OULUN AMMATTIKORKEAKOULU

MOBIILI LOUNASPASSI

Visa Poropudas
Opinnäytetyö AMK
Kevät 2025
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä: Visa Poropudas
Opinnäytetyön otsikko: Mobiili lounaspassi
Työn ohjaajat: Tuula Harju, Olli Sulasalmi
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä: 40

Opinnäytetyö käsittelee mobiilin Lounaspassi-sovelluksen kehitystä ja toteutusta. Tavoitteena oli toimiva sähköinen lounaspassi, joka kattaa toimeksiantajan tarpeet ja vähentää tai jopa poistaa tarpeen pahvisille lounaspasseille toimeksiantajan ravintoloissa.

Opinnäytetyön tarkoituksena oli toteuttaa lounaspassi-sovellus ja sen toiminnot huomioiden käytettävyys ja turvallisuustekijät, kuten väärinkäytösten estäminen. Tämä saavutettiin sovelluksen sisäisen logiikan avulla sekä hyödyntämällä Google Firestore -dokumenttipankin sisäänrakennettua sääntömäärittelyä.

Itse työ toteutettiin Single Page Applikaationa (SPA) käyttäen React-ohjelmistokehystä ja Googlen Firebase-tunnistautumista ja Firestore-tallennustilaa tiedon tallentamiseen. Sovelluksessa on kolme erillistä näkymää: sisäänkirjautuminen ja käyttäjäksi rekisteröityminen, asiakkaan näkymä ja hallinnan käyttäjän näkymä.

Mobiilin lounaspassin käyttäminen toteutettiin QR-koodilla välitettävillä parametreilla, jotka luetaan ja parsitaan hallinnan käyttöliittymän QR-koodin lukijalla. Metodia hyödynnettiin asiakas- ja passitietojen välittämiseen hallintakäyttöliittymään. Hallintakäyttöliittymän aktiivinen istunto vaaditaan passin lounasetujen käyttämiseen väärinkäytösten ehkäisemiseksi.

Opinnäytetyön lopputuloksena saatiin toimeksiantajan vaatimusmäärittelyn mukainen sovellus, johon voidaan määritellä haluttu määrä leimattavia lounaita. Sovelluksella on runsaasti jatkokehitysmahdollisuuksia, kuten push-viestintä, mainostaminen, monibrändiympäristön kehittäminen tai toimeksiantajan valmiiden API-rajapintojen hyödyntäminen lähellä olevien lounaspaikkojen löytämiseksi ja lounaslistojen esittämiseksi.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Business Information Systems

Author: Visa Poropudas
Title of thesis: Mobile Lunch Pass
Supervisors: Tuula Harju, Olli Sulasalmi
Term and year when the thesis was submitted: Spring 2025
Number of pages: 40

The thesis examines the development and implementation of the mobile Lunch Pass application. The goal was to create a functional electronic lunch pass that meets the needs of the Client and reduces or even eliminates the need for cardboard lunch passes in the Client's restaurants.

The purpose of the thesis was to implement the lunch pass application and its functionalities, considering usability and security factors, such as preventing misuse. This was achieved not only through the application's internal logic but also by utilizing the built-in rule definitions in Google Firestore's document bank.

The project itself was implemented as a Single Page Application (SPA) using the React framework and Google's Firebase authentication and Firestore storage for data management. The application includes three separate views and functionalities: logging in and registering as a user, a customer view, and an admin view.

The use of the mobile lunch pass was implemented through parameters transferred via QR codes, which are read and parsed by the admin interface's QR code reader. This method was utilized to transfer customer and pass data to the admin interface. An active session in the admin interface is required to use the lunch pass benefits to prevent misuse.

The result of the thesis was an application that met the client's requirements, allowing the client to define the desired number of lunches to be stamped. The application has many further development opportunities, such as push messaging, advertising, developing a multi-brand environment, or utilizing the client's ready-made API interfaces to find nearby lunch places and display lunch menus.

SISÄLLYS

SANASTO

käsite	selite
AJAX	Asynchronous JavaScript and XML on joukko verkkosovellusteknologioita, jotka mahdollistavat verkkosovelluksen päivittämisen reaaliajassa ja pienissä osissa.
API	Application Programming Interface eli ohjelmistorajapinta. Ohjelmistorajapinta määrittelee funktiot, joiden avulla sovellukset keskustelevat keskenään.
CSS	Cascade Stylesheet on verkkosivuille kehitetty tyylisivu, joka määrittelee verkkosivun objektien tyylisäännöt.
DOM	Document Object Model on tapa kuvata rakenteisen dokumentin, kuten HTML-sivun rakenne oliopuuna.
HTML	Hypertext Markup Language on verkkosivujen rakenteen kuvaamiseen kehitetty merkkikieli, jolla kuvataan verkkosivun osia ja niiden roolia sivulla.
HTTP	Hypertext Transfer Protocol tiedonsiirtoprotokolla, jota käytetään tiedostojen ja hypertext-dokumenttien siirtämiseen Internet-verkon yli.
HTTPS	TLS/SSL-salattu HTTP-tiedonsiirtoprotokolla
JSON	JavaScript Object Notation on standardimuotoinen tekstipohjainen JavaScript-kielen objektisyntaksiin perustuva tiedonsiirtoformaatti
JSX	Reactin JavaScript-kielen laajennus
QR	QR-koodi, eli Quick Response -koodi, on kaksiulotteinen kuvakoodi, johon on koodattu informaatiota.

REST	Representional State Transfer on ohjelmistorajapinta-arkkitehtuuri
SASS	Syntactically Awesome Stylesheet on CSS:n laajennus, joka tukee muuttujien määrittelyä CSS:n sisällä.
SPA	Single Page Application on tapa toteuttaa verkkosovelluksia siten, että ne simuloivat paikallisesti asennettavaa sovellusta.
SSL	Secure Sockets Layer on suojausprotokolla, joka luo salatun yhteyden verkkopalvelimen ja selaimen välille
TLS	Transport Layer Security on salausprotokolla, jota käytetään kahden laitteen välisen tiedonsiirron suojaamiseen.
XML	Merkintäkielen standardi, joka määrittää tietojen merkintämuodon loogisella rakenteella.

1 JOHDANTO

Tämä opinnäytetyö käsittelee mobiilin lounaspassisovelluksen kehitystyötä. Toimeksiantaja on Osuuskauppa Arina ja tavoitteena on korvata toimeksiantajan ravintoloiden pahviset lounaspassit matkapuhelimessa toimivalla sähköisellä Lounaspassi-sovelluksella. Opinnäytetyö pohjautuu toimeksiantajan kanssa aikaisemmin Tietojenkäsittelyn toimeksianto -opintojaksolla tehtyyn toiminnalliseen määrittelyyn ja käyttöliittymäprototyyppiin.

Tarve lounaspassisovellukselle on sekä taloudellinen että ekologinen. Pahvisten lounaspassien painatus ja logistiset haasteet fyysisten lounaspassien toimituksessa aiheuttavat kustannuksia. Yhä harvempi kuluttaja haluaa lompakkoonsa pahvista lounaspassia, joka ei pysy tallessa ja kuormittaa luontoa. Mobiilisovelluksessa passi on aina siellä missä mobiililaitekin. Yhdessä sovelluksessa voidaan lisäksi säilyttää usean eri toimijan leimapasseja. Sähköinen lounaspassi mahdollistaa myös toimintoja, kuten viestintä ja dynaaminen markkinointi, joita pahvinen lounaspassi ei tarjoa.

Markkinoilla on joitakin mobiileja leimapassiratkaisuja, kuten EatPass ja Intellipocket. Lisäksi yksittäisillä ravintoloilla on joitakin omia leimapassisovelluksia. Näiden ongelmana on yleisesti käytettävyys sekä mahdollisuus sovelluksen hyväksikäyttöön. Opinnäytetyön tavoitteena on toteuttaa Osuuskauppa Arinan tarpeisiin suunniteltu mobiili Lounaspassi-sovellus, jonka kehityksessä on huomioitu helppokäyttöisyys, toiminnallisuus ja väärinkäytösten minimointi.

Opinnäytetyön tietoperustassa esitellään pohdintaa ja perusteet teknologiavalinnoille. Työn toteutukseen valittiin Single Page Application -mallilla toteutettu verkkosovellus, joka toteutettiin React-ohjelmistokehyksellä. Sovelluksen autentikointirajapinnaksi valittiin Google Firebase sähköpostikirjautuminen ja tiedon tallennusalustaksi Google Firestore -dokumenttipankki, jotka tarjosivat valmiit rajapinnat ja toiminnallisuudet käyttöoikeuksien hallintaan.

2 TEKNOLOGIAT

Työn tarkoituksena oli kehittää toimeksiantajan käyttöön mobiili lounaspassi-sovellus, joka on helposti asiakkaiden saatavilla ja toimii kaikissa mobiililaitteissa. Toteutustavoiksi oli tarjolla sovelluskaupasta ladattava mobiilisovellus tai HTML-pohjainen verkkopalveluna tarjottava Single Page Application (SPA). Ladattava, mobiililaitteeseen asennettava sovellus olisi vaatinut erillisten sovellusten julkaisemisen Applen App Storessa ja Googlen Google Playssa tai muussa vastaavassa sovelluskaupassa. Se ei olisi ollut järkevä ratkaisu, koska sovelluksen toiminnot eivät vaatineet natiivikirjastojen käyttämistä. Lisäksi todettiin, että sovelluskauppojen julkaisuprosessi on kankea ja erityisesti Applen tapauksessa vaatii testausympäristön Applen katselmoijille.

Kehittäjät ovat pitkään haaveilleet verkko- ja mobiilisovellusten toteuttamisesta natiivien työpöytäsovellusten tuntumalla ja käyttökokemuksella. Erilaisia ratkaisuja natiivin kokemuksen saavuttamiseksi on kokeiltu vaihtelevalla menestyksellä. Näitä kokeiluja edustavat muun muassa IFrame-kehykset, Java-sovelmat, Adobe Flash ja Microsoft Silverlight. Vaikka nämä ovat erilaisia teknologioita, niillä kaikilla on tavoiteltu työpöytäsovelluksen tuntuman tuomista verkkoselaimen ohueen, monialustaiseen ympäristöön. Yksisivuinen verkkosovellus eli SPA pyrkii samaan ilman selainlaajennuksia. (Scott 2016, luku 1.)

Single Page Application (SPA) on verkkosovellus, joka jäljittelee natiivin mobiilisovelluksen toimintaa päivittämällä dynaamisesti sovellusnäkymää palvelimelta tulevalla datalla. Erona perinteisiin verkkosivuihin on, että koko sivua ei ladata, vaan vain tarvittavat osat sovellusnäkymästä (Document Object Model, DOM) päivitetään. SPA ei myöskään lähtökohtaisesti tarjoa sivustolinkkejä, eli sovellus toimii sille määritellyssä osoitteessa, eikä alasivuille ole reititystä selaimen osoiteriviltä.

SPA-käyttöliittymässä selain lataa aluksi yhden HTML-dokumentin. Käyttäjät pysyvät sivustolla navigoidessaan samalla sivulla. Käyttäjän ollessa vuorovaikutuksessa sovelluksen kanssa JavaScript tuhoaa ja luo uuden käyttöliittymän tai sen

osan. Tämä voi tuntua siltä kuin hypättäisiin sivulta toiselle, mutta käyttäjä on itse asiassa edelleen samalla sivulla. (Banks & Porcello 2020, luku 4.)

SPA-konsepti vie web-kehityksen uudelle tasolle laajentamalla AJAXin (Asynchronous JavaScript and XML) sivutason manipulointitekniikoita koko sovellukseen. SPA:n luomisessa yleisesti käytetyt mallit ja käytännöt voivat johtaa sovellussuunnittelun tehokkuuteen, parempaan koodin ylläpidettävyyteen ja nopeampaan kehitykseen. (Scott 2016, luku 1.)

SPA-sovelluksen etuna on sen universaali saatavuus. Verkkosovellukset ovat nopein tapa tavoittaa ihmiset. Tämä edellyttää kuitenkin myös ymmärrystä siitä, miten selaimesta tehdään paras mahdollinen sovellussäilö. (Rady & Carter 2016, luku 3.) SPA-teknologioita ja ohjelmistokehyksiä oli tarjolla useita. Suunnittelussa lähdettiin aluksi tutkimaan, mitkä ovat ammattilaisten keskuudessa tämän hetken käytetyimmät ja suosituimmat alustat. Selvityksen perusteella suosituimpia SPA-teknologioita ovat esimerkiksi React, Angular ja Vue.js.

2.1 Teknologioiden esittely

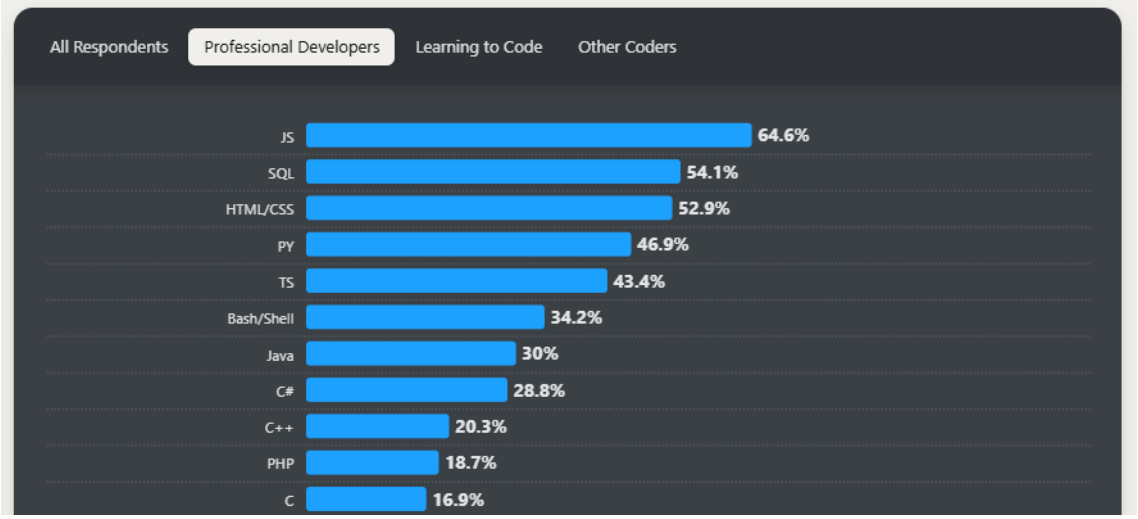
Teknologiavertailussa lähdettiin tämän hetken suosituimmista verkkosovelluksissa käytettävistä teknologioista. Eri lähteissä käytetyimpien teknologioiden joukossa ovat toistuvasti JavaScript, React, Angular, Vue ja Ember, minkä takia ne otettiin vertailuun mukaan.

Vertailun lähteenä käytettiin Stack Overflown vuotuisen kehittäjätutkimuksen aineistoa. Stack Overflow on laaja ja arvostettu kehittäjien yhteisö, joka tekee vuosittain käyttäjilleen laajan kyselytutkimuksen heidän käyttämistään teknologioista ja työkaluista. Vuoden 2024 kyselyyn vastasi 65 437 käyttäjää, joista 76,7 % ilmoitti olevansa ammatiltaan ohjelmistokehittäjiä (Stack Overflow 2024 Developer Survey).

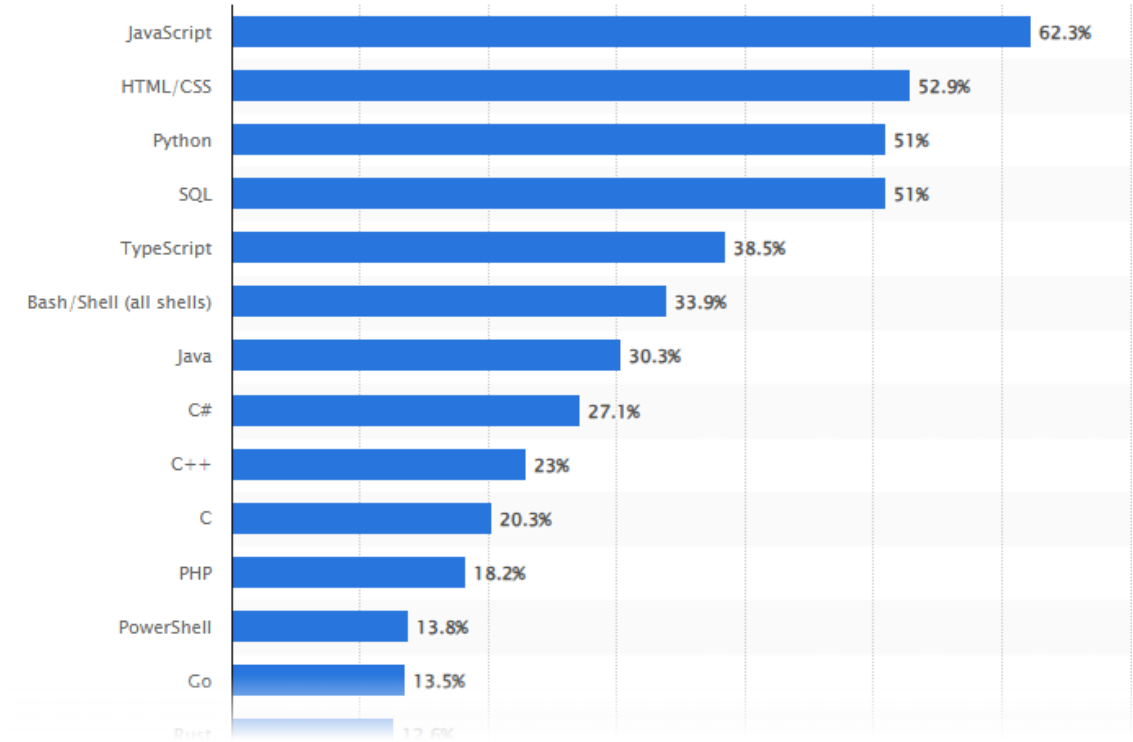
Stack Overflown tutkimusta tukemaan otettiin vertailuun Statistan vuoden 2024 tilastot. Statista on maailmanlaajuinen kaupallinen data- ja liiketoimintatiedon alusta, joka on julkaissut sivustollaan useita vapaasti käytössä olevia tilastoja muun muassa ohjelmistokehityksen trendeistä. Lisäksi työssä käytettiin Rising

Stars of JavaScriptin vuoden 2024 tilastoa käytetyimmistä ohjelmointikielistä ja ohjelmistokehyksistä. Rising Stars on Michael Rambeaun projekti, joka perustuu kehittäjäyhteisö GitHubiin annettuihin arvioihin kehittäjien projekteista. Rising Starsin tilasto ei siis perustu tutkimukseen, mutta se antaa suuntaa kehittäjäyhteisön tämän hetken trendeistä. Sovelluskehityksen valtavassa ohjelmistoviidakossa tilastot antoivat suuntaa sille, mitkä ovat varteenotettavia teknologiavalintoja. Vertailun tuloksista valittiin potentiaalisimmat teknologiat, jotka esitellään jäljempänä. Suuri merkitys tämän työn lopullisissa teknologiavalinnoissa oli myös toteuttajan omalla aikaisemmalla kokemuksella ja osaamisella.

Stack Overflown vuoden 2024 Developer Surveyn perusteella ammattilaisten keskuudessa eniten käytetyt teknologiat olivat JavaScript, SQL, HTML ja CSS, Python ja TypeScript (Kuva 1). Statistan vuoden 2024 tilasto tukee hyvin Stack Overflown kyselytutkimuksen tuloksia (Kuva 2).

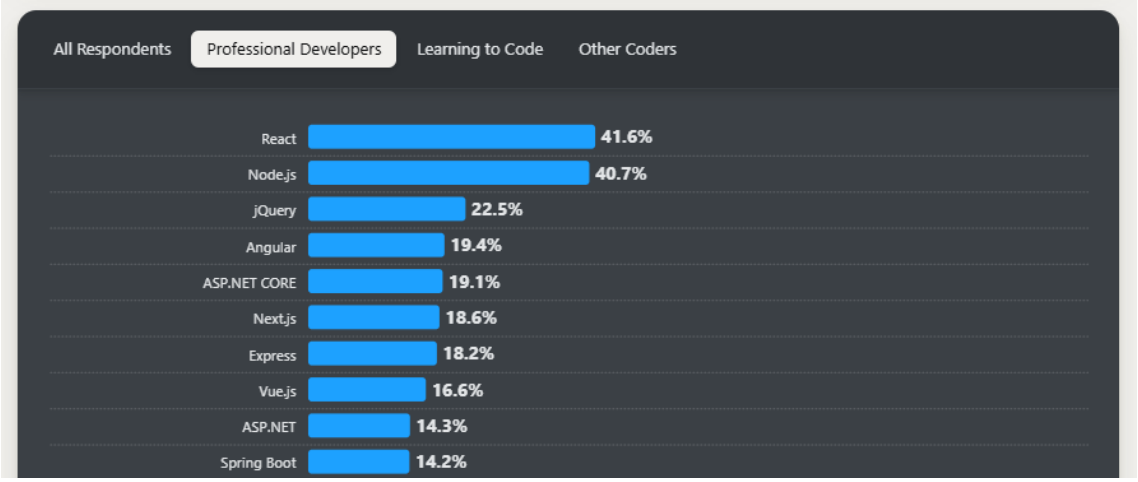


Kuva 1. Ammattilaisten käyttämät teknologiat - ohjelmointikielet (Stack Overflow 2024 Developer Survey)

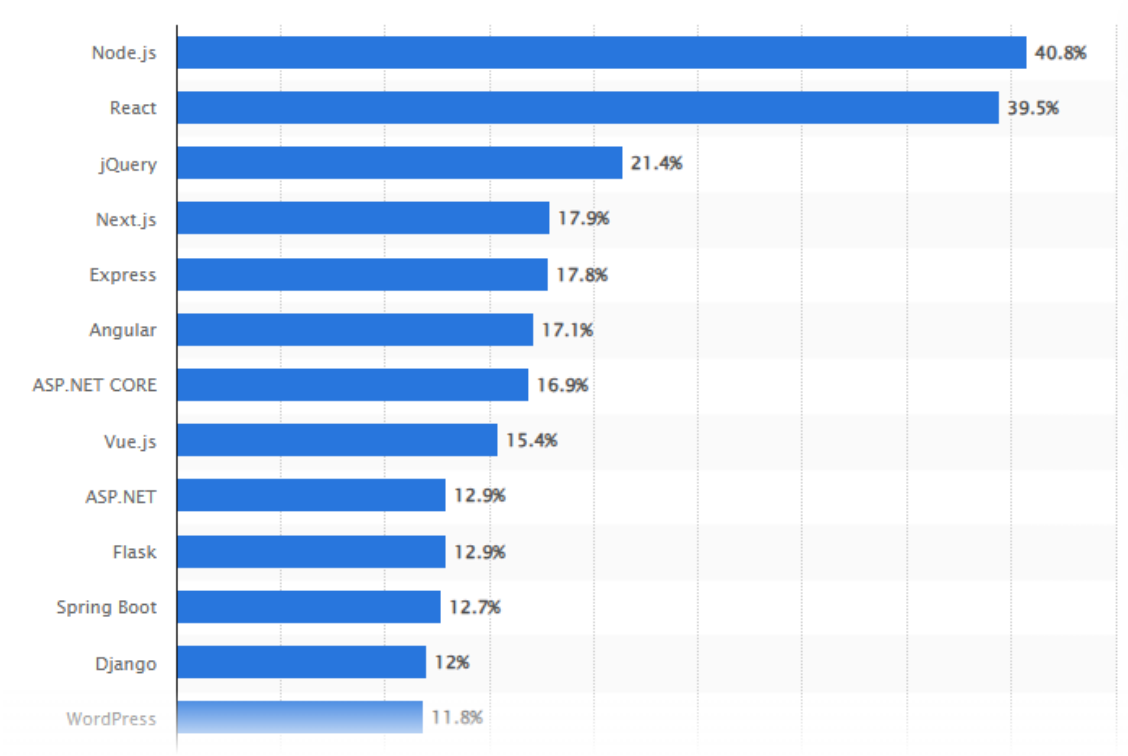


Kuva 2. Käytetyimmät ohjelmointikielet 2024 (Statista)

Vastaavasti ohjelmistokehyksistä (framework) käytetyimpiä Stack Overflown kyselyn mukaan olivat React, Node.js, jQuery, ja Angular (Kuva 3), jotka ovat kaikki JavaScript-pohjaisia teknologioita. Statistan vuoden 2024 tilastojen mukaan React oli selkeästi suosituin ohjelmistokehys perässään jQuery Next.js, Express ja Angular (Kuva 4).



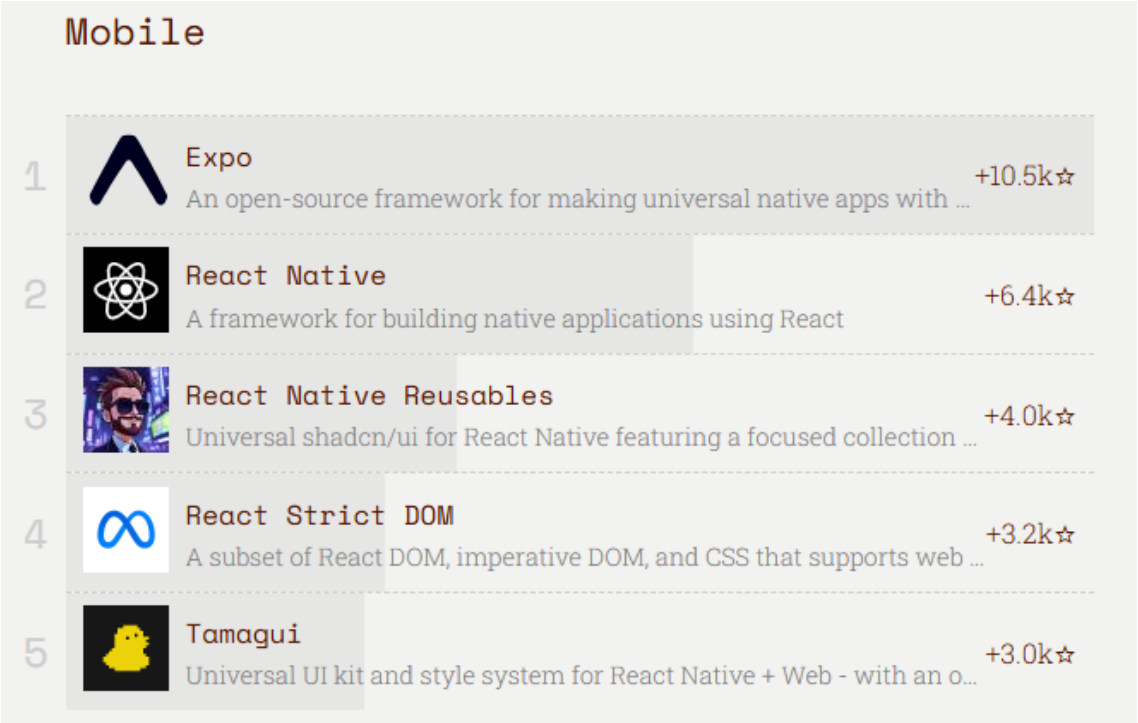
Kuva 3. Ammattilaisten käyttämät teknologiat - ohjelmistokehykset (Stack Overflow 2024 Developer Survey)



Kuva 4. Käytetyimmät ohjelmistokehykset 2024 (Statista)

Rising Stars of JavaScript -sivuston tilaston mukaan käytetyimmät ohjelmistokehykset vuonna 2024 olivat htmx, React, Svelte, Vue.js ja Angular, joista htmx ja React olivat selkeästi suosituimmat (Kuva 5). Mobiilikehityksessä käytetyimmät teknologiat Rising Stars of JavaScriptin tilaston perusteella olivat Expo, React Native, React Native Reusables ja React Strict DOM (Kuva 6).

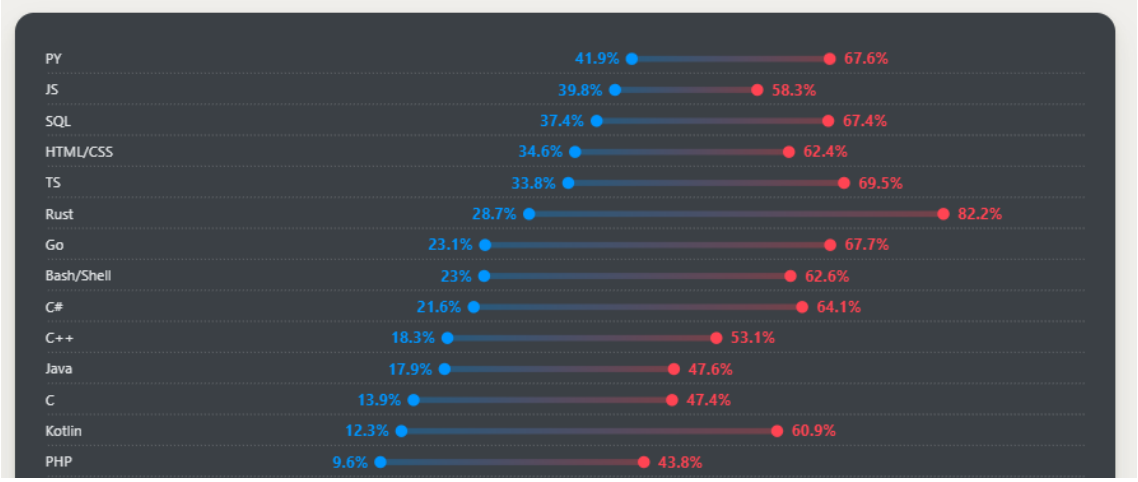
Kuva 5. Käytetyimmät ohjelmistokehykset (Rising Stars of JavaScript 2024)



Kuva 6. Käytetyimmät mobiiliteknologiat (Rising Stars of JavaScript 2024)

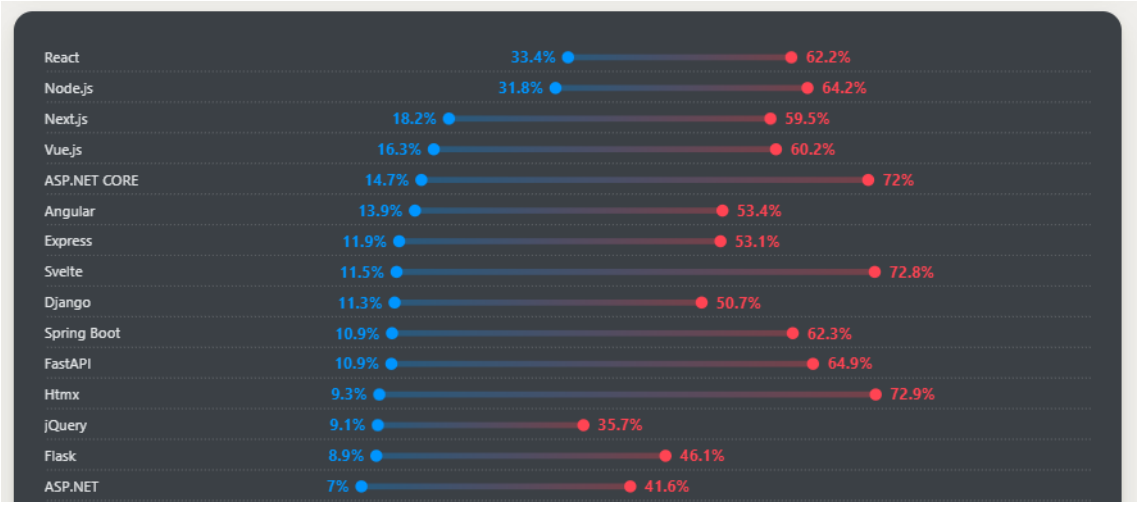
Stack Overflown käyttäjäkyselyn mukaan JavaScript, Python ja SQL ovat kaikki erittäin haluttuja ja ihailtuja ohjelmointikieliä kehittäjien keskuudessa. Tilastollisesti Rust on ihailluin ohjelmointikieli 83 %:n tuloksella. Tästä huolimatta Rustia

ilmoittaa käyttävänsä vain 12,6 % ammattilaisista. (Stack Overflow 2024 Developer Survey). Rust on kuitenkin järjestelmäohjelmoinnissa käytetty ohjelmointikieli, kuten C ja C++ (Goulding 2020), joten se ei sovellu tämän opinnäytetyön piiriin. Verkkosovelluksissa käytettävistä ohjelmointikielistä selkeästi eniten käytetty on JavaScript ja seuraavat Python ja TypeScript (Stack Overflow 2024 Developer Survey; Statista 2024).



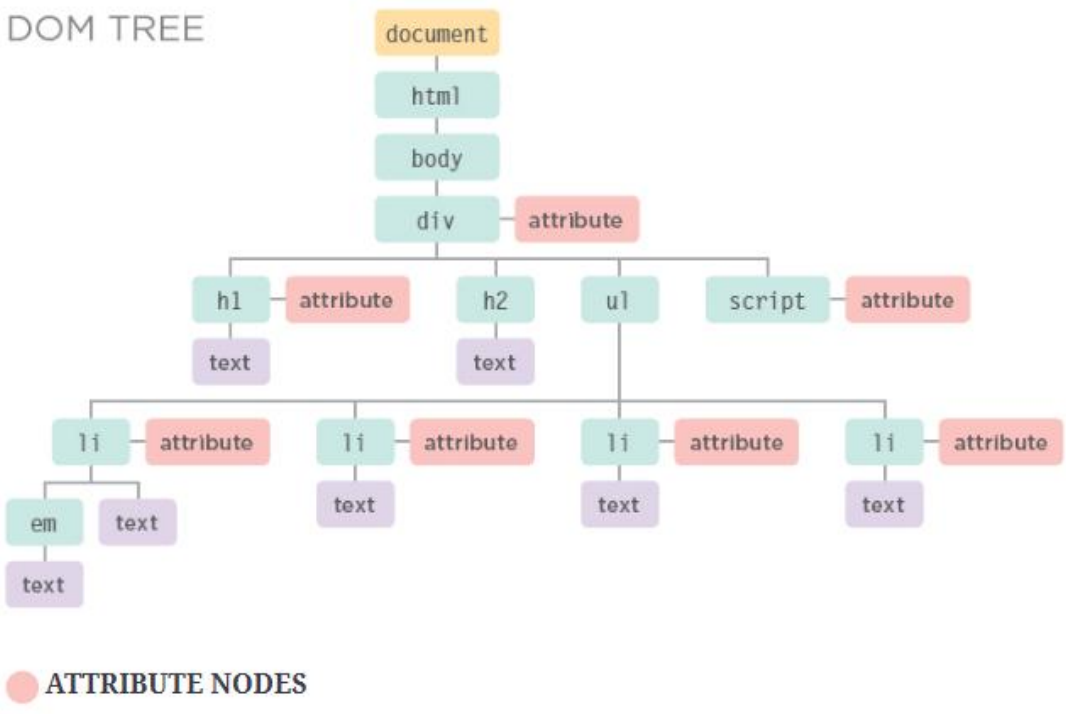
Kuva 7. Käyttäjien innostuneisuus ja arvostus ohjelmointikieliä kohtaan (Stack Overflow 2024 Developer Survey)

Tämän hetken suosituimpien käyttöliittymätason SPA-ohjelmistokehysten listauksissa toistuvat yleisimmin React, Angular, Vue.js, jQuery ja Ember.js. JavaScript on ylivoimaisesti suosituin kieli verkkosovelluskehityksessä. Täytyy kuitenkin muistaa, että kaikilla teknologioilla on kannattajansa ja kuten Kuva 7 ja Kuva 8 osoittavat, tyytyväisyys ja innostus eri ohjelmointikieliin ja ohjelmistokehyksiin vaihtelee suuresti kehittäjän mukaan. Kehittäjä ei myöskään aina pääse vaikuttamaan siihen, millä kielellä tai mihin ympäristöön ohjelmistoa kehitetään, vaan siihen voi vaikuttaa esimerkiksi vaatimusmäärittely.



Kuva 8. Käyttäjien innostuneisuus ja arvostus ohjelmistokehyksiä kohtaan (Stack Overflow 2024 Developer Survey)

Suurin osa nykyaikaisista ohjelmistokehyksistä (framework) toteuttavat virtuaalisen DOM-puun, jota ne vertaavat selaimen DOM-puuhun (Kuva 9). Esimerkiksi React päivittää selaimen DOM:ia vain niiltä osin, kuin se on muuttunut. Tämä lisää ohjelmiston nopeutta ja vähentää tarvetta tiedon siirrolle sivun päivityksissä.



Kuva 9. DOM-puu (Duckett 2014, luku 5)

2.1.1 React

React on Metan kehittämä avoimen lähdekoodin JavaScript-ohjelmistokehys, jota käyttävät Metan omien palveluiden, kuten Facebookin ja Instagramin lisäksi muun muassa Netflix, Dropbox, Airbnb ja PayPal.

Reactin kehitti alun perin Facebookin ohjelmistoinsinööri Jordan Walke ja se esiteltiin yleisölle 2011. Vuonna 2013 Reactista tuli avoimen lähdekoodin ohjelmistokehys. Tämän jälkeen React on kehittynyt paljon. 2015 ja 2016 esiteltiin muun muassa React Router, Redux ja Mobx, jotka helpottivat sovellusten reititystä ja tilanhallintaa. Vuonna 2019 esiteltiin Koukut (Hooks), jotka olivat kokonaan uusi tapa hallita komponenttien loogisia tiloja. (Banks & Porcello 2020, luku 1.)

React tarjoaa näkymäkerroksen datan renderöintiin HTML-muodossa interaktiivisten käyttöliittymäkomponenttien luomiseksi (Singh & Tanna 2018, React). React on siis suunniteltu päivittämään selaimen DOM:ia kehittäjän puolesta (Banks & Porcello 2020, luku 4).

Reactin komponenttipohjainen arkkitehtuuri mahdollistaa käyttöliittymän jakamisen uudelleen käytettäviin komponentteihin, mikä helpottaa ohjelmistokoodin ylläpitämistä ja monimutkaisten käyttöliittymien luomista. Reactin etuna on myös sen käyttämä virtuaalinen DOM, joka tarjoaa nopean ja responsiivisen vasteen käyttöliittymäpäivityksiin. Reactilla on laaja ekosysteemi, jossa on tarjolla paljon kolmannen osapuolen komponentteja.

React soveltuu hyvin niin staattisten verkkosivustojen, kuin verkko- ja mobiilisovellusten alustaksi. Reactin haittapuolina voidaan pitää sen jyrkkää oppimiskäyrää sekä ohjelmiston kokoa.

2.1.2 Angular

Angular on Googlen kehittämä ja ylläpitämä avoimen lähdekoodin JavaScript-ohjelmistokehys, jota käyttävät Googlen lisäksi useat isot toimijat kuten Microsoft 365, PayPal ja Deutsche Bank.

Angular on erittäin tehokas, yritystason kehys, jonka avulla voidaan rakentaa kaikenkokoisia, skaalautuvia ja helposti ylläpidettäviä sovelluksia. Sillä on eloisa yhteisö ja sitä päivitetään jatkuvasti uusilla ominaisuuksilla tuottavuuden parantamiseksi. (Garcia 2023, luku 1.)

Angularin etuja ovat ylläpidettävyys, tietoturva ja skaalautuvuus, minkä vuoksi Angularia käyttävät useat suuret yritykset. Angularin ekosysteemi on laaja ja siihen on saatavilla runsaasti kolmannen osapuolen komponentteja. Angularin haittapuolia ovat sen monimutkaisuus, korkea oppimiskynnys ja sen suhteellinen hitaus muihin vastaaviin ohjelmistokehyksiin nähden.

Ohjelmistokehyksenä Angular todettiin turhan raskaaksi ja monimutkainen tämän opinnäytetyön toteutukseen. Angularin kaksisuuntaisen datan sidonnan edut eivät vaikuta tällaisessa sovelluksessa ja vastineena saadaan turhan raskas ohjelmistopohja ja suhteellisen hidas ohjelmisto.

2.1.3 Vue.js

Vue.js on avoimen lähdekoodin komponenttipohjainen JavaScript-ekosysteemi, joka on suunniteltu joustavaksi ja tehokkaaksi. Vue.js:n ekosysteemi ei ole yhtä kattava, kuin esimerkiksi Reactilla ja Angularilla. Vue.js:n etuna on sen edellä mainittuja teknologioita matalampi oppimiskäyrä. Vue.js on käytössä esimerkiksi Adoben, GitLabin ja Trivagon sovelluksissa.

Vue.js julkaistiin alun perin vuonna 2014, ja se saavuttanut nopeasti kehittäjien suosion. Vue.js on suosittu kehys kehittäjäyhteisössä helppokäyttöisyytensä ja joustavuutensa ansiosta (Shavin 2023, luku 1.)

Vue.js on mielenkiintoinen ja yksi varteenotettavista vaihtoehdoista tämän opinnäytetyön ohjelmistokehykseksi. Vaihtoehtoja arvioitaessa Vue.js:n valintaa heikentää se, ettei se ole ennestään kehittäjälle tuttu, sen ekosysteemi on suhteellisen pieni ja esimerkkikoodia on saatavilla vähemmän, kuin vaikkapa Reactilla.

2.1.4 jQuery

jQuery on ensimmäisiä JavaScript-käyttöliittymäohjelmistokehyksiä. Se on helppo oppia ja osin sen takia se on myös laajasti käytetty. Sen huono puoli on sen raskaus ja hitaus, minkä lisäksi API on monilta osin vanhentunut. Monet isot toimijat käyttävät yhä jQuerya osana teknologiapinoaan, mutta modernien ohjelmistokehysten etuna jQueryyn nähden on niiden skaalautuvuus, sisäänrakennettu tilanhallinta ja komponenttipohjainen arkkitehtuuri. JQuery on omimmillaan pienissä ja yksinkertaisissa tehtävissä, kuten käyttöliittymän animoinnissa ja DOM:in manipuloinnissa.

Huhtikuussa 2013 jQuery julkaisi version 2.0 tarkoituksenaan tarkastella webin tulevaisuutta sen menneisyyden sijaan, erityisesti selaimen näkökulmasta. Siihen asti jQuery tuki kaikkia uusimpia Chromen, Firefoxin, Safarin, Operan ja Internet Explorerin versioita versiosta 6 alkaen. Version 2.0 myötä jQueryn tiimi päätti jättää vanhemmat Internet Explorer 6-, 7- ja 8-selaimet taakseen ja keskittyä webiin sellaisena kuin se tulee olemaan, ei sellaisena kuin se oli. Tämän päätöksen myötä jQuerysta poistettiin paljon koodia, joka oli luotu ratkaisemaan selainten yhteensopimattomuusongelmia ja puuttuvia ominaisuuksia vanhoissa selaimissa. Koodin vähentäminen johti arviolta 12 % pienempään ja nopeampaan koodipohjaan. (Bibeault, Katz, De Rosa & Methvin 2015, luku 1.)

Nykyinen jQueryn versio on 3.7.1. Vanhempien jQuery-versioiden käyttäjille on saatavana jQueryn verkkosivuilta jQuery Migrate plugin, joka mahdollistaa vanhan koodin ajamisen myös uudemmilla jQueryn versioilla.

Ohjelmistokehyksenä jQuery ei vastaa tämän opinnäytetyön tarpeisiin niin tilanhallinnan, skaalautuvuuden kuin komponenttien uudelleenkäytettävyytensäkään osalta.

2.1.5 Ember.js

Ember on avoimen lähdekoodin komponenttipalvelumalliin pohjautuva ohjelmistokehys (framework). Ember on kehitetty antamaan kehittäjille mahdollisuus

skaalautuvien SPA-sovellusten kehittämiseen. Emberiä käyttävät esimerkiksi Apple ja LinkedIn. Ember on esitelty vuonna 2011, eli se on suhteellisen uusi ohjelmistokehys. Sen käyttäjäyhteisö on varsin pieni ja sitä pidetään yleisesti vaikeana oppia ja hallita.

Emberissä ei usein käytetä JavaScript-vakio-objekteja. Emberin objektimalli perustuu JavaScript-objekteihin lisäten niihin tärkeitä ominaisuuksia, kuten observerit, miksaukset, lasketut ominaisuudet ja alustajat. Monet näistä ominaisuuksista on linjattu uuden ECMAScript-standardin mukaisiksi. (Hanchett 2016, luku 2). ECMAScript on tieto- ja viestintäjärjestelmien standardointiin keskittyvän Ecma Internationalin standardoima skriptikielien, kuten JavaScript, standardi, joka määrittelee muun muassa kielen syntaksin ja käyttäytymisen.

Ember.js ei sovellu tämän opinnäytetyön ohjelmistokehykseksi sen hankalan opittavuuden, suppean ekosysteemin ja vähäisen dokumentaation takia. Opinnäytetyön teknologiavalinnat pohjautuvat toteuttajalle pitkälti tuttuihin teknologioihin, jolloin mahdollisuus toteutuksen estäviin keskeytyksiin saadaan minimoitua.

2.1.6 JavaScript ja JSX

JavaScript on alun perin Netscapen kehittämä heikosti tyypitetty, tulkattava, oliopohjainen skriptikieli, jota käytetään pääasiassa verkkoympäristössä. JavaScript perustuu löyhästi C-ohjelmointikieleen ja se on saanut vaikutteita Java-kielestä muun muassa syntaksista ja standardikirjastosta, mutta on oma ohjelmointikielensä. JavaScriptillä ei itse asiassa ole juurikaan tekemistä ohjelmointikielen Java kanssa, vaan samankaltainen nimi syntyi markkinointinäkökohdista. Kun JavaScript esiteltiin, Java-kieltä markkinoitiin voimakkaasti ja se oli kasvattanut suosiotaan. Markkinoinnillisista syistä kielelle annettiin nimeksi JavaScript (Haverbeke 2025, What is JavaScript?).

JavaScriptin vahvuus ja samalla heikkous on sen syntaksin monimuotoisuus, mikä tekee siitä helposti opittavan, mutta myös hyvin virhealttiin ohjelmointikielen. Tämä tekee virheiden korjaamisesta ja koodin tulkinnasta paikoin haastavaa. Kuten Marijn Haverbeke (2025) toteaa: ”Se hyväksyi lähes kaiken kirjoittamani,

mutta tulkitsi sen täysin eri tavalla kuin tarkoitin. Tämä johtui paljon siitä, etten tiennyt mitä tein, mutta tässä on todellinen ongelma: JavaScript on naurettavan liberaali siinä, mitä se sallii.” (Haverbeke 2025, What is JavaScript).

Node.js on alustariippumaton JavaScript-suoritusympäristö, jonka avulla JavaScript-koodia voidaan suorittaa verkkoselaimen ulkopuolella. Node.js on mahdollistanut JavaScriptin käyttämisen myös palvelinpuolen ohjelmointikielenä asiakaspuolen ohessa.

JavaScript on kokenut paljon muutoksia julkaisunsa jälkeen vuonna 1995. Aluksi sitä käytettiin interaktiivisten elementtien lisäämiseen verkkosivuille. Esimerkkejä tällaisista vuorovaikutuksista olivat tapahtumakuuntelijat, hiiren osoittimen tilat ja lomakkeiden validointi. Nykyaikaiset JavaScript-ominaisuudet, kuten nuolifunktiot, luokat, arvojen hajauttaminen, purkaminen ja moduulit, yksinkertaistavat koodausta. Node.js:n myötä JavaScriptistä on tullut todellinen ohjelmistokieli, jota käytetään täysimittaisten sovellusten rakentamiseen. (Banks & Porcello 2020, luku 2; Kereki 2025 luku 1.)

JavaScript on ylivoimaisesti käytetyin ohjelmointikieli moderneissa verkkosovelluksissa. Stack Overflown käyttäjätutkimuksen mukaan 64,6 % ammattilaisista ilmoittaa käyttävänsä JavaScriptiä, kun toiseksi suosituinta Pythonia käyttää vajaa 47 % ammattilaisista (Stack Overflow 2024).

React tarjoaa eri vaihtoehtoja käyttöliittymän toteuttamiseen ja natiivin HTML:n yhdistämiseen koodin sekaan. Reactin kieli, JSX, on laajennettu JavaScript-syntaksista ja se on sekoitus JavaScriptia ja HTML:aa. Tai toisin sanoen: JSX:n avulla voidaan kirjoittaa HTML-tyyppistä koodia JavaScript-koodin sekaan (Kumar 2024, luku 2).

2.1.7 Firebase ja Firestore

Firebase on Googlen tarjoama kokoelma pilvipalveluja sovelluskehittäjille. Firebase tarjoaa muun muassa rekisteröitymis- ja tunnistautumisrajapinnan sekä Firestore-dokumenttitietokannan.

Firebasen ja Firestoren etuna on niiden skaalautuvuus ja helppo integroitavuus. Firebasen API tarjoaa valmiit rajapinnat käyttäjän rekisteröitymiseen (createUserWithEmailAndPassword), kirjautumiseen (signInWithEmailAndPassword) salasanan palauttamiseen (sendPasswordResetEmail) ja käyttäjän poistamiseen (deleteUser). Lisäksi Firebase mahdollistaa käyttäjän kirjautumisen esimerkiksi Facebook- tai Google-tunnuksilla.

Firebase Admin SDK tukee mukautettujen attribuuttien määrittämistä tunnuksen avulla. Mukautetut attribuutit mahdollistavat eri käyttöoikeustasojen määrittämisen, mukaan lukien roolipohjaisen sovelluksen hallinnan, joka pakotetaan sovelluksen suojaussäännöissä. (Singh & Tanna 2018, User profile and access management.)

Firestore on NoSQL-dokumenttitietokanta. Se tarjoaa perinteisen SQL-tietokannan ominaisuudet, mutta sen tapa kuvata kantaobjektien suhteita poikkeaa perinteisestä SQL-tietokannasta. Pilvipalveluna Firebase tarjoaa valmiiksi suojatun arkkitehtuurin sekä joukon työkaluja sovelluksen tietoturvan hallintaan (Singh & Tanna 2018, Key features of Google Cloud Storage, How secure is your Firebase).

2.1.8 QR

QR-koodi, eli Quick Response -koodi, on kaksiulotteinen kuvakoodi, johon on koodattu informaatiota. QR-koodi kehitettiin alun perin liukuhihnateollisuuden nopeaksi tuotantoseurantavälineeksi. 177 pikselin QR-koodi voi sisältää 1852 merkkiä. Teoriassa QR-koodi voi sisältää 7089 numeroa tai 4296 kirjainta (Pihkala 2018, 11). QR-koodin käytetään moniin eri tarkoituksiin, joista yleisin on verkkosivulinkkien jakaminen. Useimmat matkapuhelinten kamerasovellukset tunnistavat ja tulkitsevat automaattisesti QR-koodien sisällön.

Sovelluksen tietoturvan lisäämiseksi QR-koodi voidaan suojata salasanalla (Pihkala 2018, 27). Tällöin vain kyseisen koodin lukemiseen tarkoitettu laite voi purkaa koodin salauksen.

2.1.9 SASS

Verkkosovelluksen perusta on HTML, joka määrittelee sovelluksen rakenteen. Pelkkä rakenne ei tee sovelluksesta kuitenkaan miellyttävää ja käyttäjäystävällistä. Verkkosivustoa tai -sovellusta verrataankin usein taloon, jonka perusrakenne on HTML, mutta talo ei ole koti ilman viimeistelyjä, kuten listoja, maalia ja lattiaa. Sovelluksen viimeistely, tyylimääritykset, värit, koristelu, tehdään CSS:lla. (McFeddries 2024, luku 1.)

Syntactically Awesome Stylesheet (SASS) on CSS:n laajennus, joka tarjoaa mahdollisuuden muuttujien määrittelemiseen ja niiden käyttämiseen osana tyylimäärittelyä. Tämä taas helpottaa tyylimuutosten tekemistä ja vähentää tyylivirheiden todennäköisyyttä. SASS-tyylitiedosto ei toimi sellaisenaan verkkosivulla, vaan se vaatii sovellukseen asennetun esiprosessorin.

SASS laajentaa CSS:aa lisäämällä uusia ominaisuuksia ja syntaksia, jotka keskittyvät helpottamaan kehittäjän elämää, kuten Hampton Lintorn Catlin ja Michael Lintorn Catlin (2016) toteavat. SASS ei korvaa CSS:aa, vaan se auttaa kehittäjää kirjoittamaan parempia, selkeitä ja semanttisia CSS-tyylitiedostoja. (Lintorn Catlin & Lintorn Catlin 2016, Welcome!.)

2.2 Toteutukseen valitut teknologiat

Käyttöliittymäkerroksen toteutukseen valittiin React.js. Reactin etuna on sen sopivuus eri alustoille, komponenttien uudelleenkäytettävyys ja koukut (hooks), jotka mahdollistavat käyttöliittymän dynaamisen päivittämisen ja tilojen hallinnan. Lisäksi React ja JavaScript olivat toteuttajalle ennestään tuttuja teknologioita, jolloin oppimiskäyrä ei ollut niin jyrkkä. Lisäksi Internetistä löytyy paljon esimerkkejä ja kattava dokumentaatio Reactin syntaksista.

Kirjautuminen toteutettiin käyttäen Googlen Firebasea, joka tarjoaa valmiit kirjastot ja tuen erilaisille kirjautumismenetelmille, sekä käyttäjätunnusten turvallisen säilytyksen. Tietokantaratkaisuna käytettiin Googlen Firestorea sen helpon integroitavuuden takia. Firestore tarjoaa esimerkiksi sisään rakennetun säännöstön

käyttöoikeuksien määrittämiseen. Tämä puolestaan antaa mahdollisuuden toteuttaa sovelluksen sisäinen logiikka välikerroksessa ilman erillistä backend-ratkaisua. Firestoren eduksi katsottiin myös tallennettavan tiedon suhteellisen vähäinen määrä ja tallennetun tiedon mallintaminen suoraan JSON-muodossa.

QR-koodi valittiin tiedonsiirtomuodoksi käyttäjän ja hallinnan käyttäjän laitteiden välillä. QR-koodi on tunnettu ja maailmanlaajuisesti tuettu metodi tiedon siirtoon. Reactille löytyy valmiita ohjelmakirjastoja QR-koodin luomiseen ja lukemiseen, joten oman rajapinnan luominen on helppoa. Satunnainen käyttäjä voi lukea QR-koodin sisällön matkapuhelimen kameralla, mutta ilman Lounaspassi-sovellusta luetulla tiedolla ei ole mitään käyttöä.

Tyylitiedostoformaatiksi valittiin SASS sen tarjoaman helpon muokattavuuden takia. SASS tukee tyylitiedoston muuttujia, jolloin usein toistuvat arvot voidaan korvata muuttujilla, mikä helpottaa tyylimuutosten tekemistä ja vähentää tyylivirheiden todennäköisyyttä.

3 TYÖN TOTEUTUS

Tässä luvussa esitellään sovelluksen käyttötapaus ja työn toteutus. Työn toteutus -osiossa kerrotaan sovelluksen käyttöliittymän rakenne ja toiminnallisuudet sekä käyttöliittymätason tarkistukset ja taustajärjestelmän virhe- ja poikkeustilojen käsittely. Tässä luvussa järjestelmään rekisteröityneestä ja sisään kirjautuneesta henkilöstä, jolla ei ole hallintaoikeuksia, käytetään nimitystä asiakas. Henkilöä, joka ei ole kirjautunut tai rekisteröitynyt järjestelmään kutsutaan käyttäjäksi ja henkilö, joka on kirjautunut järjestelmään käyttäjätunnuksella, jotka antavat hallintaoikeuden, kutsutaan hallinnan käyttäjäksi.

3.1 Käyttötapaus

Lounaspassin käyttäjä on lounastava henkilö, tyypillisesti työpaikkaruokailija, joka käy säännöllisesti ruokailemassa ketjun ruokapaikoissa. Lounaspassin tarjoaja haluaa houkutella ruokailijoita ravintolaansa passin avulla ja toisaalta palkita säännölliset ruokailijat tarjoamalla kuluttajalle veloituksettoman lounaan, kun hän on ostanut riittävän määrän lounaita.

Lounaspassin käyttäjä, jäljempänä asiakas, voi käyttää passiaan ostaessaan lounaan ravintolassa. Asiakas voi käyttää passia omaan lounaaseensa, mutta hän voi tarjota lounaan myös esimerkiksi työkavereilleen, jolloin hän on oikeutettu saamaan leiman passiinsa myös heidän lounaistaan. Tässä työssä kerralla käytettävien lounaiden maksimimäärä on rajattu neljään.

Mikäli passi tulee täyteen, voi asiakas valita haluaako hän käyttää ilmaisen lounaan tällä kertaa vai siirtää sen käytön myöhemmälle, jolloin ilmainen kerta säilytetään vanhalla passilla ja yli menevät käyttökerrat leimataan toiselta passilta. Asiakkaalla on aina kaksi aktiivista passia, joita hän voi halutessaan käyttää esimerkiksi henkilökohtaiseen käyttöön ja työkäyttöön.

Passin täyttyessä asiakkaalle alustetaan automaattisesti uusi tyhjä passi täyttyneen tilalle. Mikäli toisella passilla on leimattuja lounaita, ne siirretään oletusarvoisesti ensimmäiselle passille.

Asiakkaan saapuessa ravintolan tiskille ja ostaessa lounaan, hän näyttää sovelluksen luoman QR-koodin myyjälle. Myyjä lukee koodin oman mobiililaitteensa Lounaspassi-sovelluksen hallintakäyttöliittymän QR-koodin lukijalla, joka parsii koodista parametrit ja päivittää myyjän käyttöliittymän. Myyjä valitsee käyttöliittymässä, montako lounasta passilta käytetään, sekä jos passi on täyttymässä, mitä passilta yli meneville lounaille tehdään ja haluaako asiakas käyttää ilmaisen lounaansa nyt, vai siirretäänkö leima seuraavalle passille.

3.2 Toteutus

Sovellus on toteutettu käyttäen React-käyttöliittymäkerrosta, johon on rakennettu välikerros, joka huolehtii taustalogiikasta. Sovelluksen taustalla on Google Firebase -kirjautumisen hallinta ja Google Firestore -dokumenttitietokanta, johon käyttäjän data tallennetaan globaalia käyttöä varten. Tietokantaa voidaan joutua laajentamaan, jolloin vaihtoehdoksi tulee tietokantakerroksen toteuttaminen esimerkiksi MariaDB-alustalle.

Käyttöliittymäkerros on jaettu kolmeen osaan. kirjautumattoman käyttäjän näkymään, kirjautuneen asiakkaan näkymään ja hallintanäkymään, jolla toteutetaan passin lukeminen ja lounaiden käyttäminen.

3.2.1 Kirjautumaton käyttäjä

Kirjautumattomalta käyttäjältä vaaditaan rekisteröityminen ja kirjautuminen sovellukseen. Rekisteröitymisestä haluttiin tehdä mahdollisimman suoraviivainen ja helppo. Firebasen katsottiin tarjoavan tähän parhaat työkalut. Kirjautumiseen käytetään Google Firebasen tarjoamaa sähköpostipohjaista kirjautumismenetelmää. Kirjautuminen vaaditaan käyttäjän tunnistukseen, tilastojen tallennukseen sekä lounaspassin tilan tallennukseen. Lounaspassin tila tallennetaan

paikallisesti asiakkaan laitteeseen, mutta Firebasen Firestore-tietokanta säilyttää passin tilan Googlen pilvipalvelussa, jolloin passia voidaan käyttää usealla laitteella.

Palveluun rekisteröityminen tapahtuu Googlen Firebasen rekisteröitymisominaisuudella. Sovellustasolla tarkistetaan sähköpostin muoto ja salasanan vähimmäisvaatimukset. Ennen rekisteröitymistä käyttäjää pyydetään vahvistamaan sähköpostiosoitteensa ja salasanansa virheiden välttämiseksi.

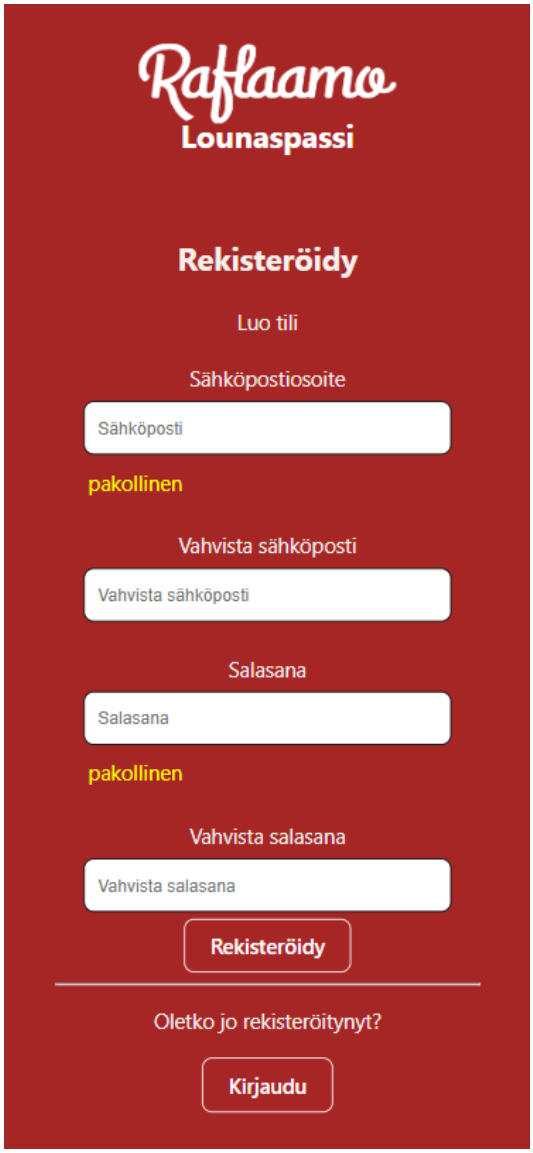
Kuva 10. Sovelluksen kirjautumisnäkymä (Kuvankaappaus sovelluksesta)

Kirjautumisnäkymä (Kuva 10) toteutettiin tilallisena lomakkeena (stateful form), jossa oletustilana on kirjautuminen. Muut mahdolliset tilat ovat rekisteröityminen

ja salasanan palautus. Kirjautumisnäkymä muuttuu dynaamisesti tilan muuttuessa. Tilanhallinta toteutettiin Reactin useState-koukulla (hook).

Kirjautumistilassa käyttäjältä kysytään sähköpostiosoite ja salasana, jotka lähetetään Googlen Firebase –palveluun vahvistettavaksi. Mahdolliset virheilmoitukset esitetään käyttäjälle kirjautumiskäyttöliittymässä.

Rekisteröitymistilassa (Kuva 11) käyttäjältä kysytään sähköpostiosoite ja pyydetään määrittämään salasana. Kirjoitusvirheiden välttämiseksi sekä sähköpostiosoite että salasana pyydetään vahvistamaan ennen rekisteröitymistä. Salasanan ja sähköpostiosoitteen vastaavuus vahvistetaan käyttöliittymätason tarkistuksella ennen Google Firebasen rajapintaan lähetystä.



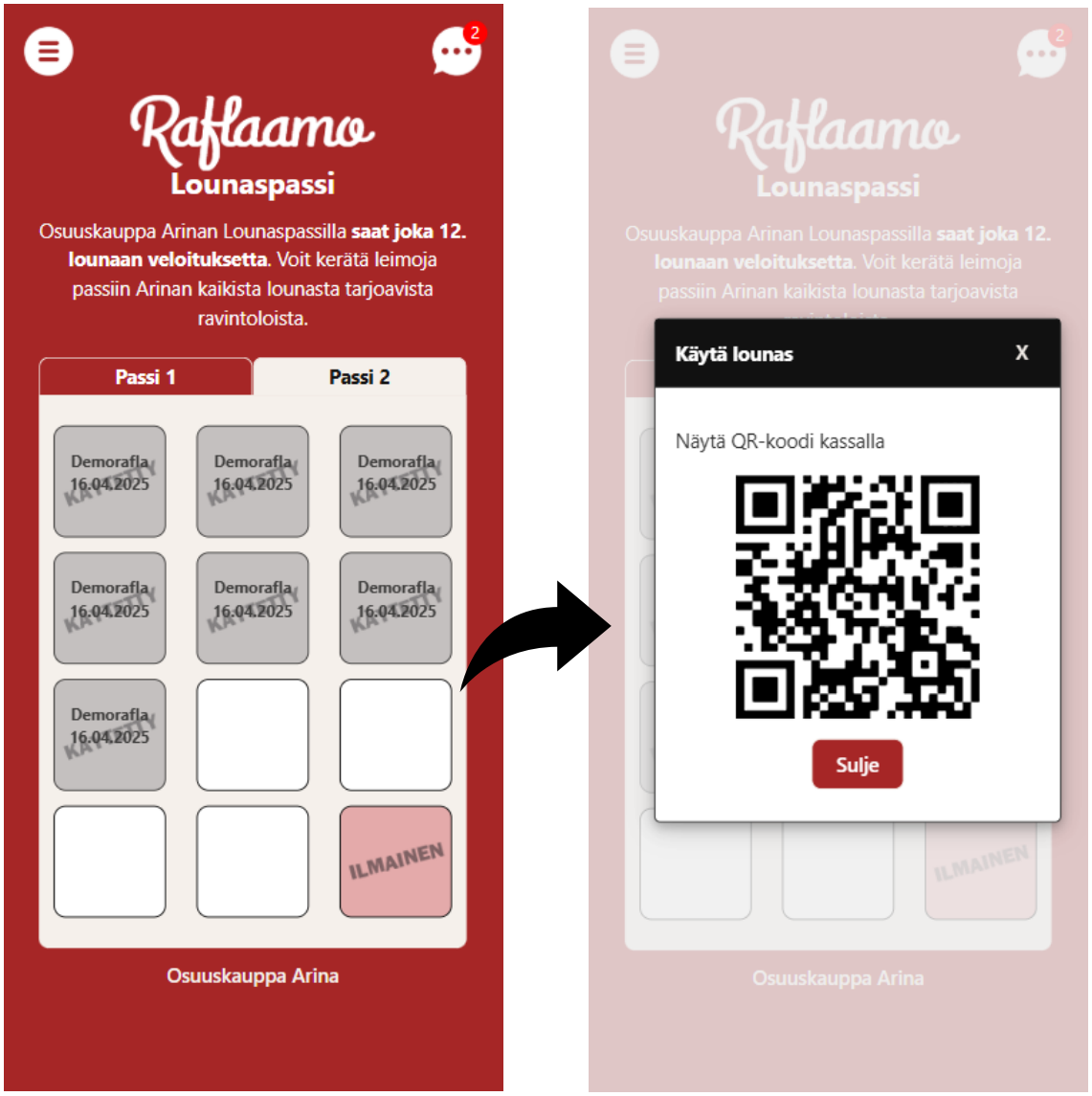
Kuva 11. Asiakkaaksi rekisteröityminen (Kuvankaappaus sovelluksesta)

Salasanan palautus –tilassa käyttäjältä kysytään sähköpostiosoite, joka lähetetään Google Firebasen rajapintaan salasanan palautuslinkin pyytämiseksi. Firebasen sisäinen logiikka käsittelee salasanan palautuspyynnön ja lähettää käyttäjän sähköpostiosoitteeseen linkin salasanan uusimiseen. Käyttäjälle lähetettävän sähköpostiviestin sisältö määritellään Firebase-palvelun asetuksissa.

Kirjautumislomake tarkistetaan dynaamisesti käyttöliittymätasolla käyttäen Formik-kirjaston FormControl-kirjastoa sekä Yup-validointikirjastoa. Kenttäkohtaiset virheilmoitukset esitetään käyttöliittymässä syöttökentän alapuolella. Kirjautumissivun kenttätiedot ovat pakollisia ja niille on määritelty vähimmäismitta ja muotovaatimukset, jotka tarkistetaan ennen lomaketietojen lähetystä Firebase-rajapintaan.

3.2.2 Kirjautunut asiakas

Kirjautuneelle asiakkaalle avautuu ensimmäisenä lounaspassinäkymä, jossa on aina näkyvillä kaksi aktiivista passia (Kuva 12). Passin lounaiden ja palkintolounaiden määrä määritellään ohjelmiston asetuksissa. Käyttöliittymä on reaktiivinen, eli se suhteutuu käyttäjän päätelaitteen mukaan kuitenkin niin, että passin lounaat on jaettu kolmeen sarakkeeseen keskelle näyttöä. Passiin kerätään leimoja käyttämätöntä lounasruutua klikkaamalla, jolloin ruudulle avautuu QR-koodi (Kuva 12), joka voidaan lukea hallinnan käyttöliittymän QR-koodin lukijalla.



Kuva 12. Lounaspassi ja QR-koodi (Kuvankaappaus sovelluksesta)

Kun passi täytyy, alustetaan se automaattisesti uudella tyhjällä passilla. Käytetyt lounaat esitetään käyttöliittymällä lounassolun värin muutoksella, käyttöpäivämäärällä ja ravintolan nimellä.

Asiakkaan passitiedot tallennetaan ensisijaisesti laitteen sisäiseen muistiin JSON-muodossa. Passidata tallennetaan myös Firestore-tietokannasta, jolloin se on asiakkaan käytettävissä millä tahansa laitteella kirjautumisen jälkeen. Passidata synkronoidaan laitteen omaan muistiin Firestore -tietokannasta käyttäen Reactin useEffect-koukkua.

Lounaspassin lisäksi asiakkaalla on käytettävissään valikko, jossa ovat toiminnot uloskirjautumiseen ja tilin poistamiseen. Lisäksi sovellukseen on rakennettu tuki käyttäjän asetuksille, lähellä olevien lounaspaikkojen haulle ja lounaslistojen esitykselle, tapahtumien esitykselle sekä viestinnälle. Asetussivulla käyttäjä voi muun muassa valita haluaako hän push-viestejä sovelluksesta.

3.2.3 Hallinnan käyttäjä

Hallinnan käyttäjän käyttäjätunnus on sähköpostiosoite ja hallinnan käyttäjä käyttää samaa kirjautumissivua kuin asiakas. Näin vältytään kirjautumissivun ylimääräiseltä logiikalta. Vaihtoehtoina olisi ollut toteuttaa hallinnan käyttäjälle oma kirjautumissivu, joka olisi vaatinut Reactin Router-kirjaston käyttämistä. Tämä ei olisi ollut järkevää ainoastaan hallinnan kirjautumissivun ohjauksen toteuttamiseksi. Toinen lähestymistapa olisi ollut lisätä admin-kirjautumiseen oma tilakoukku (hook), jota ohjataan käyttöliittymäkomponentilla. Näin Formik-validointi olisi voitu kiertää ja rakentaa kirjautumislogiikkaan ohitus admin-käyttäjälle. Tässä työssä hallinnan käyttäjän tunnistus toteutettiin Firebasen roolilla, jolloin itse kirjautuminen tapahtuu samalla tavalla, kuin asiakkaana, mutta kirjautumisen jälkeen sovelluksen tila vaihtuu administrator-tilaksi. Rooli kertoo sovellukselle lisäksi käyttäjän toimipisteen, joka tallentuu lounaan käyttöpaikaksi.

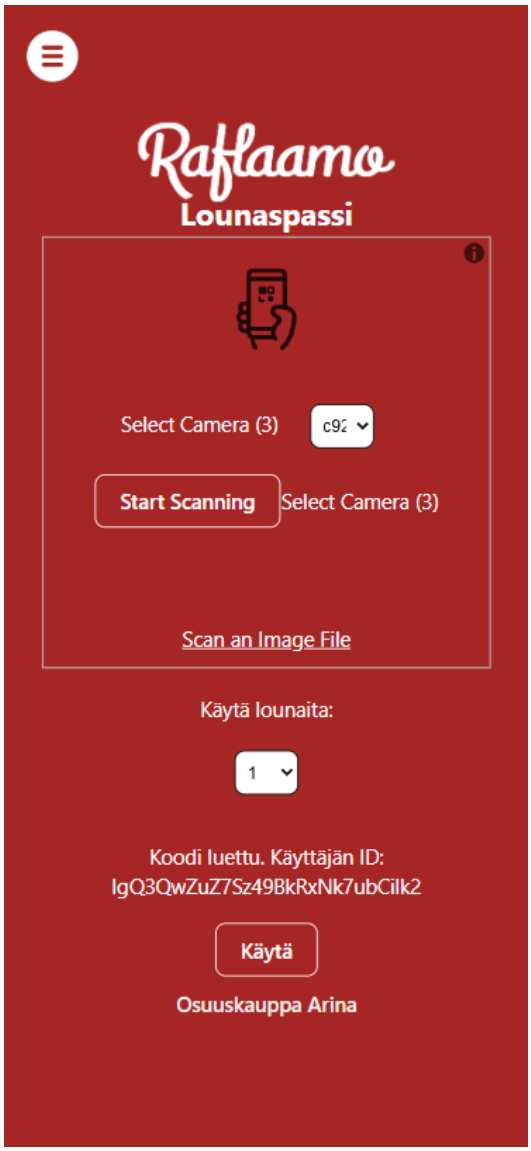
Hallintatason kirjautuneelle käyttäjälle esitetään QR-koodin lukija (Kuva 13) sekä valikkokomponentti käytettävien lounaiden määrän valintaan. Hallinnan käyttäjä voi lukea asiakkaan puhelimen Lounaspassi-sovelluksen esittämän QR-koodin, joka sisältää asiakkaan tunnisteen. Mikäli QR-koodin lukijan admin-istunto on voimassa, lähettää sovellus kutsun välikerroksen (middleware) funktiolle, jonka tehtävänä on taustalogiikan toteuttaminen ja käyttäjän verifiointi. Lisäksi Firestoren oma säännöstö huolehtii käyttäjän todentamisesta huolehtien, että luvaton käyttäjä ei pääse muokkaamaan passin tietoja.

Luettava QR-koodi välittää hallintakäyttöliittymän lukulaitteelle parametrijonona asiakkaan ja luetun passin identifioivat tiedot sekä ennakkotiedon passilla jäljellä olevista lounaista. Passilla olevien lounaiden määrä vaikuttaa dynaamisesti hallinnan käyttäjän käyttöliittymään ja käyttäjältä kysyttäviin lisätietoihin. Tällä

vähennetään mahdollisia virhetilanteita ja konflikteja välikerroksen vähentäessä passin käyttökertoja.

Kuka tahansa voi lukea koodin parametrijonon matkapuhelimen QR-koodin lukijalla, mutta parametreista ei ole yksinään mitään hyötyä ilman sovelluksen hallintakäyttöliittymää. Hallintakäyttöliittymä välittää QR-koodista saadut parametrit ja hallinnan käyttäjän määrittämän käytettävien lounaiden määrän välikerroksen funktiolle, jonka toimintalogiikka merkitsee Firestore-tietokannassa asiakkaan passista valitun määrän lounaita käytetyksi.

Oletuksena lounaita käytetään yksi, mutta hallinnan käyttäjä voi valita kerrallaan enimmillään neljä käytettävää lounasta. Mikäli passi tulee täyteen, kysytään käyttäjältä, miten sovelluksen tulee toimia ylivuodon suhteen sekä käytetäänkö ilmainen lounas, vai siirretäänkö se toiselle tai vaihtoehtoisesti uudelle passille.



Kuva 13. Hallinnan QR-koodin lukija (Kuvankaappaus sovelluksesta)

Taustajärjestelmä hakee asiakkaan aktiiviset passitiedot ja merkitsee annetun määrän lounaita käytetyksi.

3.2.4 Käyttöliittymätason tarkistukset

Käyttöliittymällä on käytännössä yksi lomake, jonka syötteet vaativat validointia. Tämä lomake on yhdistetty kolmeen erilliseen toimintoon. Käyttäjäksi rekisteröityminen, sisäänkirjautuminen ja salasanan uusintalinkin lähetys on toteutettu yhdellä ja samalla lomakkeella, jonka syöttökentät mukautuvat käyttäjän valintojen mukaan. Tämä asettaa omat haasteensa lomakkeen validoinnille.

Kirjautumissivun lomakkeen validointi on toteutettu Yup-kirjaston ehdollisella validointikaavalla (Kuva 14).

```
validationSchema : Yup.object({
    email: Yup.string().email("Sähköpostiosoite ei kelpaa").min(8, "Sähköpostin tulee olla
    vähintään 8 merkkiä pitkä").required('pakollinen'),
    verify_email: Yup.string().when("email", {
        is: (email) => isRegistration,
        then: () => Yup.string().email("Vahvistettu sähköpostiosoite ei kelpaa").min(8,
        "Sähköpostiosoitteen tulee olla vähintään 8 merkkiä pitkä").required("pakollinen")
      }),
    password: Yup.string().when("email", {
        is: (email) => !isForgotPassword,
        then: () => Yup.string().min(6, "Salasanan tulee olla vähintään 6 merkkiä pitkä").
        required('pakollinen'),
    }),
    verify_password: Yup.string().when("password", {
      is: (password) => isRegistration,
      then: () => Yup.string().min(6, "Vahvistettu salasana ei kelpaa").required("pakollinen")
    }),
  }),
```

Kuva 14. Yup-kirjaston ehdollinen validointimalli (kuvankaappaus Visual Studio Code -koodieditorista)

Käyttäjän sovelluksessa on koko ajan reaaliaikainen tieto lounaspassin tilasta. Kun lounas käytetään, välitetään asiakkaan tiedot QR-koodin parametrina hallintatason käyttöliittymään, joka kutsuu välikerroksen funktiota. Virhetilanteiden minimoimiseksi QR-koodin parametrina välitetään myös kulloinkin luetun passin tunniste sekä tiedot passilla käytössä olevista lounaista. Hallintatason käyttöliittymässä suoritetaan esitarkistus ja kysytään sovelluksen käyttäjältä valmiiksi erilaisten poikkeustilojen käsittelyt. Tällaisia esitarkistettavia asioita ovat passin ylivuoto, täyttyminen sekä ilmaisen lounaan siirtäminen seuraavaan käyttökertaan.

Esimerkiksi, asiakkaan ensisijaisessa passissa on käyttämättä yksi lounas ennen ilmaista lounasetua. Asiakas tulee syömään ravintolaan kolmen ystävänsä kanssa ja maksaessaan kaikkien lounaat, hän on oikeutettu neljään lounasleimaan. QR-koodin parametreissa hallintakäyttöliittymään välitetään käytettävän passin tunniste sekä tieto, montako leimattavaa lounasta passissa on jäljellä. Kun hallinnan käyttäjä muuttaa käytettävien lounaiden määrää käyttöliittymällä ja annettu määrä ylittää passilla käytettävissä olevien lounaiden määrän, kysytään käyttäjältä dynaamisesti ylivuodon tapauksessa suoritettavat operaatiot, kuten ilmaisen lounaan käyttäminen ja lounaiden siirtäminen kokonaan uudelle passille toissijaisen passin sijaan.

3.2.5 Taustajärjestelmän virheenkäsittely

Käyttöliittymätason virheenkäsittelyn lisäksi sovellus tekee useita taustajärjestelmän tarkistuksia, joista ilmoitetaan käyttöliittymätasolle virheviesteillä. Esimerkiksi käyttöliittymässä näkyvä passidata ei välttämättä vastaa Firestore-tietokannassa olevaa passidataa, jolloin taustajärjestelmän on kyettävä käsittelemään virhetilanne ja ilmoitettava siitä käyttöliittymätasolle.

Ensimmäinen taustajärjestelmän virheentarkistus tehdään käyttäjän kirjautuessa Firebase-rajapinnan kautta järjestelmään. Firebase palauttaa virheilmoituksen, mikäli käyttäjän syöttämät tiedot, kuten sähköpostiosoite ja salasana, eivät vastaa sinne tallennettua dataa tai käyttäjän rekisteröimä sähköpostiosoite on jo käytössä. Rekisteröitymisen ja kirjautumisen virheet esitetään kirjautumissivulla sähköpostin syöttökentän yläpuolella.

Käyttäjän kirjauduttua sisään sovellukseen, haetaan asiakkaan lounaspassien tiedot Firestore-tietokannasta. Mikäli yhteys tietokantaan ei ole saatavilla tai asiakkaalla ei vielä ole tallennettua passidataa Firebasessa, haetaan seuraavaksi passitiedot laitteen paikallisesta muistista. Mikäli täälläkään ei ole tallennettuja tietoja alustetaan asiakkaalle uusi passi, jonka tiedot tallennetaan paikalliseen laitteeseen ja synkronoidaan Firestoren passitietojen kanssa, kun yhteys tietokantaan saadaan muodostettua.

Passin käyttäminen tapahtuu QR-koodilla hallintakäyttöliittymään välitettyjen parametrien perusteella. Ensimmäinen parametri on asiakkaan identifioiva tunniste, jonka avulla taustalogiikan pitäisi löytää käyttäjän passitiedot Firestore-tietokannasta. Periaatteessa on mahdollista, että taustajärjestelmä ei löydä passidataa, jolloin tästä tulee virheilmoitus hallintakäyttöliittymään. QR-koodin parametreina välitetään myös passin identifioiva tieto sekä passilla vapaana olevien lounaiden määrä, joiden perusteella hallinnan käyttäjä voi määritellä miten passia käytetään. On kuitenkin mahdollista, että välitetyt parametrit eivät vastaakaan Firestore-tietokannan dataa, jolloin taustajärjestelmän on ilmoitettava hallinnan käyttöliittymään mahdollisesta poikkeuksesta. Mahdollisia poikkeustilanteita ovat esimerkiksi, jos käytettäväksi määriteltyjen lounaiden määrä ylittää passilla vapaana

olevien lounaiden määrän tai passi on täynnä eikä uutta passia ole voitu luoda oikein.

4 POHDINTA

Työn tavoite oli toteuttaa käyttäjäystävällinen ja helposti lähestyttävä sähköinen versio pahvisesta lounaspassista siten, että sovellus ei anna mahdollisuutta passin väärinkäyttämiseen. Helppo ratkaisu passin käyttämiseen olisi ollut käyttäjän omatoiminen passin leimaaminen ravintolan tiskille tulostetun QR-koodin avulla, mutta tämä olisi tarjonnut epärehelliselle käyttäjälle mahdollisuuden leimata passi moneen kertaan. Tämän lähestymistavan sijaan sovellus toteutettiin hallintakäyttöliittymällä, jonka avulla kassatyöntekijä leimaa passin lukemalla käyttäjän esittämän QR-koodin. Toteutuksena tämä oli turvallisuuden kannalta parempi lähestymistapa, vaikkakin selkeästi monimutkaisempi.

Käyttöliittymätoteutuksesta tuli varsin onnistunut ja intuitiivinen. Toteutus jäljentää hyvin perinteistä pahvista lounaspassia, joka on useimmille kuluttajille tuttu. Näin UX-suunnittelun perusajatus käyttäjälle tutun kontekstin käyttämisestä toteutuu hyvin ja käyttöliittymä on hyvin itseään selittävä.

Työn haaste ei ollut käyttöliittymän innovointi, vaan itse passin käyttölogiikka ja uuden passin alustaminen vanhan täyttyessä. Tämän toteutukseen oli useita eri vaihtoehtoja, joista lopulta päädyttiin helpoimpaan, eli sovelluksen omaan välikerrokseen toteutettuun logiikkaan, eikä vaihtoehtoiseen, erilliseen backend-toteutukseen. Tämä valinta oli perusteltavissa taustajärjestelmän suhteellisen pienellä roolilla.

Työn suunnittelu aloitettiin prototyypin luomisella. Ensimmäiset käyttöliittymämallit toteutettiin HTML:n ja CSS:n avulla, mutta pian myös prototyypin toteutus siirtyi Reactiin. Prototyypin toteutus Reactilla oli huomattavan paljon nopeampaa ja käyttöliittymää oli helppo kehittää mallidatalla. Prototyypistä saatiin näin myös heti valmis pohja varsinaiselle toteutukselle.

Vaikka Reactin komponenttimalli oli tuttu aikaisemmista projekteista, olisi jälkikäteen ajateltuna ollut parempiakin vaihtoehtoja ohjelmistorakenteen toteutustavalle. Toteutuksessa lähdettiin siitä, että komponentit ja sivut ovat samassa komponenttikansiossa, mutta jälkeenpäin tarkasteltuna sivuille olisi ollut hyvä tehdä

selvyyden vuoksi oma kansio. Tämä korostuu erityisesti, mikäli sovelluksessa on paljon sivuja.

Reactin malli on erittäin toimiva ja sillä on nopea kehittää monimutkaisiakin dynaamisia käyttöliittymiä. Reactissa käytetty JSX-kieli, joka on JavaScriptin laajennus, vaatii kielenä jonkin verran opettelua. JavaScript antaa moniparadigmaisena kielenä paljon vapauksia, mutta kääntöpuolena koodin virheiden selvittäminen voi olla välillä haastavaa ja aikaa vievää.

Reactin huonona puolena on, kuten tämän opinnäytetyön tietoperustassa mainitaan, sen ohjelmistopaketin suhteellisen valtava koko varsinaiseen toteutukseen nähden. Jo React-projektin alustuksessa asennetaan useita kirjastoja, jotka ovat käytettävissä sovellusta kehitettäessä. Näiden lisäksi eri lisäkirjastot omine riippuvuuksineen tuovat lisää kokoa ohjelmistopakettiin. Tämän kääntöpuolena on kuitenkin hyvin toimiva ja dynaaminen ympäristö, joka antaa kehittäjälle paljon työkaluja toimivan ohjelmakoodin kirjoittamiseen.

Jatkokehityskohteina sovellukseen on suunniteltu push-viestintäominaisuus, johon liittyen vaaditaan asetuksiin optio push-viestinnän vastaanottamiselle. Lisäksi sovellukseen on suunniteltu omat näkymät lähellä olevien ketjun ravintoloiden yhteystietojen ja lounaslistojen esitykselle sekä ketjun tulevien tapahtumien esittämiselle. Nämä toiminnot kuitenkin edellyttävät toimeksiantajan API-rajapinnan avaamista, eivätkä ne kuuluneet tämän opinnäytetyön piiriin. Jatkokehityksen kannalta sovellus tarjoaa myös tehokkaan alustan erilaisten mainoskampanjoiden toteutukseen. Sovellukseen voidaan esimerkiksi toteuttaa mainos, joka ponnahtaa esiin, kun sovellus avataan tai kun lounasetu on käytetty. Yksi kehityskohde voisi olla myös jonkin määritellyn edun saaminen, kun veloitukseton lounas käytetään. Vastaavasti 12. lounaan ei tarvitse välttämättä olla ilmainen, vaan sovellus mahdollistaa muidenkin etujen hyödyntämisen, kuten vaikkapa 20 % alennuksen kertaostosta ketjun muista palveluista. Toimeksiantajan ketjulla on myös oma bonusohjelma, johon sovellus voidaan tarvittaessa integroida. Sovellusta voidaan myös laajentaa monimerkkiketjulle soveltuvaksi, jolloin yksi sovellus voi sisältää usean eri brändin lounaspassit.

Kuten kappaleessa 3.2.3 on mainittu, tietosuojaa ajatellen QR-koodin lukeminen millä tahansa laitteella ei tarjoa mitään oleellista informaatiota ulkopuoliselle käyttäjälle. QR-koodin sisältö on käyttökelvotonta, ellei käytössä ole Lounaspassin hallintasovelluksen käyttäjä. Haluttaessa tietosuojaa voidaan kuitenkin edelleen parantaa salaamalla QR-koodin sisältö salausavaimella ja purkamalla salaus QR-koodin lukijan koodissa.

LÄHTEET

Banks, A. & Porcello E. 2020. Learning React : modern patterns for developing React apps. E-kirja. O'Reilly, Beijing.

Bibeault, B., Katz, Y., De Rosa A. & Methvin, D. 2015. JQuery in Action. E-kirja. 3. painos. Manning. Shelter Island, New York.

Duckett, J. 2014. JavaScript & jQuery : interactive front-end web development. E-kirja.John Wiley & Sons Inc.

Garcia, V.H. 2023. Getting Started With Angular : create and deploy Angular applications. E-kirja. 1. painos. Apress. New York, NewYork.

Goulding, J. 2020. Stack Overflow Blog. What is Rust and why is it so popular? Luettavissa: https://stackoverflow.blog/2020/01/20/what-is-rust-and-why-is-it-so-popular/. Luettu: 4.5.2025.

Hanchett, E. 2016. Ember.js Cookbook : arm yourself with over 65 hands-on recipes to master the skills of building scalable web applications with Ember.js. E-kirja. 1. painos. Pact Publishing. Birmingham.

Haverbeke, M. 2025. Eloquent JavaScript. E-kirja. 4. painos. No Starch Press.

Kereki, F. 2025. Data Structures and Algorithms in JavaScript. E-kirja. 1. painos. No Starch Press.

Kumar, T. 2024. Fluent React. E-kirja. 1. painos. O'Reilly Media Inc.

Lintorn Catlin, H. & Lintorn Catlin, M.L. 2016. Pragmatic Guide to SASS 3 : tame the modern style sheet. E-kirja. 1. painos. The Pragmatic Bookshelf. Raleigh, North Carolina.

McFeddries, P. 2024. Web Design Playground, E-kirja. 2. painos. Manning Publications Co LLC. New York.

Pihkala J. 2018. Mikä ihmeen QR-koodi? : QR-koodi – tiedon portti. BoD – Books on Demand. Helsinki.

Rady, B. & Carter, J. 2016. Serverless Single Page Apps : Fast, scalable, and available. E-kirja. The Pragmatic Bookshelf. Raleigh, North Carolina.

Rising Stars of JavaScript 2024. Luettavissa: https://risingstars.js.org/2024/en. Luettu 2.5.2025.

Scott, E. 2016. SPA Design and Architecture : Understanding single page web applications. E-kirja. 1. painos. Manning. Shelter Island.

Shavin, M. 2023. Learning Vue. E-kirja. 1. painos. O'Reilly Media Inc.

Singh, H.& Tanna, M. 2018. Serverless Web Applications With React and Firebase. E-kirja. 1. painos. Packt Publishing.

Stack Overflow 2024. Developer Survey. Luettavissa: https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language-prof. Luettu: 28.4.2025.

Statista 2024. Most used programming languages among developers worldwide as of 2024. Luettavissa: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/. Luettu: 5.5.2025.