

alaviiva

Metropolia Ammattikorkeakoulu

Tietotekniikan koulutusohjelma

Ohjelmistontuotanto

Innovaatioprojekti

29.12.2020

Innoryhmä 1: Esa Ryömä – Hannu Korhonen – Toni Naumanen – Timo Harju – Mika Iso-Ikala

Innovaatioprojekti: Kestävä keikkatyö

Dokumentaatio

Sisältö

[Toiminnallinen määrittely 1](#_Toc62422095)

[1. Johdanto 1](#_Toc62422096)

[1.1 Tarkoitus 1](#_Toc62422097)

[1.2 Tuote 1](#_Toc62422098)

[1.3 Määritelmät, termit ja lyhenteet 1](#_Toc62422099)

[1.4 Viitteet ja muut dokumentit 1](#_Toc62422100)

[1.5 Yleiskatsaus dokumenttiin 1](#_Toc62422101)

[2. Yleiskuvaus 1](#_Toc62422102)

[2.1 Ympäristö 1](#_Toc62422104)

[2.2 Toiminta 2](#_Toc62422105)

[2.3 Käyttäjät 2](#_Toc62422106)

[2.4 Oletukset ja riippuvuudet 2](#_Toc62422107)

[3. Tiedot ja tietokanta 2](#_Toc62422108)

[3.1 Tietosisältö 2](#_Toc62422110)

[3.2 Käyttöintensiteetti 3](#_Toc62422111)

[3.3 Kapasiteettivaatimukset 4](#_Toc62422112)

[3.4 Tiedostot ja asetustiedostot 4](#_Toc62422113)

[4 Ulkoiset liittymät 4](#_Toc62422114)

[4.1 Käyttöliittymä 4](#_Toc62422115)

[4.2 Ohjelmistoliittymät 8](#_Toc62422116)

[Tekninen määrittely 9](#_Toc62422117)

[1. Järjestelmän yleiskuvaus 9](#_Toc62422118)

[Testaus 11](#_Toc62422119)

[1. Johdanto 11](#_Toc62422120)

[2. Testauksen kohde ja tavoitteet 11](#_Toc62422121)

[3. Testausympäristö 11](#_Toc62422122)

[4. Testauksen organisointi ja raportointi 11](#_Toc62422123)

[5. Testausstrategia ja integraatiosuunnitelma 12](#_Toc62422124)

[6. Testattavat toiminnot 12](#_Toc62422125)

[7. Linkit 12](#_Toc62422126)

# Toiminnallinen määrittely

## Johdanto

### Tarkoitus

Kestävä keikkatyö-hankkeen päämääränä on, että työturvallisuus, työterveys ja työhyvinvointi (TTT) sekä näiden kautta tuottavuus ja kilpailukyky vahvistuvat keikkatyötä hyödyntävissä pk-yrityksissä sekä keikkatyötä välittävissä yrityksissä, erityisesti keikkatyöntekijöiden näkökulmasta. Yrityksiin syntyy kulttuuri, jossa yhdessä pidetään huolta työturvallisuudesta, työterveydestä, työhyvinvoinnista ja tuottavuudesta. Lisäksi päämääränä on työllisyyden edistäminen, työurien pidentäminen sekä työvoimatarpeen ja työntekijöiden nykyistä parempi kohtaaminen.

Itse innovaatioprojekti keskittyy luomaan kattavan pohjan verkkopohjaiselle applikaatiolle, josta on helppo lähteä jatkokehittämään toiminnallisuuksia ja sisältöä.

### Tuote

Tuotteena on tarkoitus tuottaa työntekijän arkikäyttöä varten verkkopohjainen sovellus, jonka pääpiirteinä on työhön liittyvien dokumenttien säilytys elektronisesti, työprosessin seuraaminen dynaamisen näkymän kautta, viestien lähetys sekä tarkistuslistojen tekeminen ja helppo käyttö.

Samaa sovellusta käytetään myös vuokratyöyritysten sekä käyttäjäyritysten toimintoihin ja työntekijöiden hallitsemiseen.

### Määritelmät, termit ja lyhenteet

Projektissa käytetään paljon erilaisia termejä ja lyhenteitä, jotka välttämättä ole aivan itsestään selviä sovelluksen kontekstissa:

* Vuokratyöyritys (Agency): Yritys, joka palkkaa työntekijöitä ja etsii heille työpaikkoja muista yrityksistä.
* Käyttäjäyritys (Business): Tavanomainen yritys tai liike, joka tarjoaa itse työn.

### Viitteet ja muut dokumentit

Virallisen dokumentaation lisäksi ryhmämme on valmistanut ”aloita tästä” nimisen epävirallisen dokumentaation seuraavan ryhmän perehdyttämiseen projektiin ja sen vaiheeseen.

Projektin koodidokumentaatio on tuotettu NodeJS-moduulilla *JSDoc.* Moduuli löytyy sekä front- että backendin package-json -tiedostosta, joten sen pitäisi asentua kehittäjälle samalla, kun tämä asentaa muutkin moduulit skriptillä *npm install*.

Backendin JSDocia pääsee katsomaan html-muodossa tiedostosta */doc/[moduuliryhmän kansio]/index.html*. Moduuliryhmiä on kirjoitushetkellä vain *controllers* ja *utils*. Dokumenttisivusto on tuotettu ajamalla backendin juurihakemistossa komento *jsdoc [moduuliryhmän kansio] -r -d doc,* jossa scripti hakee lähdekoodin kansiosta [moduuliryhmän kansio] (esim. /controllers/), -r käy läpi myös kansion alakansiot, ja –d:llä annetaan dokumentin kohdekansio (doc). Jos olisimme olleet fiksuja ja olisimme laittaneet kaikki moduulit yhteisen, esim. /src/-kansion sisälle omissa kansiossaan, koko dokumentin voisi tuottaa vain komennolla *jsdoc . -r -d doc.* Mutta emme olleet ja sen refaktorointi saattaa aiheuttaa virheitä, joita emme ehdi korjamaan.

### Yleiskatsaus dokumenttiin

Ensimmäisen innovaatioprojektiryhmän dokumentaatio on suppeampi, rakenteeseen ja metadataan kohdistuva kokonaisuus. Dokumentaatiolla pyritään kattamaan

## Yleiskuvaus



### Ympäristö

JavaScript sovelluksen frontendiä ja backendiä ajetaan tällä hetkellä paikallisesti. Tietokanta on koulun tarjoamalla palvelimella.

### Toiminta

Sovelluksen käyttö aloitetaan kirjautumalla sisään (olemassa oleva käyttäjätili) tai rekisteröitymällä (uusi käyttäjä). Rekisteröinnin jälkeen käyttäjä voi kirjautua sisään sovellukseen ja etusivu täytetään oman roolin (työntekijä/vuokratyöyrityksen edustaja/käyttäjäyrityksen edustaja) mukaisesti.

Etusivu toimii navigointipisteenä, joista pääsee sovelluksen muihin toimintoihin käsiksi. Näitä ovat esimerkiksi perehdytyksen etenemisen näkymä, dokumentit tai omat käyttäjätilin tiedot ja asetukset.

Sovelluksen toivotaan olevan hyvä apuväline etenkin työntekijän arkeen, josta voi löytää helposti hakemansa tiedot, seurata työpaikan uutisointia sekä käyttää erilaisia tarkistuslistoja.

### Käyttäjät

Sovelluksessa on tarkoituksena jakaa käyttäjät kolmeen eri luokkaan ja tarjota toiminnallisuuksia sen mukaan:

Työntekijät:

Vuokratyöyrityksen edustajat:

Käyttäjäyrityksen edustajat:

### Oletukset ja riippuvuudet

## Tiedot ja tietokanta



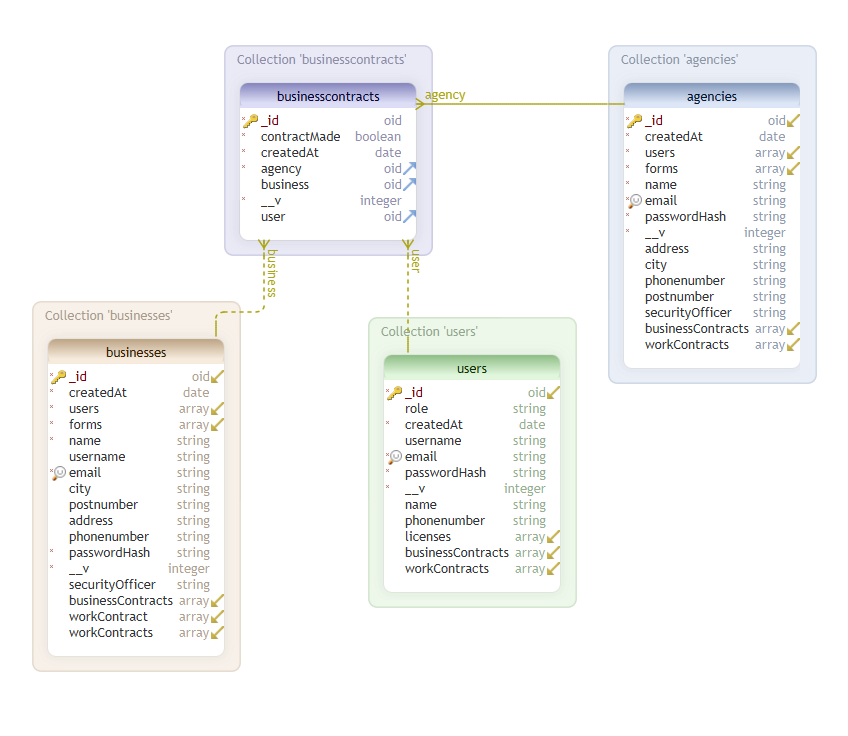
### Tietosisältö

Ohjelmiston tietosisältö muodostuu käyttäjistä, joilla on kolme eri pääroolista: Business, Agency ja Worker (Käyttäjäyritys, Vuokratyöyritys ja Työntekijä), sekä käyttäjien välisistä sopimuksista: BusinessContract:sta sekä WorkContract:sta. Kuvassa (1) tietokanta kuvattuna.

Web-sovelluksen käyttäjät voivat rekisteröityä sovelluksen etusivulla itsenäisesti ja valita itselle sopivan roolin (Käyttäjäyritys, Vuokratyöyritys ja Työntekijä). Rooli vaikuttaa käyttäjän näkymään ja toimintoihin.

BusinessContract on käyttäjäyrityksen ja vuokratyöyrityksen välinen sopimus, jonka avulla sallitaan käyttäjien lisääminen käyttäjäyritykseen vuokratyöyrityksen toimesta. Vuokratyöyritys lähettää pyynnön käyttäjäyritykselle, minkä käyttäjäyritys täytyy erikseen sallia.

WorkContract on käyttäjän ja käyttäjäyrityksen välinen sopimus, jota hallinnoi vuokratyöyritys. Sopimus pitää sisällään, missä käyttäjäyrityksessä työntekijä on, sekä kuinka pitkä työsopimus on. Käyttäjien linkittäminen käyttäjäyritykseen vaatii aktiivisen linkin käyttäjäyrityksen ja vuokratyöyrityksen BusinessContract:in avulla.



Kuva 1. Tietokanta

### Käyttöintensiteetti

Ohjelmaa tullaan käyttämään vuorokauden ympäri, joten käyttöintensiteetti tulee olemaan iso käyttäjä määrien kasvaessa. Yhtäaikaisia käyttäjiä voi olla jopa kymmeniätuhansia. Järjestelmä täytyy mitoittaa, jopa satoihin tuhansiin yhtäaikaisiin tietokantakyselyihin ja sen on suoriuduttava määritellyssä vasteajassa.

### Kapasiteettivaatimukset

Tietokannan maksimikoko tulee olemaan riippuvainen sovelluksen saamasta käyttäjämäärästä. Minimikooksi voidaan arvioida vähintään 10 käyttäjää vuoden ajan. Tämänhetkisessä prototyypissä, jossa on 25 testikäyttäjä vie loogista tilaa 295 kb. Lukuja ja kirjoituksia on todella vähän < 0.01/s, sillä ohjelmoija käyttää vain yhtä testikäyttäjää kerrallaan. Toimintojen lisääntyessä, kuten mahdollinen viestinnän mahdollisuus tulee kasvattamaan kapasiteettivaatimusta paljon. Mahdollinen ennuste tietokanta luvuille ja kirjoituksille voi pyöritä > 100 000/s ja loogisen tilan koolle yli miljoona kertaisena 295kb:een verrattuna.

### Tiedostot ja asetustiedostot

Backendin pyörittämiseen vaaditaan tiedostotyyppi (.env), johon on määritelty ympäristömuuttujat: MONGODB\_URI, PORT, TEST\_MONGODB\_URI, SECRET, SESSION\_SECRET.

MONGODB\_URI = MongoDB:n yhteyteen vaadittu URI,

PORT = Palvelimen käyttöön otettava portti

TEST\_MONGODB\_URI = Testaukseen käytettävän MongoDB URI

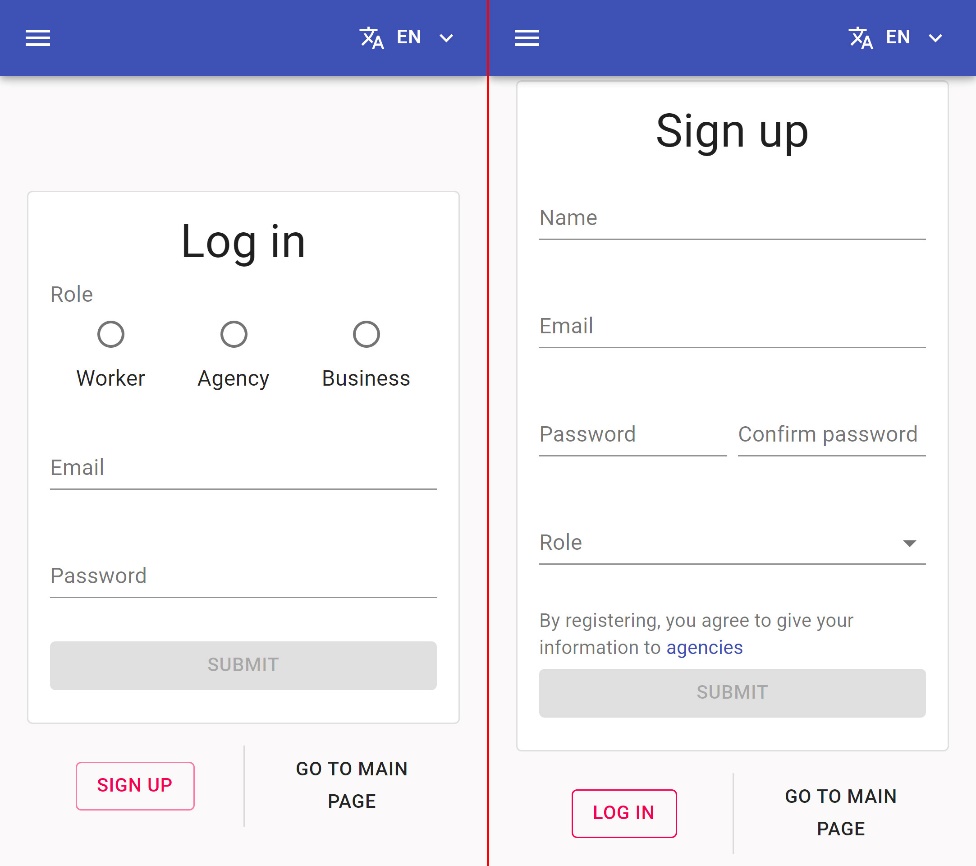
SECRET / SESSION\_SECRET= Hashin laskemista varten

1. Ulkoiset liittymät

### Käyttöliittymä

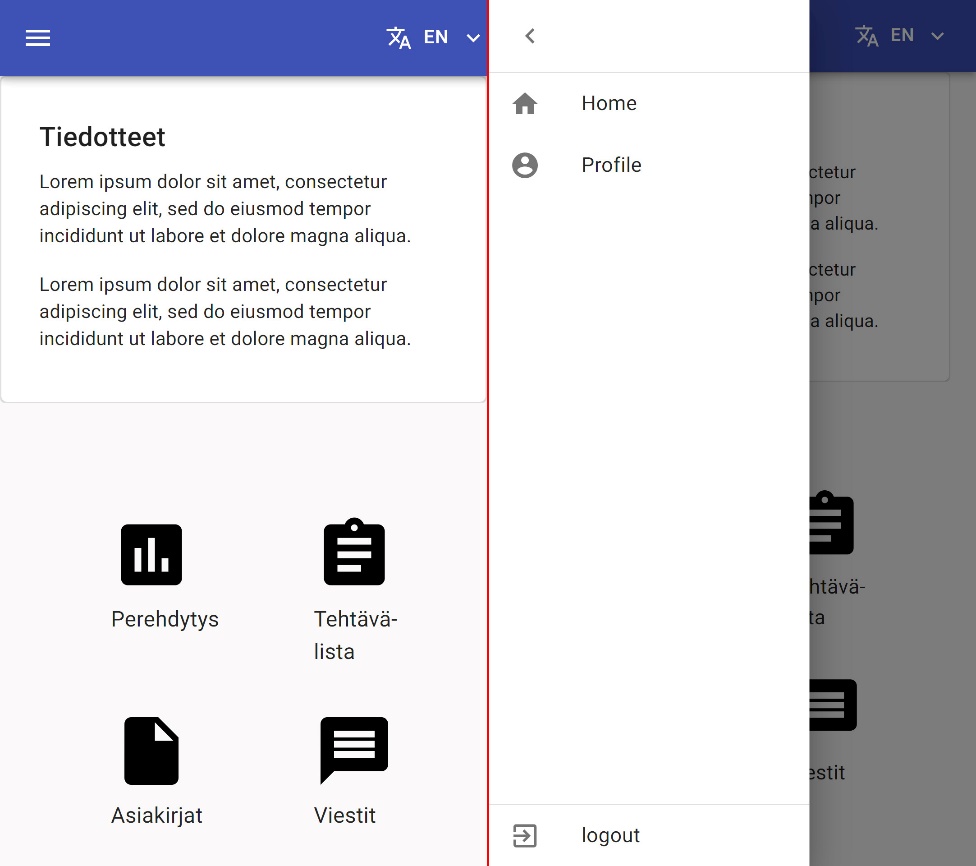
Ohjelmiston käyttöliittymä on web-pohjainen ja se on alustettu hyödyntäen ”Create React App” -pakettia. Käyttöliittymä on pyritty tekemään mahdollisimman yhteensopivaksi mobiililaiteille. Käyttöliittymän ulkoasuun (värit, muodot jne.) ei ole kiinnitetty niin paljon huomiota, sillä sen tekee ulkopuolinen graafinen suunnittelija. Kuitenkin käyttöliittymän asettelu on pyritty tekemään mahdollisimman järkevän oloiseksi, sillä asettelua on vaikeampi muuttaa jälkikäteen, vaikka sekin on tietenkin mahdollista.

Käyttöliittymän aloitussivulla tarjotaan käyttäjälle mahdollisuutta kirjautua sisään tai rekisteröityä sovelluksen käyttäjäksi (kuva 2). Käyttäjä voi myös siirtyä etusivulle vierailijana, jolloin käyttäjä näkee yksinkertaisen version sivustosta rajoitetuilla ominaisuuksilla.



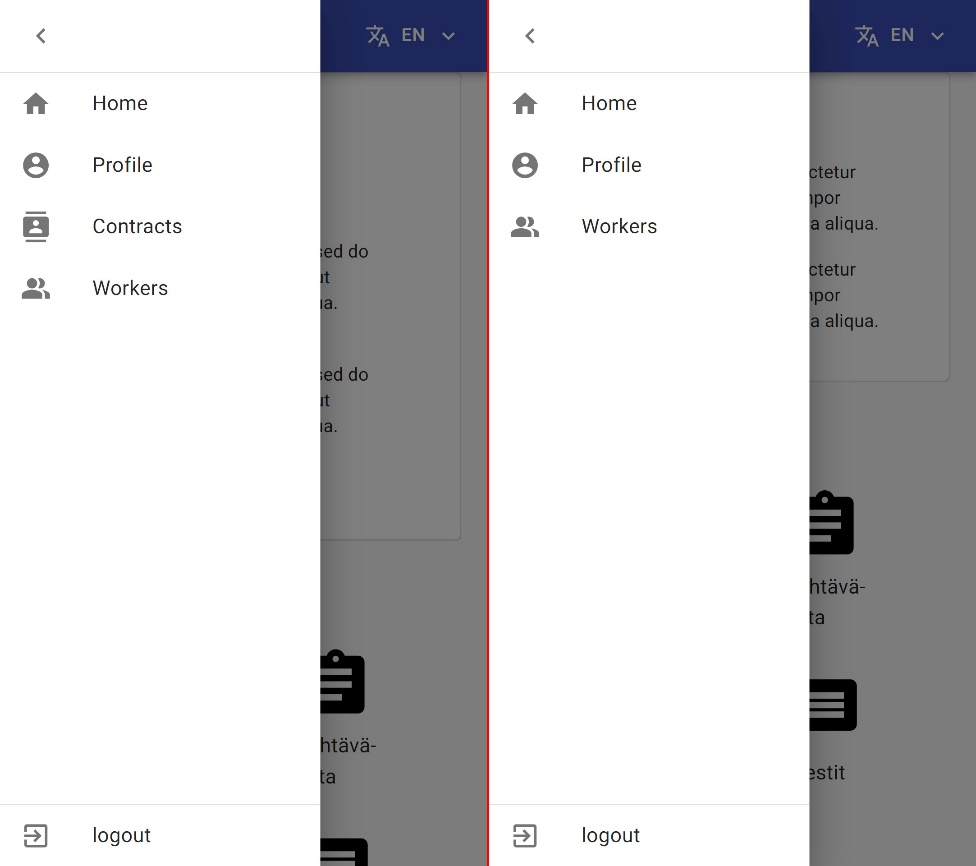
Kuva 2. Käyttöliittymän aloitussivu (kirjautumis- ja rekisteröitymislomake) mobiililaitteella.

Aloitussivulta siirryttäessä etusivulle tarjotaan käyttäjälle yleisiä tiedotteita sekä mahdollisuus siirtyä sivuston eri osiin käyttäen joko esiin vedettävää navigointivalikkoa tai etusivulla olevia painikkeita (kuva 3).



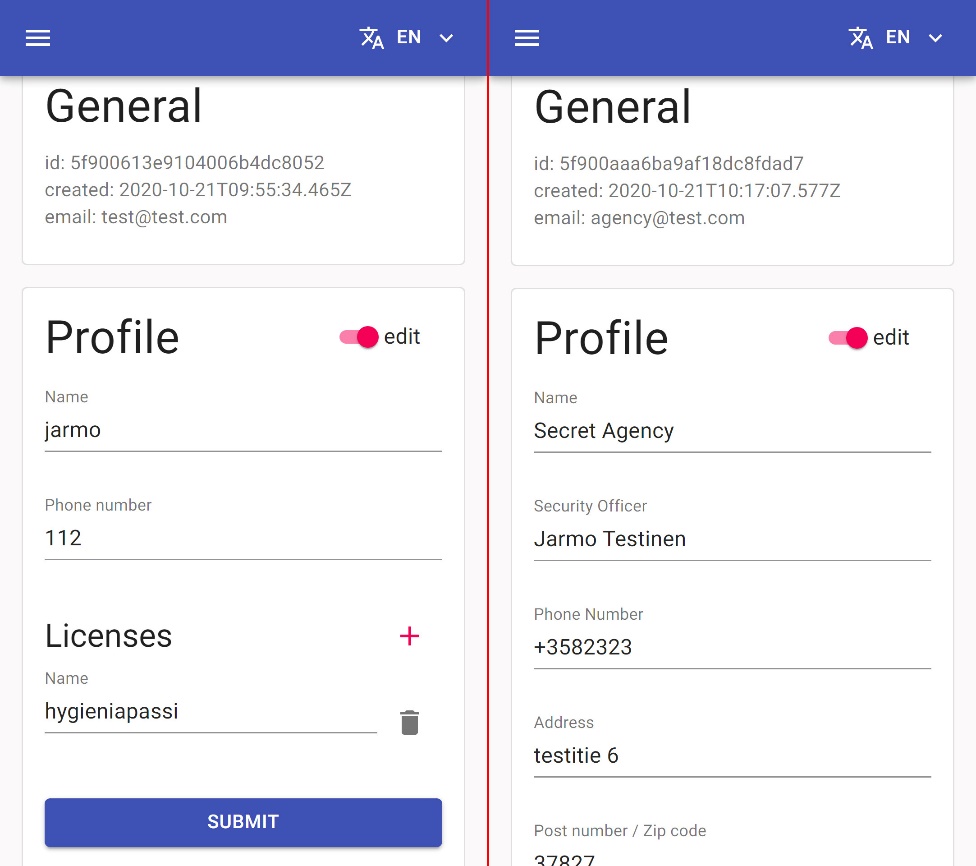
Kuva 3. Työntekijän käyttöliittymän etusivu (vasemmalla) ja esiin vedettävä navigointivalikko (oikealla).

Käyttäjälle näytettävät navigointivaihtoehdot riippuvat käyttäjän roolista. Vierailijalle näytetään vain vaihtoehdot siirtyä etusivulle tai aloitussivulle. Työntekijällä, vuokrayrityksellä ja käyttäjäyrityksellä on aiemmin mainittujen lisäksi pääsy myös mm. profiilisivulle ja perehdytyssivulle. Näiden lisäksi vuokrayrityksellä on myös pääsy sopimussivulle, jossa luodaan linkit vuokrayrityksen, käyttäjäyrityksien ja työntekijöiden välille (kuva 4).



Kuva 4. Vuokrayrityksen (vasemmalla) ja käyttäjäyrityksen (oikealla) navigointivaihtoehdot.

Profiilisivulla käyttäjän tietoja voidaan tarkastella ja muokata. Profiilisivulla näytettävät tiedot riippuvat siitä millä roolilla käyttäjä on kirjautunut sisään, mutta yleistiedot ja salasanan vaihto osiot ovat kaikille rooleille samanlaisia. Työntekijän profiilisivulla on vain perustietoja kuten esimerkiksi nimi, puhelinnumero ja työpassit (kuva 4). Vuokrayrityksellä ja käyttäjäyrityksellä on profiilisivulla enemmän kenttiä kuten esimerkiksi osoitetiedot ja turvallisuusvastaavan nimi (kuva 5).



Kuva 5. Työntekijän profiilisivu (vasemmalla) ja vuokrayrityksen profiilisivu (oikealla).

Käyttöliittymään on luotu myös muita sivuja, mutta ne jätetään vielä huomioimatta, sillä ne ovat joko vähäisiä sisällöltään ja toiminnallisuuksiltaan, saattavat muuttua hyvinkin paljon tai niiden toiminnallisuutta ei ehkä tarvita ollenkaan. Tällaisia sivuja ovat mm. perehdytyssivu, viestisivu ja sopimussivu.

### Ohjelmistoliittymät

Ohjelmiston palvelinpuoli tarjoaa REST rajapinnan, jonka avulla sovellukseen liittyvää dataa voidaan pyytää, lisätä, muokata ja poistaa. Sovellukseen liittyvä data sisältää esimerkiksi tietoja käyttäjätileistä sekä vuokrayrityksien ja työntekijöiden linkityksistä.

Palvelinpuolen rajapinnan avulla sovelluksen tietoa voidaan hakea, lisätä, muokata ja poistaa ilman, että tarvitsisi käyttää sovelluksen käyttöliittymää. Näitä toimintoja varten tarvitaan kuitenkin käyttäjätili sovellukseen ja kirjautumisen yhteydessä saatava ”token”, jonka avulla toiminnot voidaan valtuuttaa. Tämä voi olla hyödyllistä silloin kun sovellus halutaan integroida toiseen jo käytössä olevaan sovellukseen. Tällä hetkellä palvelinpuolen tarjoamaa rajapintaa hyödyntää vain sovelluksen oma käyttöliittymä.

# Tekninen määrittely

## Järjestelmän yleiskuvaus

Järjestelmä on web-pohjainen sovellus, jota käytetään pääasiassa mobiililaitteilla. Järjestelmään kuuluu sovelluksen selainpuoli (front-end), palvelinpuoli (back-end) ja tietokanta.

Koodausympäristönä käytetään Visual Studio Code -tekstieditoria, johon täytyy asentaa vähintään ESLint-lisäosa, jotta sovelluksen staattinen analyysi eli ”linttaus” toimisi. Tekstieditorilla ei ole merkitystä, mutta jos päättää käyttää eri tekstieditoria, niin tulisi kyseiselle editorille kuuluvat yksilölliset tiedostot lisätä gitignore-tiedostoon.

Projektin versionhallintana käytetään Git-versionhallintajärjestelmää ja versionhallintapalvelimena käytetään Githubia. Projektin frontend ja backend on tehty omiin repositorioihinsa GitHubiin.

Projekti on JavaScript pohjainen, joten tarvitaan myös Node.js ympäristöä. Käytössä on ollut sen aikainen LTS-versio (14.x) ja sen mukana tuleva NPM-paketinhallintajärjestelmä.

Projektin tietokantana käytetään MongoDB:tä ja projektin toimivuutta testaan Jenkins-palvelimen avulla. Selainpuolelle ja palvelinpuolelle on tehty omat Jenkins-prosessit, jolla testataan ”buildin” ja testien onnistumista.

#### Ohjelmistoarkkitehtuuri: Moduulit ja prosessit.

Backend REST-rajapinta

Rajapinnan osoitteen alkuosa on *domain.com/api/*. Rajapinta on muotoa RESTful enemmäin kuin puhdas REST, esim. */agencies/*-polussa tiettyä vuokrayritystä ei aina tunnisteta id:n perusteella */agencies/[id]* vaan kirjautuneen vuokrayrityksen id saadaan kirjautumisen yhteydessä saadusta jsonwebtokenista.

Rajapinnan tarkka kuvaus löytyy Javascript dokumentaation muodossa backend-gitin kansiossa */jsdoc/.*

Rajapinnan endpointit löytyvät backendin kansiosta */controllers.*

# Testaus

## Johdanto

Sovelluksen testaukseen on valmistettu erilaisia testejä varmistamaan toiminta erilaisissa tilanteissa ja syötteissä.

Toteutus on jaettu kahteen osioon: Frontend (käyttöliittymänäkymä) sekä backend (tietokanta puoli), joissa molemmista löytyy omat testinsä.

## Testauksen kohde ja tavoitteet

Tavoitteena testauksessa on ennen kaikkea varmistaa odotettu toimivuus ja taata ohjelman sujuva toisto. Toissijaisena testauksen tarkoituksena on ohjata käyttäjää antamaan sopivan tyyppisiä syötteitä.

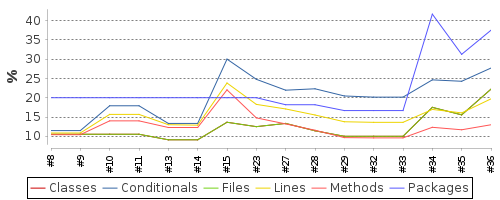
## Testausympäristö

Testaukseen käytetään Jest-testauskehystä. Testit voidaan ajaa paikallisesti käyttäen komentoa ”CI=true npm test” frontendin puolella ja ”npm test” backendin puollella. Testit ajetaan myös Jenkins-palvelimella tietyin väliajoin, jos repositorioon on tullut päivityksiä.

## Testauksen organisointi ja raportointi

Sovellus jakaantuu kahteen osaan: käyttöliittymään (frontend) ja palvelimeen (backend). Molemmille on tehty omat testit ja Jenkins-prosessit.

Tällä hetkellä testien kattavuusanalyysi toimii vain frontendin puolella, jossa tuotetaan Cobertura- ja HTML-muotoiset raportit. Jenkins ajaa frontendin testit automaattisesti sekä tuottaa testikattavuusraportin ja testikattavuusanalyysin käyrän (kuva 6). Testikattavuusraportti voidaan tuottaa myös paikallisessa ympäristössä käyttäen komentoa ”CI=true npm test -- --coverage”.



Kuva 6. Testien kattavuusanalyysin käyrä frontendin osalta.

## Testausstrategia ja integraatiosuunnitelma

Testiskriptit voidaan ajaa Jenkins-palvelimella sekä front- että backendin Master-haarasta.

## Testattavat toiminnot

Frontend: Testeillä varmistetaan, että syötteet ovat korrekteja sekä itse toiminnot (**sisäänkirjautuminen**, rekisteröinti, navigointi). Esimerkiksi sähköposti validoidaan tarkistamalla sekä sähköpostiosoite, @-merkki ja domain.

Backend: Testeillä validoidaan sekä dataa, jotta huonoja tietoja ei päädy itse tietokantaan, sekä kutsuja, mikä mahdollistaa todellisen tiedon palauttamisen.

## Linkit

Projektin Frontend GitHub: <https://github.com/Inno-Kestava-Keikkatyo/innoFrontend.git>

Projektin Backend GitHub: <https://github.com/Inno-Kestava-Keikkatyo/innoBackend>

Jenkins-palvelin: <http://10.114.34.10:8080/>

Jenkins-palvelimen tutoriaali: <https://medium.com/appgambit/ci-cd-pipeline-for-a-nodejs-application-with-jenkins-fa3cc7fad13a>