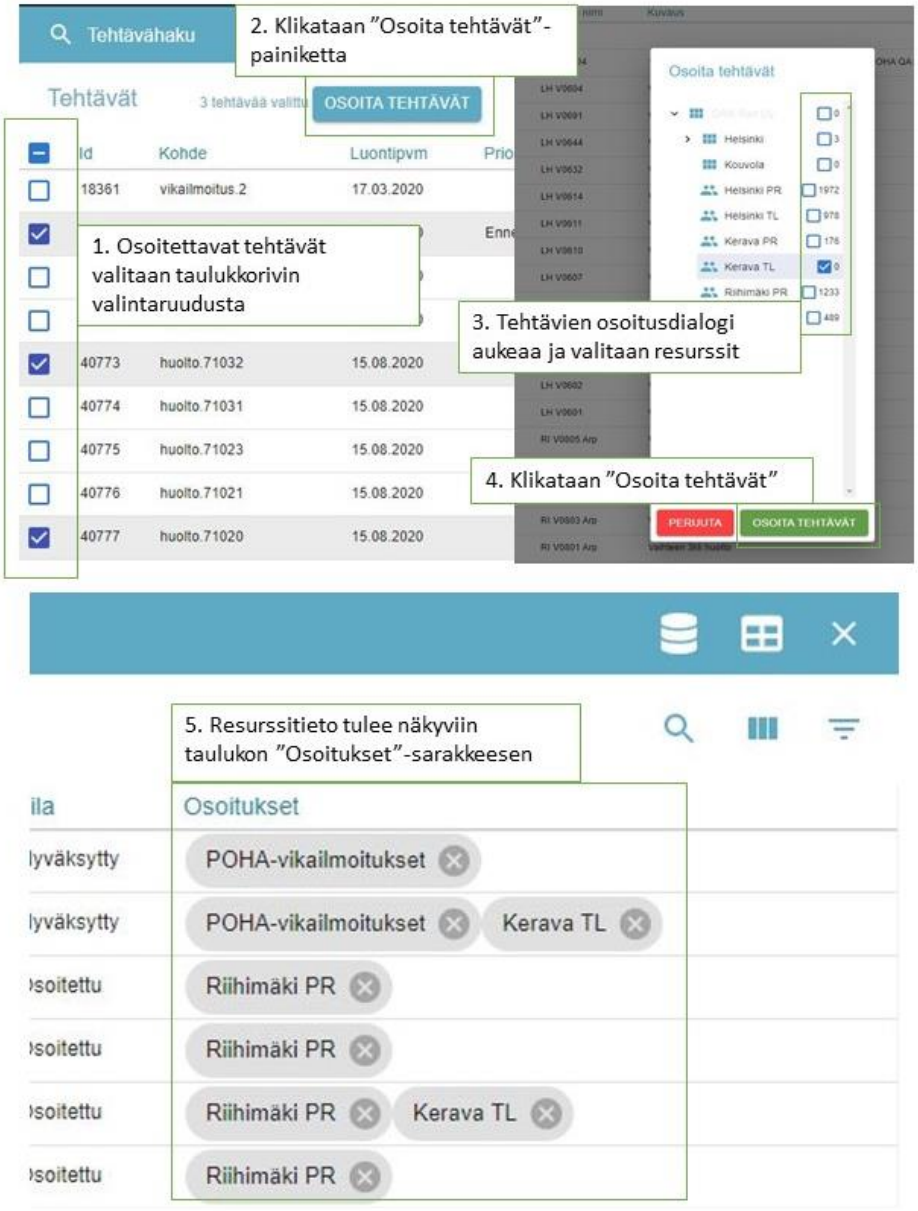
Kuva 19. Resurssipuu osoitusdialogissa



5.6 Tehtävien osoittaminen

Tehtävien osoittaminen on yksi Työnohjauksen tärkeimpiä tehtäviä. Se on sovelluksen ydintoiminnallisuutta, sillä sen avulla työnjohtaja siirtää tehtävän tietylle resurssille hoidettavaksi. Tehtävien osoittamisen prosessi toimii opinnäytetyön kirjoitushetkellä siten että käyttäjä valitsee tehtävälistauksesta osoitettavat tehtävät, klikkaa ”Osoita tehtävät”-painiketta, valitsee avautuvasta resurssipuusta (”PointingTasksDialog”) ne resurssit, joille tehtävä osoitetaan ja vahvistaa osoituksen. Prosessissa resurssitieto lisätään tehtävän tietoihin ja tehtävä lisätään resurssin tietoihin. (Kuva 20.)

Kuva 20. Tehtävien osoitusprosessi

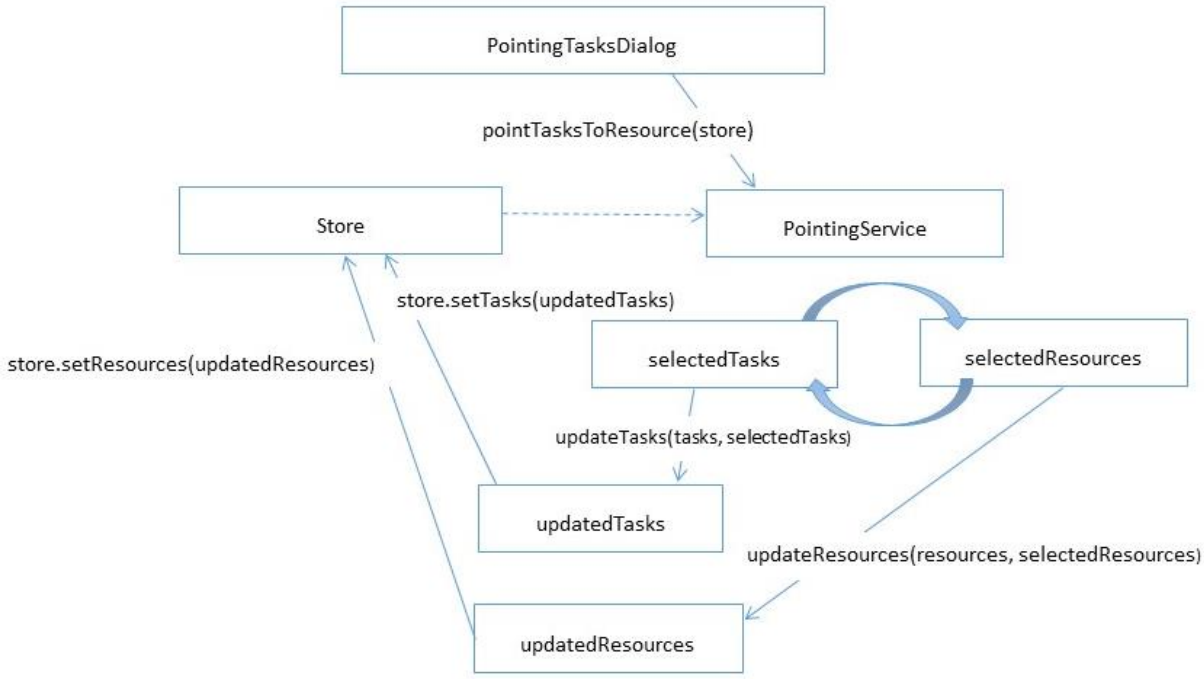


Osoitustoiminto on sellaista sovelluksen toiminnallisuutta, jonka tarkoitus ei ole vain hakea ja näyttää tietoa. Näin ollen sen toteuttamista varten luotiin oma ”Service”-luokka, ”PointingService”. ”PointingService” sisältää käytännössä kaikki osoitustoimintoon tarvittavat funktiot. Osoitustoiminnon funktioiden lisäksi se sisältää osoituksen poistamiseen liittyvät funktiot.

Osoittaminen toimii siten että ”PointingTasksDialog”-komponentista kutsutaan ”PointingService”-luokan ”pointTasksToResource”-funktiota, joka saa parametrina ”store”-objektin. ”PointTasksToResource”-funktiossa haetaan ”store”-objektista tehtävät, valitut tehtävät, resurssit ja valitut resurssit, jonka jälkeen valitut resurssit iteroidaan ja resurssisilmukan sisällä iteroidaan valitut tehtävät.

Tässä silmukkarakenteessa lisätään valittu resurssi valitulle tehtävälle ja valittu tehtävä valitulle resurssille. Tämän jälkeen käydään läpi sovelluksen kaikki tehtävät sekä kaikki resurssit ja päivitetään valitut tehtävät tehtäviin ja valitut resurssit resursseihin. Tämä tapahtuu ”updateTasks”- ja ”updateResources”-funktioissa. Päivittämisen jälkeen ”store”-objektin ”tasks”- ja ”resources”-taulukot korvataan ”PointingService”-luokan päivitetyillä ”tasks”- ja ”resources”-taulukoilla. (Kuva 21).

Kuva 21. Osoitusprosessi teknisesti



Kuvattu osoitusprosessin toteutus on osa käyttöliittymän prototyyppivaiheen toteutusta ja tässä vaiheessa ei ole vielä otettu käyttöön sovelluksen palvelinpuolelle toteutettavia rajapintoja. Osoitusprosessi tulee muuttumaan teknisesti siinä vaiheessa, kun osoitusrajapinnat otetaan käyttöön. Ohjelmistokehityksessä lähdekoodin jatkuva refaktorointi on tärkeää. Osoitusprosessia voidaan yksinkertaistaa käyttöliittymäpuolen

koodissa siinä vaiheessa, kun rajapinnat saadaan käyttöön. Opinnäytetyön toteutusvaiheessa palvelinpuolen rajapintoja ei ollut vielä toteutettu.

6 Yhteenveto

Työnohjauksen uuden web-käyttöliittymän toteutusprojektin prototyyppivaiheessa onnistuttiin saavuttamaan sellainen kokonaisuus, jota pystyttiin demonstroimaan asiakkaalle joulukuussa 2020. Prototyyppivaiheessa toteutettiin tehtävätietojen hakeminen taulukkoon, resurssien hakeminen puunäkymään sekä tehtävien osoitus valituille resursseille. Toteutetut käyttötapaukset sisälsivät sellaisia ominaisuuksia, jotka tekevät sovelluksesta tietyiltä osin käytettävämmän nykyiseen verrattuna, kuten taulukon dynaamiset suodatusvalinnat ja tehtävätaulukon helppo muokattavuus. Toteutettujen käyttötapausten lisäksi prototyyppivaiheessa on kehitetty sovelluksen ulkoasua modernimpaan suuntaan. Joulukuussa järjestetyn demonstraation päätteeksi saatu positiivinen palaute asiakkaalta vahvisti, että web-käyttöliittymän uusimisprojektissa kehitystä on viety oikeaan suuntaan. Positiivista palautetta saatiin muun muassa uuden käyttöliittymän selkeydestä, visuaalisuudesta ja siitä että jo näin alkuvaiheessa on onnistuttu tekemään paljon parannuksia.

Kehitystarpeita kartoitettiin haastattelemalla asiakasyritysten työnjohtajia, mikä toi työnohjaussovelluksen käyttäjiä lähemmäs kehittäjiä. Kun prototyyppiä esiteltiin asiakkaille, esiin nousi jälleen kehityksessä huomioitavia asioita, kuten laajempi resurssien hallinta, jossa tehtäviä voisi osoittaa resurssilta toiselle.

Työnohjauksen uuden web-käyttöliittymän uusimisprojektissa ollaan nyt siinä vaiheessa, että sovelluksella on pohja, josta kehitystä on hyvä jatkaa. Seuraavat kehitysvaiheet ovat julkaisuputken kehittäminen ja kirjautumisen toteuttaminen. Julkaisuputken kehittämisellä tarkoitetaan sitä, miten uusi käyttöliittymä saadaan osaksi Mobilenote-ohjelmistoa ja vietyä testiympäristöön sekä sen jälkeen tuotantoympäristöön.

Kirjautuminen on edellytys sille, että työnohjauksen web-käyttöliittymästä voidaan kutsua palvelinpuolen rajapintoja, jotka hakevat asiakaskohtaista tietoa tietokannasta. Tähän asti on käytetty palvelinpuolen palauttamaa kovakoodattua JSON-tietoa, mutta tarkoitus olisi

seuraavissa kehitysvaiheissa päästä tästä eroon. Tämän jälkeen ratkaistaviksi tulevat asiakaskohtaisten konfiguraatioiden liittäminen osaksi uutta käyttöliittymää ja käyttöliittymän dynamiikan kehittäminen käyttäjäkohtaiseksi.

Projektin seuraavana tavoitteena on, että ensimmäinen kokeiluversio saataisiin asiakkaille testattavaksi keväällä 2021. Tarkoituksena on, että nykyistä työnohjauksen käyttöliittymää ja uutta käyttöliittymää käytetään rinnakkain, eli kaikkia työnohjauksen toimintoja ei tarvitsisi julkaista heti kerralla.

Olen tyytyväinen työnohjauksen web-käyttöliittymän uusimisprojektin prototyyppivaiheessa aikaan saatuun kokonaisuuteen. Käyttöliittymä ei ole vielä valmis, mutta kehitystyötä on hyvä jatkaa tästä eteenpäin. Tämän prosessin aikana olen oppinut tarkastelemaan omia kehitysratkaisuja kriittisesti ja muuttamaan jo toteutettuja osia merkittävästi, mikäli on huomattu, että jossain vaiheessa on menty väärään suuntaan. Taitoni omien kehitysratkaisujen toteuttamiseen ovat karttuneet opinnäytetyöprosessin aikana ja olen kehittynyt erityisesti web-käyttöliittymien kehittäjänä. Opinnäytetyöprosessi on vahvistanut luottamustani omaan ammattitaitooni ohjelmistoalan osaajana.

Lähteet

Benyon, D. (2019). *Designing user experience: A guide to HCI, UX and in-teraction design.* 4. painos. Harlow: Pearson Education Limited.

Biro, J. (2019). *Medium. Browser Engines… Chromium, V8, Blink? Gecko? WebKit?* https://medium.com/@jonbiro/browser-engines-chromium-v8-blink-gecko-webkit-98d6b0490968

Boduch, A. & Derks, R. (2020). *React and React Native: A complete hands-on guide to modern web and mobile development with React.js*. 3. painos. Birmingham: Packt Publishing Ltd.

Cardoso, A. (2020). Storyblok. *Developer Guides. Top Javascript frame-works 2020*. https://www.storyblok.com/tp/top-javascript-frameworks-2020

Codecademy. (2020). *News. Learning to code. What is a programing language?* https://news.codecademy.com/programming-languages/

Dabbs, M. (2019). *The Fundamentals of Web Application Architecture*. https://reinvently.com/blog/fundamentals-web-application-architecture/

DigidWorks. (2020). *DigidWorks. Blog. What's a Web Application*. https://digidworks.com/what-is-web-application-non-tech-explanation

Geometrix Oy. (2020). *Työnohjaus*. https://www.geometrix.fi/?page_id=91

Geometrix Oy. (2020-b). https://geometrix.atlassian.net/browse/NOTE-2044

Github. (2020-a). *mui-org. material-ui*. Haettu 12.12.2020 osoitteesta https://github.com/mui-org/material-ui

Github. (2020-b). *gregnb. mui-datatables*. Haettu 12.12.2020 osoitteesta https://github.com/gregnb/mui-datatables

Github Language Stats. (2020). *GitHut 2.0* Haettu 23.1.2020 osoitteesta https://madnight.github.io/githut/#/pull_requests/2020/4

GlobalStats. (2020). *Statscounter. Browser Market Share Worldwide*. Haettu 25.10.2020 osoitteesta https://gs.statcounter.com/browser-market-share

Kyrnin, J. (2020). *Lifewire. Internet, Networking and Security. Web Development. What Are Markup Languages? https://www.lifewire.com/what-are-markup-languages-3468655*

Material-UI. (2020). *Material-UI*. Haettu 12.12.2020 osoitteesta https://material-ui.com/

MDN web docs. (2020-a). Learn web development. *Getting started with the Web. HTML Basics*. https://developer.mozilla.org/fi/docs/Learn/Getting_started_with_the_web/HTML_basics

MDN web docs. (2020-b). *Web technology for developers. CSS: Cascading Style Sheets.* https://developer.mozilla.org/en-US/docs/Web/CSS

MDN web docs. (2020-c). *Learn web development. Getting started with the Web. CSS basics* https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics

MDN web docs. (2020-d). *Learn web development. JavaScript – Dynamic client-side scripting. JavaScript First Steps. What is JavaScript?* https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

MDN web docs. (2020-e). *Learn web development. Tools and testing. Un-derstanding client-side JavaScript frameworks*. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks

MobX. 2020. *MobX. Ten minute introduction to MobX and React*. https://mobx.js.org/getting-started

Mozilla. (n.d.). *What is a web browser?* https://www.mozilla.org/en-US/firefox/browsers/what-is-a-browser/

Morris, S. (n.d.). *Skillcrush. Blog. Tech 101: What is JavaScript?* https://skillcrush.com/blog/javascript/

PYPL Index. (2020). *PYPL Popularity of Programming Language*. Haettu 14.11.2020 osoitteesta https://pypl.github.io/PYPL.html

React. (2020). *Docs. Hooks API Reference. Basic hooks*. https://reactjs.org/docs/hooks-reference.html#usestate

Schwartzmüller, (M. 2020-a). *Angular & NodeJS - The MEAN Stack Guide [2020 Edition]. Section 1: Getting Started. Lecture: 5. How Does the MEAN Stack Work?* Udemy. https://www.udemy.com/join/login-popup/?next=/course/angular-2-and-nodejs-the-practical-guide/learn/lecture/10416174#overview

Schwartzmüller, M. (2020-b). *React – The Complete Guide [2020 Edition]. Section 1: Getting Started. Lecture: 2. What is React?.* Udemy. https://www.udemy.com/join/login-popup/?next=/course/react-the-complete-guide-incl-redux/learn/lecture/8268490#overview

Schwartzmüller, M. (2020-c). *React – The Complete Guide [2020 Edition]. Section 3: Understanding the Base Features & Syntax. Lecture: 30. Understanding JSX.* Udemy. https://www.udemy.com/join/login-popup/?next=/course/react-the-complete-guide-incl-redux/learn/lecture/8090856#overview

Schwartzmüller, M. (2020-d). *React – The Complete Guide [2020 Edition]. Section 3: Understanding the Base Features & Syntax. Lecture: 32. Creating a functional component.* Udemy. https://www.udemy.com/join/login-popup/?next=/course/react-the-complete-guide-incl-redux/learn/lecture/8090862#overview

Schwartzmüller, M. (2020-e). *React – The Complete Guide [2020 Edition]. Section 3: Understanding the Base Features & Syntax. Lecture: 38. Understanding and using state.* Udemy. https://www.udemy.com/join/login-popup/?next=/course/react-the-complete-guide-incl-redux/learn/lecture/13556100#overview

Schwartzmüller, M. (2020-f). *React – The Complete Guide [2020 Edition]. Section 3: Understanding the Base Features & Syntax. Lecture: 39. Props and State.* Udemy. https://www.udemy.com/join/login-popup/?next=/course/react-the-complete-guide-incl-redux/learn/lecture/8124208#overview

Schwartzmüller, M. (2020-g). *React – The Complete Guide [2020 Edition]. Section 26: React Hooks. Lecture: 426. What are React Hooks?* osoitteesta: https://www.udemy.com/join/login-popup/?next=/course/react-the-complete-guide-incl-redux/learn/lecture/15700176#overview

Stack Overflow. (2020). *2020 Developer Survey*. https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-dreaded

Ström, C. (2019). *React pähkinänkuoressa*. https://joinex.fi/react-pahkinankuoressa/

TIOBE Index. (2020). *Tiobe Index for November 2020*. Haettu 14.11.2020 osoitteesta https://www.tiobe.com/tiobe-index/

TechTerms. (2013). *TechTerms The Computer Dictionary. Software Terms. Framework.* Haettu 22.11.2020 osoitteesta https://techterms.com/definition/framework

Valuy, S. (2020). *A Comparison of Single-Page and Multi-Page Applications. 17.6.2020.* https://dzone.com/articles/the-comparison-of-single-page-and-multi-page-appli

Worldometers. (2021). *Internetin käyttäjiä maailmassa.* Haettu 23.1.2021 osoitteesta https://www.worldometers.info/fi/

ZetCode. (2020). *Axios tutorial.* https://zetcode.com/javascript/axios/

Liite 1: Käsitteistö

Käsite	Määritelmä
AJAX	Asynchronous JavaScript And XML: joukko web-tekniikoita, joiden avulla tiedonsiirto selaimen käyttöliittymäsovelluksen ja palvelunpuolen sovelluksen välillä on mahdollista. Tiedonsiirto tapahtuu HTTP-protokollan yli.
API	Application Programming Interface: ohjelmointirajapinta, jonka välityksellä eri ohjelmistokomponentit ovat yhteydessä toisiinsa.
Axios	AJAX-pyyntöjen toteuttamiseen kehitetty JavaScript-kirjasto.
Back-end	Sovelluksen tietokantaan yhteydessä oleva osa, joka tarjoaa rajapinnat sovelluksen front-endille. Back-endiä kutsutaan myös web-sovellusten palvelinpuoleksi.
CSS	Cascading Stylesheets: verkkosivujen muotoiluun käytetty tyylikieli
Framework	Sovelluskehys
Front-end	Sovelluksen käyttöliittymän toteuttava osa. Front-endiä kutsutaan myös web-sovellusten asiakaspuoleksi.
Hook-funktiot	React-sovelluksessa usein tarvittavien ominaisuuksien käyttöönoton funktionaalisissa komponenteissa mahdollistavat keinot.
HTML	Hyper Text Markup Language: verkkosivujen toteuttamiseen käytetty merkintäkieli
HTTP	Hyper Text Transfer Protocol: internetin tiedonsiirtoprotokolla
JavaScript	Web-sovelluksen toimintojen toteuttamiseen käytettävä korkean tason ohjelmointikieli.
JSON	JavaScript Object Notation: Tiedonvälityksessä käytettävä yksinkertainen tiedostomuoto, joka on MIME-tyypiltään application/json.
Kirjasto	Ohjelmakoodiin lisättävä moduuli, joka yleensä sisältää valmiin ratkaisun usein toistuvaan ongelmaan.
Komponentti	Käyttöliittymäsovelluksen osa, joka toteuttaa tietyn käyttöliittymässä näkyvän ominaisuuden.
Käyttöliittymä	Käyttäjälle näkyvä web-sovelluksen osa, jonka kautta käyttäjä on vuorovaikutuksessa sovelluksen kanssa.
MobX	Web-käyttöliittymäsovellusten tilanhallintaan kehitetty JavaScript-kirjasto
MPA-sovellus	Monen sivun web-sovellus, jossa jokainen sivu haetaan palvelimelta erikseen.
Props-olio	React-komponenttien väliseen tiedon siirtoon käytettävä objekti.
React	Käyttöliittymien kehittämiseen käytettävä JavaScript kirjasto.
REST	Representational State Transfer: ohjelmointirajapintojen toteuttamiseen tarkoitettu arkkitehtuurimalli.
SPA-sovellus	Yhden sivun web-sovellus, jossa tieto päivittyy, mutta sivu on sama.
Tila	React-sovelluksen tai komponentin sisällön arvot tietyllä hetkellä.
URL	Uniform Resource Locator: web-osoite

Liite 2: Luettelo käytetyistä ikoneista

Ikoni	Opinnäytetyön kuvat	Ikonin luojan nimimerkki	Ikonin linkki
	-Kuva 3: Web-sovellusten arkkitehtuuri	Freepik	https://www.flaticon.com/free-icon/user_2698481
	-Kuva 3: Web-sovellusten arkkitehtuuri -Kuva 5: SPA-sovelluksen päivittymisen elinkaari -Kuva 6: MPA-sovelluksen päivittymisen elinkaari	catkuro	https://www.flaticon.com/free-icon/device_1605580
	-Kuva 3: Web-sovellusten arkkitehtuuri -Kuva 5: SPA-sovelluksen päivittymisen elinkaari -Kuva 6: MPA-sovelluksen päivittymisen elinkaari	itim2102	https://www.flaticon.com/free-icon/server_1442868
	-Kuva 3: Web-sovellusten arkkitehtuuri	catkuro	https://www.flaticon.com/free-icon/database_1980263
	-Kuva 3: Web-sovellusten arkkitehtuuri	itim2101	https://www.flaticon.com/free-icon/folder_1780708