Vaalikoneverkkosivu — Case viiden ammattikorkeakoulun yhteinen projekti

Alkete Ademaj & Rasmus Karjalainen

2020 Laurea

Laurea-ammattikorkeakoulu

Vaalikoneverkkosivu — Case viiden ammattikorkeakoulun yhteinen projekti

Alkete Ademaj, Rasmus Karjalainen

Tietojenkäsittely

Opinnäytetyö

Joulukuu, 2020

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittely
Tietojenkäsittelyn tradenomitutkinto (AMK)

Alkete Ademaj, Rasmus Karjalainen
Vaalikoneverkkosivu — Case viiden ammattikorkeakoulun yhteinen projekti
Vuosi 2020 Sivumäärä 25

Opinnäytetyön aiheena oli rakentaa lokaalisti toimiva vaalikoneverkkosivu, joka tulee Diakonia-ammattikorkeakoulun, Jyväskylän ammattikorkeakoulun, Laurea-ammattikorkeakoulun, Turun ammattikorkeakoulun, HAAGA-HELIA ammattikorkeakoulun ja Metropolia Ammattikorkeakoulun käyttöön. Vaalikonesivun luominen ja käyttöön ottaminen on kouluille ajankohtaista ja tarpeellista. Vaalikonesivun tarkoituksena on edesauttaa edustajistoon ehdolla olevien näkyvyyttä ja helpottaa äänestäjien valinnantekopäätöstä.

Sivu toteutettiin toiminnallisena opinnäytetyönä ja sen rakentamisessa käytettiin MERN-pinoa. MERN-pinon avulla voidaan rakentaa kolmitasoisen arkkitehtuurin front-end, back-end sekä tietokanta kokonaan JavaScriptin ja JSON:n avulla. Määrittely ja suunnittelu toteutettiin ammattikorkeakoulujen välisenä sisäisenä viestintänä.
Viestinnässä käytettiin Google Meets-sovellusta.

Verkossa olevaa materiaalia hyödynnettiin ohjelmoinnissa. Projektia jatketaan yhdessä toimeksiantajan kanssa opinnäytetyön valmistumisen jälkeen. Lokaalisti toimiva sivu siirretään käyttöönottovaiheessa verkkoon.

Asiasanat: Vaalikone, Edustajat, Kysely, Verkkosivu

Laurea University of Applied Sciences **Abstract**
Degree Program in Business Information Technology
Bachelor's Thesis

Alkete Ademaj, Rasmus Karjalainen
Candidate Selection Engine — A Case Study of the Cooperative Project of Five Universities of Applied Sciences
Year 2020 Pages 25

The aim of this thesis is to build locally working candidate selection engine, that will be used by the universities of applied sciences of Diakonia, Jyväskylä, Laurea, Turku, HAAGA-HELIA and Metropolia. Creating a voting machine and implementing it for universities to use is current and needed. The purpose of the voting machine web page is to give the deputies more visibility and help the voters in their decision making.

The selection engine was done as a project-based thesis and the MERN-stack was used to construct the page. MERN-stack can be used to construct three-tier architecture of front-end, back-end and database entirely with JavaScript and JSON. The knowledge base was formed with the commissioner and the universities participating in the project by having internal communication through Google Meets-application and it also covers the material for programming that is freely available on the internet.

When the thesis is done the project will be continued together with the commissioner, where the locally working website will be implemented to the internet as a functional and a deployable webpage.

Keywords: Candidate Selection Engine, Debuties, Questionair, Webpage

Sisällys

1 Johdanto

Opinnäytetyön tavoitteena on rakentaa sivu, jolla sopivimman ehdokkaan valintasuositus toteutetaan pistetekniikan avulla. Kun opiskelija vastaa vaalikoneen kysymyksiin, pisteytetään vastaukset. Opiskelijan kysymyksistä saamia pisteitä verrataan ehdokkaiden pisteisiin ja lopputuloksena vaalikone ehdottaa opiskelijalle sopivimpia ehdokkaita.

Vaalikoneella halutaan tuoda oppilaitoksen äänestäjät ja ehdokkaat lähemmäs toisiaan antamalla ehdokkaille lisää näkyvyyttä ja helpottamalla äänestäjien äänestysvalintaa. Ehdolla on monia eri näkökulmia omaavia henkilöitä, mutta kyselyn avulla monien eri näkökulmien joukosta valitaan äänestäjän vastauksia tukeva ehdokas. Oppilaan näkökulmasta valinnan tekeminen on helpompaa, kun hän saa varman vastauksen siitä, kenen kanssa näkemykset vastaavat. Vaihtoehtoisesti ehdokkaan on helpompi tuoda itseään esille, kun vaalikone tekee sen myös hänen puolestaan. Vaalikoneen tarjoamien ehdokkaiden lisäksi käyttäjällä on myös mahdollisuus selata tietoa yleisesti kaikkiin ehdokkaisiin liittyen.

Opinnäytetyön toteutetaan toiminnallisena opinnäytetyönä, joka tehdään projektimuotoisena. Toteutustapa sopii projektia varten, sillä se ”voidaan tehdä yksin tai ryhmässä työelämäyhteistyönä joko monialaisena tai vain omaa alaa koskevana työnä.” (Lapin AMK.) Projektissa keskitytään dynaamisen ja lokaalisti toimivan verkkosivun luomiseen, jonka tarkoituksena on suorittaa tehdyt toimintansa ongelmitta. Ohjelmointi on hyvin suuri osa projektia ja suoriutumista varten osaaminen on erittäin tärkeää. Tutkimus rajataan tietyn ohjelmointikielen käyttöön, jonka avulla voidaan tuottaa paras mahdollinen lopputulos. Kyseistä verkkosivua varten käytetään MERN-pinoa, jonka merkitys verkkokehityksen alueella kasvaa koko ajan. MERN-pino muodostuu sanoista MongoDB, Express.js, React.js ja Node.js.

2 Tietoperusta

Opinnäytetyön tietoperusta saadaan erilaisista palvelimista, sovelluksista tai verkkosivuista sekä mahdollisesti muilta tekijöiltä heidän luomansa materiaalinsa perusteella. Ohjelmointikielet ovat universaaleja kaikille tekijöille, joten tiedon löytäminen ei ole ongelmallista. Tässä opinnäytetyössä käytetään MERN-pinoa, josta kerrotaan myöhemmin lisää MERN-pino luvussa.

2.1 HTTP-protokolla

HTTP (Hyper Text Transfor Protocollaa) tarkoittaa internetin tiedonsiirtoprotokollaa, ja se on sovelluskerroksen protokolla. Tämä tarkoittaa sitä, että kyseisessä kerroksessa määritellään jaetut viestintäprotokollat ja käyttöliittymämenetelmät, joita isännät käyttävät viestintäverkossa.

HTTP-protokollan avulla asiakas lähettää palvelimelle pyynnön, johon palvelin vastaa. Palvelin on ”tietoliikenteen yhteydessä tietokoneessa suoritettavaa palvelinohjelmistoa sekä tällaista ohjelmistoa suorittavaa tietokonetta.” (Hosting palvelu.) HTTP-pyyntöjä voidaan tehdä Get-, Post-, Put- ja Delete-metodeilla.

Get-metodilla voidaan hakea resurssia, Post-metodilla lähettää dataa palvelimelle, joka esimerkiksi luo uuden resurssin, Put-metodilla voidaan päivittää resurssi tai luoda kokonaan uusi, jos aiempaa ei ole olemassa ja Delete-metodilla voidaan tuhota resurssi. Metodien avulla voidaan testata sivun toimivuutta, sillä palvelimen tila pysyy samana riippumatta siitä, kuinka monta kutsua tehdään. Esimerkiksi palvelimelle voidaan tehdä kutsu, joka menee täydellisesti eteenpäin, mutta vastausta palauttaessa antaa virheilmoituksen. Näin kutsuja voidaan testata niin kauan, kunnes saadaan palvelimelta haluttu vastaus aiheuttamatta palvelimelle mahdollisia tilamuutoksia.

2.2 JavaScript

”JavaScript on Netscapen kehittämä oliopohjainen ohjelmointikieli, jota modernit selaimet yleisesti tukevat. JS on nimensä mukaisesti ns. scriptikieli, jonka koodi tulkitaan, eli sitä ei erikseen käännetä omaksi ohjelmakseen ennen suorittamista, kuten sovellusohjelmointikielissä.” (Weppipankki.) Sen avulla sivuille voidaan lisätä dynaamista toimintaa.

JavaScript myös toimii back- ja front-end koodina projektissamme, jolloin sen kokonaisuuden rakentuminen helpottaa, kun samaa kieltä käytetään joka paikassa vaalikoneessa. Lisäksi JavaScript on ohjelmointikielenä kevyt, tehokas ja käyttöjärjestelmistä riippumaton.

”Jokaisella verkkosivustolla on kaksi puolta – front-end ja back-end. Niihin viitataan myös termeillä ”selainpuoli” ja ”palvelinpuoli”. Front-end tarkoittaa kaikkea sitä koodia, joka ajetaan verkkoselaimessa – siis silmiesi edessä, kun käytät sivua. Front-endiä ovat esimerkiksi sivun rakenne (html), ulkoasu (css) ja selaimessa tapahtuvat toiminnallisuudet (javascript).” (Dagmar, 2015.)

”Back-end puolestaan tarkoittaa koodia, joka ajetaan sivuston palvelimella – siis esimerkiksi yrityksesi serverihuoneessa tai pilvipalvelussa. Täällä tapahtuvat esimerkiksi sellaiset asiat kuin lomakkeiden käsittelyt, kirjautuminen ja salasanojen tarkistaminen, järjestelmäintegraatiot ja tietokantojen käsittely.” (Dagmar, 2015).

2.3 MERN-pino

”MERN pino on Javascriptpino, jota käytetään täysipainoisten Web-sovellusten helpottamiseen ja nopeuttamiseen. MERN-pino koostuu neljästä tekniikasta: MongoDB, Express, React ja Node.js. Sen tarkoituksena on tehdä kehitysprosessista sujuvampaa ja helpompaa.” (Geeks for geeks, 2020.)

MERN-pinon avulla voidaan rakentaa kolmitasoisen arkkitehtuurin front-end, back-end sekä tietokanta kokonaan JavaScriptin ja JSON:n avulla (Mongodb, 2020). Se tarjoaa hyvän alustan verkkosivua varten, sillä kaikki sen perustekniikat ovat avoimesta lähdekoodista ja sen käyttö on täysin ilmaista.

2.3.1 Mongodb

Tietokannat voidaan nähdä ns. kirjastoina, johon voi tallentaa tietoa tai vaihtoehtoisesti hakea sitä. Suurin osa tietokantaohjelmistoista perustuu relaatiomalleihin, jotka voivat olla relatiivisia tai ei relatiivisia. Niiden erona on se, että relatiiviset tietokannat edustavat ja tallentavat tietoa taulukoihin ja riveihin. Ei relatiiviset tietokannat edustavat tietoja JSON-asiakirjojen kokoelmissa. MongoDB on ei relatiivinen tietokanta.

MongoDB-tietokanta on rakennettua ihmisisiä varten, jotka rakentavat nopeasti kehittyviä ja tyylikkäästi mitoitettavia verkkosivuja ja -sovelluksia (Mongodb, 2020).

MongoDB käyttää JSON-muodossa olevia dokumentteja, ja ne rakennetaan tietyn kaavan mukaisesti. MongoDB:n käyttö sopii projekteihin, jossa tietorakenteiden välillä ei ole suoraa yhteyttä. Tietokanta on yksi suosituimmista ei relatiivisista tietokannoista, sillä se mahdollistaa suurien määrien datan tallentamisen ja osaa käsitellä tiedon liikkuessa nopeasti tuottamatta ongelmia henkilölle tehden siitä erinomaisen projektin käyttöä varten.

2.3.2 Express.js

Express.js on suosituin Node-verkkokehyksistä. Sitä käytetään rajapintana palvelinkoodin rakentamiseen. Verkkokehys on julkaistu ilmaisena ja avoimen lähdekoodin ohjelmistona MIT-lisenssillä. Se on suunniteltu verkkosovellusten ja sovellusliittymien rakentamiseen. Express.js-kehyksen avulla voi käyttää JavaScript-kieltä, sekä tausta- että käyttöliittymässä. Tämän seurauksena kehitysprosessi on paljon nopeampi ja helpompi, koska yksi henkilö voi hallita sekä ulkonäön että tiedon käsittelykerrosta (Apiko, 2017). Express.js sisältää myös ns. väliohjelmistopaketteja, jonka avulla voidaan ratkaista erilaisia kehitysongelmia. Tämä tekee expressistä helppo käyttöisen ja, sillä se antaa vapaammat kädet koodin käsittelemisessä.

2.3.3 React.js

React.js on JavaScript-kirjasto, jonka avulla käyttöliittymiä voidaan rakentaa. Sitä ylläpitää Facebook ja yksittäisten kehittäjien ja yritysten yhteisö. Muiden mallimoottoreiden lailla, React.js mahdollistaa modulaarisen rakenteen verkkosivuihin. React.js on kuitenkin tietoinen vain tietojen toimittamisesta selaimelle, joten React.js-sovellusten luominen vaatii yleensä ylimääräisten kirjastojen käytön valtionhallintaan ja reititykseen.

React.js on hyödyllinen projektia varten, sillä se koostuu useista komponenteista, ja jokaisella komponentilla on oma logiikkansa. Nämä komponentit vastaavat pienen, uudelleenkäytettävän HTML-koodin palautuksesta. Uudelleen käytettävä koodi tekee sovellusten kehittämisestä ja ylläpitämisestä helpompaa (Javatpoint).

2.3.4 Node.js

Avoimen lähdekoodin Node.js mahdollistaa JavaScript-koodin suorittamisen palvelimella. Palvelin on tietokoneen tietoliikennettä suorittava ohjelma.

Node.js antaa kehittäjien käyttää JavaScriptiä komentorivityökalujen kirjoittamiseen ja palvelinpuolen komentosarjojen suorittamiseen - palvelinpuolen komentosarjojen suorittamiseen dynaamisen verkkosivun sisällön tuottamiseksi ennen sivun lähettämistä käyttäjän selaimeen.

Koska Node.js käyttää Googlen V8-koneita, Node.js lisää viitekehyksen vauhtia tehden koodista nopeaa. Sen avulla voidaan rakentaa sivuja, jotka pystyvät käsittelemään rinnakkaisia yhteyksiä korkealla suorituskyvyllä. Lisäksi Node.js:n toimintojen, kuten npm:n ansiosta ohjelmistokehittäjien on helppo uusia tai päivittää kirjoittamaansa koodia ilman suurempaa vaivannäköä.

2.4 JSON

JSON on lyhenne sanoista JavaScript Object Notation. Käytön suosio noussut sitä myötä, kun XML-tiedostoista siirrytään pois. JSON on erittäin helppolukuista ihmissilmälle ja koneiden on helppo parsia sitä. JSON on käytännössä tiedostomuoto, missä tekstiä manipuloidaan.

Kuvio 1: JSON-muodossa oleva dokumentti

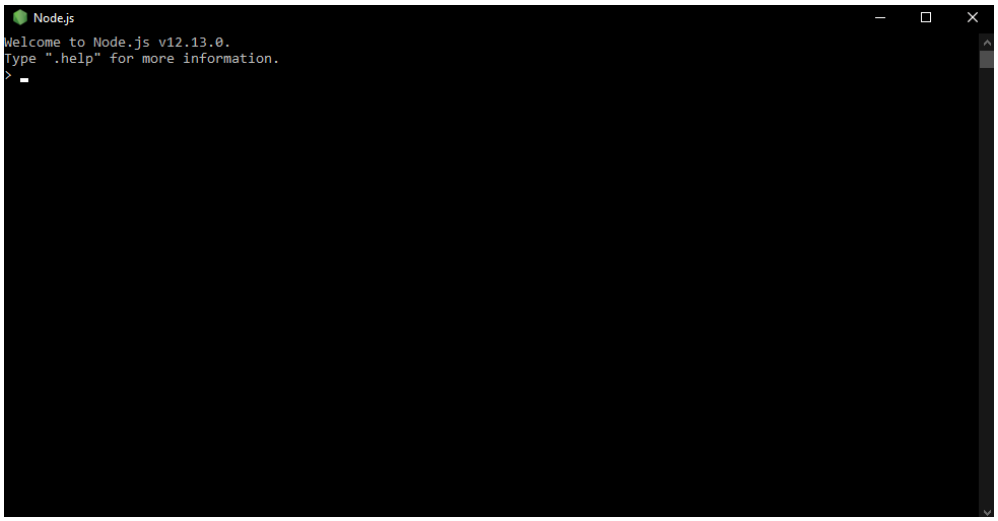
JSON:ia voi esimerkiksi hyödyntää silloin, kun tietoa halutaan lähettää verkon kautta. Tiedostomuodon yleistyessä se on alkanut korvaamaan XML-tiedostojen paikan tiedonsiiroissa helpomman rakenteen ja käsittelyn takia. JSON käsittelee tietoa objekti muodossa, kun taas XML-käsittelee sitä tunnisteiden muodossa.

3 Kohderyhmä ja kohderyhmäympäristö

Verkkosivu kohdistuu ammattikorkeakoulun opiskelijoihin, opiskelijakuntaan, edustajistoon sekä vaaliin ehdolle asettuville. Luonnollisesti opiskelija valitsevat vaalien yhteydessä mieltymystensä mukaisesti edustajiston jäsenet ehdolla olevien joukosta. Edustajisto muodostuu eniten ääniä saaneiden henkilöiden joukosta.

MERN-pino valinta perustui sen tarjoamiin etuihin, jotka tulevat ilmi sivua rakennettaessa. Käyttäjän näkökulmasta sivu on responsiivinen ja helposti toimiva. Tekijän kannalta sivun rakentaminen on vaivatonta ja yksinkertaista. Pinon avulla palvelimen puolella ja sivun, eli front-endin puolella kirjoitettava koodi on samaa. Sivu toimii lokaalisti, mutta sen siirtäminen nettiin vahvistaa edellä mainitun etuutta siinä, että koodi on yhtenäisempää. Yhtenäisen koodin lisäksi tiedon parsiminen palvelimen puolella on helpompaa. Pinon hallitseminen tuo etuuden sivun rakentamisessa ja ongelmatilanteiden ratkaisemisessa. Renderöintiä voidaan tehdä palvelimen puolella, joka nopeuttaa tiedon näkymistä sivulla. Vastaavasti, jos tietoa renderöitäisiin front-endin puolella, sivu latautuisi hitaasti.

Tekemistä varten tarvitaan Node.js-terminaali, Atom tai Visual Studion tekstieditori. Työkalujen avulla koodia rakennetaan ja Node.js terminaali on yksi tapa testata koodin toimivuutta. Komentorivin (terminaalin) suorittaa ja automatisoi tehtäviä tietokoneella ja sen avulla voimme lähettää yksinkertaisia tekstikomentoja tietokoneellemme esimerkiksi selataksemme hakemistoa tai kopioidaksemme tiedostoa. Terminaalin avulla näkee, mikä osa kirjoitetusta koodista toimii tai miten sen avulla varmistetaan tiedon kulku palvelimeen. Visual studio sisältää sisään rakennetun terminaalin, jota voi hyödyntää Node.js terminaalin lisäksi.



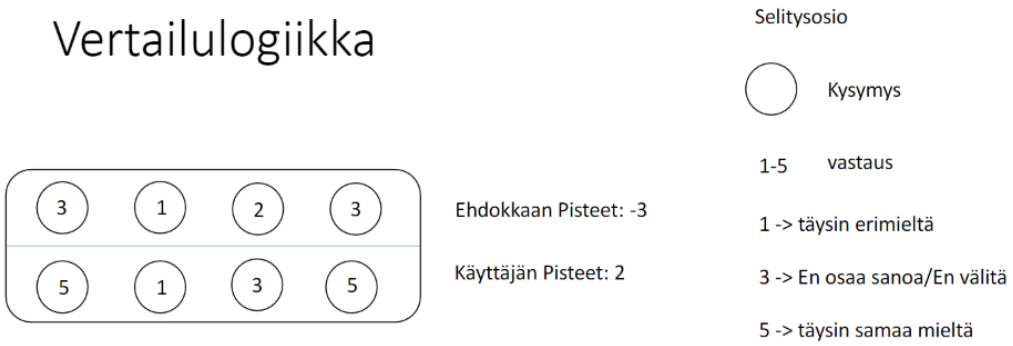
Kuvio 2: Terminaali

Terminaalin avulla sivun tiedostoihin lisätään lisäosia, joita myöhemmin käytetään ohjelmoinnin yhteydessä. Tällainen on esimerkiksi CORS-mekanismi, jonka avulla verkkosivun rajoitettuja resursseja voidaan pyytää toisesta verkkotunnuksesta sen toimialueen ulkopuolella, josta ensimmäinen resurssi on annettu.

4 Työn toteutus

Työn lähtökohtana on rakentaa toimiva sivu. Sen ulkonäkö on toissijaista, johon keskitytään tarkemmin opinnäytetyön valmistuttua. Sivun tekeminen on paikallista, eli toteutus tapahtuu opinnäytetyön tekijöiden oman tietokoneen äärellä. Toteuttamista varten on aluksi luotu teoreettinen tausta, jonka tarkoituksena on selkeyttää kyselyn ja sivun käyttäytymistä.

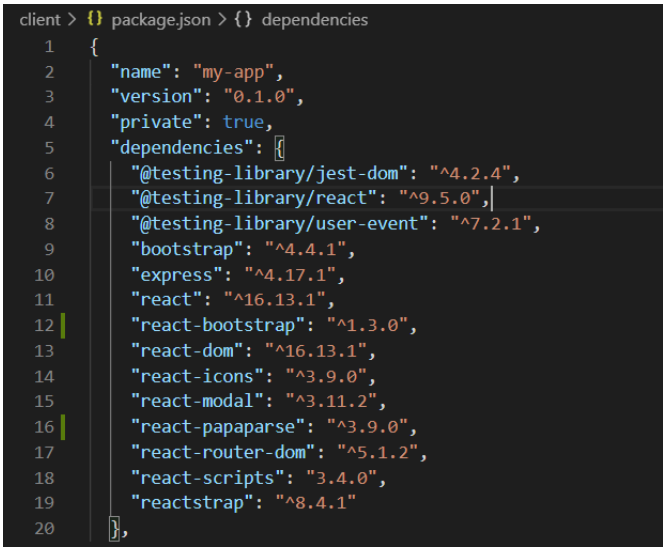
Sivun tärkein ominaisuus on kysely ja se toteutetaan teoriassa vertailulogiikkaa käyttäen. Kyse on pistetekniikasta, jossa ehdokkaan ja käyttäjän pisteitä vertaillaan yhteen. Yksi kysymys sisältää viisi vastausvaihtoehtoa, joista jokainen on pisteytetty välillä miinus kaksi ja sen itseisarvo. Miinus kaksi edustaa täysin eriävää mielipidettä ja kaksi täysin samaa mielipidettä.



Kuvio 3: Vertailulogiikka

Yllä olevassa esimerkissä ensimmäisen kysymyksen kohdalla ehdokas saa kyselystä miinus kolme pistettä ja käyttäjä kaksi. Kun ehdokas ja käyttäjä ovat vastanneet kaikkiin kysymyksiin, tuloksia vertaillaan yhteen. Tämä tapahtuu kolmessa eri osassa. Tulos verrataan kokonaispisteiden, yksittäisten vastausten sekä ääripäävastausten kannalta. Jos pisteet ovat lähellä toisiaan, mutta niiden kokonaisuus muodostuu täysin eri vaihtoehdoista, ehdokasta ei tarjota käyttäjälle.

Vertailulogiikan käsittämisen myötä kirjoitettava koodi voidaan pilkkoa pienempiin osiin. Sivun fyysinen rakentaminen alkaa tekstieditoreita käyttäen, johon koodia kirjotetaan. Terminaalia käyttäen asennetaan paketteja, jotka toimivat kehyksinä ja tarjoavat kirjastoja ohjelmointikielen käyttöä varten. Npm-toimintaa käytetään Javascript-ohjelmointikielessä pakettien järjestäjänä. Terminaaliin kirjoitetaan npm install, jonka jälkeen määritellään asennettava paketti.



Kuvio 4: Npm-toiminnalla asennetut paketit

Package.JSON-kansioon tallentuu kaikki asennetut paketit, jotka näkyvät dependencies-kohdasta. Yllä olevasta kuviosta voidaan nähdä, että Bootstrap on yksi paketti, joka on asennettu. Se tarjoaa kehyksen ja ikoneja, jota voidaan käyttää sivun tyylittelyssä.

Kuvio 5: Pakettien käyttö koodissa

Paketit tuodaan sivun käyttöön kirjoittamalla käytettävään tiedostoon import ja paketin nimi ja mistä se tulee. React.js ei itse pysty tehdä HTTP-kutsuja, vaan se tehdään kolmannen välikäden kautta. Axios on yksi ja tapa tehdä kutsuja. React.js käyttää Axios-kirjastoa asentamalla ja tuomalla se sivun käyttöön kutsumalla sitä.

Erilaisten pakettien lisäksi sivu tarvitsee tietokannan, johon tietoa tallennetaan. MongoDB-tietokannan käyttö muodostaa sivun toisen kokonaisuuden. Ensimmäisenä testataan JSON-tiedostojen lataamiseen tietokantaan. Kandidaatit kirjoitetaan ja viedään kantaan manuaalisesti. Palvelin antaa mahdollisuuden varmistaa, että henkilöt varmasti menevät tietokantaan asti ja pyydettäessä tulevat palvelimeen tai sivulle takaisin esille. Onnistuneen yrityksen jälkeen kandidaattien lisääminen on implementoitu sivuun, jonne on rakennettu lisäys-nappi.

Kandidaatit lisätään sivusta tietokantaan käyttäen CSV-tiedostoja. ”CSV on tiedostoformaatti, jonka mukaisia tiedostoja voidaan hyödyntää esimerkiksi datan tallentamisessa ja datan siirtämisessä toiseen tietokoneohjelmaan. CSV-tiedostoja voidaan käyttää esimerkiksi yksinkertaisena datan arkistointimuotona. Lisäksi niitä voidaan tuottaa yksinkertaisesti tietokannan sisältämästä datasta ja avata taulukkolaskentaohjelmassa.” (Visma). Esimerkki vaalikoneessa käytettävästä CSV-tiedostosta on Microsoft Excel. Tiedostot sisältävät kandidaattien tietoja ja koodi palvelimen puolella parsii tietoa JSON-muotoon, joka menee lopulta sellaisenaan tietokantaan. Ainoastaan sivun ylläpitäjillä on tunnukset, jota kautta he pääsee lisäämään ja muokkaamaan sivulla olevaa tietoa. Normaalit käyttäjät eivät näe lisäys-/poistotoimintoja.

Kuvio 6: Kandidaatit tietokannassa

Tietokannassa kandidaattien kohdalla näkee heidän vastauksensa kyselyyn. Kandidaateille on tässä kohtaa luotu testitunnukset, jota kautta vastataan kyselyyn ja testataan sen toimivuutta äänestäjän kannalta esimerkiksi siten, ehdottaako sivu käyttäjälle samankaltaisen ajatusmaailman omaavat kandidaatit. Kun sivu ehdottaa henkilöitä, kyse on vain numeroarvojen muokkaamisesta. Kyselyyn vastaamisen lisäksi kandidaateilla on mahdollisuus myös kirjallisesti tarkentaa antamansa vastaukset.

Kuvio 7: Esimerkki kandidaatin kommentista

Kommentti on näkyvissä jokaisen kysymyksen alla, ja se näkyy äänestäjälle, kun äänestäjä saa ehdokkaan tuloksiinsa. Ehdokkaan profiilia voi katsoa erikseen, josta näkee myös hänen kommenttinsa kysymyksiin. Äänestäjät eivät kuitenkaan pääse näitä vastauksia muokkaamaan. Sivun ylläpitäjillä on oikeus muokkaamiseen siltä varalta, jos kommentit ovat töykeitä tai epäsoveltuvia.

4.1 Tietoturva

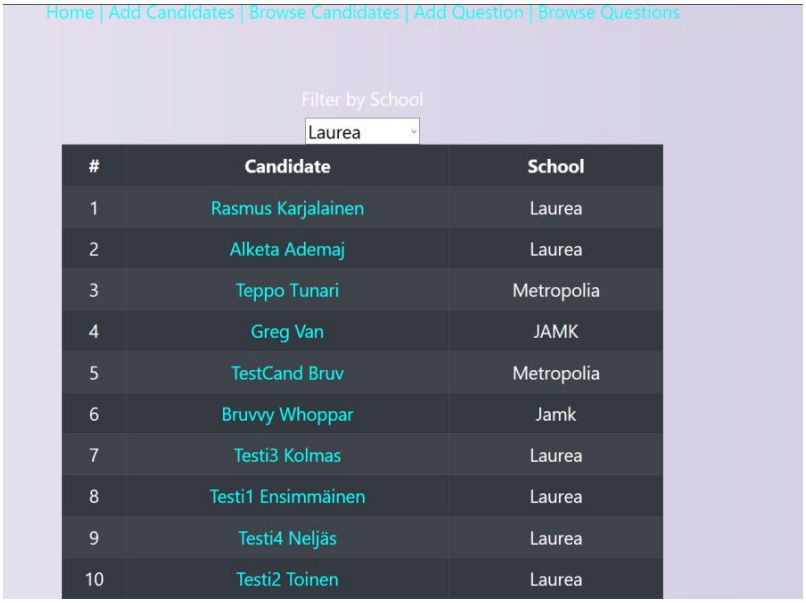
Tietoturvan potentiaalinen riski on otettu huomioon ja implementoidaan sivuun. Se ilmenee salasanojen suojaamisella, koodin rakenteella ja palvelimen käytössä. Käyttäjien ja ylläpitäjien salasanat suojataan Hash-toimintaa käyttäen. Hash muuttaa salasanan 20 numeeriseksi ja kirjaimiseksi sarjaksi, jota on hyvin vaikea rikkoa. ”Salasanojen sijaan tallennetaan niiden tiiviste (hash). Tiiviste, tai hajautusarvo, muodostetaan hajautusfunktiolla. Matemaattisesti ilmaistuna, hajautusfunktio on surjektio suuresta lähdejoukosta pieneen mallijoukkoon. Toisin sanottuna, hajautusalgoritmit ovat yksisuuntaisia, eli tiivisteestä ei pysty laskemaan alkuperäistä lähdettä.” (Sofokus, 2013). Ylläpitäjät itse eivät myöskään näe toisten käyttäjien salasanoja, koska Hash-toiminta generoi ne täysin tunnistamattomiksi.

Kuvio 8: Suojattu salasana Hash-toimintoa käyttäen

Front-end puolella, eli sivulla, koodi kirjoitetaan siten, että käyttäjä ei pääse tietoon käsiksi tai muokkaamaan sitä. Palvelin suorittaa erilaisia toimintoja, joiden vastaukset lähetetään selaimelle käsiteltäväksi. Front-end, suorittaa funktioita, jotka esimerkiksi varmistavat palvelimelta tulleen tiedon oikeaksi. Varsinainen tiedon parsiminen tulee tapahtumaan kuitenkin palvelimella, joka on täysin piilossa käyttäjältä. Siihen ei pääse käsiksi fyysisen sivun kautta vahvistaen ajatusta siitä, että sivussa olevaa tietoa ei voi muokata.

4.2 Kirjautuminen ja autentikaatio

Kirjautuminen on tarkoitettu sivun ylläpitäjille ja kandidaateille. Ylläpitäjien kirjautuessaan sisään he pääsevät muokkaamaan ja lisäämään kandidaatteja järjestelmään. Kandidaateilla on myös omat tunnukset, jota kautta he pääsevät vastaamaan kyselyyn. Kyselyn kautta ehdokkaiden antama tieto tallentuu tietokantaan asti. Käyttäjätunnus on sitä varten, jotta kyselyn kautta saatu tieto voidaan yksilöidä.



Kuvio 9: Luettelo kandidaateista sivulla

Sivu on rakennettu kontekstin sisään, joka pitää sisällään 3 arvoa; kirjautumisstatus, kirjautuneen käyttäjän tason ja sähköpostiosoitteen. Kontekstia käytetään, koska se ylittyy React.js - sovelluksessa kaiken ympärille. Muussa tapauksessa joka ikisessä komponentissa pitäisi olla leivänmuru kirjautumissivulta mukana, jotta tiedetään, onko sivulla aktiivinen kirjautuminen ja mikä sen status on.

```
render() {
  return (
    <UserContext.Provider value = {{...this.state, changeUser: this.changeUser }}>
      {this.props.children}
    </UserContext.Provider>
  );
```

Kuvio 10: Käyttäjäkontekstin tarjoaja

Kontekstia hyödynnetään käyttäjäkontekstin tarjoajan avulla. Sovellus saadaan kierrettyä kontekstin sisään sovelluksen Kirjautumissivulta saatujen tietojen mukaan vertaus; Ensin katsotaan, onko annettua sähköpostiosoitetta olemassa. Löytyneen käyttäjän salasanaa verrataan annettuun, jonka perusteella kirjautuminen onnistuu.

```
    axios.post('http://localhost:5000/login',  user)
      .then(res => {
        this.setState({serverResponse: res.data.email});
        const { changeUser } = this.context;
        changeUser(res.data.status,res.data.email,true);
      });
  }
```

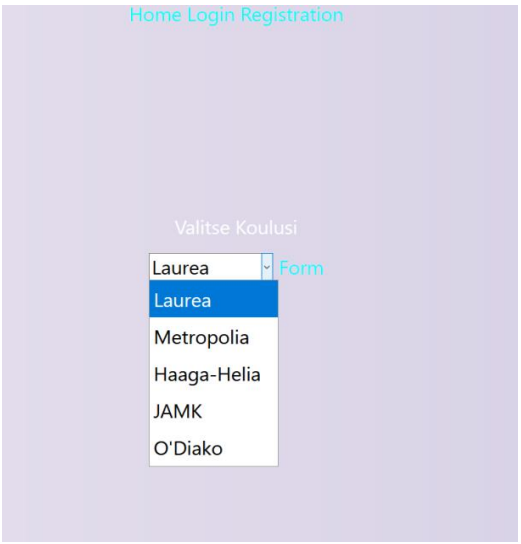
Kuvio 11: Kontekstiarvojen muuttaminen

Onnistuneen kirjautumisen jälkeen kontekstin changeUser-funktiota käytetään kirjautumisen viimeistelyyn. Uloskirjautuminen onnistuu changeUser–funktiolla, joka saa parametrit kontekstin oletusarvoista. Uloskirjautuminen palauttaa kontekstin vierailijamuotoon, joka on sivun oletustila.

4.3 Kysely

Kyselyt ovat kampuskohtaisia ja jokainen oppilaitos voi lisätä haluamansa määrän kysymyksiä kyselyyn. Kandidaatit ja äänestäjät pääsevät vastaamaan ainoastaan oman kampuksensa kyselyyn. Vain järjestelmän ylläpitäjillä on pääsy muiden kampusten tietoihin. Vastaukset pysyvät myös oman kampuksen sisällä ja niitä ei pääse ulkopuoliset muokkaamaan. Tällä varmistamme sen, että vaalikone pysyy luotettavana sivuna ja sivun ylläpito on luotettavissa käsissä.

Kampuskohtaisten kysymysten lisäksi ylläpitäjille annetaan mahdollisuus käyttää universaaleja kysymyksiä. Kysymykset ovat kaikkien kampuksien nähtävissä. Valittaessa kampuksen, listan päädyssä on vaihtoehto, jota valitsemalla luotu kysymys tulee jokaiselle koululle nähtäviin. Toiminta helpottaa kyselyn rakentamista, kun yksi henkilö itse voi luoda universaalin kysymyksen.



Kuvio 12: Kyselyn valinta

Kysymystä luodessa koodissa oleva funktio lähettää palvelimelle pyynnön. Kysymyksestä luodaan objekti, joka saa palvelimelta vastauksen. Objektista tehdään ehto lause, joka käy läpi syötettyjä kysymyksiä. Jos kysymys on jo olemassa, palvelin antaa “kysymys on jo olemassa” vastauksen ylläpitäjälle. Uuden kysymyksen luodessa palvelin antaa onnistuneen vastauksen siitä, että kysymys on lisätty tietokantaan.

```
app.post('/addQuestion', (req,res) => {
  console.log(req.body.question);

  var question = new Question ({
    question: req.body.question,
    area: req.body.area,
  });

  Question.countDocuments({question: req.body.question},function(err, count) {
    if (count == 0) {
        question.save(function(err, user) {
      if (err) return console.log(err);
        console.log("Succesfully added question to database!");
      });
    }
    else {
      console.log("question already exists!");
    }
  });
});
```

Kuvio 13: Kysymyksen lisäävä funktio

Tietokannassa on itse kysymyksen lisäksi määritelty “alue” — muuttuja. Tämän avulla voidaan määritellä kysymyksen laajuus. Universaaleille kysymyksille kuuluu muuttuja-arvo “Undefined” kun taas koulukohtaisille on oma nimi, esim. “Laurea” tai “Metropolia”.

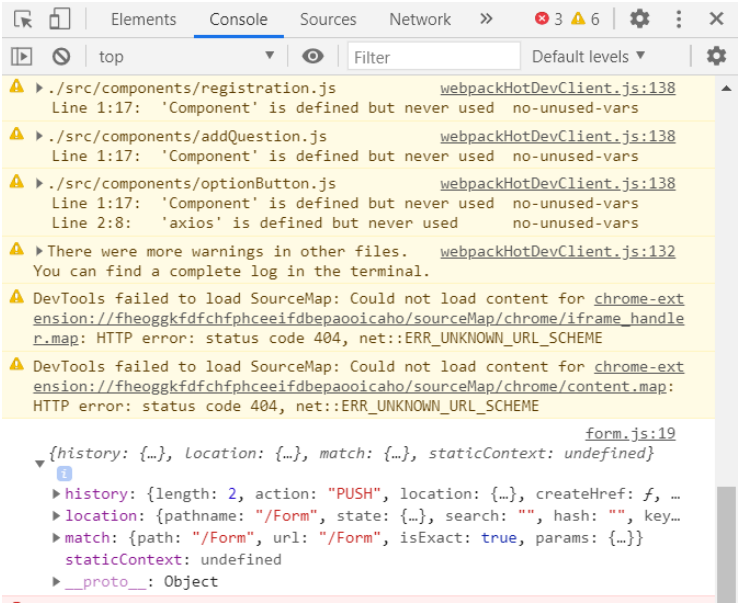
```
    if(index.area == area || index.area == 'Undefined') {
```

Kuvio 14: Kyselyn muodostuslogiikka

Sivulla Kysymykset käydään kuvion 10 valinnan perusteella läpi. Jos halutaan esim. Laurean kysely näkyviin, tuodaan kannasta kaikki kysymykset, joilla alue — arvo on “Laurea” tai “Undefined”.

4.4 Testaus ja tulokset

Sivun testaus on lokaalia ja itsenäistä, ja sen tekeminen on mahdollista monella eri tapaa. Sen tarkoituksena on ennaltaehkäistä sivuun ilmeneviä ongelmia ja korjata niitä tarpeen tullen. Testausta tehdään front-end koodin puolella antamalla funktiolle tietyn toiminnallisuuden, tai fyysisesti etsimällä sivusta ongelmia. Suurimmaksi osaksi se tapahtuu konsolin sisällä. Koodin kirjataan kohtia, jotka näkyvät front-end konsolin puolella. Tätä kautta näkee, mitä koodin sisällä tapahtuu.



Kuvio 15: Konsoli

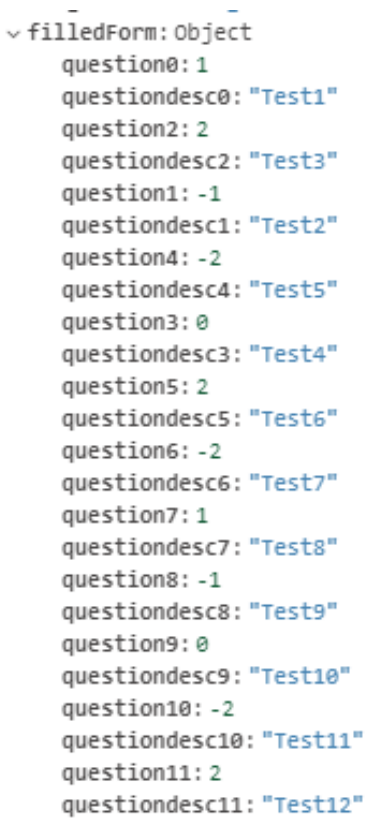
Konsoliin kirjoitettu tieto ei näy sivulla. Se ilmoittaa myös virheistä, jos niitä esiintyy koodissa. Voidaan esimerkiksi kirjoittaa funktio ja tarkistaa, tuoko se halutun vastauksen. Funktioon lisätään console.log teksti, joka tulee esille verkkosivussa sijaitsevaan konsoliin. Konsolilla on tapana ilmoittaa tiedoston rivi, missä virhe ilmenee, joka edesauttaa oikean tiedon korjaamisessa. Yksinomaan jokaiseen kysymykseen vastaaminen on pitkäaikaista, joten konsolin lisäksi testausta tapahtuu funktioiden avulla.

```
var taulukko = [-2,-1,0,1,2];

app.post('/presendtwo', function(req, res) {

 Candidate.findOne({email: req.body.email}, function (err, secondCandidate) {
  var data = secondCandidate.school;
  Question.countDocuments({
   $or: [
    {'area': data},
    {'area': "Undefined"}
   ]
  }, function (err, count) {
      for (var i = 0; i < count; i++) {
    var rndnumero = taulukko[Math.floor(Math.random() * taulukko.length)];
    var nestedOpt = 'filledForm.question'+i;
      Candidate.findOneAndUpdate({email: req.body.email},
        {$set: {[nestedOpt]: rndnumero}}, { useFindAndModify: false }, function(err, doc) {
      });
      console.log(Math.floor(Math.random() * 12)-10);
```

Kuvio 16: Funktio, joka arpoo kandidaattien vastauksia

Palvelin lähettää kutsun, johon se saa vastaukseksi kandidaatin. Kandidaatti etsitään sähköpostin ja kampuksen avulla. Alueeksi on määritelty tietokannasta saatava arvo sekä sellaisen määrittämätön siltä kannalta, jos kampusta ei ole määritelty tietylle kandidaatille.

Funktio suorittaa kierroksen, jossa taulukossa olevia numeroita arvotaan satunnaisesti. Taulukossa olevat arvot kuvailevat pistemääriä, joita tulee kyselyyn vastatessa. Arvonnan jälkeen funktio arpoo kyseiset pisteet kutsun mukana tulleelle kandidaatille ja päivittää sen tietokantaan. Olennainen osa kyselyä on, että siihen vastatessa se ehdottaa kyselyyn vastaajalle oikeanlaiset kandidaatit. Satunnaisen kyselyn esitäyttämisen tarkoituksena on helpottaa ja nopeuttaa kyselyyn vastaamista. Sen avulla voidaan nopeasti luoda suuria määriä testikyselyjä tietokantaan, joiden avulla on helpompi määritellä vaalikoneen logiikan toimivuutta. Suorittamalla funktio se arpoo jokaisella kerralla eri pistemäärät kandidaatille, jonka myötä kyselyyn ei itse tarvitse vastata vielä testausvaiheessa. Funktio suoritetaan Insomnia-ohjelmaa käyttämällä, joka lähettää kutsun tietokannalle ja tuo vastauksen takaisin.



Kuvio 17: Esimerkki täytetystä kyselystä tietokannassa

5 Yhteenveto ja jatkokehitys

Opinnäytetyön tavoitteena on luoda lokaalisti toimiva vaalikoneverkkosivu. Sen tarkoituksena on tehdä äänestämisestä helpompaa ja tuoda kandidaateille näkyvyyttä. Suurin osa sivun päätoiminnoista on paikoillaan, mutta virheiden käsittely ja tiedon oikea parsiminen vaativat jatkokehitystä. Tehty vaalikoneverkkosivu on yksinkertainen ja toimiva, mutta vaatii vielä lisäkehitystä ja optimointia, jotta se saadaan koulujen käyttöön.

Jatkokysymykseksi nousee netissä toimiva verkkosivu. Sivun siirtäminen nettiin on projekti itsessään ja tuo mukanaan uusia asioita, joita pitää huomioida siirron aikana. Sivun on tarkoitus tulevaisuudessa tulla toisten käyttäjien käyttöön, joten virheiden käsittelyä jatketaan ja varmistetaan, jotta sivu on samalla myös toimiva. Tulevaisuudessa sivun tarkoituksena on sisältää sekä eri oppilaitosten oppilaiden että edustajien tietoja, jolloin tietoturvallisuudella on suurempi ja tärkeämpi rooli, joka painottuu virheidenkäsittelyn jatkuvuudessa. Tyylittely painottuu myös jatkokehityksessä, sillä sivun toimivuus on ensisijaista, jolloin vasta sivun tullessa muiden käyttöön varmistetaan sen visuaalisesti parempi ulkonäkö.

Tällä hetkellä tiedon parsiminen tapahtuu front-endin puolella testauksen vuoksi. Sivun siirtyessä verkkoon parsimista siirretään back-endin puolelle, joka luo tietoturvariskin. Tietoturvan rooli nousee jatkossa suuremmaksi. Varmistus siitä, että ulkopuoliset eivät pääse henkilökohtaisiin tietoihin käsiksi tehdään jatkuvan testauksen kautta. Testaajina käytetään tilaajien kanssa sovittua testausryhmää. Tiiviin ja turvallisen ryhmän avulla varmistamme, että ei myöskään myöhemmissä testausvaiheissa tietoa leviä ulkopuolisten käsiin.

Lähteet

Sähköiset

Apiko. 2017. Express.js Mobile App Development: Pros and Cons for Developers. Viimeksi muokattu 23.7.2020.

https://apiko.com/blog/express-mobile-app-development/

Dagmar. 2015. Mitä markkinoijan tulee ymmärtää web-ohjelmoinnista. Viitattu

23.7.2020

https://www.dagmar.fi/verkkopalvelukehitys/mita-markkinoijan-tulee-ymmartaaweb-ohjelmoinnista/

Geeks for geeks. 2020. Mern stack. Viitattu 15.8.2020

https://www.geeksforgeeks.org/mern-stack/

Hosting palvelu. Mikä palvelu on? Viitattu 15.8.2020

https://www.hostingpalvelu.fi/ohjeet/yleiset-ohjeet/mika-on-palvelin/

HTTP. TIE-23500 Web-ohjelmointi. Luettu 15.7.2020

http://www.cs.tut.fi/~seitti/2015/kalvot/http/all.html

Javatpoint. Pros and Cons of ReactJS. Viitattu 5.8.2020

https://www.javatpoint.com/pros-and-cons-of-react

Lapin AMK. Opinnäytetyön toteuttaminen. Viitattu 15.8.2020

https://www.lapinamk.fi/fi/Opiskelijalle/Opinto-opas,-AMK-tutkinto/Opinnaytetyoohje/Opinnaytetyon-toteuttaminen

Laureamko. 2019. Edustajisto pähkinänkuoressa. Viitattu 20.7.2020

https://laureamko.fi/opiskelijakunta/edustajiston-vaalit/

Mongodb. 2020. What is the mern stack? Viitattu 18.7.2020

https://www.mongodb.com/mern-stack

Mongodb. 2020. Why and when to use Mongodb. Viitattu 18.7.2020

https://www.mongodb.com/why-use-mongodb

Sofokus. 2013. Salasanat talteen turvallisesti. Viitattu 2.12.2020

https://www.sofokus.com/fi/blogi/2013/02/07/salasanat-talteen-turvallisesti/

Visma. CSV-tiedosto - Mikä on CSV-tiedosto?. Viitattu 20.11.2020

https://www.visma.fi/epasseli/kirjanpidon-sanakirja/c/csv-tiedosto/

Webbipankki. JavaScript-kielen alkeet - osa 1. Mikä on Javascript? Viitattu 5.8.2020

https://weppipakki.com/js/opas/alkeet1.htm#mojs

Kuviot