TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

EETU KOIVU

NFC-LOUNASLIPPUJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS

Diplomityö

Tarkastaja: professori Kari Systä
Tarkastaja ja aihe hyväksytty
1.3.2017

TIIVISTELMÄ

EETU KOIVU: NFC-lounaslippujärjestelmän suunnittelu ja toteutus
Tampereen teknillinen yliopisto
Diplomityö, 54 sivua
Huhtikuu 2017
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Kari Systä

Avainsanat: NFC, lounaslippujärjestelmä, web-sovellus, mobiilisovellus

Tämän diplomityön aiheena on Wapice OY:lle kehitetty lounaslippujärjestelmä. Diplomityössä esitellään lounaslippujärjestelmän kehitysprojektin eri vaiheet: vaatimusmäärittely, teknologiavalinnat, toteutus ja testaus. Lisäksi kerrotaan, miksi tällaista järjestelmää on alettu kehittää, sekä arvioidaan projektin lopputuloksena syntynyttä järjestelmää sille asetettujen vaatimusten perusteella.

Yritykset voivat tarjota työntekijöilleen verotuksessa tuetun lounasedun. Monissa yrityksissä etu tarjotaan antamalla työntekijöille paperisia lounaslippuja, joilla aterioita pystyy maksamaan ravintoloissa. Myös Wapicella on käytössä tällainen lounasetu. Työntekijä ei maksa lounasta rahalla vaan joko paperisella lounaslipulla tai sähköisellä maksukortilla. Joissain ravintoloissa työntekijä vain laittaa merkinnän paperiseen listaan oman nimensä kohdalle. Näiden eri tavoilla tehtyjen kuittausten perusteella ravintolat laskuttavat lounaat Wapicelta, ja Wapice vähentää lounaiden omavastuuosat työntekijöiden palkoista. Työntekijäkohtaiset lounasmäärät lasketaan käsin eri ravintoloiden lähettämistä listoista. Lounaslippujärjestelmän on tarkoitus vähentää käsin tehtävää työtä. Lisäksi siitä kehitetään tuote, jota voidaan tarjota myös muille yrityksille.

Lounaslippujärjestelmään sisältyvät Node.Js-palvelin, MySql-tietokanta sekä neljä erilaista asiakassovellusta: mobiilisovellus, web-käyttöliittymä, Wapicen oma lounaslippujen raportointisovellus sekä integraatio Wapicen tuntikirjausjärjestelmään. Mobiilisovellus toimii Android-puhelimessa. Mobiilisovelluksella luetaan ravintolassa työntekijän NFC-tunniste ja täten kirjataan lounas sen ostaneelle työntekijälle. Yritys käyttää mobiilisovellusta NFC-tunnisteen rekisteröintiin. Web-käyttöliittymä on toteutettu Reactilla. Web-käyttöliittymässä voi tehdä ylläpitotoimintoja, muodostaa raportteja, rekisteröidä NFC-tunnisteita työntekijöille, luoda lounastyyppejä ravintoloille sekä hallinnoida yritysten ja ravintoloiden välisiä sopimuksia. Lounaslippujen raportointisovellus on Wapicen omassa käytössä oleva sovellus, jolla työntekijät ilmoittavat ottamiensa paperisten lounaslippujen kappalemäärät. Tuntikirjausjärjestelmään integroidun näkymän kautta lounaita pystyy lukemaan sekä ravintoloiden lähettämistä PDF-tiedostoista että lounaslippujärjestelmän palvelimelta. Järjestelmä koostaa eri lähteistä luetuista lounaista raportin, jota voidaan sellaisenaan käyttää palkanmaksussa.

Projektissa valmistui toimiva tuote. Tuote ei ole valmis mutta toiminnallisuuden osalta täyttää sille asetetut vähimmäisvaatimukset. Diplomityön kirjoitushetkellä projekti odottaa yrityksen johdon päätöstä projektin jatkumisesta.

ABSTRACT

EETU KOIVU: Design and implementation of an NFC lunch ticket system
Tampere University of Technology
Master of Science Thesis, 54 pages
April 2017
Master's Degree Programme in Information Technology
Major: Software Engineering
Examiner: Professor Kari Systä

Keywords: NFC, lunch ticket system, web application, mobile application

The subject of this master's thesis is a lunch ticket system developed for Wapice Ltd. This thesis introduces different phases of the development project: requirement specification, technology choices, implementation and testing. Additionally, the motivation for creating this kind of a system is explained and the resulting product is evaluated against the requirements set at the beginning.

Companies may offer to pay a part of employees' lunches so that the benefit is tax free. Practically, many companies give lunch vouchers to employees and the vouchers can be used for payment in restaurants. Also Wapice offers a lunch benefit to its employees. Instead of cash, an employee uses a lunch voucher or a payment card to pay for the lunch. Is some restaurants, there is only a list where employees mark lunches. Based on the various payment methods, the restaurants invoice Wapice and Wapice suspends part of the prices from employees' salaries. The number of lunches is calculated manually based on the lists sent by restaurants. The goal of the lunch ticket system is to reduce the amount of manual work. Additionally, it will be a product that may also be offered to other companies.

The lunch ticket system includes a Node.js server, MySql database and four clients: a mobile application, a web interface, a reporting application made for internal use at Wapice and integration to an hour reporting system used by Wapice. Mobile application is developed for Android. It is used by restaurants to receive payments. The application reads an employee's NFC tag in a restaurant and sends the information about a bought lunch to the server. Companies use mobile app for registering NFC tags. Web interface is created using React. Web interface is used for administrative actions, reporting, registering NFC tags for employees, defining lunch types for restaurants and managing contracts between restaurants and companies. Lunch voucher reporting app is used by Wapice employees for reporting the number of the lunch vouchers taken. A view integrated to the hour reporting system is used for reading lunch data from PDF files and the lunch ticket system. Based on the data read from different sources, the system generates a report that can be used in calculating salaries.

A working product was created in this project. The product is not complete but it satisfies the minimum functional requirements. At the moment of writing this thesis, the project is on hold, waiting for decisions about continuation.

ALKUSANAT

Tämä diplomityö on tehty Wapice OY:lle. Työn aiheena on lounaslippujärjestelmä, jota olen ollut kehittämässä kyseessä olevalle yritykselle. Haluan kiittää työn tarkastajaa professori Kari Systää ja ohjaajaa DI Jussi Haapasta ohjauksesta ja arvokkaasta palautteesta. Kiitän myös Wapice OY:tä mahdollisuudesta työn tekemiseen. Lisäksi kiitokset kuuluvat kaikille, joilta olen saanut parannusehdotuksia joko diplomityöhön tai itse järjestelmään.

Tampereella, 23.4.2014

Eetu Koivu

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

Backend	Web-sovelluksen osa, joka suoritetaan palvelimella.
CSS	Cascading Style Sheets, kieli, jolla kuvataan verkkosivun tyyli.
DOM	Document Object Model, selaimen HTML-sivun jäsentämiseen käyttämä malli sivun rakenteesta.
Frontend	Web-sovelluksen osa, joka suoritetaan asiakassovelluksessa.
HTML	Hypertext Modeling Language, verkkosivun rakenteen kuvaamiseen käytettävä kieli.
HTTP	Hypertext Transfer Protocol, selainten ja palvelinten käyttämä tiedonsiirtoprotokolla.
HTTPS	HTTP:n laajennus, jossa tiedonsiirtoyhteys on salattu.
IO	Input/output, oheislaitteet.
JSON	JavaScript Object Notation, tiedon tallennus- ja siirtoformaatti.
Middleware	Ohjelmistokomponentti, joka loogisesti sijaitsee asiakkaan ja palvelimen välissä ja tarjoaa palveluita.
NDEF	NFC Data Exchange Format, formaatti, jolla tietoa yleisesti tallennetaan NFC-tunnisteelle.
NFC	Near Field Communication, lyhyen kantaman langaton tiedonsiirtoteknologia.
OCR	Optical Character Recognition, tekstin tunnistaminen kuvasta.
PoC	Proof of Concept, tietyn idean tai menetelmän toteutus, jolla varmistetaan sen toteutuskelpoisuus.
REST	Representational State Transfer, HTTP-protokollaan perustuva arkkitehtuurimalli web-rajapintojen toteuttamiseen.
SQL	Relaatiotietokantojen kyselykieli.

.

1. JOHDANTO

Monet yritykset tarjoavat työntekijöilleen lounasetua työsuhde-etuna. Yrityksillä on vaihtelevia käytäntöjä siihen, miten lounasedun tarjoaminen on toteutettu. Monissa yrityksissä työntekijöille annetaan lounaslippuja tai -seteleitä, jotka käyvät maksuvälineinä tietyissä ravintoloissa.

Tämän diplomityön aiheena on Wapicelle kehitetty lounaslippujärjestelmä, joka käyttää NFC-teknologiaa työntekijän tunnistamiseen. Projektin tavoitteena on saada luotua järjestelmä, joka voidaan ottaa käyttöön Wapicen sisällä, mutta jota voitaisiin myös myydä muille yrityksille.

Lounaslippujärjestelmä toimii siten, että jokaiselle yrityksen työntekijälle annetaan henkilökohtainen NFC-tunniste. Ostaessaan lounaan käyttäjä tunnistautuu mobiililaitteelle ravintolassa omalla tunnisteellaan. Lounaan ostamisesta tulee merkintä järjestelmään. Web-käyttöliittymän kautta sekä ravintolat että yritykset pystyvät saamaan raportteja, joista näkyvät esimerkiksi yrityksen työntekijöiden ostamat lounaat ravintoloittain tai ravintolassa ostettujen lounaiden kokonaismäärät yrityskohtaisesti. Raporttien avulla yritys pystyy perimään lounaiden hinnat oikeilta työntekijöiltä ja ravintola pystyy laskuttamaan yrityksiä ostettujen lounaiden perusteella.

Tehtäväni oli toteuttaa projekti kokonaisuudessaan. Projektin alkuvaiheessa laadittiin vaatimusmäärittely, jota on päivitetty projektin edetessä. Aluksi vaatimuksia ei juuri tullut ulkopuolelta, vaan toiminnallisuuden määrittely sisältyi osaksi projektia. Myös projektissa käytettävät teknologiat olivat vapaasti valittavissa. Perustoiminnallisuuden määrittelyn jälkeen, alkoi järjestelmän toteuttaminen valituilla teknologioilla. Toteutuksen kanssa rinnakkain tapahtui järjestelmän testaaminen. Koska tavoitteena oli saada prototyyppi nopeasti valmiiksi, testejä kirjoitettiin vain tärkeimmille komponenteille ja muu testaus tehtiin tutkivan testauksen menetelmillä. Tämän diplomityön rakenne noudattaa projektin vaiheita. Aluksi määritellään suunniteltava järjestelmä. Luvussa 2 kerrotaan taustatietoa siitä, miksi tällaista järjestelmää on alettu kehittää ja mitkä ovat sille keskeiset tavoitteet. Luvussa 3 analysoidaan järjestelmälle asetettuja vaatimuksia tarkemmin. Luvussa 4 esitellään järjestelmän toteuttamiseen valitut teknologiat ja luvussa 5 järjestelmän toteutus ja rakenne. Lopuksi kerrotaan, miten järjestelmää on testattu ja verrataan toteutettua järjestelmää projektin alussa asetettuihin tavoitteisiin.

2. TAUSTAA

Lounaslippujärjestelmän kehittämiselle oli kaksi päätavoitetta. Tavoitteista ensimmäinen on manuaalisen työn vähentäminen prosessien automatisoinnilla. Tällä hetkellä lounaiden hinnat vähennetään Wapicen työntekijöiden palkoista manuaalisesti. Toinen tavoite on, että järjestelmästä tulisi valmis tuote, jota voitaisiin tarjota myös muiden yritysten käyttöön.

Lounaslippujärjestelmästä hyötyvät kaikki käyttäjäryhmät: yritykset pystyvät tarjoamaan lounaslippuja työntekijöilleen työsuhde-etuna ja saamaan järjestelmän kautta raportteja, joissa näkyy jokaisesta ostetusta lounaasta ostoajankohta, lounaan ostanut työntekijä sekä ravintola, josta lounas on ostettu. Myöhemmin yrityksen on myös mahdollista saada räätälöityjä raportteja, mikä takaa sen, että raportit sisältävät juuri ne tiedot, joista yritys on kiinnostunut, ja ovat sellaisessa muodossa, että ne voidaan esimerkiksi syöttää sellaisenaan yrityksen muihin tietojärjestelmiin. Työntekijä pystyy ostamaan lounaita edullisemmin lounaslippujärjestelmän ansiosta, koska valtio tukee lounasetua verotuksen kautta. Paperisiin lounaslippuihin verrattuna NFC-tunnistukseen perustuva lounaslippujärjestelmä on työntekijän näkökulmasta myös käyttäjäystävällisempi, koska työntekijän ei tarvitse kuljettaa mukana useita lounaslippuja ja muistaa ottaa mukaan uusia, kun vanhat lounasliput on käytetty. Ravintolalle lounaslippujärjestelmä tuo kilpailuetua verrattuna ravintoloihin, joissa lounaslippujärjestelmä ei ole käytössä. Ravintola saa lounaslippujärjestelmän myötä lisää asiakkaita lounaslippujärjestelmää käyttävien yritysten työntekijöistä. Myös ravintolan osalta raportointimahdollisuudet paranevat ja manuaalisen työn osuus vähenee. Lisäksi lounaslippujärjestelmä voisi tulevaisuudessa jopa lähettää laskuja automaattisesti. Wapice laskuttaa yrityksiä kuukausittain järjestelmän käyttämisestä ja hyötyy tätä kautta rahallisesti.

2.1 Wapicen aikaisempi lounaslippujärjestelmä

Vuonna 2017 sovelletaan verohallinnon joulukuussa 2016 antamaa päätöstä luontoisetujen laskentaperusteista. Päätöksen mukaan yritys voi tarjota ruokailun työntekijöilleen luontoisetuna. Jos lounasedun toteuttamiseen käytetään ruokalipukkeita tai muuta vastaavaa maksutapaa, lounasedun arvo on 75 % ruokalipukkeen nimellisarvosta, kuitenkin vähintään 6,40 € ja enintään 10,30 €. Jos yritys maksaa lounaasta yli 25 %, yli menevä osa on työntekijältä verotettavaa palkkatuloa. Jotta veroetua voidaan käyttää, maksuvälineen on toteutettava aiemmin mainitut ehdot ja oltava henkilökohtainen. Lisäksi maksuvälinettä ei saa pystyä käyttämään muuhun kuin ruokailuun. Sillä voi ostaa valmis-

ruokaa kaupan ruokatiskistä mutta ei einesruokaa eikä elintarvikkeita. (Verohallinto 2016)

Veroedun hyödyntämiseksi monet yritykset tarjoavat ruokailun työntekijöilleen. Wapicella ruokailu on toteutettu siten, että työntekijät käyvät ravintoloissa syömässä, ravintolat laskuttavat lounaat Wapicelta, ja osa lounaiden hinnoista vähennetään työntekijöiden palkoista. Eri toimipisteissä käytettävät toimintatavat vaihtelevat suuresti: Tampereella työntekijät tulostavat lounaslippuja ja kuittaavat paperiseen listaan ottamiensa lounaslippujen lukumäärän. Ravintoloissa työntekijät maksavat antamalla lounaslipun kassalle. Seinäjoella työntekijöillä on NFC-kortit, joilla he tunnistautuvat ravintoloissa. Lisäksi joissakin ravintoloissa työntekijöitä varten on olemassa paperinen lista, johon he kuittaavat ostamansa lounaan laittamalla rastin omaan sarakkeeseen.

Tieto siitä, kuinka monta lounasta kukin työntekijä on ostanut, tulee palkkatoimistolle siis monesta eri lähteestä ja monessa eri muodossa. Useimmat ravintolat lähettävät laskun yhteydessä listan ostetuista lounaista PDF-tiedostona. Eri ravintoloiden lähettämät tiedostot ovat erilaisia rakenteeltaan. Ravintoloissa, joissa on käytössä paperiset lounasliput, ei ole tietoa, kuka on lounaan syönyt, vaan tieto tulee paperisesta kuittauslistasta Wapicen toimistolta. Lisäksi nämä ravintolat lähettävät palkkatoimistolle kerätyt lounasliput, jotka laskemalla varmistetaan lounaiden lukumäärän paikkansapitävyys. PDF-tiedostot voivat olla tekstimuotoisia tai kuvia, esimerkiksi skannattuja paperilistoja. Näistä listoista lounaiden lukumäärät lasketaan manuaalisesti ja niistä koostetaan Excel-tiedosto, joka voidaan syöttää palkanmaksuohjelmaan. Manuaalista työtä vaaditaan siis nykyisellään eri menetelmissä huomattavasti.

Uusi lounaslippujärjestelmä automatisoi prosessia usealla eri tavalla. Järjestelmän lopullisena tavoitteena on, että jokaisella työntekijällä olisi NFC-tunniste ja että jokaisessa ravintolassa olisi käytössä lukulaite, jolla työntekijä tunnistetaan. Täten päästäisiin kokonaan eroon paperisista listoista, käytännöt koko yrityksen sisällä yhtenäistyisivät ja kaikki lounasdata olisi saatavissa yhdestä paikasta samassa muodossa. On kuitenkin selvää, että kaikki ravintolat eivät tule ainakaan alkuvaiheessa ottamaan sovellusta käyttöön. Näin ollen osana projektia Wapicen käyttämään tuntikirjausjärjestelmään integroidaan työkalu, jolla lounasdataa pystyy lukemaan PDF-tiedostoista ja lounaslippujärjestelmän tietokannasta, ja joka luo luetun datan perusteella Excel-tiedoston. Excel-tiedosto voidaan sellaisenaan syöttää palkanmaksuohjelmaan. Aiemmin Excel-tiedosto on luotu manuaalisesti. Lisäksi lounaslippujärjestelmään luodaan työpöytäasiakassovellus, joka tulee korvaamaan paperisen kuittauslistan niissä Wapicen toimistoissa, joissa on käytössä lounasliput.

2.2 Kilpailevat järjestelmät ja teknologiat

Erilaisia lounaslippujärjestelmiä on jo olemassa. Osa olemassa olevista järjestelmistä on toteutettu paperisten lounaslippujen avulla, mutta myös sähköisiä järjestelmiä on kehi-

tetty. Suunniteltu lounaslippujärjestelmä helpottaa lounaiden ostamista työntekijän näkökulmasta, koska työntekijän tarvitsee vain tunnistautua mukana kulkevalla NFC-tunnisteella lounaan ostamisen yhteydessä. Paperiset lounasliput annetaan ravintolatyöntekijälle, joten lounaan ostavan työntekijän täytyy muistaa ottaa niitä mukaan ennen lounaan ostamista. Tästä päästään eroon NFC-tunnistautumisen avulla. Lisäksi vältytään ylimääräiseltä työvaiheelta, jossa paperisesta kuittauslistasta lounasmäärät kirjataan tietokoneelle sähköiseen muotoon.

Edenred Finland OY on suomalainen työsuhde-eturatkaisuja tarjoava yritys. Lounasedun lisäksi Edenredillä on ratkaisuja myös työmatka-, liikunta-, ja kulttuuriedun tarjoamiseksi työntekijöille. Lounasetu voidaan toteuttaa perinteisillä paperisilla lounasseteleillä tai ladattavalla Ticket Lounas -maksukortilla. Paperisten lounasseteleiden nimellisarvot ovat 8,20 €:n ja 10,30 €:n välillä. Vaikka lounaan hinta olisi pienempi kuin setelin nimellisarvo, lounassetelistä ei saa takaisin vaihtorahaa. Lounaskorttia käytetään kuten mitä tahansa muutakin maksukorttia, ja siltä veloitetaan aina lounaan todellinen hinta, kuitenkin verohallinnon määrittelemien enimmäis- ja vähimmäisveloitusten rajoissa. Työnantaja lataa käytettävissä olevan saldon kortille etukäteen. Kortin rinnalla toimii myös mahdollisuus mobiilimaksamiseen. Mobiilimaksujen suorittamiseksi älypuhelimeen ladataan MyEdenred-sovellus, jossa voi maksamisen lisäksi tarkistaa kortin saldon ja etsiä lounasravintoloita. Mobiilimaksaminen toimii siten, että työntekijä valitsee ravintolan ja maksettavan summan sovelluksella, näyttää maksukuitin puhelimen näytöltä kassatyöntekijälle ja kassatyöntekijän valvoessa merkitsee kuitin käytetyksi. Edenredin lounasseteleitä ja -kortteja hyväksyy maksuvälineeksi Suomessa yli 13000 ravintolaa. (Edenred 2017)

Toinen suomalainen työsuhde-etuja tarjoava yritys on Smartum OY. Myös Smartumin ratkaisu mahdollistaa lounasedun lisäksi muidenkin työsuhde-etujen tarjoamiseksi työntekijöille. Vaihtoehtoina lounasedun maksutavoiksi ovat lounassetelit ja lounassaldo. Lounassaldoa käytetään maksukortin avulla. Smartum tarjoaa myös mobiiliratkaisun saldon käyttämiseksi. Smartumin mobiilimaksaminen toimii siten, että työntekijä kirjautuu Smartum-tililleen joko mobiilisovelluksella tai Smartumin verkkopalvelussa, luo maksusta QR-koodin (kuva, joka toimii kaksiulotteisen viivakoodin tavoin), ja näyttää QR-koodia kassatyöntekijälle, joka lukee sen lukulaitteella. Erona Edenredin tarjoamaan järjestelmään on se, että Smartumin järjestelmässä maksu kirjataan ravintolan kassapäätteellä, kun taas Edenredin ratkaisussa maksu kirjataan suoraan asiakkaan puhelimesta. (Smartum 2017)

Edellä mainittujen sovellusten lisäksi työsuhde-etujen tarjoamiseksi on olemassa puhtaasti sähköisiä palveluita. Yksi tällainen sovellus on Eazybreak. Eazybreakin avulla työnantaja voi tarjota työntekijöilleen lounas, liikunta-, kulttuuri- sekä työmatkaseteleitä. Eazybreak sisältää mobiilisovelluksen, jota työntekijä käyttää lounasseteleiden tilaamiseen. Palveluntarjoajat ja yritykset pystyvät hallinnoimaan tilejään Eazybreakin internetsivun kautta. Internetsivulla yritys voi määritellä mm. työntekijöille kuukausit-

tain jakautuvien lounasseteleiden määrän ja arvon. Lounasseteleiden jako tapahtuu automaattisesti kuukauden alussa. (Eazybreak 2017)

Eazybreak sisältää kaksi erilaista tapaa käyttää lounasseteleitä: lounasseteli ja sopimusruokailu. Lounassetelivaihtoehdossa yritys ostaa lounasseteleitä ennen kuin niitä jaetaan työntekijöille. Sopimuslounaat puolestaan laskutetaan jälkikäteen sen perusteella, montako lounasta yrityksen työntekijät ovat käyneet syömässä. Sopimusruokailumalli vastaa siis toteutettavassa lounaslippujärjestelmässä käytettävää mallia. Eazybreak hoitaa laskutuksen automaattisesti. Ravintoloilta Eazybreak perii pienen provision jokaisesta järjestelmän kautta myydystä lounaasta. (Eazybreak 2017)

Maksaminen Eazybreakin avulla tapahtuu siten, että ennen lounaan ostamista työntekijä tilaa lounassetelin puhelimeensa joko mobiilisovelluksella tai tekstiviestillä. Jokaisella lounassetelillä on yksilöllinen tunniste. Lounaan ostamisen yhteydessä työntekijä näyttää lounassetelin puhelimen näytöltä kassatyöntekijälle ja työntekijä tarkastaa lounassetelin tunnisteen sekä työntekijän nimen. Tilattu seteli on voimassa vain tilauspäivän. (Eazybreak 2017)

Eazybreakin etuna toteutettavaan lounaslippujärjestelmään on se, että Eazybreakia käyttääkseen yritysten, työntekijöiden tai ravintoloiden ei tarvitse hankkia ylimääräisiä laitteita. Lounaslippujärjestelmässä työntekijöille olisi annettava NFC-tunnisteet, ja lisäksi sekä yritys että ravintola tarvitsisivat älypuhelimen tunnisteiden lukemista varten. Vaikka Wapice tarjoaisi puhelimet, puhelin vie silti tilaa ravintolan kassatiskiltä ja vaatii huomiota kassatyöntekijältä. Lounaslippujärjestelmän viemistä markkinoille hankaloittaa myös se, että Eazybreakilla on jo runsaasti käyttäjiä. Yritykset käyttävät mieluummin järjestelmää, joka mahdollistaa lounaslippujen käyttämisen useissa ravintoloissa, ja vastaavasti ravintolat hyötyvät siitä, jos järjestelmässä on jo paljon yrityksiä eli potentiaalisia asiakkaita. Lounaslippujärjestelmässä puolestaan maksaminen on joustavampaa: Työntekijän ei tarvitse tilata lounaslippuja etukäteen, vaan lounaat kirjautuvat työntekijälle ostotapahtumien yhteydessä.

Eräs kilpailija suunnitellulle lounaslippujärjestelmälle on lähimaksaminen. Lähimaksamisen avulla työntekijä pystyy maksamaan lounaan viemällä maksukortin lähelle maksupäätettä. Tämä on käyttäjälle aivan yhtä vaivaton maksutapa kuin lounaslippujärjestelmän NFC-tunnistautuminen. Lounaslippujärjestelmän avulla saavutetaan kuitenkin jo luvussa 2.1 mainittu veroetu, joten lähimaksamista realistisempia kilpailijoita ovat muut työsuhde-eturatkaisujen tarjoajat.

On myös olemassa lähimaksukortteja, joille voidaan ladata rahaa tai lounaskertoja. Esimerkiksi monissa oppilaitoksissa käytettävät opiskelijakortit toimivat tällä tavalla, ja kyseistä maksutapaa tukevia järjestelmiä onkin ravintoloissa laajasti käytössä. Olemassa olevat järjestelmät ovat kuitenkin ravintolakohtaisia. Myöskään ladattavilla maksukorteilla ei saavuteta veroetua, elleivät ne täytä verohallinnon kohdennetulle maksamiselle

asettamia vaatimuksia. Suunnitellun lounaslippujärjestelmän etuna on, että samalla tunnisteella pystyy tunnistautumaan kaikissa lounaslippujärjestelmää käyttävissä ravintoloissa. Järjestelmä on myös ladattavia kortteja parempi käytettävyyden näkökulmasta, koska lounaita ei tarvitse ladata kortille etukäteen, vaan lounaat laskutetaan yritykseltä jälkeenpäin todellisten ostojen mukaisesti.

3. VAATIMUKSET

Tässä luvussa esitellään järjestelmän keskeisin toiminnallisuus ja sille asetetut vaatimukset. Vaatimukset ovat peräisin useista eri lähteistä. Projektin lähtökohtina olivat yrityksen toimitusjohtajan visio NFC-teknologiaa hyödyntävästä lounaslippujärjestelmästä ja palkkatoimiston tarve vähentää käsin tehtävää työtä. Näin ollen alussa ainoat vaatimukset olivat, että järjestelmän pitäisi käyttää NFC-teknologiaa ja että sen pitäisi helpottaa lounaiden huomioimista palkoissa Wapicen sisällä. Wapicen sisäisten vaatimusten perusteella määriteltiin, miten järjestelmän tulisi toimia, jotta vaatimukset täyttyisivät. Myöhemmin vaatimuksia tarkennettiin palkanmaksuun osallistuvien ihmisten kanssa. Kun järjestelmästä oli toteutettu toimiva prototyyppi, siitä alettiin kehittää myytävää tuotetta. Tuotteistamiseen liittyen vaatimuksia tuli mm. osaston johtajalta ja toimitusjohtajalta.

3.1 Toiminnalliset vaatimukset

Tässä luvussa kuvataan järjestelmän toiminnot. Osa toiminnallisuudesta on yhteistä koko järjestelmälle, ja tällainen toiminnallisuus kuvataan luvussa 3.1.1. Myöhemmissä aliluvuissa kuvataan tiettyihin järjestelmän osiin liittyvät vaatimukset.

3.1.1 Koko järjestelmä

Lounaslippujärjestelmässä on kolme eri käyttäjäroolia: järjestelmänvalvoja (admin), ravintola (restaurant) ja yritys (company). Järjestelmänvalvojan tehtäviin kuuluu uusien ravintoloiden ja yritysten lisääminen järjestelmään, ravintoloiden ja yritysten poistaminen järjestelmästä, käyttäjien tietojen muokkaaminen ja käyttäjien salasanojen vaihtaminen. Ravintola-käyttäjärooli edustaa järjestelmässä ravintolaa, jossa lounaslippujärjestelmä on käytössä. Vastaavasti yritys-käyttäjärooli edustaa yritystä, jonka työntekijät voivat käydä ravintoloissa syömässä käyttäen lounaslippujärjestelmän NFC-tunnisteita. Jokaisessa järjestelmän käyttöliittymässä käyttäjä kirjautuu sisään jotakin edellä mainittua roolia edustavilla tunnuksilla. Käyttäjälle näytettävä näkymä määräytyy käyttäjäroolin perusteella.

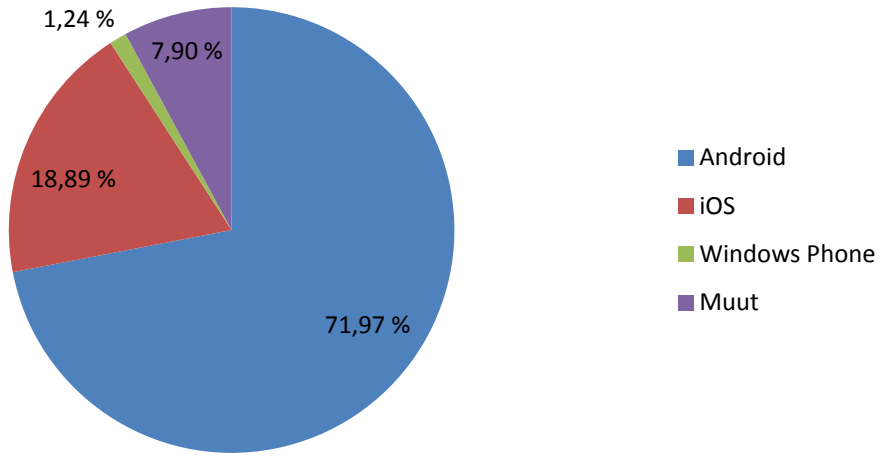
Yritykset pystyvät rekisteröimään NFC-tunnisteita työntekijöilleen. NFC-tunnisteiden rekisteröinti määritellään tarkemmin luvuissa 3.1.2 ja 3.1.3. Vain rekisteröidyt tunnisteet hyväksytään maksuvälineiksi ravintoloissa.

Yritykset ja ravintolat pystyvät solmimaan keskenään sopimuksen. Vain ne työntekijät, joiden yrityksellä on voimassa oleva sopimus tietyn ravintolan kanssa voivat käyttää

lounaslippujärjestelmän NFC-tunnisteita maksuvälineinä kyseisessä ravintolassa. Sopimus ei tässä yhteydessä tarkoita laillisesti sitovaa sopimusta, vaan se on vain järjestelmän tapa pitää kirjaa siitä, minkä yritysten tunnisteet kelpaavat missäkin ravintoloissa.

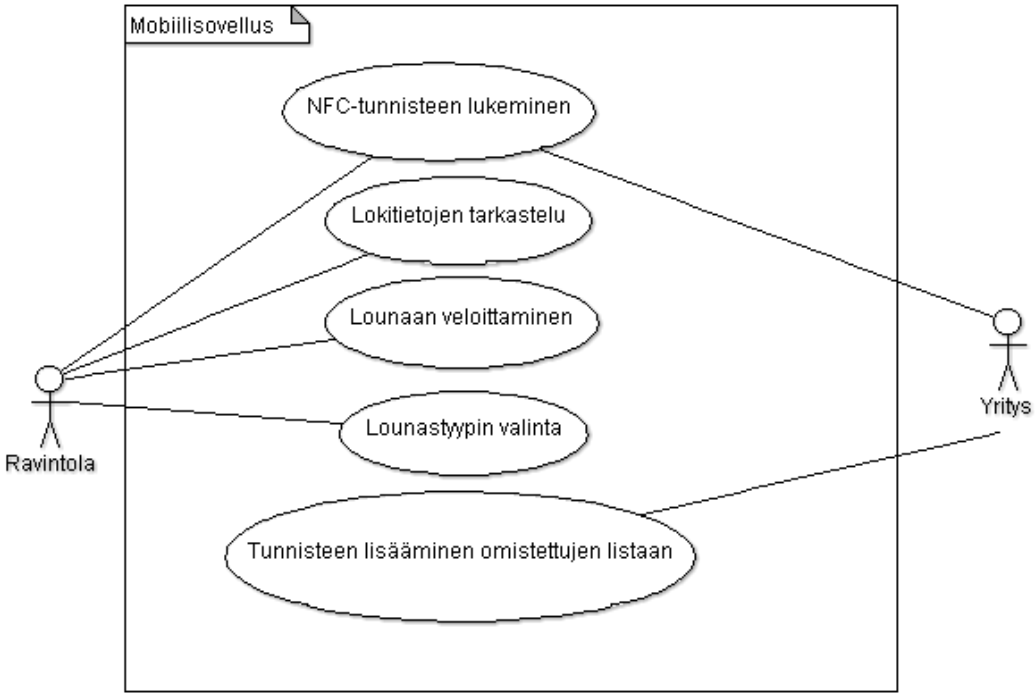
3.1.2 Mobiilisovellus

Mobiilisovellusta käyttävät sekä ravintolat että yritykset. Wapice toimittaa puhelimet ravintoloihin ja yrityksiin, joten riittää, että tuetaan yhtä ennalta määritettyä mobiilialustaa. Alustaksi on valittu Android, koska kuten kuvassa 1 on esitetty, Android on markkinoiden yleisin mobiilikäyttöjärjestelmä (Statista 2016). Android-laitteissa on useita tähän käyttötarkoitukseen soveltuvia vaihtoehtoja, ja jopa joissakin edullisissa malleissa on sovelluksen kannalta välttämätön NFC-toiminnallisuus. Myös alustariippumattomia ratkaisuja (esim. React native) on mietitty, mutta yrityksen sisällä on jo aiemmin tehty Android-sovellus NFC-tunnisteiden lukemiseksi puhelimella, joten osaa olemassa olevasta ohjelmakoodista pystyy käyttämään uudelleen.



Kuva 1. *Mobiilikäyttöjärjestelmien markkinaosuudet, perustuu lähteeseen (Statista 2016)*

Kuvassa 2 on esitetty mobiilisovelluksen käyttötapaukset. Ravintolan kannalta sovelluksen tärkein toiminto on lounaiden veloittaminen. Kun työntekijä ostaa lounaan, hän tunnistautuu ravintolassa olevalle puhelimelle omalla NFC-tunnisteellaan. Sovelluksen tulee tarkistaa, että tunnisteen omistavalla yrityksellä on sopimus kyseisen ravintolan kanssa. Tässä vaiheessa pitää myös pystyä valitsemaan, minkä tyyppinen lounas on ostettu. Lounastyypit mahdollistavat sen, että esimerkiksi erihintaisia lounaita voitaisiin käsitellä järjestelmässä eri tavoilla. Mikäli ravintola on määritellyt vain yhden lounastyypin, järjestelmä valitsee lounastyypin automaattisesti.



Kuva 2. *Mobiilisovelluksen käyttötapaukset*

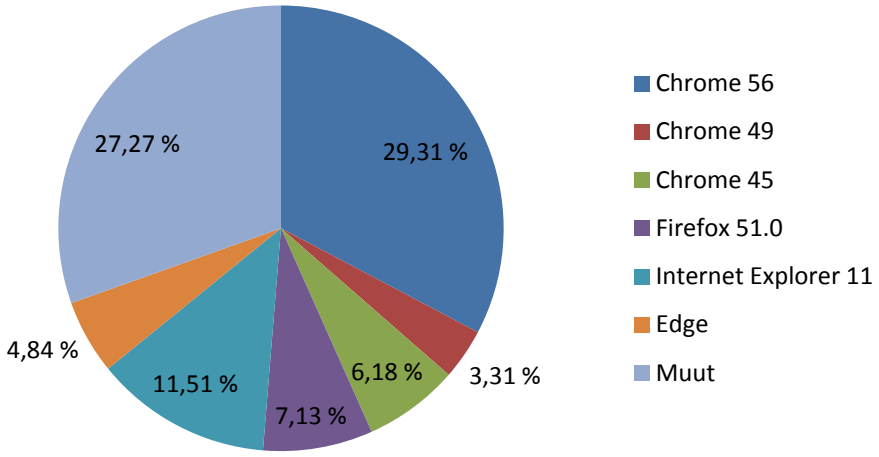
Sovelluksessa tulisi olla mahdollisuus perua lounaskuittaus siltä varalta, että tunniste luetaan kahteen kertaan. Virheen mahdollisuuden minimoimiseksi kahden hyväksytyn lukutapahtuman välissä on myös oltava ennalta määritelty viive. Viiveeksi on valittu viisi sekuntia, koska siinä ajassa asiakas on ehtinyt jo ottaa tunnisteensa pois lukijasta mutta aika on vielä tarpeeksi lyhyt, ettei se häiritse järjestelmän sujuvaa käyttöä.

Sovelluksessa on myös loki, johon kirjoitetaan luetut tunnisteet, onko tunniste hyväksytty vai ei, veloitetut lounaat sekä mahdolliset virhetilanteet. Loki on erityisen tärkeä siinä tapauksessa, että lounastyypin valinta tapahtuu automaattisesti eikä ravintolatyöntekijä osallistu aktiivisesti lounaan kuittaamiseen. Tällöin voi olla tarpeellista tarkistaa, että lounas kirjattiin järjestelmään ja ettei sitä kirjattu useaan kertaan. Lokia ei käytetä tiedon pitkäaikaiseen tallennukseen, koska ravintola pystyy tarkistamaan veloitetut lounaat web-käyttöliittymässä. Näin ollen loki tyhjennetään ravintolan kirjautuessa ulos sovelluksesta. Loki nopeuttaa ravintolatyöntekijän työtä, koska epäselvissä tilanteissa ei tarvitse avata erillistä käyttöliittymää, josta lounaan kirjautuminen voitaisiin tarkistaa.

Yritys käyttää samaa sovellusta kuin ravintola, mutta yrityksen näkymä sovellukseen on erilainen. Yritys käyttää mobiilisovellusta ainoastaan NFC-tunnisteiden rekisteröintiin. Järjestelmässä on lista yrityksen omistamista tunnisteista, johon mobiilisovelluksella luetut tunnisteet lisätään. Tästä listasta yritys pystyy rekisteröimään tunnisteita työntekijöille web-käyttöliittymässä. Kaksiosaisen rekisteröintiprosessin ansiosta kaikki NFC-tunnisteen lukemista vaativa toiminnallisuus voidaan sisällyttää mobiilisovellukseen, eikä yrityksen tarvitse hankkia erillistä kortinlukijaa tunnisteiden rekisteröintiä varten.

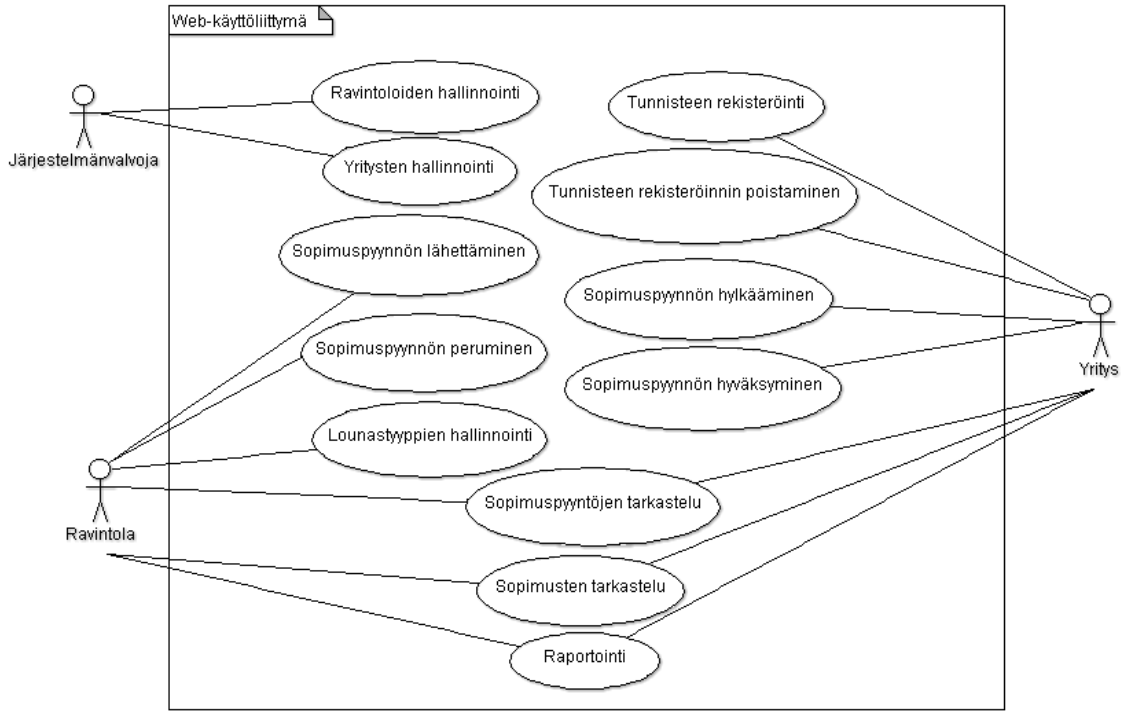
3.1.3 Web-käyttöliittymä

Web-käyttöliittymää käyttävät kaikki käyttäjäroolit. Järjestelmänvalvojalle web-käyttöliittymä on ainoa näkymä järjestelmään. Sen kautta ravintoloita ja yrityksiä pystyy lisäämään, poistamaan ja muokkaamaan. Web-käyttöliittymän tulee olla käytettävissä yleisimmillä nykyaikaisilla selaimilla. Vuoden 2016 joulukuussa kaytetyimmät selaimet olivat Google Chrome, Internet Explorer, Mozilla Firefox ja Microsoft Edge kuvassa 3 esitetyillä osuuksilla (Browser market share 2016). Tuetuiksi versioiksi on valittu Google Chrome 50, Internet Explorer 9, Mozilla Firefox 48, Microsoft Edge 21 sekä uudemmat versiot mainituista selaimista.



Kuva 3. *Internet-selainten markkinaosuudet, perustuu lähteeseen (Browser market share 2016)*

Kuvassa 4 on esitetty web-käyttöliittymän käyttötapaukset. Web-käyttöliittymää käytetään sopimusten kirjaamiseen järjestelmään. Ravintola pystyy lähettämään käyttöliittymän kautta yritykselle sopimuspyynnön, johon on sisällytettynä sopimuksen tiedot, esimerkiksi alkamis- ja päättymispäivämäärä. Lisäksi ravintola pystyy tarkastelemaan jo aiemmin solmimiaan sopimuksia ja lähettämiään sopimuspyyntöjä sekä perumaan lähetetyn sopimuspyynnön, jota ei ole vielä kuitattu.



Kuva 4. *Web-käyttöliittymän käyttötapaukset*

Ravintola käyttää web-käyttöliittymää myös lounastyyppien hallinnointiin. Ravintola voi määritellä uuden lounastyypin, muokata olemassa olevaa lounastyyppiä tai poistaa lounastyypin. Lisäksi ravintola voi tarkastella raportteja web-käyttöliittymässä. Raporttien avulla ravintola pystyy laskuttamaan yrityksiltä oikean määrän lounaita. Raportit ovat ladattavissa myös Excel-muodossa.

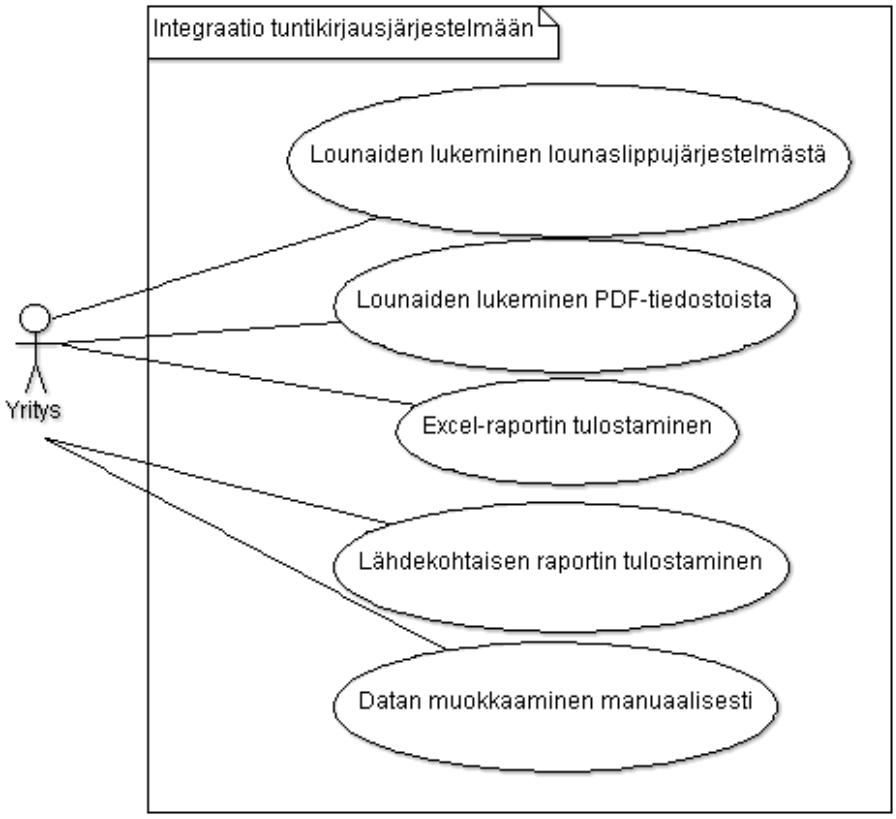
Myös yritys hallinnoi sopimuksia web-käyttöliittymän kautta. Yritys pystyy tarkastelemaan olemassa olevia sopimuksia sekä ravintoloilta saapuneita sopimuspyyntöjä. Yritys voi hyväksyä tai hylätä sopimuspyynnön. Alhaisen prioriteetin vaatimuksena on, että yritys saa kirjautuessaan ilmoituksen, mikäli uusia sopimuspyyntöjä on saapunut edellisen kirjautumisen jälkeen. Sopimuksen hyväksymisen jälkeen yrityksen tunnisteilla pystyy maksamaan sopimuspyynnön lähettäneessä ravintolassa.

Yritys käyttää web-käyttöliittymää myös NFC-tunnisteiden rekisteröintiin työntekijöilleen. Rekisteröintinäkymässä tunniste voidaan valita yrityksen omistamien vapaiden tunnisteiden listasta, johon tunnisteita lisätään mobiilisovelluksella. Tunnisteen lisäksi työntekijälle määritellään nimi ja työntekijän id, joka on mikä tahansa yrityksen sisällä työntekijäkohtaisesti yksilöllinen merkkijono. Tunnisteen rekisteröinti voidaan poistaa samassa käyttöliittymässä.

Myös yritys voi tarkastella raportteja web-käyttöliittymässä. Yritys voi käyttää raportteja oman taloutensa seuraamiseen. Raporttien avulla voidaan myös veloittaa oikea määrä lounaita työntekijöiden palkoista.

3.1.4 Integraatio tuntikirjausjärjestelmään

Yksi lounaslippujärjestelmän keskeisistä tavoitteista on manuaalisen työn vähentäminen Wapicen sisäisessä prosessissa, jossa ostettuja lounaita vastaava summa vähennetään työntekijöiden palkoista. Näin ollen osaksi lounaslippujärjestelmää sisältyy integraatio Wapicen tuntikirjausjärjestelmään. Tuntikirjausjärjestelmässä pitäisi olla näkymä, jossa lounasdataa pystyy lukemaan eri lähteistä, esimerkiksi PDF-tiedostoista ja lounaslippujärjestelmän tietokannasta. Näkymän toiminnot on esitetty kuvassa 5.



Kuva 5. *Tuntikirjausjärjestelmäintegraation käyttötapaukset*

Tuntikirjausjärjestelmän lounasnäkymään voidaan ladata ravintoloiden lähettämiä PDF-muotoisia listoja, jotka sovellus jäsentää. Koska eri ravintoloiden käyttämät formaatit poikkeavat toisistaan, näkymässä täytyy pystyä määrittelemään kunkin tiedoston jäsentämiseen käytettävä formaatti. Lisäominaisuutena järjestelmä voisi tunnistaa käytettävän formaatin automaattisesti tiedoston rakenteesta. Eri ravintoloilla voi kuitenkin olla rakenteeltaan samankaltaisia PDF-tiedostoja, ja ravintolan käyttämä formaatti voi muuttua, joten automaattisen tunnistamisen toteuttaminen olisi haastavaa. Järjestelmän ensimmäisessä versiossa riittää, että sovellus pystyy päättelemään PDF-tiedoston rakenteen tiedoston nimen alussa olevasta etuliitteestä.

Lounasnäkymässä lounasdataa tulee pystyä lukemaan myös lounaslippujärjestelmän tietokannasta. Näkymässä näytetään luetut lounasmäärät sekä muuta dataa, esimerkiksi

työntekijöiden nimet, joka on luettu tuntikirjausjärjestelmän tietokannasta. Dataa ja lounasmääriä pystyy muokkaamaan myös manuaalisesti, mikä on tärkeää, koska joiltakin ravintoloilta tulee kuvamuotoisia PDF-tiedostoja, joiden jäsentäminen ohjelmallisesti vaatisi konenäköä ja olisi vaikea saada toimimaan luotettavasti. Projektin alkuvaiheessa tekstin tunnistamista kuvasta kokeiltiin Tesseract OCR (optical character recognition) -kirjaston avulla. Kirjasto pystyi tunnistamaan merkkijonoja tiedostoista, mutta esimerkiksi työntekijän nimestä jäi joissakin tapauksessa puuttumaan merkkejä.

Järjestelmä pystyy muodostamaan luetun lounasdatan perusteella Excel-raportin, joka voidaan sellaisenaan syöttää palkanmaksujärjestelmään. Lisäksi järjestelmästä on saatavissa raportti, joka kertoo, kuinka monta lounasta kullekin työntekijälle on laskettu, ja mistä lähteestä lounasdata on peräisin. Tämä tieto tallennetaan myös lokiin tuntikirjausjärjestelmän tietokantaan.

Tuntikirjausjärjestelmäintegraatio kommunikoi lounaslippujärjestelmän kanssa REST (Representational state transfer) -rajapinnan kautta. Tuntikirjausjärjestelmä käyttää vain yhtä rajapinnan funktiota, jolla se hakee listan Wapicen työntekijöiden ostamista lounaista tietyllä aikavälillä. Rajapinnan toteutuksesta kerrotaan tarkemmin luvussa 5.1. REST-rajapinnan avulla lounaslippujärjestelmä olisi mahdollista integroida myös muiden yritysten käyttämiin järjestelmiin.

3.1.5 Käyttöskenaariot

Lounaslippujärjestelmä on mahdollista ottaa käyttöön kokonaan tai osittain. Osittainen käyttöönotto voi olla tarpeellista esimerkiksi siinä tapauksessa, että kaikki ravintolat, joissa yrityksen työntekijät käyvät syömässä, eivät ole ottaneet lounaslippujärjestelmää käyttöön. Tämä on todennäköinen skenaario erityisesti projektin pilotointivaiheessa. Alla on kuvattu, miten järjestelmä voisi toimia erilaisissa tilanteissa.

1. Lounaiden kuittaaminen ravintolassa NFC-tunnisteella

Työntekijä menee ravintolaan ja näyttää tunnistettaan kassalla olevalle puhelimelle (kuva 2: NFC-tunnisteen lukeminen). Ravintolatyöntekijä näkee näytöllä ilmoituksen lounaan kuittaamisen onnistumisesta (kuva 2: lounaan veloittaminen). Tieto ostetusta lounaasta välittyy lounaslippujärjestelmän palvelimelle. Tarvittaessa ravintolatyöntekijä kirjaa lounaan myös ravintolan omaan kirjanpitojärjestelmään.

2. Lounaiden kuittaaminen paperisilla lounaslipuilla

Wapicen työntekijä ottaa lounaslippuja yrityksen toimistossa ja kirjaa ottamansa lounaslippujen määrän työpöytäsovelluksella. Tieto otetuista lounaslipuista kirjautuu lounaslippujärjestelmään ostettuina lounaina. Koska lounaat kirjataan järjestelmään ennen lounaan todellista ostohetkeä, lounaslippujärjestelmä ei voi tietää, missä ravintolassa otetut lounasliput tullaan käyttämään. Näin ollen yritykselle on luotu järjestelmään oma

ravintola, jolle otetut lounasliput kirjataan. Otettuaan haluamansa määrän lounaslippuja työntekijä menee ravintolaan ja antaa lounaslipun ravintolatyöntekijälle. Ravintolatyöntekijä kirjaa lounaan laskutusta varten ravintolan omaan järjestelmään, joka ei ole yhteydessä lounaslippujärjestelmään.

3. Lounasraportin muodostaminen Wapicella

Työntekijä avaa lounaiden raportointinäkymän tuntikirjausjärjestelmässä. Näkyviin tulee lista yrityksen työntekijöistä. Tässä vaiheessa työntekijöille ei ole merkitty lounaita. Työntekijä valitsee PDF-tiedostojen lukutoiminnon ja raporttiin sisällytettävät PDF-tiedostot (kuva 5: lounaiden lukeminen PDF-tiedostoista). Tämän jälkeen järjestelmä kysyy, millä formaateilla tulkitaan ne tiedostot, joiden formaatteja ei pystytä tunnistamaan automaattisesti tiedostojen nimistä. Työntekijä valitsee formaatit, minkä jälkeen käyttöliittymässä näkyvään työntekijälistaan päivittyvät luetut lounasmäärät. Seuraavaksi työntekijä valitsee lounastietojen lukemisen lounaslippujärjestelmästä ja raporttiin sisällytettävän aikavälin (kuva 5: lounaiden lukeminen lounaslippujärjestelmästä). Valitulla aikavälillä järjestelmään kirjatut lounaat päivittyvät käyttöliittymään. PDF-tiedostoista, jotka eivät ole ohjelmallisesti jäsennettävissä, työntekijä laskee lounasmäärät ja kirjaa ne manuaalisesti. Tälle on järjestelmässä oma toimintonsa (kuva 5: datan muokkaaminen manuaalisesti). Lopuksi työntekijä valitsee raportin muodostamisen, minkä seurauksena järjestelmä luo Excel-tiedoston, jossa näkyvät kokonaislounasmäärät työntekijäkohtaisesti (kuva5: Excel-raportin tulostaminen).

4. Sopimuksen muodostaminen

Ravintolatyöntekijä avaa sopimusnäkymän web-käyttöliittymässä, valitsee uuden sopimuspyynnön lähettämisen, täyttää sopimuksen tiedot sisältäen kohdeyrityksen, ja lähettää pyynnön (kuva 4: sopimuspyynnön lähettäminen). Yrityksen työntekijä näkee ilmoituksen uudesta sopimuspyynnöstä kirjautuessaan lounaslippujärjestelmän web-käyttöliittymään. Työntekijä avaa sopimuspyynnön, näkee sopimuksen tiedot, ja pystyy valitsemaan, hyväksytäänkö sopimuspyyntö (kuva 4: sopimuspyynnön hyväksyminen). Hyväksymisen jälkeen sopimus on voimassa.

3.2 Ei-toiminnalliset vaatimukset

Lounaslippujärjestelmän on tarkoitus yksinkertaistaa lounaiden raportointia. Näin ollen se ei saa aiheuttaa ylimääräisiä työvaiheita kassalla toimivalle ravintolatyöntekijälle. Mobiilisovellus ei siis saa edellyttää toimia ravintolatyöntekijältä lounaiden kuittausvaiheessa, vaan pelkkä tunnisteen näyttäminen puhelimelle riittää. Käytännössä tämä tarkoittaa sitä, että lounastyyppiä ei tarvitse erikseen valita, jos ravintolalla on vain yksi lounastyyppi määriteltynä.

Lounaslippujärjestelmän kaupallisilla osilla, eli mobiilisovelluksella, puhelimella ja web-käyttöliittymällä, tulee olla selkeä brändi. Brändiä voidaan luoda esimerkiksi väreillä. Käyttöliittymien värimaailma voisi olla sinisävytteinen kuten useissa muissakin Wapicen valmistamissa tuotteissa. Puhelin voisi olla oranssi. Puhelimen selkeästi erottuva väri auttaa asiakasyritysten työntekijöitä tunnistamaan, missä ravintoloissa lounaslippujärjestelmä on käytössä. Lounaslippujärjestelmällä tulisi olla myös tunnistettava logo.

Tuntikirjausjärjestelmään toteutettavan näkymän tulisi olla helppokäyttöinen. Näin ollen PDF-tiedostojen lukeminen saa edellyttää työntekijältä korkeintaan seitsemää klikkausta ja lisäksi jokaista PDF-tiedostoa kohti kahta klikkausta. Tämä on arvio pienimmäksi toimintomääräksi, jolla tehtävä saadaan suoritettua. Järjestelmän tulee pystyä lukemaan useita PDF-tiedostoja kerralla. Myös lounasmäärien manuaalisen muokkaamisen täytyy olla mahdollista vähällä määrällä käyttäjältä edellytettäviä toimia. Ideaalitilanteessa useiden käyttäjien lounasmääriä pystyisi muokkaamaan samassa näkymässä.

Järjestelmän täytyy skaalautua, kun uusia ravintoloita ja yrityksiä lisätään järjestelmään. Skaalautuvuus täytyy ottaa huomioon suorituskyvyssä, käyttöliittymässä ja käytettävyydessä. Suorituskykyvaatimus tarkoittaa sitä, että järjestelmän tulee kyetä vastaamaan käyttäjän toimintoihin nopeasti, vaikka samanaikaisten käyttäjien määrä ja tietokannan koko kasvavat. Käyttöliittymäsuunnittelussa on otettava huomioon, että tulevaisuudessa esimerkiksi lista ravintoloista ei mahdu enää sivun kokoiseen taulukkoon kerralla, vaan taulukkoa täytyy pystyä selaamaan sivuittain. Käytettävyyden kannalta skaalautuvuus tarkoittaa sitä, että työmäärä ei kasva esimerkiksi lounaiden raportointivaiheessa. Aiemmassa järjestelmässä uusien ravintoloiden ottaminen mukaan järjestelmään lisäsi käsiteltävien PDF-tiedostojen määrää.

Järjestelmän tulee olla helposti laajennettavissa. Ravintolat ja yritykset voivat tulevaisuudessa haluta erilaisia raportteja, kuin järjestelmässä on valmiiksi tarjottuna. Uusien raporttityyppien lisäämisen pitäisi olla mahdollista ilman kohtuutonta lisätyötä. Lounastyyppeihin voidaan haluta lisätä esimerkiksi hintatieto. Myös sopimuksien sisältämää tietoa on pystyttävä laajentamaan erityisesti siinä tapauksessa, että järjestelmän laskujen lähettäminen järjestelmän kautta tulee mahdolliseksi.

3.3 Tietoturva

Käyttäjiltä vaaditaan autentikointi palvelimelle lähetettävien pyyntöjen yhteydessä. Autentikoinnin yhteydessä tarkistetaan käyttäjän rooli ja estetään pyyntö, mikäli sen suorittamiseen ei käyttäjällä ole oikeutta. Lisäksi tarkistetaan, että kukin käyttäjä pääsee tarkastelemaan vain sille tarkoitettua dataa. Esimerkiksi ravintola ei siis saa tarkastella toisessa ravintolassa syötyjä lounaita. Koska pyyntöjen mukana on lähetettävä joko käyttäjän tunnistetiedot tai sessiotunniste, kaikki asiakkaiden ja palvelimen välinen

kommunikaatio tapahtuu HTTPS-yhteyden välityksellä. Näin ollen kolmas osapuoli ei pysty kaappaamaan sessiota eikä pääse käsiksi siirrettävään dataan.

NFC-tunnisteita ei saa pystyä kopioimaan helposti. Tunnisteelle ei siis voi tallettaa tunnistautumiseen käytettävää dataa, koska data voidaan kopioida toiselle tunnisteelle millä tahansa NFC-kommunikaatioon kykenevällä puhelimella. Myös tunnisteen sarjanumeron käyttäminen työntekijän tunnistamiseen on turvaton vaihtoehto, koska on olemassa tunnisteita, joiden sarjanumeroa voi muuttaa. Järjestelmän ensimmäisessä versiossa tunnistaminen kuitenkin tehdään sarjanumeron perusteella. Jos järjestelmästä tehdään tuotantoversio, tunnistamiseen täytyy käyttää kehittyneempien NFC-tunnisteiden suojattuja alueita.

4. KÄYTETYT TEKNOLOGIAT

Tässä luvussa kerrotaan lounaslippujärjestelmän toteuttamiseen käytetyistä teknologioista. Lounaslippujärjestelmän perustana on NFC-teknologia, joka mahdollistaa käyttäjän tunnistamisen henkilökohtaisen tunnisteen avulla. Järjestelmän palvelin on toteutettu Node.js:llä. Lisäksi järjestelmään liittyy useita eri teknologioilla toteutettuja asiakassovelluksia: Android-puhelimille tarkoitettu mobiilisovellus, React-kirjaston avulla toteutettu web-käyttöliittymä, C#-raportointisovellus ja tuntikirjausjärjestelmään integroitu näkymä.

4.1 NFC ja NDEF

NFC (Near Field Communication) on teknologia, jonka avulla kaksi lähekkäin olevaa laitetta voivat kommunikoida langattomasti. NFC-standardia ylläpitää NFC-forum. NFC perustuu RFID-teknologiaan (Radio Frequency Identification). RFID-teknologia mahdollistaa tunnisteen sarjanumeron lukemisen ilman, että lukijan ja tunnisteen välillä täytyy olla näköyhteyttä toisin kuin esimerkiksi QR-koodeihin perustuvassa tunnistuksessa. RFID-tiedonsiirto voi tapahtua kahdella eri periaatteella. Tiedonsiirtoon käytettävä menetelmä vaikuttaa tiedonsiirron kantamaan. Lähikentässä tapahtuva tiedonsiirto perustuu sähkömagneettiseen induktioon. Lukija luo vaihtelevan sähkövirran, joka synnyttää laitteen ympärille magneettikentän. Magneettikenttä aiheuttaa tunnisteessa sähkövirran, jonka synnyttämän magneettikentän lukija pystyy havaitsemaan. Tunnisteen ID voidaan lukea magneettikentän pienistä muutoksista. Lähikentän ulkopuolella tapahtuva tunnistaminen perustuu sähkömagneettisen säteilyn heijastumiseen. Lukija lähettää suuritaajuista sähkömagneettista säteilyä, joka saa aikaan sähkövirran tunnisteessa. Tunniste heijastaa takaisin osan säteilystä. Muuttamalla tunnisteen impedanssia ajan kuluessa voidaan takaisin heijastuvaan säteilyyn koodata tunnisteen ID. Koska heijasteen voimakkuus heikkenee eksponentiaalisesti etäisyyden kasvaessa, tällä menetelmällä tunnistaminen vaatii lukijassa tarkkaa vastaanotinta, joka pystyy havaitsemaan heikon heijasteen. (Want 2006)

NFC-teknologia on alijoukko lähikentässä tapahtuvan tunnistamisen RFID-teknologioista. NFC-laitteet kommunikoivat 13,56 MHz taajuisilla radioaalloilla, joiden kantama on lyhyt, tyypillisesti alle kymmenen senttimetriä (Curran et al. 2012). NFC:n etuna muihin langattomiin tiedonsiirtoteknologioihin on yhteyden muodostamisen helppous. Toisin kuin esimerkiksi bluetooth- tai WLAN-teknologioissa, käyttäjän ei tarvitse etukäteen konfiguroida yhteyttä laitteiden välille, vaan yhteys muodostuu automaattisesti, kun kaksi laitetta tuodaan toistensa läheisyyteen. Verrattuna WLAN- ja bluetooth-

teknologioihin NFC:n tiedonsiirtokapasiteetti on kuitenkin pieni. NFC on myös mainittuja teknologioita turvattomampi, koska yhteyden muodostamiseksi ei tarvita salasanaa tai käyttäjän vahvistusta.

4.1.1 NFC:n käyttökohteet

Lyhyen kantamansa ansiosta NFC soveltuu hyvin käyttötarkoituksiin, joissa käyttäjän tunnistaminen tapahtuu käyttäjän ollessa kosketusetäisyydellä lukulaitteesta. NFC-teknologiaa käytetäänkin avaimissa, matkakorteissa ja lähimaksukorteissa. Näissä Sovelluksissa käyttäjä tunnistautuu koskettamalla lukukaitetta passiivisella NFC-tunnisteella. Passiivisella tunnisteella ei ole omaa virtalähdettä, vaan se saa virtansa lukulaitteen luomasta sähkömagneettisesta kentästä. Toimintatilat kuvataan tarkemmin luvussa 4.1.2.

Monet nykyaikaiset älypuhelimet kykenevät NFC-kommunikaatioon. NFC-tunnisteita käytetään käynnistämään puhelimen toimintoja. Tunnisteen sisältämä toiminto tunnistetaan tunnisteelle kirjoitetun NDEF (NFC Data Exchange Format) -tietueen tyypin avulla. NFC-tunnisteen lukeminen voi esimerkiksi ohjata puhelimen verkkosivulle, muuttaa puhelimen asetuksia tai lisätä yhteystiedon puhelimen luetteloon. Yksi puhelimen NFC-ominaisuutta hyödyntävä käyttökohde ovat kauppoihin sijoiteltavat NFC-tunnisteet, jotka luettaessa ohjaavat käyttäjän verkkosivulle, jolla on lisätietoa tuotteesta, jonka läheisyyteen tunniste on asetettu.

Puhelinta on mahdollista käyttää myös tunnisteena. Tätä ominaisuutta käyttävät hyväksi erilaiset mobiililompakkosovellukset, jotka mahdollistavat maksukorttien tallentamisen mobiilisovellukseen. Kortin tallentamisen jälkeen puhelinta voidaan käyttää maksuvälineenä lähimaksukorttien tavoin.

NFC-teknologiaa voidaan myös käyttää nopeamman yhteyden konfiguroimiseen. Näin toimii Android Beam. Android Beam on Android-puhelimien ominaisuus, joka mahdollistaa tiedostojen siirtämisen kahden lähekkäin olevan Android-puhelimen välillä. Android Beam muodostaa NFC-yhteyden avulla laitteiden välille Bluetooth-yhteyden, jota käytetään tiedoston siirtämiseen. Myöhemmin myös muita mobiilikäyttöjärjestelmiä käyttäviin puhelimiin on tullut vastaavia toimintoja. Windows-puhelimessa Android beamin tavoin toimii tap and send –toiminto, ja Iphonessa samasta ominaisuudesta käytetään nimeä Airdrop.

4.1.2 Toimintatilat

NFC-standardi määrittelee kaksi toimintatilaa: vertaistila (peer-to-peer) ja lukija-kirjoittaja-tila. Vertaistilassa molemmilla tiedonsiirtoon osallistuvilla laitteilla on oma virtalähde ja kumpikin muodostaa oman sähkömagneettisen kentän. Laitteet toimivat

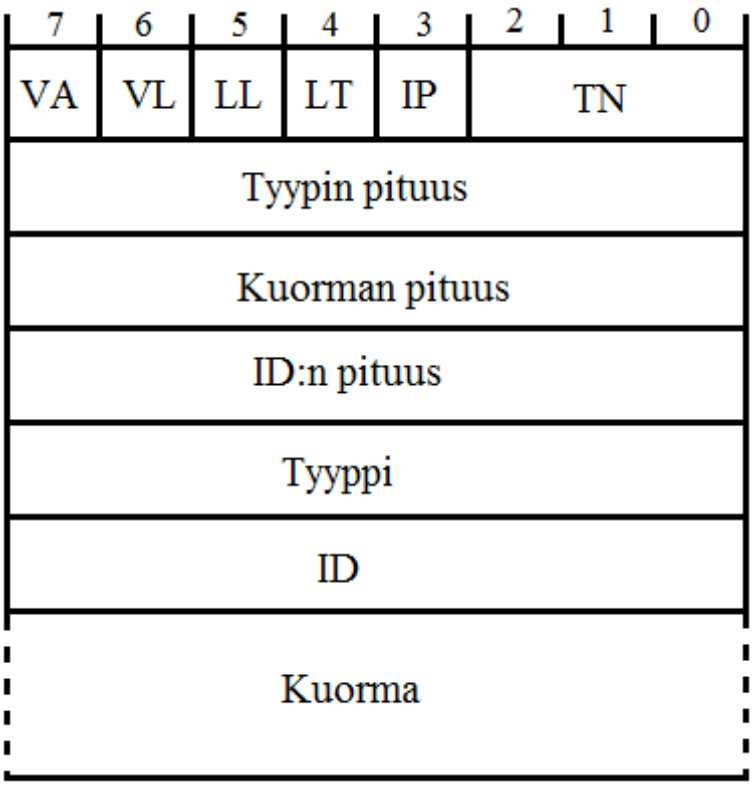
vuorotellen isäntänä. Tätä toimintatilaa käytetään esimerkiksi kahden puhelimen väliseen tiedonsiirtoon.

Lukija-kirjoittaja-tilassa vain toinen laitteista muodostaa sähkömagneettisen kentän. Tässä tapauksessa tunnisteella ei tyypillisesti ole omaa virtalähdettä, vaan se saa tarvitsemansa virran isäntälaitteen magneettikentästä. Henkilökortit ja NFC-tarrat ovat passiivisia. Tätä toimintatilaa käytetään myös tämän diplomityön aiheena olevassa lounaslippujärjestelmässä. Lounaslippujärjestelmässä aktiivisena laitteena toimii puhelin ja tunnisteina käytetään passiivisia NFC-tarroja.

Edellä mainittujen toimintatilojen lisäksi mobiililaite voi emuloida NFC-tunnistetta. Tässä tapauksessa mobiililaite toimii passiivisessa roolissa, ja näin ollen tämä toimintatila on erikoistapaus lukija-kirjoittajatilasta. Tätä tilaa käytetään lähimaksamiseen mobiililompakkosovellusten yhteydessä.

4.1.3 NDEF

NDEF on NFC Forumin ylläpitämä standardi, joka määrittelee, miten NFC-tunnisteelle tallennetaan tietoa. Tiedon tallennustavalla ei ole merkitystä teknologian kannalta, mutta jos tiedon tallennusmuoto on määritelty etukäteen, puhelin pystyy tunnisteen lukemisen yhteydessä tunnistamaan, mitä tunnisteelta luetulla datalla pitäisi tehdä. Standardi määrittelee NDEF-viestin, joka koostuu NDEF-tietueista. NDEF-tietueen rakenne on esitetty kuvassa 6.



Kuva 6. *NDEF-tietueen rakenne*

NDEF-viesti sisältää vain NDEF-tietueita. NDEF-tietue koostuu useista otsikkokentistä sekä talletettavasta datasta. VA (viestin alku) ja VL (viestin loppu) kertovat, onko tietue viestin ensimmäinen tai viimeinen. LL (lohkolippu) on 1, jos tietueen sisältö jatkuu seuraavassa tietueessa. Jos LT (lyhyt tietue) on 1, Kuorman pituus -kentän pituus on yksi tavu, muussa tapauksessa kentän pituus on neljä tavua. IP (ID:n pituus määritelty) ilmoittaa, sisältääkö tietue ID:n pituus -kentän. TN (Tyypin nimi -kenttä) kertoo, millaista dataa tietue sisältää. (Roland et al. 2012, s. 65–66) Ennalta määriteltyjen tietotyyppien perusteella lukulaite, esimerkiksi puhelin, pystyy tulkitsemaan tietueen datan oikein ja käynnistämään sille tarkoitetun toiminnon. Jos tyyppi on esimerkiksi URL, puhelin avaa NFC-tunnisteen sisältämän linkin, mutta jos tyyppi on teksti, puhelin näyttää tietueen sisältämän datan näytöllä. Jos sovellukselle on tarpeellista määritellä oma tyyppi, se voidaan tehdä typpi-kentässä.

4.1.4 Turvallisuus

NFC-standardin mukaan NFC-kommunikaatio tapahtuu radioyhteyden välityksellä eikä yhteyden salaamista ole edellytetty standardissa (Curran et al. 2012, s. 375). Näin ollen kolmannen osapuolen on mahdollista vastaanottaa lähetetty signaali. Monien muiden langattomien teknologioiden tavoin NFC on altis salakuuntelulle. Salakuuntelua vaikeuttaa NFC:n lyhyt kantama. Salakuuntelun haittoja voidaan myös estää salaamalla tunnisteelle talletettava data, jolloin siitä tulee ulkopuoliselle tarkkailijalle merkityksetöntä. Tiedon salaaminen on hyvä vaihtoehto silloin, kun tiedon kirjoittaa ja lukee sama sovel-

lus tai kun käytetty salausavain on lähetettävissä kirjoittavalta laitteelta lukijalaitteelle toista kautta. Usein tiedon pitäisi olla luettavissa millä tahansa lukijalla, missä tapauksessa tiedon salaaminen ei ole kelvollinen ratkaisu.

Kolmas osapuoli voi myös lähettää omaa radiosignaaliaan tarkoituksenaan korruptoida NFC-tiedonsiirrossa siirrettävää dataa. Tällaisen hyökkäyksen tarkoituksena on estää palvelu. Tästä erikoistapaus on radiosignaalin lähettäminen siten, että data muuttuu hyökkääjän haluamalla tavalla, mutta on edelleen vastaanottajan tulkittavissa. Tällainen hyökkäys on kuitenkin edellistä vaikeampi toteuttaa, koska sen onnistuminen riippuu amplitudimodulaation voimakkuudesta (Curran et al. 2012, s. 375).

Jos NFC-laitteet vastaavat toistensa viesteihin hitaasti, hyökkääjä voi onnistua lähettämään oman viestinsä kommunikaatioon osallistuvien osapuolten viestien välissä. Tämä vaatii kuitenkin tarkkaa ajoitusta, koska jos alkuperäisen lähettäjän ja hyökkääjän viestit saapuvat vastaanottavalle laitteelle samaan aikaan, ne korruptoituvat ja tiedonsiirto keskeytyy. (Curran et al. 2012, s. 375)

Langattomat teknologiat ovat myös alttiita man-in-the-middle-hyökkäykselle, jossa kumpikin keskustelun osapuoli luulee keskustelevansa toisen kanssa, mutta todellisuudessa osapuolten välissä toimii hyökkääjä, joka välittää viestit lähettäjältä vastaanottajalle. Tässä tapauksessa hyökkääjä voi muokata dataa haluamallaan tavalla. Tällainen hyökkäys on kuitenkin vaikea toteuttaa NFC-tiedonsiirron lyhyen kantaman ansiosta (Curran et al. 2012, s. 375).

NFC-tunnisteita voidaan lukea ja kirjoittaa millä tahansa NFC-kommunikaatioon kykenevällä laitteella. Näin ollen tunnisteen kopioiminen on helppoa. Jokaisella tunnisteella on valmistajakohtainen yksilöllinen sarjanumero, joka yleensä tallennetaan muistialueelle, jota pystyy vain lukemaan. On kuitenkin mahdollista valmistaa tunniste, jolla on sama sarjanumero kuin jollakin toisella tunnisteella. Lounaslippujärjestelmän näkökulmasta tunnisteen kopioiminen tarkoittaisi sitä, että kolmas osapuoli pystyisi syömään toisen työntekijän laskuun. Lounaslippujärjestelmän ensimmäisessä versiossa työntekijän tunnistaminen tehdään tunnisteen sarjanumeron perusteella. Tällöin ei ole merkitystä, vaikka tunnisteen sisältämän datan pystyisi kopioimaan, koska tunnisteelle ei tallenneta dataa. Sarjanumeroon perustuvassa tunnistamisessa luotetaan siihen, etteivät työntekijät tilaa omia NFC-tunnisteita, vaan käyttävät Wapicen tarjoamia tunnisteita. Tuotantoversiossa tällaista oletusta ei voida kuitenkaan tehdä, vaan järjestelmän tulisi käyttää kehittyneempiä tunnisteita, joissa on suojattuja muistialueita, joille kirjoittaminen vaatii autentikoinnin.

Kun NFC-tunnisteita sijoitetaan julkisille paikoille esimerkiksi lisätiedon tarjoamista varten, ei voida varmistua siitä, että tunniste on alkuperäinen. Näin ollen hyökkääjä voi korvata alkuperäisen tunnisteen sellaisella, joka ohjaa käyttäjän epäilyttävälle verkkosivulle tai lähettää tekstiviestin maksulliseen numeroon. Tämän ehkäisemiseksi NFC-

Forum on kehittänyt signature record -tietuetyypin, joka mahdollistaa NDEF-tietueen digitaalisen allekirjoittamisen (Roland et al. 2011, s. 67).

4.2 Node.js

Node.js on JavaScript-pohjainen palvelinympäristö (Node.js 2017). Node.js:n ensimmäisen version julkaisi Ryan Dahl vuonna 2009 (Built in node 2017). Näin ollen Node.js on teknologiana varsin uusi. Pian julkaisun jälkeen, vuonna 2010, julkaistiin Node.js:n pakkauksenhallintatyökalu NPM (Built in node 2017). NPM:n avulla voi ladata kolmannen osapuolen kirjastoja. Koska JavaScript on yleisesti käytetty kieli ja kirjastojen julkaiseminen on NPM:n ansiosta helppoa, kirjastoja on olemassa Node.js:lle sen nuoresta iästä huolimatta paljon. NPM tallentaa tiedot projektin riippuvuuksista package.json-tiedostoon. Tämä helpottaa Node.js-projektin siirtämistä toiseen ympäristöön, sillä käytettyjä kirjastoja ei tarvitse siirtää tai sisällyttää versionhallintaan, vaan projektin omien kooditiedostojen ja package.json-tiedoston siirtäminen riittää. Kohdeympäristössä NPM:n install-komento asentaa automaattisesti oikeat versiot kaikista package.json-tiedostossa mainituista kirjastoista.

Toisin kuin useimmat nykyaikaiset palvelinympäristöt, Node.js käyttää vain yhtä säiettä. Tästä johtuen Node.js kommunikoi oheislaitteiden kanssa asynkronisten tapahtumien avulla (Tilkov & Vinoski 2010, s. 80). Jos kommunikointi tapahtuisi synkronisesti yksisäikeisessä järjestelmässä, palvelin joutuisi esimerkiksi levylle kirjoittamisen yhteydessä odottamaan, että operaatio on valmis, eikä pystyisi tänä aikana palvelemaan muita asiakassovelluksia. Ohjelmoijalle Node.js:n asykronisuus näkyy siten, että useimmat Node.js:n funktiot ottavat parametrina takaisinkutsufunktion, jota kutsutaan, kun tapahtuu jotain, mistä ohjelmoija on kiinnostunut. Tällainen tapahtuma voi olla esimerkiksi asiakkaalta tuleva HTTP-pyyntö, tiedoston lukemisen valmistuminen tai ohjelman suorituksessa tapahtunut virhe. Alla on esimerkki yksinkertaisen palvelimen toteuttamisesta Node.js:llä:

```
var http = require('http');
var address = 'http://localost';
var port = '8080';

var server = http.createServer(function(req, res){
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello World!\n');
};

server.listen(address, port, function() {
    console.log('listening to port ' + port);
});
```

Ohjelma 1. *Yksinkertainen Node:js palvelin*

Node.js:n käyttämisessä on myös haasteita. Node.js:n keskeisimpiä ongelmia ovat asynkronisuudesta aiheutuvat sisäkkäiset takaisinkutsut. Tästä ilmiöstä käytetään englanninkielisessä kirjallisuudessa nimitystä callback hell. Sisäkkäiset takaisinkutsut heikentävät koodin luettavuutta ja niiden takia on vaikeampi seurata, mitä osaa koodista milloinkin suoritetaan. Koodissa ilmiön tunnistaa syvästä kamparakenteesta (ohjelma 2). Asynkronisuuden hallitsemiseksi on olemassa valmiita kirjastoja, esimerkiksi Async.

```
ioRequest1(function() {
    ioRequest2(function() {
        ioRequest3(function() {
            ioRequest4(function() {
                doSomething();
            });
        });
    });
});
```

Ohjelma 2. *Sisäkkäiset takaisinkutsut*

Takaisinkutsujen vähentämiseksi voi myös käyttää lupauksia (promise). Lupaus on JavaScriptin ominaisuus, jonka avulla funktiosta voidaan palauttaa arvo, joka tulee tulevaisuudessa sisältämään dataa. Alla on koodiesimerkki lupausten käytöstä. Get-funktio tekee pyynnön palvelimelle käyttäen asynkronista asyncGet-funktiota, jonka sisäisellä toteutuksella ei tämän esimerkin kannalta ole merkitystä. Kutsumalla resolve-funktiota lupaus ilmoittaa, että dataa on saatavilla. Reject-funktiota kutsutaan virhetilanteissa. Resolve-funktio annetaan lupaukselle then-metodin parametrina. Reject-funktio voidaan antaa then-funktion toisena parametrina tai catch-metodin parametrina. Lupauksia voi ketjuttaa, minkä vuoksi koodin kamparakenne ei syvene, vaikka peräkkäin suoritettavien asynkronisten tehtävien määrä kasvaisi. Alla olevassa esimerkissä ensin haetaan lista käyttäjistä. Kun lista on saapunut, haetaan lista ensimmäisen käyttäjän kavereista. Jos kummassa pyynnössä tahansa tapahtuu virhe, se käsitellään koodin lopussa.

```
function get (url) {
    return new Promise (function (resolve, reject) {
        asyncGet(url, function (err, data) {
            if (err) {
                reject(err);
            } else {
                resolve(data);
            }
        });
    });
}

get('https://site.com/users').then(function(res){
    console.log('users loaded')
    return get('https://site.com/friends?userid=' + res[0].id);
}).then(function (res) {
    console.log('first user has ' + res.length + ' friends');
}).catch(function(err) {
    console.log('there was an error');
});
```

Ohjelma 3. *Esimerkki JavaScript-lupauksesta*

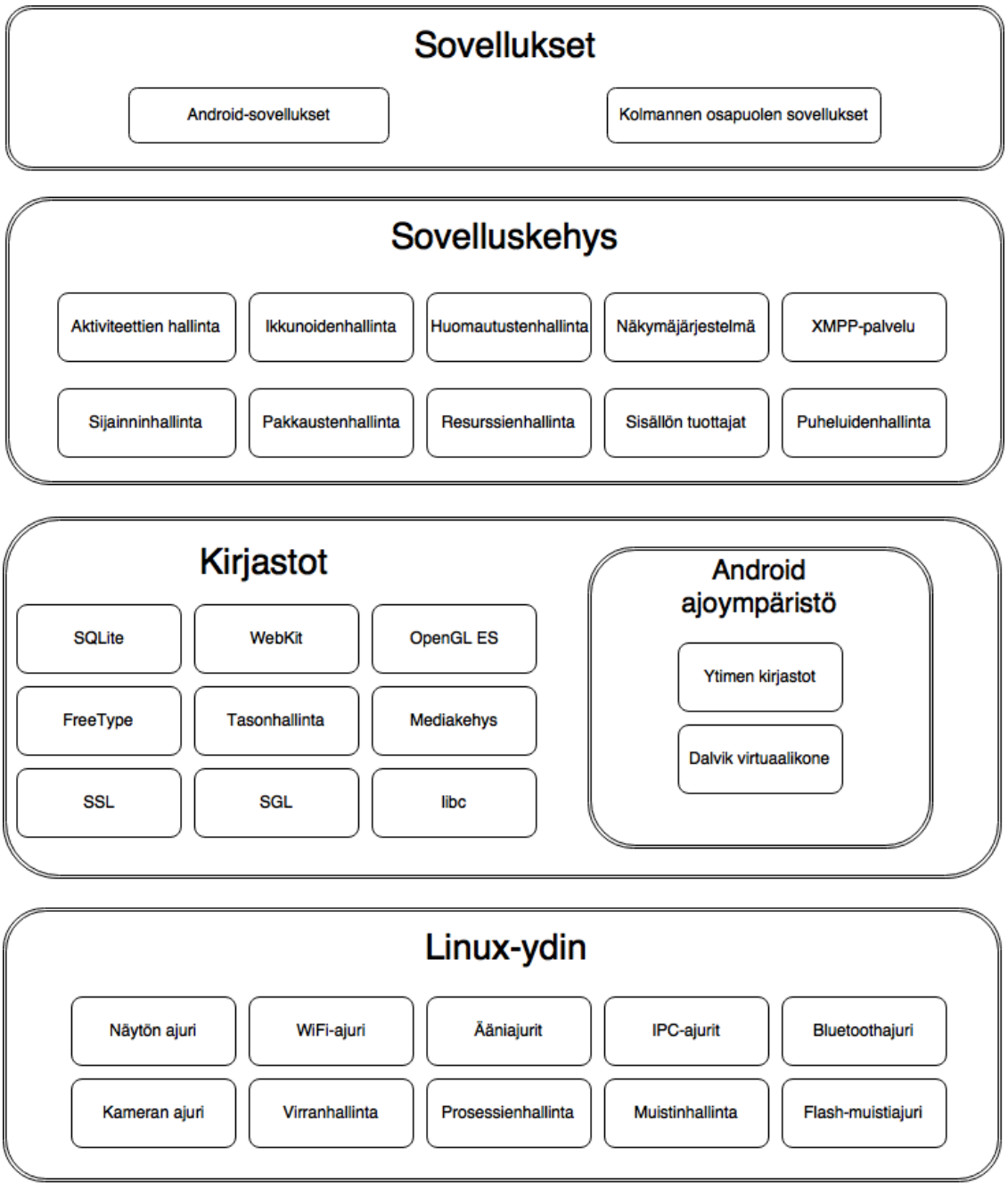
Ensisijainen peruste Node.js:n valitsemiseksi lounaslippujärjestelmän palvelinympäristöksi oli se, että Node.js:stä oli aikaisempaa kokemusta. Node.js-palvelimen pystyttäminen onnistuu nopeasti ja hyvin lyhyellä ohjelmakoodilla, mikä oli tärkeää projektin alkuvaiheessa. Lisäksi JavaScriptin käyttäminen sekä palvelin- että asiakaspuolella mahdollistaa koodin uudelleen käyttämisen. Node.js:ään on myös saatavilla runsaasti avoimen lähdekoodin kirjastoja.

4.3 Android

Android on Android Inc.:n kehittämä mobiilikäyttöjärjestelmä. Vuonna 2005 Androidin kehitys siirtyi Googlelle tämän ostettua Android Inc.:n. Nykyään Androidia ylläpitää Open Handset Alliance, johon kuuluu Googlen lisäksi myös muita tietotekniikka- ja elektroniikka-alan yrityksiä, esimerkiksi LG ja Samsung. (Techradar 2008)

Käyttöjärjestelmän rakenne on esitelty kuvassa 7. Android-käyttöjärjestelmän perustana on Linux-pohjainen käyttöjärjestelmäydin. Ydin tarjoaa alhaisen abstraktiotason palveluita, kuten moniajon sekä muistin ja virrankäytön hallinnan. Android-ajoympäristö toteuttaa Dalvik-virtuaalikoneen, joka suorittaa Java-koodia. Virtualisoinnin ansiosta Android-sovellukset eivät ole riippuvaisia laitteiston toteutuksesta. Virtualisointi myös lisää turvallisuutta, sillä sovellukset toimivat omissa hiekkalaatikoissaan eivätkä täten pääse käsiksi muihin sovelluksiin. (Techotopia 2016) Java-ohjelmointikieleen sisältyvien kirjastojen lisäksi Android sisältää omia kirjastoja, joiden avulla pystyy mm. käyttämään mobiililaitteen oheislaitteita ja antureita. Lounaslippujärjestelmän kannalta keskeisimmät IO-operaatiot (input/output) ovat NFC-tunnisteen lukeminen sekä verkkoliikenne. Android mahdollistaa myös laitteistokohtaisesti käännettyjen C- ja C++-kirjastojen käyttämisen sovelluksissa suorituskyvyn kannalta kriittisten toimintojen to-

teuttamiseksi. Useat Androidin omat kirjastot ovatkin oikeasti C++-kirjastoja, joita käytetään Java-luokan läpi (Techotopia 2016).



Kuva 7. *Android-käyttöjärjestelmän rakenne, perustuu lähteeseen (Techotopia 2016)*

Lounaslippujärjestelmässä Android-puhelimessa toimii sovellus, jolla ravintoloissa voidaan kuitata työntekijän ostama lounas ja jolla asiakasyrityksessä pystytään lukemaan NFC-tunnisteita ja lisäämään niitä järjestelmään kyseiselle yritykselle. Samaa sovellusta käyttävät siis sekä ravintolat että yritykset, ja näytettävä näkymä valitaan sisään kirjautumiseen käytettävän tunnuksen perusteella.

4.4 React

React on Facebookin kehittämä JavaScript-ohjelmistokehys frontend-ohjelmointiin (React 2017). Perinteisesti ohjelmoijan on tarvinnut muokata HTML-dokumenttia (Hypertext Modeling Language) DOM:n (Document Object Model) kautta. Reactin ansiosta ohjelmoijan ei tarvitse käsitellä DOM:a suoraan, mikä yksinkertaistaa sivun suunnittelua ja parantaa tehokkuutta (Hunt et al. 2016). Reactin filosofiana on, että sovelluksen tilan muuttuessa koko HTML-dokumentti päivitetään. Käytännössä React ylläpitää virtuaalista DOM-mallia, johon muutokset tehdään. Muutosten lopuksi todellisesta DOM:sta päivitetään vain ne osat, jotka todella ovat muuttuneet. Tämä takaa tehokkuuden, sillä JavaScriptin suorittaminen on paljon nopeampaa kuin koko DOM:n muokkaaminen, ja React minimoi DOM:iin tehtävät muutokset.

Reactissa sovellus koostuu hierarkkisesti komponenteista. Komponentti voi olla esimerkiksi näkymä, otsikkopalkki tai nappi. Sovelluksen toteuttaminen komponentteina parantaa koodin uudelleenkäytettävyyttä: Jos sovellus tarvitsee useassa paikassa tietyillä kriteereillä suodatettavaa taulukkoa, sellainen kannattaa toteuttaa komponenttina. Myös taulukko voi koostua useista komponenteista. Jokaisen komponentin tulee toteuttaa render-metodi, joka määrittelee, mistä HTML-elementeistä ja muista React-komponenteista komponentti koostuu.

Komponentin ulkoasu voidaan kuvata JavaScriptillä tai JSX-formaatilla. JSX on HTML-kielen laajennus. JSX poikkeaa HTML:stä joidenkin avainsanojen osalta: esimerkiksi HTML:n class-attribuuttia vastaa JSX:ssä className. JSX-formaatti mahdollistaa myös lausekkeiden kirjoittamisen aaltosulkeiden sisään. Alla on esimerkki JSX-määrittelystä. Kun näkymä ladataan käyttöliittymään, React muuntaa JSX:n standardimuotoiseksi HTML:ksi.

```
<div>
  <h2>Title</h2>
  <div className=”container”>
    <Table headers={[”First name”, ”Last name”]} />
  </div>
</div>
```

Ohjelma 4. *Esimerkki JSX-määrittelystä*

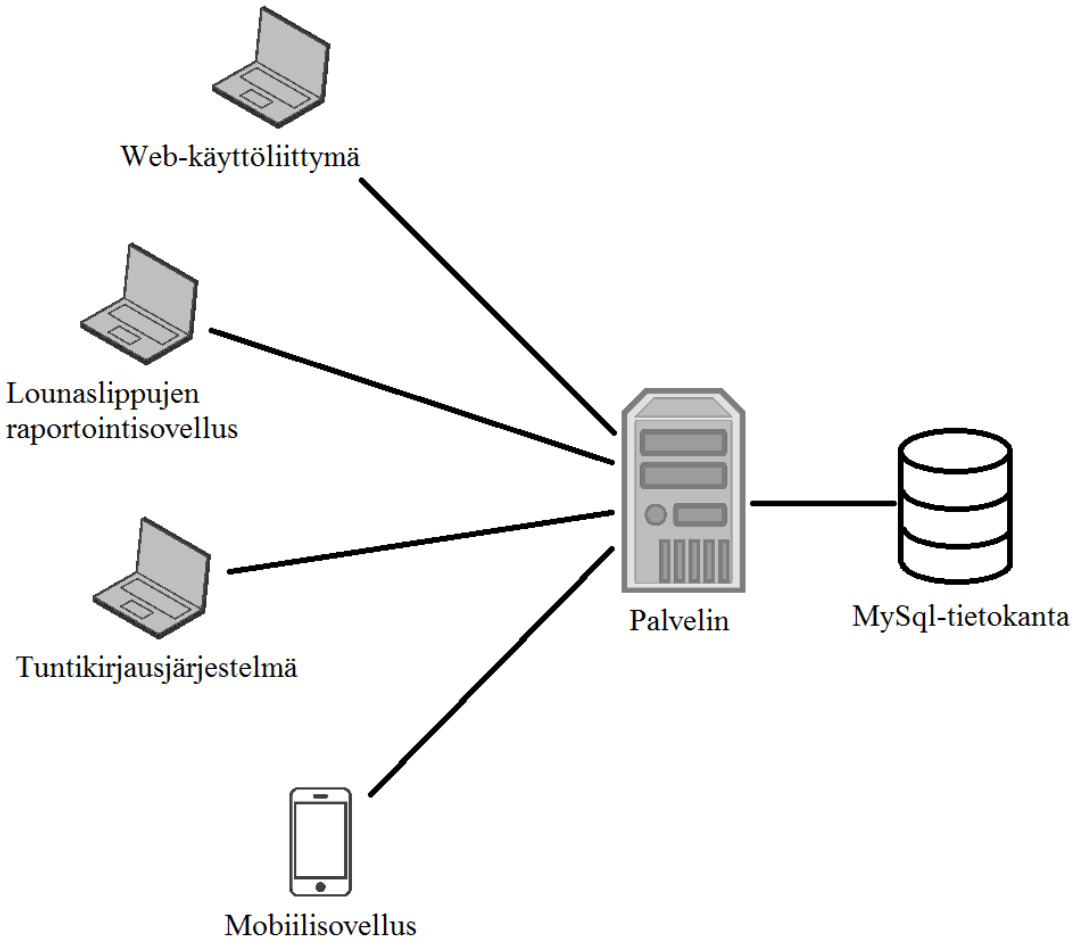
Komponentin luomisen yhteydessä sille voidaan antaa ominaisuuksia (props). Ominaisuuksien avulla samasta komponentista voidaan luoda toisistaan eroavia ilmentymiä. Ominaisuuksien voidaan siis ajatella olevan komponentin parametreja. Yleiskäyttöiselle taulukolle voidaan esimerkiksi kertoa sarakkeiden nimet kuten yllä olevassa koodissa.

Komponentti ei saa muuttaa ominaisuuksiaan. Kuitenkin komponentin täytyy pystyä muuttumaan elinkaarensa aikana. Tätä varten komponentilla voi olla tila (state). Tila eroaa ominaisuuksista siten, että tila on komponentin sisäinen ominaisuus, eikä sitä käsitellä komponentin ulkopuolella. Esimerkiksi taulukkokomponentti voisi käsitellä it-

seensä kohdistuvat hiiren painallukset ja tallentaa tilaan käyttäjän valitseman rivin numeron. Valittu rivi voidaan tässä tapauksessa näyttää käyttöliittymässä erivärisenä kuin muut taulukon rivit. Tilaoliota ei muuteta suoraan, vaan tilan muuttamiseen käytetään setState-funktiota. Jokainen setState-funktion kutsu aiheuttaa tarkistuksen, onko komponentti päivitettävä virtuaaliseen DOM:iin.

5. TOTEUTETTU JÄRJESTELMÄ

Lounaslippujärjestelmään sisältyy palvelin sekä useita erilaisia asiakassovelluksia. Palvelin on yhteydessä tietokantaan. Kuvassa 8 on esitelty järjestelmän osat. Järjestelmässä on aluksi vain yksi palvelin, mutta käyttäjämäärän kasvaessa palvelin voidaan korvata klusterilla skaalautuvuuden takaamiseksi.



Kuva 8. *Lounaslippujärjestelmän osat*

Ravintolat käyttävät mobiilisovellusta lounasdatan lisäämiseen järjestelmään. Lisäksi web-käyttöliittymässä on ravintoloille tarkoitettu näkymä, jonka kautta ne voivat hallinnoida sopimuksia ja lounastyyppejä.

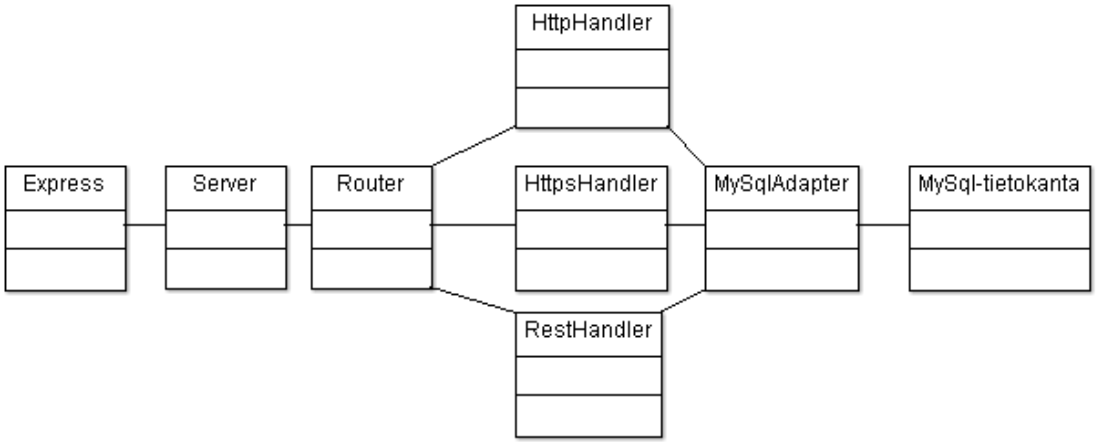
Yritykset lukevat NFC-tunnisteita mobiilisovelluksella. Luetut tunnisteet lisätään yrityksen omistamien tunnisteiden listaan. Web-käyttöliittymän kautta yritykset voivat

hallinnoida sopimuksia ja rekisteröidä omia tunnisteitaan työntekijöilleen. Sekä ravintolat että yritykset voivat ladata raportteja ostetuista lounaista web-käyttöliittymän avulla.

Wapice pystyy käyttämään kaikkia asiakassovelluksia. Wapicella on järjestelmään yritys- ja ravintolatunnukset. Yrityksen roolissa toimiessaan Wapicella on näkymään samat käyttöliittymät kuin muillakin yrityksillä. Lisäksi Wapice pystyy saamaan raportteja Wapicen käyttämään tuntikirjausjärjestelmään lisätyn näkymän kautta. Ravintolan roolissa Wapice ei käytä lounasdatan lisäämiseen mobiilisovellusta, vaan Wapicella on lounaslippujen raportointia varten oma työpöytäsovellus. Työpöytäsovellusta ei esitellä tässä diplomityössä tämän tarkemmin.

5.1 Palvelin

Kaikki lounaslippujärjestelmän asiakassovellukset ovat yhteydessä samaan palvelimeen. Palvelin on toteutettu Node.js-teknologialla. HTTP-pyyntöjen käsittelyyn ja reitittämiseen käytetään Express middlewarea (asiakkaan ja palvelimen välissä toimiva palveluita tarjoava komponentti). Kuvassa 9 on esitelty palvelimen osat.



Kuva 9. *Palvelimen osat*

Server-komponentti käynnistää ja alustaa palvelimen. Reitittimen (router) tehtävänä on ohjata saapuva verkkoliikenne oikealle käsittelijälle. Palvelimella on kaksi rajapintaa: web-rajapinta, jota käyttää web-käyttöliittymä, sekä REST-rajapinta, jota käyttävät muut asiakassovellukset: mobiilisovellus, työpöytäsovellus ja tuntikirjausjärjestelmä.

Web-käyttöliittymä käyttää yksisivuista sovellusta. Web-käyttöliittymän ja palvelimen välinen tiedonsiirto tapahtuu HTTPS-yhteyden kautta lähetettävillä AJAX (Asynchronous JavaScript And XML) -pyynnöillä. Projektin alkuvaiheessa sovelluksen aloitussivu ladattiin HTTP:n kautta ja sivulatausta varten oli oma käsittelijäkomponentti (HttpHandler). Tästä syystä käsittelijäkomponentit on nimetty kuvassa esitetyllä tavalla. Nyt myös sivun lataaminen tehdään HTTPS-yhteyden kautta. Web-käyttöliittymän

käyttäjien autentikointiin käytetään Expressin Session-middlewarea, joka toimii siten, että käyttäjän kirjautuessa lounaslippujärjestelmään käyttäjää varten luodaan istunto. Istunnolle määritellään yksilöllinen tunniste, jonka asiakas sisällyttää kirjautumisen jälkeen kaikkien pyyntöjen evästeisiin. Tunnisteen avulla palvelin tietää, mille käyttäjälle istunto kuuluu.

Myös REST-rajapintaa käytetään HTTPS-yhteyden välityksellä. Koska asiakassovelluksen tilaa ei tallenneta palvelimelle, asiakkaan on jokaisen pyynnön yhteydessä lähetettävä kirjautumiseen vaadittavat tiedot. Autentikointimenetelmäksi valittiin HTTP basic, koska se on yksinkertainen toteuttaa ja HTTPS-yhteyden yli käytettynä riittävän turvallinen. Basic-tunnistautumisessa asiakas asettaa HTTP-pyynnön authorization-otsikkokenttään käyttäjän tunnistetiedot base64-koodattuna merkkijonona muodossa käyttäjänimi:salasana. Jokaisen pyynnön yhteydessä palvelimella tarkastetaan tunnistetietojen oikeellisuus ja käyttäjän rooli. Tähän käytetään osana sovellusta toteutettua auth-middlewarea, jonka kautta kaikki REST-rajapintaan kohdistuvat pyynnöt kulkevat ja joka välittää pyynnön käsittelijälle vain siinä tapauksessa, että tunnistautuminen onnistuu. Ilman salattua yhteyttä basicia ei voi käyttää, koska tunnistetiedot siirretään yhteyden yli selkokielisinä.

Taulukossa 1 on esitetty REST-rajapinnan metodit, niiden parametrit ja paluuarvot sekä järjestelmän osat, joissa metodeita käytetään. GET-pyyntöjen parametrit annetaan osoitteen lopussa ja POST-pyyntöjen pyynnön rungossa JSON (JavaScript Object Notation)-muodossa. Kaikki pyynnöt palauttavat vastauksen JSON-muodossa lukuun ottamatta addCompanyTag-metodia, joka palauttaa pelkän HTTP-tilakoodin. Kirjautuneella käyttäjällä tarkoitetaan käyttäjää, jonka tunnistetiedot on annettu pyynnön mukana Authorization-otsikkokentässä.

Taulukko 1. *REST-rajapinta*

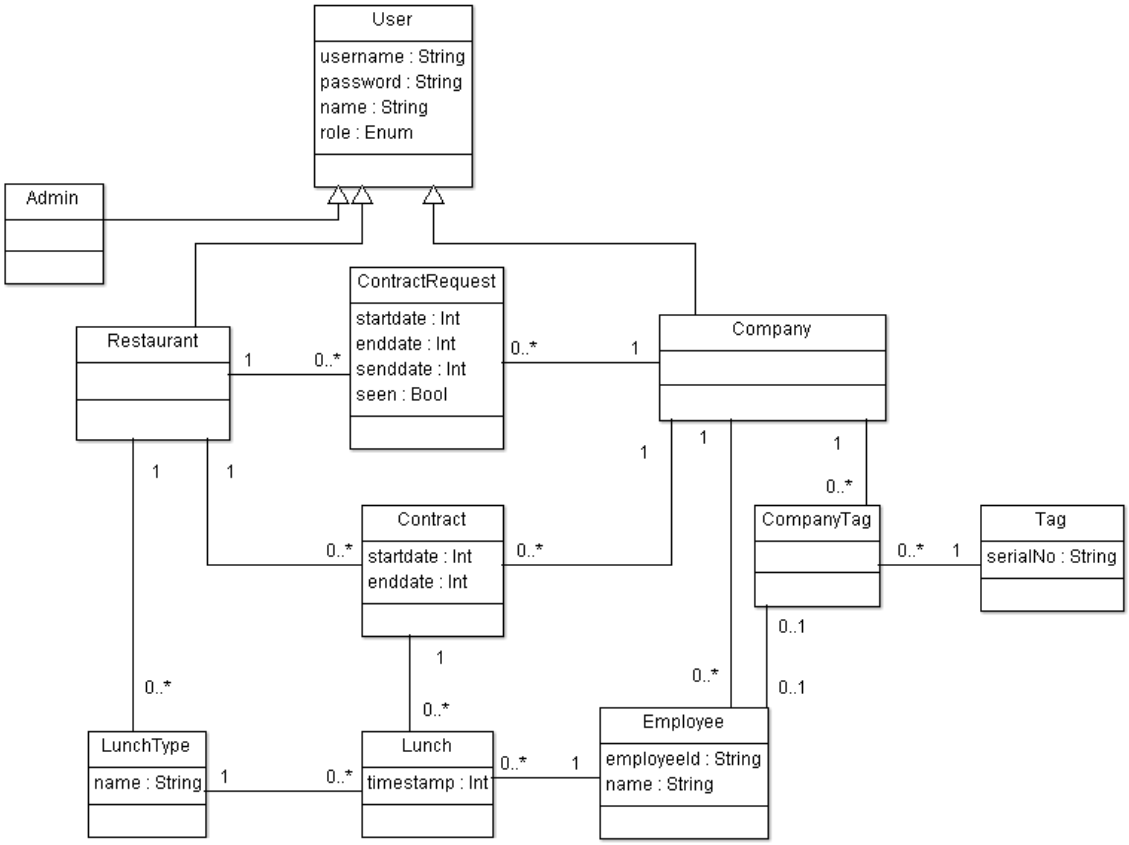
Metodi	Parametrit	Paluuarvo	Kuvaus	Asiakassovellus
GET /lunches	startdate, enddate	[{firstname, lastname, employeeId, restaurant, lunchType, timestamp}]	Palauttaa kirjautuneen yrityksen työntekijöiden annetulla aikavälillä ostamat lounaat.	Tuntikirjausjärjestelmä
GET /role	-	{role}	Palauttaa kirjautuneen käyttäjän roolin.	Mobiilisovellus, työpöytäsovellus
GET /lunchTypes	-	{lunchTypes}	Palauttaa listan kirjautuneen ravintolan lounastyypeistä.	Mobiilisovellus
GET /contracts	tagId	[{contractId, employeeId,	Palauttaa listan sisään kirjautuneen ravintolan	Mobiilisovellus, työpöytäsovellus

		firstName, lastName}]	ja NFC-tunnisteen omistavan yrityksen välisistä voimassa olevista sopimuksista.	
POST /addLunches	employeeId, amount, contract, lunchType	{writeCount}	Lisää työntekijälle parametrina annetun määrän lounaita. Aikaleimaksi määräytyy hetki, jolloin funktiota kutsutaan. Palauttaa tallennettujen lounaiden lukumäärän.	Mobiilisovellus, työpöytäsovellus
POST /addCompanyTag	tagId	-	Lisää parametrina annetun NFC-tunnisteen kirjautuneelle yritykselle.	Mobiilisovellus

Kaikki tietokannan käsittely tapahtuu MySqlAdapterin kautta. MySqlAdapterin rajapinta koostuu funktioista, joissa viimeisenä parametrina on takaisinkutsufunktio, jota kutsumalla MySqlAdapter ilmoittaa kyselyn valmistumisesta. MySqlAdapter muodostaa SQL-kyselyt ja kommunikoi tietokannan kanssa. MySqlAdapterin vaihtoehdoksi mietittiin myös ORM (Object Relational Mapping) -pohjaista ratkaisua. ORM:lle määritellään malleja, jotka ovat käytännössä JavaScript-olioita. Kun mallia muokataan, ORM muodostaa tarvittavat tietokantakyselyt. Lounaslippujärjestelmässä päädyttiin kuitenkin kirjoittamaan kyselyt itse, koska toteutettava järjestelmä on kohtalaisen pieni ja kirjoitettavat SQL-lauseet ovat melko yksinkertaisia. Näin ollen ORM:n opetteluun käytettävä aika kasvaisi helposti saavutettuja hyötyjä suuremmaksi.

5.2 Tietokanta

Järjestelmän tarvitsema data talletetaan MySql-tietokantaan. Tietokanta sisältää järjestelmän käyttäjät (user), joita ovat ravintolat (restaurant), yritykset (company) ja järjestelmänvalvojat (admin). Lisäksi talletetaan sopimukset (contract), joiden avulla tiedetään, minkä yritysten työntekijät voivat käydä syömässä missäkin ravintoloissa. Tietokannan rakenne on esitelty kuvassa 10.



Kuva 10. *Tietokannan käsitekaavio*

Tietokantaan talletetaan myös lounaat (lunch). Jokaisella lounaalla on aikaleima, ja jokaiseen lounaaseen liittyy yksi sopimus ja työntekijä (employee). Näiden tietojen perusteella on mahdollisuus selvittää, kuka on käynyt syömässä, missä ravintolassa ja mihin aikaan. Lounaaseen voi myös liittyä lounastyyppi (lunch type). Lounastyyppien avulla voidaan erotella toisistaan esimerkiksi erihintaiset lounasvaihtoehdot.

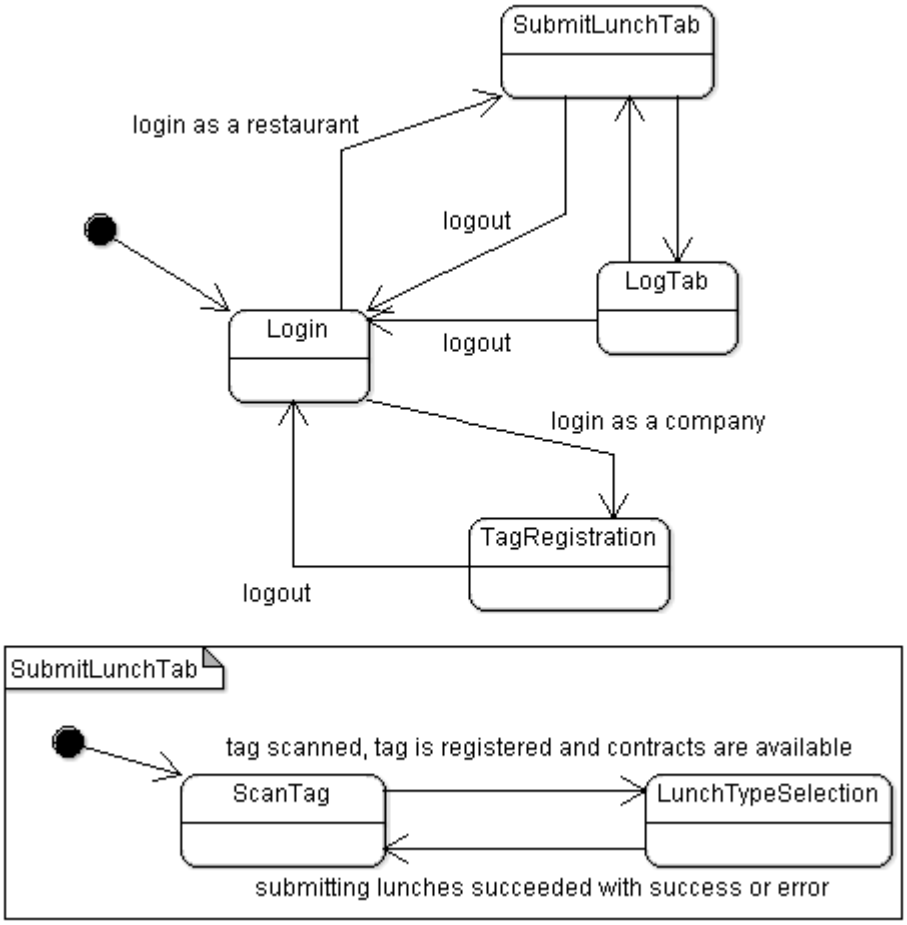
Tietokannan tietojen perusteella on pystyttävä yhdistämään ostettu lounas sen ostaneeseen työntekijään. Työntekijän tunnistaminen tapahtuu NFC-tunnisteen (tag) avulla. Jos työntekijä työskentelee useassa yrityksessä, hänellä voi olla useita NFC-tunnisteita. Tämän takia työntekijää ei yhdistetä suoraan tunnisteeseen, vaan näiden välissä on liitostaulu (company tag). Lisäksi tämän liitostaulun avulla voidaan saada lista yrityksen omistamista tunnisteista, joita ei ole vielä rekisteröity työntekijöille.

5.3 Mobiilisovellus

Tässä luvussa kuvataan toteutetun mobiilisovelluksen toiminnallisuus ja rakenne. Ravintolat käyttävät mobiilisovellusta lounaiden veloittamiseen asiakkaalta. Yritykset pystyvät sovelluksen avulla lukemaan NFC-tunnisteita niiden rekisteröimiseksi työntekijöille.

5.3.1 Käyttöliittymä

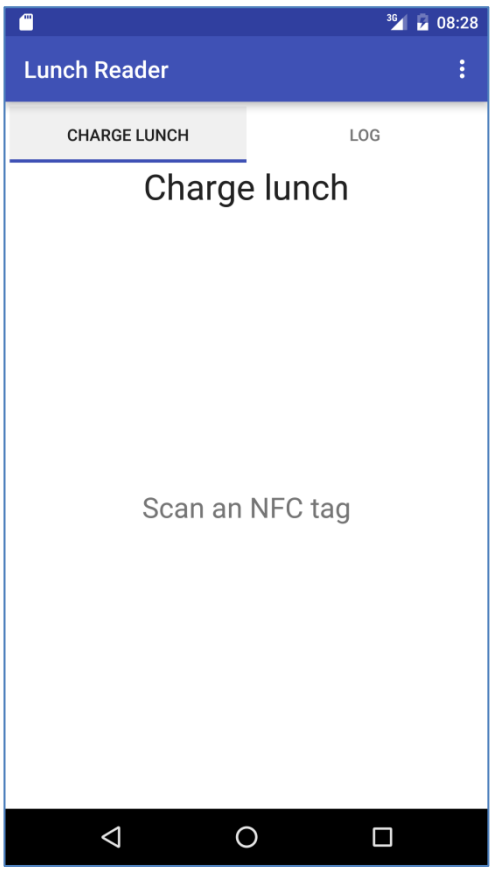
Mobiilisovelluksen käyttöliittymässä on pyritty noudattamaan Android-sovellusten yleisiä suunnitteluperiaatteita. Kuvassa 11 on kuvattu käyttäjän navigointi sovelluksessa. Sovelluksen käynnistyttyä käyttäjälle avautuu kirjautumisnäkymä (login). Tässä näkymässä käyttäjä syöttää käyttäjätunnuksensa ja salasanansa. Onnistuneen kirjautumisen jälkeen näytettävä näkymä määräytyy sen perusteella, onko käyttäjän sisään kirjautumiseen käyttämän tunnuksen rooliksi määritelty ravintola vai yritys. Mistä tahansa järjestelmän näkymästä pystyy kirjautumaan ulos, jolloin käyttäjä palaa takaisin kirjautumisnäkymään.



Kuva 11. *Navigointi mobiilisovelluksessa*

Kirjauduttaessa ravintolan tunnuksilla käyttäjä ohjataan näkymään, jossa on kaksi välilehteä: yhtä (SubmitLunchTab) käytetään lounaiden veloittamiseen ja toista (LogTab) lokitiedon tarkasteluun. Lounaiden veloitusnäkymässä käyttäjä, joka on ravintolan kassalla toimiva työntekijä, voi lukea NFC-tunnisteen lounaan ostavan työntekijän tunnistamiseksi. Jos ravintola on määritellyt useita lounastyyppejä, siirrytään lounastyypin valintaan. Muussa tapauksessa lounastyyppi valitaan automaattisesti ja veloitus tehdään heti tunnisteen lukemisen jälkeen ilman, että ravintolatyöntekijältä edellytetään toimen-

piteitä. Jos käyttäjä vaihtaa välilehteä kesken toimenpiteen, käyttäjän palatessa välilehdelle näytetään tunnisteenlukunäkymä, vaikka käyttäjä olisi aiemmin ollut valitsemassa lounastyyppiä. Tällä varmistetaan se, että luettu NFC-tunniste ei jää vahingossa järjestelmän muistiin ja ettei lounasta veloiteta väärältä työntekijältä. Kuva 12 on ravintolan lounaiden veloitusnäkymästä. Käyttöliittymän ulkoasu on vielä varsin pelkistetty, ja käyttöliittymän toiminnallisuus on pyritty pitämään mahdollisimman yksinkertaisena.



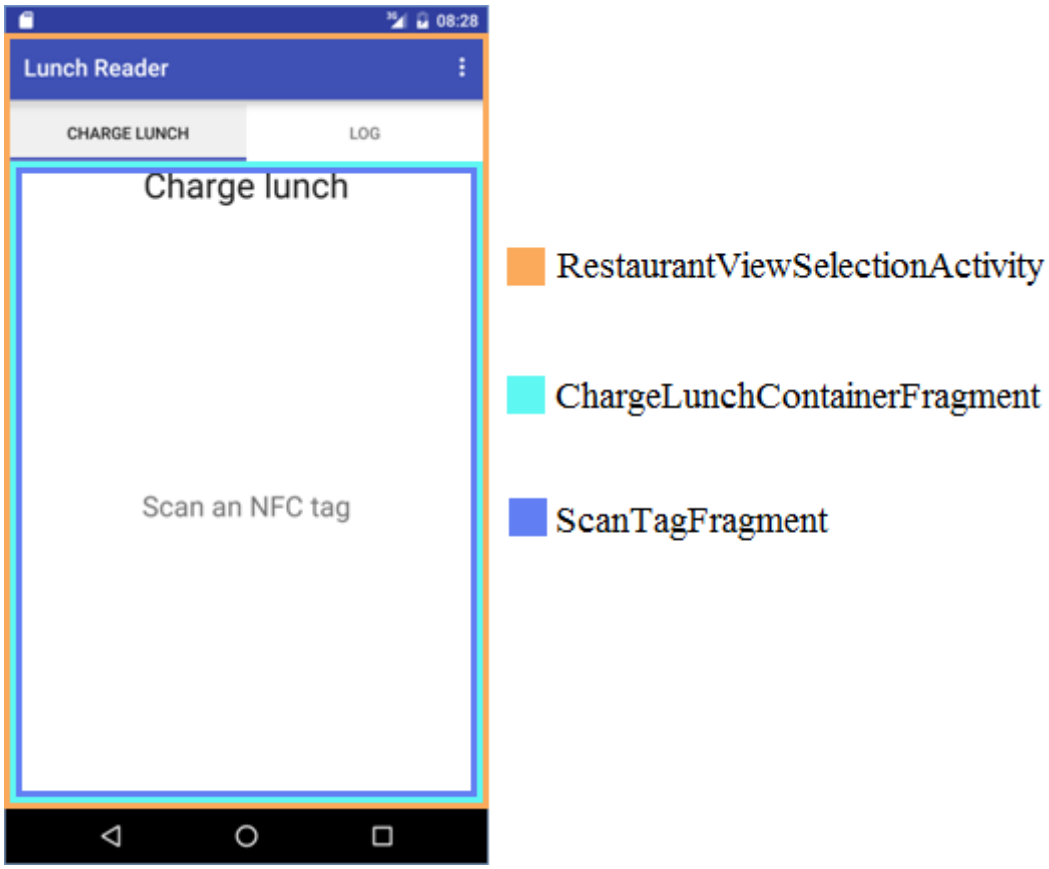
Kuva 12. *Mobiilisovelluksen käyttöliittymä*

Jos sisään kirjaudutaan yrityksen tunnuksilla, näytetään tunnisteiden rekisteröintinäkymä (TagRegistration). Tässä tunnisteiden rekisteröimisellä tarkoitetaan tunnisteen lisäämistä yritykselle ilman, että tunnistetta yhdistetään vielä yrityksen työntekijään. Näkymässä käyttäjä voi lukea NFC-tunnisteen, ja viimeksi luetun tunnisteen sarjanumero näytetään näytöllä. Luettu tunniste voidaan lisätä yritykselle. Lisäksi näkymää voi käyttää tietyn tunnisteen sarjanumeron tarkistamiseen. Tarkistaminen voi olla tarpeellista siinä tapauksessa, että yritys omistaa useita tunnisteita, joille ei ole määritelty työntekijää, ja tietty tunniste pitäisi rekisteröidä työntekijälle.

5.3.2 Toteutus

Mobiilisovellus sisältää kolme aktiviteettia (activity): LoginActivity, CompanyViewSelectionActivity ja RestaurantViewSelectionActivity. Aktiviteetti on

Android-järjestelmässä komponentti, joka kuvaa yksittäistä asiaa, jonka käyttäjä voi tehdä. Aktiviteetti muistuttaa toiminnallisuudeltaan tietokoneiden käyttöjärjestelmien ikkunoita. Usein aktiviteetin tehtäviin kuuluu huolehtiminen interaktiosta käyttäjän kanssa. LoginActivity toteuttaa kirjautumisnäkymän ja tarkistaa käyttäjän syöttämät tunnistetiedot. Onnistuneen kirjautumisen päätteeksi aktiviteetti suljetaan ja tunnuksen roolista riippuen ladataan joko RestaurantViewSelectionActivity tai CompanyViewSelectionActivity. Näiden näkymien sisällä näkymänvaihdot on toteutettu fragmenttien (fragment) avulla. Fragmentti on Android-järjestelmässä käyttöliittymän itsenäinen osakokonaisuus. Fragmenttien vaihtaminen on suorituskyvyn kannalta kevyempi operaatio kuin aktiviteetin sulkeminen ja käynnistäminen. Lisäksi, toisin kuin aktiviteettien, fragmenttien välillä ei pääse siirtymään Android-järjestelmän takaisin-napilla. Kuvassa 13 on esitetty ravintolanäkymän aktiviteetti ja fragmentit. Ylimmällä tasolla on yksi aktiviteetti (RestaurantViewSelectionFragment), joka lataa säiliöfragmenttiin (ChargeLunchContainerFragment) näkymäfragmentin (ScanTagFragment) valitun välilehden perusteella.



Kuva 13. *Ravintolanäkymän käyttöliittymän osat*

Ravintolanäkymän ylimmällä tasolla on RestaurantViewSelectionActivity, joka huolehtii NFC-tunnisteiden lukemisesta ja välittää lukutapahtuman alinäkymälle, joka on aktiivinen lukuhetkellä. Alinäkymät ovat kahdella eri välilehdellä, joiden vaihtaminen tehdään RestaurantTabAdapter-komponentin avulla. Välilehtiä on kaksi: yksi lounaiden

veloittamista ja toinen lokia varten. Lounaiden veloitusnäkymä on toteutettu MVP-suunnittelumallin (Model-View-Presenter) mukaisesti. MVP-mallissa näkymä (view) on passiivinen. Näkymä välittää käyttäjän toiminnot esittäjälle (presenter), ja niiden perusteella esittäjä päivittää mallia (model) ja formatoi käyttöliittymässä näytettävän datan esitettävään muotoon. Veloitusvälilehdellä on kaksi alinäkymää, joiden vaihtamisesta esittäjä huolehtii: ScanTagFragment ja SelectLunchTypeFragment. Esittäjä tekee käyttäjän komentojen perusteella HTTP-pyyntöjä palvelimelle ja päivittää näkymää vastausviestien perusteella. Yritysnäkymässä ei ole käytetty mitään tiettyä suunnittelumallia, koska se koostuu vain yhdestä näkymästä: RegisterTagFragment.

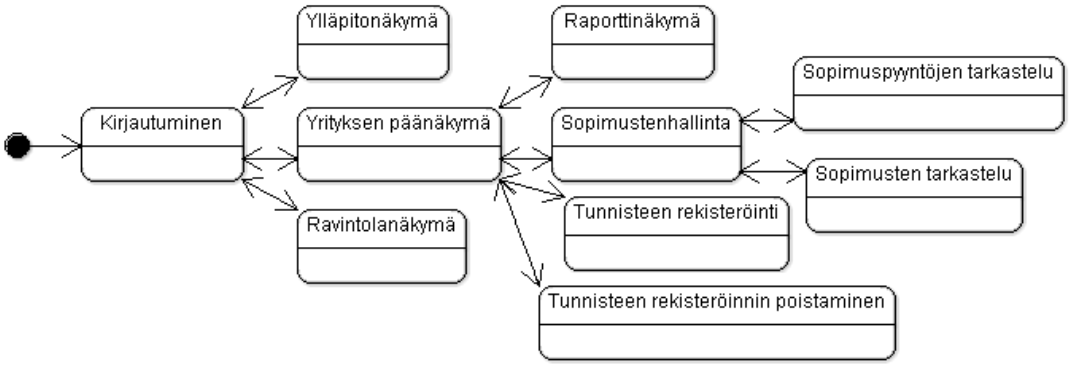
Edellä mainittujen komponenttien lisäksi sovellus sisältää muutamia apu- ja mallikomponentteja, joiden tehtävänä on abstrahoida usein käytetty toiminnallisuus yhteen paikkaan. HttpsConnection tekee kaikki HTTPS-pyynnöt palvelimelle. Jokaista erilaista pyyntöä varten se sisältää oman metodin. HttpsConnection käyttää paluuviestien mallintamiseen HttpResponse-luokkaa, joka sisältää kentät paluuviestin tilakoodille sekä viestin mukana saapuneelle datalle. ActionLog-komponentti sisältää lokimerkinnät, jotka näytetään ravintolanäkymän lokivälilehdellä. Lisäksi sopimuksille, lounastyypeille ja kirjautuneelle käyttäjälle on omat malliluokkansa.

5.4 Web-käyttöliittymä

Web-käyttöliittymässä tehdään kaikki ylläpidolliset toiminnot. Jokaisella käyttäjäroolilla on oma näkymänsä. Oikea näkymä valitaan käyttäjätunnuksen perusteella sisään kirjautumisen yhteydessä.

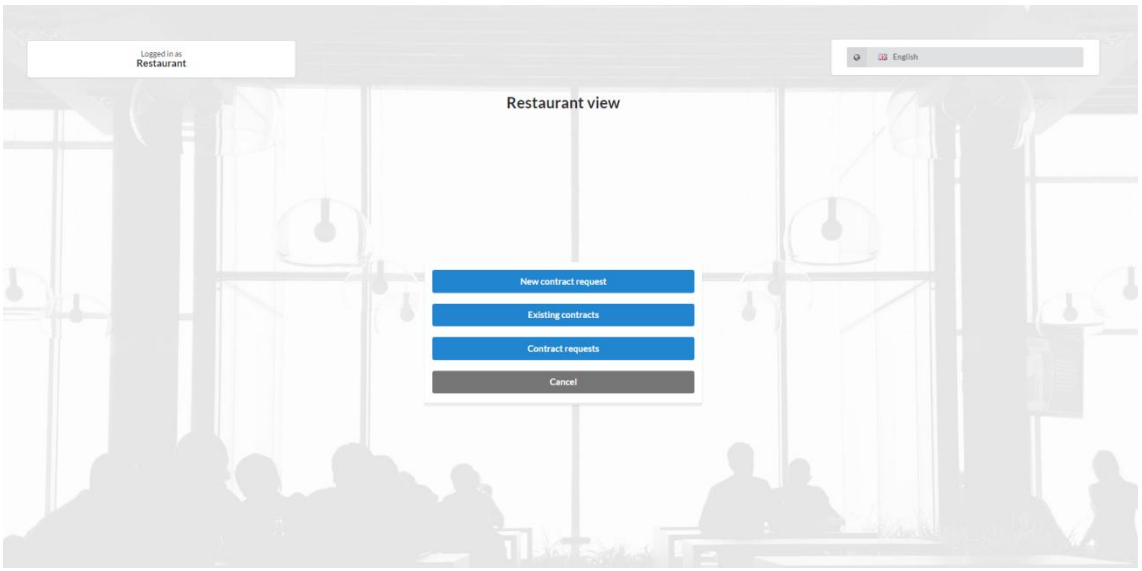
5.4.1 Käyttöliittymä

Järjestelmän etusivulle käyttäjä pääsee menemällä selaimella lounaslippujärjestelmän osoitteeseen. Ensimmäisenä käyttäjälle avautuu kirjautumisnäkymä. Kirjautumisnäkymästä siirrytään joko ravintolan, yrityksen tai järjestelmänvalvojan päänäkymään. Jokaisessa näkymässä voi vaihtaa käyttöliittymän kieltä, ja ulos kirjautuminen on mahdollista kaikissa näkymissä kirjautumisnäkymää lukuun ottamatta. Kuvassa 14 näkyy navigointi web-käyttöliittymässä. Tilan säästämiseksi vain yritysnäkymästä on esitetty myös alinäkymät ja osa navigoinnin kannalta merkityksettömistä näkymistä, esimerkiksi tunnisteen rekisteröinnin poistamisen vahvistusnäkymä, on jätetty pois kuvasta. Kuvassa näkyvien siirtymien lisäksi jokaisesta näkymästä voidaan ulos kirjautumalla palata kirjautumisnäkymään.



Kuva 14. *Navigointi web-käyttöliittymässä*

Käyttöliittymässä navigoidaan näkymän keskellä olevien nappien avulla (kuva 15). Toinen harkinnassa ollut toteutusvaihtoehto olisi ollut sivun yläreunassa oleva navigointipalkki. Nykyiseen ratkaisuun kuitenkin päädyttiin, koska navigointihierarkia on matala ja monet toiminnot edellyttävät näkymien avaamista tietyssä järjestyksessä. Esimerkiksi ravintolan poistamisen hyväksymisnäkymää ei voi avata, jos aikaisemmin ei ole valittu poistettavaa ravintolaa ylläpitäjän ravintolanäkymässä.



Kuva 15. *Ravintolan sopimustenhallintanäkymä*

Käyttöliittymässä käytetään Semantic UI:ta. Semantic UI on web-käyttöliittymäkirjasto, joka helpottaa tyylillisesti yhdenmukaisen käyttöliittymän toteuttamista. Käyttöliittymän ulkoasu määräytyy Semantic UI:ssa tarjolla olevien komponenttien perusteella. Käyttöliittymä on suurelta osin sinisävytteinen, koska sinistä väriä käytetään paljon myös muissa Wapicen tuotteissa.

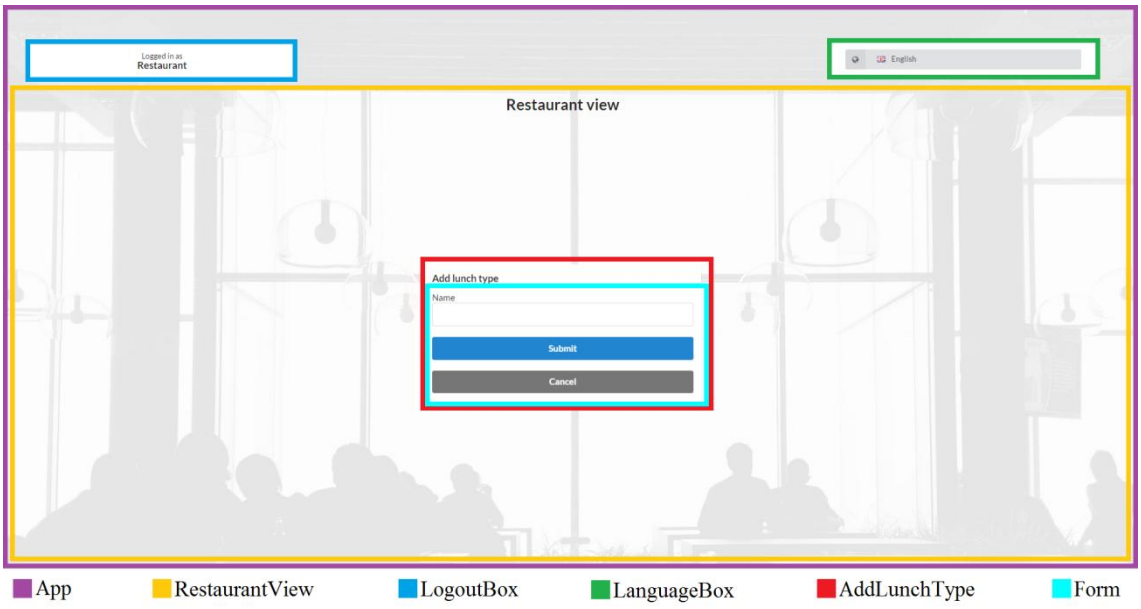
5.4.2 Toteutus

Web-käyttöliittymä on toteutettu Reactilla. Jokaista näkymää varten on oma komponentti, jonka render-metodissa määritellään näkymän sisältämät elementit ja näkymän ulkoasu. React-komponentit käyttävät Semantic UI:ta HTML-elementtien määrittelyissä, minkä ansiosta elementtien ulkoasu on yhdenmukainen. Taulukossa 2 on kuvattu järjestelmän komponentit, jotka ovat yhteisiä kaikille käyttäjärooleille.

Taulukko 2. *Käyttäjärooleille yhteiset React-komponentit*

Nimi	Kuvaus
App	Toimii ylimmän tason näkymäkomponenttina. Sisältää yläpalkin, jossa ovat uloskirjautumis- ja kielivalintalaatikot, sekä tilan, johon ladataan näytettävä näkymä.
Login	Toteuttaa kirjautumisnäkymän.
LanguageBox	Toteuttaa sivun oikeassa yläkulmassa sijaitsevan kielivalintalaatikon.
LogoutBox	Toteuttaa sivun vasemmassa yläkulmassa sijaitsevan uloskirjautumislaatikon.
Datepicker	Toteuttaa Semantic UI:n tyylin mukaisen päivämääränvalintakomponentin. Komponentti näyttää kuukausinäkymän, josta voi valita päivämäärän. Kuukautta pystyy vaihtamaan komponentin yläreunassa olevilla nuolilla. Komponenttia käytetään lomakkeissa.
Form	Toteuttaa lomakkeen, jolle voidaan antaa parametrina sen sisältämien kenttien nimet ja tyypit. Lomake sisältää myös lähetä- ja peruuta-napit.
Table	Toteuttaa taulukon, jolle voidaan antaa parametrina otsikkokenttien kuvaukset ja taulukon sisältämä data. Jos datarivien määrä ylittää taulukolle asetetun enimmäisrivimäärän, näytetään sivunvalintapalkki.

Taulukossa 2 esitetyistä komponenteista LanguageBox ja LogoutBox ovat näkyvissä joka näkymässä. Näkymät ladataan App-komponentissa näkymille varattuun tilaan, joka sijaitsee sivun keskellä. Sisään kirjautumisen jälkeen näytettävät näkymät määräytyvät kirjautumiseen käytetyn tunnuksen perusteella. Kuvaan 16 on merkitty ravintolan lounastyypin lisäysnäkymän React-komponentit. Tämä näkymä on valittu esimerkiksi, koska se sisältää hierarkkisia React-komponentteja.



Kuva 16. *React-komponentit lounastyypin lisäysnäkymässä*

RestaurantView on näkyvissä koko ajan, jonka ravintola on sisään kirjautuneena. Muut näkymät (esimerkiksi kuvassa 16 AddLunchType) ladataan sen alinäkymiksi tarvittaessa. Alinäkymistä vain yksi on samaan aikaan aktiivisena ja ne voivat edelleen koostua muista React-komponenteista. Ravintolanäkymän komponentit on esitetty tarkemmin taulukossa 3.

Taulukko 3. *Ravintolanäkymän React-komponentit*

Nimi	**Kuvaus**
RestaurantView	Toimii ravintolanäkymän ylätason komponenttina, jonka sisällä näytetään kaikki muut tässä taulukossa esitettävät näkymät.
RestaurantViewSelection	Toteuttaa näkymän, joka on näkyvissä ravintolanäkymään saavuttaessa. Sisältää navigointiyhteydet lounastyyppien ja sopimusten hallinnointiin.
LunchTypeManagement	Toteuttaa näkymän, jossa on lista ravintolan lounastyypeistä.
AddLunchType	Toteuttaa näkymän, jossa ravintolalle voi lisätä lounastyypin. Lisääminen tapahtuu lomakkeella, jossa määritellään lounastyypin nimi.
EditLunchType	Toteuttaa näkymän, jossa lounastyyppiä voi muokata. Muokkaaminen tapahtuu lomakkeella, johon tulee esitäytettynä lounastyypin tiedot.
RemoveLunchType	Toteuttaa vahvistusnäkymän lounastyypin poistamista varten.
RestaurantContractManagement	Toimii navigointinäkymänä sopimuksiin liittyville toiminnoille. Sisältää navigointimahdollisuuden sopimusten ja

	lähetettyjen sopimuspyyntöjen tarkastelunäkymiin sekä uuden sopimuspyynnön luomisnäkymään.
RestaurantContracts	Näyttää listan ravintolan sopimuksista.
RestaurantContractRequests	Näyttää listan ravintolan vahvistamattomista sopimuspyynnöistä.
RestaurantNewContractRequest	Toteuttaa lomakkeen, jolla voi lähettää uuden sopimuspyynnön yritykselle
RestaurantCancelContractRequest	Toteuttaa vahvistusnäkymän, jossa voi valita, halutaanko valittu sopimuspyyntö todella perua.

Yrityksen näkymän rakenne on samankaltainen kuin ravintolan näkymällä. Näkymän komponentit on esitetty taulukossa 4.

Taulukko 4. *Yritysnäkymän React-komponentit*

Nimi	**Kuvaus**
CompanyView	Toimii yritysnäkymän ylätason komponenttina, jonka sisällä näytetään kaikki muut tässä taulukossa esitettävät näkymät.
CompanyViewSelection	Toteuttaa näkymän, joka on näkyvissä yritysnäkymään saavuttaessa. Sisältää navigointiyhteydet sopimusten hallinnointiin, tunnisteiden rekisteröintiin, tunnisteiden rekisteröinnin poistamiseen ja raporttien tarkasteluun.
ContractManagement	Toimii navigointinäkymänä sopimuksiin liittyville toiminnoille. Sisältää navigointimahdollisuuden sopimusten ja saapuneiden sopimuspyyntöjen tarkastelunäkymiin..
CompanyContracts	Näkymä sisältää listan yrityksen voimassa olevista sopimuksista.
CompanyReceivedContractRequests	Näkymä sisältää listan ravintoloilta saapuneista käsittelemättömistä sopimuspyynnöistä. Näkymässä pyynnön voi joko hyväksyä tai hylätä.
AcceptContract	Näkymä näyttää valitun sopimuksen tiedot ja kysyy varmistuksen, halutaanko sopimuspyyntö hyväksyä.
RejectContract	Näkymä näyttää valitun sopimuksen tiedot ja kysyy varmistuksen, halutaanko sopimuspyyntö hylätä.
TagRegistration	Toteuttaa lomakkeen, jolla NFC-tunniste voidaan rekisteröidä työntekijälle. Lomake sisältää valikon, josta voidaan valita tunniste aiemmin mobiilisovelluksella luettujen tunnisteiden listasta sekä tekstikentät työntekijän ID:tä ja nimeä varten.
TagUnregistration	Näyttää listan yrityksen työntekijöille rekisteröidyistä tunnisteista. Näkymässä tunnisteen rekisteröinnin voi poistaa.

ConfirmTagUnregistration	Näkymässä vahvistetaan, halutaanko tunnisteen rekisteröinti poistaa.
CompanyReportView	Näkymässä näytetään yrityksen raportit.

Myös yrityksellä on ylätason näkymäkomponentti (CompanyView), jonka sisälle alinäkymät ladataan. Raporttinäkymään (CompanyReportView) on tarkoitus toteuttaa useita erilaisia raporttityyppejä. Järjestelmän ensimmäisessä versiossa näkymä sisältää yhden ennalta määritellyn raporttityypin. Raportti sisältää yhden rivin jokaista lounasta kohti. Rivillä on esitetty työntekijän tiedot, ravintolan nimi, yrityksen nimi ja lounaan ostamisajankohta. Rivejä on mahdollista ryhmitellä ravintolan ja työntekijän perusteella, ja valinnaiseksi raportointiväliksi voidaan valita viikko, kuukausi tai vuosi. Jos ryhmittelyvaihtoehto on valittu, tarkan ostamisajankohdan sijaan näytetään summasarake. Jos raportti on esimerkiksi ryhmitelty ravintolan ja viikon perusteella, summasarakkeessa näytetään kaikkien yrityksen työntekijöiden yhdellä viikolla tietyssä ravintolassa ostamien lounaiden summa. Tällöin raportti sisältää yhden rivin ravintola-viikko-yhdistelmää kohti.

Taulukko 5. *Ylläpitonäkymän React-komponentit*

Nimi	**Kuvaus**
AdminView	Toimii ylätason näkymänä ylläpitonäkymälle.
AdminViewSelection	Näkymä, joka näytetään ylläpitonäkymään saavuttaessa ja josta pääsee navigoimaan ravintoloiden, yritysten ja ylläpitäjien hallinnointinäkymiin.
RestaurantManagement	Näyttää listan järjestelmän ravintoloista. Näkymä sisältää toiminnot ravintoloiden lisäämiselle, poistamiselle ja muokkaamiselle.
AddRestaurantView	Sisältää lomakkeen, jolla järjestelmään voi lisätä ravintolan.
EditRestaurantView	Sisältää lomakkeen, jolla voi muokata ravintolan tietoja. Vanhat tiedot on esitäytettynä.
RemoveRestaurantView	Näkymässä vahvistetaan, halutaanko ravintola poistaa.
CompanyManagement	Näyttää listan järjestelmän yrityksistä. Näkymä sisältää toiminnot yritysten lisäämiselle, poistamiselle ja muokkaamiselle.
AddCompanyView	Sisältää lomakkeen, jolla järjestelmään voi lisätä yrityksen.
EditCompanyView	Sisältää lomakkeen, jolla voi muokata yrityksen tietoja. Vanhat tiedot on esitäytettynä.
RemoveCompanyView	Näkymässä vahvistetaan, halutaanko yritys poistaa.
AdminManagement	Näyttää listan järjestelmän ylläpitäjistä. Näkymä sisältää

	toiminnot ylläpitäjien lisäämiselle, poistamiselle ja muokkaamiselle.
AddAdminView	Sisältää lomakkeen, jolla järjestelmään voi lisätä ylläpitäjän.
EditAdminView	Sisältää lomakkeen, jolla voi muokata yrityksen tietoja. Vanhat tiedot on esitäytettynä.
RemoveAdminView	Näkymässä vahvistetaan, halutaanko ylläpitäjä poistaa. Ylläpitäjä ei pysty poistamaan omaa käyttäjätunnustaan.

Web-kayttöliittymä on yhden sivun sovellus, joten se vaatii vain yhden sivulatauksen koko istunnon aikana. Kun kirjautumissivu on ladattu, kaikki muu tiedonsiirto käyttöliittymän ja palvelimen välillä tehdään AJAX-pyynnöillä. Jokainen näkymä luo tarvitsemansa pyynnöt sekä käsittelee niihin tulevat vastaukset. Pyyntöjen käsittelijät alustetaan kunkin näkymän componentDidMount-metodissa, jota kutsutaan automaattisesti, kun komponentti ladataan näkymään.

React-komponenttien lisäksi web-käyttöliittymässä on käännöskomponentti LanguageSelector, jota käytetään käyttöliittymän tekstien vaihtamiseen. Käännöskomponentilla on setLanguage-funktio, jolla komponentille asetetaan uusi käännösolio. Olio koostuu avain-arvo-pareista. Ohjelmassa 5 on esimerkki käännösolion rakenteesta ja käännettyjen tekstien käyttämisestä HTML-elementeissä.

```
var finnishTexts = {
    login: ’kirjaudu’,
    logout: ’kirjaudu ulos’,
    name: ’nimi’
}

var languageSelector = require(‘./helpers/languageSelector’);
var tr = languageSelector.getTranslation;
var globalEmitter = require(‘./globalEmitter’);

languageSelector.setLanguage(‘fi’, finnishTexts);

var Login = React.createClass({
    componentDidMount: function() {
        globalEmitter.on('languageChanged', this.forceUpdate);
    },
    render: function() {
        return (
            …
            <button>{tr(‘login’)}</button>
            …
       );
    }
});
```

Ohjelma 5. *Esimerkki languageSelector-komponentin käytöstä*

Kielen vaihtamisen jälkeen käännöskomponentti ilmoittaa kielen vaihtumisesta languageChanged-tapahtumalla, jota käyttöliittymäkomponentit kuuntelevat. Kaikki käyttöliittymän tekstit menevät getTranslation-funktion kautta. Funktiolle annetaan parametrina käännösolion avaimen nimi, ja funktio palauttaa avainta vastaavan merkkijonon.

5.5 Integraatio tuntikirjausjärjestelmään

Wapicella käytössä olevaan tuntikirjausjärjestelmään lisättiin näkymä lounasmäärien raportointia varten. Näkymän tärkein ominaisuus on PDF-tiedostojen jäsentäminen, mutta näkymään voi ladata dataa myös lounaslippujärjestelmän palvelimelta. Integraatiossa käytetyt teknologiat ovat ASP .NET ja AngularJS, koska niillä on toteutettu myös tuntikirjausjärjestelmän muut osat.

5.5.1 Palvelin

Palvelin on toteutettu ASP .NET:llä. Palvelimelle lisättiin lounasnäkymälle erillinen pyyntöjen käsittelijä (controller). Näkymä voi lähettää palvelimelle neljä erilaista pyyntöä: Sivun lataus, PDF-tiedostojen lukeminen, lukeminen lounaslippujärjestelmästä ja Excel-raportin muodostaminen. Sivun lataamisen yhteydessä näkymään haetaan työntekijätiedot tuntikirjausjärjestelmästä ja niihin lisätään lounaita varten summasarake, jossa sivun lataamisvaiheessa ei vielä ole yhdellekään työntekijälle lounaita.

PDF-tiedostojen lukemista varten tehtiin erillinen komponentti PdfParser. PdfParserilla on tiedostojen lukemista varten metodi, jolle annetaan parametrina HTTP-pyynnön mukana saapunut tiedosto sekä tiedoston formaatin nimi. Jokaiselle formaatille on käsittelijäfunktio, joka lukee kunkin työntekijän lounasmäärät PDF-tiedostosta ja lisää ne työntekijälistan summasarakkeeseen. PDF-tiedostojen lukeminen tehdään käyttäen avoimen lähdekoodin iTextSharp-kirjastoa. Kirjaston avulla PDF-tiedosto voidaan jäsentää tekstiriveiksi. Kirjasto mahdollistaa myös tekstielementtien lukemisen tietyn suorakulmion sisällä. Tämän ansiosta esimerkiksi tiedoston otsikkokenttien, esimerkiksi ravintolan nimen ja osoitetietojen, jättäminen jäsennettävän alueen ulkopuolelle yksinkertaistuu. PDF-tiedoston jäsentämislogiikka tietylle formaatille voisi olla esimerkiksi seuraavanlainen: Luetaan rivejä silmukassa aloittaen riviltä 7. Työntekijän nimi haetaan kunkin rivin toisesta sarakkeesta ja lounaiden kappalemäärä viidennestä sarakkeesta.

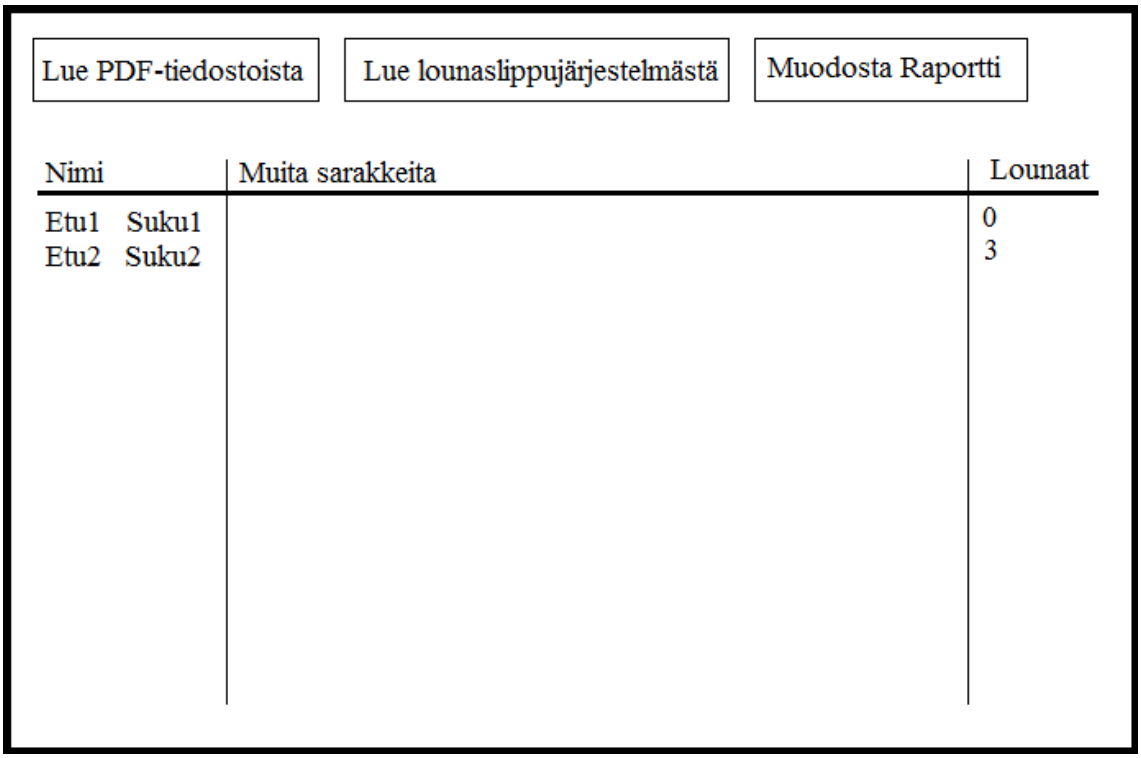
Lounaslippujärjestelmästä lounaita luetaan sen REST-rajapinnan kautta. Tuntikirjausjärjestelmästä lähetetään pyyntö lounaslippujärjestelmän palvelimelle. Rajapinnalle tuntikirjausjärjestelmä autentikoituu basic-menetelmällä käyttäen Wapicen lounaslippujärjestelmän tunnuksia. Rajapinta palauttaa listan, joka sisältää yhden alkion jokaista ostettua lounasta kohti parametrina annetulla aikavälillä. Tuntikirjausjärjestelmässä listasta lasketaan kunkin työntekijän lounaat ja lisätään ne työntekijälistan summasarakkeeseen.

Lounastiedot olisi mahdollista lukea myös suoraan lounaslippujärjestelmän tietokannasta, mutta REST-rajapintaa käytettäessä muutoksia ei tarvitse tehdä useaan paikkaan, jos lounaslippujärjestelmän toteutus muuttuu.

Excel-tiedoston luomiseen käytetään Microsoftin COM (Component Object Model) teknologiaa. Tiedostoon sisältyvät samat tiedot, jotka ovat näkyvillä lounasnäkymässä. Näin ollen tiedoston luomiseen tarvittavat tiedot ovat jo näkymässä ja tiedosto voitaisiin luoda myös frontendin puolella. Selaimen JavaScriptillä ei kuitenkaan ole oikeuksia luoda tiedostoja, minkä takia tiedosto luodaan palvelimella ja ladataan näkymän kautta.

5.5.2 Näkymä

Tuntikirjausjärjestelmän lounasnäkymä on toteutettu AngularJS:llä. AngularJS mahdollistaa HTML-elementtien ja JavaScript-muuttujien sitomisen joko yksi- tai kaksisuuntaisesti. Näin ollen näkymän HTML-sivu päivittyy automaattisesti, kun lounaat sisältävä tietorakenne päivittyy JavaScriptin puolella.



Kuva 17. *Lounasnäkymän asettelu*

Tuntikirjausjärjestelmä on Wapicen sisäiseen käyttöön kehitetty tuote, eikä sitä ole tarkoitettu esiteltäväksi yrityksen ulkopuolisille ihmisille. Tästä syystä siitä ei voi laittaa tähän kuvaa. Kuvassa 17 on kuitenkin esitetty näkymän rakenne. Näkymässä voi lukea lounaita, ja lukemisen seurauksena määrät päivittyvät työntekijälistaan. Luettujen lounaiden perusteella muodostetaan Excel-tiedosto, joka on ladattavissa palvelimelta. Lounasnäkymän toiminnallisuus on kuvattu tarkemmin luvussa 3.1.4.

6. TESTAUS

Järjestelmää on testattu enimmäkseen tutkivaa testausta hyväksi käyttäen. Joillekin komponenteille on kirjoitettu myös automatisoituja testejä. Lisäksi toteutettiin pilottivaihe, jossa järjestelmää testasivat loppukäyttäjät. Tässä luvussa kerrotaan testiautomaatioon käytetyistä menetelmistä sekä kuvataan pilotin kulku ja siitä saadut tulokset.

6.1 Palvelin

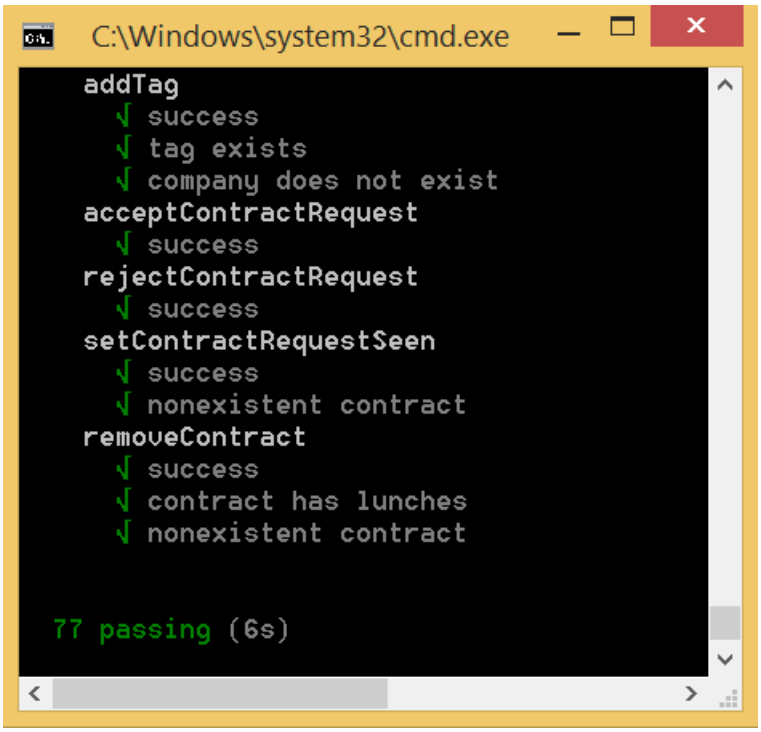
Palvelimen ainoa yksikkötestattava komponentti on MySqlAdapter. MySqlAdapterin testaamiseen on käytetty Mocha-testausympäristöä ja assert-kirjastoa. Mocha suorittaa testit sarjallisesti ja testin päätyttyä raportoi, mitkä testit onnistuivat ja mitkä eivät. Alla on esimerkki Mocha-testistä funktiolle, joka tarkistaa, onko käyttäjä olemassa tietokannassa.

```
describe("doesUserExist", function() {
  it("user exists", function(done) {
    msa.doesUserExist(restaurant1, function(err, exists) {
      assert(!err && exists, 'user should exist');
      done();
    });

  });

  it("user doesn't exist", function(done) {
    msa.doesUserExist(newRestaurant, function(err, exists) {
      assert(!err && !exists, 'user should not exist');
      done();
    });
  });
});
```

Ohjelma 6. *Esimerkki Mocha-testistä*

Mocha-testi koostuu describe- ja it-lauseista. Describe-lauseet kuvaavat kokonaisuuden, johon testi kuuluu, ja lauseita voi olla sisäkkäin useita. Ylemmän tason describe-lauseena voi olla esimerkiksi yritysten hallinnointi, minkä alla puolestaan olisi uuden yrityksen lisääminen. Describe-lauseille voi myös asettaa before-, beforeEach-, after- ja afterEach-käsittelijöitä. Niissä voidaan määritellä alustuksia, joita tehdään ennen tai jälkeen yksittäisen testitapauksen tai koko testijoukon suorittamista. Lounaslippujärjestelmää testattaessa jokaista testitapausta ennen varmistetaan, että tietokanta on ennalta määritellyssä tilassa. Esimerkiksi ohjelmassa 6 esiintyvä restaurant1 täytyy olla olemassa ja newRestaurant ei saa olla olemassa.

It-lauseet kuvaavat testitapaukset. Aiemmin mainitun yrityksen lisäysesimerkin tapauksessa testitapauksena voisi olla yrityksen lisääminen liian lyhyellä käyttäjätunnuksella. It-lauseen sisällä kutsutaan MySqlAdapterin testattavaa funktiota ja tarvittaessa tehdään kysely tietokantaan suoraan. Tämän jälkeen tarkistetaan, että tietokantakyselyn tulos on sellainen kuin odotettiin. Esimerkiksi yritettäessä lisätä yritystä liian lyhyellä käyttäjänimellä funktiokutsun jälkeen tarkistetaan, että yritystä ei lisätty tietokantaan. Kuvassa 18 esitetään Mochan testitulosten raportointinäkymä.



Kuva 18. *Mocha-testitulokset*

Koodissa tarkistusten tuloksista ilmoitetaan käyttämällä assert-kirjastoa, joka aiheuttaa poikkeuksen ja näyttää virheilmoituksen, jos sille parametrina annettu ehto ei toteudu. Jos testiä ajettaessa ei aiheudu poikkeuksia, Mocha ilmoittaa testin onnistumisesta vihreällä oikeinmerkillä. Testeissä havaitut virheet on korjattu heti testien ajamisen jälkeen, joten testeistä ei ole olemassa erillistä testausraporttia.

6.2 Web-käyttöliittymä

Myös web-käyttöliittymän testaamiseen on käytetty Mocha-testausympäristöä. Mocha-testien sisällä käyttöliittymän toimintojen simulointiin käytetään Selenium webdriver-kirjastoa. Selenium käynnistää selaimen, jossa on Seleniumin avulla mahdollista suorittaa toimintoja JavaScript-koodista. Toiminnot ovat sellaisia, joita järjestelmän käyttäjä voisi tehdä. Selenium voi esimerkiksi klikata elementtiä, kirjoittaa tekstiä tai navigoida tiettyyn www-osoitteeseen. Lisäksi Seleniumin avulla voi etsiä tiettyä elementtiä sen id:n, CSS-luokan tai DOM-sijainnin perusteella. Alla on esimerkki Seleniumia hyväksi käyttäen toteutetusta testistä.

```
it("register tag, success", function(done) {

  // Ensure that tag is not registered
  var query =  'delete from tag where tag.serial_no = "04C387FAD94980"';
  db.query (query, function (err, rows, fields) {

    // Click register tag and wait for page load
    driver.findElement(By.id('register-tag-button')).click();
    driver.wait(until.elementLocated(By.id('register-tag-div')), 5000);

    // Enter employee id and click submit
    driver.findElement(By.id('employeeid')).sendKeys('99999');
    driver.findElement(By.id('register-tag-submit')).click();

    driver.wait(until.elementIsVisible(driver.findElement(By.id(
    'status-msg'))), 5000);

    driver.findElement(By.id('status-msg'))
    .getAttribute('innerHTML').then(function(text) {

      // Check that success message is correct
      assert.equal (text, 'Tag registered successfully.');

      // Check that tag is added to db
      query = 'select count(*) as count from tag where serial_no =
      "04C387FAD94980"';

      db.query (query, function (err, rows, fields) {
        assert (rows[0].count === 1, 'Tag was not added to db');
        done();
      });
    });
  });
});
```

Ohjelma 7. *Esimerkki Selenium-testitapauksesta*

Seleniumin avulla testaamisen haasteena on, että kaikki toiminnot ovat asynkronisia, koska HTTP-kommunikaatiossa ja näkymien vaihdossa on viiveitä. Näin ollen testikoodista tulee vaikeasti luettavaa sisäkkäisten takaisinkutsujen takia. Ongelma ja sen ratkaisut on kuvattu tarkemmin luvussa 4.2. Valtaosa testikoodista on kirjoitettu käyttäen lupauksia, mikä mahdollistaa asynkronisen koodin kirjoittamisen siten, että se näyttää sarjalliselta. Tietokantakyselyt on kuitenkin tehtävä asynkronisesti.

6.3 Pilottivaihe

Lounaslippujärjestelmän pilotti toteutettiin 22.11.2016 – 23.12.2016. Pilotissa testauksen kohteena oli Wapicen omaan käyttöön toteutettu työpöytäsovellus. Kohteeksi valittiin työpöytäsovellus, koska se on järjestelmän osa, joka tulee ensimmäisenä käyttöön. Työpöytäsovelluksen testaaminen oli myös mahdollista toteuttaa yrityksen sisällä ilman ravintoloiden osallistumista. Mobiilisovelluksen pilotoinnista täytyisi sopia erikseen ravintoloiden kanssa. Mobiilisovellusta voidaan tulevaisuudessa pilotoida siten, että

Wapicen toimistolle rakennetaan valeravintola, jossa lounaat veloitetaan mobiilisovelluksella samalla tavalla kuin oikeissakin ravintoloissa. Tällaisen pilotin aikatailua ei ole vielä suunniteltu. Web-käyttöliittymä oli mukana pilotissa siten, että sen kautta Wapicelle lisättiin yritys- ja ravintolatili. Pilotin tavoitteena oli varmistaa, että järjestelmä toimii suunnitellulla tavalla, oikea määrä lounaita kirjautuu oikealle työntekijälle ja että tunnisteen rekisteröinti onnistuu. Pilotin osallistujilta kerättiin myös palautetta ja kehitysideoita liittyen järjestelmän ulkoasuun ja yleiseen toimintalogiikkaan.

Pilotin osallistujiksi valittiin aluksi seitsemän Wapicen työntekijää. Valitut työntekijät olivat sellaisia, jotka ovat olleet seuraamassa projektin edistymistä tai joiden arveltiin olevan kiinnostuneita lounaslippujärjestelmästä. Pilotti alkoi siten, että osallistujille lähetettiin sähköpostiviesti, jossa heitä pyydettiin osallistumaan pilottiin ja joka sisälsi ohjeet pilottia varten. Tämän jälkeen osallistujille jaettiin NFC-tunnisteet. Käyttäessään järjestelmää ensimmäisen kerran työntekijän tulee rekisteröidä tunniste. Rekisteröinti tapahtuu siten, että käyttäjä lukee tunnisteen kortinlukijalla ja kirjautuu järjestelmään omilla intratunnuksillaan vapaavalintaisessa järjestyksessä. Rekisteröinnin jälkeen työntekijä voi kirjautua järjestelmään kummalla tahansa tavalla. Rekisteröinnin jälkeen työntekijä ottaa lounaslippuja, kirjaa ottamansa määrän sovelluksella ja merkitsee ottamansa lounaslippujen määrän viralliseen listaan, joka lähetetään palkkatoimistolle, sekä pilottia varten tulostettuun listaan. Pilottivaiheessa järjestelmässä voi kirjata ainoastaan yhden tai kolme lippua kerralla, mutta ottaessaan enemmän lounaslippuja työntekijä voi kirjautua järjestelmään useaan kertaan. Tämä rajoitus varmistaa sen, että järjestelmään tulee paljon tapahtumia.

Järjestelmä toimi pilotin aikana hyvin: palvelin ei kaatunut kertaakaan ja pilotin lopussa tarkistetut lounasmäärät järjestelmän tietokannassa ja paperisessa tarkistuslistassa olivat samat. Käyttäjiltä tuli myös positiivista palautetta sovelluksen selkeydestä ja lounasmäärien kuittaamisen vaivattomuudesta. Ainoa hämmennystä aiheuttanut toiminto oli NFC-tunnisteen rekisteröinti ensimmäisellä käyttökerralla. Rekisteröinnin voi tehdä kahdella tavalla, eli ei ole väliä, luetaanko NFC-tunniste vai syötetäänkö kirjautumistiedot ensin. Joidenkin käyttäjien mielestä olisi selkeämpää, jos rekisteröinnin voisi tehdä vain yhdellä tavalla. Lisäksi moni pilottikäyttäjä mainitsi viiveestä, joka järjestelmässä ilmenee, kun se ottaa ensimmäistä kertaa yhteyttä palvelimelle. Järjestelmän palvelin sijaitsee virtuaalikoneella, joka menee lepotilaan oltuaan tietyn aikaa käyttämättä. Virtuaalikoneen aktivoituminen uudelleen kestää hetken, ja näin ollen se aiheuttaa viiveen. Järjestelmän toteuttajalla ei ollut oikeuksia palvelimen asetuksiin, joten asialle ei ollut mitään tehtävissä. Jos järjestelmä päätyy tuotantoon, sen palvelin voidaan konfiguroida niin, ettei viivettä esiinny.

7. ARVIOINTI

Projekti poikkesi tyypillisestä ohjelmistoprojektista siten, että siinä ei ollut varsinaista projektitiimiä, vaan koko projektin suunnittelu ja toteutus oli yhden ihmisen vastuulla. Projekti toteutettiin iteratiivisesti, ja projektin vaatimukset muuttuivat melko paljon projektin aikana. Projektissa ei ollut yksittäistä asiakasta tai tilaajaa, mikä oli haasteena vaatimusten hallinnalle.

7.1 Vaatimusten täyttyminen

Projektin aikana saatiin valmiiksi toimiva järjestelmä, jonka toiminnallisuus täyttää sille asetetut välttämättömät vaatimukset. Järjestelmä sisältää luvussa 3 määritellyt käyttöliittymät (web-käyttöliittymä, mobiilisovellus ja tuntikirjausjärjestelmään integroitu näkymä). Jokaisessa käyttöliittymässä käyttäjältä edellytetään tunnistautuminen, ja tunnistautumisen yhteydessä tarkistetaan käyttäjän rooli.

Mobiilisovelluksen ravintolan käyttöön tarkoitetussa näkymässä pystyy veloittamaan lounaita ja tarkastelemaan lokia. Luvussa 3.1.1 esitetyn vaatimuksen mukaisesti vain työntekijät, joiden tunniste on liitetty yritykseen, jolla on voimassa oleva sopimus ravintolan kanssa, pystyvät käyttämään tunnistetta maksuvälineenä kyseisessä ravintolassa. Kuten luvussa 3.1.2 on määritelty, sovelluksessa voidaan valita veloitettavan lounaan tyyppi, ja jos tyyppejä on määritelty vain yksi, valinta tapahtuu automaattisesti. Näin ollen kassatyöntekijältä ei vaadita lounaan veloittamisen yhteydessä ylimääräisiä toimia. Veloitetun lounaan peruminen sovelluksella ei vielä ole mahdollista, mutta tunnistetta ei voi lukea, jos edellisestä lukutapahtumasta on kulunut alle viisi sekuntia. Kaikki tapahtumat tallennetaan sovelluksen lokiin. Lokiviestien kääntäminen on toteutettu vain osittain: Lokiviestit kirjoitetaan järjestelmän kielivalinnan mukaisesti, mutta jos kieltä vaihdetaan sovelluksen ollessa käynnissä, jo kirjoitettuja viestejä ei käännetä. Muilta osin käyttöliittymätekstit käännetään järjestelmän kielen mukaisesti. Mobiilisovellukseen on toteutettu myös tunnisteiden rekisteröintinäkymä yrityksen käyttöön.

Web-käyttöliittymä on käytettävissä nykyaikaisilla selaimilla, kuten määritelty luvussa 3.1.3. Käyttöliittymässä on mahdollista lähettää ja hyväksyä sopimuspyyntöjä, tarkastella jo solmittuja sopimuksia, rekisteröidä tunnisteita työntekijöille, poistaa tunnisteen rekisteröinti, määritellä lounastyyppejä, saada raportteja ja hallinnoida järjestelmän ravintoloita ja yrityksiä. Ravintolan raportointinäkymä on vielä toteuttamatta, eikä yritys vielä saa ilmoituksia uusista sopimuspyynnöistä, mutta muilta osin web-käyttöliittymälle asetetut toiminnalliset vaatimukset täyttyvät.

Tuntikirjausjärjestelmään tehty näkymä on jo ollut käytössä Wapicen sisällä. Tähän asti tehtyjen havaintojen perusteella vaatimus raportoinnin tehostumisesta on täyttynyt, sillä jo nyt erityisesti PDF-tiedostojen lukutoiminto on nopeuttanut työntekijäkohtaisten lounasmäärien laskemista huomattavasti. Luvussa 3.1.4 esitetyn vaatimuksen mukaisesti näkymässä luettujen PDF-tiedostojen formaatti päätellään automaattisesti tiedoston etuliitteestä, jos etuliite on annettu. Muussa tapauksessa käyttäjä valitsee kullekin tiedostolle käytettävän formaatin. Formaatin tunnistamista tiedoston rakenteesta ei ole toteutettu. Luettujen lounasmäärien perusteella järjestelmä muodostaa Excel-raportin, joka noudattaa ennalta määriteltyä rakennetta.

Järjestelmän ulkoasu on vielä viimeistelemätön, mutta jo nyt se on tyyliltään yhtenäinen ja sinistä väriä on käytetty teemana niin mobiilisovelluksessa kuin web-käyttöliittymässäkin. Logoa lounaslippujärjestelmällä ei vielä ole. Järjestelmä on suunniteltu skaalautuvaksi, mutta suorituskyvyn osalta skaalautuvuutta ei ole testattu. Järjestelmä toimii yhdellä palvelimella, mikä saattaa käyttäjien määrän kasvaessa muodostua pullonkaulaksi. Web-käyttöliittymän taulukoita voi selata sivu kerrallaan, joten käyttöliittymä ei rajoita esimerkiksi ravintoloiden määrää. Käytettävyyden parantamiseksi taulukoihin pitäisi vielä sisällyttää haku- ja suodatustoiminnot, koska esimerkiksi tietyn ravintolan etsiminen tuhannen ravintolan listasta on työlästä.

7.2 Jatkokehitysideat

Lounaslippujärjestelmän ensimmäinen versio haluttiin pitää yksinkertaisena, joten siihen sisällytettiin vain järjestelmän toiminnan kannalta välttämätön toiminnallisuus. Tulevaisuudessa järjestelmä voisi tarjota lisäominaisuuksia, joiden käyttämisestä veloitettaisiin lisähinta.

Yksi tällainen lisäominaisuus voisi olla mahdollisuus lähettää laskuja järjestelmän kautta. Tätä ominaisuutta käyttäisivät ravintolat. Nykyisessä järjestelmässä ravintola saa lounaslippujärjestelmän kautta raportin ostetuista lounaista mutta hoitaa laskutuksen erillisellä järjestelmällä. Laskutusominaisuuden lisääminen lounaslippujärjestelmään poistaisi yhden työvaiheen laskutuksesta. Ominaisuuden toteuttamiseen voi liittyä laillisia näkökulmia, joita ei ole vielä tässä vaiheessa selvitetty.

Lounaslippujärjestelmä voisi lounaan ostamisen yhteydessä lähettää lounaan ostaneelle työntekijälle vahvistuksen lounaan ostamisesta sähköpostiviestillä. Vaihtoehtoisesti työntekijällä voisi olla jokin toinen tapa tarkistaa ostamiensa lounaiden tiedot. Tätä ominaisuutta ei ole vielä toteutettu, koska se edellyttäisi työntekijän kirjautumista järjestelmään, eikä järjestelmässä ole työntekijälle käyttäjäroolia.

Tulevaisuudessa raporttityyppejä tulee lisää. Tällä hetkellä lounaslippujärjestelmässä on yksi ennalta määritelty raporttityyppi ravintolalle ja yksi yritykselle. Kun selviää, millaisia raportteja järjestelmän käyttäjät tarvitsevat, raporttityyppejä voidaan lisätä järjes-

telmään. Lounaslippujärjestelmään voisi myös lisätä ominaisuuden, jolla yritykset ja ravintolat pystyisivät luomaan omia raporttityyppejä ja valitsemaan, mitä tietoja raportteihin sisällytetään. Raporttityypit pitäisi tällöin pystyä tallentamaan malleina, joita ravintolat ja yritykset pystyvät hyödyntämään myöhemmin luodessaan raportteja.

Lounaslippujärjestelmän web-käyttöliittymässä autentikointiin käytetään sessioita ja REST-rajapinnassa HTTP-basicia. Tulevaisuudessa molemmissa voisi käyttää JWT:a (Json Web Token). Etuna nykyiseen menetelmään olisi, että kaikki asiakassovellukset voisivat tunnistautua samalla menetelmällä. Sessioiden kanssa tulee myös ongelmia, jos palvelin hajautetaan usealle fyysiselle palvelimelle. JWT-menetelmässä käyttäjän oikeuksien perusteella muodostetaan kirjautumisen yhteydessä tunniste, jonka asiakassovellus lähettää jokaisen kirjautumista seuraavan pyynnön mukana palvelimelle. Tunniste koostuu pistein erotelluista base64-koodatuista otsikko-, kuorma- ja allekirjoituskentistä. Tämän jälkeen palvelin tarkastaa saapuvien pyyntöjen mukana tulevien tunnisteiden oikeellisuuden allekirjoituksen perusteella. Menetelmä on tilaton, joten se soveltuu hyvin käytettäväksi myös hajautetun palvelimen kanssa. (Jwt.io 2017)

7.3 Hylätyt ratkaisut

Lounaslippujärjestelmään suunniteltiin ominaisuutta, joka mahdollistaisi sopimusten muodostamisen ravintolan ja yksityisasiakkaan välillä. Ominaisuus kuitenkin monimutkaistaisi tietokannan rakennetta. Tarkemman pohdinnan jälkeen todettiin, että yksityisasiakkaat eivät käyttäisi järjestelmää, koska he eivät saa järjestelmää käyttämällä helpotusta verotukseen ja näin ollen järjestelmä ei toisi asiakkaalle lisäarvoa lähimaksamiseen verrattuna.

Joitakin NFC-sovelluksia on mahdollista käyttää ilman jatkuvaa internet-yhteyttä. Esimerkiksi matkakorttien yhteydessä rahaa tallennetaan kortille ja matkan veloittaminen tapahtuu offline-tilassa. Tiedot välitetään palvelimelle, kun kulkuneuvolla on internet-yhteys. Vastaava offline-toiminallisuus olisi käytännöllinen myös lounaslippujärjestelmässä. Toiminto ei kuitenkaan ole toteutettavissa luotettavasti nykyisellä tavalla määritellyssä järjestelmässä. Offline-toiminnallisuus edellyttäisi lounaiden tallentamista tunnisteelle etukäteen, mikä ei lounaslippujärjestelmässä ole mahdollista, koska ei voida tietää, missä ravintolassa työntekijä tulee syömään. Keskeinen kysymys on, miten tarkistettaisiin NFC-tunnisteeseen kytketyt sopimukset, jos internet-yhteyttä ei ole. Lounaslippujärjestelmässä offline-toiminnallisuus ei myöskään ole samalla tavalla välttämätön kuin matkakorteissa, koska lounaslippujärjestelmän lukijalaite on paikallaan, kun taas ajoneuvo voi mennä esimerkiksi tunneliin, jossa matkapuhelinverkon signaali on heikko.

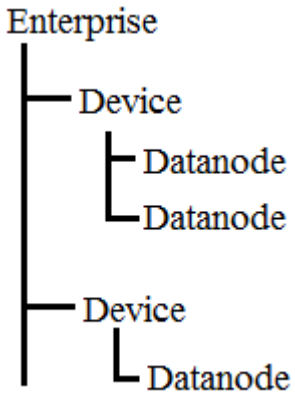
Tunnisteen rekisteröimistä helpottaisi, jos sen sijaan, että työntekijän tiedot syötetään järjestelmään manuaalisesti, järjestelmä näyttäisi listan yrityksen työntekijöistä, josta työntekijä voitaisiin valita. Tämä kuitenkin edellyttäisi järjestelmän yhdistämistä kaik-

kien asiakasyritysten omiin tietojärjestelmiin, eikä jokaisella yrityksellä välttämättä edes ole työntekijätietokantaa. Lisäksi tämä heikentäisi järjestelmän skaalautumista, sillä aina kun järjestelmään tulee uusia asiakasyrityksiä, järjestelmän ohjelmakoodiin täytyisi tehdä muutoksia. Yksi vaihtoehto olisi toteuttaa REST-rajapinta, jonka kautta yritys voisi tuoda järjestelmään työntekijöitä. Tämäkään ratkaisu ei ole ongelmaton. Tässä vaihtoehdossa yrityksen käyttämän tuntikirjausjärjestelmän tai muun vastaavan sovelluksen pitäisi käyttää REST-rajapintaa. Isoihin käytössä oleviin järjestelmiin on kuitenkin vaikea saada muutoksia.

7.4 Toteutuksen siirto IoT-Ticketiin

IoT-Ticket on Wapicen kehittämä ja ylläpitämä IoT (internet of Things) -alusta. IoT-Ticketiin voi lähettää dataa erilaisista laitteista. Datan visualisointiin ja raportointiin IoT-Ticket sisältää monipuoliset työkalut, ja yleisiin käyttötarkoituksiin on olemassa valmiita sovelluksia. Lisäksi IoT-Ticketin kautta on mahdollista hallita laitteita. Käyttäjä voi esimerkiksi kytkeä tiettyjä laitteita pois päältä, tai järjestelmä voi säätää laitteita jopa automaattisesti ennalta määriteltyjen ehtojen mukaisesti.

IoT-Ticketin tietomalli on puumainen, ja se on esitetty kuvassa 19. Puun juuressa on käyttäjä (Enterprise). Jokaisella käyttäjällä on tunnukset järjestelmään. Käyttäjä pystyy rekisteröimään laitteita (Device). Laitteelle generoidaan yksilöllinen id, jonka avulla se tunnistetaan. Laite voi olla esimerkiksi jokin mittauslaite, puhelin, tai yksittäinen sensori. Laite tallentaa dataa datasolmuihin (Datanode). Datasolmu tunnistetaan sen nimen ja polun perusteella. Jos laite esimerkiksi kuvaa moottoriin kytkettyä mittauslaitetta, sillä voi olla eri datasolmut moottorin lämpötilaa ja kierroslukua varten. Mittauspisteet tallennetaan datasolmujen alle. Jokaisella mittauspisteellä on jokin arvo ja aikaleima.



Kuva 19. *IoT-Ticketin tietomalli*

Lounaslippujärjestelmän kehityksen alkuvaiheessa lounasdata tallennettiin IoT-Ticketiin. Tällöin järjestelmän ylläpitäjällä oli IoT-Ticketiin tunnukset, ja laitteina toimivat puhelimet. Koska joka ravintolassa on yksi puhelin, ravintola pystyttiin tunnista-

maan laitteen id:n perusteella. Yrityksen tunnistamiseksi datanoden polkuun sisällytettiin yrityksen nimi. Tällä toteutustavalla ei kuitenkaan pystytä hyödyntämään IoT-Ticketin laajoja raportointimahdollisuuksia, vaan jotta yritykset ja ravintolat voisivat saada raportteja itselleen kuuluvasta datasta, jokaisella yrityksellä ja ravintolalla tulisi olla järjestelmään omat tunnukset. Näin ollen samaan dataan tulisi olla lukuoikeus sekä ravintolalla, jossa lounas on ostettu, että yrityksellä, jonka työntekijä on ostanut lounaan. Tällainen datan yhteisomistus ei ole IoT-Ticketissä mahdollista. Lounaslippujärjestelmän täytyy myös tallettaa tieto, minkä yritysten työntekijät voivat käydä syömässä kussakin ravintolassa ja mikä NFC-tunniste kuuluu millekin työntekijälle. Näin ollen dataa täytyy joka tapauksessa tallettaa erilliseen tietokantaan, joten IoT-Ticketin käyttämisestä päätettiin luopua myös lounasdatan osalta.

Kun lounaslippujärjestelmän toiminnallisuus oli jo toteutettu, järjestelmälle tuli vielä uusi vaatimus, jolla on suuri vaikutus koko järjestelmän rakenteeseen. Lounaslippujärjestelmä tulisi toteuttaa sovelluksena IoT-Ticketin päälle. Tämä hyödyttäisi monella tavalla niin lounaslippujärjestelmää kuin IoT-Ticketiäkin. Jos lounaslippujärjestelmä toteutetaan osana jo olemassa olevaa järjestelmää sen sijaan, että se toteutettaisiin erillisenä järjestelmänä, lounaslippujärjestelmän ylläpito helpottuu. Lounaslippujärjestelmä pystyisi myös hyödyntämään IoT-Ticketin raportointiominaisuuksia, mikä voisi auttaa sitä erottumaan kilpailevista sovelluksista. Lisäksi IoT-Ticketin markkinointi helpottuu, kun sen avulla toteutetaan uusia sovelluksia. Koska IoT-Ticketiä ei ole suunniteltu tällaiseen käyttötarkoitukseen, lounaslippujärjestelmän toteuttaminen IoT-Ticketin avulla edellyttäisi muutoksia IoT-Ticketiin. Lisäksi lounaslippujärjestelmä täytyisi tehdä lähes kokonaan uusiksi. Tämä voidaan kuitenkin nähdä IoT-Ticketin kannalta kehitysmahdollisuutena: kun muutokset on tehty IoT-Ticketiin, sitä voidaan käyttää joustavammin myös muiden sovellusten toteuttamiseen. Tässä vaiheessa projekti jää odottamaan yrityksen johdon päätöstä siitä, halutaanko näin isoja muutoksia tehdä.

8. YHTEENVETO

Projektin aikana määriteltiin, suunniteltiin ja toteutettiin NFC-teknologiaa hyödyntävä lounaslippujärjestelmä. Projektin tavoitteena oli toteuttaa toimiva PoC (Proof of Concept) -sovellus ja täten varmistua siitä, että määritelty järjestelmä on mahdollista toteuttaa valituilla teknologioilla. Projektin aikana saatiin valmiiksi sovellus, joka toteuttaa sille asetetut minimivaatimukset. Sovellus ei kuitenkaan ole vielä sellaisenaan valmis tuote, vaan sitä pitäisi kehittää niin, että se olisi ominaisuuksiltaan ja ulkoasultaan kilpailukykyinen markkinoilla olevien järjestelmien kanssa.

Jatkosuunnitelmat projektin osalta ovat vielä avoimia. Markkinoilla on olemassa hyviä kilpailevia sovelluksia, mutta tilaa on myös uudelle sovellukselle, jos se on ainakin joiltain osin kilpailijoita houkuttelevampi. Toinen projektin jatkumiseen vaikuttava tekijä on luvussa 7.4 mainittu toteutuksen siirto IoT-Ticketiin. Siirto edellyttäisi muutoksia IoT-Ticketiin, ja lounaslippujärjestelmä täytyisi toteuttaa monelta osin uusiksi, eikä ole varmaa, halutaanko näin isoja muutoksia lähteä tekemään. Toisaalta IoT-Ticketin monipuoliset raportointiominaisuudet voisivat auttaa järjestelmää erottautumaan sen kilpailijoista.

LÄHTEET

Android Developers. Android 3.0 APIs. [verkkosivu] Saatavissa: https://developer.android.com/about/versions/android-3.0.html [Viitattu 18.1.2017].

Browser market share (2016). December 2016. [verkkosivu] Saatavissa: https://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0 [Viitattu 18.1.2017].

Built in node (2016). A history of Node.js. [verkkosivu] Saatavissa: http://blog.builtinnode.com/post/a-history-of-node-js [Viitattu 25.1.2017].

Curran, K., Millar, A. & Mc Garvey, C. (2012). Near Field Communication. International Journal of Electrical and Computer Engineering (IJECE), Yogyakarta, Vol. 2(3), pp. 371.

Eazybreak. Helppokäyttöinen mobiilipalvelu työsuhde-eduille. [verkkosivu] Saatavissa: https://eazybreak.fi/ [Viitattu 17.1.2017].

Edenred. Suomen johtava työsuhde-etujen tarjoaja. [verkkosivu] Saatavissa: http://www.edenred.fi [Viitattu 17.1.2017].

Hunt, P., O'Shanessy, P., Smith, D. & Coatta, T. (2016). React: Facebook's Functional Turn on Writing JavaScript. ACM Queue, Vol. 14(4).

Jwt.io. JSON Web Token Introduction. [verkkosivu] Saatavissa: https://jwt.io/introduction/ [Viitattu 31.1.2017].

Node.js. [verkkosivu] Saatavissa: https://nodejs.org/en/ [Viitattu 21.3.2016].

React. [verkkosivu] Saatavissa: https://facebook.github.io/react/ [Viitattu 21.3.2017].

Roland, M., Langer, J. ja Scharinger, J. (2011). Security Vulnerabilities of the NDEF Signature Record Type. 2011 Third International Workshop on Near Field Communication. Hagenberg: IEEE, pp. 65–70.

Smartum (2017). [verkkosivu] Saatavissa: https://www.smartum.fi/fi [Viitattu 17.1.2017]

Statista (2016). Global mobile OS market share 2012-2016. [verkkosivu] Saatavissa: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ [Viitattu 8.11.2016].

Techotopia (2016). An Overview of the Android Architecture. [verkkosivu] Saatavissa: http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture [viitattu 9.11.2016].

Techradar (2008). A complete history of Android. [verkkosivu] Saatavissa: http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327 [viitattu 26.1.2017].

Tilkov, S. & Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs, IEEE Internet Computing, Vol. 14(6), pp. 80-83.

Verohallinto (2016). Verohallinnon päätös vuodelta 2017 toimitettavassa verotuksessa noudatettavista luontoisetujen laskentaperusteista. [verkkosivu] Saatavissa: http://www.vero.fi/fi-FI/Henkiloasiakkaat/Verokortti/Verohallinnon_paatos_vuodelta_2017_toimi(41508) [viitattu 17.1.2017].

Want, R. (2006). An introduction to RFID technology, IEEE Pervasive Computing, Vol. 5(1), pp. 25-33.