Tampereen ammattikorkeakoulu

Digitaalisen allekirjoituksen toteuttaminen

Mira Lindroos

OPINNÄYTETYÖ
Huhtikuu 2025

Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

LINDROOS, MIRA
Digitaalisen allekirjoituksen toteuttaminen

Opinnäytetyö 27 sivua, joista liitteitä 0 sivua
Huhtikuu 2025

Opinnäytetyön tarkoituksena oli luoda ratkaisu toimeksiantajayrityksen järjestelmään, mikä mahdollistaa digitaaliset allekirjoitukset työntekijän ja asiakasyrityksen välillä. Toteutus tehtiin hyödyntäen olemassa olevia JavaScript-kirjastoja sekä Signicatin autentikointiin tarkoitettua REST APIa. Ratkaisu toteutettiin taustajärjestelmään Firebase Cloud Functions -funktioina ja lisäksi työssä kehitettiin käyttöliittymät mobiilisovellukselle React Nativea käyttäen sekä hallinta- ja tilausjärjestelmään Vue.js:n avulla.

Opinnäytetyöraportissa käsitellään keskeisiä termejä ja avataan aiheeseen liittyviä käsitteitä. Lisäksi esitellään käytetyt teknologiat ja kuvataan toteutusvaiheet yksityiskohtaisesti.

Työn tuloksena syntyi kehitysympäristössä toimiva digitaalinen allekirjoitusprosessi, jossa allekirjoitus luodaan vahvan sähköisen tunnistautumisen jälkeen. Toteutus vastasi toimeksiantajayrityksen odotuksia. Jatkokehityksen kannalta keskeistä on sopimusten luontiin tarkoitetun työkalun saattaminen loppuun asti, jotta rajapinta voidaan ottaa käyttöön osana yrityksen laajempaa järjestelmää.

Asiasanat: digitaalinen allekirjoitus, sähköinen tunnistautuminen, Signicat, Firebase, REST API, ohjelmistokehitys

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Software Development

LINDROOS, MIRA
Implementation of digital signature

Bachelor's thesis 27 pages, appendices 0 pages
April 2025

The purpose of the thesis was to create a solution for the client company's system that enables digital signatures between the employee and the client company. The implementation utilized existing JavaScript libraries and the Signicat Authentication REST API. The solution was built in the backend as Firebase Cloud Functions, and additionally, user interfaces were developed for the mobile application as well as the management and ordering system.

The report examines key concepts and technologies used in the implementation and describes the development process in detail. As a result, a working digital signature process was created in the development environment, where a digital signature is generated after strong authentication. The implementation met the company's expectations, and future development will focus on finalizing the contract creation tool so that the API can be integrated into the company's system.

Key words: digital signature, electronic identification, Signicat, Firebase, REST API, software development

SISÄLLYS

LYHENTEET JA TERMIT

AES	Advanced Electronic Signatures
API	Application Programming Interface, rajapinta
BaaS	Pilvipalvelu, joka tarjoaa valmiita taustajärjestelmän toimintoja
Base64	Tiedon esitysmuoto, jossa data muunnetaan pelkiksi kirjaimiksi ja numeroiksi
eIDAS	electronic IDentification, Authentication and Trust Services
Hajautusarvo	Hajautusfunktion palauttama kiinteän pituinen arvo, joka toimii syötetyn tiedon yksilöllisenä tunnisteena
HTTPS	Hypertext Transfer Protocol Secure
Käyttöoikeustunnus	Rajapinnan käyttölupa, joka vahvistaa, että tunnuksen haltijalla on oikeus käyttää APIa ja suorittaa valtuutetut toiminnot (engl. Access token)
Päätepiste	Osoite (esim. URL), johon voi lähettää pyyntöjä ja saada vastauksia (engl. Endpoint)
PKI	Public Key Infrastructure, julkisen avaimen menetelmää hyödyntävä järjestelmä, joka mahdollistaa turvallisen viestinnän ja identiteettien varmentamisen
Public key encryption	Salausmenetelmä, jossa toinen salausavaimista on julkinen avain ja toinen on yksityinen avain
REST	Representational State Transfer
RSA	Rivest-Shamir-Adleman, salausalgoritmi, joka perustuu julkisen avaimen menetelmään
SHA-256	Secure Hash Algorithm 256-bit
SSL	Secure Sockets Layer
URL	Uniform Resource Locator

1 JOHDANTO

Opinnäytetyö toteutetaan toimeksiantona Hoiwa Oy:lle. Hoiwa on vuonna 2020 perustettu startup, joka keskittyy tarjoamaan vuokratyövoimaa terveydenhuolto-alalla. Hoiwan tavoitteena on turvata asiakkaidensa henkilöstön saatavuus, ja työt voivat kestää päivän sijaisuuksista kuukausien mittaisiin työsuhteisiin. Yksi oleellisista osa-alueista keikkatyönhallintajärjestelmissä on työntekijöiden sopimusten hallinta. Hoiwan Aurora-järjestelmään tarvitaan ratkaisu, joka mahdollistaa digitaaliset allekirjoitukset työntekijän ja asiakasyrityksen välillä.

Opinnäytetyön tarkoituksena on kehittää digitaalinen allekirjoitusratkaisu, joka käyttää vahvaa tunnistautumista työntekijöiden ja asiakasyritysten välisissä sopimuksissa. Toteutusta varten on olemassa kolmannen osapuolen ratkaisuja, mutta tavoitteena on luoda oma ratkaisu hyödyntäen olemassa olevia JavaScript-kirjastoja. Digitaaliseen allekirjoitukseen tarvitaan vahvaa tunnistautumista työntekijältä - tätä varten täytyy käyttää kolmannen osapuolen rajapintaa. Projektissa keskitytään ensisijaisesti taustajärjestelmän toiminnallisuuksien toteuttamiseen, ja käyttöliittymien kehittäminen nähdään mahdollisena laajennuksena kokonaisuuteen.

Oman ratkaisun kehittäminen tarjoaa suojan odottamattomilta muutoksilta, kuten hinnankorotuksilta tai jopa palvelun lakkauttamiselta. Lisäksi oma ratkaisu on täysin muokattavissa yrityksen tarpeisiin. Näin yrityksellä säilyy kontrolli järjestelmän kehityksessä ja mukauttamisessa.

2 TAUSTA

2.1 Vahva sähköinen tunnistautuminen

Vahvaa sähköistä tunnistautumista käytetään henkilöllisyyden todentamiseen. Suomessa vahvan sähköisen tunnistautumisen palveluntarjoajia on kahdenlaisia: toiset tarjoavat tunnistusvälineitä käyttäjille ja toiset tunnistuspalveluita asiointipalveluille. Halutessaan palveluntarjoaja voi toimia molempien palveluiden tarjoajana. (Kyberturvallisuuskeskus 2024.)

Euroopan unionin eIDAS-asetus (electronic IDentification, Authentication and Trust Services) edistää turvallista rajat ylittävää kaupankäyntiä luomalla puitteet digitaaliselle identiteetille ja tunnistautumiselle. Asetuksen tavoitteena on vahvistaa luottamusta sähköisissä vuorovaikutuksissa ja edistää saumattomia digitaalisia palveluja EU:ssa. EIDAS-asetus sähköisessä tunnistautumisessa ja luottamuspalveluissa oli merkittävä askel kohti ennakoitavampaa sääntely-ympäristöä. Asetus kohdistuu erityisesti sähköiseen tunnistautumiseen (eID) ja luottamuspalveluiden tarjoajiin. Sen tavoitteena on purkaa olemassa olevia esteitä, jotka haittaavat luottamuspalveluiden ja eID:n sujuvaa käyttöä EU:n jäsenvaltioiden välillä. (European Commission 2024.)

EIDAS-asetus mahdollistaa kansallisten eID-järjestelmien yhteensopivuuden EU:n jäsenvaltioiden välillä. Tämä edellyttää teknologianeutraalin viitekehyksen kehittämistä, joka ei suosi mitään tiettyä teknistä ratkaisua eID:n toteutuksessa. EU-maiden välisen yhteistyön helpottamiseksi on määritelty menettelyllisiä ja teknisiä standardeja, joilla pyritään varmistamaan sähköisten tunnistustietojen sujuva vaihto ja edistämään yhtenäistä digitaalista ekosysteemiä kaikkialla EU:ssa. (European Commission 2024.)

2.2 Digitaalinen allekirjoitus

Digitaalinen allekirjoitus on erityinen tekninen toteutus sähköisestä allekirjoituksesta (eSignature). Kynän ja paperin sijaan digitaalinen allekirjoitus hyödyntää

matemaattisia algoritmeja ja salausmenetelmiä asiakirjan, tiedoston tai ohjelmiston allekirjoittamiseen ja aitouden varmistamiseen. Se on innovatiivinen ja turvallinen ratkaisu, jolla varmistetaan allekirjoittajan henkilöllisyys ja asiakirjan sisältö. (Guide to digital signatures 2023.)

Vaikka digitaalista ja sähköistä allekirjoitusta käytetään usein synonyymeinä, niiden välillä on selkeä ero, joka perustuu niiden tekniseen toteutukseen ja käyttötarkoitukseen. Sähköinen allekirjoitus on verrattavissa käsin kirjoitettuun allekirjoitukseen, ja se on paperiversion tavoin juridinen käsite ja sen on allekirjoittanut ihminen. Se ei kuitenkaan aina sisällä todisteita allekirjoittajan henkilöllisyydestä. (Guide to digital signatures 2023.) Digitaaliset allekirjoitukset sen sijaan ovat erityinen sähköisen allekirjoituksen muoto, joka perustuu julkisen avaimen infrastruktuuriin (PKI, Public Key Infrastructure). PKI:n mukaisesti jokainen digitaalinen allekirjoitus vaatii parin digitaalisia avaimia, yhden julkisen ja yhden yksityisen, jotka luodaan, käytetään ja tallennetaan turvallisesti. Vain asiakirjan allekirjoittajalla on pääsy yksityiseen avaimeen, kun taas allekirjoituksen tarkistamiseen käytetään julkista avainta. (DocuSign 2024.)

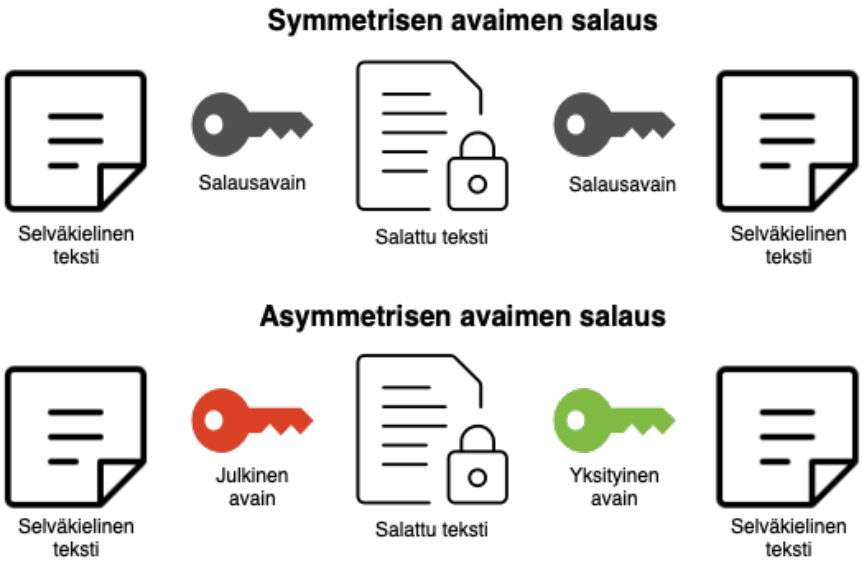
Euroopan unionin eIDAS-asetuksessa määriteltyjä digitaalisia allekirjoituksia on kahta päätyyppiä: Advanced Electronic Signatures (AES) ja Qualified Electronic Signatures (QES). AES-allekirjoitukset lisäävät henkilöllisyyden todentamisvaatimuksen perinteisen sähköisen allekirjoituksen päälle. Allekirjoitusten on oltava yksilöllisesti linkitettyjä allekirjoittajaan ja kyettävä tunnistamaan allekirjoittaja. (DocuSign 2024.)

AES-allekirjoitusten ja tavallisten sähköisten allekirjoitusten suurin ero on se, että AES vaatii allekirjoittajan todistamaan henkilöllisyytensä asiakirjaa allekirjoittaessaan. AES-allekirjoituksia käytetään parhaiten keskitason tietoturvatarpeisiin liittyvissä asiakirjoissa, kuten työsopimuksissa. AES-allekirjoitukset tarjoavat enemmän turvallisuutta ja varmistamista kuin tavalliset sähköiset allekirjoitukset, mutta niiden kokoaminen ei vie paljon enemmän aikaa. (DocuSign 2024.)

2.3 Kryptografia eli salaus

Kryptografia on prosessi, jossa tiedot suojataan siten, että vain viestin vastaanottaja voi lukea sen. Nykyaikaiset salaustekniikat perustuvat algoritmeihin ja matemaattisiin menetelmiin, jotka mahdollistavat tiedon salauksen ja sen purkamisen. Kryptografiassa käytetään muun muassa salausavaimia ja digitaalisia allekirjoituksia, joilla viestit muunnetaan vaikeasti tulkittaviksi koodeiksi tietosuojan, luottokorttitapahtumien, sähköpostin ja verkkoselailun suojaamiseksi. Kryptografia on keskeinen työkalu tietojen ja käyttäjien suojaamisessa, luottamuksellisuuden takaamisessa ja verkkorikollisten estämisessä sieppaamasta arkaluonteisia yritystietoja. (Fortinet n.d.)

RSA-salausjärjestelmä (Rivest-Shamir-Adleman) mullisti kryptografian ensimmäisenä julkisen avaimen salausmenetelmänä (public key encryption) vuonna 1977. Perinteiset symmetrisen avaimen salausmenetelmät käyttävät samaa avainta viestien salaamiseen ja purkuun, mutta julkisen avaimen salaus, eli asymmetrinen salaus, hyödyntää kahta avainta: julkista avainta, joka salaa, ja yksityistä avainta, joka purkaa salauksen. (Aumasson 2025.) Kuviossa 1 havainnollistetaan tätä eroa. Kuvion harmaat avaimet edustavat symmetristä salausta, jossa sama avain toimii sekä salaamiseen että purkamiseen. Punainen ja vihreä avainpari puolestaan kuvaa asymmetristä salausta, jossa julkinen avain salaa ja yksityinen avain purkaa viestin. RSA:ta voidaan käyttää salauksen lisäksi myös digitaalisten allekirjoitusten luomiseen. Yksityisen avaimen omistaja on ainoa, joka voi allekirjoittaa viestin, ja julkisen avaimen avulla kuka tahansa voi tarkistaa allekirjoituksen aitouden. (Aumasson 2025.)

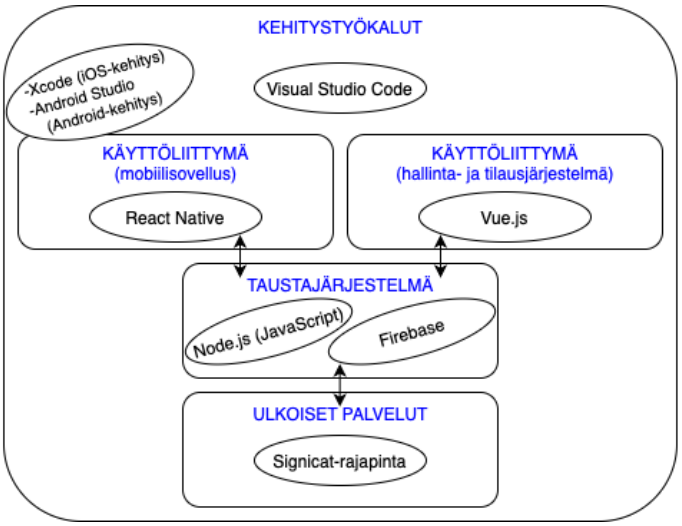


KUVIO 1. Symmetrisen ja asymmetrisen salauksen ero

Julkinen avain on yleensä julkisesti saatavilla kaikille, jotka haluavat lähettää sinulle salattuja viestejä. Yksityisen avaimen tulee kuitenkin aina pysyä salassa. Julkinen avain voidaan laskea yksityisestä avaimesta, mutta yksityistä avainta ei voida laskea julkisesta avaimesta, ja tämä on julkisen avaimen kryptografian (public key cryptography) perusperiaate. Asymmetrisen avainparin luomiseen voi käyttää OpenSSL:ää (open-source software library) RSA-yksityisavaimen luomiseksi. Avain saadaan base64-koodattuna datana, sillä useimmat järjestelmät tukevat tätä muotoa ja voivat muuntaa sen binaarimuotoiseksi dataksi. Voit myös luoda avainparin itse, mutta tämä vie huomattavasti enemmän aikaa. (Aumasson 2025.)

3 KÄYTETYT TEKNOLOGIAT

Tässä luvussa esitellään projektissa käytetyt keskeiset teknologiat, kuten vahvan tunnistautumisen rajapinta ja ohjelmointikehykset. Teknologiat on valittu toimeksiantajayrityksen jo ennestään käytössä olevien tekniikoiden perusteella. Taustajärjestelmän kehityksessä ja integraatiossa Signicatin rajapintaan on käytetty JavaScriptiä Node.js-ympäristössä ja Firebase-alustaa. Käyttöliittymissä on hyödynnetty React Nativea ja Vue.js:ää. Kehitystyö on toteutettu Visual Studio Code -ohjelmalla. Sovelluksen testaamiseen sekä simulointiin on hyödynnetty Xcodea iOS-sovellukselle ja Android Studiota Android-sovellukselle. Kuviossa 2 kuvataan, kuinka eri teknologiat ja työkalut ovat osana projektia.



KUVIO 2. Projektissa käytettyjen teknologioiden ja työkalujen rakenne ja vuorovaikutus.

3.1 Signicat-rajapinta

Tähän projektiin vahvan tunnistautumisen rajapinnan toimittajaksi valikoitui Signicat heidän laajan ja selkeän dokumentaationsa myötä. Kilpailijoihin verrattuna dokumentaatio oli huomattavasti parempi, mikä lopulta ratkaisi valinnan. Toimeksiantaja ei antanut muita valintaan vaikuttavia perusteita. Signicat on Euroopan johtava luottamus- ja varmennepalvelujen sekä sähköisten allekirjoitusratkaisujen toimittaja (Signicat – Tietoa meistä n.d.). Signicat tarjoaa monipuolisen vali-

koiman digitaalisia identiteettipalveluita kuten varmennepalveluita, sähköisiä allekirjoitusratkaisuita sekä vahvan tunnistautumisen ratkaisuja. Lisäksi Signicat tarjoaa Trust Orchestration -ratkaisuja, jotka auttavat hallinnoimaan ja automatisoimaan tunnistamisen ja riskinhallinnan työnkulkuja. (Signicat n.d.)

Tässä projektissa käytettiin Signicat eID Hubia, joka mahdollistaa käyttäjien henkilöllisyyden vahvistamisen kirjautumisen yhteydessä. Tämä voidaan toteuttaa yhdistämällä sovellus sähköiseen tunnistusmenetelmään (eID), kuten tässä tapauksessa Suomen FTN-järjestelmään (Finnish Trust Network). Signicatin dokumentaation mukaisesti integraatio voidaan toteuttaa kolmella eri tavalla, joista tähän projektiin valittiin Signicat Authentication REST API (Representational State Transfer Application Programming Interface). (eID Hub 2024.)

Signicat Authentication REST API tarjoaa helpon ja turvallisen tavan tunnistaa loppukäyttäjä. Se on yksinkertaisempi integroida kuin muut Signicatin tarjoamat vaihtoehdot, kuten OIDC (OpenID Connect) ja SAML 2.0 (Security Assertion Markup Language), mutta saattaa edellyttää enemmän räätälöityä koodia. Authentication REST API tukee tällä hetkellä redirect- ja headless-tunnistautumisia, mutta headless-tunnistautuminen on saatavilla vain Ruotsin pankkitunnistautumiselle. Tästä syystä toteutuksessa on käytetty redirect-tunnistautumista, jossa käyttäjä ohjataan selaimessa avattavaan URL-osoitteeseen tunnistautumista varten. (Authentication REST API 2025.)

3.2 JavaScript ja Node.js

Node.js on avoimen lähdekoodin JavaScript-ajoympäristö, joka käyttää Google Chromen V8 JavaScript -moottoria, selaimen ulkopuolella. Tämä tekee Node.js:stä erittäin suorituskykyisen. Sen avulla kehittäjät voivat rakentaa nopeita ja skaalautuvia sovelluksia JavaScriptillä myös verkkoselaimien ulkopuolella. (Introduction to Node.js n.d.) Node tarjoaa JavaScriptille pääsyn koko käyttöjärjestelmään, minkä ansiosta JavaScript-ohjelmat voivat lukea ja kirjoittaa tiedostoja, lähettää ja vastaanottaa dataa verkon kautta sekä käsitellä HTTP-pyyntöjä (Hypertext Transfer Protocol) (Flanagan 2020).

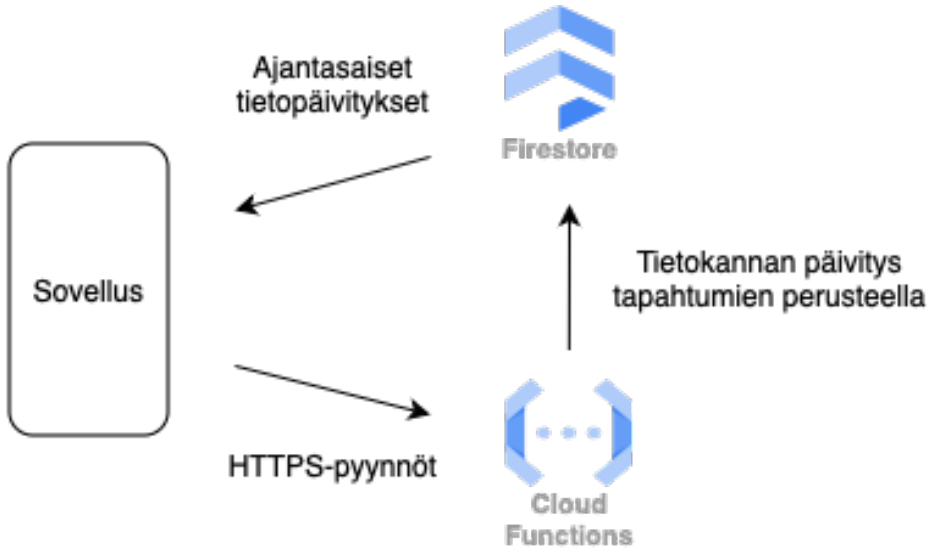
JavaScript tunnetaan ensisijaisesti käyttöliittymien ohjelmointikielenä, mutta Node.js:n myötä siitä on tullut suosittu myös taustajärjestelmien kehityksessä. Taustajärjestelmien kehityksellä, eli backend-kehityksellä, tarkoitetaan verkkosivuston tai sovelluksen taustalla tapahtuvaa toimintaa, joka mahdollistaa sivuston toimivuuden ja reagoinnin käyttäjän toimintoihin. JavaScriptin käyttö sekä käyttöliittymien että taustajärjestelmien kehittämisessä mahdollistaa koodin uudelleenkäytön, koska kehittäjät voivat jakaa koodia asiakkaan ja palvelimen välillä. Lisäksi JavaScriptin asynkroninen ohjelmointimalli mahdollistaa taustasovellusten käsitellä useita tehtäviä samanaikaisesti, mikä parantaa suorituskykyä ja reagointikykyä. JavaScriptillä on laaja ja elinvoimainen ekosysteemi, joka sisältää runsaasti kirjastoja, kehyksiä ja työkaluja erityisesti taustakehitystä varten. (JavaScript for backend development 2024.)

3.3 Firebase

Googlen kehittämä Firebase on suosittu Backend-as-a-Service (BaaS) -alusta, joka on suunniteltu verkko-, Android- ja iOS-sovellusten kehittämiseen. Se tarjoaa laajan valikoiman työkaluja ja palveluita, joiden avulla kehittäjät voivat rakentaa, hallita ja skaalata verkko- ja mobiilisovelluksia nopeasti ja tehokkaasti. Firebase on suunniteltu yksinkertaistamaan kehitysprosessia tarjoamalla kattavan valikoiman pilvipohjaisia ominaisuuksia. Sen monipuoliset palvelut voivat nopeuttaa kehitystyötä, parantaa käyttäjäkokemusta ja helpottaa taustajärjestelmän hallintaa. (Barochiya 2023.) Firebasella on useita eri palveluita, mutta tässä projektissa keskityimme kahteen niistä. Kuviossa 3 kuvataan sovelluksen ja Firebasen välistä vuorovaikutusta.

Cloud Firestore on tietokanta mobiili-, web- ja palvelinkehitykseen. Se pitää tiedot synkronoituna reaaliaikaisten kuuntelijoiden avulla sekä tukee offline-käyttöä mobiili- ja verkkosovelluksissa, mikä mahdollistaa sovelluksen sujuvan toiminnan myös ilman verkkoyhteyttä. Lisäksi Cloud Firestore integroituu saumattomasti muihin Firebase- ja Google Cloud -tuotteisiin, kuten Cloud Functions -palveluun. (Cloud Firestore 2025.)

Cloud Functions for Firebase on palvelimeton viitekehys, jonka avulla voidaan suorittaa taustakoodia automaattisesti esimerkiksi taustatapahtumien tai HTTPS-pyyntöjen (Hypertext Transfer Protocol Secure) perusteella. Käyttäjän ei tarvitse huolehtia omien palvelimien hallinnasta. Tässä projektissa käytetty JavaScript on yksi kolmesta ohjelmointikielestä, joita Cloud Functions tukee. (Cloud Functions for Firebase 2024.)



KUVIO 3. Sovelluksen ja Firebasen välinen vuorovaikutus.

3.4 React Native

React Native on Metan (entinen Facebook) kehittämä kirjasto mobiilisovellusten rakentamiseen. Se sai alkunsa Facebookin sisäisenä hackathonprojektina kesällä 2013 ja julkaistiin avoimena lähdekoodina vuonna 2015. Verkkokehityksessä käytettävä React oli niin suosittu, että alettiin miettiä, miksei sitä voisi hyödyntää myös mobiilisovelluksissa. (Boduch & Sakhniuk 2024.) React Nativen suorituskyky oli julkaisunsa aikaan huomattavasti parempi kuin sen kilpailijoilla (esim. Ionic ja Cordova). Lisäksi React Nativen avulla sovellusten kehitys oli paljon nopeampaa verrattuna erillisten Android- ja iOS-sovellusten luomiseen. (Kuttig 2022.)

Perinteisesti mobiilisovelluksen kehittäminen sekä Androidille että iOS:lle on vaatinut kahden eri ohjelmointikielen opettelua. Androidilla käytetään Javaa tai Kotlinia, kun taas iOS-sovellukset tehdään Objective-C:llä tai Swiftillä. React Native

mahdollistaa sovellusten kehittämisen yhdellä koodipohjalla, joka toimii molemmilla alustoilla. (Boduch & Sakhniuk 2024.) Julkaisunsa jälkeen React Native on kehittynyt merkittävästi, ja samalla markkinoille on tullut uusia kilpailijoita, kuten Flutter ja Kotlin Multiplatform Mobile (Kuttig 2022).

React Native hyödyntää Reactia käyttöliittymien rakentamiseen, mutta selainympäristön DOM:n (Document Object Model) sijaan se renderöi suoraan mobiililaitteiden natiiveihin käyttöliittymäkomponentteihin. Se kommunikoi mobiilikäyttöjärjestelmän rajapintojen kanssa asynkronisten kutsujen avulla, hyödyntäen JavaScript-moottoria. (Boduch & Sakhniuk 2024.)

Vuonna 2022 React Native oli suositumpi kuin koskaan ja huomattavasti kehittäjäystävällisempi kuin alkuvuosinaan. Se ei ole rajoittunut vain iOS- ja Android-sovelluksiin, vaan sitä voidaan käyttää myös macOS-, Windows-, web- ja VR-sovelluksissa sekä muilla alustoilla. (Kuttig 2022.) Sen periaate ei ole "kirjoita kerran, suorita kaikkialla", vaan "opettele kerran, kirjoita kaikkialle", mikä tarkoittaa, että kehittäjät voivat hyödyntää alustakohtaisia ominaisuuksia parantaakseen sovellusten käyttökokemusta (Boduch & Sakhniuk 2024). Vuoden 2024 State of React Native -kyselyn mukaan React Native on jatkanut kehittymistään ja vakiinnuttanut asemansa johtavana monialustaisena mobiilikehyksenä, erityisesti Expon ja uusien arkkitehtonisten ominaisuuksien myötä (Bukowski, B. 2024).

3.5 Vue.js

Vue.js, lyhyemmin Vue, on avoimen lähdekoodin JavaScript-kehys, jonka Evan You kehitti vuonna 2014 vaihtoehdoksi raskaammille kehyksille kuten AngularJS ja React (Corbo 2022). Se kattaa suurimman osan käyttöliittymien kehityksessä, eli frontend-kehityksessä, tarvittavista yleisistä ominaisuuksista. Vue on suunniteltu joustavaksi ja asteittain omaksuttavaksi, sillä verkkokehitys on monimuotoista, ja verkkosovellukset voivat vaihdella merkittävästi sekä muodoltaan että laajuudeltaan. (Vue.js. n.d.)

Vue käyttää kaksisuuntaista tiedonsidontaa varmistaakseen, että sovelluksen eri osat, jotka käyttävät samaa dataa, pysyvät ajan tasalla ja hyödyntävät aina uusinta tietoa. Se on ensisijaisesti suunniteltu käyttöliittymien kehittämiseen ja voidaan helposti integroida osaksi mitä tahansa JavaScript-projektia tehostamaan käyttöliittymäsuunnittelua ja -kehitystä. (Corbo 2022.)

4 TOTEUTUS

Toimeksiantajayrityksellä on käytössään taustajärjestelmä, joka hallinnoi käyttäjätietoja, keikkoja ja asiakastietoja sekä tallentaa ja päivittää niitä reaaliaikaisesti. Käyttöliittymä puolestaan vastaa mobiilisovelluksen sekä hallinta- ja tilausjärjestelmän näkymien esittämisestä käyttäjille. Taustajärjestelmä ja käyttöliittymä kommunikoivat keskenään Firebase-alustan kautta.

Toteutus on jaettu kahteen pääosaan: taustajärjestelmään ja käyttöliittymään. Taustajärjestelmä vastaa integraatiosta Signicatin rajapintaan, tietojen käsittelystä ja tallentamisesta sekä digitaalisen allekirjoituksen luomisesta. Toteutuksessa on hyödynnetty Firebase Cloud Functionsia, joka varmistaa taustajärjestelmän sujuvan kommunikoinnin käyttöliittymän sekä Signicatin rajapinnan kanssa ja reagoi käyttäjän toimintoihin. Käyttöliittymä puolestaan tarjoaa käyttäjälle visuaalisen käyttöliittymän, jonka kautta hän voi suorittaa toimintoja.

Toteutuksessa hyödynnetään HTTPS-protokollaa, joka mahdollistaa viestinnän asiakkaiden ja palvelimien välillä. HTTPS toimii pyyntö-vastausmallilla, jossa asiakas lähettää pyynnön ja palvelin vastaa siihen. Esimerkiksi tässä toteutuksessa käytettyä POST-pyyntöä käytetään tietojen lähettämiseen palvelimelle, erityisesti resurssien luomiseen ja päivittämiseen. (W3Schools n.d.) Käyttöliittymä tekee POST-pyyntöjä Firebase Cloud Functions -päätepisteihin ja kommunikoi näin taustajärjestelmän kanssa. Lisäksi taustajärjestelmä tekee POST-pyyntöjä Signicatin rajapintaan käyttäen samaa HTTPS-protokollaa.

Projektin pääfokus oli taustajärjestelmän toteuttamisessa ja integraatiossa Signicatin Authentication REST APIin. Tavoitteena oli varmistaa, että Signicatin tarjoama vahva tunnistautuminen toimii saumattomasti osana toimeksiantajayrityksen järjestelmää. Ratkaisu tallentaa saadut tiedot Firestoreen ja luo digitaalisen allekirjoituksen dokumentille.

4.1 Taustajärjestelmän toteutus

Taustajärjestelmän keskeiset tehtävät ovat Signicatin vahvan tunnistautumisen rajapinnan integrointi, dokumenttien hallinta sekä digitaalisen allekirjoituksen luominen. Signicatin Authentication REST API on integroitu järjestelmään siten, että taustajärjestelmän avulla voidaan hallita käyttäjän tunnistautumista suomalaisilla pankkitunnuksilla tai Mobiilivarmenteella. Käyttäjä aloittaa tunnistautumisprosessin käyttöliittymässä, ja taustajärjestelmä huolehtii yhteydenpidosta Signicatin rajapintaan. Taustajärjestelmä luo tarvittavan tunnistautumisistunnon, käsittelee Signicatilta saadut vastaukset ja varmistaa tunnistautumisen onnistumisen. Kun prosessi on suoritettu, tunnistautumistiedot tallennetaan Firestoreen.

Taustajärjestelmän vastuulla on myös virheenkäsittely, kuten mahdollisten tunnistautumisvirheiden tai tietojen tallentamiseen liittyvien ongelmien käsittely. Mikäli virhe ilmenee, taustajärjestelmä palauttaa virheen tiedon käyttöliittymälle, joka näyttää käyttäjälle selkeän virheilmoituksen. Tämä varmistaa, että järjestelmä toimii luotettavasti ja käyttäjäkokemus on sujuva, vaikka tunnistautumisen aikana ilmenisi ongelmia.

4.1.1 Vahva tunnistautuminen Signicat-rajapinnan avulla

Signicat tarjoaa tähän projektiin vahvaan tunnistautumiseen rajapinnan nimeltä Authentication REST API. Rajapinnan käyttämiseksi toimeksiantajayritykselle on luotu käyttötili Signicatiin, mikä on tehty aiemmin, ja minulle on myönnetty tarvittavat oikeudet. Tämä rajapinta mahdollistaa käyttäjien tunnistautumisen suomalaisilla pankkitunnuksilla tai Mobiilivarmenteella.

Signicatin testi käyttäjälle on luotu asiakas, jolla on oma ID ja salaisuus. Signicatiin voi tehdä POST-pyyntöjä asiakkaan tietojen sekä tilin ID:n avulla. Traficomin vaatimusten mukaisesti FTN:n (Finnish Trust Network) todentamiseen on käytettävä Full Message-Level Encryption (MLE) -salausta, mikä tarkoittaa, että salattujen vastausten vastaanottaminen Signicatilta on pakollista. Kuviossa 2 kuvataan tietojen salaamista ja purkamista tunnistautumisprosessissa.

Toteutuksessa suurin osa koostuu omasta työstä, ja osittain se sisältää integraatiota Signicatin palveluun, joka tukee käyttäjän tunnistautumista suomalaisilla pankkitunnuksilla tai Mobiilivarmenteella. Taustajärjestelmässä on luotu omat funktiot, jotka kommunikoivat Signicatin Authentication REST API:n kanssa. Toteutus kattaa muun muassa salauksen purkamisen, tietojen käsittelyn ja virheenkäsittelyn, jotka mahdollistavat Signicatin palauttaman salatun datan hyödyntämisen. Lisäksi istunnon luominen ja käyttäjätietojen hakeminen on osa toteutusta, joka hyödyntää Signicatin rajapintaa.

Tunnistautumisen prosessi etenee seuraavasti:

1. **Käyttöoikeustunnuksen hakeminen:** Ensin haetaan Signicatista käyttöoikeustunnus, jotta voimme tehdä POST-pyynnön istunnon aloittamiseksi.
2. **Istunnon aloittaminen:** Kun istunto on aloitettu, järjestelmä palauttaa meille salatun viestin. Viestin salaus täytyy purkaa, jotta saamme viestin sisällön selkokielelle.
3. **Käyttäjä ohjaaminen tunnistautumiseen:** Kun viestin salaus on purettu, saamme vastauksesta autentikaatio-URL-osoitteen (Uniform Resource Locator), jonne ohjaamme käyttäjän tunnistautumaan.
4. **Istunnon tietojen tarkastelu tunnistautumisen jälkeen:** Kun käyttäjä on tunnistautunut, teemme GET-pyynnön nähdäksemme istunnon tiedot. Tiedot tulevat taas salattuina, joten meidän tulee purkaa ne.
5. **Istunnon tietojen tarkastelu:** Mikäli istunto on onnistunut, saamme pyydetyt käyttäjätiedot, kuten nimen ja henkilötunnuksen.

KUVIO 4. Tietojen salaaminen ja purkaminen tunnistautumisprosessissa.

4.1.2 Digitaalisen allekirjoituksen luominen

Digitaalisen allekirjoituksen luomiseen käytettiin Node.js:n sisäänrakennettua crypto-moduulia ja sen createSign-metodia. Tämä metodi luo ja palauttaa Sign-

objektin, joka käyttää annettua algoritmia. (Crypto n.d.) Tässä tapauksessa algoritmina käytettiin SHA-256-hajautusalgoritmia (Secure Hash Algorithm 256-bit), joka kuuluu SHA-2-algoritmien perheeseen. Nimen mukaisesti 256 viittaa siihen, että luotava hajautusarvo on aina 256 bittiä pitkä, riippumatta syötetyn tiedon koosta. (SHA-256 and SHA-3 2024.)

Digitaalisen allekirjoituksen prosessi etenee seuraavasti:

1. **Dokumentin muuntaminen:** Allekirjoitettava dokumentti muunnetaan ensin merkkijonoksi, ja sitten binaaridataksi, sillä crypto.createSign-metodille ei voi syöttää suoraan merkkijonoa tai objektia.
2. **Allekirjoituksen luonti:** Kun allekirjoitus on luotu, se muunnetaan Base64-muotoon.
3. **URL-yhteensopivuuden varmistaminen:** Lopuksi Base64-muotoista allekirjoitusta muokataan, jotta se on URL-yhteensopiva. Tämä tarkoittaa tiettyjen merkkien muuttamisen ja välilyöntien poistamisen.

4.2 Tietoturvan varmistaminen

Tietoturvan varmistaminen on oleellinen osa tätä projektia, sillä käsittelyssä on henkilötietoja. Firebaseen myönnetyt käyttöoikeudet on rajoitettu vain niille, jotka tarvitsevat pääsyn tietoihin. Tämä vähentää riskiä luvattomasta pääsystä ja parantaa järjestelmän turvallisuutta.

Projektissa käytössä oleva Firebase Cloud Functions käyttää päätepisteissään HTTPS-protokollaa, mikä varmistaa, että viestintä on salattua. HTTPS on HTTP:n suojattu versio, ja se on ensisijainen protokolla, jota käytetään tiedon lähettämiseen verkkoselaimen ja verkkosivuston välillä. HTTPS käyttää salausprotokollaa viestinnän salaamiseen ja turvallisuuden parantamiseen. Tämä on erityisen tärkeää, kun käyttäjät välittävät arkaluontoisia tietoja, kuten kirjautuessaan pankkitilille. Tämä protokolla on nimeltään Transport Layer Security (TLS), joka tunnettiin aiemmin nimellä Secure Sockets Layer (SSL). TLS suojaa viestintää käyttämällä asymmetristä julkisen avaimen infrastruktuuria. (Cloudflare n.d.)

Tietoturvan vahvistamiseksi toteutuksessa käytetään Crypto-kirjastoa, joka mahdollistaa luotettavan hajautuksen. SHA-256 hajautusalgoritmi on laajalti tunnettu ja suosittu tietoturvassa. Hajautus on prosessi, jossa raa'at tiedot hajautetaan siten, että sen palauttaminen alkuperäiseen muotoon on mahdotonta. Yksinkertaisemmin sanottuna, hajautus ottaa tietyn osan tiedoista ja syöttää sen hajautusfunktioon, joka suorittaa matemaattisia operaatioita raakatiedoilla. Tämä prosessi tuottaa hajautusjäljen, joka on palautumaton, eli alkuperäisiä tietoja ei voida palauttaa millään tavalla. (SHA-256 and SHA-3 2024.)

Toteutuksessa on käytetty RSA avainpareja, jotka on luotu käyttäen OpenSSL-työkalua. RSA:n avulla varmistetaan digitaalisen allekirjoituksen turvallisuus ja luotettavuus yhdessä hajautusalgoritmin kanssa. Tämä suojaa asiakirjoja luvattomalta muokkaukselta ja takaa niiden eheyden.

4.3 Käyttöliittymän toteutus

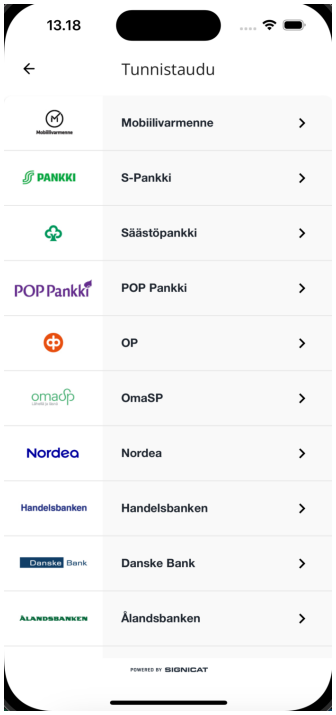
Käyttöliittymä vastaa sen esittämisestä käyttäjälle ja käyttäjän toimenpiteisiin reagoinnista. Toimeksiantajayrityksellä on käytössään mobiilisovellus, joka on suunnattu keikkatyöläisille. Sovellus mahdollistaa muun muassa työvuorojen varaamisen ja vahvistamisen. Sovellus on rakennettu React Nativea hyödyntäen, mikä mahdollistaa sovelluksen tehokkaan toiminnan niin Android- kuin iOS-laitteilla.

Lisäksi yrityksellä on asiakkaille suunnattu hallinta- ja tilausjärjestelmä, joka on toteutettu Vue.js:llä. Tämä selainpohjainen järjestelmä tarjoaa asiakkaille selkeän ja yksinkertaisen käyttöliittymän tilausten tekemiseen ja hallintaan. Vue.js:n käyttö mahdollistaa joustavan ja ajantasaisen käyttöliittymän.

Käyttöliittymien ulkoasuihin ei tarvinnut tehdä suuria muutoksia, sillä ne perustuvat toimeksiantajayrityksen ennalta määrittelemiin tyyleihin. Käyttöliittymät on suunniteltu tarjoamaan käyttäjille miellyttävän ja selkeän käyttökokemuksen. Selkeät palautteet varmistavat, että järjestelmän toiminnot ovat käyttäjille helposti ymmärrettävissä.

4.3.1 Mobiilisovellus

Mobiilisovelluksessa työntekijät voivat varata ja vahvistaa vuorojaan. Tässä opinnäytetyössä sovellukseen luotiin mahdollisuus allekirjoittaa työsopimus. Työntekijä tunnistautuu ensin pankkitunnuksilla tai Mobiilivarmenteella, jonka jälkeen hän voi tarkastella työsopimustaan, antaa tilinumeronsa ja allekirjoittaa sopimuksen sähköisesti. Kuviossa 5 kuvataan käyttäjän, sovelluksen ja taustajärjestelmän välistä vuorovaikutusta.



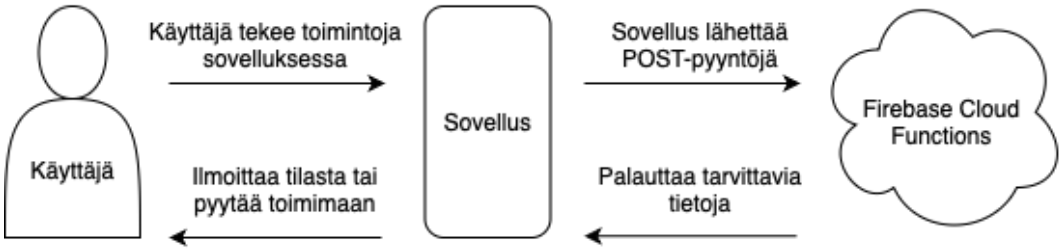
KUVA 1. Signicatin vahva tunnistautuminen mobiilisovelluksessa.

Mobiilisovelluksen keskustelu taustajärjestelmän kanssa etenee seuraavasti:

1. **Allekirjoittamisen aloitus:**
    - Käyttäjä painaa banneria, jossa lukee "Allekirjoita työsopimus".
    - Tämä lähettää POST-pyynnön päätepisteeseen, joka luo uuden dokumentin Firestoreen, ja samalla toiseen endpointiin, joka hakee Signicatista käyttöoikeustunnuksen ja käynnistää uuden istunnon.
    - Päätepiste palauttaa käyttöliittymälle autentikaatio-URL-osoitteen, johon käyttäjä ohjataan WebView:n sisällä.
2. **Tunnistautuminen:**
    - Käyttäjä tunnistautuu autentikaatio-URL:n kautta.

- Mikäli tunnistautuminen onnistuu, käyttöliittymä tekee POST-pyynnön päätepisteeseen, joka hakee istunnon tiedot.
- Tämän jälkeen käyttäjä ohjataan SignPdf-sivulle.

3. **IBAN:n täyttäminen:**
    - SignPdf-sivulla käyttäjälle näytetään allekirjoitettava PDF-dokumentti.
    - PDF:n alapuolella pyydetään käyttäjää täyttämään IBAN-numeronsa (International Bank Account Number).
4. **IBAN:n validointi ja allekirjoittaminen:**
    - Kun käyttäjä on täyttänyt tilinumeron, MaskInput-komponentti tarkistaa syötteen eli varmistaa, että IBAN sisältää vain numeroita. Syötteen pituus tarkistetaan erillisellä isValid-metodilla, joka varmistaa, että IBAN on tarpeeksi pitkä.
    - Kun syöte on hyväksytty sekä merkkien että pituuden osalta, käyttäjä voi painaa "Allekirjoita" -nappia.
    - Painallus lähettää POST-pyynnön päätepisteeseen, joka luo digitaalisen allekirjoituksen ja päivittää Firestore-dokumenttiin käyttäjän nimen selkokielellä sekä henkilötunnuksen ja IBAN-numeron salattuina.

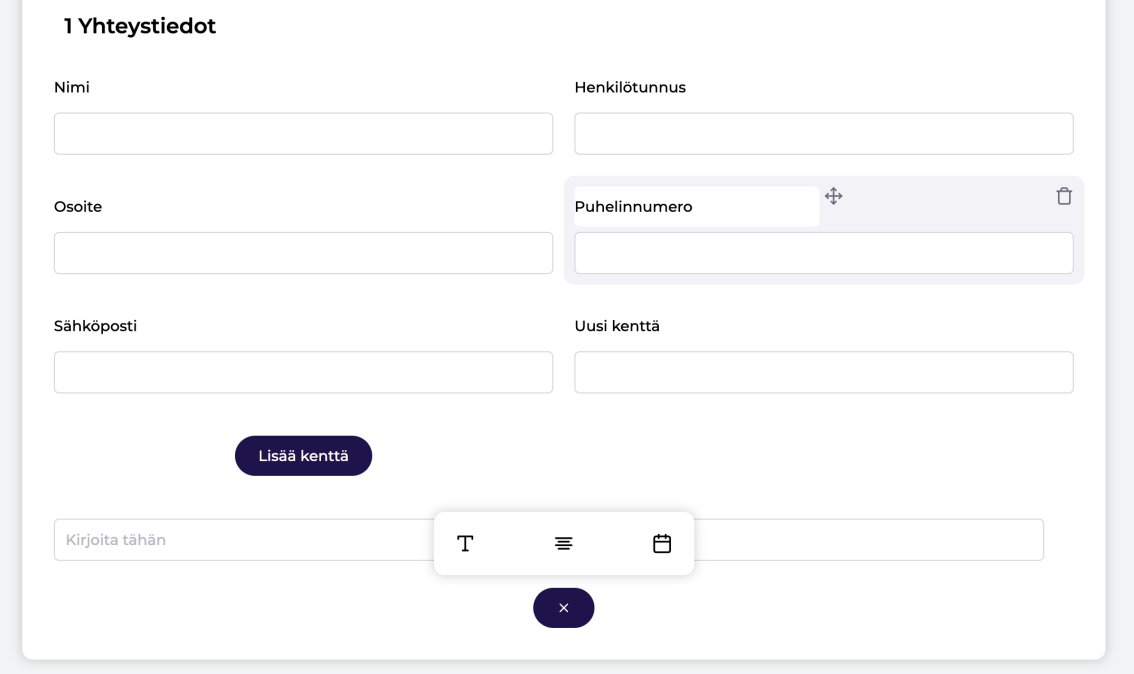


KUVIO 5. Käyttäjän, sovelluksen ja taustajärjestelmän välinen vuorovaikutus.

4.3.2 Hallintajärjestelmä

Hallintajärjestelmä, joka on rakennettu käyttäen Vue.js:ää, toimii toimeksiantajayrityksen hallinta- ja tilausjärjestelmänä asiakkaille. Tavoitteena oli kehittää työkalu, jolla sopimuksia voidaan luoda dynaamisesti. Sivun ulkoasu mukailee toimeksiantajayrityksen olemassa olevia sivuja, ja siinä on hyödynnetty toimeksiantajan valmista koodia, integroiden osia muilta sivuilta osaksi omaa toteutusta.

Kuten kuvasta 2 nähdään, käyttöliittymä mahdollistaa uusien kenttien ja tekstikenttien lisäämisen sekä niiden otsikoiden muokkaamisen. Kenttiä voi siirtää samanlaisten kenttien kesken ja poistaa tarpeen mukaan. Lisäksi käyttäjä voi lisätä päivämääräkentän. Tässä vaiheessa keskityttiin tarjoamaan joustava tapa luoda sopimuksia, ja jatkokehityksessä voidaan laajentaa toiminnallisuuksia, kuten mahdollisuutta siirtää kenttiä vapaasti sivulla sekä kehittää tallennuslogiikka ja PDF-vienti.



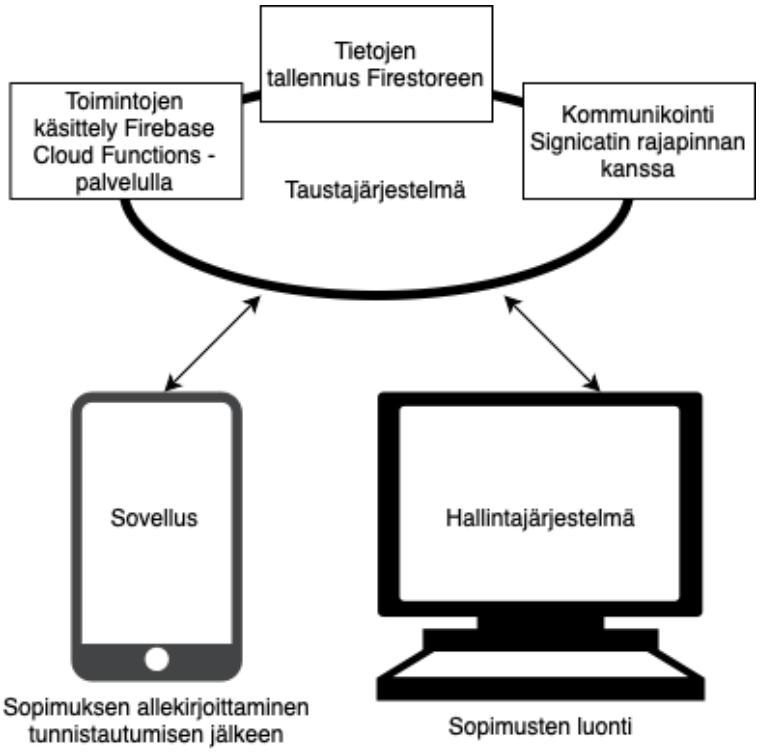
KUVA 2. Sopimuksen luontityökalun käyttöliittymä.

4.4 Toteutuksen käyttöönotto ja käyttöympäristö

Toteutuksen käyttöönotto edellyttää ensisijaisesti sopimusten luontityökalun viimeistelyä, sillä se on olennainen osa käyttöliittymää, jonka avulla voidaan luoda sopimuksia dynaamisesti. Toteutus toimii kahdessa pääasiallisessa ympäristössä: mobiilisovelluksessa ja selainpohjaisessa hallintajärjestelmässä. Mobiilisovellus on ladattavissa sovelluskaupoista ja se on yhteensopiva sekä Android- että iOS-laitteiden kanssa. Sovelluksen avulla keikkatyöntekijät voivat hallita työ-

vuorojaan, tarkastella työsopimuksia ja allekirjoittaa ne digitaalisesti. Hallintajärjestelmä on selainpohjainen sovellus, joka tarjoaa asiakasyrityksille työkalut tilausten tekemiseen, hallintaan sekä sopimusten luontiin ja hallintaan.

Molemmat ympäristöt pyörivät Firebase-pilvipalvelussa. Järjestelmän tiedot tallennetaan Firestore-tietokantaan, ja taustalla käytetään Firebase Cloud Functions -palvelua tietojen käsittelyyn ja toimintoihin. Tämä pilvipohjainen ratkaisu varmistaa, että kaikki toiminnot, kuten käyttäjien vuorovaikutus ja tietojen käsittely, tapahtuvat Firebase-palveluissa ilman tarvetta erillisille fyysisille palvelimille. Tämä mahdollistaa järjestelmän sujuvan skaalautuvuuden ja nopean käytettävyyden. Kuviossa 6 havainnollistetaan, kuinka käyttöympäristöt vuorovaikuttavat taustajärjestelmän kanssa Firebase-alustan kautta.



KUVIO 6. Toteutuksen käyttöympäristöjen vuorovaikutus taustajärjestelmän kanssa.

5 POHDINTA

Toimeksiantajayritys odotti lopputuloksena rajapintaa, jonka avulla allekirjoittaja voi: 1. tunnistautua vahvasti Mobiilivarmennetta tai suomalaista pankkitunnusta käyttäen, ja 2. luoda digitaalinen allekirjoitus vahvasta tunnistautumisesta saatujen tietojen avulla. Toimeksiantajan mukaan projekti sujui vauhdikkaasti ja toteutus vastasi toimeksiannon vaatimuksia. Alkuun pääseminen oli kuitenkin haastavaa vähäisen aiemman kokemuksen vuoksi, mutta keskityin rakentamaan toteutuksen vaihe vaiheelta, aloittaen Signicatin rajapinnan integraatiosta. Signicatin laaja dokumentaatio tuntui alkuun isolta tietomäärältä, ja sen hahmottaminen vei jonkin aikaa. Käytyäni dokumentaatiota huolellisesti läpi ja päästyäni testaamaan yhteyksiä käytännössä, käsitys integraation tekemisestä selkeytyi, ja työ alkoi edetä. Projektin eteneminen helpottui vaiheittain: pienempien osakokonaisuuksien toteuttaminen ja testaaminen auttoivat hahmottamaan kokonaisuutta paremmin.

Projektissa oli myös mahdollisuus laajentaa kokonaisuutta toteuttamalla sopimusten luontiin tarkoitettu työkalu, jota ehdittiinkin jo hieman aloittaa. Toteutusta testattiin kuitenkin vain kehitysympäristössä, ja sen käyttöönotto tuotantoympäristössä edellyttää vielä jatkokehitystä. Jatkokehityksessä sopimusten luontiin tarkoitetun työkalun saattaminen loppuun asti on avainasemassa, jotta rajapinta voidaan ottaa käyttöön osana yrityksen laajempaa järjestelmää.

Tämä projekti on syventänyt ymmärrystäni pilvipohjaisista ratkaisuista sekä käyttöliittymien ja erityisesti taustajärjestelmien kehittämisestä. Olen saanut arvokasta käytännön kokemusta Signicatin rajapinnan integroinnista ja oppinut, kuinka tärkeää on yhdistää eri järjestelmät saumattomaksi kokonaisuudeksi. Kokemus on vahvistanut ohjelmistokehityksen osaamistani ja antanut valmiudet työskennellä entistä monimutkaisempien integraatioiden parissa tulevaisuudessa.

LÄHTEET

Käytetty ChatGPT kirjoituskielen sujuvuuteen

Aumasson, J.-P. 2025. Serious cryptography: a practical introduction to modern encryption. Second edition. E-kirja. San Francisco, CA: No Starch Press. Viitattu 12.2.2025. Vaatii käyttöoikeuden. https://www.oreilly.com/library/view/serious-cryptography-2nd/9781098182472/

Authentication REST API. 2025. Signicat. Verkkosivu. Viitattu 17.2.2025. https://developer.signicat.com/docs/eid-hub/authentication-api/

Barochiya, N. 2023. Medium. Verkkosivu. Viitattu 17.2.2025. https://medium.com/@nandeepbarochiya/unlocking-the-power-of-firebase-a-comprehensive-guide-to-googles-baas-platform-827ed88567f2

Boduch, A. & Sakhniuk, M. 2024. React and React Native - Fifth Edition. E-kirja. Packt Publishing. Viitattu 13.2.2025. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/react-and-react/9781805127307/

Bukowski, B. 2024. State of React Native 2024. Verkkosivu. Viitattu 19.3.2025. https://results.stateofreactnative.com/en-US/

Cloud Firestore. 2025. Google. Verkkosivu. Viitattu 20.1.2025. https://firebase.google.com/docs/firestore

Cloud Functions for Firebase. 2024. Google. Verkkosivu. Viitattu 10.12.2024. https://firebase.google.com/docs/functions

Cloudflare. n.d. What is HTTPS? Verkkosivu. Viitattu 18.2.2025. https://www.cloudflare.com/learning/ssl/what-is-https/

Corbo, A. 2022. What is Vue JS? Builtin. Verkkosivu. Viitattu 13.2.2025. https://builtin.com/software-engineering-perspectives/vue-js

Crypto. n.d. Nodejs. Verkkosivu. Viitattu 7.1.2025. https://nodejs.org/api/crypto.html#cryptocreatesignalgorithm-options

DocuSign. 2024. Digital Signature vs. Electronic Signature: When to Use Each. Verkkosivu. Viitattu 17.2.2025. https://www.docusign.com/blog/digital-signature-vs-electronic-signature

eID Hub. 2024. Signicat. Verkkosivu. Viitattu 27.12.2024. https://developer.signicat.com/docs/eid-hub/

European Commission. 2024. eIDAS Regulation. Verkkosivu. Viitattu 17.2.2025. https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation

Flanagan, D. 2020. JavaScript: Master the World's Most-Used Programming Language. 7th edition. E-kirja. Sebastopol: O'Reilly Media, Incorporated. Viitattu

14.2.2025. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/javascript-the-definitive/9781491952016/

Fortinet. n.d. Cryptography Definition. Verkkosivu. Viitattu 17.12.2024. https://www.fortinet.com/resources/cyberglossary/what-is-cryptography

Guide to digital signatures. 2023. Signicat. Verkkosivu. Viitattu 23.12.2024. https://www.signicat.com/blog/guide-to-digital-signatures

Introduction to Node.js. n.d. Nodejs. Verkkosivu. Viitattu 10.12.2024. https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

JavaScript for backend development. 2024. GeeksforGeeks. Verkkosivu. Viitattu 30.12.2024. https://www.geeksforgeeks.org/javascript-backend-basics/

Kuttig, A. B. 2022. Professional React Native. E-kirja. United Kingdom: Packt Publishing, Limited. Viitattu 13.2.2025. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/professional-react-native/9781800563681/

Kyberturvallisuuskeskus. 2024. Sähköinen tunnistaminen. Verkkosivu. Viitattu 5.12.2024. https://www.kyberturvallisuuskeskus.fi/fi/toimintamme/saantely-ja-valvonta/sahkoinen-tunnistaminen

Signicat n.d. Etusivu. Verkkosivu. Viitattu 27.12.2024. https://www.signicat.com/fi

Signicat – Tietoa meistä n.d. Tietoa meistä. Verkkosivu. Viitattu 5.12.2024. https://www.signicat.com/fi/tietoa-meista

SHA-256 and SHA-3. 2024. GeeksforGeeks. Verkkosivu. Viitattu 7.1.2025. https://www.geeksforgeeks.org/sha-256-and-sha-3/

Vue.js. n.d. Introduction. Verkkosivu. Viitattu 13.2.2025. https://vuejs.org/guide/introduction.html

W3Schools. n.d. HTTP Request Methods. Verkkosivu. Viitattu 18.2.2025. https://www.w3schools.com/TAgs/ref_httpmethods.asp