Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

EETU NYMAN

# Päivystystuntien kirjaus- ja laskuohjelman suunnittelu ja toteutus

SÄHKÖ- JA AUTOMAATIOTEKNIIKAN
TUTKINTO-OHJELMA
2022

Tekijä(t)
Nyman, Eetu

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
Marraskuu, 2022

Sivumäärä
35

Julkaisun kieli
Suomi

Julkaisun nimi
Päivystystuntien kirjaus- ja laskuohjelman suunnittelu ja toteutus

Tutkinto-ohjelma
Sähkö- ja automaatiotekniikka

Tiivistelmä
Tässä opinnäytetyössä suunniteltiin ja toteutettiin 02 Palvelut Oy:n tarpeisiin työkalu, jonka avulla yrityksen IT-päivystyksestä vastaavan henkilöstön on mahdollista kirjata tehdyt työtehtävät tehokkaasti tietokantaan vähentäen aikaisemmin vaadittuja kirjauksen vaiheita.

Työkalun ohjelmoinnissa avainasemassa olivat ReactJS JavaScript kirjasto, jonka avulla työkalun käyttöliittymä toteutettiin, Serverless-arkkitehtuuri, jonka avulla työkalun palvelimen toiminnasta muodostui kustannustehokas ratkaisu sekä MariaDB tietokanta jonka sisällöksi vuorot tallennetaan.

Työtehtävien kirjauksen lisäksi työkaluun ohjelmoitiin ominaisuus, jonka avulla halutulta ajanjaksolta on mahdollista muodostaa taulukko, jonka sisällöksi lasketaan eri työntekijöille kirjatut varallaolovuorot sekä päivystystehtävät oikeisiin palkkaluokkiin niiden ajankohtien perusteella.

Avainsanat
Ohjelmointi, Käyttöliittymä, Tietokanta, ReactJS, NodeJs, Serverless, MariaDB, AWS

| Author(s)<br>Nyman Eetu | Type of Publication<br>Bachelor's thesis | Date<br>November 2022 |
|---|---|---|
| | Number of pages<br>35 | Language of publication:<br>Finnish |

Title of publication
Planning and implementation of the recording and calculation program for on-call hours

Degree programme
Electrical and automation engineering

Abstract

This thesis describes the planning and implementation of a computer program developed to meet the needs of 02 Palvelut Ltd. The purpose of the tool was to streamline the process that personnel of IT-department previously had to do in order to document work done during on call-work tasks.

Essential frameworks for development of this program were ReactJS which was used to create the user interface for the program. Backend was created using Serverless-framework which helped the backend be more cost efficient compared to more traditional backend solutions. MariaDB was chosen as the database to store all of the work shifts.

In addition to keeping track of added work tasks the program has a feature to calculate time spent for each employee in a way that each database entry for specific time period is calculated in the correct salary grade based on the time of the worktask.

Keywords
Programming, User Interface, Database, ReactJS, NodeJs, Serverless, MariaDB, AWS

# SISÄLLYS

# LYHENNELUETTELO

AWS = Amazon Web Services, pilvipalveluita tarjoava alusta.

JS = JavaScript, ohjelmointikieli

SQL = Structured Query Language, kyselykieli relaatiotietokannoille.

UI = User Interface, Käyttöliittymä

URL = Uniform Resource Locator, verkkosivun osoite.

VPC= Virtual Private Cloud, AWS:n palvelu, joka muodostaa loogisesti eristetyn virtuaalisen verkon.

# 1 JOHDANTO

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa opinnäytetyön toimeksiantajan: 02 Palvelut Oy:n tarpeisiin soveltuva työkalu, jonka avulla yrityksenä tietoteknisistä järjestelmistä vastaavan tiimin on mahdollisimman vaivatonta kirjata ja laskea, päivystystehtäviin sekä varallaoloon kulunut aika. Työkalu on suunniteltu ja toteutettu kesän 2022 aikana.

02 Palvelut Oy tuottaa asiakkailleen nimi- ja numeropalveluita. Kattavaan palveluvalikoimaan kuuluvat brändit kuten: 020202, 16400, 02 Rekkari sekä taksien tilauksiin ja hintavertailuihin keskittyvä sisaryritys 02 Taksi, jonka puhelinpalvelu on tuotettu kuitenkin 02 Palvelut Oy:n puolella. (020202 Palvelut, n.d., kohta Tutustu Nollakakkoseen!)

Yrityksen tarjoamat puhelinpalvelut ovat avoinna vuoden jokaisena päivänä ja ympäri vuorokauden, näin ollen on myös välttämätöntä, että IT-tiimissä on vähintään yksi henkilö tavoitettavissa mahdollisten ongelma- ja virhetilanteiden pikaisen korjaamisen mahdollistamiseksi. Opinnäytetyössä toteutettavan työkalun tavoitteena on nopeuttaa paitsi päivystystehtävien kirjaamista niin myös viikoittain kiertävien päivystysvuorojen suunnittelua, tämä puolestaan edesauttaa muun työnteon tehostamista, sillä työntekijöiden on mahdollista käyttää aikaisemmin kirjauksiin kulutettu aika muihin työtehtäviin. Kirjausten ja suunnittelun lisäksi työkaluun liitetään mahdollisuus muodostaa kuukausikohtainen koonti tehdyistä tunneista, jossa tunnit lasketaan niiden ajankohtien mukaan eritellysti omiin palkkaluokkiinsa ICT-alan toimihenkilöiden työehtosopimuksen mukaisesti.

Olen ollut 02 Palvelut Oy:llä asiakasneuvojana heinäkuusta 2018 lähtien ja joulu-marraskuussa 2021 tiedustelin mahdollisuutta suorittaa kesällä opintoihini liittyvää työharjoittelua osana IT-tiimiä. Alun perin työharjoitteluun liittyvänä projektina alkanut työkalun toteuttaminen paisui kesän aikana tarpeidenkartoittamisen sekä

vaatimusmäärittelyjen myötä siinä määrin kattavaksi kokonaisuudeksi, että se soveltui myös opinnäytetyönaiheeksi.

Tässä opinnäytetyössä kerron ensin, miten ennen työkalun kehittämistä on siihen liittyvät työtehtävät tehty sekä mistä tarve kyseiselle työkalulle on muodostunut. Tämän jälkeen kerron tarkemmin työkalun suunnitteluprosessista sekä työkalun toteutukseen käytetyistä työkaluista ja järjestelmistä. Suunnittelun ja järjestelmien käsittelemisen myötä tutustutaan itse työkalun toteuttamiseen ja lopuksi vielä projektin haasteisiin sekä työkalun mahdolliseen jatkokehittämiseen.

Opinnäytetyössä esitettävissä kuvissa sekä taulukoissa on nimet sekä muut tunnistetiedot muutettu tuotantokäytössä olevasta versiosta poikkeavaksi tietoturvan takaamiseksi, ellei kyseisen kuvan tai taulukon yhteydessä toisin mainita.

# 2 NYKYTILANNE JA TARPEET

Jokaisella vuoden ja vuorokauden hetkellä on 02 Palvelut Oy:llä vähintään yksi henkilö vastuussa IT-päivystäjän tehtävistä. Päivystysvuoroista käytetään myös termiä varallaolo, varallaolovuoro vaihtuu jokaisen viikon maanantaina klo 8 edelliseltä päivystäjältä seuraavalle. Varallaoloksi lasketaan aika normaalin säännöllisen työajan ulkopuolella, tässä tapauksessa arkisin kello kahdeksan ja iltapäivä neljän välillä on säännöllistä työaikaa.

Yllä olevan perusteella on muodostettu Microsoft Exceliä hyödyntämällä varallaolovuorot jokaiselle kuukauden päivälle kuun alussa. Viikonloppuisin varallaolovuoron kesto on 24 tuntia, kun taas arkipäivisin varallaolovuorot on jaoteltu taulukossa kahteen erilliseen riviin keskiyöstä aamun kello kahdeksaan sekä kello 16 alkaen jälleen keskiyöhön. Mikäli varallaolovuorossa olevalle työntekijälle on tullut työtehtävä säännöllisen työajan ulkopuolella, on kyseiseen Exceliin lisätty uusi rivi oikean viikon kohdalle ja riviin on lisätty tiedoiksi: päivämäärä, alku- ja päättymiskellonaika sekä selite. Päivystystehtävän kirjaamisen lisäksi on kyseinen aika otettava pois merkatusta varallaolovuorosta, jolloin jo olemassa ollut merkintä on jaettu kahdeksi uudeksi riviksi: aika ennen ja jälkeen päivystystehtävän. Jokaisen kuukauden varallaololistassa on näin ollen vähintään 48 vuoroa mikäli yhtään päivystystehtävää ei kuukauden aikana olisi, jokainen tehtävä lisää määrää suuremmaksi. Kuukauden lopussa listasta on laskettu jokaiselle päivystäjälle tunnit oikeisiin palkkaluokkiin niiden ajankohtien perusteella ICT-alan toimihenkilöiden työehtosopimuksen mukaisesti.

Opinnäytetyön tutkimuskysymyksiksi muodostuu: ”Miten varallaolovuorojen sekä päivystystehtävien merkitsemistä on mahdollista sujuvoittaa?” sekä ”Kuinka vuorojen laskeminen oikeisiin palkkaluokkiin voitaisiin automatisoida?” Näihin tutkimuskysymyksiin vastaamalla saadaan toteutettua toimeksiantajan tarpeisiin vastaava työkalu.

Uuden työkalun osalta ei 02 Palvelut Oy:llä ollut vaatimusta siinä käytettävästä ohjelmointikielestä, toiveena kuitenkin oli, että se saataisiin toimintaan sisäisessä verkossa ja selaimella. Microsoft Excelillä aikaisemmin toteutetulla tavalla oli haasteeksi muodostunut myös se, että päivystysvuoroja tekevien työntekijöiden piti aina olla varmoja,

että kirjaukset tulevat uusimpaan mahdolliseen versioon, jotta aikaisempia kirjauksia ei katoa, mikäli uusia merkintöjä tehdäänkin vahingossa väärään tiedostoon. Kaikki kirjatut vuorot olivat tämän lisäksi vain tallessa aina kyseisen kuukauden Excel-tiedostossa eikä esimerkiksi omassa tietokannassaan.

# 3 KÄYTETTÄVÄT JÄRJESTELMÄT

Työkalun toteutuksessa käytettävät tärkeimmät järjestelmät on jaoteltu kolmeen osaan sen perusteella mikä niiden osuus työkalun toiminnassa on. Ensimmäinen osa on käyttöliittymä, jonka avulla käyttäjä pystyy lisäämään, lukemaan, muokkaamaan ja poistamaan merkintöjä vuorolistoista. Toinen osio kertoo tarkemmin työkalun palvelinpuolen toiminnasta ja viimeisenä osiona tutustutaan tarkemmin työkalun käyttämään tietokantaan sekä tapaan, jolla tietokannasta välitetään tietoja palvelimen kautta käyttöliittymään.

Jotta voi ymmärtää tarkemmin työkalun front-endin eli selainpuolen sekä back-endin eli palvelinpuolen toimintaa on pakko ensin käsitellä hieman nykyaikaisen verkko kehityksen kulmakiviä, jotka ovat: HyperText Markup Language (jatkossa HTML), Cascading Style Sheets (jatkossa CSS) sekä Javascript. Näistä ensimmäinen eli HTML on niin sanottu merkintäkieli, se määrittää verkkosivujen rakenteen lisäämällä sivuille osioita kuten otsikot, tekstikappaleet, taulukot ja kuvat. CSS puolestaan vastaa sivuston visuaalisesta ilmeestä: sen avulla voidaan vaikuttaa HTML elementtien sijoitteluun, kokoihin ja väreihin (W3C, n.d, kohta HTML & CSS.) CSS on myös avainasemassa, kun toteutetaan verkkosivuja tai verkkosivuilla toimivia sovelluksia, joita käytetään tietokoneiden lisäksi myös mobiililaitteilla. Seuraavien alaotsikoiden kannalta tärkein kulmakivistä on kuitenkin Javascript joka on ohjelmointikieli. Javascriptin avulla verkkosivuilla voidaan toteuttaa erilaisia toiminnallisuuksia kuten sisällön muuttuminen kellonajan perusteella tai käyttäjäsyötteen perusteella laskea haastaviakin matemaattisia ongelmia (What Is Javascript? 2022, kohta A high-level definition). Toukokuussa 2021 ohjelmistokehittäjille järjestetyn kyselyn mukaan Javascript oli jo yhdeksättä kertaa peräkkäin yleisimmin käytetty ohjelmointikieli (Stack Overflow Developer Survey, 2021, kohta Most popular technologies).

## 3.1 Selain/Käyttöliittymä

Työkalun käyttäjälle näkyvä osuus toteutetaan käyttämällä Facebookin (nyk. Meta) kehittämää avoimen lähdekoodin JavaScript kirjastoa nimeltä React, se on suunniteltu käytettäväksi nimenomaan interaktiivisten käyttöliittymien toteutuksessa (React, n.d.

kohta Declarative). React oli yleisimmin käytetty verkkokehys samassa kyselyssä, jossa Javascript oli yleisin ohjelmointikieli, kyselyyn vastanneista 67593 henkilöstä 40,14 % kertoi käyttäneensä Reactia kuluneen vuoden aikana runsaasti ja suunnitteli käyttävänsä sitä myös tulevan vuoden aikana (Stack Overflow Developer Survey, 2021, kohta Web frameworks.)

React kirjastolla toteutettujen sovellusten toiminta perustuu komponentteihin, jotka ovat kapseloituja moduuleja (bin Uzayr ym., 2019, kohta What makes React special?). Komponentit voivat esimerkiksi tehdä laskelmia tai reagoida käyttäjäsyötteeseen ja tämän perusteella renderöidä HTML elementtejä näkyviin tai pois näkyvistä. Yksi tärkeistä ominaisuuksista tämän työkalun toteutuksen kannalta Reactissa on sen tarjoama mahdollisuus käsitellä kattavia määriä dataa tehokkaasti, sillä se tarkkailee jokaisen sovelluksessa olevan komponentin tilaa. Tilamuutokset voivat mahdollistaa yksittäisten komponenttien tai tarvittaessa koko sovelluksen uudelleen renderöinnin. Tilamuutos voi olla esimerkiksi sivulla olevan napin painallus tai tekstin syöttäminen sille tarkoitettuun kenttään. Näiden toiminnallisuuksien keskiössä on kaksi Reactin tärkeää ominaisuutta: JSX ja Virtual DOM.

Kun verkkoselain avaa verkkosivun, jotta käyttäjä voi nähdä sen sisällön, muodostaa selain HTML:n perusteella Document Object Modelin (myöhemmin DOM). DOM määrittää verkkosivun rakennetta ja sen osien vaikutusta toisiinsa. JSX on lisäosa Reactiin joka helpottaa verkkokehittäjien työtä kun kehittäjällä on tarve muokata sivuston DOM:ia käyttämällä yksinkertaista HTML-tyyliä mukailevaa koodia. Sovelluskehittäjän käyttäessä Reactia sekä JSX:ää muodostaa React erillisen Virtual DOM:in joka on virtuaalinen kopio sivun varsinaisesta DOM:ista. Mikäli sivustolla tapahtuu muutoksia esimerkiksi jonkin komponentin tilassa vertaa React kahta DOM:ia keskenään ja sen perusteella tekee muutokset sivuston varsinaiseen DOM:iin, näin vältetään turhat päivitykset ja muutokset sivun rakenteessa, joka puolestaan nopeuttaa latausaikoja ja vähentää vaadittua sivun laskentatehoa (Morris, n.d., kohta Why Do JavaScript Developers Use React JS?)

### 3.2 Palvelin

Palvelinpuolen toiminnan pohjana työkalulle käytetään Node-nimistä ajoympäristöä, joka on suunniteltu JavaScript koodin suorittamiseen palvelimella. Node on asynkroninen ja tapahtumalähtöinen ajoympäristö, tämä mahdollistaa sen, että palvelimelle välitettyjen pyyntöjen käsittelyä ei tarvitse odottaa yhtä pitkään kuin järjestelmillä joista asynkronisuus puuttuu. Node package manager eli npm on Noden oletus pakетinhallinta järjestelmä, jonka avulla siihen on mahdollista liittää kehittäjien luomia paketteja sen toiminnan laajentamiseksi entisestään (Hoque, 2018, s.7.)

Npm:n avulla saadaan liitettyä osaksi Nodea verkkosovellusten ja ohjelmointirajapintojen toteutukseen tarkoitettu paketti nimeltä Express, joka yhdessä Noden kanssa mahdollistaa palvelimelle saapuvien kutsujen reitittämisen sekä väliohjelmistojen käytön. Väliohjelmistoja voidaan käyttää esimerkiksi tarkastamaan palvelimen vastaanottaman kutsun tyylin oikeanlaiseksi tai kirjoittamaan erillistä lokia kaikista vastaanotetuista kutsuista (Mardan, 2014, kohta How Express.js Works.)

Työkalun backendin kannalta viimeinen tärkeä osio on sovellusarkkitehtuuri nimeltään Serverless joka suomeksi käännettynä tarkoittaa palvelimetonta, tämä kuulostaa varmasti erikoiselta, kun aikaisemmin on puhuttu koko ajan palvelinpuolen toiminnasta. Nimestään huolimatta Serverless ei kuitenkaan toimi täysin ilman palvelinta. Serverless sovellusarkkitehtuurissa palvelin on käytössä vain silloin kuin sitä tarvitaan. Tällainen pilvipalvelu-muotoinen toiminta on käyttäjän kannalta taloudellisesti kustannustehokkaampi vaihtoehto kuin palvelin, joka on jatkuvasti päällä ja käytössä (Cloudflare, n.d., kohta What is serverless computing?) Serverlessin avulla voidaan kapsuloida Express:in sekä Noden avulla luodut toiminnallisuudet funktioihin, joita pilvipalvelu kutsuu tarvittaessa.

### 3.3 Tietokanta

Jotta työkalulla olisi mahdollista tarkastella jo kirjattuja vuoroja ja tämän lisäksi myös lisätä ja poistaa uusia vuoroja tulee niiden olla lisättynä tietokantaan. Projektissa käytettäväksi tietokannaksi valikoitui yksi suosituimmista avoimen lähdekoodin relaatiotietokannoista: MariaDB (MariaDB, n.d., kohta New to MariaDB Server?)

Lähtökohtaisesti työkalu olisi ollut mahdollista toteuttaa myös muilla tietokantapalveluilla kuten MongoDB joka käyttää perinteisemmän relaatiomallin sijaan NoSQL-tietokantajärjestelmää. Tämän tyyppinen tietokantamalli olisi ollut mahdollinen sen vuoksi että lähtökohtaisesti jokainen tietokantaan tuleva tieto vuorosta voidaan sijoittaa samaan tauluun ja erillisiä relaatioita ei tietokannan sisällä taulujen välille näin ollen muodostu. Tästä huolimatta MariaDB valikoitui tietokannaksi sen aktiivisen käyttäjäyhteisön sekä yleisen suosion vuoksi. MariaDB:n suosiota kuvaa myös se, että Fortune-talouslehti julkaisee vuosittain listan 500. suurimmasta Yhdysvalloissa toimivista pörssiyhtiöistä niiden liikevaihdon mukaan, tämän ”Fortune 500”-listan yhtiöistä 75 % käyttää toiminnoissaan MariaDB:tä (MariaDB, 2021).

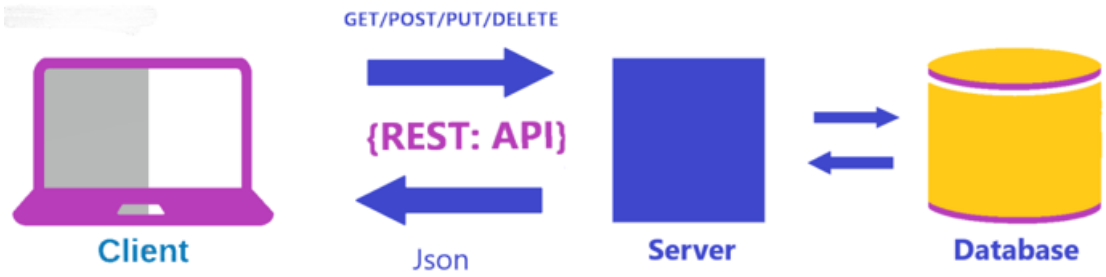
## 3.4 REST

Palvelimen ja käyttöliittymän väliseen kommunikointiin tarvitsee käyttää sopivaa rajapintaa. Tässä tapauksessa rajapintana toimii REST api (myös nimellä RESTful api) jonka avulla lähetetään pyyntöjä frontendin ja backendin välillä käyttäen HTTP sekä HTTPS-protokollia. REST lyhenne tulee sanoista REpresentational State Transfer. Kun käyttöliittymältä tai päätelaitteelta lähetetään REST kutsu palvelimelle, palvelin palauttaa kutsujalle kuvauksen (engl. representation) lähetetyn kutsun perusteella (Red Hat, 2020, kohta REST.)

REST-kutsu koostuu pääsääntöisesti neljästä osasta: HTTP verbistä, joka määrittää mitä kutsulla halutaan tehdä. Verbejä on neljää perustyyppiä: GET, jonka avulla pyydetään vastaanottaa resursseja joko yhden tai useita kerralla, POST, joka luo uuden resurssin, PUT-verbiä käytetään yksittäisen olemassa olevan resurssin päivittämiseen sekä DELETE, joka poistaa yksittäisen resurssin. Verbin lisäksi palvelimelle lähetettävään pyyntöön lisätään otsikko (engl. header) jolla voidaan välittää käyttäjältä palvelimelle tietoja kuten minkä muotoista dataa käyttöliittymä voi ottaa palvelimelta vastaan. URL polku (engl. path) johon kutsu lähetetään, on verbin ja otsikon ohella välttämätön osa kutsua, verbin tyyppi myös määrittää millainen polku kutsuun vaaditaan, esimerkiksi PUT ja DELETE vaativat aina polkuun uniikin tunnisteen eli id:n jonka perusteella kutsu tietää mihin resurssiin muutokset kohdistuvat, toisaalta GET

verbiä on mahdollista käyttää myös ilman id:tä jolloin kutsu hakee kaikkia mahdollisia resursseja kyseisestä polusta. Viimeinen kutsun osa on valinnainen viesti, jonka käyttäjä voi halutessaan lisätä osaksi kutsua (Codecademy, n.d., kohta REpresentational State Transfer)

Palvelimen vastaanottaessa käyttöliittymän kautta kutsun, joka on muodoltaan ja tiedoiltaan hyväksyttävä avaa palvelin tietokantayhteyden MariaDB tietokantaan ja toimii kutsussa olleen verbin mukaisella tavalla. Tietokanta palauttaa pyydetyn datan palvelimelle, joka välittää sen edelleen käyttäjälle. Tapauksissa, joissa oikeaa resurssia ei löydetä, palvelin lähettää käyttäjälle HTTP-tilakoodin sen perusteella millainen virhe on aiheutunut. Alla olevassa kuvassa palvelin palauttaa käyttäjälle JSON muotoisena kutsussa pyydetyn datan, mahdollisia vaihtoehtoja on kuitenkin myös monia muita kuten HTML, PHP tai Python (Red Hat, 2020 kohta REST).




Kuva 1: Käyttöliittymän, palvelimen ja tietokannan välinen kommunikointi (Bhosale, 2021)

# 4 SUUNNITTELU

Suunnittelun osalta isoimpia kysymysmerkkejä olivat käyttöliittymän ulkoasu sekä laskukaavojen muodostaminen, jotta kirjatut merkinnät tulevat lasketuksi oikein palkkakoontiin. Käyttöliittymän osalta tarpeita vastaisi parhaiten niin kutsuttu Single-page application (myöhemmin SPA), jossa selain lataa yksittäisen dokumentin ja muokkaa sen näyttämää sisältöä käyttäjän valintojen sekä datan perusteella (SPA, 2022, Single-page application). Huomioon on myös otettava, että osa työntekijöistä, jotka työkalua käyttäisivät työskentelevät pelkästään kannettavan tietokoneen näytöllä ja osa käyttäen erillisiä näyttöjä, joten työkalun toteutuksessa tulisi ottaa huomioon myös responsiivisuus erilaisilla näyttöjen kokoluokilla, mobiililaitteita ei työkalun toistaiseksi tarvinnut tukea. Responsiivisuus on yksinkertaisinta toteuttaa näytöt ja päätelaitteet huomioon ottaen käyttämällä CSS:n kirjoituksessa yksikköinä relatiivisia mittayksiköitä, kuten vw ja vh, jotka ovat lyhenteitä sanoista ”Viewport width” ja ”Viewport height” yksi 1vw tai 1vh vastaa 1 % selaimen ikkunan koosta, näin saadaan työkalun komponenttien koko mukautumaan myös näytön koon perusteella.

Selaimen ikkunan päätin jakaa kahteen osaan, joista ylempi olisi noin 30 % käytettävästä pinta-alasta. Tämä ylempi osio sisältäisi toiminnallisuuden: vuorojen lisäys sekä muokkaus, suodatintoiminnot vuorojen hakemiselle, vuorojen alustamisen kuukauden alussa sekä palkkakoontien lataamisen. Alempi osio täyttäisi loput näytöstä ja se sisältäisi allekkain järjestettynä suodatusten perusteella tietokantaan lisätyt vuorot, koska vuoroja ja merkintöjä on kuukaudessa runsaasti alempi osio sisältäisi myös vierityspalkin, jotta vuorojen datan sisältöä ei tarvitse esittää liian pienellä tekstillä.

Ensimmäinen yläosion komponenteista tulisi olla lomake, jonka avulla lisätään uusi vuoromerkintä tietokantaan, samaa lomaketta voitaisiin myös käyttää, mikäli jo tehtyä merkintää halutaan muokata, mikäli kyseessä olisi olemassa olevan vuoron muokkaus tämä tulee indikoida selvästi, jotta käyttäjä ymmärtää, että käsittelee jo tietokannassa olevaa dataa. Lomakkeen ensimmäinen mahdollinen syöte on työtyypin valinta: onko kyseessä varallaolo vai varsinainen päivystystehtävä. Toisena valintana valittaisiin työntekijä, toteutus tapahtuisi käyttämällä HTML:n select-elementtiä, jonka sisään luodaan vaihtoehdot päivystysvuoroja tekeville työntekijöille, näin voidaan välttää

esimerkiksi kirjoitusvirheistä johtuvat poikkeamat datassa. Työntekijän valinnan jälkeen valittaisiin työtehtävän aloitus- sekä lopetuspäivämäärät ja kellonajat, HTML sisältää valmiin elementin päivämäärän ja kellonajan valintaan yhdessä syötteessä, syötteen muoto riippuu kuitenkin tietokoneen ja käyttöjärjestelmän asetuksista, joten tähän tarpeeseen tulisi käyttää erillistä moduulia ja komponenttia, jolle voitaisiin määrittää alue eli lokalisaatio. Viimeinen lomakkeen osio on avoin tekstikenttä työtehtävän kuvauksen kirjoittamiseksi. Kaikkien lomakkeen osioiden ollessa täytettynä hyväksyttävillä tiedoilla se olisi mahdollista lähettää tietokantaan. Muokattaessa jo tehtyä kirjausta, kyseisen kirjauksen data täydentyisi automaattisesti lomakkeen kenttiin ja yksittäisen aseta-painikkeen sijaan ilmestyy painikkeet muokkauksen hyväksynnälle sekä peruutukselle.

Toinen komponentti toimii hakutyökaluna ja mahdollistaa vuorojen suodattamisen. Suodattaminen tapahtuu ensimmäisen komponentin kaltaisilla valintaelementeillä. Suodatuksen parametrejä ovat: työtyyppi, työntekijä, ajankohta sekä järjestys. Ajankohta on aina yksittäinen kuukausi tietyltä vuodelta, ajankohdalle määritellään lisäksi vakioarvo kuluvan päivän perusteella, näin työkalun avauksen yhteydessä näkee kuluvan kuukauden tiedot. Tiedot olisivat suodatuksessa myös mahdollista järjestää päivämäärän perusteella nousevaan tai laskevaan järjestykseen.

Usean varallaolovuoron alustamiseen tarkoitettu komponentti tarvitsee ainoastaan avoimen tekstikentän, kuukausivalinnan sekä painikkeen alustamisen toteuttamiseksi. Tekstikenttä ottaa vastaan käyttäjän syötteen, joka sisältää työntekijöiden nimet pilkuilla eroteltuna ja siinä järjestyksessä, jossa varallaolovuorot kyseiselle kuukaudelle tulisi sijoittaa. Koska varallaolovuoro kierto on maanantaista maanantaihin alustaminen voi tarvittaessa ylittää myös valitun kuukauden, mutta vain seuraavaan maanantaihin asti, tämän myötä alustamisessa tulee ensin lisäksi tarkastaa, onko varallaolovuoroja jo lisätty kyseiselle kuukaudelle ja jatkaa tarkastuksen tuloksen perusteella vuorojen lisäämistä annettujen nimien mukaan, annetut nimet myös tarkastetaan vastaavan sallittuja vaihtoehtoja.

Viimeinen työkalun yläosan komponenteista sisältää kuukausivalinnan sekä painikkeen datan tuomiseksi oikeassa muodossa Microsoft Exceliin, joka myöhemmin lähetettäisiin palkanlaskentaan. Tämän komponentin tulisi siis laskea työehtosopimuksen

mukaisesti tunnit työtyypin sekä ajankohdan mukaan oikeisiin palkkaluokkiin. Toiveena esitettiin lisäksi, että erillisellä Excelin sivulla olisi mahdollista nähdä kaikista kuukauden vuoroista vastaavanlainen listaus kuin aikaisemmin on tehty. Varallaolovuorot sekä varalta töihin eli päivystysvuorot jaetaan neljään eri palkkaluokkaan seuraavasti:

Luokka 1: Varallaolo-vuoro arkisin klo 16.00–08.00 sekä lauantai 00.00–24.00
Luokka 2: Varallaolo-vuoro sunnuntai ja pyhäpäivät 00.00–24.00 sekä joulu-, juhannus- ja uudenvuoden aatto klo 17.00–24.00
Luokka 3: Varalta töihin eli päivystys arkisin klo 16.00–21.00, lauantaisin 08.00–16.00, joulu- ja juhannusaatto 08.00–17.00
Luokka 4: Varalta töihin eli päivystys arkisin klo 21.00–08.00, lauantaina klo 16.00–24.00, sunnuntai ja pyhäpäivät klo 00.00–24.00 sekä joulu- ja juhannusaatto klo 17.00–24.00

Pseudokoodina eri luokkiin jaottelu tehtäisiin seuraavasti: Tarkastetaan kirjauksen työ tyyppi. Varallaolovuoron kohdalla tarkastetaan ensin, onko kyseessä pyhäpäivä tai sunnuntai, mikäli lause on tosi, lisätään rivitiedon työaika luokkaan 2. Jos kyseessä on joulu-, juhannus tai uudenvuodenaatto tarkastetaan ensin rivitiedon alku- sekä loppuaika ja kellonaikojen perusteella jaetaan työaika luokkiin 1 ja 2, kaikki loput varallaolovuorot menevät luokkaan 1. Päivystys kirjauksissa tarkastetaan ensimmäiseksi, onko kyseessä arkipäivä ja työnajankohta sijoittuu klo 16–21 välille lisätään päivystyksen kesto luokkaan 3. Mikäli kyseinen rivitieto sijoittuu lauantaille klo 8-16 välille taikka joulu- tai juhannusaattoon klo 8-17 välille niin lisätään luokkaan 3. Kaikki muut päivystyskirjaukset lisätään luokkaan 4.

Työkalun alaosa sisältää listan kaikista valitun ajanjakson vuoroista. Vuorot on listattu allekkain ja jokainen rivitieto näyttäisi kirjauksen tyypin, työntekijän nimen, aloituspäivämäärän sekä kellonajan, lopetuspäivämäärän ja kellonajan, näiden perusteella lasketun työtehtävän keston sekä selitteen työtehtävälle. Näiden tietojen perässä tulee jokaiselle vuorolle omat painikkeet, joiden avulla voi muokata tietoja tai poistaa tieto. Mikäli tieto poistetaan ja kyseessä on päivystystehtävä, lisätään kyseinen aika takaisin varallaolo ajaksi, vastaavasti jos aikaa muokataan pidemmäksi, se poistetaan oikeasta varallaolovuoron merkinnästä. Vuorolistauksen yläosaan tulee paikallaan pysyvä

staattinen otsikkopalkki, jonka avulla käyttäjän on mahdollista koko ajan nähdä mistä datasta kyseisessä sarakkeessa on kyse.

# 5 KÄYTÄNNÖN TOTEUTUS

Kuten kappaleessa 3 esitetään myös käytännön toteutus erikseen käyttöliittymälle sekä palvelimelle. Käyttöliittymä-alaotsikon lopussa ennen siirtymistä palvelintoteutuksen käsittelyyn esitellään miten käyttöliittymä lähettää kyselyt palvelimelle.

## 5.1 Käyttöliittymä

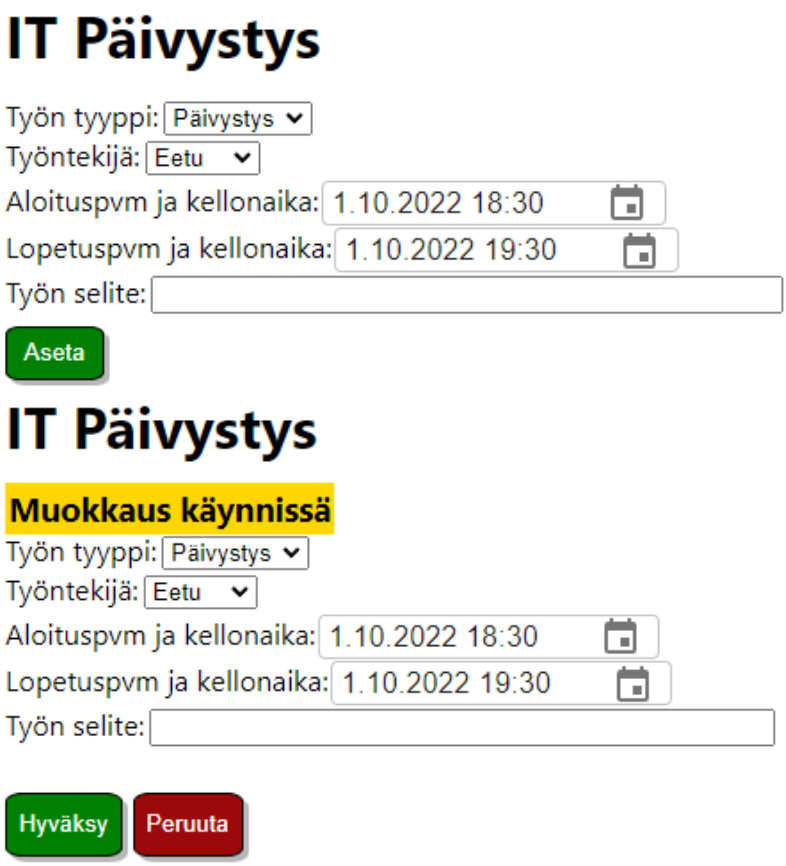
Työkalun pääkomponenttina toimii App.js jonka sisälle muut komponentit renderöidään. React mahdollistaa datan välittämisen komponenttien välillä myös muilla tavoilla kuin ylhäältä alas luomalla kontekstin, jonka sisään itse App komponentti lisätään. Konteksti käyttää lisäksi hyödykseen useState-nimistä React hookia työkalun eri toiminnallisuuksien lähtökohtaisen tilan hallinnoimiseen. Kontekstiin saadaan myös määritettyä työkalun lokalisaatio, näin sitä ei tarvitse erikseen määrittää komponenteissa, joissa käsitellään päivämääriä ja kellonaikoja. Lokalisaatioon käytetään npm kirjastoa nimeltä ”date-fns”.

App.js renderöi näkyviin yhteensä kahdeksan komponenttia, jotka muodostavat käyttöliittymän päänäkymän. Komponentit on jaettu aikaisemmin mainittuihin ylä- sekä alaosiin käyttämällä jaotteluun HTML:n div-elementtejä, elementeille on kaikille annettu luokkanimet ja osalle lisäksi myös uniikit tunnisteet, joita käytetään eri funktioissa. Kuvasta yksittäisen komponentin tunnistaa sen tekstin vihertävästä väristä. Komponenteista ”Header” sekä ”Footer” eivät sisällä toiminnallisuutta sillä niiden ensisijainen tehtävä on pääasiassa auttaa visuaalisen ilmeen muotoilussa, joten niiden sisältöön ei ole tarvetta tarkemmin perehtyä. Ensimmäisenä listassa nähtävä Login-komponentti on puolestaan lisätty, kun työkalu oli ollut jo hetken tuotantokäytössä ja todettiin että jonkinlainen kirjautumismahdollisuus olisi hyvä olla, kirjautumisesta tarkemmin lisää jatkokehitystä käsittelevässä kappaleessa.

```
36    return (
37      <div className="AppTop">
38        <div className="loginDiv" id="loginDiv">
39        <Login/>
40        </div>
41
42          <div className="App" id="appDiv">
43            <div className="upperHalf">
44              <Header Title="IT Päivystys" />
45              <div className="TopSection">
46                <Forms />
47                <Filter />
48                <Initialize />
49                <Exports />
50              </div>
51            </div>
52
53          <div className="lowerHalf">
54            <ShiftList />
55          </div>
56          <div className='appFooter'>
57            <Footer/>
58          </div>
59
60          </div>
61    </div>
62    );
```

Kuva 2: App.js pääkomponentin sisältämä rakenne, Visual Studio Code ohjelmointiympäristössä.

Forms-komponentti vastaa uusien yksittäisten kirjausten lisäämisestä osaksi tietokantaa sekä muokkausten käsittelystä. Työtehtävän aloitus ja lopetuskellonajan valintaan on käytetty MUI-nimisen yhtiön valmistamaa ja ilmaista React-komponenttia, saman yhtiön vastaavanlaisia komponentteja on käytetty myös muissa työkalun komponenteissa ilman kellonajan valintaa. Forms komponenttiin on lisäksi ohjelmoitu sisään funktioita, joiden avulla mahdollisia virhetilanteita voidaan välttää kuten popup joka aukeaa ja vaatii hyväksynnän mikäli käyttäjä on lisäämässä päivystystehtävää muun työntekijän varallaolovuorolle sekä vaatimus että lopetuspäiväys sekä kellonaika ovat myöhemmin kuin aloituksen ajankohta. Virhetilanteiden välttämistä edesauttaa myös komponentin visuaalisen ilmeen muutos tavalliseen verrattuna, kun kyseessä on vuoron muokkaus.

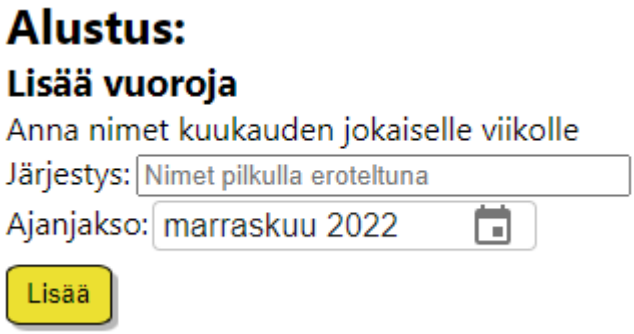



Kuva 3: Yllä Forms-komponentin tavallinen kirjauksen lisäys ja alapuolella näkymä kun kyseessä on olemassa olevan kirjauksen muokkaus.

Visuaaliselta ilmeeltään työkalun toiminnalliset komponentit ovat hyvin pelkistettyjä sillä niiden suhteen ei ollut minkäänlaisia vaatimuksia ja toiveita, tärkeää oli kuitenkin että työkalua olisi selkeää ja helppoa käyttää. Filter on komponenteista toiminnallisuudeltaan yksinkertaisin, sen avulla voidaan suodattaa näkyviin halutut vuorot. Se sisältää valintamahdollisuudet työtyypille, työntekijälle, kuukaudelle sekä halutaanko vuorot nousevaan vai laskevaan järjestykseen päivämäärän perusteella. Mikäli työntekijäksi valitaan yksittäinen työntekijä näyttää suodatettu lista kyseisen työntekijän kokonaistuntimäärän valitulta ajanjaksolta.

Initialize-komponentilla saadaan luotua varallaolovuoroja useaksi viikoksi kerrallaan, ylärajaa kerralla alustettaville viikoille ei ole joten varallaolovuorot voi lisätä tietokantaan kerralla vaikka koko vuodeksi. Alustaessa vuoroja komponentti tarkastaa onko valitulla ajanjaksolla jo varallaolovuoroja tietokannassa mikäli vuoroja löytyy alkaa alustus kyseisestä maanantaista kello 16 ja päättyy seuraavaan maanantaihin klo 8, jos vuoroja ei kuitenkaan kuukauden kohdalla ole merkittynä alkaa se kuun ensimmäisestä päivästä lisätä varallaolovuoroja vaihtaen työntekijää aina maanantaina

kello 16 kirjauksesta eteenpäin niin monta kertaa kuin nimiä on listaan kirjoitettuna. Kirjoitetut nimet lisäksi tarkastetaan ennen vuorojen alustamista jotta varmistetaan että tietokantaan ei lisätä kirjauksia henkilöille, jotka eivät ole varallaolo- ja päivystystehtävien piirissä.




Kuva 4: Alustuksen ilme vastaa muita komponentteja, nimille tarkoitettu laatikko muistuttaa käyttäjää oikeasta tavasta lisätä ne listaan.

Exports-komponentti tuo käyttäjälle näkyviin ainoastaan mahdollisuuden valita kuukauden ja vuoden sekä lataus-painikkeen. Visuaalisesti yksinkertaisesta ilmeestään huolimatta kyseessä on työkalun toiminnan kannalta varmasti tärkein komponentti sillä se muodostaa laskelmat valitun ajanjakson kirjauksista ja niiden perusteella luo taulukot Excel-tiedostoon joka voidaan lähettää suoraan palkanlaskentaan kuukauden päätteeksi. Komponentin laskukaavoihin lisättiin työkalun tuotantokäyttöön ottamisen jälkeen sen käyttäjien toiveesta muuttuja jonka totuusarvoa muuttamalla voidaan valita lasketaanko päivystystuntien palkkaluokka työtehtävän alkamis- vai loppumisajan perusteella. Tuotettu Excel-tiedosto sisältää erilliset sivut sekä palkkakoontiin joka laskee viikkokohtaiset tuntimäärät palkkaluokkiin että sivun josta voi tarvittaessa katsoa vielä kaikki kuukauden vuorot korostaen rivitiedot joissa palkkaluokka on kahdesta vaihtoehdosta suurempi. Pyhäpäivät otetaan laskelmissa myös huomioon, työkaluun on sisällytetty arkipyhat.json niminen tiedosto joka sisältää päivämäärät ja pyhäpäivien nimet tuleville viidelle vuodelle.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | MARY | HENO | VVVV | TKNO | PALA | MAARA | PVM8 | PVML |
| 2 | TAPKOODI | 1 | 28100 | 2022 | 12 | 1 | 69 | 1.11.2022 | 7.11.2022 |
| 3 | TAPKOODI | 1 | 28100 | 2022 | 12 | 2 | 47 | 1.11.2022 | 7.11.2022 |
| 4 | TAPKOODI | 1 | 28100 | 2022 | 12 | 4 | 3 | 1.11.2022 | 5.11.2022 |
| 5 | TAPKOODI | 1 | 28100 | 2022 | 12 | 3 | 1 | 1.11.2022 | 5.11.2022 |
| 6 | TAPKOODI | 1 | 1234 | 2022 | 12 | 1 | 104 | 7.11.2022 | 14.11.2022 |
| 7 | TAPKOODI | 1 | 1234 | 2022 | 12 | 2 | 24 | 7.11.2022 | 14.11.2022 |
| 8 | TAPKOODI | 1 | 567 | 2022 | 12 | 1 | 104 | 14.11.2022 | 21.11.2022 |
| 9 | TAPKOODI | 1 | 567 | 2022 | 12 | 2 | 24 | 14.11.2022 | 21.11.2022 |
| 10 | TAPKOODI | 1 | 28602 | 2022 | 12 | 1 | 104 | 21.11.2022 | 28.11.2022 |
| 11 | TAPKOODI | 1 | 28602 | 2022 | 12 | 2 | 24 | 21.11.2022 | 28.11.2022 |
| 12 | TAPKOODI | 1 | 28100 | 2022 | 12 | 1 | 40 | 28.11.2022 | 30.11.2022 |
| 13 | | | | | | | | | |
| 14 | Vara arki 50% | 1 | | | | | | | |
| 15 | Vara arki 100% | 2 | | | | | | | |
| 16 | Päivystys 100% | 3 | | | | | | | |
| 17 | Päivystys 200% | 4 | | | | | | | |

Kuva 5: Esimerkki Exceliin muodostetusta palkkakoonnista, vain ensimmäisellä kuukauden viikolla on kirjattu päivystystehtäviä. Osan sarakkeista tietoja muutettu tietosuojan vuoksi.

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | **Päivystys Marraskuu 2022** | | | | | | |
| 2 | | | | | | | |
| 3 | | **Nimi** | **PVM** | **Alkuaika** | **Loppuaik** | **Tunnit** | |
| 4 | **Ti** | Eetu | 1.11.2022 | 00:00 | 08:00 | 8:00 | |
| 5 | **Ti** | Eetu | 1.11.2022 | 16:00 | 22:45 | 6:45 | |
| 6 | **Ke** | Eetu | 2.11.2022 | 00:45 | 08:00 | 7:15 | |
| 7 | **Ke** | Eetu | 2.11.2022 | 16:00 | 16:50 | 0:50 | |
| 8 | **Ke** | Eetu | 2.11.2022 | 17:50 | 00:00 | 6:10 | |
| 9 | **To** | Eetu | 3.11.2022 | 00:00 | 08:00 | 8:00 | |
| 10 | **To** | Eetu | 3.11.2022 | 16:00 | 00:00 | 8:00 | |
| 11 | **Pe** | Eetu | 4.11.2022 | 00:00 | 08:00 | 8:00 | |
| 12 | **Pe** | Eetu | 4.11.2022 | 16:00 | 00:00 | 8:00 | |
| 13 | **La** | Eetu | 5.11.2022 | 00:00 | 09:33 | 9:33 | |
| 14 | **La** | Eetu | 5.11.2022 | 10:33 | 00:00 | 13:27 | |
| 15 | **Su** | Eetu | 6.11.2022 | 00:00 | 00:00 | 24:00 | |
| 16 | **Ma** | Eetu | 7.11.2022 | 00:00 | 08:00 | 8:00 | |
| 17 | | | | | | | |
| 18 | | | | | 50% | 69:00 | |
| 19 | | | | | 100% | 47:00 | |
| 20 | | | | | | | |
| 21 | **Varalta töihin** | | | | | | **Selitys** |
| 22 | **Ti** | Eetu | 1.11.2022 | 22:45 | 00:45 | 2:00 | Järjestelmän kommunikaatiovirhe |
| 23 | **Ke** | Eetu | 2.11.2022 | 16:50 | 17:50 | 1:00 | Verkkohäiriöitä |
| 24 | **La** | Eetu | 5.11.2022 | 09:33 | 10:33 | 1:00 | Ongelmia kirjautumisessa |
| 25 | | | | | | | |
| 26 | | | | | 100% | 1:00 | |
| 27 | | | | | 200% | 3:00 | |

Kuva 6: Esimerkki Exceliin muodostetusta kaikkien kirjausten listasta. Huomioväri viittaa korkeamman palkkaluokan kirjaukseen. Lauantai 5.11 on 100% varallaoloa, sillä kyseessä on pyhäinpäivä.

Valtaosan työkalun käyttöliittymästä täyttävä lista vuoroista sisältää paikallaan pysyvät otsikot sekä listauksen kaikista Filter-komponentin ehdot täyttäivästä kirjauksista. Kun suodatuksiin on valittu yksittäinen työntekijä ilmestyy ”Kesto”-otsikon yläpuolelle tuntimäärä kyseisen työntekijän kuukauden kirjauksille. Jokaisella kirjauksella on lisäksi painikkeet joista voi tarvittaessa ottaa kirjauksen muokattavaksi Forms-komponenttiin tai halutessaan poistaa vuoron. Poistettaessa ennen varsinaista poisto-toimenpidettä käyttäjän on vahvistettava popup ikkunasta että haluaa poistaa kyseisen kirjauksen. Työkalu ottaa lisäksi huomioon poistoissa ja muokkauksissa kyseistä kirjausta ennen ja jälkeen olevat kirjaukset ja tarvittaessa muuttaa myös ne oikeanlaisiksi jotta järjestelmään ei jää aukkoja kellonaikoihin tai tule päällekkäisyyksiä.

| Tyyppi | Työntekijä | Aloituspvm ja aika | Lopetuspvm ja aika | Kesto | Työn kuvaus | |
|---|---|---|---|---|---|---|
| Varallaolo | Eetu | **Ti** 1.11.2022 00:00 | 1.11.2022 08:00 | 8:00 | | Muokkaa Poista |
| Varallaolo | Eetu | **Ti** 1.11.2022 16:00 | 1.11.2022 22:45 | 6:45 | | Muokkaa Poista |
| Päivystys | Eetu | **Ti** 1.11.2022 22:45 | 2.11.2022 00:45 | 2:00 | Järjestelmän kommunikaatiovirhe | Muokkaa Poista |
| Varallaolo | Eetu | **Ke** 2.11.2022 00:45 | 2.11.2022 08:00 | 7:15 | | Muokkaa Poista |
| Varallaolo | Eetu | **Ke** 2.11.2022 16:00 | 2.11.2022 16:50 | 0:50 | | Muokkaa Poista |
| Päivystys | Eetu | **Ke** 2.11.2022 16:50 | 2.11.2022 17:50 | 1:00 | Verkkohäiriöitä | Muokkaa Poista |
| Varallaolo | Eetu | **Ke** 2.11.2022 17:50 | 3.11.2022 00:00 | 6:10 | | Muokkaa Poista |

Kuva 7: Kirjausnäkymä jossa esimerkkivuoroja.

```
<div className="AllShifts">
    <div className="rowLabels">
        <h3 className="typeHeader">Tyyppi</h3>
        <h3 className="workerHeader">Työntekijä</h3>
        <h3 className="startHeader">Aloituspvm ja aika</h3>
        <h3 className="endHeader">Lopetuspvm ja aika</h3>
        <h3 className="timeHeader">Kesto</h3>
        <h3 className="descHeader">Työn kuvaus</h3>
    </div>

    <div className="listedShifts" id="listedShifts">
        {filteredShifts.map(({id,tyotyyppi,tyontekija,aloituspvm,aloitusaika,lopetuspvm,lopetusaika,selite,tyoaika,viikonpaiva},index) => {
            return (
                <Shift
                key={tyontekija + "-" +index}
                id={id}
                tyotyyppi = {tyotyyppi}
                tyontekija = {tyontekija}
                aloituspvm = {aloituspvm}
                aloitusaika = {aloitusaika}
                lopetuspvm = {lopetuspvm}
                lopetusaika = {lopetusaika}
                selite = {selite}
                tyoaika = {tyoaika}
                viikonpaiva = {viikonpaiva}
                />
            );
        })}
    </div>
</div>
```

Kuva 8: Vuorolistan muodostaminen päätelaitteelle tapahtuu JavaScript funktiolla: map. Visual Studio Code

Kun vuorolistaan tehdään muutoksia tulee siitä lähettää tieto palvelimelle, samoin palvelimelta on saatava jo kirjatuista vuoroista tiedot takaisin käyttöliittymään. Tämä tapahtuu käyttämällä asynkronisia funktioita, jotka odottavat kunnes saavat vastauksen palvelimelta. Esimerkiksi kaikki vuorot palauttava ”getAllShifts” funktio suoritetaan kun käyttäjä lataa työkalun verkkosivun ja on antanut oikeat kirjautumistunnukset. Myös muut palvelimen kanssa kommunikoivat funktiot vaativat että käyttäjä on Login-komponentin sisällä antanut oikeat tiedot.

```
25  const getAllShifts = async() =>{
26
27      return await axios.get(baseURL).catch(error => errorHandler(error))
28  }
29
30  const addShift = async(shiftinfo)=>{
31      try {
32        const response = await axios.post(baseURL, shiftinfo)
33      } catch (err){
34        errorHandler(err)
35      }
36  }
37
38  const initAddShifts = async(shiftArr)=>{
39      try {
40          shiftArr.forEach(shift => {
41            axios.post(baseURL,shift).catch(error => errorHandler(error))
42          })
43      console.log("Multiple shifts added")
44      } catch (error) {
45        errorHandler(error)
46      }
47
48  }
49
50  const deleteShift = async(shift) =>{
51      try {
52        const response = await axios.delete(`${baseURL}/${shift.id}`)
53
54      } catch (error) {
55        errorHandler(error)
56      }
57  }
```

Kuva 9: Funktiot joiden avulla käyttöliittymä kommunikoi palvelimelle. Visual Studio Code.

## 5.2 Palvelin

Palvelin toteutettiin Node sekä Express.js pohjaisena ratkaisuna, joka oli mahdollista sisällyttää Serverless arkkitehtuurin hyväksymään muotoon käyttämällä ”serverless-http” npm-pakettia. Palvelimelle määritellään väliohjelmistoja, jotka varmistavat, että sen vastaanottama data ja lähettämä data on muodoltaan oikeanlaista, lisäksi määritetään reitit, jotka vastaavat käyttöliittymän yhteydenottoihin sekä muut haluttavat väliohjelmistot, joiden avulla voidaan esimerkiksi ylläpitää lokia yhteydenotoista tai käsitellä virheitä.

```
1   const serverless = require('serverless-http')
2   const express = require('express')
3   const cors = require('cors')
4   const middleware = require('./utils/middleware')
5   const config = require('./utils/config')
6   const app = express()
7
8   app.use(express.json())
9
10
11  app.options('*',cors())
12
13  app.use('/api/shifts', require('./routes/shiftroutes'))
14
15
16  app.use(middleware.unknownEndpoint)
17  app.use(middleware.errorHandler)
18
19  module.exports.handler = serverless(app);
```

Kuva 10: Palvelin koodin päänäkymä. Visual Studio Code.

Riippuen käyttöliittymässä tehdystä toiminnosta ottaa käyttöliittymä yhteyttä palvelimen tiettyyn reittiin, reitin perusteella palvelin muodostaa yhteyden tietokantaan. Yhteyden muodostumisen onnistuessa palvelin lähettää SQL kyselyn, jonka perusteella tietokanta joko lisää, poistaa tai tuo palvelimelle dataa. Alla olevassa esimerkissä käyttöliittymältä lähetetty datan perusteella muodostetaan SQL kysely, joka poistaa halutun kirjauksen tietokannasta. Poiston onnistuessa palvelin lähettää käyttöliittymään onnistuneesta poistosta tiedon ja vastaavasti tapahtuu myös epäonnistuneen poiston kohdalla.

```
const deleteShift = asyncHandler(async (req, res) => {
    const id = req.params.id


    pool.getConnection((err,conn) => {
      if(err) throw err;

      try {
        conn.query(`DELETE FROM SHIFTS WHERE id = ?`,[id],(error,result) =>{
        conn.release();
        res.status(200).send("Poisto onnistui")
        if(error) throw error;
        })
      }
      catch(error){
        console.log(error)
        res.status(404).send("Vuoron poisto epäonnistui")
      }
    })

  })
```

Kuva 11: Palvelimen kontrolleri vuorojen poistoille tietokannasta. Visual Studio Code

Serverless arkkitehtuurin mahdollistama palvelimeton toiminta vaatii lisäksi yhden, muusta projektin koodista poikkeavan yml-tiedostomuotoa mukailevan tiedoston, jossa määritetään esimerkiksi ympäristön tärkeät muuttujat, pilvipalvelu ja sen käyttämän palvelimen alue ja käsittelijät eri funktioiden tapahtumille. Alla olevasta kuvasta on mahdollista nähdä kyseisen määritystiedoston rakenne. Tietoturvan kannalta tärkeät elementit kuten sisäisen verkon tiedot sekä ympäristön muuttujat on piilotettu.

```yaml
service: support-backend

useDotenv: true

provider:
  name: aws
  runtime: nodejs16.x
  stage: dev
  region: eu-central-1
  vpc:
    securityGroupIds:
      -
    subnetIds:
      -
  environment:
    USER:
    SECRET:
    ENDPOINT:
    ENDPOINT_PORT:
    DB_NAME:
  httpApi:
    cors: true

functions:
  app:
    handler: index.handler
    timeout: 10
    events:
      - httpApi:
          method: '*'
          path: /api/shifts

      - httpApi:
          method: '*'
          path: /api/shifts/{proxy+}
```

Kuva 12: Serverless määrittelytiedosto, osa tiedoista on piilotettu, Visual Studio Code

## 5.3 AWS

Työkalun valmis palvelin sekä käyttöliittymä asennetaan toimivaksi 02 Palvelut Oy:n sisäisessä verkossa käyttämällä AWS:n VPC palvelua näin työkaluun ja sen tietoihin ei pääse yrityksen sisäisen verkon ulkopuolelta. Muut työkalun käyttämiseen tarvittavat palvelut ovat myös AWS:n tarjoamia: Tietokanta muodostetaan Relational Database Servicellä perustamalla oma instanssi työkalun MariaDB tietokannalle. Simple Storage Servicen (AWS S3) sisään tallennetaan käyttöliittymän koontiversio ja muodostetaan URL-osoite, jonka kautta käyttäjä pääsee työkaluun käsiksi. Palvelin puolestaan käyttää Cloudfrontia, AWS Lambdaa ja API Gatewayta,Cloudfrontin avulla saadaa palvelimelle URL joka vastaanottaa kutsut käyttöliittymältä. Serverless arkkitehtuurin käyttö palvelimessa mahdollistaa Lambdan käytön, näin ollen palvelin ei ole jatkuvasti päällä vaan pelkästään silloin kun käyttöliittymältä sitä kutsutaan. Tämä on

perinteistä palvelinratkaisua kustannustehokkaampi tapa varsinkin, kun kyselyjä palvelimelle tulee vuodessa noin muutama tuhat.

# 6 HAASTEET JA JATKOKEHITYS

Teknisen toteutuksen osalta haastavimmaksi asiaksi muodostui laskukaavojen muodostaminen, jotta jokainen tunti lasketaan oikeaan palkkaluokkaan, haasteen hahmottamisessa ja ratkaisemisessa parhaaksi tavaksi osoittautui erilaisten vuorojen hahmotteleminen käyttäen avuksi kynää ja paperia. Toinen hieman päänvaivaa aiheuttanut aspekti oli Serverless-arkkitehtuurin hyödyntäminen projektissa, sillä ennen projektin alkua se oli itselleni täysin tuntematon tapa toteuttaa palvelinratkaisuja, suureksi onneksi Serverless on suosittu palvelinratkaisu, näin ollen siihen liittyen löytyi runsaasti dokumentaatiota, jonka avulla ongelmatilanteista selvittiin nopeassa aikataulussa.

Käyttäjän vikatilanteiden estäminen ennen sellaisten tapahtumista oli käyttöliittymän osalta haaste ja ennakkoon myös pieni huolenaihe sillä kyseisen datan avulla maksettaisiin kuitenkin työntekijöiden palkkoja. Haasteen ratkaisemiseksi työkaluun on tehty useita erilaisia tarkastus metodeja, jotta esimerkiksi virheellinen käyttäjäsyöte on mahdollista estää ennen tietojen lisäämistä. Syötteen tarkastamisen lisäksi työkaluun lisättiin erilliset popup-huomiot tilanteisiin, jos käyttäjä pyrkii lisäämään toisen työntekijän ajankohdalle omaa päivystystehtävää taikka poistamaan listasta jo tehtyjä kirjauksia.

Työkalua on mahdollista myös jatkokehittää. Suurin potentiaalinen kohde jatkokehitykselle on tuki ylitöiden kirjauksille ja laskemiselle. Ylitöiden kirjaaminen on työkaluun jo mahdollistettu samalla lomakkeella, joka on käytössä päivystysten kirjaamiseen valitsemalla työtyyppi-valikosta vaihtoehdon ”Ylityö”, tällainen vuorokirjaus myös poistaa työntekijältä tunteja varallaolovuoroista, mikäli ylityökirjaus on saman henkilön varallaoloviikolla. Jos kuukausi, joka on Export-komponentissa valittuna sisältää ylityökirjauksia, muodostetaan niiden osalta samaan Exceliin oma sivu, jossa ylityöt lasketaan omiin palkkaluokkiin sekä listataan kaikki kirjatut ylityöt listaan. Ylityöt on tässä tapauksessa määritetty kuitenkin jatkokehityskohteeksi, sillä työntekijöillä on liukuva työaika ja työkalu ei ole yhteydessä järjestelmään, joka laskee varsinaiset työpäivät ja niiden työtunnit, työkalua ei myöskään toistaiseksi haluttu kyseiseen järjestelmään yhdistää. Tämän vuoksi työkalussa tehtävät laskutoimitukset

ylitöille on tehtävä, sillä olettamuksella että jokainen työntekijä tekee vuoronsa arkisin klo 8–16 välillä.

Toinen jatkokehityskohde on työkaluun kirjautumisen kehittäminen entistä tietoturvallisemmaksi. Käyttöliittymässä tapahtuva kirjautuminen on mahdollista ohittaa, jos pahaa tarkoittava käyttäjä tietää mitä tekee. Yhdessä työkalua käyttävien työntekijöiden kanssa kuitenkin totesimme, että koska työkaluun pääsee käsiksi ainoastaan sisäisen verkon sisällä ja työkalun URL osoite on tiedossa vain kourallisella työntekijöitä, olisi tämä alustavasti riittävä tapa suojata kirjaukset ja niiden sisältämä data. Kirjautumista olisi mahdollista kehittää liittämällä kirjautuminen käyttämään AWS Cognito palvelua, jonka yhteyteen luotaisiin käyttäjäryhmä, joka ainoastaan saisi pääsyn työkaluun, itse kirjautuminen tapahtuisi tässä tapauksessa AWS tunnuksella.

# 7 JOHTOPÄÄTÖKSET

Lähdettäessä työkalua suunnittelemaan ja toteuttamaan muodostettiin tutkimuskysymyksiksi: ”Miten varallaolovuorojen sekä päivystystehtävien merkitsemistä on mahdollista sujuvoittaa?” sekä ”Kuinka vuorojen laskeminen oikeisiin palkkaluokkiin voitaisiin automatisoida?”. Työkalun avulla tutkimuskysymyksiin on vastattu luomalla järjestelmä, jota käyttämällä työntekijöiden on mahdollista vähentää kirjauksiin käytettyä aikaa. Aikaisemmin käyttäjä on itse joutunut tekemään muutokset kirjauksiin, joihin uusi kirjaus tai sen poisto vaikuttaa, työkalun avulla käyttäjä voi tehdä kirjauksen tehdystä työtehtävästä ja työkalu tekee muihin vuoroihin vaaditut muutokset kirjausta lisättäessä, muutettaessa sekä poistettaessa. Varallaolovuorojen lisäämisen sujuvoittamiseksi työkaluun muodostettiin komponentti, jonka avulla vuorot voi lisätä halutessaan niin pitkälle ajanjaksolle, kuin on tarve kirjoittamalla vain työntekijöiden nimet halutussa järjestyksessä, tällaisessa tilanteessa työkalu lisää vuorot nimijärjestyksessä ja oikean pituisina vuoroina päättäen vuorojen alustuksen aina uuteen alkavaan varallaoloviikkoon. Palkkojen laskemisen automatisointiin luotiin kaavat, jotka käsittelevät halutulta ajanjaksolta kaikki kirjaukset ensin niiden työtyyppien perusteella ja sen jälkeen laskemalla viikko kerrallaan työntekijöiden kirjaukset niiden ajankohtien perusteella osaksi oikeaa palkkaluokkaa. Palkkakoonnin lisäksi muodostetaan myös erillinen sivu, joka sisältää kaikki kyseisen ajanjakson vuorot sekä työtehtäviin tehdyt huomiot.

Työkalu otettiin testikäyttöön vanhan kirjausjärjestelmän ohelle elokuun 2022 lopussa ja täysin tuotantokäyttöön lokakuusta 2022.

# LÄHTEET


020202 Palvelut, (n.d.), Tutustu Nollakakkoseen!, haettu 21.9.2022 https://www.020202.fi/020202-palvelut/

Bhosale V. (2021), What is a Rest API…? [Kuva 1] https://www.codementor.io/@vishalbhosle/what-is-a-rest-api-1e5s5gefgc

Cloudflare, (n.d.), What is serverless computing? haettu 8.10.2022 https://www.cloudflare.com/learning/serverless/what-is-serverless/

Codecademy (n.d.), What is REST haettu 8.10.2022 https://www.codecademy.com/article/what-is-rest

Hoque S, (2018), Full-Stack React projects: modern web development using React 16, Node, Express, and MongoDB. Packt Publishing

Mardan, A, (2014). Pro Express.js: Master Express.js: The Node.js Framework For Your Web Development (1st ed. 2014.). Apress.

MariaDB, (n.d.), New to MariaDB Server? haettu 8.10.2022 https://mariadb.org/

MariaDB. (22.4.2021) MariaDB Roadshow 2021: MariaDB Overview [video]. Youtube https://www.youtube.com/watch?v=aurboE_q1Yg

Morris S, (n.d.), Tech 101: What Is React JS? haettu 7.10.2022 https://skillcrush.com/blog/what-is-react-js/

React, (n.d) Declarative, haettu 7.10.2022 https://reactjs.org/

Red Hat, (2020), What is a REST API? haettu 8.10.2022 https://www.redhat.com/en/topics/api/what-is-a-rest-api

SPA, (2022), SPA (Single-page application), haettu 13.10.2022 https://developer.mozilla.org/en-US/docs/Glossary/SPA

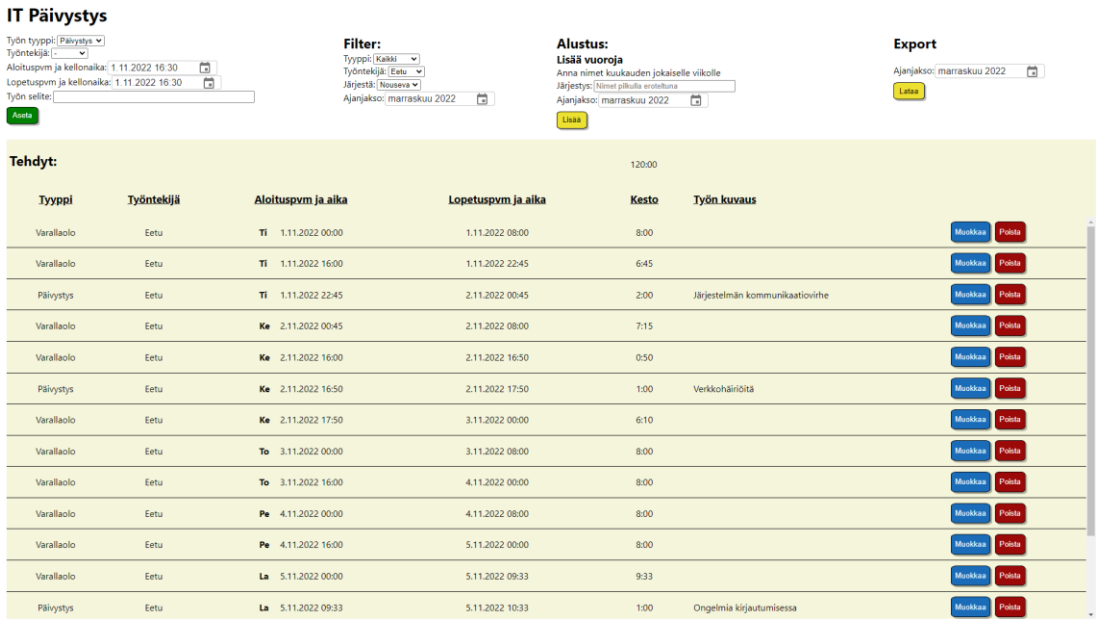
Stack Overflow Developer Survey, (2021), Most popular technologies, haettu 7.10.2022 https://insights.stackoverflow.com/survey/2021#overview

Stack Overflow Developer Survey, (2021) Web frameworks, haettu 7.10.2022 https://insights.stackoverflow.com/survey/2021#most-popular-technologies-webframe

bin Uzayr, S., Cloud, N. & Ambler, T. (2019). JavaScript Frameworks for Modern Web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now. (2. painos)

W3C, (n.d.) HTML & CSS, Haettu 7.10.2022 https://www.w3.org/standards/webdesign/htmlcss

What is JavaScript?, (2022) A high-level definition, haettu 7.10.2022 https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript




Liite 1: kuva käyttöliittymästä kokonaisuudessaan