Tampereen ammattikorkeakoulu

Web-sovelluksen visuaalinen päivitys

Osaamispankki

Eemil Väänänen

OPINNÄYTETYÖ
Huhtikuu 2023

Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotanto

VÄÄNÄNEN, EEMIL:
Web-sovelluksen visuaalinen päivitys
Osaamispankki

Opinnäytetyö 38 sivua, joista liitteitä 0 sivua
Huhtikuu 2023

Tämän opinnäytetyön tarkoituksena oli toteuttaa ja dokumentoida Netum Oy:n sisäisen Osaamispankki-web-sovelluksen käyttöliittymän päivittäminen. Osaamispankin tarkoituksena on toimia Netumin työntekijöiden tietopankkina. Siellä säilytetään työntekijöiden sinne tallentamia tietoja heidän taidoistaan, työkokemuksistaan ja koulutuksistaan. Sitä käytetään pääasiallisesti tukemaan myynnin työtä.

Osaamispankista on tehty jo useampi versio. Ensimmäisenä toteutettiin Microsoftin PowerApps -palveluun pohjatuva versio, joka ei kuitenkaan ollut haluttuun tarkoitukseen kovinkaan hyvin soveltuva. Tämän jälkeen tehtiin versio talon sisällä käyttäen nykyaikaisia ja moderneja verkkokehitystekniikoita. Tähän valitut tekniikat olivat React, TypeScript ja .NET Core. Tässä opinnäytetyössä puhutaan tämän viimeisimmän version päälle rakennetusta päivitetystä versiosta, jossa tarkoituksena on ollut suoraviivaistaa ja yhtenäistää käyttöliittymä, päivittää käytössä olevat kirjastot sekä päivittää käyttökokemusta yleisesti. Käyttöliittymäkomponenttikirjastoksi valittiin tässä tapauksessa Material UI.

Hyvin toteutetun käyttöliittymäsuunnittelun integrointi sovelluksen kehitykseen on todella hyödyllistä. Se nopeuttaa sovelluksen kehittämistä huomattavasti, varmistaa käyttöliittymän yhtenäisen ulkonäön sekä se avustaa mahdollisten rajatapausten ja pullonkaulojen havaitsemisessa. Suunnitteluun täytyy kuitenkin panostaa resursseja ja aikaa. Huonosti toteutettu suunnittelujärjestelmä aiheuttaa vain epätietoisuutta kehittäjien keskuudessa ja hidastaa kehittämistä.

Asiasanat: react, typescript, material ui, javascript, jatkokehitys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Business Information Systems
Bachelor of Business Administration

VÄÄNÄNEN, EEMIL:
Visual Update of a Web Application
Osaamispankki

Bachelor's thesis 38 pages, appendices 0 pages
April 2023

The purpose of this thesis was to document the updating of the user interface of Netum OYs internal web application called Osaamispankki. Osaamispankki is Netums internal competence management system. It allows the employees of Netum to document their knowledge, skills, work experience and educations. The system is mostly used by the sales team.

The first version of Osaamispankki was based on PowerApps, which had technologies that were not suitable for the purpose of this app. That is why it was decided that the application was to be rebuilt from the ground up by a team inside Netum. Technologies for this release were chosen based on their modernity, efficiency, and maintainability. React, TypeScript and .NET Core were chosen as the technologies. This thesis is about an update that was built upon that version of Osaamispankki. It was meant to integrate and streamline the user interface, update all the libraries used, and update the user experience on more general level. Material UI was selected as the component library.

Integrating well-executed user interface design into application development is highly beneficial. It speeds up the development process, ensures a consistent appearance of the user interface, and assists in identifying any edge cases or bottlenecks. However, investing resources and time into the design process is necessary. Poorly executed design systems only cause uncertainty among developers and slow down the development process.

Key words: react, typescript, material ui, javascript, further development

SISÄLLYS

LYHENTEET JA TERMIT

JavaScript	Web-sovelluksien kehittämiseen käytettävä ohjelmointi-/komentosarjakieli
Material UI	Material UI on käyttöliittymäkomponenttikirjasto, jonka avulla voidaan luoda käyttöliittymiä nopeasti
Microsoft AD	Microsoftin Active Directory (AD) tarjoaa organisaatiossa keskitetyn hallinnan käyttäjätileille, tietokoneille, sovelluksille ja verkkoresursseille
Microsoft Azure	Microsoftin pilvipalvelualusta
MSSQL	Microsoftin kehittämä relaatiotietokantojen hallintajärjestelmä
React	Facebookin (nyk. Meta) kehittämä JavaScriptin päälle rakennettu käyttöliittymien valmistamiseen tehty JavaScript-kirjasto
REST	REST (Representational State Transfer) on malli, joka kuvaa resurssien hallintaa verkkoympäristössä
TypeScript	Microsoftin kehittämä, avoimeen lähdekoodiin perustuva ohjelmointikieli. Merkittävin hyöty TypeScriptin käytöstä on vahva staattinen tyypittäminen JavaScriptin päälle

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on esitellä jo valmiin web-sovelluksen käyttöliittymän uudelleenrakentaminen sekä käyttöliittymäsuunnittelun hyödyt ja haitat. Käytännössä tällä kertaa ei siis rakenneta koko sovellusta uudelleen, vaan keskitytään käyttöliittymän uudistamiseen. Tämä tarkoittaa sitä, että rajapintojen tai niiden kutsujen suunnittelu sekä mahdollinen päivittäminen on minimaalista.

Hyvin toimiva käyttöliittymä on käyttäjäkokemuksen tärkein osa. Tämän takia on tärkeää, että käyttöliittymä on hyvin suunniteltu ja sen johdonmukaisuuteen on käytetty resursseja. Tavoitetilana on, että käyttäjän ei missään vaiheessa tarvitsisi miettiä, mitä hän tekee ja miksi tai että miten hän pääsee tekemään tiettyä asiaa. Siksi Osaamispankille rakennettiin Figma-sovellusta käyttäen suunnittelujärjestelmä (engl. Design System) jotta varmistutaan yhtenäisistä näkymistä ja komponenteista.

Raportin toisessa luvussa käydään läpi Osaamispankissa käytettyjä teknologioita sekä miten niitä on hyödynnetty sovelluksen kehityksessä. Kolmannessa luvussa tutustutaan Osaamispankkiin. Alkuun käydään läpi sovelluksen historiaa tarkemmin. Sen jälkeen verrataan vanhan ja uuden version eri alasivujen ja komponenttien eroja. Käydään tarkemmin läpi mikä on muuttunut ja miksi tämä muutos on toteutettu. Näiden jälkeen tutustutaan vielä Material UI komponenttien muokkaamiseen sekä lopuksi vielä pohdintaa työstä ja sen toteutuksesta.

2 TEKNOLOGINEN TOTEUTUS

2.1 JavaScript

JavaScript on yleiskäyttöinen ohjelmointikieli, jonka pääasiallinen käyttötarkoitus on verkkosivujen interaktiivisten ominaisuuksien luominen. Ensimmäinen JavaScriptin versio julkaistiin vuonna 1995 osana Netscape Navigator 2.0 -selainta. (Negrino & Smith 2012.)

Nykyisin JavaScriptiä ylläpidetään Ecma International -standardointijärjestön ja useiden verkkoselaimia kehittävien teknologiayritysten, kuten Microsoftin, Googlen ja Mozilla Foundationin, kanssa. Tämä yhteistyö varmistaa JavaScriptin modernien versioiden yhteensopivuuden useiden erilaisten käyttöjärjestelmien ja verkkoselainten kanssa. (Negrino & Smith 2012.)

2.2 Typescript

TypeScript on Microsoftin kehittämä avoimen lähdekoodin ohjelmointikieli. TypeScriptiä kutsutaan myös JavaScriptin ylijoukkokieleksi, mikä tarkoittaa, että kaikki JavaScript-koodi on myös täysin kelvollista TypeScript-koodia. TypeScriptin ensimmäinen versio julkaistiin vuonna 2012. TypeScript on ollut Microsoftin ja avoimen lähdekoodiyhteisön aktiivisessa kehityksessä siitä eteenpäin. (Goldberg 2022.)

TypeScriptin merkittävimmät hyödyt ovat sen tarjoama tyypitys ja myös lisäominaisuudet, kuten natiivit rajapinnat ja tuki luetelluille tyypeille. TypeScript kääntyy ajettaessa JavaScriptiksi, joten se on täysin yhteensopiva JavaScript-koodin ja -kirjastojen kanssa. (Goldberg 2022.)

TypeScript on tärkeä osa Osaamispankin kehitystä. Kaikki Osaamispankin lähdekoodin luokat ja komponentit ovat rajapinnoitettuja ja tyypitettyjä. Näin vähennetään virheitä ja saadaan merkittävä osa ohjelman suorituksen ennenaikaisesti keskeyttävistä virheistä havaittua ja korjattua jo ennen lähdekoodin ajoa.

2.3 React

React on Facebookin (nyk. Meta) kehittämä JavaScriptin kirjasto. Sen ensimmäinen julkinen versio julkaistiin vuoden 2013 toukokuun lopussa. Reactin lähdekoodi on avoin, ja sen kehityksestä vastaa Meta ja laaja avoimen lähdekoodin yhteisö. (Gackenheimer 2015.)

Vaikka React tarjoaa monipuolisesti ominaisuuksia, kuten tilan hallintaa ja komponentteja, niin sen käyttöön tarvitaan usein myös muita kirjastoja. Jos React sovellukseen halutaan lisätä esimerkiksi reititystoimintoja, voidaan käyttää kolmannen osapuolen kirjastoja, kuten React Routeria. Mikäli sovelluksessa on tarve mukautetuille toiminnoille tai komponenteille, niitä voidaan kehittää myös itse. (Gackenheimer 2015.)

Osaamispankissa käytetään Reactia selainpuolen käyttöliittymäkomponenttien luomiseen ja piirtämiseen. Näiden lisäksi myös tilan hallinta on tietyiltä osin toteutettu käyttäen Reactista löytyviä ominaisuuksia.

2.4 Material UI

Material UI on avoimen lähdekoodin React-komponenttikirjasto, joka toteuttaa Googlen Material Design -suunnitteluperiaatteet. Se sisältää suuren määrän valmiiksi suunniteltuja ja toteutettuja komponentteja, joita pystyy kuitenkin vielä muokkaamaan ja jatkokehittämään käyttäjän haluamalla tavalla. (Overview – Material UI n.d.)

Material UI:n ensimmäinen versio julkaistiin vuonna 2014, ja sen tavoitteena oli yhdistää React ja Material Design -suunnitteluperiaatteet. Nykyään Material UI:ta käyttää yli kaksi miljoonaa kehittäjää ympäri maailmaa. (Overview – Material UI n.d.)

Material UI on valittu Osaamispankin komponenttikirjastoksi, eli lähes kaikkien Osaamispankissa käytössä olevien komponenttien taustalla on Material UI:n vastaava komponentti. Näitä komponentteja on kuitenkin jatkokehitetty merkittävästi haluttujen toiminnallisuuksien ja visuaalisten ulkonäköjen saavuttamiseksi.

2.5 Figma

Figma on suunnitteluun ja prototyyppien toteuttamiseen kehitetty selainsovellus. Sen ensimmäinen versio julkaistiin vuonna 2016. Figman suosio on kasvanut lähes räjähdysmäisesti sen helppokäyttöisyyden ja selainpohjaisuuden sekä sovelluksen tarjoaman usean käyttäjän samanaikaisen työskentelyn tuen ansiosta. (What is Figma n.d.)

Figmassa on tarjolla Material UI:n tarjoama komponenttikirjasto, jonka perusteella Osaamispankin UI/UX-suunnittelijat suunnittelivat kaikki Osaamispankkiin käyttöön tulevaksi tarkoitetut komponentit. Myös kaikkien Osaamispankin eri sivujen pohjapiirustukset ovat suunniteltuina ja piirrettyinä Figmassa. Näiden visuaalisten piirustusten pohjalta on luotu kaikki käytössä olevat komponentit ja sivujen visuaaliset ulkonääöt.

2.6 C# / .NET Core

C# on moderni, olio-orientoitunut ohjelmointikieli. Sen on kehittänyt Microsoft, ja sitä käytetään laajalti tietokantaohjelmoinnissa sekä web-palvelinten kehittämisessä. C# on osa .NET-ohjelmistokehystä, joka sisältää kokoelman kirjastoja ja työkaluja kehittäjille. (A tour of the C# language 2023.)

.NET Core on myös Microsoftin kehittämä, mutta avoimeen lähdekoodiin perustuva sovelluskehys. Se mahdollistaa sovelluksen kehittämisen monille eri alustoille. .NET Core on uudempi ja kevyempi versio alkuperäisestä .NET Framework -sovelluskehyksestä. (What is .NET 2023.) Osaamispankin taustajärjestelmät ja rajapinnat on kirjoitettu käyttäen C#-kieltä ja .NET Core-sovelluskehystä.

2.7 MSSQL

Microsoft SQL Server (MSSQL) on Microsoftin kehittämä tietokantojen hallintajärjestelmä. Pääasiallisesti sitä käytetään relaatiotietokantojen hallintaan. Sen etuna on helppokäyttöisyys, jonka ansiosta siitä on tullut vuosien saatossa erittäin suosittu. (What is Microsoft SQL Server n.d.)

MSSQL-järjestelmiä käytetään Osaamispankin tietokantojen hallintaan ja ylläpitämiseen. Koska Osaamispankin tietokanta on myös Microsoftin tarjoamalla Azure alustalla, on sen käyttö MSSQL-järjestelmiä käyttäen helppoa ja toimivaa.

2.8 Azure / Azure AD / MSAL

Azure on Microsoftin kehittämä pilvipalvelualusta. Sen tarjontaan kuuluu infrastruktuuri-, palvelinalusta, ja ohjelmistopalveluita. (How does Azure work 2023.)

Azure Active Directory (Azure AD) on Microsoftin kehittämä pilvipohjainen identiteetti- ja oikeuksienhallintajärjestelmä. Azure AD:n avulla työnantaja pystyy jakamaan työntekijöilleen pääsyoikeuksia ohjelmistoihin ja työpaikan intranetin sisäisiin resursseihin, kuten työpaikan pilvipohjaisiin sovelluksiin. (What is Azure Active Directory 2023.)

Microsoft Authentication Libraryn (MSAL) avulla pystytään hakemaan security tokeneita tunnistamalla käyttäjä Azure AD:n tietoja vasten. Näillä tokeneilla saadaan suojattu yhteys mm. Microsoft Graph -järjestelmiin. Näitä tokeneita voidaan käyttää hyödyksi myös omien rajapintojen suojauksessa. (Overview of the Microsoft Authentication Library 2023.)

Osaamispankki käyttää Microsoft Azuren pilvikomponentteja. Taustajärjestelmät ovat Azuren app service -palvelussa. Tietokanta on myös Azuren tarjoama Azure SQL. Kirjautumiseen ja käyttäjän autentikointiin käytetään Azure AD:ssa olevia tietoja ja selainpuolella MSALia. Sovelluksen taustajärjestelmä pystyy myös lähettämään sähköposteja Azure AD:ta hyödyntäen.

3 OSAAMISPANKKI

3.1 Osaamispankki yleisesti

Netum Oy kehitti syksyllä 2018 sisäisen osaamisenhallintajärjestelmän nimeltä Osaamispankki. Idea ja kehittäminen lähti siitä, että Netumin henkilöstöratkaisut huomasivat tarpeen osaamiskartoitukselle, jotta he voisivat selvittää, millaista osaamista Netumilta löytyy. Osaamispankin tarkoituksena on helpottaa työntekijöiden arkea ja auttaa myyntiä löytämään osaajia projektitarjouksiin, tunnistamaan henkilöstön vahvuuksia ja kehityskohteita sekä tukemaan osaamisen kehittämistä.

Osaamispankin avulla säilytetään Netumilla työskentelevien henkilöiden tiedot, taidot ja työkokemus. Järjestelmään on luotu jokaiselle Netumin työntekijälle oma profiili, jota henkilö voi täydentää. Täten saadaan tarkempi käsitys henkilön osaamisesta, kiinnostuksista ja työkokemuksesta.

Koska markkinoilta ei löytynyt sopivaa tuotetta, tehtiin Netumilla päätös, että Osaamispankki toteutetaan itse. Ensimmäinen versio Osaamispankista toteutettiin käyttäen Microsoftin Power Apps -palvelua, mutta sen jatkokehittäminen ja haluttujen ominaisuuksien lisääminen todettiin liki mahdottomaksi. Täten tehtiin päätös rakentaa sovellus uudestaan täysin talon sisäisesti. Näin pystyttiin hallitsemaan mitä ominaisuuksia haluttiin kehitettäväksi ja miten sekä varmistettiin, että sovelluksen jatkokehitys on mahdollista.

3.2 Vanhan ja uuden version erot

3.2.1 Yleistason muutokset

Merkittävin koko sovelluksen toimintaan vaikuttava muutos on komponenttikirjaston vaihtaminen Semantic UI:sta Material UI:hin. Syy tälle muutokselle oli siinä,

että Material UI:n komponenttien muokkaus on suoraviivaisempaa ja yksinkertaisempaa. Material UI on myös huomattavasti suositumpi, eli ongelmatilanteissa oli helpompi löytää apua ja esimerkkiratkaisuja.

Muuten pyrittiin käyttämään mahdollisimman samoja JavaScript-kirjastoja tukemaan kehittämistä. Näitä kuitenkin, mahdollisuuksien mukaan, päivitettiin uusimpiin versioon. Kaikissa kirjastoissa tämä ei ollut kuitenkaan mahdollista, joten niiden tilalle pyrittiin kehittämään vaihtoehtoisia ratkaisuja. Täysin uusien kirjastojen lisäämistä vältettiin ja samalla myös pyrittiin vähentämään turhiksi jääneiden kirjastojen määrää.

Suunnittelutasolla käyttöliittymän suunnitteluun saatiin apuja design-tiimiltä. Suunnittelijoiden tehtävänä oli luoda Osaamispankille suunnittelujärjestelmä Figma-sovelluksen avulla. Suunnittelujärjestelmällä varmistetaan yhtenäinen näkymä komponenttien ja sivujen ulkonäölle. Näiden määrittelyiden avulla oli huomattavasti helpompi luoda halutun näköisiä komponentteja ja sivujen näkymiä.

Monilla alasivuilla koettiin ongelmalliseksi se, että sivulta saattoi löytyä yli tuhat listausta ja niiden näyttäminen samaan aikaan johti huonoon käyttäjäkokemukseen. Vanhoilla tehottomilla koneilla se saattoi jopa hidastaa selaimen toimintaa. Tähän ratkaisuksi kehitettiin listauksien rajoittaminen näyttämään alkuun vain kymmenen ensimmäistä tulosta. Sen alle rakennettiin uudelleenkäytettävä komponentti, josta piirtyy nappula, jota painamalla käyttäjä voi ladata kymmenen tulosta lisää. Näin käyttäjä ei koe isoa informaatiotulvaa, käyttöliittymän toiminta pysyy sulavana ja käyttäjä itse pystyy hallitsemaan näkemänsä tiedon määrää.

3.2.2 Kirjautuminen / käyttäjän autentikointi

Käyttäjälle näkyvä toteutus pysyy hyvin yhtenäisenä vanhan toteutuksen kanssa. Käyttäjän todentaminen tapahtuu Microsoftin henkilökohtaisilla AD-tunnuksilla. Näillä tunnuksilla saadaan autentikoinnissa tarvittavat käyttäjän tiedot, joita hyödyntäen pystytään rajaamaan esimerkiksi käyttäjän pääsyä sovelluksen eri osa-alueille.

Taustalla on kuitenkin toteutettu muutoksia toimintalogiikkaan. Vanhassa sovelluksessa käytettiin Azure AD Authentication Librarya (ADAL), tarkennettuna vielä sen integrointiin soveltuvaa react-adal-nimistä JavaScript-kirjastoa. Se toimi tarkoituksessaan hyvin Microsoftin Azure AD:n kanssa, mutta Microsoft ilmoitti jo kesäkuussa 2020, että ADAL-järjestelmät eivät enää saa uusia ominaisuuksia. Vuoden 2022 joulukuussa Microsoft ilmoitti, että ADAL-järjestelmien tuki loppuu kokonaan, mukaan lukien turvallisuuspäivitykset, vuoden 2023 kesäkuussa. Tämän takia käyttäjän autentikointi ja kirjautuminen täytyi päivittää ADAL-järjestelmistä uudempaan ja jatkuvan kehityksen alaiseen MSAL-järjestelmään.

MSAL toimii autentikoinnissa hyvin samalla tavalla kuin ADAL-järjestelmäkin. ADAL tukee vain AzureAD V1 -päätepisteitä, kun taas MSAL tukee tuettua ja päivitettyä AzureAD V2 -päätepistettä. Tämä päätepiste tukee käytännössä kaikkia mahdollisia Microsoft-käyttäjätilejä yhdellä päätepisteellä. Käyttöön otettiin Microsoftin kehittämä ja ylläpitämä avoimen lähdekoodin msal-react niminen JavaScript-kirjasto. Sen käyttöönotto oli hyvän dokumentaation avulla suhteellisen helppo toteuttaa.

3.2.3 Rajapinnat

Rajapintoihin ei tehty juurikaan sen isompia muutoksia. Käytössä pysyvät jo ennestään käyttöön otetut REST-rajapinnat sekä Axios-niminen JavaScript-kirjasto. Jokaisen taustajärjestelmiin lähetettävän kutsun mukana lähetetään myös tarvittavia tietoja käyttäjästä ja hänen oikeuksistaan sekä siitä, onko käyttäjä kirjautuneena sovellukseen. Näin varmistetaan, että käyttäjä ei pysty muokkaamaan muita kuin haluttuja asioita. Tätä hyödyntäen pystytään myös jättämään käsittelemättä kutsut, jotka eivät sisällä tarvittavia tietoja.

Isoimmat muutokset tällä saralla ovat Axios-JavaScript-kirjaston päivittäminen, sekä muutaman uuden REST-rajapinnan luominen uusien tarvittavien kutsujen toteuttamiseksi. Ennestään löytyvistä rajapinnoista myös osaan tehtiin muutoksia muuttuneiden määrittelyiden takia. Suurin osa kutsuista kuitenkin pysyi samanlaisina, ja niitä käytetään kuten aikaisemminkin.

3.2.4 Tietojen lisäys ja muokkaus

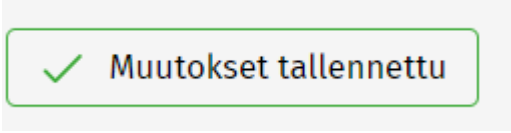
Vanhassa Osaamispankissa käyttäjän tietojen lisäys ja muokkaus nojautui vahvasti modaali-ikkunoiden käyttöön. Jos käyttäjä esimerkiksi halusi lisätä itselleen taidon, niin listauksesta taitoa painaessaan hänelle aukesi modaali-ikkuna, jonka sisällä oli tietoja taidosta sekä taidon lisäämiseen tarvittavien tietojen lomake. Nämä ikkunat eivät kuitenkaan ole kovin hyvä ratkaisu tässä tapauksessa, sillä se pysäyttää käyttäjän sivulla toimimisen täysin ja pakottaa täyttämään tiedot ennen kuin käyttäjä pääsee eteenpäin. Muutamassa paikassa oli vielä käytössä modaali-ikkunan päällä toinen modaali-ikkuna, jolloin käytettävyys laski vain entisestään.

Uudessa Osaamispankissa tehtiin tiukka rajaus, jossa pyrittiin välttämään modaali-ikkunoiden käyttöä kokonaisuudessaan. Jos käyttäjä lisää itselleen taitoa, niin taito avautuu taulukkoon suoraan painetun rivin alle. Täten käyttäjää ei lukita enää yhteen lomakkeeseen, vaan hän pystyy katsomaan jonkun toisen taidon tietoja samalla. Näin ylläpidetään hyvää käyttäjäkokemusta ja käyttäjälle annetaan ainakin näennäisesti enemmän vapauksia toiminnalleen.

Modaali-ikkunoiden poistaminen aiheutti kuitenkin ongelmia tilan hallinnan kanssa. Aikaisemmassa versiossa oli käytössä mobX-tilanhallintakirjasto, jonka avulla saat lähetettyä modaali-ikkunalle halutut tiedot. Tämä toimi erittäin hyvin, kun oli mahdollista avata vain yhden lomakkeen tiedot kerralla. Mutta useamman lomakkeen samaan aikaan avaaminen aiheutti ongelman, jossa päivitetyt mobx-muuttujat yliajoivat aiemmin avatun lomakkeen tiedot. Ratkaisuksi tähän poistettiin mobx-muuttujat käytöstä näiden klikattujen lomakkeiden kohdalla ja tiedot lähetetään lomakekomponenteille käyttäen props-ominaisuutta

Toinen iso muutos tuli jo käyttäjän tiedoista löytyvän datan muokkaamiseen. Ennen tämä toimi samalla tavalla modaali-ikkunan kautta ja erillisellä tallennusnäppäimellä. Lomakkeen sulkemisessa muuten kuin tallennusnäppäintä painamalla ei myöskään ollut mitään erillistä varoitusta tai tuplavarmistusta, eli käyttäjä saattoi mahdollisesti menettää paljonkin tallentamatonta dataa, jos hän erehdyksissään painoi lomakkeen kiinni.

Muokattujen tietojen tallentamiseen mietittiin myös hieman nykyaikaisemmat ratkaisut. Ensinnäkin tallennusnäppäin otettiin kokonaan pois ja tilalle rakennettiin automaattitallennusjärjestelmä, joka tallentaa käyttäjän muokkaaman datan itsestään, kun käyttäjä ei tee mitään sekuntiin. Tämä toteutettiin omaan komponenttiin, joka oli helppo kytkeä kaikkiin haluttuihin lomakekomponentteihin. Tässä apuna käytettiin Lodash-JavaScript-kirjaston debounce-metodia. Haastavinta tässä kytkennässä oli saada se olemaan kutsumatta rajapintakutsuja ja indikoimaan käyttäjälle, jos lomakkeella oli epäkelpoa dataa. Tämä automaattitallennusjärjestelmä kytkettiin myös mobx-muuttujia hyödyntäen navigointipalkkiin, jossa käyttäjälle indikoitiin missä tilassa tietojen tallennus on.



KUVA 1. Muutokset tallennettu infoteksti (Kuva: Eemil Väänänen 2023).

Kun käyttäjä muokkaa dataa ja aikaa kuluu määritelty sekunti, niin automaattitallennusjärjestelmä kutsuu kyseisen lomakkeen tallennusmetodia ja asettaa navigointipalkissa näkyviin tekstin ’Tietoja tallennetaan’. Kun tiedot ovat tallentuneet tietokantaan, niin teksti muuttuu kuvan 1 mukaisesti tilaan ’Tiedot tallennettu’. Jos lomakkeella on epäkelpoa dataa, lukee navigointipalkissa ’Tarkista lomake’. Komponentti, joka sisältää epäkelpoa dataa vaihtuu virhetilaa indikoivaan punaiseen väritykseen, sekä sen alle tulee näkyviin teksti, joka indikoi käyttäjälle miksi kyseinen data on epäkelpoa.

Käytössä on myös ilmoitus, joka näytetään, jos käyttäjä pyrkii navigoimaan pois sivuilta, kun auki on lomake, jossa on tallentamatonta dataa. Tässä tilanteessa käyttäjälle näytetään modaali-ikkuna, jossa varmistetaan, että käyttäjä on tietoinen siitä, että tietoja on tallentamatta. Näin annetaan käyttäjälle vielä mahdollisuus palata takaisin tarkistamaan, miksi tiedot ovat tallentamatta, tai sitten jatkaa pois sivuilta tietoisesti. Tähän tilanteeseen voidaan päätyä esimerkiksi, jos lomakkeella on epäkelpoa dataa, jota ei voida tallentaa tietokantaan ja käyttäjä ei ole tätä dataa korjannut kelvolliseksi.

3.2.5 Navigointi

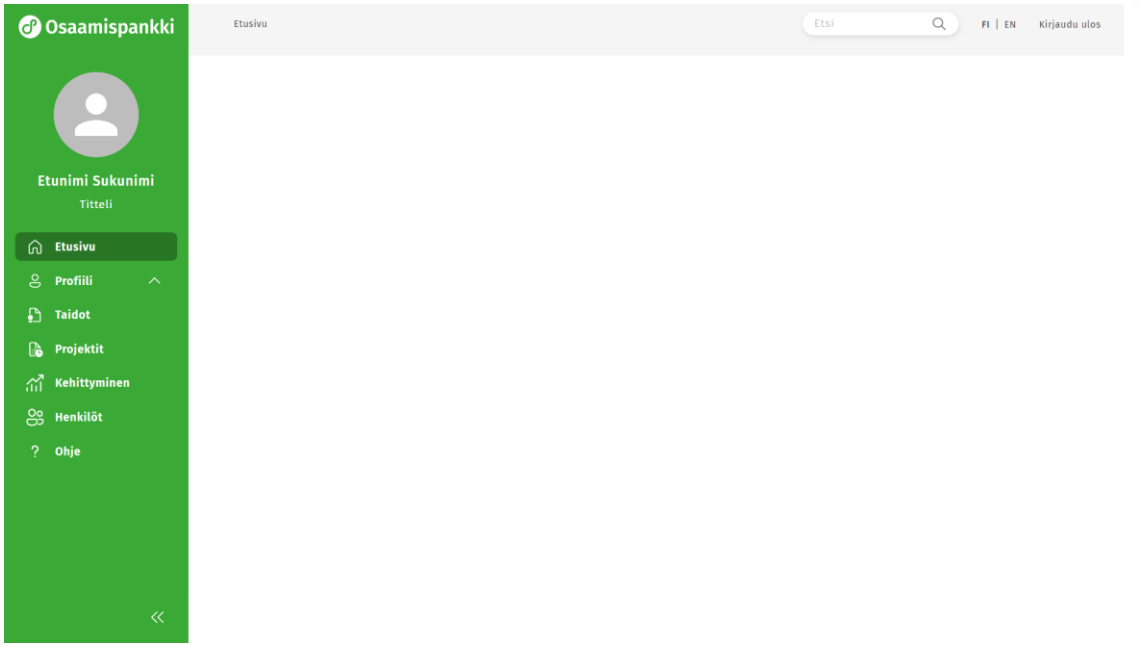
Osaamispankin vanhassa versiossa käyttäjän navigointi oli toteutettu käyttäen kuvan 2 mukaista, sivun yläosaan kiinnitettyä yläpalkkia. Navigaatiopalkista löytyi linkit kaikille Osaamispankin eri alasivuille ja siihen pystyttiin lisäämään ja poistamaan linkkejä riippuen käyttäjän oikeuksista. Sieltä löytyi myös yleisiä hallinta-asetuksia, kuten käyttöliittymän kielen valinta ja uloskirjautuminen.

KUVA 2. Osaamispankin vanha navigointi-/yläpalkki (Kuva: Eemil Väänänen 2023).

Kuvassa 3 näkyy uuden version navigointi. Ensisijaisesti navigointi on toteutettu käyttäen sivupaneelia, joka sisältää kaikki samat linkit mitä vanha sisälsi. Paneelissa olevia linkkejä pystyy myös dynaamisesti muokkaamaan riippuen käyttäjän oikeuksista. Eli käytännössä tämä toimii samalla tavalla kuin vanha yläpalkkiversio. Sivupaneelista on kaksi versiota, täyslevyinen sivupalkki, jossa on käyttäjän kuva, titteli sekä navigointilinkit tekstien kera, sekä pienennetty versio, joka otetaan käyttöön automaattisesti, kun selainikkuna on tarpeeksi pieni tai kun käyttäjä itse pienentää sen. Tässä versiossa ei ole profiilikuvaa, titteliä, eikä muuta tekstiä. Pienennetyn sivupaneelin navigointi toimii käyttäen alasivujen ikoneita sekä hiirellä maalattaessa ilmestyviä vihjelaatikoita.

Uudessa yläpalkissa on kuitenkin navigointiin liittyen kehitetty linkkipolku. Tämän ansiosta navigointipuu pysyy huomattavasti selkeämpänä ja käyttäjä tietää, miten hän on joutunut kyseiselle sivulle sekä mistä hän pääsee takaisin edelliselle sivulle. Yläpalkissa on myös hallinta-asetukset sekä myös uutena ominaisuutena sivustonlaajuinen hakukenttä, jolla käyttäjä voi hakea mitä tahansa tietoja Osaamispankista.

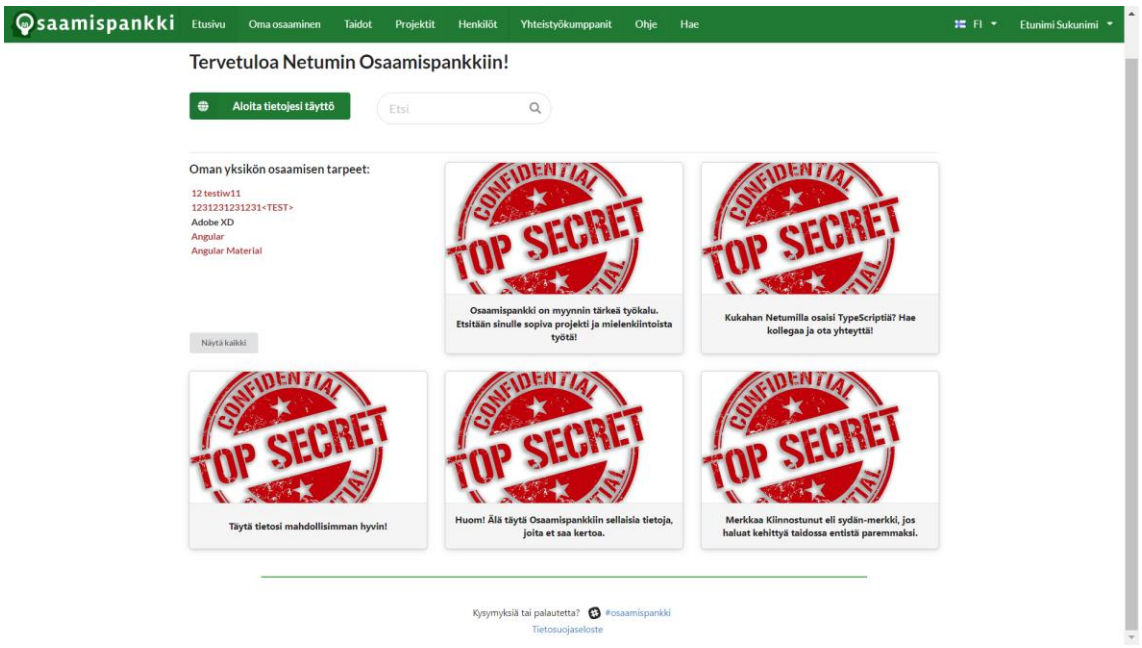
Navigoinnin tekniseen toteuttamiseen käytettiin ja käytetään edelleen react-router nimistä kirjastoa. Kyseinen kirjasto auttaa muun muassa linkkipolun ylläpitämisessä, sillä sen avulla saadaan tieto siitä, missä sovelluksen sisällä liikutaan ja miten sinne on päädytty.



KUVA 3. Osaamispankin uusi navigointi (Kuva: Eemil Väänänen 2023)

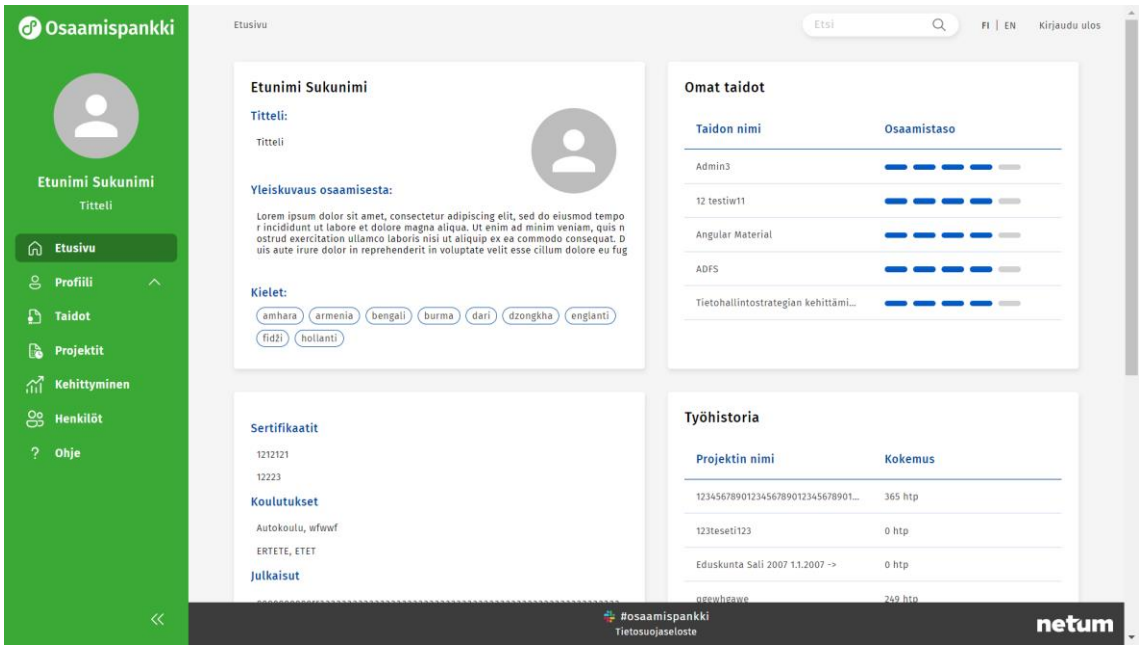
3.2.6 Etusivu

Ehkä merkittävimmät muutokset koko kokonaisuudessa kohdistuivat etusivuun ja sen sisältöön. Vanhan Osaamispankin etusivulla (kuva 4) oli nappula, joka ohjasi käyttäjän yläpalkista löytyvälle Oma osaaminen -alasivulle, hakupalkki sekä kuusi laatikkoa, joista löytyi oman yksikön tarvittavien osaamisten tarpeet ja pieniä tietoiskuja.



KUVA 4. Osaamispankin Vanha etusivu (Kuva: Eemil Väänänen 2023).

Uudessa kokonaisuudessa (kuva 5) päätettiin muuttaa tämä ehkä hieman turha etusivu käyttäjän omaksi profiilisivuksi. Tältä uudistetulta etusivulta/profiilisivulta löytyy käyttäjän lisäämiä tietoja itsestään. Tämä kokonaisuus korvaa käytännössä vanhasta versiosta löytyvän Oma osaaminen -alasivun. Sivulla on näkyvissä käyttäjän titteli, yleiskuvaus osaamisesta, kieliosaaminen, parhaiksi merkatut taidot, työhistoria, tietoja henkilökohtaisesta kehittymisestä ja koulutuksista sekä vanhaltakin etusivulta löytynyt oman yksikön osaamisen tarpeet.



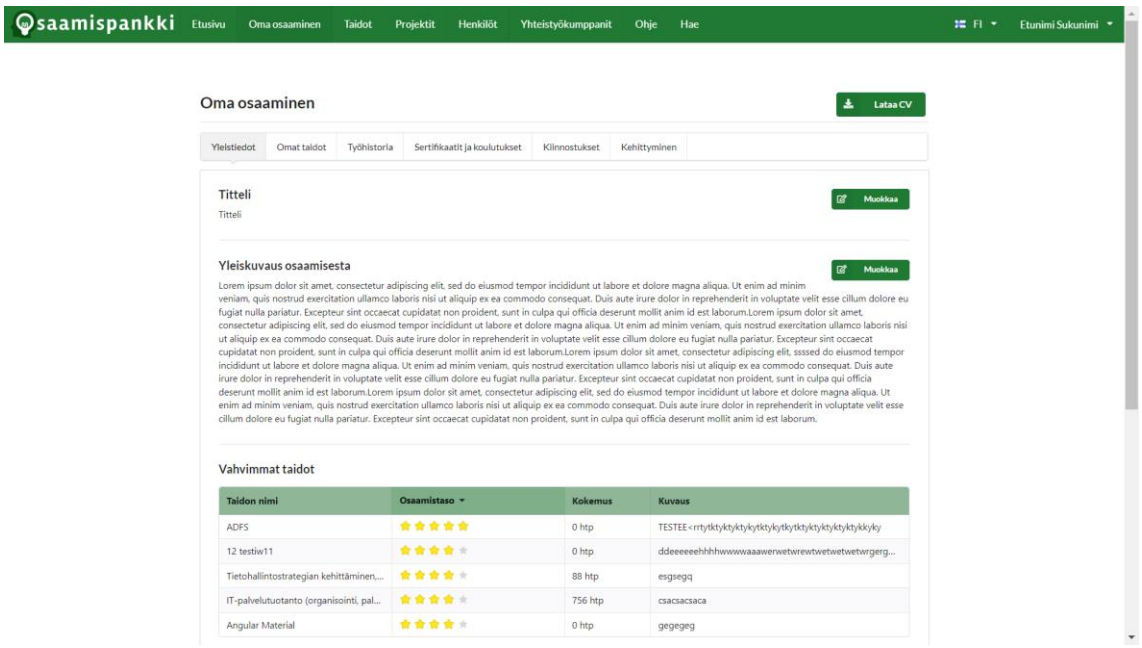
KUVA 5. Osaamispankin uusi etusivu (Kuva: Eemil Väänänen 2023).

Näin kun käyttäjä tulee sovellukseen, näkee hän heti senhetkiset Osaamispankista löytyvät tietonsa ja pystyy näiden perusteella tekemään päätöksen, tarvitseeko niitä päivittää vai ovatko ne ajan tasalla. Etusivu on toteutettu korttimenetelmällä ja käyttäjän klikatessa korttia hän päätyy halutulle alasivulle, josta hän pääsee muokkaamaan haluamiaan tietoja.

3.2.7 Oma osaaminen

Tämä menee samaan kategoriaan luvussa 3.2.6 mainittujen etusivun muutoksien kanssa, sillä uudessa Osaamispankissa etusivu ja Oman osaamisen -alasivu ovat hyvin vahvasti kytköksissä toisiinsa. Kun käyttäjä navigoi itsensä jollekin Oman osaamisen alasivulle, aukeaa samalla myös sivupalkista löytyvä Oma osaaminen -alasvetovalikko automaattisesti. Alasvetovalikko auttaa käyttäjää ymmärtämään missä ollaan, sekä tukee oman osaamisen eri alasivujen välillä navigointia.

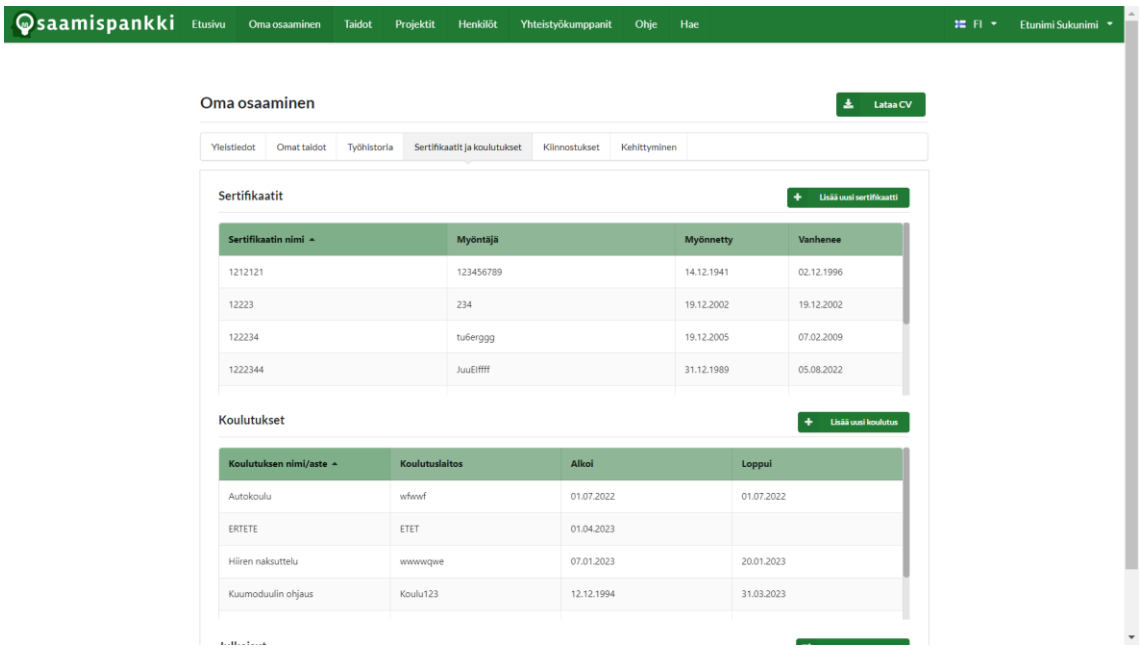
Vanhassa versiossa Oma osaaminen oli toteutettuna omana alasivunaan (kuva 6), jossa oli oma navigointipalkki. Toiminnallisuudeltaan tämä oli ihan pätevä ja toimiva ratkaisu, mutta käyttöliittymässä kokonaisuus oli aika sekava ja vaikeakäyttöinen. Oman osaamisen alaiset osuudet olivat siis saman sivun sisällä, mutta erillisen navigoinnin takana.



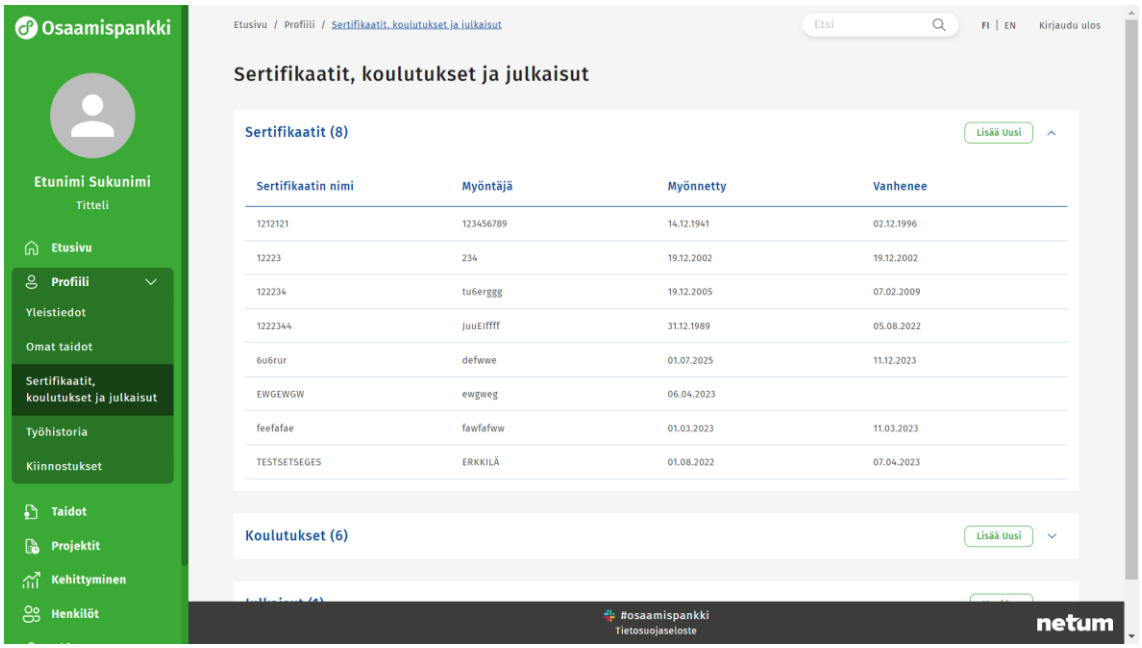
KUVA 6. Oman osaamisen vanha versio (Kuva: Eemil Väänänen 2023).

Uudistetussa versiossa nämä on eroteltu toisistaan. Uudistettu etusivu (kuva 5) toimii pohjana Omalle osaamiselle ja entiset alasivut on nyt muutettu omiksi kokonaisuuksikseen. Näiden muutettujen kokonaisuuksien välillä liikkuminen on toteutettu kahdella eri tavalla. Ensimmäinen vaihtoehto on palata etusivulle käyttäen joko linkkipolusta löytyvää Etusivu -linkkiä tai sitten painamalla sivupalkista Etusivu-painiketta. Tämän jälkeen käyttäjä voi valita etusivun laatikoista haluamansa Oman osaamisen alasivun painamalla tätä. Toinen vaihtoehto on avata sivupalkista löytyvä Profiili-alasvetovalikko painamalla sitä ja sen jälkeen painaa sieltä haluamaansa alasivuvaihtoehtoa, jolloin käyttäjä navigoidaan kyseiselle sivulle.

Oman osaamisen -alasivujen ulkonäköä on myös päivitetty suunnittelujärjestelmän mukaisesti. Kuvissa 7 ja 8 näkyy eroavaisuuksia Sertifikaatit ja koulutukset -alasivusta. Aiemmin visuaalisesti ahdas kokonaisuus on muutettu käyttämään alasvetokomponentteja, joiden sisällä on kyseisen komponentin otsikon mukaisia tietoja. Alasvetokomponenteista ylin on auki aina kun sivulle navigoidaan. Kuvissa 9 ja 10 verrataan vanhan ja uuden version omat taidot -alasivuja.



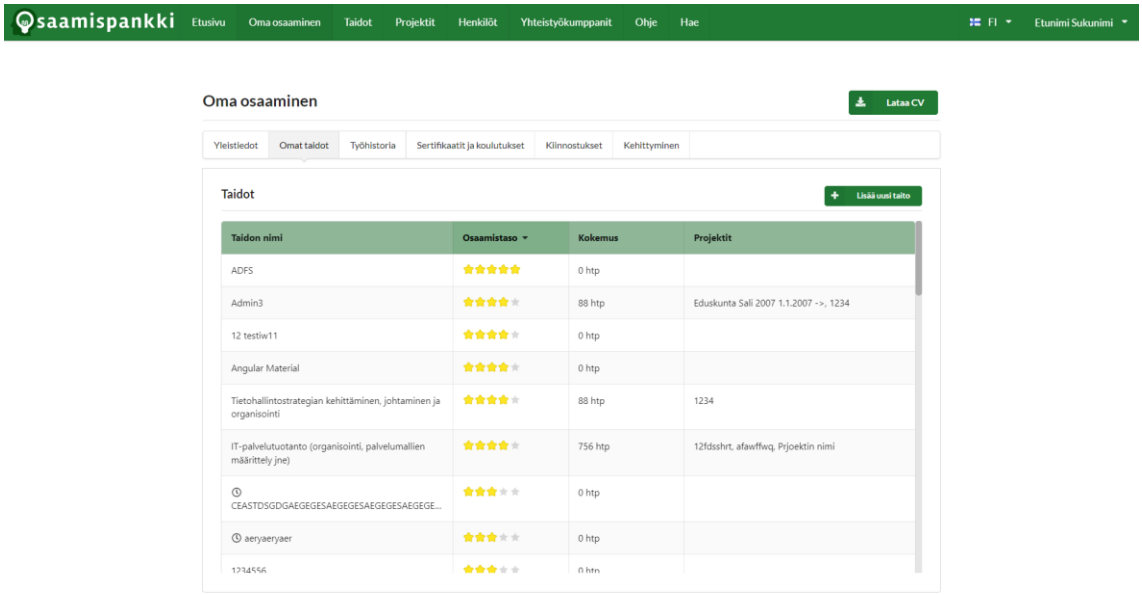
KUVA 7. Vanha versio Oman osaamisen Sertifikaatit ja koulutukset -alasivusta (Kuva: Eemil Väänänen 2023).



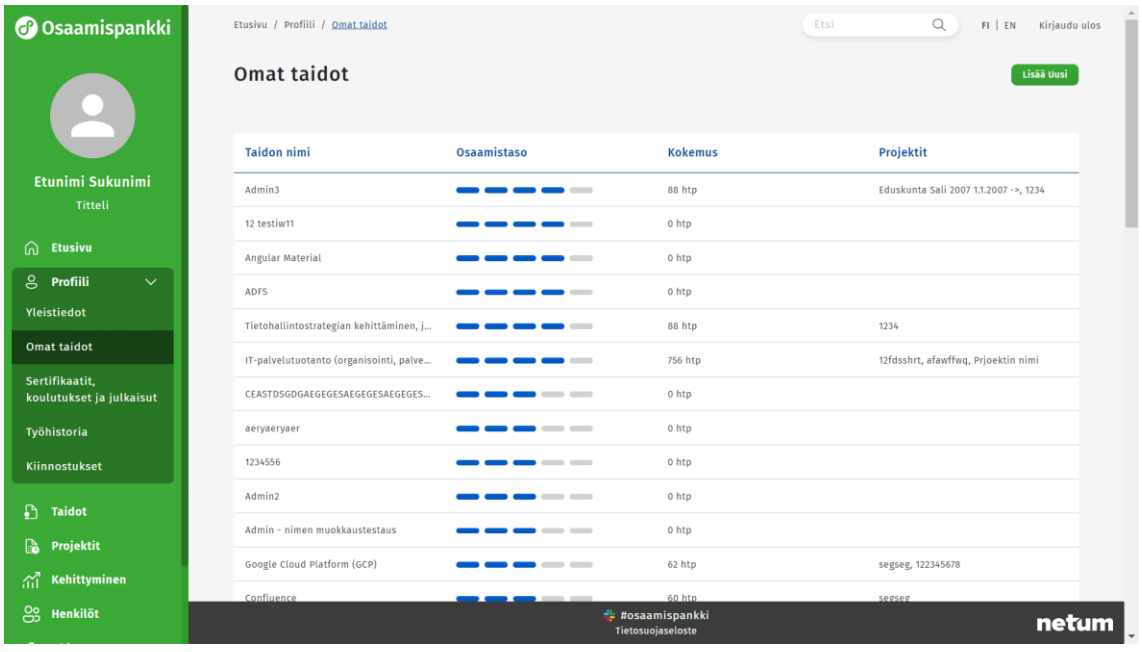
KUVA 8. Päivitetty versio Oman osaamisen Sertifikaatit ja koulutukset -alasivusta (Kuva: Eemil Väänänen 2023).

Funktionaalisesti tietojen lisääminen toimii samalla tavalla kuin vanhassa Osaamispankin versiossa. Rajapintakutsut tehdään samanlaisilla tiedoilla kuin aiemmin ja kaikki lomakkeet on rakennettu niin, että niistä saadaan rakennettua identtiset tiedot vanhojen lomakkeiden kanssa. Taustalta jo löytyvät hyvin toteutetut

rajapintakutsut sekä tehokkaasti toimivat käyttöliittymäkomponentit mahdollistivat nopean ja toimivan järjestelmän kehittämisen.



KUVA 9. Vanha Oman osaamisen omat taidot -alasivu (Kuva: Eemil Väänänen 2023).



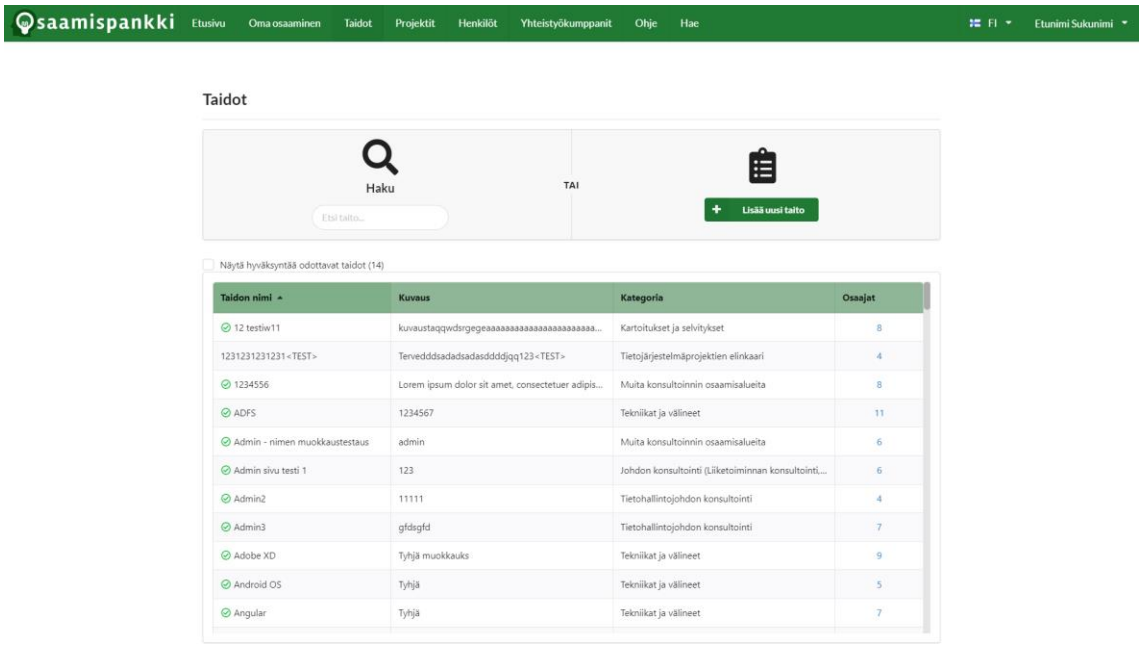
KUVA 10. Uusi Oman osaamisen omat taidot -alasivu (Kuva: Eemil Väänänen 2023).

Käyttöliittymäpuolella muutoksia on tullut lomakkeiden toimintaan. Ensinnäkään mitkään lomakkeet eivät enää aukea modaalisessa ikkunassa, vaan ne aukeavat

samalle tasolle muiden käyttöliittymäkomponenttien kanssa. Tietoja muokattaessa on käytössä automaattitallennusjärjestelmä, eli käyttäjän ei tarvitse erikseen painaa mitään tallennusnäppäintä, vaan tiedot tallentuvat automaattisesti tietokantaan. Tämän toiminnasta on tarkempaa tietoa luvussa 3.2.4.

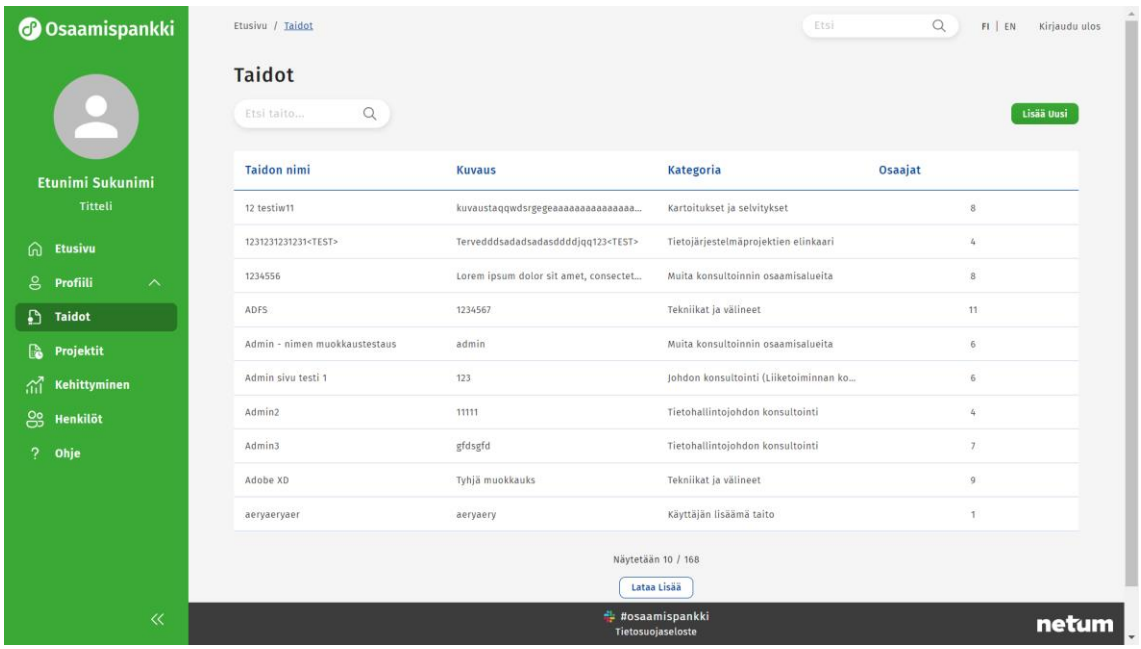
3.2.8 Taidot ja Projektit

Taidot-sivu (kuva 12) on toiminnallisuudeltaan identtinen vanhan Taidot-sivun (kuva 11) kanssa. Taulukossa on kaikki samat sarakkeet, ja tiedot haetaan samoilla rajapintakutsuilla mitä vanhassakin käytettiin. Taulukon sarakkeina käytetään nimeä, kuvausta, taidon kategoriaa sekä taidon osaajia.



KUVA 11. Vanha taidot-sivu (Kuva: Eemil Väänänen 2023).

Eroavaisuudet ovat enimmäkseen kosmeettisia. Uuden taidon lisäyksessä nappulasta on poistettu sana taito, sillä sen ei koettu olevan relevantti, koska kontekstin perusteella voidaan päätellä, mitä ollaan tekemässä. Nappulan painaminen avaa luontilomakkeen nappulan ja taulukon väliin. Näin käyttäjä voi esimerkiksi vielä varmistaa, että kyseisellä nimellä ei löydy jo taitoa.



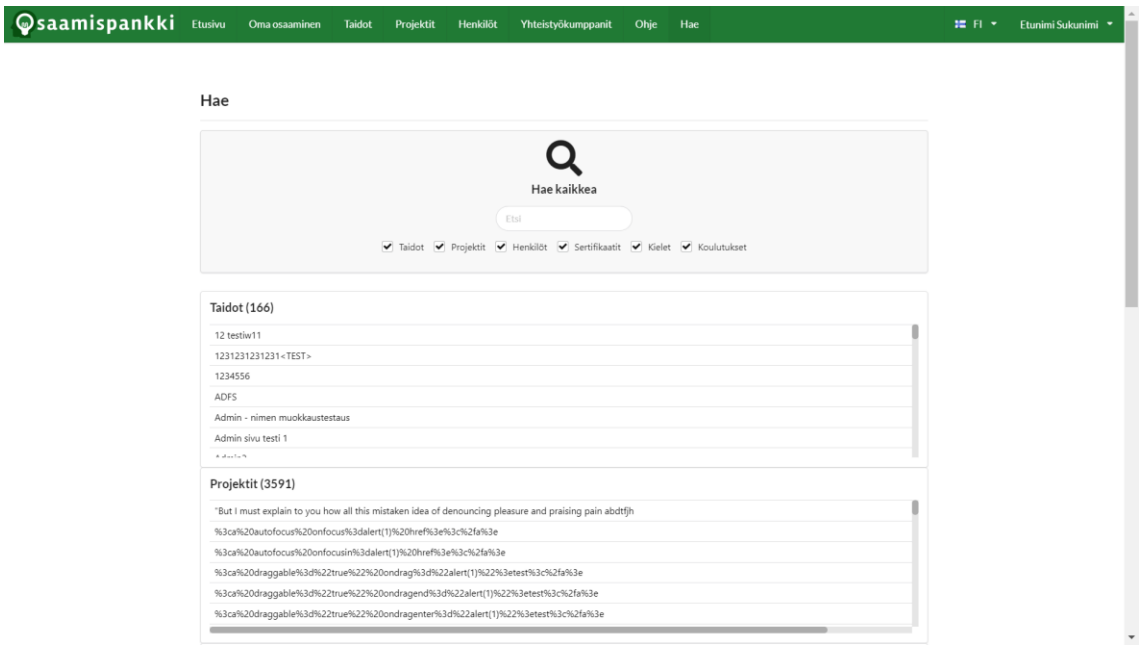
KUVA 12. Uusi taidot-sivu (Kuva: Eemil Väänänen 2023).

Samoin kun avaa listauksesta taidon, niin se aukeaa listauksen alapuolelle modaali-ikkunan sijaan. Jos käyttäjä on lisäämässä uutta taitoa, niin silloin hänen täytyy painaa erikseen tallennusnäppäintä, mutta jos käyttäjältä löytyy jo kyseinen taito tiedoista, on käytössä automaattitallennusjärjestelmä, jonka avulla käyttäjän muokkaamat tiedot pysyvät ajan tasalla vaivattomasti.

Muutoksien samat perusperiaatteet toimivat myös Projektit-sivulla. Projektin tietoja muokattaessa käytössä on automaattitallennus, modaali-ikkunoiden käyttö on lopetettu, kieliasu yksinkertaistettu ja käyttöliittymän visuaalinen ulkoasu päivitetty vastaamaan Figmaan piirrettyjä muutoksia.

3.2.9 Haku

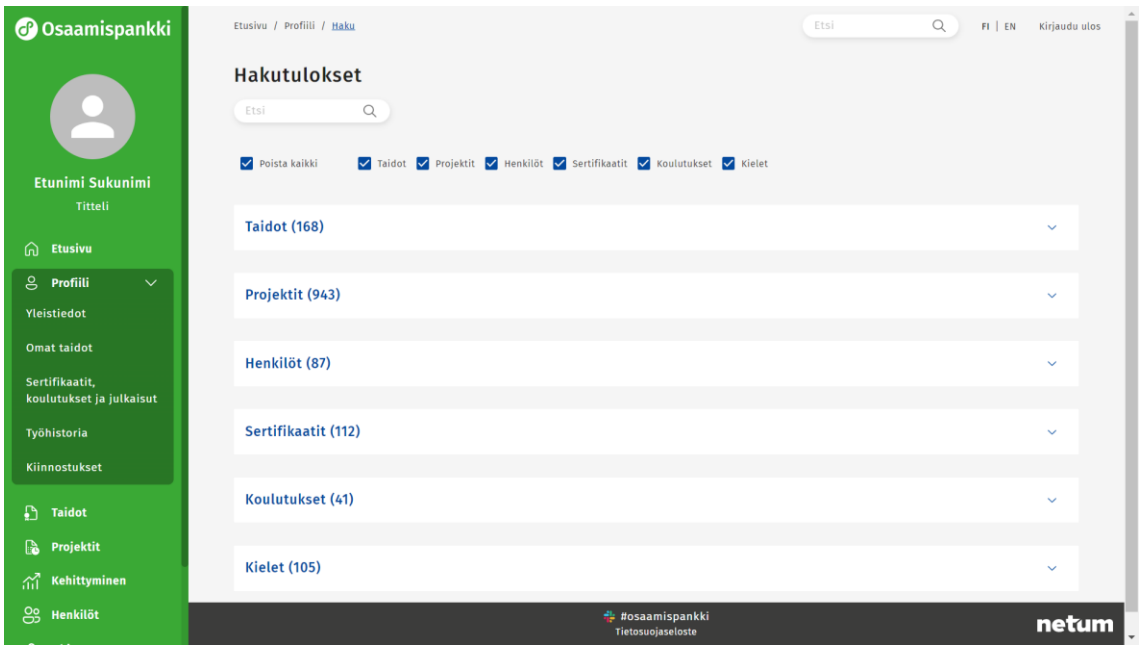
Vanhan Osaamispankin hakusivun (kuva 13) ja uuden Osaamispankin hakusivun (kuva 14) välillä ei ole myöskään juuri funktionaalisia muutoksia. Pari pienempää muutosta on toteutettu, muun muassa se, että valintalaatikoissa on myös erikseen vaihtoehto, jolla voi valita kaikki hakukohteet tai vaihtoehtoisesti poistaa nämä kaikki valinnat.



KUVA 13. Vanha hakusivu (Kuva: Eemil Väänänen 2023).

Hakujen suodatus toteutetaan jokaisen näppäimenpainalluksen jälkeen, jolloin käyttöliittymä on nopea ja sulava. Tämän kääntöpuoli on se, että sovellus joutuu tekemään useita suodatuksia samaan aikaan jokaisen kirjainmuutoksen jälkeen, sekä se, että kaikki tiedot täytyy hakea tietokannasta ennen ensimmäistä hakua.

Hakusivulle pääsy on hieman kokenut muutoksia. Vanhassa versiossa hakusivulle pääsi kahdella eri tavalla: painamalla yläpalkista erikseen löytyvää Hae-nappulaa, tai menemällä etusivulle ja kirjoittamalla sieltä löytyvään hakukenttään haluamansa hakusanan.



KUVA 14. Uusi hakusivu (Kuva: Eemil Väänänen 2023).

Uudessa versiossa hakusivulle pääsy toteutettiin pelkästään navigointipalkista löytyvää hakukenttää käyttäen. Käytössä ei siis enää ole erillistä Hae-nappulaa. Tämän etu on, että käyttäjä voi aloittaa hakunsa mistä tahansa alasivulta ilman etusivulle navigointia tai ylimääräistä Hae-nappulan kautta tehtävää navigointia haun alasivulle.

3.3 Material UI komponenttien muokkaus

3.3.1 Yksittäisen komponentin muokkaaminen

Komponentin yksittäisen instanssin muokkaamiseen Material UI:ssa löytyy kaksi virallista tapaa. Ensimmäisenä vaihtoehtona on käyttää komponenteilta löytyvää sx-ominaisuutta ja toisena vaihtoehtona tyylittely voidaan yliajaa mukautetulla CSS-luokalla.

Material UI:n sx-ominaisuus on tyylikäsittelyominaisuus, joka mahdollistaa komponentin tyylittelyn suoraan komponenttia luotaessa. Se sisältää laajan CSS:n ylijoukon sekä Material UI:n omia tyylittelyominaisuuksia, joilla tyylittely on help-

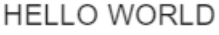
poa ja nopeaa. Sitä käytetään, kun halutaan säilyttää tyylien määrittely komponentin yhteydessä. Ominaisuus ottaa vastaan objektin, joka sisältää halutut tyylittelymuutokset avain-arvo-pareina. Tässä objektissa voi olla niin monta avain-arvo-paria kuin halutaan, ja sille voidaan myös antaa pseudoluokkia tai pseudoelementtejä käyttämällä &-merkkiä ennen luokan nimeä. On kuitenkin hyvä pitää mielessä, että sx-ominaisuus ei skaalaudu kovin hyvin. Jos tehdään useita samantyyppisiä komponentteja, tiedostoihin saattaa tulla hyvin paljon koodin toistoa samanlaisten tyylittelyiden muodossa.

Esimerkkinä voidaan luoda Button-komponentti, jonka taustaväri muutetaan sx-ominaisuutta hyödyntäen siniseksi. Ensimmäisenä täytyy tietenkin luoda muokattava komponentti (kuva 15). Nyt komponentti näyttää samalta kuin kuvassa 16 oleva nappula.

Tämä on kuitenkin hyvin vaikea tunnistaa nappulaksi ilman erottuvaa taustaväriä, joten seuraavaksi lisätään sille taustaväri kuvasta 17 löytyvää sx-ominaisuutta käyttäen. Tämän jälkeen nappulalla on kuvassa 18 näkyvä sininen taustaväri. Jos halutaan vielä lisätä spesifisyyttä ja muokata nappulan väriä, kun sen päälle viedään hiiri, voidaan se toteuttaa lisäämällä sx-ominaisuuden objektiin pseudoluokka &:hover ja määrittelemällä sille backgroundColor-muuttuja (kuva 19).

```
return (
  <Button>
    Hello World
  </Button>
);
```

KUVA 15. Tyylittelemättömän Material UI:n Button-komponentin luonti (Kuva: Eemil Väänänen 2023).



KUVA 16. Muokkaamaton Material UI:n nappula (Kuva: Eemil Väänänen 2023).

```
  return (
  <Button sx={{ backgroundColor: "blue" }}>
    HELLO WORLD
  </Button>
  );
```

KUVA 17. Nappulalle on lisätty sx-ominaisuus, joka sisältää sinisen taustavärin avain-arvo-parin objektin sisällä (Kuva: Eemil Väänänen 2023).

KUVA 18. Material UI:n nappula, jonka taustaväri on vaihdettu (Kuva: Eemil Väänänen 2023).

```
return (
    <Button
      sx={{
        backgroundColor: "blue",
        "&:hover": {
          backgroundColor: "green"
        }
      }}
    >
      HELLO WORLD
    </Button>
  );
```

KUVA 19. Nappulan sx-ominaisuudessa olevaan objektiin on lisätty spesifisyyttä pseudoluokkaa ’&:hover’ käyttäen (Kuva: Eemil Väänänen 2023).

Komponentteja voidaan muokata myös perinteisemmin käyttäen CSS-tiedostoa tyylien määrittelyyn. Näin voidaan toimia antamalla komponentille className-ominaisuuden halutulla nimellä ja luomalla tyylittelytiedoston, jossa määritellään tyylittelyominaisuudet kyseisen luokkanimen alle. Muokkaamisessa voi myös käyttää Sass-, Stylus-, Less- tai muuta vastaavaa CSS-esiprosessoria.

Hyödyt verrattuna sx-ominaisuuteen ovat mahdollisuus käyttää haluamaansa CSS-esiprosessoria ja se, että useammalle komponentille voi halutessaan käyttää samaa pohjaluokkaa, jolloin monen komponentin tyylittely pysyy järjestelmällisempänä ja helpommin ylläpidettävänä. Jos komponentilla on monimutkaiset tyylittelyt, niin ne on parempi luoda omassa tiedostossaan, jotta itse komponenttitiedosto pysyy siistinä ja luettavana sekä helpommin ylläpidettävänä.

Kuvan 18 mukaisen nappulan voi toteuttaa myös käyttäen CSS-tyylittelyä. Aloitetaan luomalla CSS-tiedosto ja annetaan sille nimeksi customButton. Koska kyseessä on CSS-tiedosto, niin käytetään tiedoston nimessä .css-päätettä. Tämän tiedoston sisälle luodaan tyylittelyluokka ja nimetään se custom-button (kuva 20).

Seuraavaksi siirrytään takaisin nappulakomponentin lähdekooditiedostoon ja lisätään sinne import-lauseessa polku yllä luotuun customButton CSS-tyylittelytiedostoon. Koska tässä tapauksessa sekä lähdekoodi- että tyylittelytiedosto ovat samassa kansiossa, voidaan käyttää import-lauseessa kuvan 21 mukaista tekstiä. Tämän jälkeen voidaan kyseisessä lähdekooditiedostossa käyttää kyseisen tyylittelytiedoston tyylittelyluokkia, kuten Reactissa CSS-tyylittely tapahtuu – käyttämällä className-ominaisuutta (kuva 22).

Nyt nappula näyttää samalta kuin kuvassa 18 oleva esimerkki. Jos halutaan muokata taustaväri vihreäksi hiiren ollessa nappulan päällä, palataan takaisin customButton tyylittelytiedostoon ja lisätään custom-button-tyylittelyluokan alle :hover-pseudoluokka kuvan 23 mukaisesti.

Tämän voi myös toteuttaa ilman className-ominaisuuden käyttöä, jos CSS-tyylittelytiedosto on oikein importattu ja luokan nimenä käytetään kyseisen komponentin Material UI:ssa annettua CSS-luokkanimeä. Tällöin yliajetaan suoraan kyseisen komponentin tyylittelyt omilla. Material UI:n nappulan juuriluokka on nimeltään ’MuiButton-root’, eli sen globaali yliajo tehtäisiin kuvan 24 mukaisesti.

Nyt nappulalta voi poistaa className-ominaisuuden, ja sen taustaväri on silti sininen (kuva 18) ja vihreä kun hiiri on sen yllä. Tässä vaihtoehdossa on kuitenkin kohtalainen riski siihen, että tyylittelyt vuotavat muihin komponentteihin ja aiheuttavat epätoivottuja muutoksia muualla.

```
.custom-button {
  background-color: blue;
}
```

KUVA 20. customButton.css tiedoston sisältä löytyvä tyylittelyluokka custom-button (Kuva: Eemil Väänänen 2023).

```
import "./customButton.css";
```

KUVA 21. import-lauseke kun tyylittelytiedosto löytyy samasta kansiosta lähdekooditiedoston kanssa (Kuva: Eemil Väänänen 2023).

```
  return (
     <Button className="custom-button">HELLO WORLD</Button>
  );
```

KUVA 22. Nappulan tyylittely käyttäen className-ominaisuutta (Kuva: Eemil Väänänen 2023).

```
.custom-button {
  background-color: blue;
}
.custom-button:hover {
  background-color: green;
}
```

KUVA 23. Nappulan tyylittelylle lisätty :hover-pseudoluokka (Kuva: Eemil Väänänen 2023).

```
.MuiButton-root {
  background-color: blue;
}
.MuiButton-root:hover {
  background-color: green;
}
```

KUVA 24. Nappulan tyylittelyn yliajo käyttäen Material UI:n määrittelemää CSS-luokkanimeä (Kuva: Eemil Väänänen 2023).

Material UI:n komponenteille on tarjolla myös erilaisia tilaluokkia. Näiden avulla voidaan kohdistaa tyylittelyitä tiettyihin tiloihin, vaikka niille ei ole erikseen määriteltyä CSS-pseudoluokkaa. CSS-pseudoluokilla on korkea spesifisyysaste, joten Material UI:n vastaavilla tilaluokilla on sama spesifisyysaste johdonmukaisuuden varmistamiseksi.

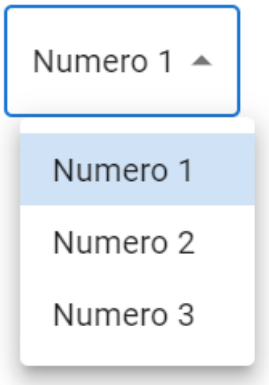
Jos esimerkiksi haluttaiisiin vaihtaa MenuItem-komponentin väriä, kun se on valittuna listasta, voidaan käyttää Material UI:hin sisäänrakennettua MUI-selected-tilaluokkaa. Tämä tilaluokka toimii CSS-pseudoluokan tavoin korkealla spesifisyydellä.

Aluksi luodaan yksinkertainen Select-komponentti ja annetaan sille kolme erillistä MenuItem-komponenttia kuvan 25 mukaisesti. Select-komponentti ja valittu MenuItem-komponentti näyttävät nyt samalta kuin kuvassa 26. Tämän jälkeen luodaan customMenu.css-tyylittelytiedosto ja importataan kuvassa 25 luotuun Select-komponenttiin (kuva 27). Seuraavaksi voidaan lisätä customMenu.css-tyylittelytiedostoon kuvassa 28 näkyvä tilaluokan yliajo. Kun tämä on lisätty, on Select-komponentista valitun MenuItem-komponentin taustaväri muuttunut vaaleanvihreäksi (kuva 29).

```
  const [val, setVal] = useState(1);

  return (
    <Select value={val} onChange={(e) => setVal(e.tar-
get.value as number)}>
      <MenuItem value={1}>Numero 1</MenuItem>
      <MenuItem value={2}>Numero 2</MenuItem>
      <MenuItem value={3}>Numero 3</MenuItem>
    </Select>
  );
```

KUVA 25. Tyylittelemättömän Select-komponentin luonti (Kuva: Eemil Väänänen 2023).



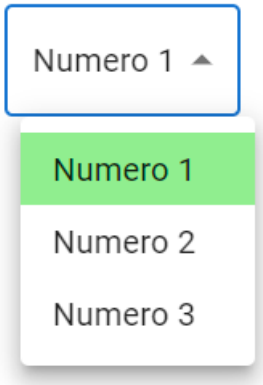
KUVA 26. Muokkaamaton Material UI:n alasvetovalikko (Kuva: Eemil Väänänen 2023).

```
import "./customMenu.css";
```

KUVA 27. customMenu.css-tyylittelytiedoston importtaus (Kuva: Eemil Väänänen 2023).

```
.MuiMenuItem-root.Mui-selected {
  background-color: lightgreen;
}
```

KUVA 28. CSS-tyylittelytiedostoon lisätty tilaluokan yliajo (Kuva: Eemil Väänänen 2023).



KUVA 29. Muokattu Material UI:n alasvetovalikko (Kuva: Eemil Väänänen 2023).

3.3.2 Uudelleenkäytettävän komponentin tyylittely

Jos halutaan luoda uudelleenkäytettävä komponentti omilla tyylittelyillä, voidaan se tehdä käyttäen Material UI:n styled-apufunktiota. Tämä mahdollistaa muun

muassa dynaamisen CSS-tyylittelyn käytön esimerkiksi yläkomponentin tilan mukaan ja helpottaa komponentin ylläpitoa huomattavasti.

Kuvasta 18 löytyvän Button-komponentin uudelleenkäytettävän version voi toteuttaa esimerkiksi seuraavalla tavalla. Aloitetaan luomalla uusi tiedosto ja nimeämällä se customButton.tsx. Käytetään tässä esimerkissä .tsx-tiedostopäätettä, näin saadaan käyttöön TypeScriptin tuomat hyödyt. Nyt voidaan edellä luodun tiedoston sisälle lisätä kuvan 30 mukainen vakiomuuttuja.

Seuraavaksi voidaan tähän samaan tiedostoon lisätä kuvassa 31 näkyvä uudelleenkäytettävä Button-komponentti. Tämä komponentti käyttää pohjana kuvassa 30 luotua tyyliteltyä Buttonia. Nyt Button-komponentilla on halutut tyylittelyt, sekä sille voi antaa haluamansa tekstin kuvassa 32 osoitetulla tavalla.

```
const CustomButton = styled(Button)<ButtonProps>(({ theme })
=> ({
  backgroundColor: "blue",
  "&:hover": {
    backgroundColor: "green"
  }
}));
```

KUVA 30. Muokatun Button-komponentin luonti styled-apufunktiota käyttäen (Kuva: Eemil Väänänen 2023).

```
export default function StyledButton(props: { buttonText:
string }) {
  const { buttonText } = props;
  return <CustomButton>{buttonText}</CustomButton>;
}
```

KUVA 31. Uudelleenkäytettävä Button-komponentti tyylittelyillä. Se ottaa vastaan nappulassa näkyvän tekstin props-ominaisuuden kautta (Kuva: Eemil Väänänen 2023).

```
const App = () => {
  return <StyledButton buttonText={"Hello World"} />;
};
```

KUVA 32. Kuvasta 31 löytyvän nappulan käyttäminen (Kuva: Eemil Väänänen 2023).

3.3.3 Teeman yliajo globaalisti

Kolmas tapa muokata komponentteja on yliajaa niiden tyylittely globaalisti käyttäen createTheme-funktiota, jonka palauttama objekti annetaan ThemeProvider-komponentille props-ominaisuutena. Tämän jälkeen kaikki ThemeProvider-komponentin sisällä olevat komponentit käyttävät createThemellä muokattuja tyylittelyitä. ThemeProvider-komponentti on suositeltavaa asettaa web-sovelluksen komponenttihierarkiassa ylimmäksi. Näin varmistutaan, että muokkaukset näkyvät kaikkialla.

CreateTheme-funktio ottaa vastaan tyylittelyitä sisältävän objektin ja lisää siihen automaattisesti kaikki puuttuvat tyylittelyt, näin kehittäjän ei tarvitse itse määritellä jokaisen komponentin jokaista tyylittelyä, vaan hän voi määritellä vain haluamansa yliajettavat tyylittelyt.

Kuvassa 18 olevan Button-komponentin tyylittely globaalisti tapahtuu seuraavalla tavalla. Aloitetaan luomalla vakiomuuttuja, joka sisältää halutut tyylittelymuutokset. Koska halutaan muokata Button-komponentin värejä, luodaan vakiomuuttuja kuvan 33 mukaiseksi. Seuraavaksi lisätään kuvasta 34 löytyvä Themeprovider-komponentti, jolle annetaan theme-ominaisuuteen kuvassa 33 luotu vakiomuuttuja.

Nyt kaikki ThemeProvider-komponentin sisälle asetetut komponentit noudattavat theme-vakiomuuttujassa määriteltyjä tyylittelyjä ilman erillisiä sx-, tai className-ominaisuuksia. Jos sen sisään asetetaan kuvan 35 mukaisesti Button-komponentti, noudattaa se nyt kuvan 33 määrittelyjä.

```
const theme = createTheme({
  components: {
    MuiButton: {
      styleOverrides: {
        root: {
          backgroundColor: "blue",
          "&:hover": {
            backgroundColor: "green"
          }
        }
      }
    }
  }
});
```

KUVA 33. Teeman vakiomuuttujan luonti. Määritellään nappulan taustavärit (Kuva: Eemil Väänänen 2023).

```
return (
    <ThemeProvider theme={theme}>
    </ThemeProvider>
);
```

KUVA 34. ThemeProvider-komponentin luonti. Annetaan theme-ominaisuuteen kuvassa 33 luotu vakiomuuttuja (Kuva: Eemil Väänänen 2023).

```
  return (
    <ThemeProvider theme={theme}>
      <Button>Hello World</Button>
    </ThemeProvider>
  );
```

KUVA 35. Kaikki ThemeProvider-komponentin sisällä olevat komponentit noudattavat theme-ominaisuudessa olevan vakiomuuttujan avain-arvo-pareja (Kuva: Eemil Väänänen 2023).

4 POHDINTA

Käyttöliittymän päivityksen ensimmäinen versio on saatettu tuotantoon tämän opinnäytetyön aikana. Mitään tiukkoja aikatauluja ei päivitykselle ollut asetettu, mutta ajallisesti työn toteutukseen meni odotettua pidempään. Suurimpia syitä tälle olivat teknisten vaatimusten muuttumiset, työskentelyvalmiuksien ongelmat, sekä Figmapiirrosten tarkennukset ja muutokset. Näiden lisäksi myös testaus ja löytyneiden bugien korjaus vei yllättävässä määrin aikaa.

Projektin tiimityöskentely oli kuitenkin hyvin toimivaa. Kommunikaatio sujui erinomaisesti tiimin jäsenten kesken, ja muuttuneista määrittelyistä saatiin informaatio nopeasti. Tiimin kesken pystyttiin myös hyvin keskustelemaan ja käsittelemään ongelmakohdat nopeasti. Esimerkiksi jos oltiin rakentamassa jotain tiettyä ominaisuutta, niin saatiin hyvää, kehittävää keskustelua aiheesta ja lopputuloksena oli usein hyvin valmiita kokonaisuuksia, joita oli helppo lähteä toteuttamaan teknisesti. Myös epäkohdista saatiin hyvää keskustelua. Jos jonkun mielestä jossain asiassa oli parantamisen varaa, niin kaikki pääsivät siitä sanomaan oman mielipiteensä ja näiden keskustelujen perusteella tehtiin tarvittavia muutoksia.

Osaamispankin kehitys jatkuu jälleen tulevana kesänä. Työlistalla on teknisten ongelmien, eli bugien korjausta sekä myös uusien ominaisuuksien suunnittelua ja kehittämistä. Mahdollisiin palautteisiin reagoidaan tarvittaessa. Palautteet kerätään yhteen ja kehitystiimi tekee päätöksiä komponenttien muokkaamisesta niiden perusteella. Jatkokehitykselle on jo kuitenkin aika tarkat suunnitelmat mietitty.

Kokonaisuutena työ saatiin ihan asiallisesti hoidettua valmiiksi. Muutamia komponentteja olisi voinut tehdä eri tavalla. Varsinkin kuin osan niistä tekniset vaateet muuttuivat. Myös koodipohjaista testaamista olisi ehdottomasti pitänyt lisätä. Nyt lähes kaikki käyttöliittymän testaukset tehtiin manuaalisesti, mutta tulevaisuudessa koodipohjainen testaus tulee välttämättömästi esiin ja sille täytyy myös antaa sen tarvitsema kehitysaika.

Kokonaisuutena suunnittelujärjestelmän käyttöönotto oli hyvä päätös. Sen avulla sovelluksen kehittäminen kehittäjän näkökulmasta huomattavasti tehokkaampaa. Kattava suunnittelujärjestelmä poisti paljon epätietoisuutta komponenttien ja sivujen määrittelystä ja nopeutti epäkohtien korjaamista huomattavasti. Rajatapaukset ja virhetilojen suunnittelu oli myös huomattavasti helpompaa järjestelmää hyödyntäen.

LÄHTEET

A tour of the C# language. Kuvaus C# ohjelmointikielestä. Viitattu 21.2.2023 https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Cory Gackenheimer. 2015. Introduction to React. Berkeley, CA: Apress: Imprint: Apress

How does Azure work? Kuvaus Azuresta. Viitattu 21.2.2023 https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/get-started/what-is-azure

Josh Goldberg. 2022. Learning Typescript. O'Reilly Media, Inc

Overview – Material UI. Kuvaus Material UI:n historiasta ja toiminnasta. Viitattu 21.2.2023 https://mui.com/material-ui/getting-started/overview/

Overview of the Microsoft Authentication Library (MSAL). Kuvaus MSAL kirjastosta. Viitattu 21.2.2023 https://learn.microsoft.com/en-us/azure/active-directory/develop/msal-overview

Tom Negrino, Dori Smith. 2012. JavaScript: Visual QuickStart Guide, Eighth Edition. Peachpit Press

What is .NET? Introduction and overview. Kuvaus .NET sovelluskehyksestä. Viitattu 21.2.2023 https://learn.microsoft.com/en-us/dotnet/core/introduction

What is Azure Active Directory? Kuvaus Azure AD:sta. Viitattu 21.2.2023 https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis

What is Figma. Kuvaus Figmasta ja sen historiasta. Viitattu 21.2.2023 https://www.nobledesktop.com/learn/figma/what-is-figma

What is Microsoft SQL Server and what is it for? Kuvaus MSSQL tietokantajärjestelmistä. Viitattu 21.2.2023 https://intelequia.com/en/blog/post/2948/what-is-microsoft-sql-server-and-what-is-it-for