jamk.fi

Tuntikirjanpitojärjestelmä moderneilla web-teknologioilla

Juuso Kyyrä

Opinnäytetyö
Joulukuu 2020
Tekniikan ala
Insinööri (AMK), tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Kyyrä, Juuso	Opinnäytetyö	Joulukuu 2020
	Sivumäärä 38	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Tuntikirjanpitojärjestelmä moderneilla web-teknologioilla

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Manninen Pasi, Niemi Kari

Toimeksiantaja(t)
-

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa tuntikirjanpitojärjestelmä, joka kilpailee jo valmiina olemassa olevia järjestelmiä vastaan. Valmiina olemassa olevat järjestelmät ovat joko hyvin kalliita tai valmistettu vanhentuneita teknologioita hyödyntäen, joten uudelle tuntikirjanpitojärjestelmälle oli tarvetta. Tavoitteena oli myös valmistaa responsiivinen toteutus ja käyttää uusimpia teknologioita, jotta kuka tahansa pystyisi käyttämään verkkosovellusta nopeasti ja helposti millä tahansa laitteella.

Tuntikirjanpitojärjestelmä on Node.js:n päälle rakennettu React.js-sovelluskehyksellä valmistettu verkkosovellus, jonka taustalla on Express sekä MongoDB-tietokanta. React.js ottaa yhteyttä rajanpinnan avulla Expressiin, joka hakee tiedot tietokannasta ja palauttaa ne rajapinnan läpi Reactiin. Express:n ja MongoDB:n välistä kommunikaatiota on helpottamassa mongoose, joka muotoilee MongoDB:stä tulevan datan JSON-muotoon ja helpottaa sen luettavuutta.

Opinnäytetyön tuloksena saatiin luotua yksinkertainen ja responsiivinen vekkosovellus. Verkkosovelluksessa pystytään kirjaamaan ja tarkastelemaan tunteja, poistamaan tunteja sekä luomaan uusia käyttäjiä. Verkkosovelluksen avulla työnantaja pystyy tarkastelemaan työntekijöiden tekemiä tuntimääriä ja työntekijä pystyy merkitsemään tuntinsa nopeasti ja vaivattomasti laitteesta riippumatta.

Opinnäytetyötä ei toteutettu asiakasprojektina, joten varsinaista palautetta lopputuloksesta ei saatu. Tulevaisuudessa on tarkoitus tarjota tuntikirjanpitojärjestelmää organisaatioille ilmaiseksi testikäyttöön. Käyttäjätestaus on jatkokehityksen kannalta olennaista. Opinnäytetyön tekijä on kuitenkin tyytyväinen tulokseen ja näkee siinä potentiaalia niin jatkokehityksen kannalta kuin mahdollisena tuotteena organisaatioillekin.

Avainsanat (asiasanat)
Express, MongoDB, NodeJS, React, verkkosovellus, Web-kehitys

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Kyyrä, Juuso	Bachelor's thesis	December 2020
		Language of publication: Finnish
	Number of pages 38	Permission for web publication: x

Title of publication

Timesheet web application with modern web technologies

Bachelor's thesis

Degree programme

Information and communications technology

Supervisor(s)

Manninen Pasi, Niemi Kari

Assigned by

-

Abstract

The objective of this bachelor's thesis was the planning and creation of a working hours tracking web application that would compete against pre-existing applications. The pre-existing applications are either pretty expensive or made by using dated technologies, therefore a new working hours tracking application is necessary. The objective was also to create a responsive implementation using the newest possible technologies so that anyone would be able to use the application quickly and easily on any device.

Working hours tracking application is a Node.js based application that uses React.js as a JavaScript library to form a front end. The back end is formed using Express and a MongoDB database. React.js uses an Application Programming Interface API to fetch data from MongoDB through Express. Mongoose is used to parse MongoDB data to JSON format which makes it easier to utilize in front end.

The outcome of bachelor's thesis is a simple and responsive web application. In the web application it is possible to mark hours, examine hours, delete hours and create new accounts. With the application employer will be able to examine hours marked by employee, and the employee will be able to mark hours done very quickly and easily regardless of a device they are using.

The bachelor's thesis was not done for a specific company and therefore it was not possible to obtain user feedback. In the future it would be wise to offer the application for different organizations for free in order to obtain user feedback. The author of the bachelor's thesis is content with the results and see potential in the application as a product and for future development purposes.

Keywords/tags (subjects)

Express, MongoDB, NodeJS, React, web application, web development

Miscellaneous (Confidential information)

Sisältö

Kuviot

1 Työn lähtökohdat

1.1 Taustaa

Tuntikirjanpitojärjestelmiä on olemassa jo useita, mutta ne ovat kalliita. Erityisesti räätälöidyt versiot ja suurempien yritysten käyttämät tuntikirjanpitojärjestelmät maksavat useimmiten käyttäjäkohtaisen kuukausihinnan. Esimerkiksi Visman valmistama Severa, joka sisältää paljon muitakin toiminnallisuuksia kuin tuntikirjanpidon, maksaa halvimmillaan 25 euroa kuukaudessa jokaista käyttäjää kohtaan. (Visma Severa Hinnoittelu. Visma n.d.) Opinnäytetyön aihe keksittiin, kun Espoon Taitoluisteluklubin edustaja otti tekijään yhteyttä tuntikirjanpitojärjestelmään liittyen. Suomalaisilla urheiluseuroilla ei ole varaa maksaa kuukausittaista käyttäjäkohtaista hintaa tuntikirjanpitojärjestelmästä. Jo 20 työntekijän urheiluseura saattaa joutua maksamaan 500 euroa kuukaudessa pelkästään kirjanpitojärjestelmästä. Opinnäytetyön tavoitteena oli valmistaa tuntikirjanpitojärjestelmä, jonka organisaatio pystyy lunastamaan kertamaksulla käyttöönsä.

Opinnäytetyö on toiminnallinen kehittämistyö. Opinnäytetyöprosessin aikana toteutettiin käytännönläheinen projekti. Opinnäytetyötä ei tuotettu toimeksiantajalle.

1.2 Tavoitteet

Opinnäytetyön tavoitteena oli valmistaa nykyisiä tuntikirjanpitojärjestelmiä vastaan kilpaileva oma tuntikirjanpitojärjestelmä. Tuntikirjanpitojärjestelmän täytyi olla pidemmällä tähtäimellä hinnoiteltu huomattavasti edullisemmin kuin tällä hetkellä useimmiten käytetyt järjestelmät.

Verkkosovelluksen oli tarkoituksena olla sulavasti toimiva ja helppokäyttöinen niin puhelimella kuin tietokoneellakin. Hyödynnettävyyttä ajatellen ideaalia olisi, jos järjestelmästä voisi tulla yrityksille tai seuroille päivittäistä työntekemistä helpottava ja

nopeuttava työkalu. Työntekijä pystyisi päivittäin kirjautua sisään ja merkitä tekemänsä tunnit muutamassa sekunnissa. Työnantaja voisi tarkastella työntekijöiden tekemiä tunteja yksinkertaisesti ja helposti.

Tavoitteena olisi myös, että opinnäytetyön tekijä oppisi käyttämään mahdollisimman sulavasti ja laajasti toteutukseen käytettyjä teknologioita ja pystyisi hyödyntämään niitä mahdollisesti tulevaisuudessa niin työelämässä kuin töitä hakiessakin.

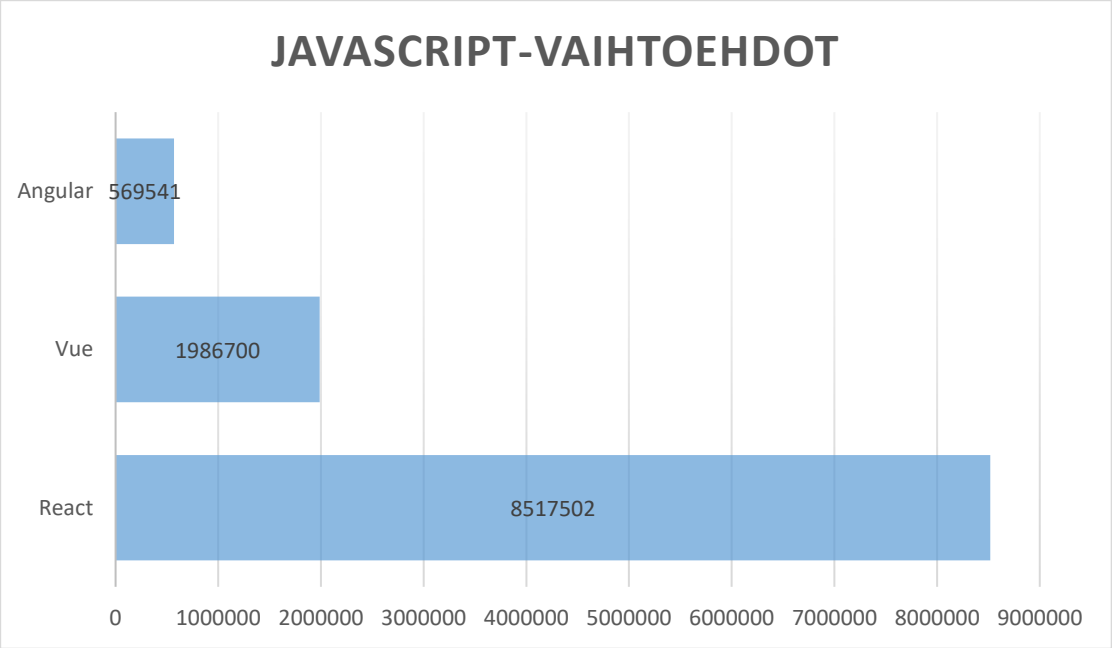
2 Työkalut ja teknologiat

2.1 Valitut teknologiat ja perustelut

Ohjelmoimiseen käytettiin Visual Studio Code -sovellusta Windows 10 -käyttöjärjestelmällä. Visual Studio Code on kätevä, koska siihen saa lisättyä useita lisäosia, jotka helpottavat ja nopeuttavat ohjelmointiprosessia. Opinnäytetyö valmistettiin käyttämällä tekijälle valmiiksi vain vähän tuttuja, mutta haastavia ja hyödyllisiä teknologioita. Valituiden teknologioiden kuuluisi olla hyödyllisiä tulevaisuuden työnhaussa. Opinnäytetyössä tutkittiin erilaisia teknologiapaketteja, joita suositellaan käytettäväksi vuonna 2020. Teknologiapaketti, tunnetumpi termillä ”technology stack” tai ”tech stack”, on termi, jota käytetään yhdistelmästä teknologioita, joita voidaan käyttää yhdessä luomaan hyvä ja monipuolinen kokonaisuus. Esimerkiksi verkkosovellusta varten tarvitaan useimmiten jokin ohjelmointikehys, paketinhallintatyökalu, tietokanta sekä serveripuolen teknologia. Suositelluimpiin ohjelmointikehyksiin sisältyy kolme eri JavaScript-vaihtoehtoa Angular, React ja Vue. (Web development stacks – what stacks (should) we use in 2020? 2020). Suosituin paketinhallintatyökalu on ollut vuoden 2020 alussa ylivoimaisesti Node.js, joten sen valitseminen osaksi käytettävää teknologiapakettia oli helppo päätös (Most used libraries, frameworks, and tools among developers, worldwide, as of early 2020. Statistic 2020).

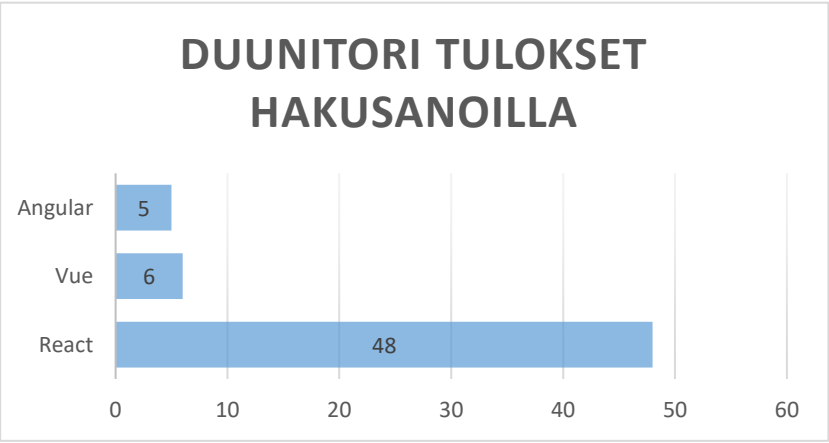
Angularia, Reactia ja Vuea vertaillessa huomattiin käyttäjien keskuudessa selkeä suosikki. Noden kanssa on viimeisen kuuden kuukauden aikana ladattu ylivoimaisesti

eniten Reactia. React on ollut jopa neljä kertaa suositumpi kuin kumpikaan muista vaihtoehdoista. Katso kuvio 1. (Potter 2020.)



Kuvio 1. NPM JavaScript framework lataukset viimeisen kuuden kuukauden ajalta. (Potter 2020).

Myös yleisestä Suomalaisesta työnhakupalvelusta Duunitorista löytyi 48 työpaikkaa, joiden ilmoituksessa mainittiin React. Angular mainittiin vain viidessä ilmoituksessa ja Vue kuudessa. Kuviossa 2 vertaillaan näitä. (Duunitori 2020.)



Kuvio 2. Duunitori tulokset hakusanoilla "React", "Angular" sekä "Vue" (Duunitori 2020)

78 henkilön kyselyn, sekä suorituskykytestin mukaan performanssiltaan vaihtoehdot ovat tasaisia, mutta käyttäjät kokevat Reactin ja Vuen auttavan heidän produktiivisuuttaan sekä säästävän heidän aikaansa huomattavasti enemmän kuin Angularin. Reactin käyttäjistä useammat kuin Vuen käyttäjistä vastasivat, että teknologia toimii heidän haluamallaan tavalla. (Salomaa 2020.)

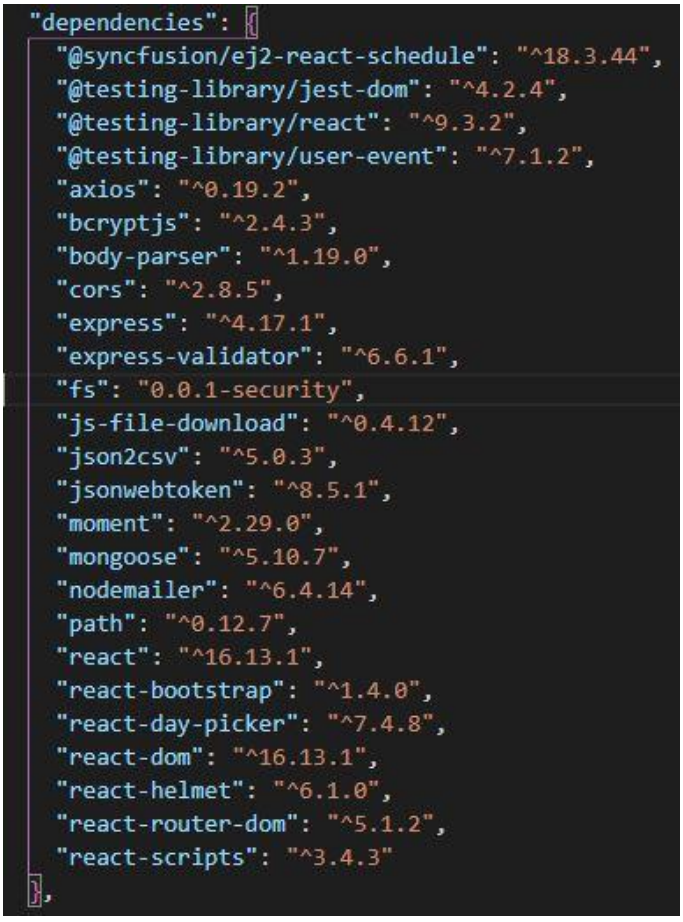
Kokonaan JavaScript-pohjaisen kokonaisuuden käyttämisestä on hyötyä. Etenkin kun backend-teknologiat ovat niin vahvat ja laajasti käytetyt kuin Node, Express ja MongoDB. Ohjelmoijan on helpompi käyttää vain yhtä ohjelmointikieltä usean sijaan. Node ja Express luovat ”backendin” yhdessä. MongoDB toimii NoSQL-tietokantana, johon ”backend” ottaa yhteyttä. React luo käyttöliittymän. Kaikki nämä teknologiat ovat ilmaisia avoimen lähdekoodin JavaScript-pohjaisia teknologioita ja ne luovat yhdessä ”MERN”-stackin. (Hoque 2018.)

2.2 Node

Node on Ryan Dahlin vuonna 2009 valmistama Google Chromen V8 JavaScript -moottorilla rakennettu avoimen lähdekoodin alusta (Node.js -Introduction. tutorialspoint n.d.). Sen avulla pystytään rakentamaan sovelluksia ilman selaimien luomia rajoituksia. Noden päälle rakennetut sovellukset voivat myös skaalautua, mutta pysyä silti kevyinä ja tehokkaina. Verkkosovelluksen valmistamisessa käytetään Node package manager npm:ää, jonka avulla pystyy liittämään useita erinäisten valmistajien tekemiä Node-paketteja sovellukseen. (Hoque 2018.) Verkkosovelluksen projektiin asennettiin esimerkiksi React ja Express käyttämällä Node package manageria.

Muita käytettyjä NPM-paketteja ovat muun muassa mongoose, jsonwebtoken jwt, Bcrypt, body-parser, cors, express-validator, moment, react-day-picker, fs, js-file-download, json2csv sekä ej2-react-schedule. Mongoose on MongoDB:n käyttöä Nodessa helpottava työkalu, joka muuttaa MongoDB:n luoman datan JSON-muotoon ja JSON-muotoisen listan MongoDB:n suositeltuun datamuotoon. Jsonwebtokenin

jwt:n avulla saa tallennettua istunnon tiedot turvallisesti sen ajaksi. Body-parser jäsentelee axios-pyyntöjä. Corsin eli Cross-Origin Resource Sharingin avulla voidaan kertoa selaimelle, että sen täytyy antaa sovellukselle lupa tehdä pyyntöjä eri web-osoitteeseen, kuin missä sovellus sijaitsee. Corsia ei luultavasti tarvita julkaistussa versiossa. Express-validatorin avulla saadaan pyydettyä backendistä muotoiltua virhetietoa. Momentin ja react-day-pickerin avulla saadaan lisättyä päivämäärän valintaa helpottava kalenteri tuntien merkkaamista varten. CSV-tiedoston lataaminen onnistuu ainoastaan Fs, js-filedownload ja json2csv-pakettien ansiosta. Tuntikalenteri rakennettiin ej2-react-schedule-paketin päälle. Kuviossa 3 näkyy käytetyt NPM-paketit.



Kuvio 3. Node package managerin projektiin asennetut teknologiat

2.3 React

React on käyttöliittymän rakentamista varten kehitetty komponenttipohjainen JavaScript-kirjasto. Reactin avulla pystytään luomaan suuria responsiivisia sovelluksia, jotka pystyvät näyttämään muutokset erittäin nopeasti ja tehokkaasti. Reactin käyttäminen pakottaa myös ohjelmoijia kirjoittamaan modulaarista ja helposti uudelleenkäytettävää koodia. (Hoque 2018.) Reactin kehittämistä johtaa Facebookin palkkaama pieni, pelkästään sitä varten luotu yhteisö, mutta sitä auttaa myös useat yksittäiset ohjelmoijat ympäri maailmaa. Reactin on alun perin luonut Jordan Walke vuonna 2013. (Team. React n.d.)

2.4 Express

2.4.1 Yleistä

Express on ohjelmistokehys, jonka avulla pystytään luomaan verkkosovelluksia ja rajapintakutsuja Noden luomalla alustalla. Kaikissa Nodella valmistetuissa verkkosovelluksissa voidaan käyttää Expressiä reititys- ja väliohjelmistona, jolla on myös omia ominaisuuksiaan. Väliohjelmistot käsittelevät HTTP-pyyntöjä ja vastauksia. (Hoque 2018.)

2.4.2 Axios

Axios on työkalu, jolla helpotetaan Noden ja Expressin välistä kommunikaatiota. Sen avulla voidaan tehdä HTTP-pyyntöjä suoraan Nodesta ja käsittelemään pyyntöjä sekä vastauksia. Axios myös muuttaa automaattisesti vastaukset JSON-tiedostomuotoon, joka helpottaa tiedonkäsittelyä sovelluksessa. (Axios. Github n.d.)

2.4.3 Rajapinnat

Expressin avulla luodaan sovellukseen rajapintoja. Sovelluksessa pyydetään rajapintojen avulla tietoja tietokannasta käyttöliittymään. Expressin reitityksen avulla voidaan luoda sovelluksen rajapintapyynnöille vastaukset.

2.5 MongoDB

MongoDB on NoSQL-tietokanta. Se on hyvin laajasti skaalautuva ja helposti ”lennossa” päivitettävä tietokanta (Hoque 2018). Sovelluksen valmistuksessa käytetään MongoDB Cloudia, jonka myötä tietokanta on tallennettuna pilveen ja siihen voi saada yhteyden miltä tahansa haluamalta tietokoneelta käyttäjätunnusta ja salasanaa vastaan. MongoDB:n tietokannaksi valittua ei tarvitse suunnitella tietokantaa ja sen sisäisiä relaatioita yhtä tarkasti kuin esimerkiksi MySQL:ää käytettäessä tarvitsisi. MongoDB:ssä kaikki tieto on samassa kasassa Collectionin sisällä. Esimerkiksi kaikki käyttäjät lisätään samaan Collectioniin ”users” ja sieltä pystytään hakemaan tietoa esimerkiksi automaattisesti luodun id:n, timestampin tai sähköpostin perusteella.

2.6 HTML

Kaikki Reactin käyttöliittymässä näkyvä on tuotu esille HTML:n avulla. HTML eli HyperText Markup Language on yleisin verkkosivustojen merkintäkieli. HTML:n avulla verkkosivustoille pystytään luomaan erilaisia visuaalisia elementtejä. (HTML: HyperText Markup Language. MDN web docs n.d.)

3 Vertailu valmiisiin tuntikirjanpitojärjestelmiin

3.1 Valintaperusteet

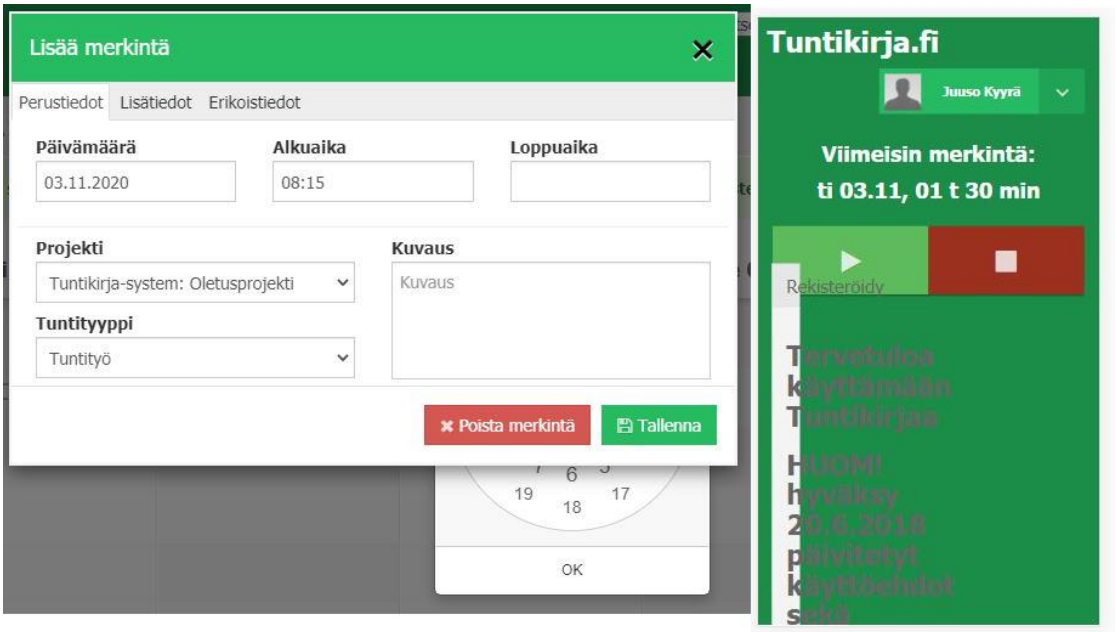
Käyttäjäkokemus on hyvin tärkeä osa-alue päivittäin käytettävää verkkopalvelua valittaessa. Heikko käyttäjäkokemus aiheuttaa käyttäjissä negatiivisia ajatuksia ja heikentää heidän haluaan käyttää verkkopalvelua. Tärkeintä käyttäjäkokemuksen kannalta on valmistaa helposti ja nopeasti käytettävä palvelu. Verkkopalvelun täytyy nykyaikana toimia hyvin myös mobiililaitteilla. (Boyer 2018.)

Hinta on myös erittäin suuressa arvossa verkkopalvelua valittaessa. Esimerkiksi Espoon Taitoluisteluklubin edustajan mukaan Suomalaisilla urheiluseuroilla ei ole varaa

maksaa kuukausittaista käyttäjäkohtaista hintaa tuntikirjanpitojärjestelmästä. Urheiluseuroilla ja muilla voittoa tavoittelemattomilla organisaatioilla on usein rahan puolesta rajoituksia, joita olemassa olevat tuntikirjanpitojärjestelmät eivät ota huomioon hinnoitteluissaan.

3.2 Tuntikirja.fi

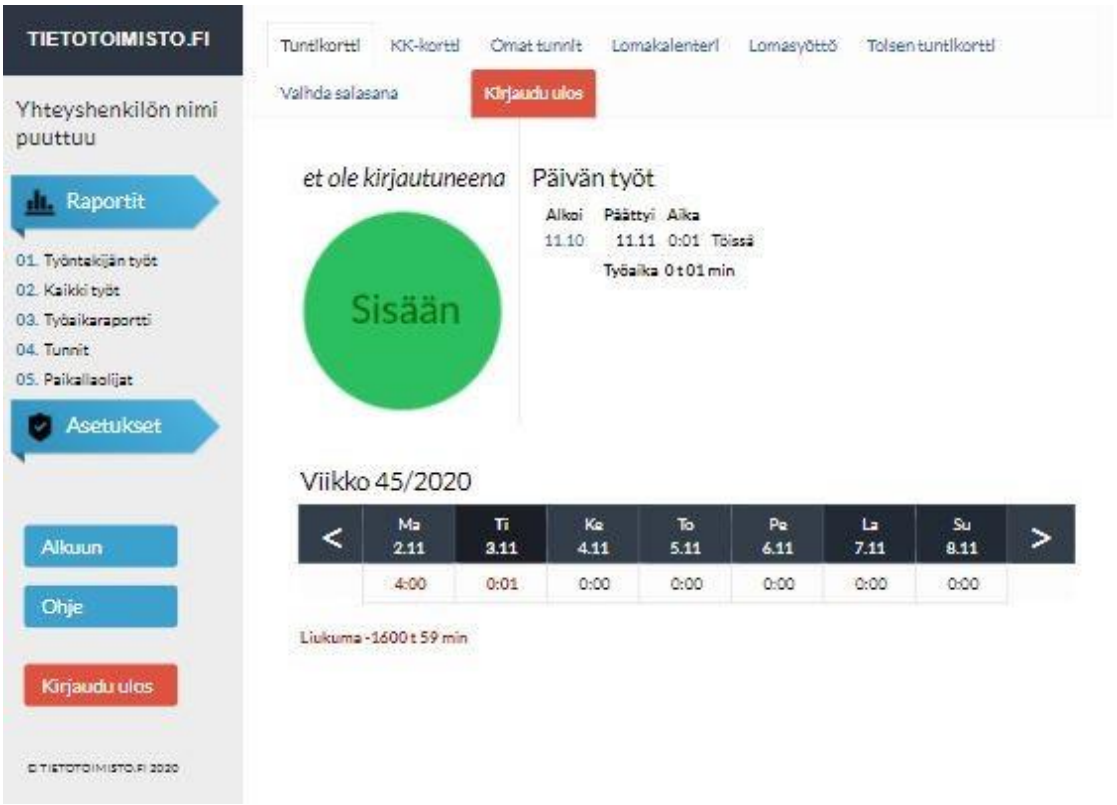
Tuntikirja.fi on ebrand Group Oy:n toteuttama verkkopalvelu. Tuntikirja.fi:n avulla pystyy kirjaamaan tunteja niin työntekijöille kuin yrityksen johdollekin. Tuntikirja.fi on suhteellisen edullinen maksaessaan vain 5 € kuukaudessa alle 20 käyttäjän organisaatiolle. Yli 20 käyttäjän yritykselle täytyy pyytää erillistä hintaa. (Hinnoittelu. Tuntikirja n.d.) Tuntikirjanpito.fi ei kuitenkaan ole käyttäjäkokemukseltaan laadukas. Palvelussa on pakko asettaa kuvaus työpäivälle, tuntien valintaa ei saa ajoittain auki, ja kuten kuviossa 4 näkyy, mobiiliresponsiivisuus ei ole selkeästikään ollut keskeisessä roolissa palvelun suunnittelemisen aikana.



Kuvio 4. Kuvakaappaukset kellon toiminnasta sekä mobiilinäkymästä.

3.3 Tietotoimisto.fi Työaika

Tietotoimisto.fi:n Työaika-palvelu on tietotoimiston tarjoama työajanseurantapalvelu. Työaika maksaa alle kymmenen käyttäjän organisaatiolle 7 € kuukaudessa käyttäjää kohtaan. Yli kymmenen käyttäjän organisaation täytyy pyytää erillinen tarjous palvelusta. (Työaika. Tietotoimisto n.d.) Työaika-palvelussa pystyy merkitsemään tunteja samalle päivälle vain leimaustyyliin, joten siellä pitäisi käydä joka päivä vähintään kaksi kertaa, mutta myös ennen ruokataukoa, mikäli sitä ei lasketa työajaksi. Jatkuva pakko käyttää järjestelmää ei ole kovin käyttäjäystävällistä. Mobiiliresponsiivisuutta ei ole nimeksikään kuten kuviosta 5 voidaan nähdä.

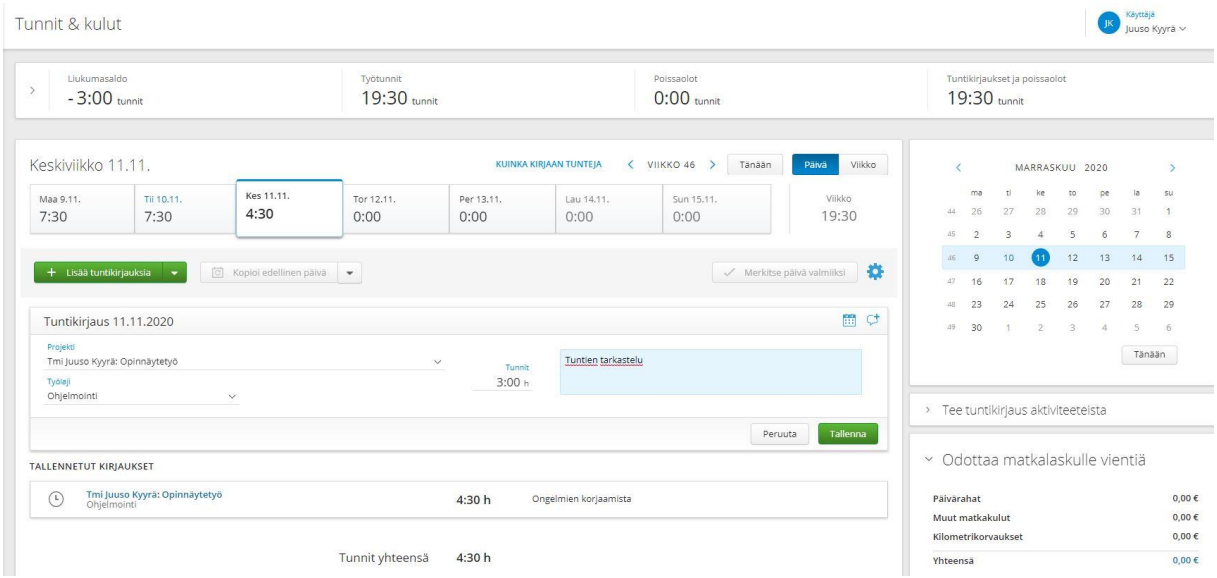


Kuvio 5. Työaika-palvelun näkymä mobiililaitteella.

3.4 Visma Severa

Severa on Visma Solutions Oy:n tarjoama projektinhallintatyökalu. Se on hyvin monipuolinen ja monta eri palvelua tarjoava työkalu. Siihen sisältyy tuntikirjanpidon lisäksi

esimerkiksi laskutus, raportointi, projektinhallinta sekä toiminnanohjausjärjestelmä. Monipuolisuus kuitenkin näkyy myös hinnassa ja Severa on kaikista vertailluista palveluista selkeästi kallein. Se maksaa halvimmillaan 25 € kuukaudessa käyttäjää kohtaan. Severaan tuntien kirjaaminen on helppoa ja yksinkertaista kuten kuviosta 6 voidaan nähdä. (Visma Severa Hinnoittelu. Visma n.d.)

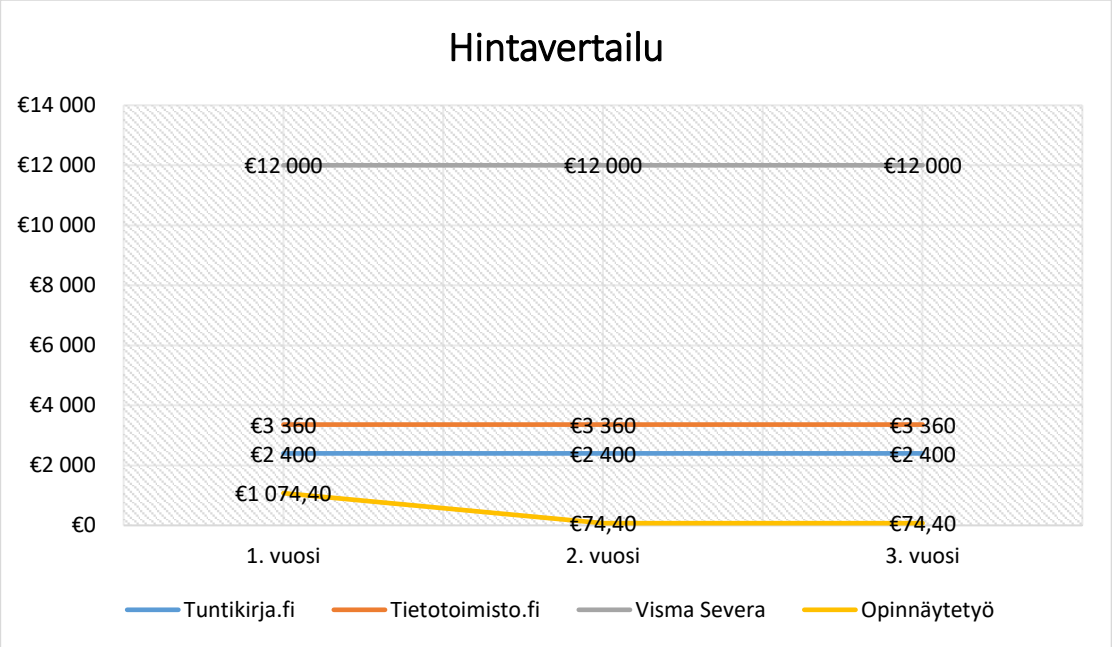


Kuvio 6. Visma Severa. Tunnit ja kulut -osio.

3.5 Perustelut oman järjestelmän toteuttamiseen

Suurin perustelu oman tuntikirjanpitojärjestelmän toteuttamiseen on hinta. Valmiit tuntikirjanpitojärjestelmät ovat pitkällä aikavälillä hyvin kalliita ja esimerkiksi 40 työntekijän organisaatio säästäisi jo ensimmäisen käyttövuoden aikana 1325,6 € verrattuna halvimpaan 5 € kuukaudessa työntekijää kohtaan maksavaan palveluun. Seuraavina vuosina säästö olisi 2325,6 € vuodessa. Kuvio 7 näyttää hintavertailun. Valmii-

den tuntikirjanpitojärjestelmien hinnat ovat liian kalliit etenkin voittoa tavoittelemattomille organisaatioille. Edullisimmat valmiit järjestelmät ovat myös vanhentuneilla teknologioilla tehtyjä ja kaipaisivat päivitystä.



Kuvio 7. Hintavertailua valmiina olevien palveluiden ja opinnäytetyön välillä.

Toinen perusteluista on sivuston responsiivisuus ja hyvän käyttäjäkokemuksen luominen. Suomalaisista jopa 80 prosenttia käyttää internetiä matkapuhelimella. (Suomen virallinen tilasto. Suomalaisten internetin käyttö 2019). Mobiiliresponsiivisuus on siis erittäin tärkeä asia ja erityisesti organisaatiot, joiden työntekijöistä suuri osa tekee töitä ilman tietokonetta, tarvitsevat mobiililaitteella hyvin ja helposti käytettävän tuntikirjanpitojärjestelmän. Mikäli organisaatio laittaa työntekijät kirjaamaan päivittäin tunnit, olisi tuntien kirjaamisen hyvä olla mahdollisimman vaivatonta ja nopeaa.

3.6 Millä järjestelmä erottuu valmiista järjestelmistä

Hinta tulee olemaan suurin mitattava ero. Kuukausittainen hinta muissa tuntikirjanpitojärjestelmissä ei ole oikeutettu. Verkkopalvelimen hinta on suhteellisen edullinen. Cloudcitystä voi esimerkiksi saada Suomalaiselta yritykseltä tarpeeksi hyvän palvelimen verkkosovellukselle 6,20 € kuukaudessa. (Hinnasto. Cloudcity n.d.) MongoDB cloud on ilmainen 512mb asti ja maksaa $9 kuukaudessa 2gb asti. Tuntikirjanpitojärjestelmän käytössä yksi merkkaus vie tilaa alle 0.15 kb ja 512 mb sisältää 512 000 kilobittiä. (Pricing. MongoDB n.d.) Merkintöjä voi siis ilmaisella versiolla olla sovelluksella lähes 3,5 miljoonaa kappaletta. Organisaatiolta pystyisi siis laskuttamaan 6,20 € kuukaudessa, kunnes organisaatio on lisännyt merkintöjä tilan loppumiseen saakka ja siinä vaiheessa ruveta pyytämään $9 kuukaudessa lisää tietokannasta. Toinen, luultavasti järkevämpi vaihtoehto, olisi tallentaa vuosittain kaikki merkinnät muualle ja poistaa kaikki yli vuoden vanhat merkinnät. Opinnäytetyössä valmistettavaa verkkosovellusta on tarkoitus myydä käyttöön 1000 € kertahinnalla. Kertahinnan jälkeen organisaatiolle jää maksettavaksi vain verkkopalvelimen hinta sekä mahdollinen tietokannan kalliimpi versio, mikäli asiakas käyttää tuntikirjanpitojärjestelmää hyvin paljon ja pitkään.

Responsiivisuus tulee olemaan keskinäinen osa verkkosovelluksen suunnittelua ja toteutusta. Verkkosovelluksen tavoitteena on olla helposti ja nopeasti päivittäin käytettävä. Responsiivisuus ei ole helposti mitattavissa, mutta verkkopalvelun toimintaa erilaisilla laitteilla on mahdollista kokeilla esimerkiksi Google Chromen Inspect -työkalun avulla.

4 Tuntikirjanpitojärjestelmän suunnitteleminen

4.1 Rautalankamalli ja rakennesuunnitelma

Sivuston rakenteen suunnitteleminen täytyy tehdä ajoissa. Sen tehtävänä on selittää yhteydet sivuston tärkeimpien alueiden välillä. Rakennesuunnitelman tehtävänä on

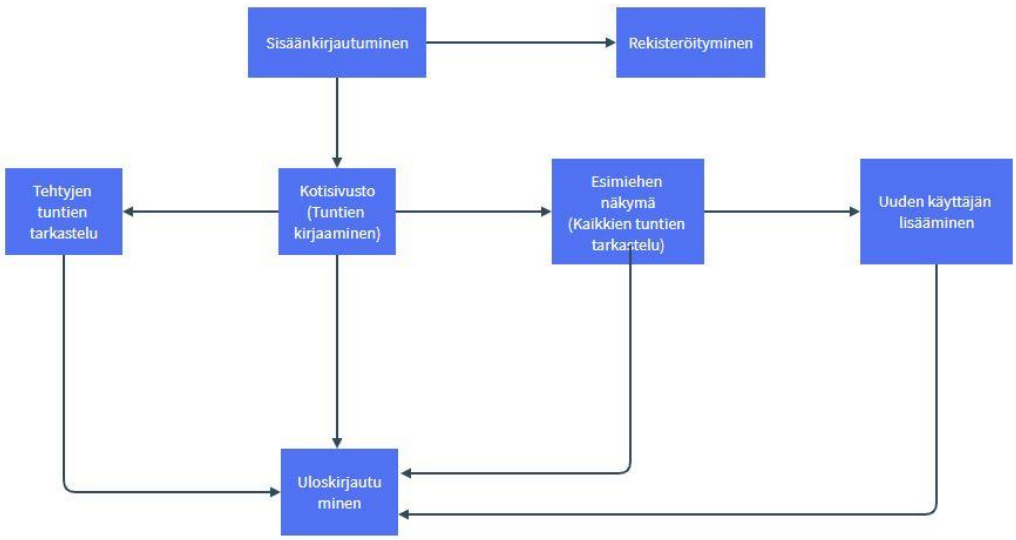
kertoa miltä sivuston rakenne tulee näyttämään, mutta sen tehtävänä ei ole kuvailla käyttöliittymää tai sen ulkonäköä. (Gordiyenko 2015.) Rakennesuunnitelman avulla pystytään suunnittelemaan selkeä rakenne, jotta käyttäjillä olisi mahdollisimman helppo ja hyvä käyttökokemus. Rakennesuunnitelman tekemisen aloitettaessa on tärkeä tehdä aluksi lista kaikista verkkosovelluksen sivustoista ja reiteistä niille, sekä sivustoista, joihin pääsee suoraan navigaatiopalkin avulla. (Duffy 2018.)

Rakennesuunnitelman tekemiseen käytettiin Moqups-nimistä verkkosovellusta. Siitä löytyi ilmainen yhden projektin sisältävä versio. (Pricing. Moqups n.d.)

Sivuston rakennesuunnitelmaan sisältyvät seuraavat sivustot:

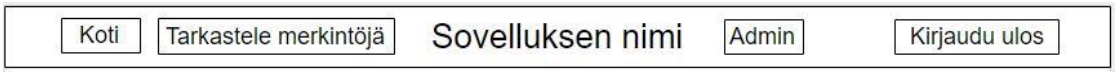
- Sisäänkirjautuminen
- Rekisteröityminen
- Kotisivusto (tuntien kirjaaminen)
- Tehtyjen tuntien tarkastelu
- Esimiehen näkymä
- Uuden käyttäjän lisääminen
- Uloskirjautuminen

Kuviossa 8 on näkyvissä opinnäytetyössä käytettävä rautalankasuunnitelma.



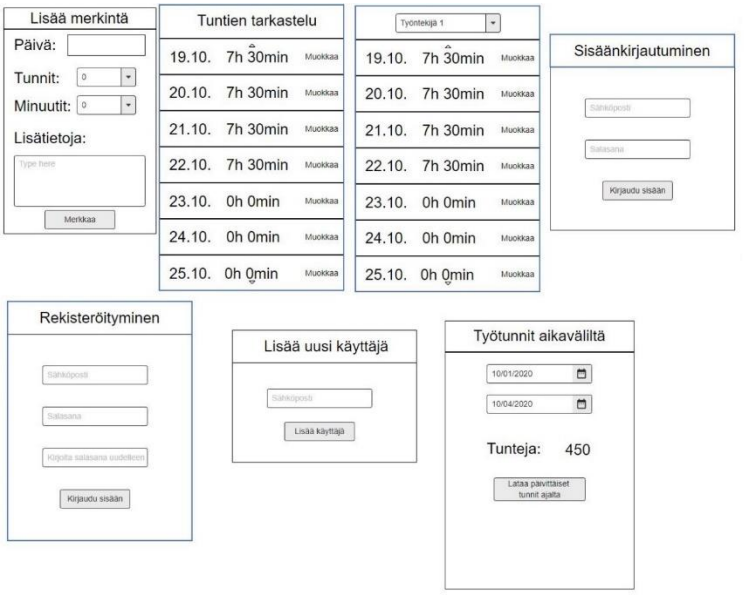
Kuvio 8. Moqupsilla tehty rakennesuunnitelma

Sivusto tarvitsee ennen sen valmistamista myös rautalankamallin. Rautalankamallin ero rakennesuunnitelmaan on se, että rautalankamallissa esitetään myös jokaisen sivuston pääkomponentit. Rautalankamallin pystyy jakamaan kolmeen eri osaan. Ensimmäinen niistä on informaatioarkkitehtuuri, jonka tehtävä on hyvin samanlainen kuin rakennesuunnitelman tehtävä. Toinen osa on navigaatio/rakenne. Navigaation suunnitteleminen on hyvin tärkeää, jotta käyttäjät pystyvät liikkumaan verkkosovelluksen sisällä helposti ja vapaasti. Kolmas osa on asettelusuunnitelma. (Costa 2019.) Kuviossa 9 on sovelluksen yläpalkki, jossa näkyy navigaatio tietokoneella.



Kuvio 9. Sovelluksen yläpalkki

Asettelusuunnitelmassa rautalankamalliin lisätään muutama visuaalinen elementti ilman tarkempaa visuaalista suunnittelua. Sivustolle tulee useita komponentteja, jotka pyritään valmistamaan yhtenevän tyylin mukaan. Komponenttien rautalankamalleja näkyy kuviossa 10. Yhtenevä tyyli helpottaa käytettävyyttä ja tekee sivuston ulkonäöstä myös selkeän.

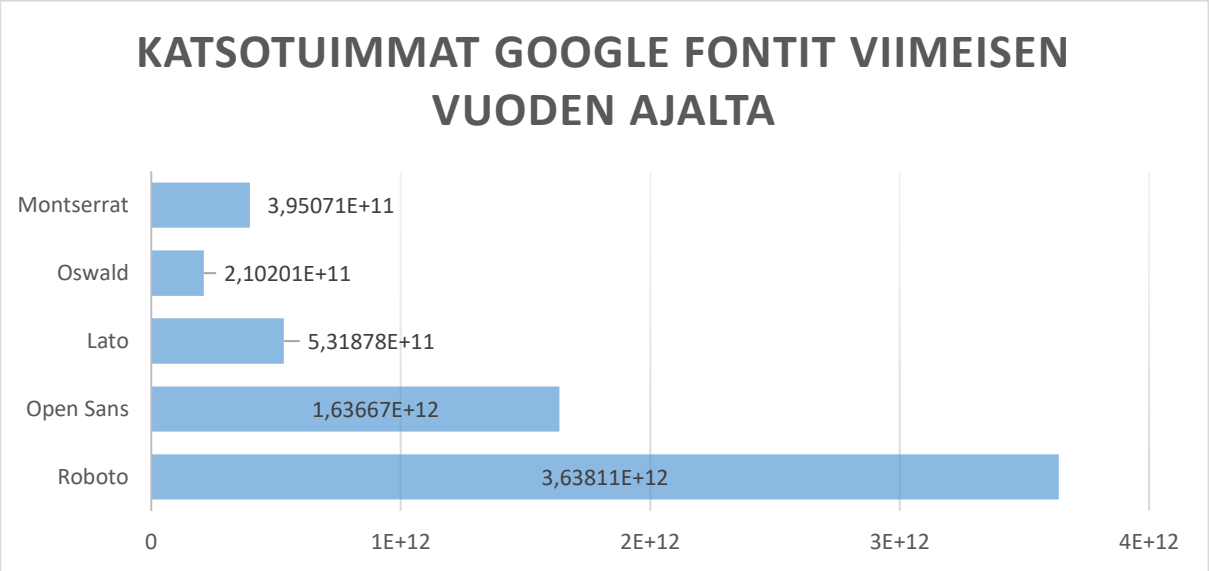


Kuvio 10. Komponenttien rautalankamalleja.

4.2 Visuaalinen pohja

4.2.1 Yleistä

Visuaalinen pohja ja sen suunnitteleminen on tärkeä tehdä, jotta käyttäjän kokemus sovellusta käytettäessä olisi mahdollisimman mieluisa. Olennainen osa verkkosovelluksen ulkonäköä on fontti. Tuntikirjanpitojärjestelmään etsittiin suosituin ja katsotuin Google Fontti viimeisen vuoden ajalta. (Ks. kuvio 11.) Se on nimeltään Roboto. (Analytics. Google Fonts n.d.)



Kuvio 11. Katsotuimmat Google Fontit viimeisen vuoden ajalta (Analytics. Google Fonts).

4.2.2 Värin valitseminen

Värin valitseminen verkkosivustolle voi olla haastavaa. Sivustolla täytyy olla ainakin kolme pääväriä mukaan lukien dominoiva väri, taustaväri sekä korostusväri. Dominoivan värin olisi hyvä löytyä myös logosta, mikäli sellainen löytyy. Dominoivaa väriä ei saa löytyä kaikkialta sivulta, vain ainoastaan paikoista, joihin haluaa käyttäjän kiinnittävän huomionsa. Korostusvärien kannattaa olla tarpeeksi erilainen taustaväristä, että sen erottaa, mutta se ei myöskään saisi hyppiä silmille. (Wong 2019.)

Verkkopalvelun väriyhdistelmä valikoitui ohjelmoijan visuaalisen silmän mukaan. Ohjelmoija halusi yksinkertaisen ja vaimean värimaailman. Tässä väriyhdistelmässä on kahta eri harmaan sävyä sekä punaista. (Ks. kuvio 12.) Niistä taustavärinä toimii tummempi harmaa, korostusvärinä vaaleampi harmaa sekä dominoivana värinä punainen.

Kuvio 12. Värit ja niiden hexakoodit.

4.2.3 Brändin valmistaminen

Brändin nimen on hyvä olla yksinkertainen, lyhyt ja liittyä tuotteeseen. Se ei saa olla jo valmiiksi rekisteröity. (Brand Building 101: How to Build a Brand. The Daily Egg 2020.) Patentti- ja rekisterihallituksen sivuilta löytyy hakupalvelu, jolla pystyy etsimään yritysten ja tavaramerkkien nimiä (Yrityksen nimipalvelu. Patentti- ja rekisterihallitus n.d.). Hakupalvelun kautta ei löytynyt ensimmäisenä kokeillulla ”Tunnit” -nimellä yhtään valmiiksi rekisteröityä tavaramerkkiä tai yritystä. Täten ”Tunnit” valikoitui verkkosovelluksen nimeksi.

Logon olisi hyvä olla oikeissa väreissä ja liittyä jollain tavalla tuotteeseen. Logon täytyy myös olla uniikki ja helposti muistettava. (Peate 2017.) Verkkosovelluksen logo luotiin yksinkertaisesti muokkaamalla valmiilla fontilla ”Bauhaus 93” luotua tekstiä. Kuviossa 13 nähdään luotu logo.

Kuvio 13. Logo.

4.3 Tietoturva

Verkkosovellus käyttää istunnon tokenina satunnaisesti luotua merkkijonoa. Salasanan suojatakseen sovellus tallentaa salasanan hyödyntäen bcryptiä. Salasanaa sovelluksesta ei myöskään näe network-tabin tai selaimen konsolin kautta. Bcrypt on Niels Provoksen ja David Mazièresin suunnittelema salausfunktio. Bcrypt skaalautuu laitteiston nopeuden mukaan ja pystyy siten estämään myös hyökkääjien mahdollisesti nopeamman laitteiston tuoman edun. Bcrypt käyttää 128-bittistä ”salt”:a ja salaa 192-bittisen salasana-arvon. Vaikka Bcrypt on tehokas ja hyvä salaus, käyttäjältä vaaditaan sivustolla silti vähintään kahdeksan merkin pituista salasanaa. (Arias 2018). “Brute Force” -hyökkäyksen varulta olisi myös hyvä, jos verkkosovellus pystyisi lukitsemaan tietyn käyttäjän tai ip-osoitteen joksikin ajaksi muutaman epäonnistuneen kirjautumisyrityksen jälkeen. (Ali 2018.)

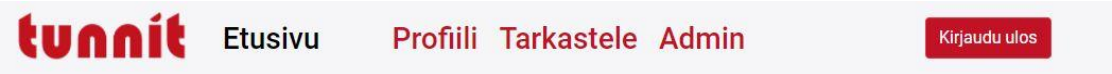
Tiedot täytyy tallentaa EU:n tietosuoja-asetuksen mukaan, koska sovellus käsittelee henkilötietoja kuten sähköpostia. (Usein kysyttyä EU:n tietosuoja-asetuksesta. Tietosuojavaltuutetun toimisto n.d.) Tämä tarkoittaa, että sovelluksen täytyy kertoa käyttäjille mitä tietoja se kerää, kuka käsittelee niitä tietoja ja miksi. Henkilö voi milloin vaan pyytää tietojaan poistetuksi tai muutetuksi. Henkilötietojen käsittely täytyy myös dokumentoida, mikäli sovellus julkaistaan. (Yleinen tietosuoja-asetus. Euroopan Unioni n.d.)

5 Tuntikirjanpitojärjestelmän toteuttaminen

5.1 Valikko

Valikon avulla pystytään navigoimaan verkkosovelluksen sisällä. Valikkoon sisältyy logo, aktiivisen sivuston nimi, linkit muille sivuille sekä uloskirjautumis-painike. (Ks. kuvio 14.) Admin-sivustolle tulee linkki vain, jos sisäänkirjautunut käyttäjä on admin.

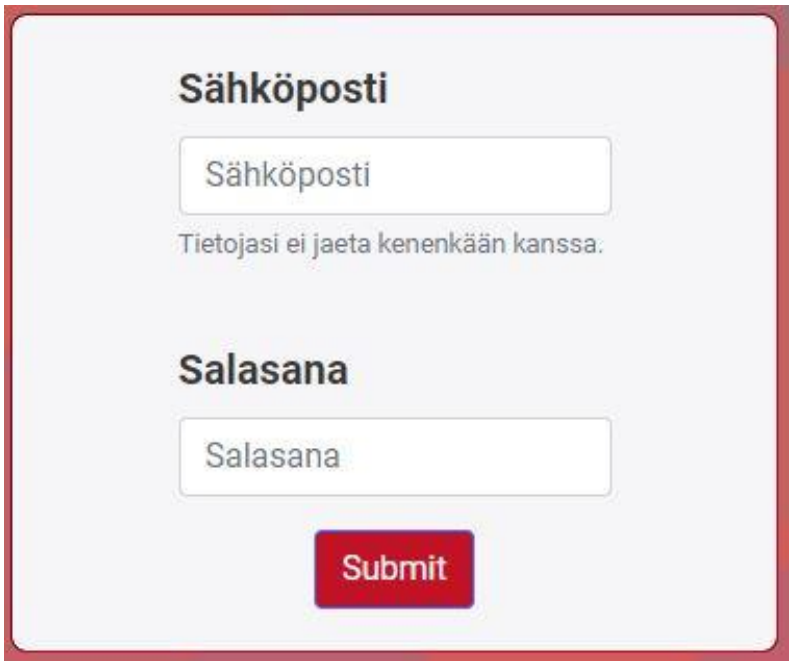
Valikon linkit ovat toteutettu react-router-dom npm:llä. Valikon taustaväriksi on valikoitunut verkkosovelluksen taustaväri ja sekä linkeissä, että uloskirjautumisnapissakin käytetään korostusväriä.



Kuvio 14. Kuvakaappaus valikosta.

5.2 Sisäänkirjautuminen

Sisäänkirjautuminen tapahtuu React-komponentin avulla. Sisäänkirjautumiskomponentti on myös verkkosovelluksen ”etusivu”, johon sovellus ohjaa kaikki sivustolle saapuvat käyttäjät. Sisäänkirjautuminen on käyttäjän silmissä yksinkertainen ja selkeä. Se on perinteinen sisäänkirjautumislomake, jonka lähettämällä käyttäjä voi kirjautua sisään. (Ks. kuvio 15.) Lomake pyytää käyttäjän sähköpostin ja salasanan.



Kuvio 15. Kuvakaappaus sisäänkirjautumisesta.

Käyttäjän painettua nappia ja lähetettyä lomakkeen, verkkosovelluksen React ottaa yhteyttä Axioksen avulla Expressiin, jonka avulla käyttäjän sähköposti ja salasana tarkistetaan tietokannasta. Tarkistuksen jälkeen käyttäjälle luodaan ”session access token” jwt:n avulla. Jwt on Json Web Token, joka tallentaa sisäänkirjautuneen käyttäjän tiedot salatusti istunnon ajaksi sovellukseen. Jwt:lle on asetettu vanhentumisaika, jonka avulla käyttäjä voidaan kirjata ulos, mikäli hän ei ole tehnyt mitään sivustolla pitkään aikaan. Mikäli salasanan ja sähköpostin tarkistus ei mene läpi, verkkosovellus näyttää käyttäjälle virheilmoituksen. Kuviossa 16 näkyy verkkosovellukselle perinteinen rajapintapyyntö. Kyseisen rajapintapyynnön tarkoituksena on uusia jwt sekä hakea sisäänkirjautuneen käyttäjän tietoja tietokannasta.

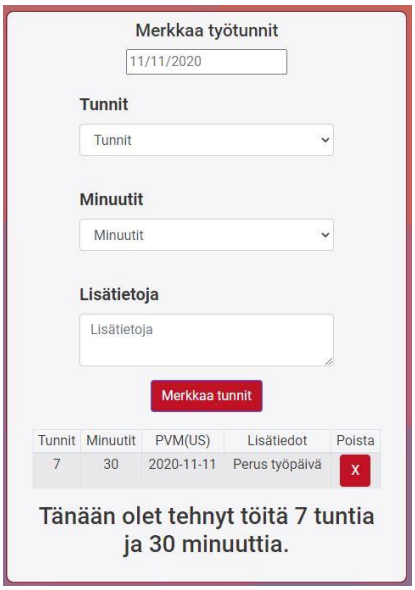
```
axios
  .get(API_BASE_URL + "/user/me", {
    headers: { token: localStorage.getItem(ACCESS_TOKEN_NAME) },
  })
  .then(function (response) {
    if (state.userid === undefined) {
      const userid = response.data.response;
      const useremail = response.data.email;

      setState({
        userid,
        useremail,
        selectedDay: state.selectedDay
       });
    }
    if (response.status !== 200) {
      redirectToLogin();
    }
  })
  .catch(function (error) {
    window.alert("Sessio aikakatkaistu. Palautetaan sisäänkirjautumiseen.");

    redirectToLogin();
  });
```

Kuvio 16. Kuvakaappaus rajapintapyynnöstä.

5.3 Tuntien kirjaaminen

Tuntien kirjaaminen on osa verkkosovelluksen kotisivustoa. Tuntien kirjaaminen toimii lomakkeen avulla. (Ks. kuvio 17.) Tunteja kirjatessa valitaan ensimmäiseksi päivä React day picker -komponentilla. Päivä täytyy valita vain, mikäli haluaa valita jonkun muun päivän kuin päivän, jona tunnit kirjataan. Mikäli päivää ei valita erikseen, verkkosovellus asettaa merkkauspäivän päivämääräksi. Toiseksi valitaan tunnit ja kolmanneksi minuutit valintakentillä. Halutessaan käyttäjä pystyy kirjaamaan myös lisätietoja päivästä textarea-komponenttiin.

Tuntien kirjaaminen hakee ensiksi Axiosin ja Expressin avulla tietokannasta sisäänkirjautuneen käyttäjän sähköpostin ja id:n. React tallentaa lomakkeen kenttiin tapahtuvat muutokset useStaten avulla muuttujiksi, jotka lähetetään Axiosin ja Expressin avulla tietokantaan käyttäjän lähettäessä lomakkeen. Lomakkeen alapuolella näkyvät kyseisen päivän valmiiksi merkatut tunnit ja tehty työmäärä. Merkatut tunnit näkyvät taulukossa, josta pystyy myös poistamaan tunnit helposti ja nopeasti.



Kuvio 17. Kuvakaappaus tuntien kirjaamisesta.

Kyseisen päivän merkatut tunnit haetaan tietokannasta listaan, joka tulostetaan taulukkoon. Päivän kokonaistuntimäärä saadaan laskettua funktiolla, joka muuttaa minuutit tunneiksi ja minuuteiksi, mikäli niitä on päivänä 60 tai enemmän. Kuviossa 18 nähdään kuvakaappaus tästä prosessista.

```
var arraylength = this.state.response.length;
let tuntisumma = 0;
let minuuttisumma = 0;
for (var i = 0; i < arraylength; i++) {
  this.state.tuntiarray.push(this.state.response[i].hours);
  this.state.minuuttiarray.push(this.state.response[i].minutes);
 // console.log(this.state.tuntiarray);
 // console.log(this.state.minuuttiarray);
  tuntisumma += this.state.response[i].hours;
  minuuttisumma += this.state.response[i].minutes;
}
function time_convert(num) {
  var hours = Math.floor(num / 60);
  var minutes = num % 60;
  return {
    hours: hours,
    minutes: minutes,
  };
}
if (minuuttisumma >= 60) {
 // console.log(time_convert(minuuttisumma));
  var minuutit = time_convert(minuuttisumma);
  var ekstratunnit = minuutit.hours;
  var ekstraminuutit = minuutit.minutes;
  if (ekstratunnit > 0) {
    tuntisumma = tuntisumma + ekstratunnit;
    this.setState({
      minuutit: ekstraminuutit,
      tunnit: tuntisumma,
    });
```

Kuvio 18. Kuvakaappaus tuntien ja minuuttien yhteenlaskusta.

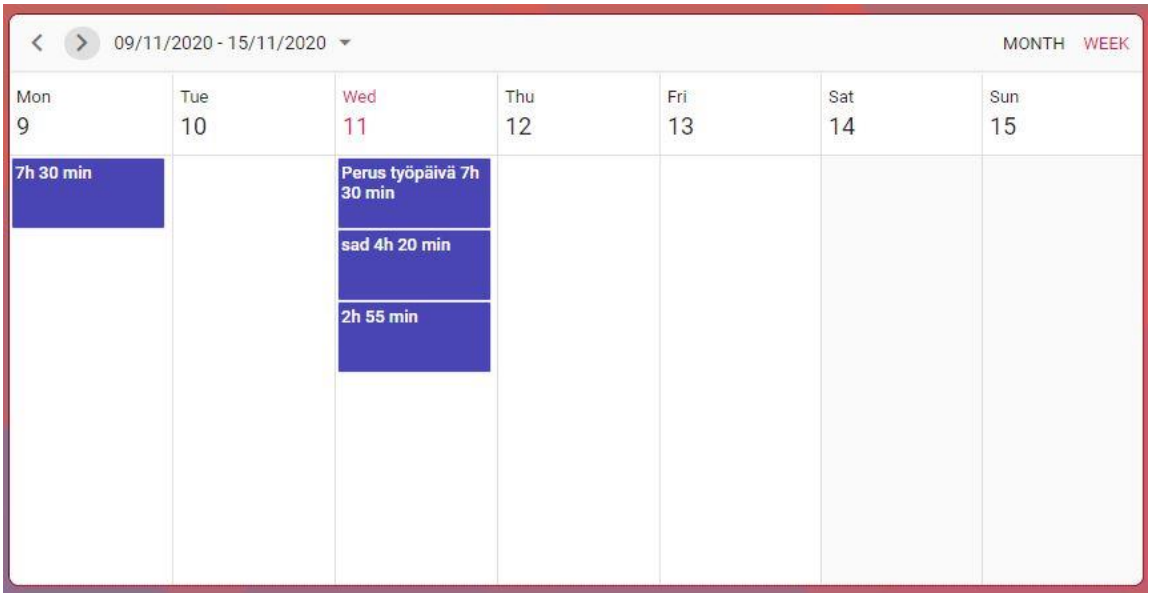
5.4 Tuntien tarkasteleminen

5.4.1 Yleistä

Tuntien tarkasteleminen on oma sivusto verkkosovelluksessa. Tuntien tarkasteleminen -sivulla on neljä komponenttia. Jokaisella komponentilla on oma tehtävänsä.

5.4.2 Kalenteri

Kalenterin avulla pystytään tarkastelemaan tehtyjä tunteja viikko- tai kuukausitasolla. Se on valmistettu hyödyntäen react schedule -komponenttia, joka on suunniteltu käytettäväksi aikatauluna. Verkkosovellus hakee sisäänkirjautuneen käyttäjän kaikki tuntikirjaukset ja asettaa ne kalenteriin. Kuviossa 19 näkyy kalenterin ulkonäkö.



Kuvio 19. Kuvakaappaus kalenterista.

5.4.3 Tuntimäärä ja tuntien lataaminen tietyltä aikaväliltä

Tuntimäärä ja tuntien lataaminen tietyltä aikaväliltä -komponentin avulla käyttäjä pystyy tarkastelemaan tekemäänsä työmäärää haluamaltaan aikaväliltä. Aikavälin valitseminen tapahtuu react day picker -komponentin avustuksella. Käyttäjän valittua

aikavälin verkkosovellus laskee ja näyttää kaikki käyttäjän tekemät työtunnit kyseiseltä aikaväliltä. Käyttäjä pystyy halutessaan luomaan CSV-tiedoston kyseisen aikavälin tehdyistä tunneista ja lataamaan sen. Kuviossa 20 näkyy komponentin ulkonäkö.

Kuvio 20. Tuntimäärä ja tuntien lataaminen aikaväliltä -komponentti.

CSV-tiedoston luominen tapahtuu lähettämällä Reactista Axiosin avulla rajapintapyyntö Expressiin. Rajapintapyynnön mukana lähetetään sisäänkirjautuneen käyttäjän userid sekä valitut päivämäärät. Expressissä tallennetaan lähetetyt tiedot muuttujiin ja tehdään niitä hyödyntämällä haku tietokannasta. Haun jälkeen asetetaan CSV-tiedostoon halutut kentät ja tallennetaan muuttujaan tietokantahaun tulokset CSV-tiedostomuodossa. Viimeisenä CSV-tiedosto tallennetaan fs:n avulla haluttuun kansioon ja nimetään userid:n mukaan.

Expressiin otetaan yhteyttä rajapinnan kautta. Esimerkiksi CSV-tiedoston luomista varten verkkosovellus lähettää Axios-pyynnön sivuston web-osoitteeseen tyylillä

”web-osoite/mark/download”. Web-osoitteen tilalle tulee sivuston osoite. Rajapinta-pyyntö vastaa pyyntöön joko epäonnistuessaan virheilmoituksella ja koodilla 500 tai onnistuessaan statuskoodilla 200 sekä palauttamalla json-tiedostona luodun tiedoston nimen. Kuviossa 21 nähdään kuvakaappaus Expressin koodista.

```
router.route('/download').get((req, res) => {

    const userid = req.query.userid;
    let pvm1 = req.query.pvm1;
    let pvm2 = req.query.pvm2;
    pvm1 = new Date(pvm1+"Z");
    pvm2 = new Date(pvm2+"Z");
    Mark.find({'userid':userid, 'date':{$gte: pvm1, $lt: pvm2}}).lean().exec(function (err, data) {
        if(err) {
            return (err)
        } else {
            //let testi = JSON.stringify(data);
            //console.log(testi);
            let csv;
            try {
                const csvFields = ['_id', 'hours', 'minutes', 'textarea', 'date', 'userid', 'useremail'];
                const json2csvParser = new Json2csvParser({csvFields});
                csv = json2csvParser.parse(data);
                //console.log(csv);
            }
            catch (err) {
                return res.status(500).json({err});
            }
            fs.writeFile("../oppari/public/files/" + userid +".csv" , csv, (err) => {
                if (err) {
                    console.log("Rippendahlen");
                    return res.json(err).status(500);
                }
                else {
                    //console.log("tiedoston luominen onnistui");
                    return res.json(`/exports/${userid}.csv`);
                }
```

Kuvio 21. Kuvakaappaus CSV-tiedoston luomisesta.

5.4.4 Viimeiset kymmenen merkkausta ja niiden poistaminen

Viimeiset kymmenen merkkausta ja niiden poistaminen -komponentti näyttää käyttäjälle viimeiset kymmenen merkkausta päivämäärän perusteella. Merkkaukset näkyvät taulukossa, joka sisältää tunnit, minuutit, päivämäärän, lisätiedot sekä poista -kentät. Käyttäjä pystyy halutessaan poistamaan minkä tahansa haluamistaan merkkauksista. Verkkosovellus hakee tietokannasta rajapintapyynnön avulla sisäänkirjautuneen käyttäjän viimeiset 10 uusinta tuntikirjaamista päivämäärän mukaan ja esit-

tää ne taulukon muodossa. (Ks. kuvio 22.) Mikäli käyttäjä painaa Poista-kentässä olevaa nappia, verkkosovellus lähettää Axios-pyynnön, joka poistaa kyseisen merkinnän tietokannasta sen id:n perusteella.

Kuvio 22. Viimeiset kymmenen merkkausta ja niiden poistaminen.

5.4.5 Tuntien poistaminen valitulta aikaväliltä

Tuntien poistaminen valitulta aikaväliltä-komponentissa käyttäjä pystyy tarkastelemaan taulukkona hänen merkkauksiaan valitulta aikaväliltä ja poistamaan niitä halutessaan. Aikavälin valitseminen tapahtuu react day picker -komponenttia hyödyntämällä. Käyttäjän valittua aikavälin, verkkosovellus kutsuu Axioksen avulla Expressistä funktiota, joka noutaa tietokannasta kaikki valitun aikavälin merkkaukset ja palauttaa ne tuntien poistaminen valitulta aikaväliltä-komponentille json-tiedostomuodossa. Komponentti tulostaa json-tiedoston sisällön taulukkona. Taulukko on samanlainen kuin muissakin taulukkoa hyödyntävissä komponenteissa, mutta se ilmestyy välittömästi react day picker -komponentin alapuolelle käyttäjän valittua päivämäärävälin.

(Ks. kuvio 23.) Päivämääräväliä on mahdollista vaihtaa päivittämättä sivustoa Reactin avulla. Reactin yksi parhaista puolista on, että sivustolla tapahtuvien muutoksien esittäminen käyttäjälle on mahdollista sivua päivittämättä.

Kuvio 23. Tuntien poistaminen valitulta aikaväliltä -komponentti.

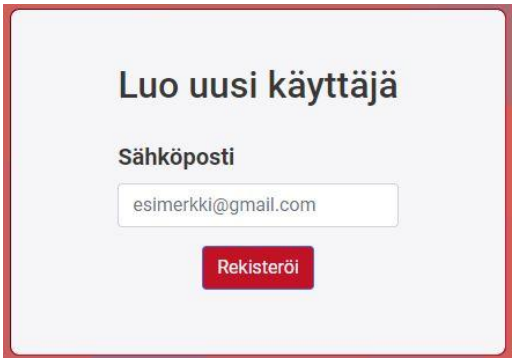
5.5 Admin

5.5.1 Yleistä

Admin-sivustolla verkkosovelluksen tai organisaation admin pystyy tarkastelemaan työntekijöiden tehtyjä tunteja sekä luomaan uuden käyttäjän verkkosovellukseen. Admin-sivustolla on kolme komponenttia. Admin-sivustolle pääsevät vain organisaation esimiehet sekä verkkosovelluksen admin. Admin-status määritellään käyttäjäkohtaisesti tietokantaan ja on oletuksena poissa päältä.

5.5.2 Uuden käyttäjän rekisteröiminen

Uuden käyttäjän rekisteröiminen -komponentti on yksinkertainen lomake. Lomake pyytää vain uuden käyttäjän sähköpostin. (Ks. kuvio 24.) Uuden käyttäjän rekisteröinnissä sovellus luo tietokantaan uuden käyttäjän syötetyllä sähköpostilla ja automaattisesti generoidulla salasanalla. Sähköposti ja salasana lähetetään syötettyyn sähköpostiin viestinä sovellukselta. Sähköpostin lähettäminen tapahtuu Expressissä nodemailer-nimisen npm:n avustuksella. Kuviossa 24 näkyy miltä komponentti näyttää.



Kuvio 24. Kuvakaappaus uuden käyttäjän rekisteröiminen -komponentista.

5.5.3 Viimeisen kymmenen merkinnän tarkasteleminen

Viimeisen kymmenen merkinnän tarkasteleminen -komponentissa voidaan tarkastella halutun työntekijän kymmentä viimeisintä merkintää ja poistamaan merkintöjä tarvittaessa. Haluttu työntekijä valitaan pudotusvalikosta. Työntekijän valittua verkkosovellus noutaa Axioksen avulla Expressistä viimeiset kymmenen valitun käyttäjän tekemää merkintää. Express palauttaa ne json-muodossa ja viimeisen kymmenen merkinnän tarkasteleminen -komponentti esittää noudetut merkinnät taulukon muodossa. (Ks. kuvio 25.) Komponentissa on oletuksena näkyvillä ensimmäisenä aina sisäänkirjautuneen käyttäjän omat merkinnät. Tarkasteltavaa käyttäjää pystyy vaihtamaan sulavasti ja nopeasti, eikä sivuston täydy ladata etukäteen kaikkia tietokannasta löytyviä merkintöjä komponenttia ladatessa. Reactin avulla pystytään tekemään ja esittämään rajapintapyyntöjä ja niiden vastauksia reaaliaikaisesti.

Kuvio 25. Viimeisen kymmenen merkinnän tarkasteleminen -komponentti.

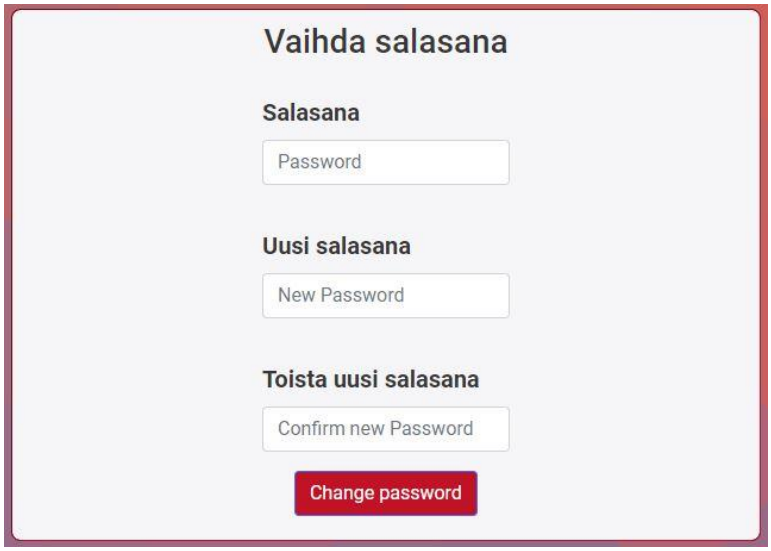
5.5.4 Tuntien tarkasteleminen ja lataaminen tietyltä aikaväliltä

Tuntien tarkasteleminen ja lataaminen tietyltä aikaväliltä -komponentissa esimies tai admin pystyy tarkistamaan valitun työntekijän tehdyt työtunnit tietyltä aikaväliltä. (Ks. kuvio 26.) Aikaväli valitaan react day picker -komponenttia hyödyntäen. Verkko-sovellus laskee tehdyt työtunnit valittuna aikana. Työnantaja pystyy luomaan CSV-tiedoston valitun aikavälin tunneista ja lataamaan työtunnit tiedostona. K

Kuvio 26. Tuntien tarkasteleminen ja lataaminen tietyltä aikaväliltä.

5.6 Käyttäjän profiili

Käyttäjä pystyy vaihtamaan salasanansa profiilissaan. Salasanan vaihtaminen toimii lomakkeen kautta. (Ks. kuvio 27.) Lomakkeessa pyydetään vanha salasana, uusi salasana ja uuden salasanan varmistus. Salasanan vaihtamisesta tulee ilmoitus joko onnistumisesta, tai virheilmoitus. Salasanan vaihtaminen tapahtuu lähettämällä Reactista Axiosin avulla rajapintapyyntö Expressiin.



Kuvio 27. Kuvakaappaus salasanan vaihtaminen -komponentista.

Rajapintapyynnön mukana lähetetään sisäänkirjautuneen käyttäjän sähköposti sekä täytettyjen kenttien arvot. Lähetetyt arvot tallennetaan muuttujiin ja ensimmäisenä niistä hyödynnetään sisäänkirjautuneen käyttäjän sähköpostia, jonka perusteella haetaan tietokannasta sisäänkirjautunut käyttäjä ja tallennetaan se muuttujaan user. Käyttäjän tallennettua muuttujaan vertaillaan lomakkeen asetettua salasanaa käyttäjän tietokannasta löytyvään salasanaan. Mikäli salasanan vertailu epäonnistuu esittää verkkosovellus käyttäjälle virheilmoituksen väärästä salasanasta ja keskeyttää salasanan vaihtamisen. Mikäli salasanan vertailu onnistuu, verkkosovellus vertailee uutta salasanaa ja sen toistettua versiota. Vertailun onnistuttua verkkosovellus hyödyntää

bcryptiä muokatakseen salasanan salattuun muotoon ja tallentaa salasanan tietokantaan. Kuviossa 28 näkyy kuvakaappaus koodista, jolla salasanan vaihtaminen tapahtuu.

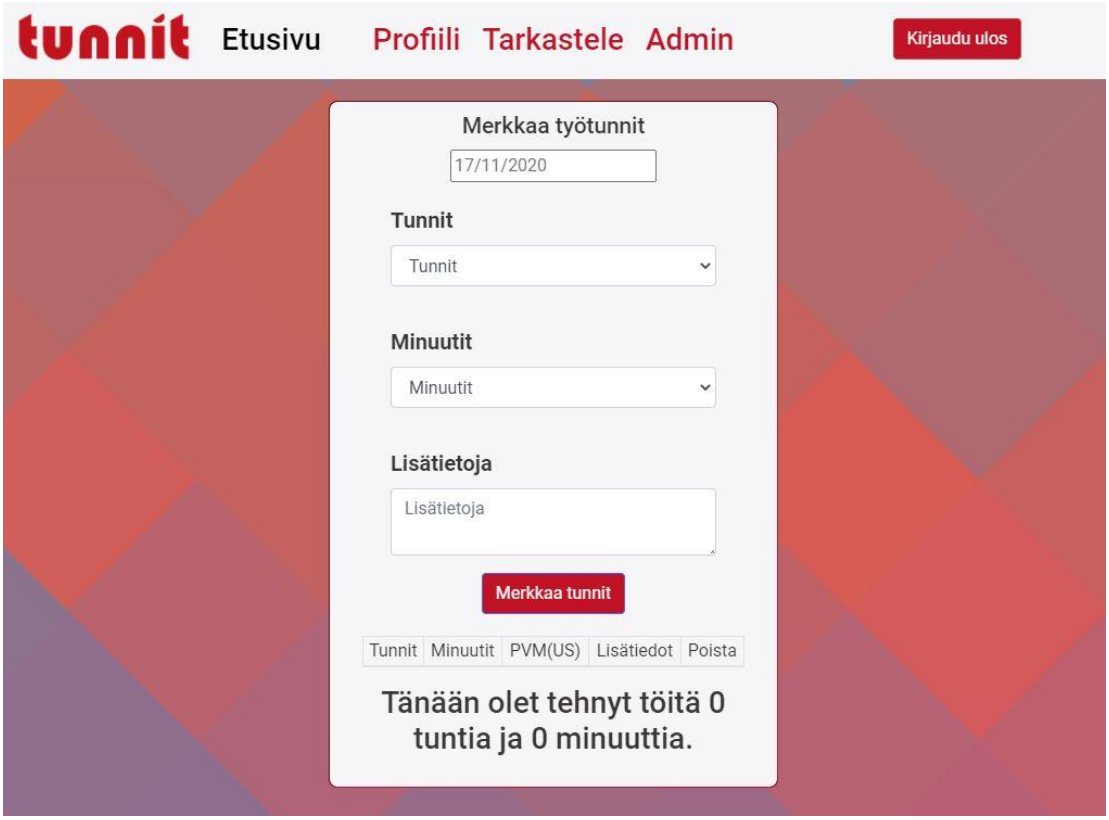
```
async (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({
      errors: errors.array(),
    });
  }

  const { email, password, newPassword, ConfirmnewPassword } = req.body;
  try {
    let user = await User.findOne({
      email,
    });
    if(!user) {
      console.log("ei löydy käyttäjää");
      return res.status(400).json({
        message:"Käyttäjää ei löydy",
      });
    }

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch){
      console.log("Väärä salasana");
      return res.status(400).json({
        message: "Väärä salasana!",
      });
    }
    if (user) {
      if(newPassword === ConfirmnewPassword){
      const isMatch = true;
      if (!isMatch){
        return res.status(400).json({
          message: "Salasanan varmistus väärin!",
        });
      }
      else if(isMatch){
        //User.changeUserPassword(user.email, user.newPassword)
        const salt = await bcrypt.genSalt(10);
        user.password = await bcrypt.hash(newPassword, salt);
        await user.save();
        //console.log("salasanat mätsää");
        res.status(200).json({
          successMessage: "Salasanan vaihtaminen onnistui"
        });
```

Kuvio 28. Kuvakaappaus salasanan vaihtamisesta Expressissä.

6 Tulokset ja pohdinta

Opinnäytetyön lopputuloksena saatiin suunniteltua ja toteutettua helppokäyttöinen ja toimiva tuntikirjanpitojärjestelmä. Valmis tuntikirjanpitojärjestelmä on responsiivinen ja nopeasti toimiva. Sen avulla pieni- tai keskikokoinen organisaatio pystyy kirjaamaan ja tarkastelemaan tehtyjä tunteja. Tuntikirjanpitojärjestelmälle ei ole tehty käyttäjätestausta, joten on mahdollista, että siitä löytyisi käytettävyyden kannalta

paljonkin parannettavaa. Kuviossa 29 näkyy tuntikirjanpitojärjestelmän lopullinen ulkonäkö.



Kuvio 29. Kuvakaappaus sovelluksen etusivusta.

Opinnäytetyö valmistettiin sen tekijälle etukäteen hyvin vierailla teknologioilla, joten oppimista tapahtui runsaasti. Teknologiat valittiin opinnäytetyön aloittamishetken suosituimpien teknologioiden mukaan, joten opitut teknologiat voivat varmasti auttaa opinnäytetyön tekijää työnhaussa. Teknisessä toteutuksessa oli ajoittain haasteita. Erityisesti alkuun kesti pitkään opetella miten hakea ja näyttää tietoa tietokannasta Reactissa. Haasteista kaikki selätettiin pysyvästi. Haasteita on hyvä olla, jotta tekijä oppisi mahdollisimman paljon. Valittujen teknologioiden yksi parhaista puolista on kaikki internetistä löytyvä apu niiden käyttöä varten. Työnteko ja opinnäytetyön valmistuminen etenivät tekijälle uusista teknologioista huolimatta suunnitellusti ja tehokkaasti koko prosessin ajan.

Jatkokehitystä ajatellen ensimmäinen askel olisikin varmasti verkkosovelluksen saaminen testattavaksi yhteen tai muutamaan organisaatioon. Mikäli organisaatiot tykkäisivät verkkosovelluksesta, he saattaisivat olla kiinnostuneita ostamaan sen käyttöönsä. Joka tapauksessa testauksesta saisi varmasti hyvin paljon hyötyä ja mahdollisesti paljonkin parannettavaa verkkosovelluksen toimintaan.

Lähteet

10 Trending 2020 Website Color Schemes. 2019. Julkaisu Quicksprout -verkkosivustolla. Viitattu 22.10.2020. https://www.quicksprout.com/trending-website-color-schemes/ .

Ali, J. 2018. How Developers got Password Security so Wrong. Blogijulkaisu Cloudflare -verkkosivustolla. Viitattu 22.10.2020. https://blog.cloudflare.com/how-developers-got-password-security-so-wrong/ .

Analytics. N.d. Google Fonts. Analytiikka Google Fonts -verkkosivustolla. Viitattu 22.10.2020. https://fonts.google.com/analytics .

Arias, D. 2018. Hashing in Action: Understanding bcrypt. Blogikirjoitus Auth0 -verkkosivustolla. Viitattu 22.10.2020. https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ .

Axios. N.d. github. Repositorio github -verkkosivustolla. Viitattu 22.10.2020. https://github.com/axios/axios .

Boyer, C. 2018. Why is it Important to Have a User Friendly Website. Blogijulkaisu Charley Grey -verkkosivustolla. Viitattu 3.11.2020. https://www.charleygrey.com/web-design/why-is-it-important-to-have-a-user-friendly-website/.

Brand Building 101: How to Build a Brand. 2020. The Daily Egg. Blogijulkaisu The Daily Egg -verkkosivustolla. Viitattu 11.11.2020. https://www.crazyegg.com/blog/how-to-build-a-brand/.

Costa, R. 2019. The complete guide to website wireframe design. Blogijulkaisu Justinmind -verkkosivustolla. Viitattu 22.10.2020. https://www.justinmind.com/blog/website-wireframe-design-guide/ .

Duffy, C. 2015. Valuable Player on Your Website Team! Blogijulkaisu Impact -verkkosivustolla. Viitattu 22.10.2020. https://www.impactbnd.com/blog/why-sitemaps-are-undervalued .

Duunitori. 2020. Duunitorin verkkosivut. Viitattu 21.10.2020. https://duunitori.fi/tyopaikat/ .

Gordiyenko, S. 2015. Website Development Process: Full Guide in 7 steps. Blogijulkaisu XB Software -verkkosivustolla. Viitattu 22.10.2020. https://xbsoftware.com/blog/website-development-process-full-guide/ .

Hinnasto. N.d. Cloudcity. Cloudcity. Viitattu 22.10.2020. https://cloudcity.fi/webhotellit/hinnasto/ .

Hinnoittelu. N.d. Tuntikirja. Tuntikirja. Viitattu 22.10.2020. https://tuntikirja.fi/sivut/hinnoittelu/ .

Hoque, S. 2018. Full-Stack React Projects: Modern Web Development Using React 16, Node, Express, and MongoDB. Englanti: Packt Publishing / Limited. https://ebookcentral-proquest-com.ezproxy.jamk.fi:2443/lib/jypoly-ebooks/detail.action?docID=5405708.

HTML: HyperText Markup Language. N.d. MDN web docs. Julkaisu MDN web docs -verkkosivustolla. Päivitetty 10.09.2019. Viitattu 22.10.2020. https://developer.mozilla.org/fi/docs/Web/HTML .

Most used libraries, frameworks, and tools among developers, worldwide, as of early 2020. 2020. Statistic. Statista. Viitattu 21.10.2020. https://www.statista.com/statistics/793840/worldwide-developer-survey-most-used-frameworks/ .

Node.js -Introduction. N.d. Tutorialspoint. Julkaisu sivustolla tutorialspoint. Viitattu 21.10.2020. https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm/ .

Peate, S. 2017. How to create a brand logo: Your first step in visual branding. Blogijulkaisu Fabrik -verkkosivustolla. Viitattu 11.11.2020. https://fabrikbrands.com/how-to-create-a-brand-logo/.

Potter, A. N.d. angular vs react vs vue. Statistiikka npm trends-verkkosivustolla. Viitattu 21.10.2020. https://www.npmtrends.com/angular-vs-react-vs-vue.

Pricing. MongoDB. N.d. Mongodb. Viitattu 22.10.2020. https://www.mongodb.com/pricing **.**

Pricing. Moqups. N.d. Moqups. Viitattu 22.10.2020. https://moqups.com/pricing/ .

Salomaa, J. 2020. Evaluating JavaScript Frameworks. Opinnäytetyö. Ruotsi: Linnaeus University.

Senecki, A. 2020. Web development stacks – what stacks (should) we use in 2020? Blogijulkaisu The Software House -verkkosivustolla. Viitattu 21.10.2020. https://tsh.io/blog/web-development-stacks/ .

Suomen virallinen tilasto. 2019. Suomalaisten internetin käyttö 2019. Julkaisu Tilastokeskuksen verkkosivustolla. Viitattu 3.11.2020. https://www.stat.fi/til/sutivi/2019/sutivi_2019_2019-11-07_kat_001_fi.html

Team. N.d. React. Julkaisu React -verkkosivustolla. Viitattu 22.10.2020. https://reactjs.org/community/team.html .

Työaika. N.d. Tietotoimisto.fi. Tietotoimisto. Viitattu 3.11.2020. https://www.tietotoimisto.fi/ty%C3%B6ajanseuranta/ .

Työajanseuranta. N.d. Deltabit. Deltabit. Viitattu 22.10.2020. https://www.deltabit.fi/tyoajanseuranta/ .

Usein kysyttyä EU:n tietosuoja-asetuksesta. Tietosuojavaltuutetun toimisto. N.d. Tietosuoja.fi. Viitattu 22.10.2020. https://tietosuoja.fi/gdpr .

Visma Severa Hinnoittelu. N.d. Visma. Viitattu 22.10.2020. https://psa.visma.fi/visma-severa-hinnoittelu/ .

Wong, C. 2020. How to Choose a Good Color Scheme For Your Website. Julkaisu WebsiteBuilderExpert -verkkosivustolla. Viitattu 22.10.2020. https://www.websitebuilderexpert.com/designing-websites/how-to-choose-color-for-your-website/ .

Yleinen tietosuoja-asetus. N.d. Euroopan Unioni. Informatiivinen julkaisu Europa.eu -verkkosivustolla. Päivitetty 14.09.2020. Viitattu 22.10.2020. https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_fi.htm .

Yrityksen nimipalvelu. N.d. Patentti- ja rekisterihallitus. Informatiivinen julkaisu PRH.fi -verkkosivustolla. Viitattu 11.11.2020. https://nimipalvelu.prh.fi/nipa/fi