LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

ÄÄNESTYSSOVELLUS PÄÄTÖSTENTEKOON

LAHDEN AMMATTIKORKEAKOULU
Insinööri (AMK)
Tieto- ja viestintätekniikka
Ohjelmistotekniikka
Kevät 2019
Nino Manninen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Manninen, Nino	Opinnäytetyö, AMK	Kevät 2019
	Sivumäärä	
	36	

Työn nimi

Äänestyssovellus päätöstentekoon

Tutkinto

Tieto- ja viestintätekniikan koulutus, ohjelmistotekniikka, Insinööri (AMK)

Tiivistelmä

Opinnäytetyössä tutkittiin single-page application -sovelluksen rakennetta sekä siihen olennaisesti liittyviä muita teknologioita ja luotiin sovellus äänestämiseen.

Työ tehtiin kiinalaiselle mobiilipeliyritys Avalonille, joka on perustettu Kiinan Chengdussa. Sovelluksen avulla haluttiin helpottaa yrityksen sisäistä toimintaa ja tehdä yritystä yhtenäisemmäksi. Avalonin suosituimmat pelit kuuluvat The Greedy Cave -pelisarjaan.

Sovellus rakennettiin SPA-arkkitehtuurin avulla. SPA-sovellukset ovat nykyaikaisen ohjelmistokehityksen suuri trendi, niiden nopeuden ja siten sulavamman käytettävyyden vuoksi.

Selainpohjainen SPA-sovellus toteutettiin Vue.js-ohjelmointikehystä käyttämällä ja WWW-palvelimena käytettiin NGINX-nimistä ohjelmistoa. Käytetyt teknologiat ovat ajankohtaisia, ja niiden suosio on selvässä kasvussa.

Sovelluksen avulla voidaan luoda kyselyitä, äänestää ja tarkastella äänien jakautumista. Äänestysten pohjalta on mahdollista tehdä päätöksiä yrityksen sisäiseen toimintaan.

Asiasanat

Single-Page Application, Vue.js, JavaScript, Webpack, NGINX

Abstract

Author(s)	Type of publication	Published
Manninen, Nino	Bachelor's thesis	Spring 2019
	Number of pages	
	36	

Title of publication

Polling web application for decision-making

Name of Degree

Information and Communication Technology, Software Engineering

Abstract

The goal of this thesis was to study the structure of a single-page application and other technologies that are essentially related to it. It was implemented by creating a polling web application.

The application was created for mobile game company Avalon based in Chengdu, China. The purpose of the application was to make internal operations easier for the company and make the company more unified. Avalon's top games are part of The Greedy Cave game series.

The application was built with the SPA-architecture. Single-page applications are a big trend in modern web development, due to their performance and, therefore, their user-friendliness.

Web-based SPA was implemented using the JavaScript framework called Vue.js and NGINX was used as the web server software. These technologies are topical, and their popularity has been constantly increasing.

The application can be used to create polls, vote and view the results of the vote. Based on the vote, it is possible to make decisions on the internal activities of the company.

Keywords

Single-Page Application, Vue.js, JavaScript, Webpack, NGINX

SISÄLLYS

1 JOHDANTO

Äänestyssovellukset ovat tulleet viime vuosina suosituiksi hyvin erilaisissa käyttötarkoituksissa. Äänestyssovellusta voidaan käyttää leikkimielisesti kavereiden kesken tai keskustelufoorumeille jaettuna kuin myös merkityksellisemmässäkin päätöksen teossa tai mielipidemittauksissa. Sovelluksesta saatua dataa voidaan myös hyödyntää osana tutkimuksia. Tunnetuimpana tällaisista sovelluksista on mahdollisesti StrawPoll.Me sen yksinkertaisuuden ja vaivattomuuden vuoksi. Toisinaan kuitenkin tarvitaan monipuolisempaa ja räätälöidympää sovellusta erilaisiin käyttötarkoituksiin, jolloin sellainen täytyy luoda vastaamaan käyttötarkoituksia.

Tässä opinnäytetyössä alettiin tehdä yksilöityä äänestyssovellusta kiinalaiselle tarkemmin Chengdussa 2013 perustetulle ja toimivalle mobiilipeliyritys Avalonille, joka työllistää tällä hetkellä noin 50 työntekijää ja jonka uusin loppuvuodesta 2018 julkaistu luomus on The Greedy Cave 2: Time Gate. The Greedy Cave -pelisarjan ensimmäistä osaa on ladattu Google Play -kaupasta yli miljoona kertaa (Google Play 2019).

Työn tarkoituksena on luoda Avalonille räätälöity selainpohjainen äänestyssovellus. Sovelluksen avulla he voivat tehdä yrityksen sisäisiä kyselyitä omiin tarpeisiinsa esimerkiksi kokoustilanteessa. Sovelluksessa kyselyihin on mahdollista myös lisätä kuva sekä kyselyn voi avata omalle mobiililaitteelleen lukemalla sovelluksen generoiman yksilöidyn QR-koodin. Koodin lukemalla voi antaa äänensä luodulle kyselylle ja lopulta tuloksia voidaan tarkastella omalta tai yhteiseltä näytöltä ja tehdä päätöksiä niiden pohjalta.

Teknologisena tavoitteena työssä on myös perehtyä tuoreimpiin web-teknologioihin. Yksi näistä on Vue.js-ohjelmointikehys, jolla voidaan luoda yksisivuinen responsiivinen selainsovellus. Työssä tutustutaan muun muassa myös WWW-palvelimen konfigurointiin sekä selvitetään, mitä muita teknologioita tarvitaan kokonaisvaltaisen selainsovelluksen rakentamiseksi.

2 NYKYAIKAISET WEB-SOVELLUKSET

2.1 SPA

Yksisivuinen selainsovellus (SPA) on nettisivu, joka uudelleen renderöi sen sisältöä käyttäjän tekemien toimintojen mukaan. Sivulla käytettävät toiminnot voivat olla navigointilinkkejä tai painikkeita, ja niitä klikattaessa kyselyitä ei tarvitse erikseen tehdä palvelimelle uuden HTML-dokumentin lataamiseksi. Tällä tavoin vältetään käyttäjäkokemuksen keskeytyminen siirryttäessä seuraavaan näkymään, jolloin web-sovelluksen toiminnasta tulee enemmän työpöytäsovelluksen kaltainen. (blog.pshrmn 2018.)

Jokaisella www-sivulla on osoite ja useimmiten osoite on verkkotunnus eli domain, mutta se voi olla myös IP-osoite. Verkkoselaimessa osoiterivillä olevaa osoitetta kutsutaan URL:ksi. URL koostuu verkkotunnuksesta sekä sen perään tulevista määritteistä, jotka ovat erotettuina verkkotunnuksesta vinoviivalla (/). Vinoviivan jälkeisillä määritteillä kerrotaan tiedon tarkempi sijainti, joka usein kuvaa tiedostohakemiston rakennetta. Määritteitä URL:ssä erotellaan yhtäsuuruus- (=) sekä et-merkein (&). Näiden lisäksi voidaan käyttää hash eli ristikkomerkkiä (#), mutta sillä kuvattu tieto etsitään paikallisesti ilman tietojen lähettämistä, jolloin sivuakaan ei päivitetä. Perinteisesti polun vaihtuessa sivu kuitenkin ladataan uudestaan.

SPA-sovellukset myös useimmiten päivittävät URL:n selaimen osoiterivillä jäljitelläkseen perinteistä sivulla navigointia, kuitenkin siten, että täydellistä sivulatausta ei tarvitse tehdä uudelleen (Google Analytics 2018). Aina yksisivuista selainsovellusta ei ole rakennettu näin, vaan toisinaan selainsovellus seuraa sisäisesti sovelluksen tilaa, jolloin URL ei päivity. Tämänkaltainen lähestymistapa on kuitenkin haasteellinen silloin, kun esimerkiksi halutaan jakaa sisältöä jollekin toiselle. Tällöin henkilö, jolle sisältö on jaettu, joutuu aloittamaan sovelluksen käytön täysin alusta ja hänelle on neuvottava, miten päästä haluttuun näkymään. Tästä johtuen tällainen SPA-sovellus on käytöltään rajoittunut. Tämän välttääkseen SPA-sovellus on mielekkäämpää rakentaa siten, että URL päivittyy osoiterivillä ja täten voidaan sisältöä jakaessa olla varmoja siitä, että näkymä on myös vastaanottajalla sama kuin jakajalla, kunhan molemmilla on yhtäläiset oikeudet sisältöön. (blog.pshrmn 2018.)

Siinä missä yksisivuisten selainsovellusten toteutukset voivat vaihdella, useimmat niistä kuitenkin nojaavat samanlaiseen selainkäyttäytymiseen, jossa ohjelmointirajapinnoilla aktivoidaan sovelluksen keskeisimmät toiminnallisuudet (blog.pshrmn 2018). Yleisimpiä JavaScript-ohjelmointikehyksiä, joilla SPA-sovelluksia kehitetään, ovat muun muassa Googlen ylläpitämä Angular, Facebookin ylläpitämä React sekä Ember.js ja Vue.js.

2.2 Responsiivisuus

Responsiivinen web-suunnittelu on tullut kuluvalla vuosikymmenellä olennaiseksi osaksi nettisivujen ja selainsovellusten rakentamista. Oleellisimpana tekijänä sille on kosketusnäytöllisten älypuhelimien yleistyminen räjähdysmäisesti. Puhelinten lisäksi myös muiden kannettavien laitteiden suosio on kasvanut. Tämä on johtanut siihen, että markkinoilla käytössä olevia näyttökokoja on lukematon määrä. Näyttökokojen suuren ja jatkuvasti kasvavan määrän takia jokaiselle näytölle erikseen optimoitua näkymää ei ole mielekästä lähteä toteuttamaan.

Responsiivisen eli mukautuvan suunnittelun ja toteutuksen avulla voidaan kuitenkin palvella käyttäjää laitteesta ja näyttökoosta riippumatta. Responsiivisesti suunnitellut sovellukset ja sivut mukautuvat käytettävän näyttökoon mukaan. Sovelluksen yksittäinen näkymä voi vaihdella, mutta sama sisältö pyritään kuitenkin sisällyttämään sovellukseen näyttökoosta riippumatta.

```
<meta name="viewport" content="width=device-width,initial-scale=1.0">
```

KUVA 1. Viewport

Helpoin keino toteutukseen on lisätä sovelluksen HTML-koodiin määritys *viewport meta*-elementtiin (KUVA 1). Määrityksellä annetaan selaimelle ohjeita, miten sen tulisi skaalata sivua. *Viewport* itsessään kertoo alueen, jonka käyttäjän on mahdollista sivusta nähdä. Tämä lähestymistapa ei kuitenkaan ole kovin kokonaisvaltainen hyvän käyttäjäkokemuksen kannalta. Parhaimmillaan se toimiikin vanhojen ei-responsiivisten sovellusten tekohengittäjänä ennen kokonaan uuden sovelluksen luomista. Ongelmalliseksi tämä muodostuu käytettäessä sitä yhdessä *media query* CSS -tekniikan kanssa, jolloin *viewport meta* rajoittaa sen toimintaa kapeilla näytöillä. (MDN Web Docs 2019a.)

Media query -tekniikalla sovelluksen tyylejä voidaan mukauttaa eri kokoisille näytöille, kuten asettamalla raja ruudun leveydelle (KUVA 2). Rajaa pienemmillä ruuduilla tekstin koko voidaan esimerkiksi asettaa pienemmäksi kuin rajaa suuremmilla ruuduilla.

```
@media only screen and (max-width: 600px)
```

KUVA 2. CSS:n media query tekniikka

*Media query*n lisäksi CSS-tyyleissä voidaan käyttää suhteellisia yksiköitä responsiivisuuden saavuttamiseksi. Yleisimpänä näistä käytetään prosentteja (%) sekä *em*-elementtiä. Pikseli (px) arvojen käyttöä on pääsääntöisesti syytä välttää responsiivisen sovelluksen rakentamisessa.

Grid-view eli ruudukkonäkymä on ikään kuin responsiivisen sivun pohjapiirustus. Ruudukon kokonaisleveys on 100 %, ja se on tyypillisesti jaettu kahteentoista sarakkeeseen, ja täten yksittäisen sarakkeen leveydeksi jää 8,33 % sovelluksen kokonaisleveydestä. Tällöin sarakkeiden absoluuttinen koko vaihtelee ruudun leveyden mukaan, mutta suhteellinen koko pysyy samana. (W3Schools 2019b.)

Responsiivisen suunnittelun tueksi on olemassa myös CSS-sovelluskehyksiä. Sovelluskehykset tarjoavat responsiivisuuden tueksi muun muassa ruudukko-pohjan ja valmiita tyylimäärittelyjä. Sovelluskehys myös nollaa oletustyylit epäjohdonmukaisuuksien välttämiseksi.

3 SPA-SOVELLUKSEN KEHITTÄMINEN VUE.JS:LLÄ

3.1 Vue.js

Vue on 2014 julkaistu Evan You:n kehittämä avoimen lähdekoodin JavaScript-ohjelmointikehys käyttöliittymien sekä yksisivuisten selainsovellusten kehittämiseen (Vue.js 2019). Vue on yhteisöpohjaisesti kehitettävä tarkoittaen sitä, että Vue:n käyttäjät voivat yhteisöllisesti vaikuttaa sen kehitykseen. Vuen vahvuuksia ovat sen pieni koko sekä suorituskyky. (Copes 2018c.)

3.1.1 Asennus ja projektin luonti (Vue CLI)

Vue:sta on mahdollista asentaa kaksi eri versiota, joko komentorivi versio Vue CLI tai tavallinen versio ilman CLI:tä. CLI helpottaa projektin luomisessa sekä konfiguroinnissa. CLI muun muassa konfiguroi Webpackin. CLI myös mahdollistaa graafisen käyttöliittymän käytön sovelluksen kehittämisessä.

```
npm install -g @vue/cli
```

KUVA 3. Vue CLI:n asennus

Vuen asentamista varten tulee Node.js ja sen mukana tuleva paketinhallinta ohjelma npm olla myös asennettuna. Vuen asennus on yksinkertaista, ja se tapahtuu komentoriviltä ja asennetaan globaalisti järjestelmään (KUVA 3).

```
vue create TestApp
```

KUVA 4. Projektin luonti

Asennuksen jälkeen projekti luodaan *vue create* -komennolla haluttuun hakemistoon. Komennon perään laitetaan haluttu nimi projektille (KUVA 4). Seuraavaksi komentokehote pyytää valitsemaan halutut ominaisuudet projektille, joko oletusasetukset tai manuaalisesti käyttäjän valitsemana (KUVA 5). Oletusasetukset riittävät testaamiseen, mutta tuotantokelpoisten sovellusten rakentamiseen manuaalisesti valitut ominaisuudet tulevat todennäköisesti tarpeellisiksi.

```
Vue CLI v3.5.5
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection)
>(*) Babel
 ( ) TypeScript
 ( ) Progressive Web App (PWA) Support
 ( ) Router
 ( ) Vuex
 ( ) CSS Pre-processors
 (*) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing
```

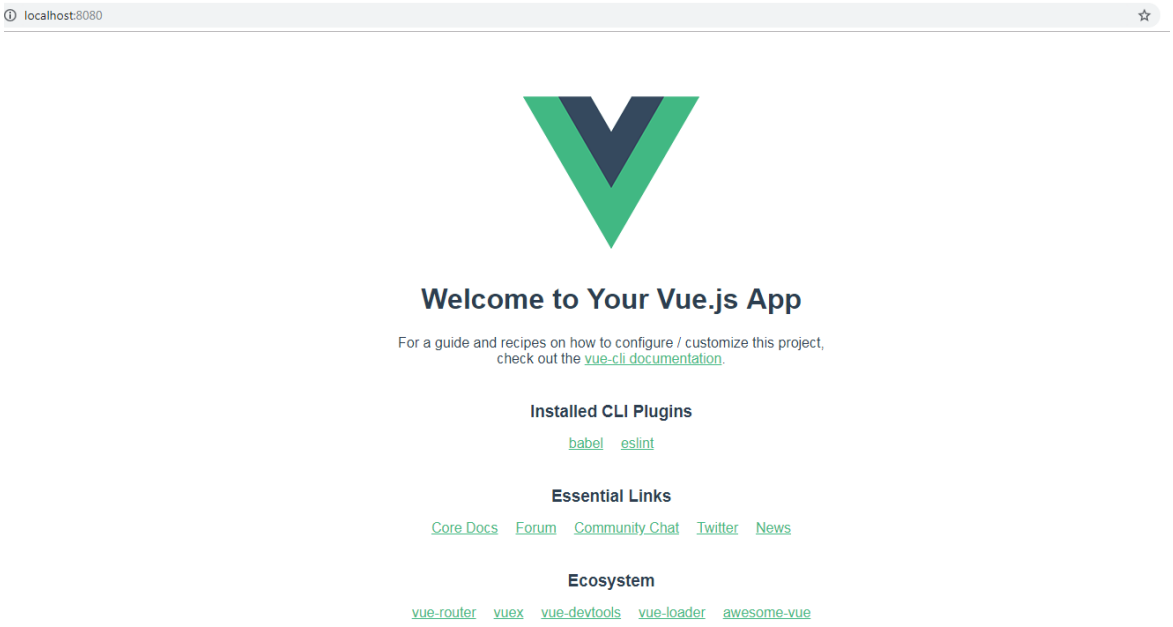
KUVA 5. Ominaisuuksien manuaalinen valintanäkymä, jossa oletusasetukset valittuna

Liitännäisten asentaminen projektin luonnin jälkeen tapahtuu Vue CLI:ssä komennolla *vue add* (KUVA 6). Komento *vue add* on kuitenkin tarkoitettu vain Vue CLI -liitännäisten asentamiseen, eikä sitä ole tarkoitettu npm:n korvaajaksi pakettien asentamiseen (Vue CLI 2019).

```
vue add router
```

KUVA 6. Router liitännäisen lisääminen projektiin

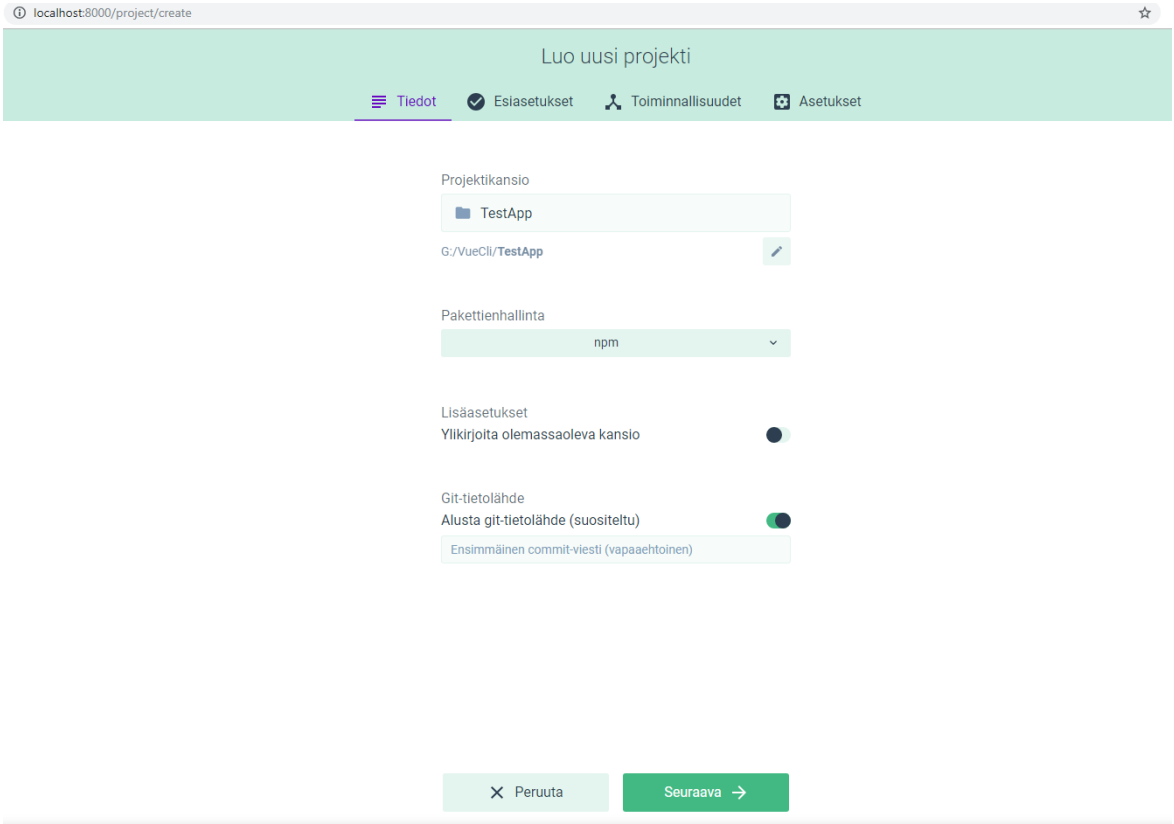
Kun projekti on luotu ja alustettu, se voidaan saattaa selaimeen nähtäville komennolla *npm run serve*, joka suoritetaan projektikansiossa. Komento käynnistää paikallisen kehityspalvelimen ja ajaa luotua sovellusta oletuksena osoitteessa *localhost:8080* (KUVA 7).



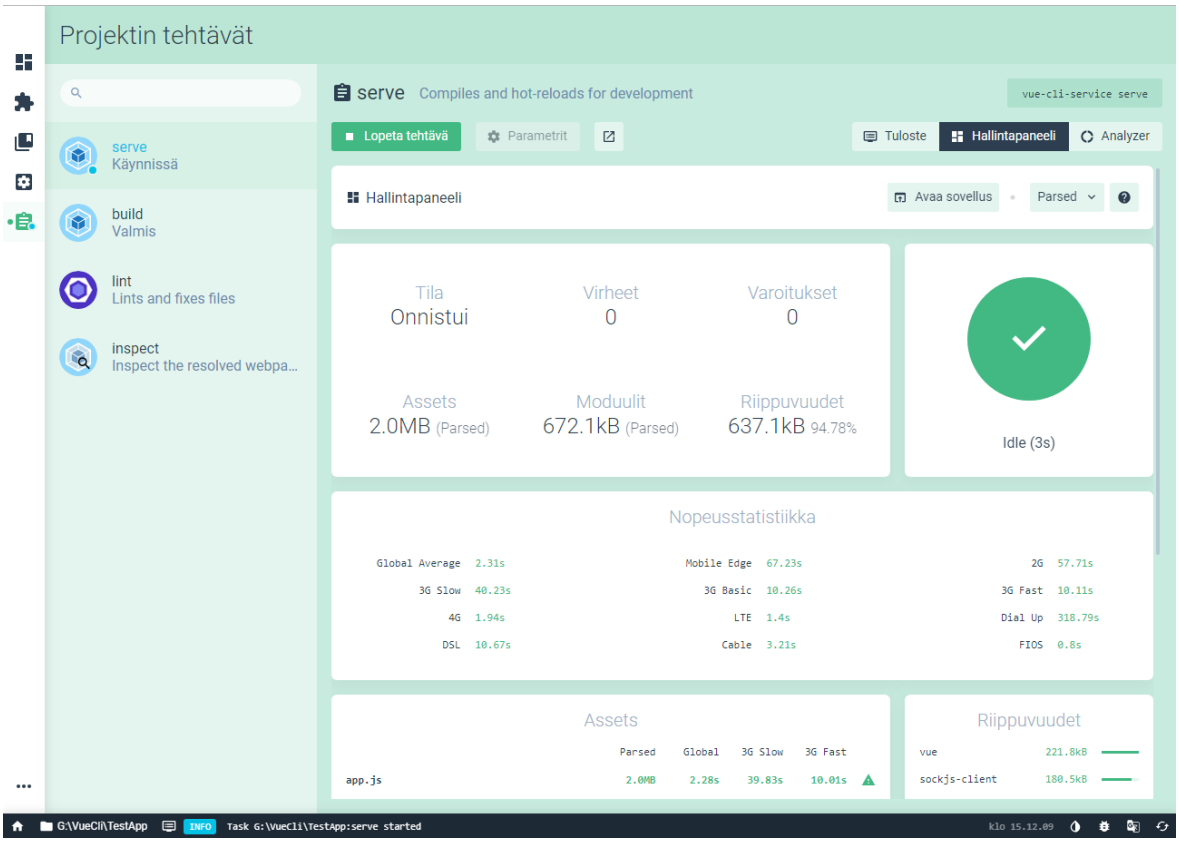
KUVA 7. Vue.js-sovellus

3.1.2 Graafinen käyttöliittymä (GUI)

Projekti on mahdollista myös luoda graafisen käyttöliittymän avulla. Kirjoittamalla *vue ui* komentoriville avautuu graafinen käyttöliittymä selaimeen. Graafinen käyttöliittymä antaa samat mahdollisuudet projektin alustamiseen, konfigurointiin sekä lisäosien ja riippuvuuksien asentamiseen kuin komentokehotekin (KUVA 8). Graafinen käyttöliittymä kuitenkin mahdollistaa projektin analysointia helposti projektinhallinnan statistiikan avulla (KUVA 9). Projekteja on mahdollista myös tuoda graafiseen käyttöliittymään, vaikkei niitä alun perin olisikaan sillä luotu.



KUVA 8. Projektin luonti graafisen käyttöliittymän avulla



KUVA 9. Projektin hallinta ja statistiikka

3.2 Vue.js ja Webpack

Toisinaan on hyödyllistä myös ymmärtää, mitä työkalujen, kuten Vue CLI:n, alla tapahtuu. Vue CLI hoitaa muun muassa Webpackin konfiguroinnin, mutta konfigurointi voidaan hoitaa myös manuaalisesti. Ilman Vue CLI:tä luotu Vue.js projekti käy esimerkkinä tilanteesta, jossa Webpack on konfiguroitava manuaalisesti. Tällöin Webpack on asennettava ja konfiguroitava erikseen.

3.2.1 Webpack

Webpack on JavaScript moduulien niputtaja. Ensisijaisesti sitä käytetään selainpuolella niputtamaan suuri määrä tiedostoja yhdeksi tai muutamaksi tiedostoksi, mutta sitä voidaan myös käyttää palvelinpuolen kehityksessäkin (Copes 2018b). Sitä käytetään muun muassa Vuen tiedostojen niputtamisessa ja sen voi asentaa npm:n avulla. Oletuksena Webpack ei vaadi konfigurointi tiedoston käyttöä, mutta siitä on hyötyä toiminnallisuuksien laajentamisessa (Webpack 2019).

3.2.2 Webpackin asennus

Ennen Webpackin asennusta on myös syytä varmistaa, että Node.js:stä on mahdollisimman uusi versio asennettuna, jotta vältytään epämiellyttäviltä yhteensopivuusongelmilta. Projektille täytyy myös luoda hakemisto, jonne se voidaan asentaa. Hakemistoon on myös luotava *package.json* tiedosto, jonka tehtävänä on hallinnoida projektin riippuvuuksia. Se myös sisältää tekijän tietoja ja projektin versionumeron. Package.json tiedoston saa luotua komentoriville syöttämällä komennon *npm init* (KUVA 10).

```
npm init
```

KUVA 10. Luodaan package.json tiedosto

Alustamisen jälkeen voidaan asentaa Webpack. Webpackin asentaminen tapahtuu yhtä lailla npm paketinhallinta työkalun avulla kuin package.json tiedostonkin luominen (KUVA 11). Webpack voitaisiin myös asentaa globaalisti tietokoneelle, mutta yhteensopivuus ongelmien välttämiseksi on perusteltua asentaa se vain projektikohtaisesti, jolloin sovelluksen kehittäjän on mahdollista hallita, mitä versiota käytetään. Jos Webpackista on käytössä versio 4 tai uudempi, tarvitsee Webpack-CLI myös asentaa. Webpack-CLI on komentorivillä käytettävä lisätyökalu Webpackin konfigurointiin ja sen päivittämiseen.

```
npm install webpack --save-dev
npm install webpack-cli --save-dev
```

KUVA 11. Webpackin asennus

3.2.3 Webpackillä niputus

Niputtamista varten hakemistoon tarvitsee luoda lisää tiedostoja, jotta niitä voidaan niputtaa. Ennen kuin tiedostoja voidaan luoda lisää, tarvitsee tiedostoille luoda kaksi kansiota: ensimmäinen kansio lähdekoodi tiedostoille, johon varsinainen koodi kirjoitetaan, ja toinen kansio jakelukoodille. Jakelukoodi on optimoitua ja minifioitua, ja se tullaan lopulta lataamaan selaimeen. Jakelukoodi kansion sisään luodaan *index.html* -tiedosto ja lähdekoodi kansioon *index.js* -niminen JavaScript-tiedosto. Sovellushakemiston sisällä rakenne näyttää siis tältä:

- *package.json*
- */dist*
    - *index.html*
- */src*
    - *index.js*

Niputuksen toimivuus voidaan todeta kirjoittamalla tiedostoihin koodia. Tiedostossa *index.html* osoitetaan skripti elementissä *src*-attribuutilla niputettujen lohkojen aloituspisteen tiedostonimi, tässä tapauksessa *main.js* (KUVA 12). Vastaavasti tiedostoon *index.js* luodaan esimerkkinä konsoli näkymään tulostuva ”Hello World!” -sovellus (KUVA 13).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Testi sovellus</title>
  </head>
  <body>
     <script src="main.js"></script>
  </body>
</html>
```

KUVA 12. index.html

Niputtaminen tapahtuu komentokehotteen kautta. Webpackin suorittamiseen tarvitsee löytää oikea tiedostopolku ja se löytyy ”*npm bin*”-komennolla sovelluksen hakemistosta. Kun oikea polku on löytynyt, suoritetaan Webpackilla niputtaminen käyttämällä komentoa: ”*node_modules\.bin\webpack --mode development*”. Vaihtoehtoisesti ”*--mode development*” voidaan korvata ”*--mode production*” määritteellä, joka on ensisijaisesti tarkoitettu käytettäväksi sovelluksen valmistuessa. Määritteellä ”production” saadaan niputetun tiedoston kokoa pienemmäksi, kun taas ”development” -tilassa on mahdollista saada enemmän tietoa, joka auttaa ohjelmoijia sovellusta kehitettäessä.

```
console.log('Hello World!')
```

KUVA 13. index.js

Niputuksen jälkeen havaitaan, että */dist* kansioon on ilmestynyt uusi tiedosto nimellä *main.js.* Tämän jälkeen tiedosto *index.html* voidaan avata selaimessa sekä sen jälkeen avataan selaimen kehittäjän työkaluista konsoli, jossa nähdään tiedostossa *index.js* määritelty tuloste. Tässä tapauksessa konsoliin tulostuu teksti ”Hello World!”.

3.2.4 Webpackin konfigurointi

Webpackin konfigurointi tapahtuu oletuksena JavaScriptillä, mutta sitä on mahdollista myös konfiguroida muita ohjelmointikieliä käyttämällä, kuten esimerkiksi TypeScriptiä. Konfigurointia varten tarvitaan konfigurointi tiedosto, jonka nimi on *webpack.config.js*. Se luodaan sovelluksen pääkansioon. Perus konfigurointia varten konfiguraatio tiedostoon määritetään haluttu niputus tila, jolloin vältytään edellisessä kappaleessa käydyltä sen

erikseen komentoriville kirjoittamiselta. Tämän lisäksi tiedostoon määritetään tiedosto, josta sovellus suoritetaan ja samalla mistä Webpack aloittaa niputtamisen sekä sen polku. Sen lisäksi määritetään kohdehakemisto tulostettavalle tiedostolle, ja nimi tuloste lohkojen aloituspisteelle. Tulosteen kohdalla hakemiston tiedostopolun täytyy olla täydellinen.

```
const path = require('path');

module.exports = {
  mode: 'development',
  entry: './src/index.js',
  output: {
    filename: 'main.js',
    path: path.resolve(__dirname, 'dist')
  }
};
```

KUVA 14. Webpackin konfigurointia

Konfiguroinnin onnistuminen voidaan testata komentokehotteella, syöttämällä komento: ”*node_modules\.bin\webpack --config webpack.config.js*”. Tällä komennolla Webpackille kerrotaan, että sen halutaan käyttävän sille luotua konfigurointitiedostoa.

```
"scripts": {
  "build": "webpack --config webpack.config.js",
```

KUVA 15. npm skriptausta

Jotta sovelluksen kehittäminen olisi entistä vaivattomampaa ja komentojen kirjoittaminen vähäisempää, voidaan syötettyjä komentoja lyhentää muokkaamalla tiedostoa *package.json*. Tämä tapahtuu npm-skriptin avulla (KUVA 15). Skriptiksi kirjoitetaan aiemmin komentoriville kirjoitettu komento, jonka jälkeen kirjoittamalla ”*npm run build*” -komento saadaan suoritettua niputus.

Sovelluksen kehittämistä on kuitenkin mahdollista tehdä vieläkin vaivattomammaksi webpack-dev-serverin eli WDS:n avulla. WDS:llä voidaan luoda paikallinen kehityspalvelin, joka mahdollistaa automaattisen sivunpäivityksen selaimessa ja vieläkin vähäisemmän komentojen kirjoittamisen. WDS:n asentaminen tapahtuu komentoriviä hyödyntäen komennolla: ”*npm install webpack-dev-server --save-dev*”. WDS:n käyttöönotto tapahtuu myös npm skriptin avulla (KUVA 16).

```
"scripts": {
  "build": "webpack --config webpack.config.js",
  "start": "webpack-dev-server"
```

KUVA 16. npm skripteihin lisättiin kehityspalvelimen käynnistys skripti

Kehityspalvelin käynnistetään komennolla ”*npm start*”, jonka jälkeen oletuksena palvelin toimii osoitteessa: *http://localhost:8080/.* Avaamalla edellä mainitun osoitteen selaimessa nähdään sama näkymä kuin aiemmin *index.html* -tiedoston avaamalla. Tästä eteenpäin tallentamalla tiedosto koodiin tehtyjen muutosten jälkeen selaimessa oleva näkymä vaihtuu automaattisesti. Muokattuja tiedostoja varten ei siis tarvitse erikseen kirjoittaa uusia komentoja komentoriville, eikä myöskään niputtaa tiedostoja uudestaan muutosten havainnoimiseksi. Niputtaminen tulee ajankohtaiseksi, kun halutaan saattaa sovellus valmiiksi ja siirtää se WWW-palvelimelle.

3.3 Vue-tiedostot

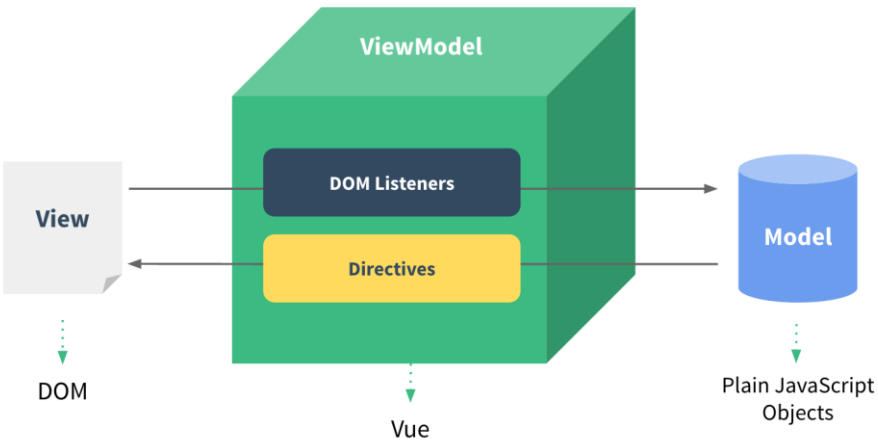
Vue:ssa komponenttien luontiin käytetään siihen erityisesti luotuja *.vue* -tiedostoja. *.vue* -tiedostot eroavat tavallisista JavaScript (*.js)* tiedostoista siten, että niiden sisälle voidaan määritellä JavaScript-koodia, HTML-malli sekä CSS-tyyliä. Näitä tiedostoja kutsutaan myös *Single File Component:ksi*. Jotta *.vue* -tiedostoja voidaan käyttää vaativat ne Webpackin toimiakseen. Webpackin mukana olo kuitenkin sallii mm. monien esikääntäjien käytön, kuten TypeScript ja SCSS.

```vue
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'app',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

KUVA 17. Esimerkki *.vue* -tiedostosta

3.4 Vue.js-arkkitehtuuri

Vue on suunniteltu MVVM malli-näkymä-näkymämalli-arkkitehtuurin pohjalta (KUVIO 1), mutta ei noudata sitä täysin ehdottomasti. Vue keskittyy näkymämalli kerrokseen, joka yhdistää näkymän ja mallin kaksisuuntaisella datan sidonnalla. Jokainen Vue-instanssi on näkymämalli. Näkymämalli sisältää myös sovelluksen tilan.



KUVIO 1. Vue.js-arkkitehtuurimalli (Vue.js 2019b)

Näkymäkerros on varsinainen dokumenttioliomalli (DOM), jota hallinnoidaan Vue-instansseilla. Näkymä siis sisältää käyttöliittymän sekä siihen kuuluvan logiikan. Datan muuttuessa näkymä päivittyy automaattisesti.

Vue:ssa mallikerros on vain hieman muokattu tavallinen JavaScript-objekti. Mallikerros sisältää datan sekä bisneslogiikan. Kun objekti on käytetty Vue instanssissa datana, tulee siitä reaktiivinen. Reaktiivisuudella tarkoitetaan näkymäkerroksen päivittymistä silloin, kun mallikerroksen dataa muutetaan. (Vue.js 2019b.)

3.5 Instanssin luonti

Seuraavaksi kun ohjelmointikehys eli tässä tapauksessa Vue.js on saatu asennettua ja projekti luotua sekä konfiguroitua haluamallaan tavalla, voidaan alkaa rakentamaan SPA-sovelluksen ensimmäistä näkymää.

Konfiguraatiotiedosto package.json:in lisäksi tiedostorakenne koostuu tässä vaiheessa pääosin seuraavista tiedostoista:

- *index.html*
- *src/index.js*
- *src/App.vue*

Näiden tiedostojen lisäksi projektissa tarvitaan *App.vue* tiedoston lisäksi muita *.vue* komponentti sekä reititys -tiedostoja, joista kerrotaan seuraavissa kappaleissa lisää.

```
1  import Vue from 'vue'
2  import App from './App.vue'
3
4  import { router } from './router';
5
6
7  new Vue({
8    el: '#app',
9    router: router,
10   render: h => h(App)
11 })
```

KUVA 18. index.js on projektin pää-JavaScript-tiedosto

Kuvassa 18 (KUVA 18) luodaan Vue-instanssi määrittelemällä sen *#app*-tunnisteella DOM-elementtiin, joka on määriteltynä *index.html*-tiedostossa (KUVA 19). Tämän jälkeen renderöimällä App.vue komponentti (KUVA 20) saadaan selaimeen sen sisältö (KUVA 21), joka toimii myös samalla sovelluksen juurikomponenttina. Tämä tarkoittaa sitä, että

myös muiden *.vue*-komponenttien ollessa renderöitynä on juurikomponentinkin näkymä esillä. Tästä syystä juurikomponenttiin on otollista sijoittaa esimerkiksi navigaatiopalkki, joka pysyy näkyvillä muiden komponenttien vaihtuessa.

```
<div id="app"></div>
```

KUVA 19. div-elementti

```
<template>
  <div id="app">
    <nav>
        <p> Hello World! </p>
    </nav>
    <router-view></router-view>
  </div>
</template>
```

KUVA 20. App.vue-komponentti

Sovelluksen ensimmäinen alkeellinen perusrakenne on valmis, kun edellä mainituilla toimenpiteillä selaimeen saadaan tulostettua alla olevan kuvan kaltainen teksti (KUVA 21).

KUVA 21. Hello World -sovellus

3.6 Reititys

Jotta SPA-sovelluksen sisällä näkymiä voitaisiin vaihdella, täytyy niiden välille luoda reititin. Reitittimen tehtävänä on vaihtaa sovelluksessa olevaa näkymää kulloinkin tarvittavan komponentin luomaan sisältöön sekä muuttamaan URL halutun kaltaiseksi.

Vue:lla reitittäminen on tehty yksinkertaiseksi Vue:n virallisen reitittimen Vue Router:in avulla. Se integroituu hyvin Vue.js ytimen kanssa ja näin ollen tekee SPA-sovelluksien rakentamisen vaivattomaksi (Vue Router 2019).

Reititintä varten täytyy luoda uusi JavaScript-tiedosto, joka voidaan nimetä kuvaavasti nimellä *router.js*. Tähän tiedostoon tuodaan kaikki komponentit, joita halutaan käyttää reitteinä. Sen lisäksi, kun käytössä on moduulijärjestelmä on reititin erikseen asennettava komennolla Vue.use() (KUVA 22).

```
import Vue from 'vue';
import Router from 'vue-router';

import FrontPage from '../views/FrontPage'
import User from '../views/User'

Vue.use(Router);

export const router = new Router({
  mode: 'history',
  routes: [
    { path: '/', component: FrontPage },
    { path: '/user/:id', component: User }
  ]
});
```

KUVA 22. Reititin

Reittiä vastaava komponentti renderöidään *router-view* elementissä (KUVA 20). Usein vastaan tulee kuitenkin tilanne, jolloin on tarpeellista, että sama komponentti täytyy määrittää erilaisilla tiedoilla. Esimerkiksi jokaisella käyttäjällä on yksilöity profiilisivu omilla tiedoilla, mutta ulkoasu muuten on sama, jolloin sama komponentti täytyisi renderöidä jokaiselle profiilille. Erottamalla jokainen käyttäjä omalle id-tunnisteelleen voidaan dynaamisesti erotella käyttäjät toisistaan. Nämä dynaamiset polkujen osat merkitään kaksoispisteellä (KUVA 22).

Oletuksena Vue Router käyttää ristikkomerkkejä URL:ssa, jotta sivu ei päivittyisi, kun URL vaihtuu osoiterivillä. Historia tilaa käyttämällä on kuitenkin mahdollista saada tavallisen näköinen URL ilman sivun päivitystä, kuten esimerkiksi ”*https://www.esimerkki.fi/user/id*”.

3.7 Data ja komponentit

Komponentit tai niistä muodostuvat näkymät ovat oleellinen osa SPA-sovellusta. Komponentit ovat uudelleen käytettäviä, eli ne eivät ole sidottuja vain yhteen näkymään, vaan voivat olla osana useampaa kokonaisuutta. SPA-sovelluksen yksittäiset näkymät toimivat ikään kuin perinteisen selainsovelluksen HTML-dokumenttien sijasta, sillä erotuksella, että SPA-sovelluksessa ei erillisiä sivulatauksia tapahdu.

Vue:lla yksisivuista selainsovellusta rakennettaessa usein on tapana luoda *.vue*-näkymätiedostoja, jotka itsessään ovat myös komponentteja, mutta näiden näkymäkomponenttien sisältö voi monesti koostua yhdestä tai useammasta erillisestä komponentista. Esimerkiksi edellisessä kuvassa (KUVA 22) voidaan nähdä, että sovelluksessa on käytetty kahta eri näkymäkomponenttia, eli toisin sanoen sovellus koostuu kahdesta toisistaan eroavasta näkymästä.

Useimmiten sovelluksissa halutaan myös näyttää jotain sisältöä, kuten kuvia tai tekstiä. Nykyisin sisältö harvoin pysyy samana koko ohjelman elinkaaren ajan, jolloin saattaa tulla tarve muuttaa sisältöä tai data voi tulla rajapinnasta sekä muuttua ajan kanssa. Tällaisessa tilanteessa kun data on dynaamista, ei ole kovinkaan mielekästä kirjoittaa tekstiä staattiseksi suoraan HTML-ohjelmakoodiin.

```
<h1>{{ header }}</h1>
```

KUVA 23. Deklaratiivista ohjelmointia

Ohjelmakoodin jatkuvalta muokkaamiselta välttyäkseen on käytännöllisempää tehdä se dynaamisesti. Vue:ssa dataa voidaan renderöidä deklaratiivisesti eli kerrotaan mitä halutaan eikä niinkään, että miten halutaan. Esimerkkinä kun halutaan antaa sivulle otsikko, kirjoitetaan otsikko elementtien sisään kahdet aaltosulkeet joiden sisälle annetaan halutun asian nimi, tässä tapauksessa otsikko (KUVA 23). Jotta aaltosulkeiden sisään saataisiin haluttu otsikko täytyy JavaScript ohjelmakoodiin kirjoittaa haluttu merkkijono (KUVA 24).

```
12  export default {
13    data() {
14      return {
15        header: 'Otsikko'
16      }
```

KUVA 24. Yksisuuntaista datan sidontaa

Edellä saatiin deklaratiivisesti haluttu otsikko yksisuuntaisen datan sidonnan avulla. Toisinaan käyttäjän on myös mahdollista syöttää tietoa esimerkiksi lomakkeen tai muun syötekentän muodossa. Tavallisesti muutoksia ei kuitenkaan nähdä dokumenttioliomallin sisällä, mutta kaksisuuntaisen datan sidonnan avulla on myös mahdollista muokata DOM:ia suoraan syötekenttään kirjoittamalla (KUVA 25). Yksinkertaisimmillaan kaksisuuntaisessa datan sidonnassa on kyse, että DOM:ssa tehdyt muutokset siirtyvät takaisin JavaScript-koodiin. Toisin sanoen se tarkoittaa sitä, kun data vaihtuu mallissa se heijastuu näkymään eli DOM:iin, ja kun data muuttuu näkymässä se päivittyy myös mallissa (W3Schools 2019a.).

```
<input v-model="header" placeholder="otsikoi">
<h1>{{ header }}</h1>
```

KUVA 25. Kaksisuuntaista datan sidontaa

Määritettä *v-model* käytetään syöte, tekstikenttä sekä valinta -elementeissä. Se korvaa edellä mainittujen elementtien arvo ja valinta -attribuutit ja käyttää Vue instanssin dataa totuuden lähteenä. (Vue.js 2019.)

```
<h2 v-if="pollData">
    {{ pollData[0] }}
</h2>
```

KUVA 26. Ehdollista renderöintiä

Dataa on myös mahdollista renderöidä ehtolauseiden avulla. Vue:ssa renderöintiin käytetään *v-if* määritystä. Silloin elementti lohkot renderöidään vain, jos lausekkeen palauttama arvo on tosi (KUVA 26), muussa tapauksessa elementtiä ei renderöidä. Vaihtoehtoista määritystä *v-else* voidaan myös käyttää, jos v-*if* ei päde. Periaate *v-if* ja *v-else* määrittteiden kanssa on sama kuin ohjelmoinnissa tavallisestikin *if* ja *else* -lauseilla.

```
1  <template>
2    <div>
3      <ul>
4        <li v-for="mon in months" v-bind:key="mon">
5          {{ mon.name }} {{ mon.temp }}
6        </li>
7      </ul>
8    </div>
9  </template>
10
11 <script>
12 export default {
13   data() {
14     return {
15       months:  [
16         {name: 'Huhtikuu', temp: 24.9},
17         {name: 'Toukokuu', temp: 31},
18         {name: 'Kesäkuu', temp: 32.8},
19         {name: 'Heinäkuu', temp: 37.2},
20         {name: 'Elokuu', temp: 33.8},
21         {name: 'Syyskuu', temp: 28.8},
22         {name: 'Lokakuu', temp: 20.9}
23       ]
24     }
25   }
26 }
27 </script>
```

KUVA 27. *v-for* toistorakenne ja yhden *.vue*-komponentin sisältämä data

Vue:ssa *loop* eli toistorakennetta voidaan toteuttaa *v-for* määrityksellä. Sen avulla voidaan käydä läpi taulukoita tai objekteja. Lisäämällä *v-for* määritys *li* lista elementtiin, voidaan käydä läpi *months* -taulukon objektit (KUVA 27) ja tulostaa ne sovelluksen HTML:ään (KUVA 28).

KUVA 28. *v-for*:lla luotu taulukko

4 JAVASCRIPT JA KIRJASTOT

JavaScript on tulkattava ohjelmointikieli. Tulkattava ohjelmointikieli tarkoittaa sitä, että tietokone ei ymmärrä sitä sellaisenaan kuin se on kirjoitettu, vaan se tarvitsee avukseen tulkin. Tulkki muuntaa ohjelmakoodin sellaiseen muotoon, jota tietokone ymmärtää. (Wilton & McPeak 2009, 2.)

Alun perin JavaScript kehitettiin vastaamaan selaimen toiminnallisuudesta. Nykyisin JavaScriptiä voidaan ajaa myös palvelimella tai missä tahansa laitteessa, jossa on JavaScript-moottori. JavaScript-moottori vastaa ohjelmakoodin suorittamisesta. (Javascript.info 2019.)

Ajan saatossa JavaScriptistä on kuitenkin kasvanut yksi suosituimmista ohjelmointikielistä, jolla voidaan tehdä niin suuren mittakaavan käyttäjäpuolen ohjelmistokehitystä kuin natiiveja mobiilisovelluksia tai jopa peliohjelmointiakin (Sheiko 2015, preface). JavaScript-sovellusten rakentamisen tueksi on tarjolla paljon vapaasti käytettäviä kirjastoja, jolloin kaikkia toiminnallisuuksia ei tarvitse tehdä itse. Kirjastoja on saatavilla moniin eri käyttötarkoituksiin. Kirjastoja löytyy niin visualisointiin ja käyttöliittymän rakentamiseen, kuin puhtaasti toiminnallisuuksiakin varten.

4.1 JSON

JavaScript Object Notation eli JSON on tiedostomuoto, joka on suunniteltu datan säilömiseen. JSON datan kirjoittaminen ja lukeminen on helppoa, ja useat ohjelmointiympäristöt osaavat lukea ja generoida JSON:a. JSON:a käytetään usein datan lähettämisessä palvelimelta verkkosivulle. JSON muistuttaa paljon JavaScript-objektia, mutta on kuitenkin täysin ohjelmointikieliriippumaton. (JSON 2019.)

```
{"user": "käyttäjä", "userid": 1}
```

KUVA 29. JSON-objekti

4.2 Node.js ja npm

Javascript on perinteisesti ollut selaimen päässä käytettävä ohjelmointikieli, mutta Node.js:llä JavaScript-koodia voidaan suorittaa myös palvelimen päässä. Node.js on siis palvelinpään ajonaikainen ympäristö, joka on rakennettu Google Chromen V8 JavaScript-moottoriin. Node.js on tapahtumapohjainen ja käyttää asynkronista I/O:ta eli se sallii muiden prosessien jatkumisen ennen kuin se itse on valmis. Tämä tekee Node.js:stä tehokkaan. (Patel 2018.)

Node.js package manager (npm) on paketinhallinta työkalu ja maailman suurin ohjelmistorekisteri sisältäen yli 800 000 koodipakettia (W3Schools 2019c). Npm:n avulla moduuleja on mahdollista julkaista muiden käyttöön sekä asentaa muiden julkaisuja omaan käyttöönsä. Esimerkiksi Vue.js on asennettavissa npm:llä. Npm vaatii Node.js:n toimiakseen ja onkin sen oletus paketinhallinta ohjelma.

4.3 Axios

Axios on JavaScript-kirjasto, jolla voidaan suorittaa HTTP-pyyntöjä, ja joka toimii selaimessa sekä Node.js alustoilla. Axios on promise-pohjainen, joten se sallii asynkronisen koodin käytön XMLHttpRequestien suorittamiseksi. Axios ei tarvitse erillistä polyfilliä toimiakseen. Polyfillillä tarkoitetaan koodia, joka implementoi toimintoja, jotka eivät alun perin ollut tuettuja vanhemmissa selaimissa. (Copes 2018a.)

Axiosin käyttöönotto aloitetaan asentamalla se ensin komennolla *npm install axios*, jonka jälkeen Axios tuodaan projektiin *import* -lausekkeella (KUVA 30). Tämän jälkeen Axiosilla voidaan tehdä erilaisia HTTP-pyyntöjä. Axios tukee kaikkia yleisimpiä HTTP-menetelmiä, mutta useimmiten niistä käytetään vain GET:ia ja POST:ia.

```
import axios from 'axios';
```

KUVA 30. Axiosin tuonti

Axiosissa GET-menetelmä voidaan suorittaa *axios.get()* ja vastaavasti POST-menetelmä *axios.post()* -pyynnön avulla. Pyynnölle usein halutaan myös vastaus ja se saadaan käyttämällä *then*-metodia, jonka avulla vastaus voidaan esimerkiksi tulostaa selaimen konsoliin *console.log()*-metodilla (KUVA 31).

```
axios.get('http://esimerkki.fi/api/user')
  .then(response => {
    console.log(response)
})
```

KUVA 31. Axios GET-pyyntö

Pyyntöön saatu vastaus sisältää datan lisäksi mm. myös HTTP-tilakoodin, otsikkotiedot ja pyynnössä annetut Axiosin konfigurointitiedot. HTTP-tilakoodi kertoo oleellista tietoa pyynnöstä, kuten oliko se onnistunut vai ei. Otsikkotiedoissa taas voidaan muun muassa määritellä autentikointitietoja.

4.4 Chart.js

Chart.js on avoimen lähdekoodin kirjasto datan kuvantamiseen. Chart.js:llä voi muun muassa kuvantaa dataa web-sovelluksiin erilaisten taulukoiden ja diagrammien avulla. Kuvaajat pohjautuvat HTML:n *canvas* -elementtiin ja ovat laajasti muokattavissa.

```
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
```

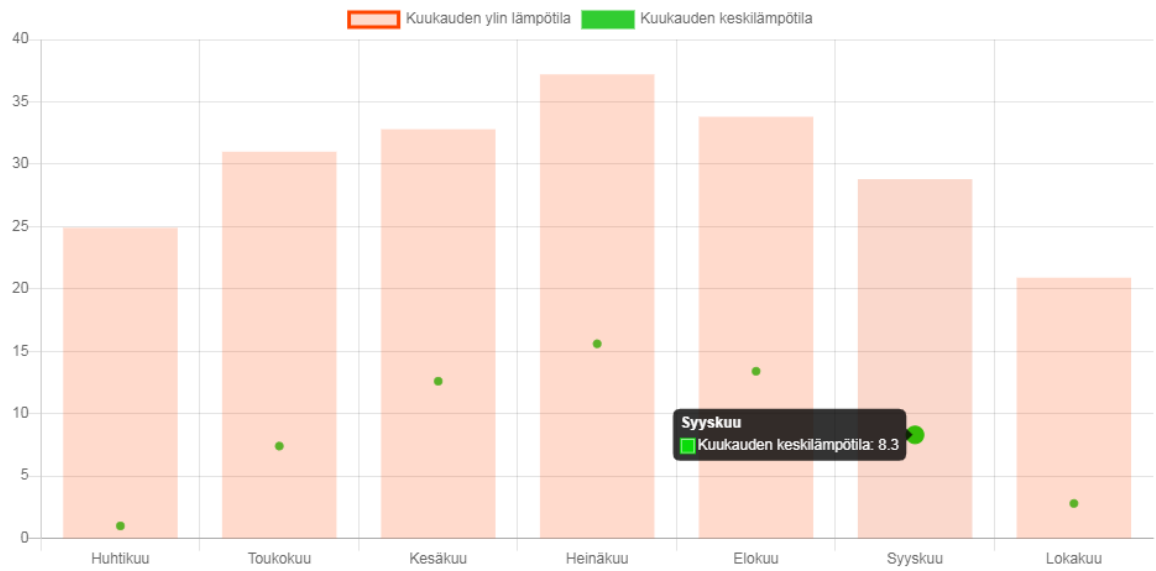
KUVA 32. Chart.js kirjaston lisääminen sovellukseen

Chart.js:n voidaan asentaa npm:llä käyttämällä komentoa: ”*npm install chart.js --save*”. Käyttöönotto voidaan suorittaa myös sisällyttämällä skripti suoraan sivulle (KUVA 32).

```
<canvas id="myChart"></canvas>
<script>
  var ctx = document.getElementById('myChart').getContext('2d');
  var chart = new Chart(ctx, {
      type: 'bar',
      data: {
          labels: ['Huhtikuu', 'Toukokuu', 'Kesäkuu', 'Heinäkuu', 'Elokuu', 'Syyskuu', 'Lokakuu'],
          datasets: [{
              label: 'Kuukauden keskilämpötila',
              backgroundColor: 'rgb(50,205,50)',
              borderColor: 'rgb(50,205,50)',
              data: [1, 7.4, 12.6, 15.6, 13.4, 8.3, 2.8]
          }]
      },
      options: {}
  });
</script>
```

KUVA 33. Chart.js kaavion luonti

Kaavion luonti tapahtuu lisäämällä *canvas* -elementti HTML-koodiin kaavion renderöimiseksi sekä halutun kaltaista kaaviota vastaava JavaScript-koodi. JavaScript-koodin avulla voidaan määrittää kaavion tyyppi ja antaa haluttu data sekä myös muokata kaavion ulkoasua muun muassa eri värein (KUVA 33).



KUVA 34. Chart.js:llä luotu kaavio

Chart.js:llä luotujen kaavioiden koko muokkautuu automaattisesti näyttökoon mukaan eli ne toimivat responsiivisesti. Kaikki kaaviot ovat myös animoituja. Yhdessä kaaviossa on mahdollista myös käyttää useampaa erilaista kaaviotyyppiä, ja täten näyttää useampaa dataa samaan aikaan (KUVA 34).

5 MUUT TEKNOLOGIAT

5.1 Tietokannat ja WWW-palvelimet

Kokonaisen web-sovelluksen rakentamiseen tarvitaan usein myös tietokanta. Web-sovelluksissa käytettävät tietokannat voidaan jakaa karkeasti kahteen kategoriaan, joista ensimmäinen on relaatiomalli sekä toinen siitä poikkeava ei-relaatio tietokanta. Ei-relaatio tietokannalla viitataan NoSQL-tietokantoihin, jotka useimmiten ovat skeemattomia eli niiltä puuttuu ennalta määritelty rakenne. Relaatiomallisella tietokannalla tässä yhteydessä tarkoitetaan SQL-tietokantoja. Tällaiset tietokannat koostuvat useista tauluista, ja niillä on jokin etukäteismääritelty rakenne. Tällä hetkellä yksi suosituimmista NoSQL-tietokantaohjelmistoista on MongoDB, kun taas MySQL on suosittu SQL-pohjainen relaatiotietokantaohjelmisto. (Hovi 2015.)

Tietokannan lisäksi sovellukselle tarvitaan WWW-palvelin, jotta sivusto tulisi myös muiden saataville. WWW-palvelimen tehtävänä on palauttaa selaimen tekemän pyynnön mukainen tiedosto HTTP:n välityksellä fyysiseltä web-palvelimelta takaisin selaimeen. (MDN Web Docs 2019b.)

WWW-palvelinohjelmistoista suosituimpia ovat Apache, Microsoftin IIS sekä NGINX. Nämä kolme muodostavat selvästi suurimman osan kaikkien WWW-palvelimien markkinaosuudesta. (Netcraft 2019.)

5.2 NGINX

NGINX on venäläisen Igor Sysoevin luoma ja 2004 julkaistu WWW-palvelin. NGINX on pitkään ollut käytössä monilla suurilla venäläisillä sivustoilla sekä muun muassa myös Netflixillä. (nginx 2019a.)

Alun perin NGINX luotiin ratkaisemaan C10K-ongelma. C10K-ongelmalla viitataan siihen, että web-palvelimella on haasteita käsitellä suurta määrää rinnakkaisia käyttäjiä, joka silloin tarkoitti kymmentä tuhatta samanaikaista yhteyttä. Koska NGINX:n taustalla on ollut tavoite optimoida suorituskykyä, on se myös suoriutunut WWW-palvelimien suorituskykytesteistä menestyksekkäästi. (RootUsers 2016; nginx 2019a.)

Perinteisesti WWW-palvelimet luovat yhden säikeen jokaiselle pyynnölle, mutta NGINX toimii toisin. NGINX käyttää asynkronista tapahtumapohjaista arkkitehtuuria. Se tarkoittaa sitä, että samanlaisia säikeitä hallitaan yhden työskentelijäprosessin alla, ja jokainen prosessi sisältää pienempiä yksiköitä, joita kutsutaan työskentelijäyhteyksiksi. Tämä koko yksikkö on vastuussa pyyntölankojen käsittelystä. Työskentelijäyhteydet toimittavat pyynnöt

työskentelijäprosessille, joka lähettää sen eteenpäin pääprosessille ja se lopulta välittää pyyntöjen tuloksen. Yksi työskentelijäyhteys pystyy käsittelemään jopa 1024 samanlaista pyyntöä. (Aldwin N. 2019.)

5.2.1 Konfigurointi

NGINX:n konfigurointiin perehdyttiin Windows-ympäristössä. Tällä hetkellä on kuitenkin todettava, että toistaiseksi kaikkia UNIX-pohjaisen version ominaisuuksia ei Windows-versiossa ole saatavilla. Tämän lisäksi Windows-versiossa on joitakin tunnettuja ongelmia, joten siihen tulee suhtautua vielä beeta versiona. Toiminnallisuudet ovat kuitenkin lähes samat kuin UNIX versiossa muutamaa poikkeusta lukuun ottamatta. Samat peruslainalaisuudet pätevät kuitenkin molempien konfigurointiin. (nginx 2019c.)

```
2   user  user;
3   worker_processes  1;
4   error_log  logs/error.log;
```

KUVA 35. Yksinkertaisia määrityksiä

Oletuksena NGINX:n konfigurointiasetukset löytyvät *nginx.conf*-tiedostosta. Konfiguraatiotiedosto koostuu määrityksistä. Määritykset on jaettu yksinkertaisiin- ja lohko määrityksiin (KUVA 35). Yksinkertaiset määritykset päättyvät puolipisteeseen (;) ja lohko määrityksiä seuraa aaltosulkeet ({}), jos niiden sisään on mahdollista asettaa muita määrityksiä, niitä kutsutaan konteksteiksi (KUVA 36). (nginx 2019b.)

```
16  http {
17      server {
18          listen       80;
19          server_name  192.168.100.1;
```

KUVA 36. Näitä lohko määrityksiä kutsutaan konteksteiksi

Oletuksena työskentelijäprosesseja on yksi, joka riittää ongelmitta rajoitetulle kohdeyleisölle, kuten yrityksen sisällä käytettävälle selainsovellukselle. Staattisen sisällön jakamiseksi tarvitsee konfiguroida palvelimen kuuntelema portti sekä antaa haluttu nimi palvelimelle, josta sen löytää. Jotta halutulta palvelimelta myös löytyisi sisältöä, täytyy se konfiguraatiossa määritellä. Määrittely tapahtuu *location*-lohkossa (KUVA 37.). *Root* eli juuri kohdassa määritetään tiedostopolku, josta tai johon jaettava tai ladattava tiedosto halutaan. Toisinaan on myös tarpeellista käyttää välityspalvelinta, esimerkiksi rajapintaa varten. Välityspalvelimen perus konfigurointi onnistuu myös suhteellisen vaivattomasti, lisäämällä halutulle sijainnille vain välityspalvelimen osoite. Lisäkonfigurointi mahdollisuuksia sen sijaan löytyy sitäkin enemmän, mutta alkuun pääsee varsin helposti.

```
26          location / {
27              root   D:/Users/nino/Apps/TestApp/dist;
28              index  index.html index.htm;
29          }
30
31          location /uploads/ {
32              root   D:/Users/nino/Apps/TestApp/upload;
33          }
34
35          location ^~ /api/ {
36              proxy_pass   http://localhost:6060/;
37              proxy_set_header   Host    $host;
38              proxy_set_header   X-Real-IP   $remote_addr;
39              proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
40              proxy_buffer_size 64k;
41              proxy_buffers   32 32k;
42              proxy_busy_buffers_size 128k;
43              proxy_connect_timeout       6000s;
44              proxy_send_timeout          6000s;
45              proxy_read_timeout          6000s;
46          }
```

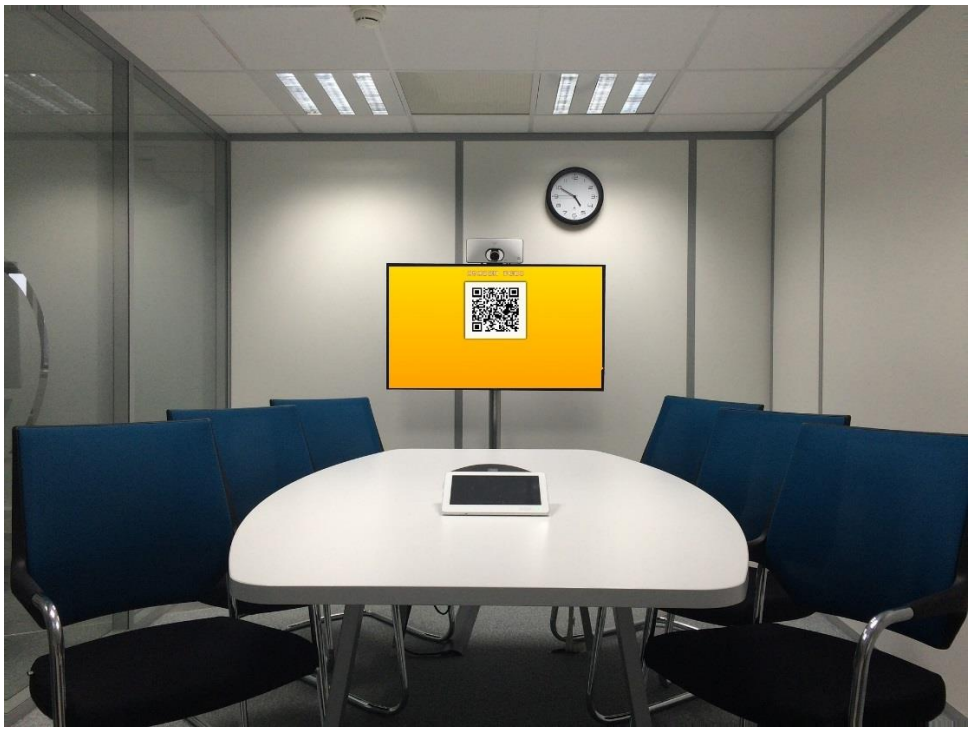
KUVA 37. Alimmaisena välityspalvelimen konfigurointia

5.2.2 Lisäkonfigurointi

Määritteellä *proxy_pass* määritetään välityspalvelimen osoite, jossa voidaan valita joko *http* tai *https* protokolla sekä valinnaisena portti. Palvelimelle lähetettävässä pyyntö otsikkoa voidaan määrittää *proxy_set_header:*in avulla. Palvelimelta palautuksena tulevaa vastauksen kokoa voidaan puskuroida *proxy_buffer_size, proxy_buffers* sekä *proxy_busy_buffers_size* määritteillä. Erilaisilla *timeout* määritteillä voidaan asettaa välityspalvelimelta saatavien vastausten aikakatkaisujen kesto. (nginx 2019c.)

6 ÄÄNESTYSSOVELLUS

Sovelluksen lähtökohtana oli luoda responsiivinen ja käyttäjäystävällinen äänestyssovellus, jonka avulla on mahdollista tehdä yrityksen sisällä tapahtuvia kyselyitä, kuten esimerkiksi kokoushuoneessa kaikkien työntekijöiden kesken tai yksittäisen tiimin sisällä (KUVIO 2). Tällä tavoin voidaan helposti ja nopeasti saada jokaisen mielipide kulloinkin vallitsevaan kysymykseen selville reaaliaikaisesti.



KUVIO 2. Havainnollistava kuva kokoushuoneesta (mukailtu jraffin 2016)

Vaatimuksina sovellukselle oli mahdollisuus lisätä kuva kyselyyn, vastausvaihtoehtojen määrä tulisi olla kyselyn luojan määritettävissä, kyselyyn vastaaminen tulisi olla mahdollista lukemalla QR-koodin sekä tulokset täytyisi olla mahdollista nähdä automaattisesti päivittyvässä diagrammissa.

Sovelluksesta haluttiin tehdä nimenomaan selainpohjainen, sillä mobiilisovellus olisi tässä tapauksessa ollut vähemmän käyttäjäystävällinen johtuen siitä, että jokaisen täytyisi tällöin erikseen ladata kyseinen sovellus puhelimeensa, sekä haluttiin jättää mahdollisuus myös käyttää sovellusta tietokoneella. Myöskään QR-koodin lukemisen kannalta ei erilliselle mobiilisovellukselle ollut tarvetta, sillä Kiinassa erittäin suosituissa monikäyttöisessä mm. sosiaalisen median, pikaviestin sekä mobiilimaksamis -applikaatio WeChatissa on myös sisäänrakennettu QR-koodin lukija, jota käytetään paljon myös muihin QR-koodien lukemiseen. Täten voidaan olettaa, että jokaisella tämän äänestyssovelluksen käyttäjällä on myös QR-koodin lukija käytössään.

6.1 Kyselyn luominen

Sovelluksen avautuessa selaimeen ensimmäisenä näkymänä on suoraan tyhjä lomakekenttä (KUVA 38), jonka täyttämällä käyttäjä saa luotua kyselyn. Lomakekentän ylimpänä on valinnainen kuvan lisäyspainike, josta voi halutessaan liittää kuvan kyselyyn. Sen lisäksi lomakkeessa on pakollinen kysymyskenttä sekä vapaasti valittava määrä vastausvaihtoehtoja. Jokaista sovelluksessa olevaa näkymää yhdistää keskellä sivun ylälaidassa oleva Avalon poll -tunnus, jossa olevat kirjaimet pyörähtävät kuin hedelmäpelin lailla oikeille paikoilleen. Kyseinen animaatio on toteutettu CSS:llä.

KUVA 38. Kyselylomake

Painamalla ”Create Poll” -painiketta lomakkeen tiedot lähetetään palvelimelle Axios moduulin avulla. Axios on promise-pohjainen HTTP-client, jonka avulla voi tehdä HTTP-pyyntöjä niin selaimessa kuin node.js alustallakin. Axiosta käytettiin siis tekstin ja kuvan talletamiseksi tiedostojärjestelmään.

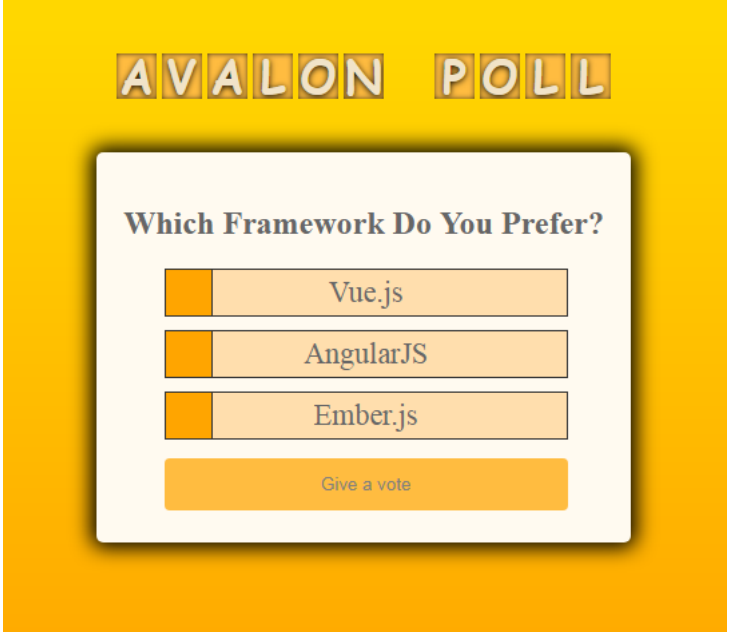
6.2 QR-koodin luominen ja lukeminen

QR-koodin luomiseen käytettiin qrcode.vue nimistä valmista single-file komponenttia, joka on asennettavissa muun muassa npm:llä ja jonka ominaisuuksia on kuitenkin mahdollista muokata haluamallaan tavalla. Tämä komponentti generoi datasta saatujen tietojen avulla oikeanlaisen QR-koodin, joka ohjaa käyttäjän äänestysnäkymään.

KUVA 39. QR-koodi

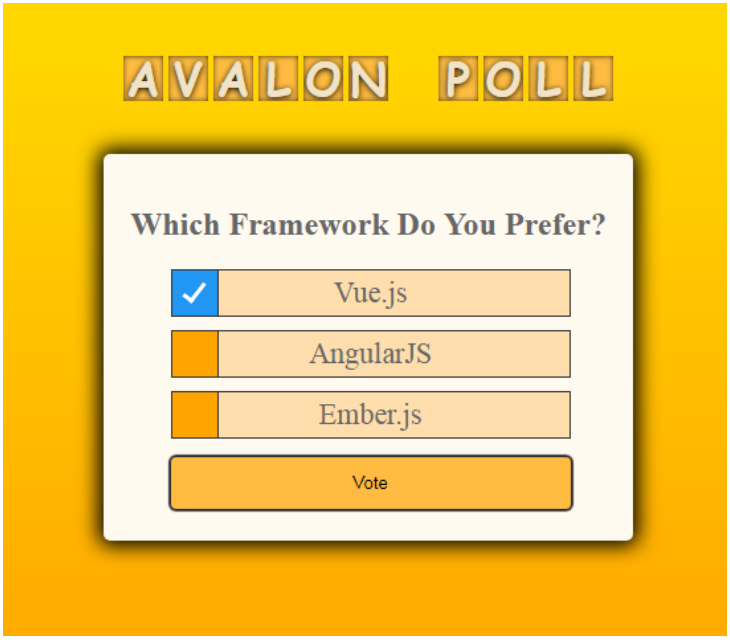
QR-koodin lukemiseen käytetään pääsääntöisesti puhelimessa olevaa QR-koodin lukijaa, jolla useampi henkilö voi sen lukea samaan aikaan esimerkiksi television ruudulta etäisyydenkin päästä ja siten saa kyselyn avattua omassa puhelimessaan. Jos QR-koodin lukijaa ei ole käytössä tai linkin saa jaettuna kuvan 39 kaltaisella näkymällä (KUVA 39) tietokoneelle tai mobiililaitteeseensa, voidaan QR-koodia käyttää myös linkkinä painettaessa, jolloin yhtä lailla käyttäjä ohjautuu äänestämisnäkymään.

6.3 Äänestäminen



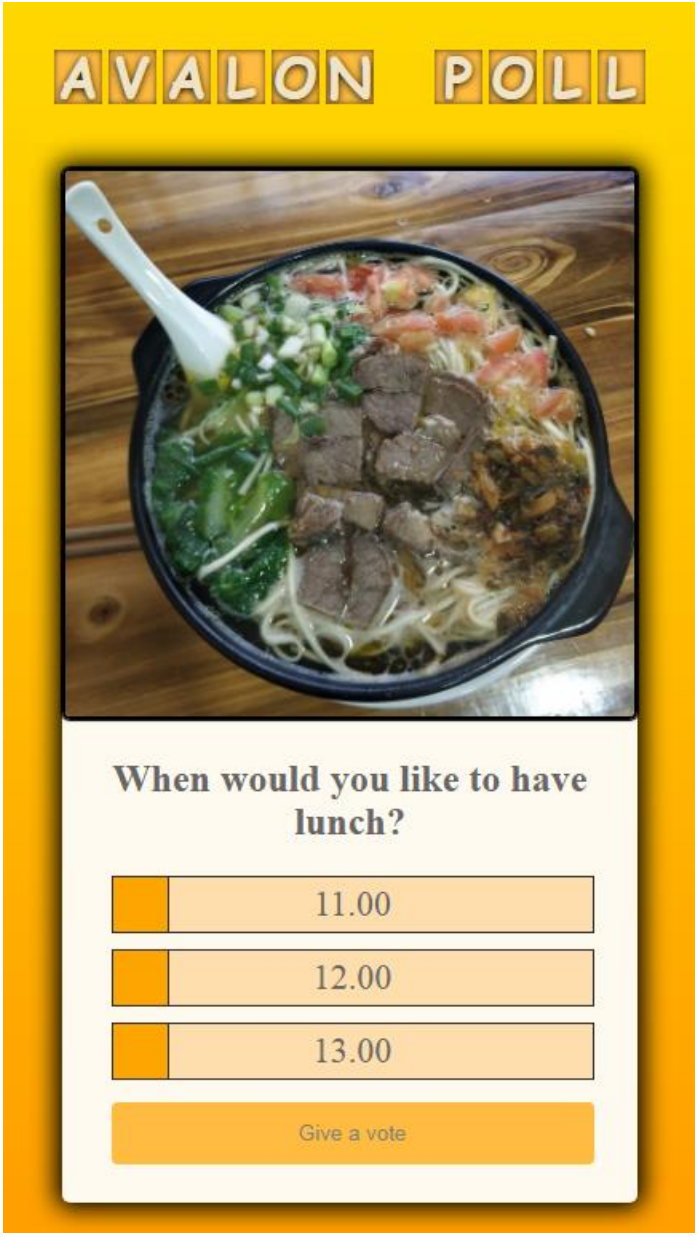
KUVA 40. Äänestysnäkymä

Äänestysnäkymän ulkoasusta haettiin mahdollisimman selkeää ja asiallista, mutta kuitenkin tyylikästä siten, että se noudattaa yrityksen värimaailmaa, kuten myös koko muukin sovellus. Ulkoasu vaihtelee riippuen siitä, onko kyselyyn lisätty kuvaa vai ei. Koska sovellus on responsiivinen, niin ulkoasu vaihtelee myös riippuen siitä onko käytössä mobiililaite vaiko perinteinen työpöytäkone (KUVA 43). Mobiililaitetta käytettäessä ulkoasu on optimoidumpi pienempää ruutukokoa silmällä pitäen (KUVA 42).

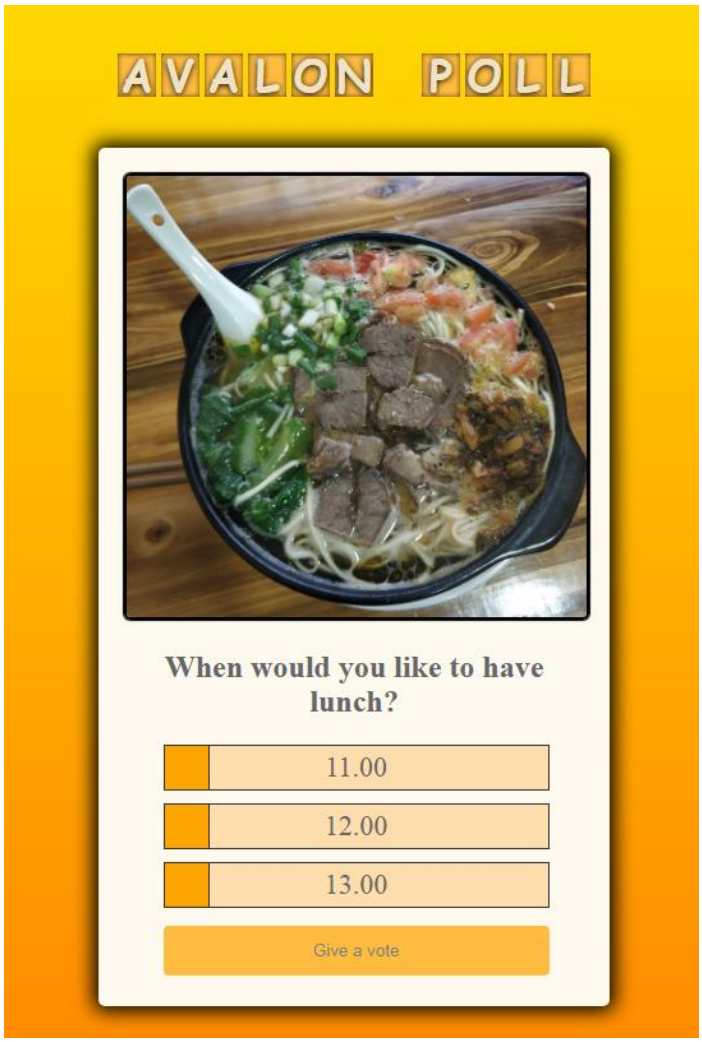


KUVA 41. Vastausvaihtoehto valittuna

Kuvan ollessa lisättynä kyselyyn asettuu se kysymyksen yläpuolelle (KUVA 42). Ilman kuvaa olevassa kyselyssä kysymys on ylimpänä (KUVA 40), jonka alle asettuvat vastausvaihtoehdot yhtä lailla kuin kuvankin kanssa. Vastausvaihtoehtoja valittaessa vaihtoehtojen vasemmalla puolella olevaan neliönmuotoiseen oranssiin laatikkoon ilmestyy sinipohjainen indikaattori valitun vaihtoehdon kohdalle (KUVA 41). Vastausvaihtoehtojen alle sijoitettu äänenantopainike on pois käytöstä siihen asti kunnes, jokin vastausvaihtoehdoista on valittu ja tällöin siitä tulee aktiivinen. Tämän jälkeen painiketta painettaessa annetaan ääni, ja tieto tästä lähetetään palvelimelle, jonka kautta se siirtyy MySQL tietokantaan. Samanaikaisesti näkymä vaihtuu äänestystuloksiin.



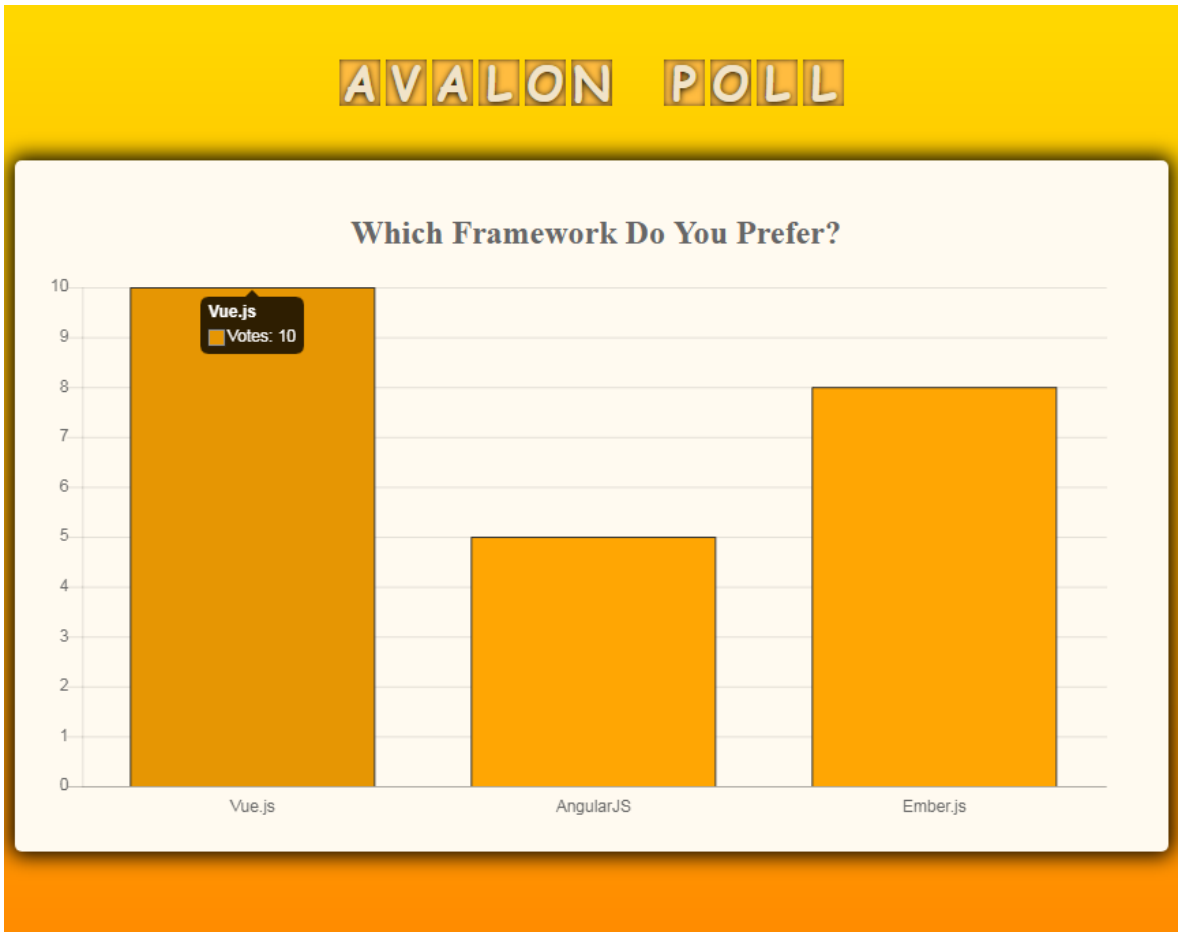
KUVA 42. Mobiilinäkymä



KUVA 43. Työpöytänäkymä

6.4 Tulokset

Kun äänestäjät ovat QR-koodin laitteillansa lukeneet ja siirtyneet äänestämään, voidaan tulosnäkymä avata kaikkien nähtäville yhteiselle näytölle, ja seurata äänestysprosessin etenemistä sekä miten äänet jakautuvat vastausvaihtoehtojen välille. Lopulta kun kaikki ovat äänensä antaneet voidaan kaikkien asianosaisten kesken tarkastella tuloksia ja tehdä valintoja tai päätöksiä niihin pohjautuen.



KUVA 44. Tulosnäkymä

Äänestystulosnäkymää hallitsee pylväsdiagrammi, jossa jokaiselle äänestysvaihtoehdolle on olemassa oma pylväänsä. Viemällä osoittimen pylvään päälle saa siitä lisätietoa siihen ilmestyvän kuplan muodossa (KUVA 44). Tulokset diagrammissa päivittyvät automaattisesti ennalta määritetyn aikajakson välein. Muutoin ulkoasu on varsin samankaltainen edellisen näkymän kanssa.

Pylväsdiagrammi on tehty käyttäen Chart.js-nimistä diagrammikomponenttia. Komponentilla voi tehdä monenlaisia räätälöityjä ja skaalautuvia taulukoita. Tässä projektissa data on tallennettu tietokantaan, ja täten tieto diagrammeihin saadaan ohjelmointirajapinnan kautta. Diagrammi renderöidään vasta, kun data saapuu käyttäjälle.

7 YHTEENVETO

Tässä opinnäytetyössä tavoitteena oli luoda Avalonille äänestämiseen tarkoitettu SPA-sovellus käyttämällä nykyaikaisia web-teknologioita. Samalla työssä perehdyttiin myös WWW-palvelimen sekä moduulien niputtaja Webpackin asennukseen ja konfigurointiin.

Teknologioiksi valikoitui Vue.js selainpuolen ohjelmointikehykseksi sekä NGINX WWW-palvelimeksi. Teknologia valintoihin vaikutti se, että NGINX on ollut suorituskyvyltään erittäin kilpailukykyinen muita WWW-palvelin teknologioita vastaan. Vue.js:n kohdalla keveys, nopeus sekä sen muunneltavuus vaikuttivat valintaan. Tämän lisäksi edellä mainitut teknologiat vaikuttivat sopivilta projektin toteuttamisen ja onnistumisen kannalta sekä kummankin suosio on ollut viime aikoina selvässä nousussa.

Äänestyssovelluksen valmistuttua jo ensi testauksilla voitiin todeta, että käyttöliittymän ulkoasu on selkeä ja vastaa yrityksen värimaailmaa sekä kyselyiden luonti ja sovelluksen käytettävyys on vaivatonta ja yksinkertaista. Teknologia valinnat osoittautuivat myös onnistuneiksi, ja ne eivät toimineet rajoitteina tai hankaloittavina tekijöinä sovelluksen luomisessa vaan se onnistui sujuvasti.

Toistaiseksi pidempiaikaisia käyttäjäkokemuksia ei sovelluksen käytöstä ole ehditty saamaan, jotta niiden pohjalta voitaisiin tarkemmin arvioida jatkokehityskohteita. Potentiaalisena jatkokehityskohteena voisi kuitenkin toimia esimerkiksi mahdollisuus yksilöidä kyselyitä oman käyttäjän alle, jotta niihin voi helpommin palata myöhemmin.

LÄHTEET

Aldwin N. 2019. What is NGINX? How does it work? [viitattu 30.3.2019]. Saatavissa: https://www.hostinger.com/tutorials/what-is-nginx

blog.pshrmn 2018. How Single-Page Applications Work [viitattu 25.3.2019]. Saatavissa: https://blog.pshrmn.com/entry/how-single-page-applications-work/

Copes, F. 2018a. HTTP requests using Axios [viitattu 13.4.2019]. Saatavissa: https://flaviocopes.com/axios/

Copes, F. 2018b. Introduction to Webpack [viitattu 1.4.2019]. Saatavissa: https://flaviocopes.com/webpack/

Copes, F. 2018c. The Vue Handbook: a thorough introduction to Vue.js [viitattu 31.3.2019]. Saatavissa: https://medium.freecodecamp.org/the-vue-handbook-a-thorough-introduction-to-vue-js-1e86835d8446

Google Analytics 2018. Single Page Application Tracking [viitattu 26.3.2019]. Saatavissa: https://developers.google.com/analytics/devguides/collection/analyticsjs/single-page-applications

Google Play 2019. The Greedy Cave [viitattu 29.3.2019]. Saatavissa: https://play.google.com/store/apps/details?id=com.avalon.cave&hl=en

Hovi, A. 2015. MongoDB haastaa relaatiokantoja [viitattu 29.3.2019]. Saatavissa: https://www.arihovi.com/mongodb-haastaa-relaatiokantoja/#

Javascript.info 2019. An Introduction to JavaScript [viitattu 29.3.2019]. Saatavissa: https://javascript.info/intro

jraffin 2016. Image by jraffin from Pixabay [viitattu 26.3.2019]. Saatavissa: https://pixabay.com/photos/meeting-room-table-business-1806702/

MDN Web Docs 2019a. Using the viewport meta tag to control layout on mobile browsers [viitattu 14.4.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

MDN Web Docs 2019b. What is a web server? [viitattu 29.3.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

Netcraft 2019. February 2019 Web Server Survey [viitattu 29.3.2019]. Saatavissa: https://news.netcraft.com/archives/2019/02/28/february-2019-web-server-survey.html

nginx 2019a. nginx [viitattu 30.3.2019]. Saatavissa: https://nginx.org/en/

nginx 2019b. Beginner's Guide [viitattu 30.3.2019]. Saatavissa: http://nginx.org/en/docs/beginners_guide.html

nginx 2019c. Module ngx_http_proxy_module [viitattu 9.4.2019]. Saatavissa: https://nginx.org/en/docs/http/ngx_http_proxy_module.html

nginx 2019d. nginx for Windows [viitattu 30.3.2019]. Saatavissa: http://nginx.org/en/docs/windows.html

Patel 2018. What exactly is Node.js? [viitattu 31.3.2019]. Saatavissa: https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5

RootUsers 2016. Linux Web Server Performance Benchmark – 2016 Results [viitattu 30.3.2019]. Saatavissa: https://www.rootusers.com/linux-web-server-performance-benchmark-2016-results/

Sheiko, D. 2015. JavaScript Unlocked. Packt Publishing.

Vue CLI 2019. Plugins and Presets [viitattu 13.4.2019]. Saatavissa: https://cli.vuejs.org/guide/plugins-and-presets.html

Vue.js 2019a. Basic Usage [viitattu 9.4.2019]. Saatavissa: https://vuejs.org/v2/guide/forms.html

Vue.js 2019b. Getting Started [viitattu 12.4.2019]. Saatavissa: https://012.vuejs.org/guide/

Vue.js 2019c. Introduction [viitattu 31.3.2019]. Saatavissa: https://vuejs.org/v2/guide/index.html

Vue Router 2019. Introduction [viitattu 27.3.2019]. Saatavissa: https://router.vuejs.org/

W3Schools 2019a. AngularJS Data Binding [viitattu 28.3.2019]. Saatavissa: https://www.w3schools.com/angular/angular_databinding.asp

W3Schools 2019b. Responsive Web Design - Grid-View [viitattu 14.4.2019]. Saatavissa: https://www.w3schools.com/css/css_rwd_grid.asp

W3Schools 2019c. What is npm? [viitattu 31.3.2019]. Saatavissa: https://www.w3schools.com/whatis/whatis_npm.asp

Webpack 2019. Configuration [viitattu 1.4.2019]. Saatavissa: https://webpack.js.org/configuration/

Wilton, P. & McPeak, J. 2009. Beginning JavaScript. Hoboken: Wrox.