Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

MATTI TAMMINEN

Itsearviointimittarin digitalisoiminen

Ohjelmistokehitysprojekti

TIETOJENKÄSITTELYN TUTKINTO-OHJELMA
2022

Tekijä(t)
Tamminen Matti

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
03/2022

Sivumäärä: 43
Liitteet: 2

Julkaisun kieli
suomi

Julkaisun nimi
Itsearviointimittarin digitalisoiminen

Tutkinto-ohjelma
Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Tässä opinnäytetyössä esitellään sovelluskehitysprojektin valmistelu-, suunnittelu- ja toteutusvaiheet anonyymin hankeorganisaation tilaaman paperisen itsearviointimittarin digitalisoimisesta esittelytasoiseksi verkkosovellukseksi.

Tilaajaorganisaatio on käyttänyt työssään paperista itsearviointimittaria asiakkaidensa työllistymispotentiaalin kartoittamiseen sekä siihen liittyvien ongelmien ratkaisuun ja konkreettisten ja tavoitteellisten toimien suunnitteluun asiakkaan työllistymisedellytyksien parantamiseksi. Itsearviointimittarin käyttäminen tapahtuu pääosin tilaajan agentin ja asiakkaan välisessä vuorovaikutuksessa mutta digitaalisen version kautta toivottiin myös asiakkaan itsenäisen käytön mahdollisuutta työkalulle.

Projektin sopimusvaiheessa selvitettiin osapuolien kesken luotavan sovelluksen tavoitteet ja reunaehdot sekä luotiin yhteisesti projektisuunnitelma, joka hyväksyttiin projektin toimintasuunnitelmaksi. Projektisuunnitelman tehtävänä oli selventää projektin resurssit, toteutustapa, osapuolien oikeudet ja velvollisuudet sekä projektin päättämisen ja siirtämisen edellytykset.

Toteutuksen suunnittelu ja projektin alkuvalmistelun käynnistyivät heti sopimuksen tekemisen ja toimintasuunnitelman hyväksymisen jälkeen ja kattoivat sovellusarkkitehtuurin, ulkoasun ja rakenteen suunnittelun. Yhteisenä versionhallinta- ja projektinhallintatyökaluna työssä käytettiin GitHub-verkkopalvelua ja projektin läpivienti toteutettiin ketterien ohjelmistokehitysmenetelmien tiiviillä iteraatiomallilla.

Sovellus toteutettiin lopulta asiakaspään React-sovelluksena, joka keräsi käyttäjän tekemät valinnat ja syötteet ja tarjosi lopulta niiden tallennusta ja tulostusta pdf-muotoisena tiedostona. Esittelyversio julkaistiin Netlify-verkkopalvelun kautta ja luovutettiin tilaajan testikäyttöön vaatimustensa mukaisena.

Avainsanat
Ohjelmointi, Systeemityö, Projektinhallinta, Javascript, React

Author(s) Tamminen, Matti	Type of Publication Bachelor's thesis	Date 03/2022
	Number of pages: 43 Attachments: 2	Language of publication: Finnish

Title of publication
Self-evaluation tool digitalization

Degree program
Business Information Systems

Abstract

In this thesis I describe the planning, preparation and implementation of a software project commissioned by an anonymous organization to replace the paperback version of their self-evaluation tool with a fully functional web application.

The paperback version of the self-evaluation tool was used in client organization to assess their customers employability, to solve potential problems related to their employment, and to identify targeted measures to improve their employment prospects. The new digitized version of the tool was planned to replace the old paperback version in meetings between the client organization representative and the customers, and to provide an option for the customer to use the tool independently.

The planning phase consisted of negotiations with the client to form a unified understanding about the goals, resources, and limitations related to this project. These factors formed the basis of the project contract, which was signed by both parties and used as a common set of rules to define the project resources, the development strategy, the conditions for project termination and postponement and the basic rights and obligations of the project parties.

Project preparations started immediately after the project contract was signed and consisted of software architecture planning, web layout design, and application flow design. GitHub was chosen as a combined version control and project management application for the project and the project was implemented using an agile software development model.

The self-evaluation tool was eventually created as a client-side React-application that collected all the user selections and offered the results based on these selections in PDF format, which could be downloaded or printed out from the browser. The application was released on the Netlify-platform and handed over to the client organization for the testing as promised.

Keywords
Programming, Software development, Project management, Javascript, React

SISÄLLYS

SYMBOLI- JA LYHENNELUETTELO

Käsite:	Selitys:
create-react-app	Ohjelmointiympäristö ja sen rakentamiseen käytetty NPX*-komento React* -sovelluksia varten.
DOM, dokumenttioliomalli	Verkkosivun puumainen rakenne, joka määrittää dokumentin ulkoasun ja tavat sen manipulointiin.
Gauge-kaavio	React-gauge-chart-apukirjaston tarjoama kaavio, jossa data esitetään nopeusmittarimaisena puoliympyränä.
Leivänmuru-navigaatio	Navigointilinkit, joiden kautta käyttäjä voi siirtyä sovelluksen sivunäkymien välillä. (eng. breadcrumb navigation)
NPM tai Node Packet Manager	Paketinhallintajärjestelmä, jonka kautta voi jakaa ja ladata ohjelmointikieli Javascriptin lisäominaisuuksia ja ohjelmia sovelluskehitystöitä varten.
NPX tai Node Packet Executioner	NPM*-pakettien suorittamista varten rakennettu paketinajo-ohjelma.
Radar-kaavio	React-chartjs-2-apukirjaston tarjoama kaavio, jossa data esitetään tutkamaisena viuhkana nollapisteen ympärillä.
React tai React.Js	Javascript-ohjelmointikielen kirjasto käyttöliittymien rakentamista varten.
React-Bootstrap	Komponenttikirjasto, joka sisältää lukuisia valmiita React-komponentteja ja niiden muotoilutapoja.
React-router	Apukirjasto React-näkymien rakentamiseen url-osoite-pohjaisen navigoinnin kautta.

1 JOHDANTO

Tässä opinnäytetyössä esitellään sovelluskehitysprojekti, joka toteutettiin anonyymin hankeorganisaation (myöhemmin ’tilaaja’) ja opinnäytetyön tekijän (myöhemmin ’toimittaja’) yhteistyössä. Projektin tarkoituksena oli kehittää esittelyversio sovelluksesta, jonka avulla tilaaja voisi tehokkaammin kartoittaa asiakkaidensa työllistymisedellytyksiä ja parantaa heidän valmiuksiaan työelämään.

Tilaaja on nykyisessä hankeympäristössään rakentanut omaan käyttöönsä itsearviointimittarin, jota on käytetty paperiversiona osana asiakaskohtaamista, tarkoituksena kartoittaa asiakkaan työllistymisedellytyksiä. Koska valintapohjainen lomake kääntyy perinteisesti erittäin hyvin digitaaliseksi, tilaaja halusi yrittää sen digitalisoimista testattavaan muotoon, jonka perusteella mahdollista jatkokehitystä voidaan paremmin lähteä suunnittelemaan ja budjetoimaan. Tämän uusitun itsearviointimittarin tarkoituksena on tukea asiakkaan ja tilaajan työntekijöiden välistä kohtaamista herättämällä ajatuksia niistä tekijöistä, jotka vaikuttavat eniten asiakkaan työllistymismahdollisuuksiin ja konkretisoida näistä tekijöistä tavoitteita, johon pyrkimällä asiakas voi parantaa työllistymismahdollisuuksiaan ja elämänlaatuaan. Mittaria tulisi pystyä käyttämään osana asiakaskohtaamista tilaajan työntekijän ja asiakkaan välisissä tapaamisissa ja myöhemmässä vaiheessa itsenäisesti asiakkaan toimesta. Mittarin tulee lisäksi tuottaa selkeä tulos, joka sisältää ainakin mittarin täyttämisen yhteydessä selvitetyt kehitystavoitteet ja joka voidaan tulostaa ja/tai tallentaa istunnon päätteeksi.

Näiden vaatimuksien pohjalta oli luontevaa lähestyä mittarin rakentamista kokonaisvaltaisena sovelluskehitysprojektina, joka toteutettaisiin yhdessä tilaajan edustajien kanssa. Tämä projekti tuottaisi tuloksenaan testikäyttövalmiin sovelluksen, jonka kautta tilaaja voisi kartoittaa mittarin soveltuvuutta nykyisen mittarinsa korvaajaksi.

Sovimme kyseisen mittarin digitalisoimisesta sovelluskehitysprojektin muotoisena opinnäytetyönä, jota kuvailen tarkemmin seuraavaksi.

2 PROJEKTISUUNNITELMA

Projektista sopimisen yhteydessä toimittaja valmisteli siihen liittyvän projektisuunnitelman, joka yhdessä tilaajan kanssa soviteltiin molempien vaatimuksien mukaiseksi ja sopivaan tulokseen tähtääväksi. Projektin sovittiin alkavan opinnäytetyösopimuksen allekirjoittamisesta, jonka yhteydessä myös sen liitteenä oleva projektisuunnitelma vahvistettiin toteuttamissuunnitelmaksi. Samassa yhteydessä sovittiin myös alkumäärittelyjen laatimisesta, joiden kautta voitiin vielä paremmin täsmentää vähimmäisvaatimukset toimitettavasta sovelluksesta. Nämä määrittelyt liitettiin myöhemmin osaksi toteuttamissuunnitelmaa.

Projektisuunnitelma on opinnäytetyösopimuksen ensimmäinen liite ja sitä käytettiin projektin koko aikana toiminnan ohjaamiseen. Suunnitelma on kokonaisuudessaan tämän raportin liitteenä. (Liite 1)

2.1 Määrittelyt

Projektisuunnitelman määrittelyosuuden puolella eriteltiin sovelluskehitysprojektien kolme tyypillisintä rajoitusehtoa: laajuus, aikataulu ja resurssit (Brewer & Dittman, 2018, s. 15). Näitä määrittelyjä käytettiin myöhemmin toteutuksen suunnittelun perustana.

2.1.1 Projektin laajuus

Projektin laajuudeksi sovittiin itsenäisen sovelluksen suunnittelun, rakentamisen ja sen luovutukseen liittyvän käyttökoulutuksen ja dokumentoinnin toteuttaminen. Toteuttavan sovelluksen oli tarkoitus sopia testikäytettäväksi tilaajan toimesta tehtävään käyttökokeiluun, jonka yhteydessä tilaaja voisi arvioida sovelluksen soveltuvuutta itsearviointimittarin paperisen version korvaamiseen.

2.1.2 Projektin aikataulu

Koska projekti suoritettiin vastikkeettomana opinnäytetyönä, katsottiin aiheelliseksi sopia projektille aikataulu, joka rajoittaa suorittamisen tarpeetonta venymistä ja asettaa samalla raamit sen vaatimuksille. Suoritusajaksi sovittiin tasan kaksi kuukautta sopimuksen allekirjoittamispäivästä.

2.1.3 Projektin resurssit

Resurssien osalta tehtiin linjaus maksullisten ominaisuuksien ja palvelujen käyttämisestä vain erikseen kirjallisesti sovittuna ja perustaa sovellus lähtökohtaisesti avoimien lähdekoodien ratkaisuihin. Toimittaja lupasi lisäksi järjestää itse sovelluksen kehittämiseen tarvittavat laitteet ja ohjelmistot. Tilaaja puolestaan lupasi osoittaa projektille tarvittavat henkilöt suunnittelua ja sisällön tuottamista varten.

2.2 Projektin läpivienti

Projektille varatun aikataulun puitteissa suoritettaville töille katsottiin aiheelliseksi rakentaa läpivienti, jonka kautta projektia voidaan samanaikaisesti suunnitella, testata ja toteuttaa tehokkaasti. Koska merkittävä osa sovelluksen toimintalogiikasta ja muotoilusta ei ollut etukäteen suunniteltu, katsottiin tarpeelliseksi kehittää sovellus iteratiivisesti sykli kerrallaan, jolloin sovelluksen toiminnallisuuksia voidaan testata tekemisen ohella ja suunnittelu voi siten joustaa tarpeen mukaan sovelluksen edistyessä. Tämä menettely noudattaa ketterien ohjelmistokehitysmallien periaatetta ja on koettu toimivaksi pienten sovellusten kehitystapana (Unhelkar, 2012, s. 34.)

Alkuvalmistelujen jälkeen sovelluksen kehityksen sovittiin jatkuvan noin viikon mittaisissa sykleissä, jolloin uudet toteutettavat ominaisuudet ja mahdolliset muutostyöt voitiin aina sopia sykliä edeltävässä palaverissa. Tämän palaverin yhteydessä oli aina mahdollista esitellä sovelluksen uusia ominaisuuksia sekä testata niitä käytännössä ja sopia yhdessä seuraavista askeleista sovelluksen kehittämisessä.

Jokaisen syklin aikana oli tarkoitus rakentaa palaverissa sovitut ominaisuudet sovellukseen ja valmistautua esittelemään ja raportoimaan ne seuraavan palaverin aikana. Tässä vaiheessa muutoksia ei meneillään oleviin töihin otettu enää vastaan, vaan työrauha taattiin puolin ja toisin projektihenkilöiden välillä. Mahdollisia muutoksia, korjauksia ja bugeja voitiin kuitenkin edelleen kirjata projektikansion töihin, josta niitä voitiin ottaa käsittelyyn aina kun sykliin sisältyvä kehitystyö oli saatu päätökseen tai seuraavan syklin valmistelut olivat käynnissä.

Näitä tiiviitä iteraatiokierroksia oli tarkoitus jatkaa aina projektin valmistumiseen saakka, kuitenkin projektiaikataulun rajoitukset huomioiden. Varautuminen muutoksien tuomiin komplikaatioihin ja muihin mahdollisiin ongelmiin korostui valmistumisen lähestyessä ja syklien sisällön määrä pieneni sen mukana. Viimeisen syklin yhteydessä voitiin lopulta esitellä valmis sovellus, sen dokumentaatio ja lähdekoodi sekä arvioida projektin onnistuminen niiden perusteella.

2.3 Oikeudet ja velvollisuudet

Tilaajan ja toimittajan oikeudet ja velvollisuudet tässä projektissa mukailivat vahvasti opinnäytetyösopimuksen vaatimuksia, mutta käsitteiden avaaminen ja laajentaminen nähtiin silti tarpeelliseksi myös projektisuunnitelman yhteydessä.

Tilaajan keskeisimmät oikeudet sisälsivät tämän toimittamien tietojen ja resurssien luottamuksellisen käsittelyn sekä alkumäärittelyä vastaavan sovelluksen ja sen lähdekoodin vastaanottamisen projektiaikataulun puitteissa tai viimeistään kohtuullisessa ajassa sen jälkeen. Keskeisimmät velvollisuudet puolestaan sisälsivät riittävän tarkan ja realistisen määrittelyn tuottamisen kohtuullisessa ajassa sovelluksen toiminnallisuuksista sekä rahallisten ja ajallisten kulujen välttämisen toimittajan suuntaan.

Toimittajan keskeisimmät velvollisuudet sisälsivät vähintään alkumäärittelyä vastaavan ja määritettyyn käyttötarkoitukseensa mahdollisimman hyvin soveltuvan sovelluksen ja sen lähdekoodin tuottamisen projektiaikataulun puitteissa tai viimeistään kohtuullisessa ajassa sen jälkeen, sekä sovelluksen käsittelyä ja jakamista koskevan

perustasoisen opastuksen järjestämisen sovelluksen luovutuksen yhteydessä. Toimittajan keskeisin oikeus oli kohtuullisen työmäärän jälkeen sovelluksen luovuttaminen keskeneräisenä.

2.4 Siirtäminen ja päättäminen

Projektin suorittaminen vastikkeettomana opinnäytetyönä edellytti reilua tapaa määritellä sekä tilaajalle että toimittajalle sopiva käsitys valmiista sovelluksesta. Vaikka sovellus itsessään toimitettiin vastikkeetta, sen suunnitteluun ja sisällöntuottamiseen voitiin olettaa kuluvan merkittävästi palkallista työaikaa tilaajan organisaatiosta ja tilaajan olevan siten kiinnostunut saamaan ajalleen vastinetta. Toimittajan piti puolestaan varautua siihen, ettei ilmaista työtä voida jatkaa loputtomiin näennäisen keskeneräisyyden varjolla.

Ensisijainen tapa määrittää sovelluksen valmistuminen oli tässä projektissa erikseen sovittu alkumäärittely, jonka yhteydessä molemmat osapuolet sopivat minimaalisista toiminnallisuuksista, joiden kautta sovellus voitiin nähdä tarkoitukseensa sopivana. Näiden määrittelyjen tarkoituksena oli toimia reunaehtoina, joten mitään tarkkoja ominaisuuksia tai keinoja niiden saavuttamiseen ei mainittu, ainoastaan tulevan sovelluksen lopullinen käytettävyys. Alkumäärittelyjen täyttymisen ei vielä automaattisesti sovittu päättävän projektia vaan projektia oli tarkoitus jatkaa vähintään sovitun projektiajan loppuun riippumatta alkumäärittelyjen täyttymisestä. Projektin venyessä määritellyn toteusaikansa yli,voitiin kuitenkin projekti päättää alkumäärittelyjä vastaavana milloin tahansa toimittajan niin päättäessä. Tämän projektin alkumäärittely on kokonaisuudessaan opinnäytetyön liitteenä. (Liite 2)

Lisäksi tilaaja ja toimittaja sopivat, että projekti voidaan päättää tai sen ajankohtaa voidaan siirtää milloin tahansa määritellyn toteutusajan sisällä erikseen tehtävällä kirjallisella sopimuksella. Tämä mahdollisti joustoa tilanteissa, joissa olosuhteet muuttuvat yllättäen. Siirtämisen yhteydessä aiheutuva lisärasite voitiin lisäksi huomioida

myös alkumäärittelyn täyttämisen suhteen, käytännössä lieventäen siihen annettuja määritteitä.

Epäselvyyksien välttämiseksi projektiaikataulua täsmennettiin vielä niin, että projektin aloituspäivämääräksi määritettiin sopimuksen allekirjoituspäivämäärä ja toteutusajaksi sitä seuraavat 60 vuorokautta. Lisäksi sovelluksen kehittämiseen sovittiin kohtuulliseksi työmääräksi 50 tuntia, jonka jälkeen sopimuksen alkumäärittely voitaisiin katsoa täyttyneeksi sovelluksen tilasta riippumatta. Tämän oli tarkoitus kohtuullistaa toimittajan velvollisuuksia tapauksessa, jossa alkumäärittely ylittäisi toimittajan resurssit tai kyvyn sovelluksen rakentamiseen.

3 TOTEUTUKSEN SUUNNITTELU

Projektin aloittamisen jälkeen varsinaisen työosuuden ensimmäisenä vaiheena oli toteutuksen suunnittelu, jonka tarkoituksena oli luoda yhtenäinen ajatus siitä, miltä sovellus näyttää, miten sitä tulee käyttää ja mitä voidaan pitää sen onnistuneena lopputuotoksena. Tämä työvaihe toteutettiin tiiviissä yhteistyössä tilaajan kanssa, jotta tekninen toteutus ja tilaajan kaipaama hyöty kohtaisivat mahdollisimman hyvin lopputuloksessa.

3.1 Käyttötapaus

Käyttötapauksen tai käyttötapausten selvittäminen on looginen ensimmäinen askel sovelluksen suunnittelussa, koska se tarjoaa hyvää käytännön tietoa sovelluksen käyttäjistä, käytöstä ja mahdollisista tavoitteista (usability.gov, 2022).

Suunniteltavan sovelluksen perustana käytettiin tilaajalla nykyisellään käytössä olevaan paperista itsearviointimittaria, jonka käyttö oli jo vakiintunutta ja joka tarjosi siten hyvin relevanttia tietoa myös uuden version tarpeisiin.

Tilaajan tyypillinen itsearviointimittarin käyttämiseen liittyvä asiakaskohtaaminen alkaa keskustelulla, jossa ohjaaja kertoo tapaamisen kulusta ja tavoitteista sekä tapaamisessa käytettävästä mittarista. Koska mittari täytetään yhdessä ohjaajan kanssa, erityisiä ohjeita ei välttämättä tarvita ja ohjeistuksen voi pääosin keskittää navigoinnin ja perustoiminnan kertaamiseen.

Seuraavaksi ohjaaja ja asiakas siirtyvät osioon, jossa arvioidaan karkealla tasolla asiakkaan lähtötilanne eri työllistymiseen liittyvistä asiakokonaisuuksista. Tähän osioon sisältyy osa-aluekohtaisten valintojen tekeminen asteikolla 1–5, neljän eri osa-alueen osalta. Näiden valintojen perusteella syntyy välitulos, joka yhdessä ohjaajan kanssa käydään läpi paperiversion puolella käsin piirrettävän kaavion avulla.

Kohtaaminen jatkuu kaavion läpikäynnin jälkeen tarkennuksilla, joita tehdään seuraavaksi, jälleen osa-aluekohtaisesti. Näiden tarkennuksien tehtävänä on paremmin selvittää syyt valinnan tekemiseen ja löytää merkitseviä asioita sen taustalta. Tarkennukset kirjataan ylös ja niiden perusteella etsitään merkittävimmät tekijät nykyiselle tilanteelle ja kirjataan nämä tavoitteiksi, joihin asiakas voi keskittyä omien työllistymismahdollisuuksiensa parantamiseksi. Nämä tavoitteet annetaan asiakkaalle kirjallisena mukaan tapaamisen päätteeksi.

3.2 Arkkitehtuurin suunnittelu

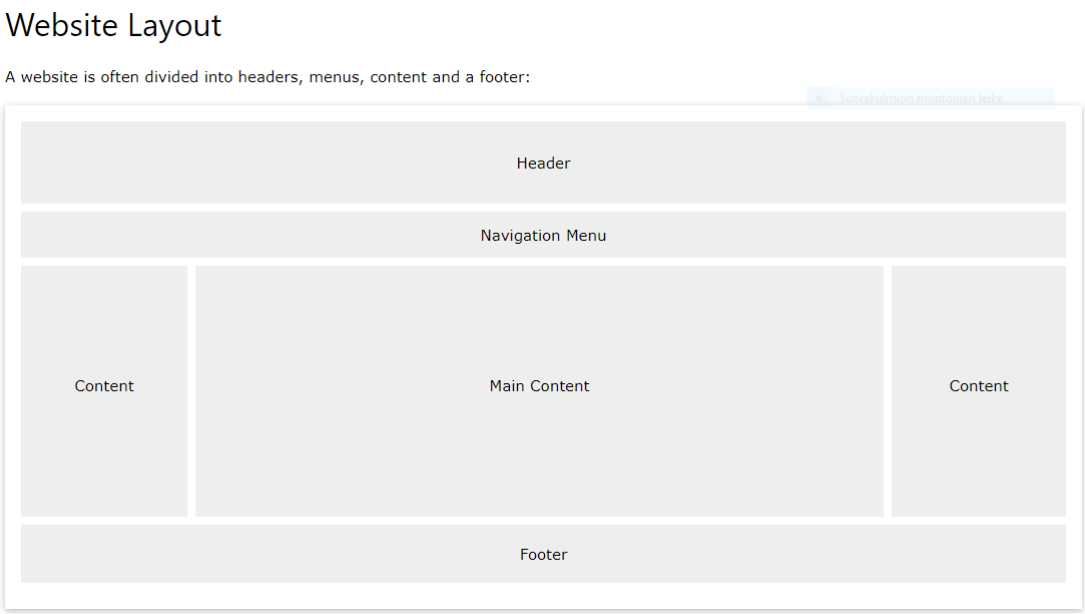
Projektin tavoite, projektin resurssit ja sovelluksen tyypillinen käyttötapaus määrittelivät sovellukselle raamit, joiden kautta voitiin suunnitella sovellusarkkitehtuuri. Tämän arkkitehtuurin tehtävä on linjata ne tekniset toteutustavat, jotka parhaiten mahdollistavat halutun käyttökokemuksen ja tuloksen saavuttamisen niillä resursseilla, jotka projektille on annettu. (Goto & Cotler, 2003, s. 47 ja s. 52–53.)

Sovelluksen laajuudeksi päätettiin rajata pelkkä käyttäjäpuolen toteutusmalli yksisivuisesta verkkosovelluksesta, ilman serveripuolta ja tietokantayhteyttä. Ohjelmointikieleksi valittiin Javascript ja sen apukirjasto React, joka on nopeutensa ja laajuutensa

puolesta hyvä vaihtoehto yksinkertaisen verkkosovelluksen rakentamiseen. Kirjautumista sovellukseen ei vaadita, eikä sovelluksen loki- tai käyttötietoja kerätä lainkaan.

3.3 Ulkoasun suunnittelu

Vaikka tilaaja ei antanut sovelluksen ulkoasulle mitään valmista suunnitelmaa, hän toivoi sen mukailevan aikaisemmin käytössä ollutta paperiversiota, jota sekä tilaajan henkilökunta että asiakkaat osasivat jo sujuvasti käyttää. Klassinen verkkosivuston ulkoasu (Kuva 1.) muistutti kuitenkin niin vahvasti esimerkkirakennetta, että ulkoasun suunnittelu oli luontevaa aloittaa sen pohjalta. Lopulliseen suunnitelmaan jäsentyikin näin ylätunnisteen, tietoalueen ja alatunnisteen sisältämä näkymä, jossa ylätunniste vastaa otsikkotietojen ja navigaation järjestämisestä, alatunniste yhteystietojen ja yhteystyökumppaneiden esittelystä ja varsinainen toimintalogiikka esitetään näiden välissä.



Kuva 1. Verkkosivuston klassinen ulkoasu (W3schools, 2022).

Kuvassa näkyvä perinteinen sivustorakenne toteutettiin sovelluksen pohjaratkaisuksi lähes sellaisenaan, ainoa muutos rakenteeseen oli navigaation siirtäminen osaksi ylätunnisteen toiminnallisuutta. Tätä pohjaratkaisua suunniteltiin käytettävän läpi koko sovelluksen.

3.4 Rakenteen suunnittelu

Sovelluksen rakenteen suunnittelu käsitti projektissa arkkitehtuuriratkaisuun perustuen tärkeimpien React-komponenttien suunnittelun sekä niiden osana käytettävien ulkopuolisten kirjastojen valinnat.

Ulkopuolisista kirjastoista päätettiin heti sovelluksen aluksi ottaa käyttöön React-Bootstrap-kirjasto, joka tarjoaa valmiiden peruskomponenttien lisäksi yksinkertaistettuja muotoiluasetuksia ja lukuisia valmiita erikoiskomponentteja sovelluksen käyttöön (React-Bootsrap dokumentaatio, 2021). Muiden apukirjastojen käyttäminen päätettiin suunnitteluvaiheessa jättää tarpeen mukaisen päätännän varaan.

Laajin komponenttinäkymä sovelluksessa on sivunäkymä, joka edustaa yhtä selaimessa kerrallaan esitettävää sivua (Ogidan, 2021). Näiden sivunäkymien muuttaminen edustaa yksisivuisen verkkosovelluksen navigointia, toimien siten luontaisena rakennuspalikkana toimintalogiikan suunnittelussa.

Sovelluksen sivunäkymät:

1. **päänäkymä**, jonka kautta staattinen ylä- ja alatunniste sekä muut näkymät esitetään ja ylläpidetään globaalia tilaa ja navigaatiota
2. **etusivu**, looginen aloituspaikka, jossa tarjotaan ohjeistusta käyttäjille
3. **valintasivu**, jossa tehdään karkean tason arviointi asiakkaan tilanteesta ja jota voidaan käyttää uudelleen jokaisen kategorian osalta
4. **yhteenvetosivu**, jossa kaikki kategoriat esitellään vielä kerralla ja tehdyt valinnat vahvistetaan
5. **välitulossivu**, jossa esitellään karkean tason arviointien tulos
6. **tarkennussivu**, jossa tehdään tarkennetut havainnot asiakkaan tilanteesta ja jota voidaan käyttää uudelleen jokaisen kategorian osalta
7. **tavoitesivu**, jossa tarkennukset esitellään vielä kerralla ja valitaan tärkeimmät kohteet omiksi tavoitteiksi
8. **lopputulossivu**, jossa esitellään kokonaistulos mittarin täytöstä ja henkilökohtaisten tavoitteiden valinnat

Sivunäkymien suunnittelun yhteydessä nähtiin myös tarpeelliseksi erotella sovellus kahteen loogiseen kokonaisuuteen toimintalogiikan perusteella. Ensimmäinen näistä osioista vastaa asiakkaan tilan karkean tason selvittämisestä ja keskustelun avaamisesta ja toinen osio konkreettisten ja tarkempien havaintojen tekemisestä. Osiot nimettiin toimintansa perusteella valintaosioksi ja tarkennusosioksi.

3.5 Navigoinnin suunnittelu

Verkkosovelluksen tehokas käyttäminen edellyttää sujuvaa ja intuitiivista navigointia, jonka suunnittelua varten pitää selvittää säännöt käyttäjän sallituista ja kielletyistä liikkeistä sekä niihin liittyvistä rajoituksista. (Goto & Cotler, 2003, s. 101–104.)

Sovelluksen käyttäminen alkaa etusivulta, jolle ohjataan kaikki juuriosoitteeseen tai sivun tuntemattomiin osoitteisiin kohdistuvat kutsut. Etusivu toimii sovelluksen alkupisteenä ja sen kautta voidaan sovelluksen käyttöä jatkaa sivulta toiselle tapahtuvalla nappiohjauksella aina lopputulossivulle saakka, jossa lopulta tarjotaan käyttäjälle mahdollisuus pyyhkiä istunnon tiedot puhtaaksi ja palata takaisin etusivulle.

Koska yksisivuinen verkkosovellus ei mahdollista liikkumista taaksepäin selaintoimintoja käyttäen ja sovellus sisältää useita peräkkäisiä sivuja, oli tarpeellista järjestää paluumahdollisuus käyttäjälle ohjelmallisesti toteutettuna, jotta korjailuja ja muutoksia voidaan tehdä. Tätä varten päätettiin toteuttaa ylätunnisteelle osiokohtainen leivänmuru-navigaatio (eng. breadcrumb navigation) (Kuva 2), jonka linkkejä seuraamalla osion sivujen välillä voi liikkua sujuvasti.

Kuva 2. Leivänmuru-navigaation esimerkki (React-Bootsrap dokumentaatio).

Tämä navigaatio sisältää valintaosion suhteen sivunäkymät valintasivujen ja yhteenvetosivun osalta sekä tarkennusosion suhteen tarkennussivujen ja tavoitesivun osalta,

jotta voidaan estää tarpeeton liikkuminen suoraan tulossivuille ja näin ehkäistä irrelevanttien tuloksien luomista ja tulossivujen tarpeetonta lataamista.

4 REACT-SOVELLUKSEN OMINAISUUDET

React tai React.Js on Javascript ohjelmointikielen käyttöliittymien rakentamiseen tarkoitettu apukirjasto (React-dokumentaatio, 2021). React pyrkii luomaan tehokkaampia ja suorituskykyisempiä verkkosovelluksia tarjoamalla mahdollisuuden renderöidä vain tilan muutoksen tarvitsemia sivun osa-alueita interaktiivisen käytön yhteydessä kustannusoptimoituja React-elementtejä hyödyntäen.

4.1 React-komponentit

React-sovellukset rakennetaan modulaarisesti hajottamalla monimutkainen rakenne pienempiin, selvästi yhdestä toiminnosta vastaaviin osiin. Näitä rakennuspalikoita kutsutaan komponenteiksi. Komponentti voidaan rakentaa käyttöliittymän jokaisesta eriteltävästä osasta ja komponentti voi käsittää myös useita muita komponentteja sisältävän komponentin. Komponentteja voidaan ajatella atomisen mallin mukaisesti, jossa yksittäisen html-elementin palauttavaa komponenttia kutsutaan atomiksi, useamman atomin muodostavan yhtenäisen elementin palauttavaa komponenttia molekyyliksi ja useammasta molekyylistä rakentuvan yhtenäisen elementin palauttavaa komponenttia organismiksi. (Ogidan, 2021.)

Tässä työssä esitelty React-sovellus rakentaa komponentit käyttämällä omaa JSX-syntaksiaan, joka yhdistää perinteisen HTML-merkintäkielen ja Javascript-ohjelmointikielen yhdeksi kokonaisuudeksi (Cassio de Sousa, 2015, luku 2.3). Tämä kokonaisuus sisältää kaikki komponentin tarvitsemat muuttujat ja toimintalogiikan sekä sen renderöimiseen tarvittavat HTML-elementit. Komponentti on siten oma riippumaton yksikkönsä, joka ei ole lainkaan sidottu sen luontiympäristöön tai toimintaan. Useimmiten monimutkaisempien komponenttien tehokas uudelleenkäyttäminen vaatii kuitenkin toimiakseen parametreja, joiden avulla komponentit voivat syöttää dataa toisilleen.

React mahdollistaa tällaisien parametrien antamisen elementtien ominaisuuksina, joista rakentuu jokaiseen komponenttiin yksi ”props-objekti”. Tämä sisältää ominaisuuksinaan kaiken sen datan mitä sen elementille on emokomponentissa annettu. Objektin ominaisuuksiin viittaamalla voidaan vapaasti kutsua muuttujia ja funktioita, joita elementille on syötetty sekä syöttää ne samalla tavoin eteenpäin komponentin omille lapsikomponenteille.

Vaikka komponentti voi sisältää hyvinkin laajan toimintalogiikan, on se pohjimmiltaan puhdas Javascript-funktio, joka palauttaa yhden React-objektin. Tätä objektia voidaan hallita kuten muitakin Javascript-objekteja, joten syntaksin tulkitseminen on helppoa myös kokemattomampien kehittäjien osalta. Komponenttirakenne mahdollistaa myös erittäin luettavan koodin kirjoittamisen, jossa jokainen komponentti voidaan nimetä kuvaavasti ja piilottaa sen sisälle oma toimintalogiikkansa, pitäen ylätason näkymien tulkinnan yksinkertaisena ja siistinä.

4.2 DOM ja React DOM

DOM, eli dokumenttioliomalli, on ohjelmoinnin termi kuvata verkkosivun puumaista rakennetta, joka määrittää dokumentin ulkoasun lisäksi tavat sen osien viittaamiseen ja manipulointiin. Tämä rakenne piirretään selaimen toimesta näkymiin ennalta sovittujen sääntöjen mukaan, tarjoten käyttäjälle yhdenmukaisen käyttökokemuksen vierailtavista verkkosivuista riippumatta. (Mozilla developer network, 2022a.)

Koska käyttäjillä on useimmiten tarvetta hakea, muuttaa, lisätä tai poistaa jotain sivun sisältöä, on tarpeellista varautua interaktiiviseen käyttöön manipuloimalla DOM:n sisältöä jollain tapaa. Tähän tarkoitukseen on olemassa valmiita Javascript-ominaisuuksia ja funktioita mutta toiminnan korkea abstraktiotaso ja monimutkaisuus on luonut kysyntää ulkopuolisille kirjastoille, joiden kautta toiminta voidaan suorittaa selkeämmin ja tehokkaammin. Tässä yhteydessä keskitymme lähinnä React-kirjaston toimintaan, mutta sen lisäksi sovelluksessa on hyödynnettävissä myös muiden kirjastojen tekniikoita.

React nojaa DOM-manipuloinnissaan hyvin omaperäiseen ratkaisuun, jossa se ylläpitää omaa virtuaalista dokumenttioliomalliaan varsinaisen DOM:n rinnalla ja rakentaa sen pohjalta uuden elementin varsinaiseen DOM:iin aina kun elementin tila muuttuu. React-DOM vertaa siis jatkuvasti varsinaista DOM-rakennetta omaan React-DOM-rakenteeseensa ja muuttaa vain tarvittavaa elementtiä, kun tilanmuutos havaitaan. Koska React-elementit ovat puhtaita objekteja ja niihin kohdistetaan vain yksinkertaiset käskyt: tuhoa tai rakenna, on operaation resurssikustannus minimaalinen. Näin sovelluksen suorituskyky säilyy erittäin korkealla virheriskin samalla pysyessä pienenä. React-sovelluksen näkymää voikin ajatella tavallaan valokuvana käyttöliittymästä jollain ajan hetkellä, joka korvataan uudella valokuvalla taas uudella ajan hetkellä. (Cassio de Sousa, 2015, luku 2.8.)

4.3 Tilan hallinta

React-sovelluksen tila muuttuu erilaisten tapahtumien seurauksena. Näitä voivat olla esimerkiksi lomakkeen lähettäminen, painikkeen tai linkin painaminen, funktion palautusarvon lataaminen tai mitä tahansa sellainen tapahtuma, joka muuttaa jotain näkymän sisältä. Näiden tapahtumien seuraamiseen voidaan käyttää tilamuuttujia, joiden tehtävänä on hallita tilan muuttamista ja muuttamisen seurauksia, sekä tarvittaessa alustaa tila, komponenttien sisällä.

Komponentti voi ylläpitää tilamuuttujia sen kirjoitusmuodon mukaan joko konstruktorin ja luokan ominaisuuksien kautta luokkakomponentissa tai React-tilakoukkujen kautta funktiokomponenteissa. Tämä työ on rakennettu täysin funktiokomponenttien ja tilakoukkujen kautta. Tilakoukut ovat valmiita React-toimintoja, joiden avulla voidaan ylläpitää tietoa tilasta ja muuttaa sitä hallitusti suorituksen eri vaiheissa.

Yksinkertaisin ja yleisin tilakoukku on useState, jossa luodaan muuttuja – funktiopari. Tämän parin muuttuja säilyttää tilan näkymän piirtojen välillä ja funktio vastaa muuttujan arvon muuttamisesta turvallisesti. Tyypillinen käyttökohde on arvon säilyttäminen käyttäjän tekemästä valinnasta tai lähettämästä lomakkeesta. Tämä arvo voidaan

alustaa ensimmäisen piirtämisen yhteydessä ja sitä voidaan koska tahansa muuttaa, kutsumalla funktiota uudella arvolla.

Toinen erittäin yleinen tilakoukku on useEffect, jossa luodaan funktio, jonka sisällä voidaan hallita muita tilakoukkuja tai komponentin ominaisuuksia aina kun seurattava tapahtuma tapahtuu. Tällainen tilakoukku on kätevä suorittamaan toimenpiteitä esimerkiksi ennen komponentin piirtämistä tai suorittamaan siivoustoimia sen tuhoamisen jälkeen tai seuraamaan käyttäjän toimista riippumattomien muutoksien tapahtumista komponentin lataamisen jälkeen.

4.4 Yksisivuinen verkkosovellus ja react-router

React-kirjaston pääperiaatteisiin kuuluun verkkosivun rakentaminen yksisivuisena sovelluksena, jossa ohjelma rakennetaan toimimaan yhden verkkosivun kautta, ladaten käyttäjälle näkyviin sivun osia kulloisenkin tarpeen mukaisesti. Käytännössä useimmat verkkosivut ovat kuitenkin niin laajoja, että kaiken toiminnallisuuden rakentaminen yhdelle näkymälle käy nopeasti epäselväksi ja hankalaksi ylläpitää. Tähän ongelmaan ratkaisuna on apukirjasto react-router, jonka kautta voi rakentaa useita erillisiä sivunäkymiä ja siten simuloida liikkumista eri sivujen välillä menettämättä tallennettuja tiloja ja lisättyjä elementtejä (Cassio de Sousa, 2015, luku 5.3).

5 PROJEKTIN ALOITTAMINEN

5.1 Projektiympäristön luominen

React-projektiympäristön luominen on mahdollista tehdä käsin mutta helpommin ja nopeammin se tapahtuu käyttämällä NPM-paketinhallintajärjestelmän (Node Packet Manager) NPX-paketinajo-ohjelmaa. Yksirivinen komento ajaa lyhyen alustusohjelman, joka luo valmiiksi projektikansion ja sen sisälle keskeiset tiedostot React-projektin käynnistämiseen.

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

Kuva 3. React-projektiympäristön luomiskomento (React dokumentaatio, 2022).

Luotavien tiedostojen muoto ja määrä vaihtelevat voimakkaasti kirjaston versioiden välillä. Lisäksi komentoon on mahdollista lisätä lippuja, jotka muuttavat lopputulosta eri suuntiin. Tyypillisen alustuksen mukana tuodaan kaikki toimivan sivuston rakentamisen tiedostot, perustason testitiedosto sekä esimerkkisivua varten tarjottavat logot ja ikonit.

```
my-app/
  README.md
  node_modules/
  package.json
  public/
    index.html
    favicon.ico
  src/
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
```

Kuva 4. Tyypillinen luomiskomennon kautta rakennettu tiedostopuu (Create-React-App, 2022).

Projektin edetessä tätä valmista projektiympäristöä laajennetaan tarpeiden mukaan lisäämällä paketti, kirjasto tai ominaisuus paketinhalllintajärjestelmän saatavilla olevien vaihtoehtojen mukaan. Koska sekä ohjelmointikieli Javascript että sen kirjasto React ovat äärimmäisen suosittuja sekä asiakas- että serveripään ohjelmoijien parissa, on valikoima erittäin laaja ja mahdollistaa hyvin monipuolisten kokonaisuuksien rakentamisen valmiiden työkalujen avulla.

5.2 Versionhallinta

Laajempien sovelluskehitysprojektien osalta keskeinen riski on tehdyn työn menettäminen tai sen korruptoituminen. GitHub (GitHub, 2022) on ilmainen ja erittäin suosittu

versionhallintasivusto, joka tarjoaa valmiit ja testatut ratkaisut juuri näihin riskeihin sekä useita elämää helpottavia lisäpalveluja projektinhallintaan.

GitHub toimii verkkosivustolle rakennettavien projektikohtaisten säiliöiden kautta, jonne voidaan ladata nopeasti paikalliset projektitiedostot joko suoraan kansiosta tai sovelluskehitystyökalun kautta. Jokainen päivitys kirjataan automaattisesti lokiin, jonka kautta muutoksia voidaan erikseen vertailla tai vanhoja versioita palauttaa. Kehitystä voidaan lisäksi erotella eri haaroihin, joita voidaan kehittää toisistaan riippumatta ja tarpeen mukaan yhdistää päähaaraan myöhemmin. Sivustolla on lisäksi mahdollista ylläpitää muutospyyntöjen hallintaa, jossa voidaan ilmoittaa virheistä, bugeista tai lisäominaisuuspyynnöistä ja keskustella tarvittaessa aiheesta. Näiden muutospyyntöjen osalta voidaan myös erikseen ylläpitää niihin liittyvää projektinäkymää, jossa toimenpiteisiin voidaan yhdistää tiloja tulevien, rakenteilla olevien ja valmiiden muutoksien suhteen.

5.3 Viestintä ja yhteydenpito

Koska projektiin liittyy paljon vielä määrittelemätöntä työtä ja sisältöä, ennakoimme tarvitsevamme suhteellisen tiivistä yhteydenpitoa projektin aikana. Projektin aloittamiseen liittyvä aloituspalaveri päätettiin pitää toimistolla ja jatkaa sen jälkeen säännöllistä yhteydenpitoa videoneuvotteluohjelmistoja ja sähköpostia hyödyntäen.

Videoneuvottelujen välineiksi valittiin Microsoft Teams ja Google Meet, joiden osalta molemmilla osapuolilla oli ennestään kokemusta. Videoneuvottelut päätettiin ajoittaa jokaisen kehityssyklin alkuun, jolloin edellisen syklin aikana rakennettuja ominaisuuksia pystytään esittelemään ja samalla sopimaan uuden syklin aikana tehtävistä ominaisuuksista tai jo rakennettujen ominaisuuksien muutoksista. Näiden säännöllisten palaverien kautta projektin osallistujat pääsevät tutustumaan sovelluksen toimintaan jo kehitysvaiheessa ja varmistamaan sen soveltumisen käyttöönsä ennen kuin väärinkäsitykset tai tulkintavirheet hukkaavat liikaa aikaa kehitystyöstä.

Lisäksi katsottiin aiheelliseksi rajoittaa materiaalin jakamista sähköpostin kautta projektiin liittyen, jotta kanava säilyisi paremmin avoimena muiden töiden suhteen.

Materiaalin jakamiseen ja toimeksiantoihin sovittiin pääasialliseksi kanavaksi GitHub, jonka kautta resursseja voidaan ladata suoraan projektikansioon ja sitä kautta kaikkien osapuolien käyttöön sekä jättää toimeksiantoja tai ilmoituksia projektin kulloisiinkin työvaiheisiin liittyen. Ohjelman käyttöön suoritettiin pikakoulutus toimittajan järjestämänä.

6 PROJEKTIN TOTEUTUSVAIHE

Projektin toteutusvaiheessa suoritettiin varsinainen ohjelmointityö, jossa sovellus rakennettiin osio kerrallaan lyhyiden iteraatiokierroksien kautta valmistumiseen asti. Seuraavaksi on tarkoitus kuvailla lyhyesti sovelluksen rakentamisen prosessia, sovelluksen tärkeimpiä komponentteja ja ulkoisia apuvälineitä.

6.1 Päänäkymä

Sovelluksen rakentaminen aloitettiin päänäkymästä, jonka tehtävänä sovelluksessa on globaalien tilojen ja perusrakenteen ylläpito. Päänäkymän elementti mukailee tarkasti suunnitteluvaiheessa päätetyn pohjarakenteen muotoa, sisältäen staattisen ylä- ja alatunnisteen sekä niiden väliin aina tarpeen mukaisesti luotavan sivunäkymän. Sivunäkymä tuotetaan päänäkymän alle React-routerin tarjoamalla Switch-elementillä, jossa sivun vaihtamista simuloidaan renderöimällä kohdeosoitteen mukainen sivunäkymä aina käyttäjän navigoinnin mukaan. Koska kaikki tarjottavat sivunäkymät rakennetaan juuri päänäkymän alla, on niiden käyttämät tilat luontevaa varastoida päänäkymässä ja tarjota hierarkiassa alaspäin aina tarpeen mukaan.

```
// Päänäkymän komponentti, sisältäen ylätunnisteen (Header), react-routerin (Switch) ja alatunnisteen (Footer).
return (
  <Container className="main" fluid>
      <Header/>
      <Switch>
        <Route path={adjustPath[5]} component={ () => <AdjustReady targetList={targetList} all={all} avg={average(all) * 20 /
        <Route path={adjustPath[4]} component={ () => <AdjustTotal plusList={shuffle(plusLists.flat())} minusList={shuffle(m
        <Route path={adjustPath[3]} component={ () => <Adjust plusList={dPlusList} setPlusList={setDPlusList}  minusList={dM
        <Route path={adjustPath[2]} component={ () => <Adjust plusList={cPlusList} setPlusList={setCPlusList} minusList={cMi
        <Route path={adjustPath[1]} component={ () => <Adjust plusList={bPlusList} setPlusList={setBPlusList} minusList={bMi
        <Route path={adjustPath[0]} component={ () => <Adjust plusList={aPlusList} setPlusList={setAPlusList} minusList={aMi
        <Route path={choicePath[7]} component={ () => <ChoiceReady all={all} /> } />
        <Route path={choicePath[6]} component={ () => <ChoiceTotal all={all} sets={allSets} init={ totalInit } setInit={ tot
        <Route path={choicePath[5]} component={ () => <Choice choice={dChoice} setChoice={setDChoice} link={ choicePath[6] }
        <Route path={choicePath[4]} component={ () => <Choice choice={cChoice} setChoice={setCChoice} link={ choicePath[5] }
        <Route path={choicePath[3]} component={ () => <Choice choice={bChoice} setChoice={setBChoice} link={ choicePath[4] }
        <Route path={choicePath[2]} component={ () => <Choice choice={aChoice} setChoice={setAChoice} link={ choicePath[3] }
        <Route path={choicePath[1]} component={ () => <Front /> } />
        <Route exact path={choicePath[0]}><Redirect to={choicePath[1]} /></Route>
        <Route path="*"><Redirect to={choicePath[1]} /></Route>
      </Switch>
      <Footer />
  </Container>
);
};

export default Main;
```

Kuva 5. Päänäkymän komponentti, jossa ylä- ja alatunniste sekä sivunäkymät sijaitsevat. Sivunäkymä renderöidään ehdollisesti käyttäjän navigoinnin mukaan.

Sovelluksen kaikki varsinaiset sivunäkymät toimivat päänäkymän alikomponentteina ja voivat säilyttää tarvitsemansa tilat päänäkymän alla ilman, että elementtien tuhoaminen ja uudelleen luominen käyttämisen yhteydessä kadottaa mitään olennaisia tietoja. Sovelluksen sisällä tehtävät valinnat ja asetetut tavoitteet voidaan näin ylläpitää puhtaasti laitteen keskusmuistissa, eikä selaimen omaa muistia tai erillistä serveripuolen toteutusta tässä sovelluksessa tarvita.

6.2 Ylä- ja alatunniste

Sovelluksen päänäkymän alla esitetyt ylä- ja alatunnisteet ovat näkyvissä läpi koko sovelluksen käytön ja niillä on tärkeä rooli sovelluksen käytössä. Molemmat komponentit toteutettiin React-Bootstrap-kirjaston valmiin Navbar-komponentin kautta. Valmiin elementin etuna toteutuksessa oli sen tarjoama yksinkertaistettu responsiivinen muotoilu ja sisäisten elementtien asettelu, jota hyödyntämällä yhtenäinen ilme molempien tunnisteiden välillä oli helppo saavuttaa.

```
const Header = () => {
  const location = useLocation()
  return (
    <Navbar expand={false}>
      <Container fluid className="row m-0">
        <Navbar.Brand className="col-lg-3">
          <Link to={choicePath[1]} className="headerLogoContainer"><Image src="/images/logofull.jpg" alt="Kokemuksella Duuniin logo" className="headerLogo" /></Link>
        </Navbar.Brand>
        <Navbar.Text className="col-lg-3 text-center">
          <Navbar.Brand className="headerText">{location.pathname.substring(1).toUpperCase()}</Navbar.Brand>
        </Navbar.Text>
        <Nav className="col-md-12 col-lg-3">
          <Breadcrumb className="">
            {
              choicePath.includes(location.pathname) &&
              <>
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: choicePath[2], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: choicePath[3], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: choicePath[4], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: choicePath[5], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: choicePath[6], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              </>
            }
            {
              adjustPath.includes(location.pathname) &&
              <>
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: adjustPath[0], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: adjustPath[1], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: adjustPath[2], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: adjustPath[3], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              <Breadcrumb.Item linkAs={NavLink} linkProps={{ to: adjustPath[4], activeStyle:{ color: headerTexts.activeNavElementColor }, className:"choicePathElement"
              </>
            }
          </Breadcrumb>
        </Nav>
      </Container>
    </Navbar>
  );
};

export default Header;
```

Kuva 6. Ylätunnisteen komponentti, sisältäen logon (linkki etusivulle), dynaamisen otsikon ja osiokohtaisesti ehdollisen leivänmuru-navigaation.

Ylätunnisteen tehtävänä sovelluksessa oli esitellä vasemmalta oikealle kolme eriteltyä kokonaisuutta: tilaajan logo (linkki etusivulle), sivunäkymän otsikko ja osiokohtainen navigaatio. Koska otsikon ja osiokohtaisen navigaation toteutus haluttiin pitää dynaamisena, päätettiin niiden muodostamiseen käyttää useLocation-tilakoukkua, jonka avulla sivunäkymän kulloinenkin osoite saatiin parsittua suoraan osoiteriviltä. Tätä arvoa voitiin otsikon osalta hyödyntää suoraan ja navigaation suhteen kahden osoitteen välillä toimivan ehtolauseen kautta. Osion mukainen navigointi voitiin renderöidä tunnisteelle käyttämällä toista React-Bootstrap-kirjaston valmista elementtiä, leivänmuru-navigaatio-komponenttia.

Alatunnisteen tehtävänä sovelluksessa oli esitellä tilaajan yhteystiedot ja yhteistyökumppanien logot sekä sovelluksen version. Tilaajan yhteystiedot sijoitettiin alatunnisteen vasempaan reunaan. Yhteistyökumppanien logot ja niihin liitetyt kotisivulinkit on listattu tunnisteen keskellä ja oikeassa reunassa esitetään vielä sovelluksen versionumero.

6.3 Valintaosio

Valintaosioon rakennettavat sivunäkymät olivat etusivu, valintasivu neljälle kategorialle sekä yhteenvetosivu kaikkien kategoriakohtaisten valintojen vahvistamiselle. Tämän osion tarkoituksena oli kartoittaa asiakkaan työllistymismahdollisuudet ja nykytila karkealla tasolla ja avata luontevasti sen tavoitteellisen parantamiseen tähtäävä keskustelu.

6.3.1 Etusivu

Osion käyttäminen, ja samalla koko sovelluksen käyttäminen, alkaa etusivulta, jossa kerrotaan lyhyesti sovelluksen tarkoitus ja annetaan ohjeet sen käyttämiseen. Sivulta siirrytään ohjatusti painikkeen kautta ensimmäiselle valintasivulle mutta ylätunnisteen navigoinnin kautta voidaan tarvittaessa heti hypätä myös muille kategoriasivuille aina yhteenvetosivulle saakka. Etusivulle voidaan myös palata koska tahansa käytön aikana ylätunnisteen logolinkin avulla.

6.3.2 Valintasivu

Seuraavaksi osion käyttämisessä siirrytään ensimmäisen valintasivun puolelle, josta liikutaan painikkeen avulla tai ylätunnisteen navigointia käyttämällä yksi kerrallaan neljä rakenteeltaan identtistä valintasivua läpi. Valintasivunäkymä on sovelluksen ensimmäinen uudelleenkäytettävä komponentti, jossa kaikki osion neljä aihekategoriaa rakennetaan parametrisoimalla sama valintasivunäkymä omien tilakoukkujensa ja teksti- ja kuvarakenteidensa kautta. Tällä tavoin voidaan kategorioiden välillä ylläpitää yhtenäistä tyyliä ilman, että samaa koodia pitää turhaan toistaa ja päivittää erikseen muutoksia tehtäessä.

Varsinaisen valintasivun näkymä on hyvin yksinkertainen, sisältäen vain valintataulualikomponentin ja ohjekuvan. Varsinainen valinta ja sen kautta tallennettu tila suoritetaankin valintataulukomponentin ja tämän sisältämän taulurivikomponentin sisällä.

Valintaosion toimintalogiikka perustuu käyttäjälle tarjottavan taulun esittelemiseen ja sen sisällä tehtyjen valintojen tallentamiseen. Valintataulu-komponentti vastaa tämän toimenpiteen suorittamisesta ja valintaa vastaavan arvon toimittamisesta takaisin päänäkymälle varastointia ja uudelleenkäyttöä varten.

```
// Valintataulun komponentti, jonka sisällä listataan kasa Taulurivejä (ChoiceTableItem)
return (
  <Container className="" fluid>
    <Col className="choiceSubtitle-green m-1">
      { props.subTitle_green }
      <OverlayTrigger trigger={['hover', 'focus']} placement={props.po_direction} overlay={popoverSub}>
        <Container id={props.c_class + "sub"} className="choiceQuestion" tabIndex="0"><Paragraph classes="choiceInfo" text={"?"} /></Container>
      </OverlayTrigger>
    </Col>
    <Col className="m-1">
      <ChoiceTableItem c_class={props.c_class} value={5} choice={props.choice} handleSelect={handleSelect} text={props.texts.text5} hint={ props
    </Col>
    <Col className="m-1">
      <ChoiceTableItem c_class={props.c_class} value={4} choice={props.choice} handleSelect={handleSelect} text={props.texts.text4} hint={ props
    </Col>
    <Col className="m-1">
      <ChoiceTableItem c_class={props.c_class} value={3} choice={props.choice} handleSelect={handleSelect} text={props.texts.text3} hint={ props
    </Col>
    <Col className="m-1">
      <ChoiceTableItem c_class={props.c_class} value={2} choice={props.choice} handleSelect={handleSelect} text={props.texts.text2} hint={ props
    </Col>
    <Col className="m-1">
      <ChoiceTableItem c_class={props.c_class} value={1} choice={props.choice} handleSelect={handleSelect} text={props.texts.text1} hint={ props
    </Col>
  </Container>
);
```

Kuva 7. Valintataulun komponentti, jossa esitellään otsikko (ja sen sisältämä ohjeistus) sekä taulurivin alikomponentit.

Valintataulun komponentti rakentuu otsikosta ja viidestä taulurivialikomponentista, jotka kaikki renderöidään yhtenä päällekkäisenä sarakkeena sivulle. Komponentin sisällä luodaan rakentamisen yhteydessä samalla funktio vastaamaan valinnan visuaalisesta vahvistamisesta ja tilan tallentamisesta. Tämä funktio reagoi enter-näppäimellä tai hiiren klikkauksella tehtävään valintaan taulurivialikomponenttien kohdalla ja tallentaa lähdettä vastaavan arvon päänäkymän alla säilytettävään tilaan. Visuaalinen vahvistus hoidetaan lisäämällä värikäs reunus valintakohteen elementin ympärille.

Taulurivialikomponentit ovat keskenään identtisiä ja uudelleenkäytettyjä komponentteja, joihin parametrien kautta toimitetaan valintataulun kautta syötetyt tekstit ja tilat sekä valinnan tekemisestä vastaava funktio. Funktion kutsuminen hoidetaan näin alikomponentin puolella, josta tilamuutos välitetään valintataulun kautta takaisin päänäkymälle.

6.3.3 Yhteenvetosivu

Osion toiminnallisuuden päättää yhteenvetosivu, johon siirrytään painikkeen avulla viimeisen valintasivun kautta tai suoraan ylätunnisteen navigaation kautta. Koska sovellus keskittyy hyvin henkilökohtaisiin ja vaativiin aiheisiin, nähtiin tarpeelliseksi toteuttaa osion loppuun sivu, jossa voidaan vielä käydä läpi sovelluksessa tehdyt valinnat ja muuttaa niitä, jos se nähdään tarpeelliseksi. Käyttäjä voikin yhteenvetosivulla vielä tarkistaa tekemänsä valinnat kokonaisuutena ja varmistaa, että ne todella vastaavat hänen nykyistä tilannettaan ennen siirtymistä välituloksen esittelyyn.

Yhteenvetosivu koostuu neljästä valintataulukomponentista, jota käytetään taas uudelleen samalla tapaa kuin valintasivujenkin puolella. Näihin valintatauluihin renderöidään nyt kaikkien neljän valintasivun aiheen valinnat ja alustetaan niiden arvoiksi tiloihin tallennetut arvot. Tällä tavoin käyttäjä näkee kerralla kaikki tekemänsä valinnat ja voi vielä vapaasti muuttaa niitä samaa logiikkaa hyödyntäen. Taulujen alla on lopuksi valinnat vahvistava painike, joka toimii ainoana ohjelmallisena siirtymänä osion tuloksen esittelyyn ja sitä kautta tarkennusosion puolelle.

6.4 Tarkennusosio

Tarkennusosioon rakennettavat sivunäkymät ovat tarkennussivu neljälle kategorialle ja tavoitesivu, jossa tarkennussivujen alla kirjattujen huomioiden joukosta valitaan itselle lopulta tärkeimmät tavoitteet. Tämän osion tarkoituksena on löytää konkreettisia vahvuuksia ja kehityskohteita jokaisen kategorian alta ja luoda sen kautta rakentavaa keskustelua niiden ympärille ja jalostaa niiden pohjalta lopulta tavoitteita, joihin pyrkimällä asiakkaan työllistymisedellytykset voisivat parantua.

6.4.1 Tarkennussivu

Osion käyttäminen alkaa ensimmäiseltä tarkennussivulta, johon siirrytään suoraan valintaosion välitulossivulla olevan painikkeen avulla. Siirtymisen yhteydessä ylätunnisteen navigaatio päivittyy samalla tarkennusosion puolelle ja mahdollistaa heti siirtymät kaikkien tarkennussivujen ja tavoitesivun välillä mutta estää samalla ohjelmallisen

palaamisen valintaosion puolelle. Siirtyminen tarkennusosion sivujen välillä onnistuu lisäksi etenemällä sivu kerrallaan painikkeiden avulla, samalla tapaa kuin valintaosiossakin.

Tarkennussivunäkymä on uudelleenkäytettävä komponentti, jonka kautta kaikki neljä tarkennussivua on toteutettu. Sivunäkymän rakenne on karkeasti kaksiosainen, jossa sivuston yläosan tehtävänä on ohjeistaa sivun käyttämisessä ja herättää ajatuksia ja tarjota vinkkejä niistä asioista, joita kyseinen kategoria voisi sisältää, ja sivuston alaosa puolestaan tarjoaa tekstilaatikon ja taulukot sen kautta syötettyjen huomioiden esittelemiseen. Jälkimmäinen osa-alue on luonnollisesti rakennettu kahden oman komponenttinsa, tekstinsyöttökomponentin ja tekstitaulukomponentin, kautta.

Tekstinsyöttökomponentti on rakennettu käyttäen perustana React-Bootsrapin Form-elementtiä, joka sisältää tyypillisimmät lomaketoiminnallisuudet valmiiksi muotoiltuna. Koska tarkoituksena on kerätä käyttäjältä kategoriakohtaisia vahvuuksia ja heikkouksia, usein lauseen mittaisina huomioina, valittiin monirivinen tekstilaatikko tähän toteutukseen sopivimmaksi kentäksi. Tekstilaatikon perään rakennettiin samalla kaksi painiketta, joiden kautta funktiota syötteen tallentamiseen voitiin kutsua, joko vahvuuden tai kehityskohteen puolesta.

```
// Tekstinsyöttökomponentti, jossa luodaan lomake ja sen tekstikenttä sekä napit syötteen tallentamiseen
return (
    <Form>
        <Row className="g-2">
            <Col md="10" >
                <Form.Group className="mb-2" controlId="form-group">
                    <FloatingLabel controlId="form-group" label={props.label}>
                        <Form.Control
                        as="textarea"
                        style={{ height: '85px', border:'solid 1px black' }}
                        onChange={(e) => setText(e.target.value)}
                        />
                    </FloatingLabel>
                </Form.Group>
            </Col>
            <Col md="2" className="" >
                <Row className="px-3">
                    <Button className="justify-content-center m-auto border-dark" variant="success" type="button" onClick={(e) =>
                </Row>
                <Row className="px-3 py-2">
                    <Button className="justify-content-center m-auto border-dark" variant="danger" type="button" onClick={(e) =>
                </Row>
            </Col>
        </Row>
    </Form>
);
```

Kuva 8. Tekstinsyötön komponentti, jossa käyttäjän kirjoittama vahvuus/kehityskohde tallennetaan tilaan.

Vahvuuksien ja kehityskohteiden kerääminen käyttäjältä useiden sivujen aikana ei kuitenkaan ole mielekästä, ellei tietoja voida esitellä ja poistaa käyttämisen aikana. Tätä tehtävää varten luotiin toinen komponentti, jossa luodut kohteet voidaan esitellä taulukkona tekstinsyötön alla ja tarjota mahdollisuus niiden hallintaan. Tekstitaulukomponentti rakennettiin syöttämällä sille parametrina samainen lista, johon tekstinsyöttökomponentin kautta lisättiin kohteita ja rakentamalla niistä dynaamisesti uusi listaelementti taulukkoon. Tämän elementin sisälle voitiin syöttää lisäksi poistosymboli, jonka kautta kyseinen kohde voitiin tarvittaessa klikkaamalla poistaa. Näin käyttäjä voi vapaasti lisätä kohteen, tarkastella sitä ja poistaa sen aina halutessaan sivulla ollessaan.

```
// Apufunktio, joka vastaa tekstisyötteen lisäämisestä
export const addText = (text, type, setter) => {
    if(type === '+') {
        setter(plusList => [...plusList, text])
    } else if (type === '-') {
        setter(minusList => [...minusList, text])
    } else {
        console.log('addText fail')
    }
}

// Apufunktio, joka vastaa tekstisyötteen poistamisesta
export const removeText = (type, removeIndex, setter) => {
    if(type === '+') {
        setter(plusList => plusList.filter((_val, index) => index !== removeIndex))
    } else if(type === '-') {
        setter(minusList => minusList.filter((_val, index) => index !== removeIndex))
    } else {
        console.log('removeText fail')
    }
}
```

Kuva 9. Tekstin lisäämisen ja poistamisen apufunktiot, joiden avulla voidaan operoida tekstinsyöttökomponentin kautta lisäyksiä ja tekstitaulukomponentin kautta poistoja.

6.4.2 Tavoitesivu

Tarkennussivujen puolelta kerätyt vahvuus- ja kehityskohteet ovat kaikki merkityksellisiä välineitä asiakkaiden tilanteen kartoittamisessa, mutta tilaajan kokemuksen mukaan konkreettisien tavoitteiden määrän rajoittaminen on ratkaisevaa niiden toteutumisen kohdalla. Kaikkien merkityksellisten seikkojen avaaminen keskustelun ja

reflektoinnin kautta nähtiin kuitenkin tilaajan osalta tärkeäksi osaksi mittarin käyttöä, joten ennen lopullisen tuloksen rakentamista, oli tarpeellisesta rakentaa logiikka tavoitteiden priorisoinnille. Tavoitesivun tehtävänä onkin esitellä kaikki käyttäjän tallentamat vahvuus- ja kehityskohteet kerralla ja antaa käyttäjän itse suorittaa priorisointi niiden kohteiden osalta, jotka hän kokee tärkeimmiksi tai saavutettavimmiksi tavoitteiksi itselleen.

Lyhyen ohjeistuksen lisäksi tavoitesivu esittelee käyttäjän tekemät valinnat taulumuodossa, josta käyttäjä voi enter-näppäimellä tai hiiren klikkauksella valita itselleen tärkeimmät kohteet henkilökohtaisiksi tavoitteikseen. Tämä logiikka toteutettiin sivun rakentamisen yhteydessä luodun funktion kautta, jossa elementin valinta joko lisää kohteen valittujen listalle tai poistaa kohteen uudelleen tehdyn valinnan yhteydessä.

```
const AdjustTotal = (props) => {
    // Tavoitesivun oma tila, jota käytetään tavoitevalintojen varastoimiseen
    const [selectedItems, setSelectedItems] = useState([])
    // funktio hallitesemaan tavoitteiden valintaa
    const addSelectedItem = (item) => {
        if (selectedItems.includes(item)) return setSelectedItems(selectedItems => selectedItems.filter(sItem => sItem !== item))
        if (selectedItems.length >= 5) return
        setSelectedItems([...selectedItems, item])
    }
    // Funktio hoitamaan valittujen tavoitteiden lähettäminen kun sivulta liikutaan
    const handleSubmit = () => {
        props.setTargetList(selectedItems)
    }
```

Kuva 10. Tavoitesivun tila ja funktiot tavoitteiden valintaan ja tallentamiseen.

6.5 Tulossivut

Sovelluksen molemmat osiot päättyvät omiin tulossivuihinsa, joiden tarkoituksena on kerätä osioiden alla tehdyt valinnat ja lisäykset ja esitellä ne käyttäjälle sekä tarjota mahdollisuus keskeyttää istunto tietoja menettämättä. Koska näiden sivujen alla ei tehdä enää uusia valintoja, on navigointi niille rajoitettu ohjelmallisesti vain yhteenvetosivun ja tavoitesivun nappisiirtymään. Näin estetään turhaan raskaiden kaavioiden ja pdf-lomakkeiden rakentaminen ilman relevanttia sisältöä ja varmistetaan samalla, että käyttäjä liikkuu harkiten osioiden välillä.

6.5.1 Välitulossivu

Välitulossivu päättää valintaosion ja esittelee sen puolella tehdyt valinnat käyttäjälle kaavioina sekä tarjoaa mahdollisuuden istunnon keskeyttämiseen ja tehtyjen valintojen tallentamiseen tai tulostamiseen. Mikäli tässä vaiheessa istunto keskeytetään, voi uuden istunnon aloittaa hyppäämällä suoraan yhteenvetosivulle ja täyttämällä edellisen istunnon valinnat suoraan sen taulukoihin. Näin sovelluksen käyttäminen pitkien istuntojen osalta hieman helpottuu. Session jatkuessa voidaan liikkua ohjelmallisesti eteenpäin vain painikkeen avulla ja sen kautta liikutaankin suoraan tarkennusosion ensimmäisen tavoitesivun puolella.

Välitulossivun toteutus on hyvin suoraviivainen, sivun keskellä on toteutettu React-chartjs-2-kirjaston avulla Radar-kaavio, jonka alle tuodaan valintaosion tiloihin tallennetut valinnat. Kaavion alle on toteutettu painikkeet pdf-lomakkeen tallentamiseksi ja tulostamiseksi sekä seuraavan osion puolelle siirtymiseksi.

```
// Välitulossivun rakenne, sisältäen radarkaavion sekä pdf-lomakkeen tallennuksen ja tulostuksen
const ChoiceReady = (props) => {
  const [image, setImage] = useState('')
  return (
    <Container className="content" fluid>
      <Row className="justify-content-center m-auto">
        <Col xs={6} className="">
          <Container className="choiceChart w-100">
              <RadarChart all={props.all} setImage={setImage} />
          </Container>
          {
            image !== '' &&
            <Container className="text-center" fluid>
              <PDFDownloadLink document={<Doc1 all={props.all} image={image} />} fileName="valintani.pdf" >{({ _blob, _url, loading, _error }) =>
                loading ? <Paragraph text="Lataa..." /> : <Button className="secButton w-25 m-1">{pdfButtons.download}</Button>
              }</PDFDownloadLink>
              <BlobProvider document={<Doc1 all={props.all} image={image} />} fileName="valintani.pdf" >{({ _blob, url, loading, _error }) =>
                loading ? <Paragraph text="Lataa..." /> : <Button onClick={() => printJS(url)} className="secButton w-25 m-1">{pdfButtons.print}</Button>
              }</BlobProvider>
            </Container>
          }
        </Col>
      </Row>
      <Row className="justify-content-center m-auto">
        <Link to={adjustPath[0]} className="btn mainButton m-2 col-3" >{choiceButtons.move}</Link>
      </Row>
    </Container>
```

Kuva 11. Välitulossivun komponentti, jossa esitellään kaavio ja pdf-lomakkeen toiminnot.

Pdf-lomakkeen tehtävänä välitulossivulla on toimia muistilappuna, joten sen rakenne on siten hyvin suoraviivainen. Otsikoiden ja tuloksista johdettujen arvioiden lisäksi varsinaisella sivunäkymällä esitelty Radar-kaavio tuodaan myös lomakkeen puolelle ja lopulta lomakenäkymä viimeistellään alaviitteen logoilla ja sovelluksen osoitteella.

6.5.2 Lopputulossivu

Lopputulossivu on sovelluksen viimeinen sivu ja esittelee lopputuloksen koko prosessista. Tarkennusosion keräämät tavoitteet voidaan nyt esitellä yhdessä valintaosion puolelta tuodun yleistilan kanssa ja tuottaa kokonaisvaltainen pdf-lomake itsearvioinnin tuloksista. Tämän lomakkeen tarkoituksena on toimia käyttäjälle omana muistilistana valituista kehityskohteista ja tavoitteista ja samalla tarjota tietoa prosessin edistymisestä, josta voi olla hyötyä tilaajan agentin ja käyttäjän välisissä jatkokeskusteluissa. Viimeisenä sivunäkymänä on myös tässä yhteydessä hyvä tarjota valinta istunnon tyhjentämiseen ja etusivulle palaamiseen sekä tilaajan toivomuksesta linkki ulkopuoliseen palautesivustoon, jossa tilaaja voi kerätä käyttökokemuksia sovelluksen käyttäjiltä.

Ulkoasultaan sivu mukailee välitulossivun rakennetta ja sisältää samalla tapaa kaavion tuloksen osalta ja painikkeet pdf-lomakkeen tallentamiseen ja tulostamiseen sen alla. Lopputuloksen kaavioksi valittiin Gauge-kaavio ja sen tuloksesta johdettu tekstikortti, joka esitellään kaavion päällä. Pdf-lomakkeen käsittelypainikkeiden alle on luotu lisäksi kaksi uutta painiketta, joista ensimmäinen tarjoaa mahdollisuuden istunnon tyhjentämiseen ja etusivulle palaamiseen hallitusti ja toinen ohjaa tilaajan omaan palautesovellukseen käyttäjäpalautteen antamiseksi. Koska kumpikaan toiminto ei kuulu sovelluksen normaaliin suorittamiseen, niiden osalta rakennettiin myös vahvistuksen pyytävä hälytysnäkymä, joka pyytää vielä vahvistamaan toiminnon suorittamisen.

```
{/* Lopputulossivun pdf-lomakkeen staattisia ohjetekstejä */}
<View style={styles.section} >
  <View  >
    <Text style={styles.subHeader}>{props.text[0]}</Text>
  </View>
  <View >
    <Text style={styles.text} >{props.text[1]}</Text>
  </View>
  <View >
    <Text style={styles.text} >{props.text[2]}</Text>
  </View>
  <View >
    <Text style={styles.text} >{props.text[3]}</Text>
  </View>
</View>
{/* Dynaamisesti tuotettu tekstikappaleiden lista käyttäjän tekemistä valinnoista */}
<View style={styles.section} >
  <View  >
    <Text style={styles.subHeader}>Tavoitteesi</Text>
  </View>
  {
    props.targetList.map((item, index) => {
      return <View><Text key={'T' + index} style={styles.text} >{item}</Text></View>
    })
  }
</View>
```

Kuva 12. Ote lopputulossivun pdf-lomakkeesta, joka sisältää staattisia tekstikappaleita ja dynaamisen listan käyttäjän valinnoista.

7 SOVELLUKSEN JULKAISEMINEN

Sovelluksen kehittämiseen valittiin ketterän ohjelmistokehityksen malli, jossa toteutus rakennetaan tiiviiden iteraatiokierroksien kautta julkaisemalla ja testaamalla jatkuvasti sovelluksen uusia ominaisuuksia. Tällaista prosessia ei voida tehokkaasti suorittaa ilman, että tilaajalla on mahdollisuus päästä testaamaan ja tarkastamaan ominaisuuksia koko ajan projektin edetessä ja siksi sovelluksen julkaiseminen mahdollisimman varhain oli tärkeä osa toteutuksen suunnittelua (Goto & Cotler, 2003, s. 189–190). Kätevä tapa tiiviin julkaisutahdin ylläpitämiseen GitHubin-sivuston alla ylläpidettävällä sovelluksella on rakentaa julkaisuputki suoraan GitHub-säiliön yhteyteen ja päivittää sovelluksen julkaisu aina kun säiliö itsessään päivitetään.

Julkaisuputki on vahvasti kaikkien versionhallintaohjelmien ja useimpien julkaisupalvelun tarjoajien tukema, joten sen käyttöönotto on verrattain triviaali operaatio kehittäjälle ja antaa runsaasti valinnanvaraa julkaisupalvelun tarjoajien suhteen. Tämän sovelluksen osalta julkaisun alustaksi valittiin Netlify.com, joka tarjoaa varsin kattavan ilmaisen julkaisualustan ja reilusti maksutonta päivitystä ja liikennettä sivulleen. Julkaisuputken rakentaminen Netlifyn alustalle on sekin erittäin helppoa ja onnistuu vain rekisteröimällä GitHubin alustalle ja sen kautta noudetun säiliön perustietojen ja julkaisukansion sekä halutun url-osoitteen valitsemista. Määrittelyjen jälkeen jokainen versiopäivitys GitHubiin laukaisee automaattisesti päivityksen myös Netlifyn puolelle ja käytännössä uusi versio sovelluksesta on käytettävissä vain muutamien minuuttien kuluttua päivityksen tekemisestä.

Koska tässä yhteydessä rakenteilla on testikäyttöön suunniteltu esittelysovellus, ei varsinaiselle tuotantoversiolle ole tarvetta suunnitella omaa julkaisua ja kehitysvaiheen aikana luotu julkaisu on riittävä palvelemaan läpi koko sovelluksen elinkaaren.

8 PROJEKTIN PÄÄTTÄMINEN

Projektisopimuksen ja toteuttamissuunnitelman yhteydessä sovittiin, että projekti päättyy, kun lopputarkastus sovelluksesta on suoritettu ja molemmat sopimusosapuolet toteavat sovelluksen valmistuneen. Tämä operaatio suoritettiin tilaajan edustajan ja toimittajan yhteisvoimin ja sovellus läpäisi sen yhteydessä sille annetut vaatimukset ja sovelluskehittämisprojekti voitiin samassa yhteydessä todeta päätettäväksi.

8.1 Dokumentointi ja käyttökoulutus

Päättymisen myötä valmistelut koodin luovuttamisesta voitiin aloittaa ja koska sopimukseen ei liitetty ylläpitovelvoitteita, samassa yhteydessä tuli tarpeelliseksi suorittaa lyhyt käyttökoulutus ja dokumentaatio mahdollisen jatkokehityksen varalta.

Ensimmäisenä askeleena projektin resurssikansiot siivottiin projektin aikana tarpeettomiksi jääneiden materiaalien osalta ja projektiympäristön dokumentaatio kirjoitettiin loppuun. Dokumentaation kautta tiedot projektiympäristön rakentamisesta, tärkeimmistä riippuvuuksista ja koodin jäsentelystä voidaan esittää näppärästi uusille kehittäjille tai oman muistin virkistyksenä, jos mahdollista jatkokehitystä vielä seuraa. Tästä syystä myös koodi käytiin tässä yhteydessä vielä kertaalleen läpi ja merkittävimmät kommentit päivitettiin ja lisättiin sen joukkoon tarpeen mukaan.

Seuraavaksi suoritettiin tilaajan edustajien lyhyt käyttökoulutus pienien muutoksien tekemiseen sovellukseen ja datan muokkauksiin, jotta ilman ylläpitoa luovutettavan sovelluksen käsittely edes jollain tapaa onnistuisi kehittämisen jälkeenkin.

Kun riittävä osaaminen sovelluksen muutoksista oli saavutettu ja projektiympäristö oli päivitetty lopulliseen muotoonsa, projekti päätettiin luovuttamalla koodi tilaajan haltuun.

8.2 Loppusanat

Itsearviointimittarin digitalisoiminen oli pitkä ja mielenkiintoinen projekti, jossa pääsin tutustumaan sekä React-ohjelmoinnin että projektinhallinnan syvempään olemukseen. Aikaisempaa kokemusta sovelluskehitysprojekteista asiakkaan kanssa ei ollut kertynyt kummallekaan osapuolelle, joten pääsimme yhdessä tilaajan kanssa harjoittelemaan yhteistyötä puhtaalta pöydältä. Yhteistyö sujui kokemuksen puutteesta huolimatta hienosti läpi koko projektin ja kehitimme hyvin nopeasti yhteisen vision siitä, miltä sovelluksen piti näyttää ja toimia. Tämän vision rakentaminen React-kirjastoa käyttäen tuntui alusta alkaen oikealta valinnalta arkkitehtuurin suhteen ja kyvyt kirjaston käyttämiseen paranivat nekin koko ajan projektin edetessä.

Rakennettu sovellus on edelleen käytössä tilaajan organisaatiossa ja sen osalta saatu asiakaspalaute on ollut pääosin positiivista. Tilaajan tavoite päästä testaamaan sovelluksen käyttämistä onnistui, ja lopputulokseen oltiin erittäin tyytyväisiä, sovellus oli tarkoitukseensa sopiva ja projekti valmistui sille sovitun budjetin ja aikataulun puitteissa. Näin ollen voidaan todeta, että projekti osoittautui kaikin puolin onnistuneeksi.

LÄHTEET

Brewer J. & Dittman K. (2018). Methods of IT Project Management: Third Edition. Purdue University Press.

Cassio de Sousa, A. (2015). Pro React. Apress.

Create-React-App dokumentaatio. (2022). Haettu 11.01.2022 osoitteesta https://create-react-app.dev.

GitHub. (2022). Haettu 11.01.2022 osoitteesta https://github.com.

Goto, K. & Cotler E. (2003). Verkkopalveluprojekti. Edita Publishing Oy.

Mozilla Developer Network. (2022a). Haettu 11.01.2022 osoitteesta https://developer.mozilla.org/en-US/.

Ogidan, B. (2021). Structuring Your React Application – Atomic Design Principles. Haettu 08.01.2022 osoitteesta https://andela.com/insights/structuring-your-react-application-atomic-design-principles.

React dokumentaatio. (2022). Haettu 11.01.2022 osoitteesta https://reactjs.org.

React-Bootstrap dokumentaatio. (2022). Haettu 11.01.2022 osoitteesta https://react-bootstrap.github.io.

W3schools CSS verkkosivujen muotoilu. (2022). Haettu 11.01.2022 osoitteesta https://www.w3schools.com/css/css_website_layout.asp.

Usability.gov. (2022). Use Cases. Haettu 11.01.2022 osoitteesta https://www.usability.gov/how-to-and-tools/methods/use-cases.html.

Unhelkar, B. (2012). The Art of Agile Practice. Auerbach Publishers, Incorporated.

LIITE 1

MÄÄRITTELYT

Tämä projektisuunnitelma on tarkoitettu liitteeksi opinnäytetyönä tehtävään sovellusprojektiin, jossa anonyymi hanke (myöhemmin Tilaaja) tilaa töissään käyttämänsä paperisen mittarityökalun digitaalisoinnin Satakunnan Ammattikorkeakoulun opiskelija Matti Tammiselta (myöhemmin Toimittaja). Projektin lopputuloksena tuotetaan Tilaajalle itsenäisesti testattavissa oleva verkossa jaettu sovellus, jonka kautta digitaalisen mittarin käyttämistä hankkeen töissä voidaan koekäyttää ja sen sopivuutta siten arvioida.

Projektin vaiheet:

- **Projektin aloittaminen –** Projekti aloitetaan hyväksymällä tämä projektisuunnitelma toteuttamissuunnitelmaksi ja allekirjoittamalla opinnäytetyösopimus. Aloittamisen yhteydessä jaetaan tarvittavat yhteystiedot, sovitaan tarkemmin säännöt yhteydenpidosta sekä esitellään ja testataan raportointiin ja yhteydenpitoon käytettävä ohjelmisto.
- **Projektin läpivienti -** Projekti toteutetaan ketterien ohjelmistokehitysmallien toimintaperiaatteen mukaisesti, määrittelemällä ensin seuraavaksi toteutettava ominaisuus, sen toteuttamisen aikataulu ja siihen tarvittavat resurssit. Tarkemmin kohdassa Läpivienti.
- **Projektin aikataulu -** Projekti suoritetaan sopimuspäivästä tasan kahden kuukauden aikana. Mikäli tässä ajassa vähintään alkumäärittelyä vastaava sovellus ei ole Tilaajan testattavassa käytössä, Toimittaja sitoutuu toteuttamaan alkumäärittelyä vastaavan sovelluksen ja toimittamaan sen Tilaajalle kohtuullisessa ajassa.
- **Projektin resurssit –** Toimittaja vastaa itse sovelluksen kehittämiseen liittyvien laitteiden ja ohjelmien järjestämisestä. Sovellus pyritään myös ensisijaisesti rakentamaan ilman maksullisia ominaisuuksia ja palveluja. Mikäli kuitenkin jokin maksullinen ominaisuus tai palvelu katsotaan sovelluksen kannalta oleelliseksi, voidaan sen käyttämisestä sopia erikseen kirjallisesti.
- **Projektin päättäminen –** Projekti päätetään, kun yhteisesti on todettu sen vastaavan sille annettuja määritteitä. Projekti päättyy kuitenkin viimeistään projektin suoritusajan päättyessä, kun vähintään sen alkumäärittely ovat täytetty.

PROJEKTISUUNNITELMA
Itsearviontimittarin digitalisoiminen

LÄPIVIENTI

Projekti toteutetaan ketterien ohjelmistokehitysmallien toimintaperiaatteiden mukaisesti, luomalla jokaisesta toteutuksen vaiheesta oman määrittelyn, esittelyn, rakentamisen ja raportoinnin sisältämä iteraatio. Jokaisen iteraation määrittely tehdään yhteisesti ja sovitaan sen aikataulusta ja rakentamisen aikaisesta raportoinnista. Kun määrittelyt sisältävä toteutus on valmis, se esitellään ja sovitaan uuden iteraation määrityksistä ja aikataulusta. Tätä sykliä jatketaan, kunnes projekti päättyy.

Ennen ensimmäisen kehityssyklin alkua:

- Sovitaan yhdessä sovelluksen alkumäärittely, eli mitä sovellus vähintäänkin tulee pitää sisällään, jotta se palvelee Tilaajan tarkoitusta. Alkumäärittely lisätään liitteeksi tähän suunnitelmaan.

Ohjelmiston kehityssykli (iteraatio):

- Määritetään yhdessä seuraava ominaisuus, joka sovellukseen rakennetaan.
    - Miksi ominaisuus rakennetaan?
    - Miten ominaisuus tarkalleen toteutetaan?
    - Miten ominaisuuden voidaan todeta onnistuneen?
- Sovitaan rakentamiseen käytetyistä resursseista.
    - Käytetäänkö maksullisia ominaisuuksia tai palveluja?
    - Tilaajan osalta syklin käyttöön annetut resurssit, kuten työkalut ja henkilöt.
    - Miten eteneminen raportoidaan, palaverit ja kokoukset.
- Ominaisuuden rakentaminen.
- Toteutuksen esittely.
    - Testaukset.
    - Onnistuiko ominaisuuden toteuttaminen, mahdolliset virheet ja puutteet.

OIKEUDET JA VELVOLLISUUDET

Tilaaja:

- Ensisijainen velvollisuus on toimittaa riittävän tarkka ja realistinen määrittely toteutettavan sovelluksen toiminallisuuksista ja sen käyttämisestä.
- Koska projekti suoritetaan palkattomana opinnäytetyönä, Tilaaja vastaa siitä, että sen toteuttaminen ei aiheuta rahallisia kuluja tai kohtuutonta ajallista kulua Tekijälle.
- Tilaaja vastaa siitä, että Toimittajan tarvitsemat lisätiedot ja tarkennukset voidaan tarjota kohtuullisessa ajassa niiden pyytämisestä ja niitä kohtaan vaadittavalla tarkkuudella.
- Kaikkia Tilaajan tietoja, resursseja ja välineitä käsitellään luottamuksellisesti koko projektin ajan ja sen jälkeen, myös opinnäytetyö huomioiden.
- Tilaaja on oikeutettu saamaan vähintään alkumäärittelyä vastaava toimiva sovellus ja sen lähdekoodi Toimittajalta aikataulun puitteissa tai viimeistään kohtuullisessa ajassa sen jälkeen.

Toimittaja:

- Ensisijainen velvollisuus on rakentaa vähintään alkumäärittelyä vastaava ja käyttöönsä mahdollisimman soveltuva sovellus aikataulun puitteissa tai viimeistään kohtuullisessa ajassa sen jälkeen ja toimittaa se lähdekoodeineen Tilaajalle.
- Viivästyksien osalta tarkemmin Siirtäminen ja Päättäminen.
- Toimittaja vastaa tyypillisen dokumentointitason järjestämisestä sovelluksen yhteyteen sekä sen käsittelyn ja jakamisen perustasoisen opastuksen järjestämisestä projektin päättämisen yhteydessä.
- Toimittaja vastaa koodin luettavuuden, muotoilun ja kommentoinnin järjestämisestä keskimääräisen hyvän ohjelmointitavan mukaisesti.
- Toimittaja ei vastaa ylläpidosta, tuesta tai toimivuudesta luovuttamisen jälkeen.
- Toimittaja on oikeutettu saamaan riittävän tarkan ja realistisen määrityksen toteuttavan sovelluksen toiminnallisuuksista ja sen käyttämisestä kohtuullisessa ajassa sen pyytämisen jälkeen.

SIIRTÄMINEN JA PÄÄTTÄMINEN

Projektin päättyminen:

- Projekti päättyy luonnollisesti, kun viimeisen kehityssyklin esittelyvaiheessa voidaan yhdessä todeta, että sovellus vastaa sille annettuja vaatimuksia toimintansa suhteen ja sovelluksen koekäyttö voidaan Tilaajan puolesta aloittaa. Projekti päättyy myös siinä tapauksessa, että aikataulu projektin osalta on ylitetty ja projekti on todettu yhteisesti jossain vaiheessa täyttävän sille annetut alkumäärittelyt.
- Mikäli projekti ei aikataulunsa puitteissa ole vielä täyttänyt sille annettuja alkumäärittelyjä, sovitaan aikataulun lopussa kohtuullinen jatkoaika sovelluksen toimittamiselle. Ellei jatkoajan jälkeenkään ole luovutettavissa alkumäärittelyä täyttävää sovellusta mutta kohtuullinen työmäärä sen toteuttamiseksi Toimittajan osalta voidaan todeta täyttyneen, toimitetaan sovellus ja sen lähdekoodi siinä kunnossa mitä se on Tilaajalle ja sopimuksen katsotaan päättyneen.
- Projektin päättymisen yhteydessä kaikkia projektiin osallistuneita sekä tarvittavia sidosryhmiä informoidaan projektin päättymisestä.
- Päättymisen jälkeen Toimittaja toimittaa sovelluksen lähdekoodin Tilaajalle sekä sovelluksen jakaminen ja sisältö käydään läpi. Lähdekoodi on yhteisomisteinen, eli kummatkin osapuolet saavat sen käyttöönsä (pl. luottamukselliset tiedot).

Projektin siirtäminen:

- Projekti voidaan siirtää yhteisymmärryksessä myöhemmäksi, jos sen toteuttaminen normaalin aikataulun puitteissa aiheuttaa osapuolille kohtuutonta haittaa. Siirtämisestä sovitaan erikseen kirjallisesti.
- Mikäli projektin siirtäminen lisää sovelluksen työrasitetta, on se huomioitava lieventävästi alkumäärittelyjen täyttämisen suhteen.

Muuta:

- Mikäli projektiaikataulun suhteen tulee epäselvyyttä, pidetään opinnäytetyösopimuksen allekirjoituspäivää projektin aloituspäivänä ja tasan kahden kuukauden jälkeistä päivää projektin aikataulun päätöspäivänä.
- Kohtuullisena työmääränä sovelluksen kehittämiseen pidetään tässä yhteydessä 50h.

LIITE 2


Itsearviointimittarin digitalisoiminen
Alkumäärittely
Sovellus on testattavissa paikallisesti valittujen agenttien toimesta tai julkaistuna verkkosovelluksena.
Sovellus sisältää ainakin etusivun, valintasivun, tarkennussivun ja tulossivun.
Sivujen ja osa-alueiden välillä on pystyttävä liikkumaan loogisesti.
Yksittäisten sivujen rakenteessa on nähtävissä hankkeen logo sekä rahoittajien logot.
Kaikki toimintalogiikka ohjeistetaan käyttäjälle.
Valinnat voidaan esitellä yhtenäisenä tuloksena kaavion tai kuvan muodossa.
Sovellus tuottaa loogisen tuloksen, jota voidaan jatkojalostaa.