Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Aleksi Suominen

# Reactin, Angularin ja Vuen vertailua

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

25.5.2020

| | |
|---|---|
| Tekijä<br>Otsikko<br><br>Sivumäärä<br>Aika | Aleksi Suominen<br>Reactin, Angularin ja Vuen vertailua<br><br>37 sivua<br>25.5.2020 |
| Tutkinto | Insinööri (AMK) |
| Tutkinto-ohjelma | Tieto- ja viestintätekniikka |
| Ammatillinen pääaine | Ohjelmistotuotanto |
| Ohjaajat | Juha-Pekka Kämäri |


Insinöörityössä oli tavoitteena vertailla kolmea tällä hetkellä suosituinta JavaScriptin frontendissä käytettävää sovelluskehystä ja kirjastoa. Verrattavana oli Facebookin kehittämä JavaScript-kirjasto React, Googlen kehittämä JavaScript-sovelluskehys Angular ja Evan Youn kehittämä JavaScript-sovelluskehys Vue.js.Vue.js on näistä kolmesta teknologiasta uusin, ja Vuen suosio on koko ajan kasvamassa.

Tällä hetkellä React on näistä suosituin. Vue.js toisena ja Angular kolmantena. Reactilla, Angularilla ja Vuella rakennetaan SPA (Single Page Application) -sovelluksia. Työn aikana saatiin aikaan kolme samankaltaista sovellusta, joita vertailtiin keskenään. Sovellukset olivat aika yksinkertaiset muistiosovellukset, joissa on CRUD-toimenpiteet.

Vertailussa selvisi, että varsinaisesti parasta ei näistä kolmesta voi oikein valita ja oma suosikki riippuu omasta mielipiteestä. Kaikissa kolmessa kuitenkin rakennetaan käyttöliittymä komponenteista. Reactissa ja Vuessa komponentit ovat yhdessä tiedostossa ja Angularissa kolmessa.

React ja Vue myös käyttävät JavaScriptiä ja Angular TypeScriptiä. Angular on myös koko ratkaisu web-sovelluksen frontendiin ja sisältää paljon ominaisuuksia. Tästä syystä Angular vaatii enemmän aikaa opetella kuin React ja Vue.js. React ja Vue.js myös käyttävät Virtual DOM:ia ja Angular ei.



| | |
|---|---|
| Avainsanat | React, Angular, Vue.js |

| | |
|---|---|
| Author<br>Title<br><br>Number of Pages<br>Date | Aleksi Suominen<br>Comparing React, Angular and Vue.js<br><br>35 pages<br>25 May 2020 |
| Degree | Bachelor of Engineering |
| Degree Programme | Information and Communication Technology |
| Professional Major | Software Engineering |
| Instructors | Juha-Pekka Kämäri |

The aim of the thesis was to compare the three currently most popular JavaScript frameworks and libraries used in the frontend web development. React, Angular and Vue were compared. React is Javascript-library made by Facebook. Angular is framework made by Google. Vue.js is JavaScript-framework made by Evan You. Vue.js is latest technology and Vue's popularity is growing.

Currently, React is the most popular of these. Vue.js is second and Angular is third. Applications built using React, Angular and Vue.js are SPA (Single Page Application) applications. During work three similar applications were made and compared with each other. The applications were simple notepad-applications with CRUD operations.

The comparison revealed that the best of the three cannot be chosen correctly and own favorite depends on own opinion. However, all three build an user interface from components. In React and Vue, the components are in one file and in Angular in three.

React and Vue also use JavaScript and Angular uses TypeScript. Angular is also a complete solution for the web application frontend and includes a lot of features. For this reason, Angular requires more time to learn than React and Vue.js. React and Vue.js also use the Virtual DOM and Angular does not.

| | |
|---|---|
| Keywords | React, Angular, Vue.js |

## Sisällys

## Lyhenteet

| | |
|---|---|
| Frontend | Asiakas eli selainpuoli websovelluksesta. |
| DOM | Document Object Model. Puurakenne, jolla voidaan kuvata esimerkiksi HTML-sivun rakenne. |
| React | Facebookin kehittämä sovelluskirjasto websovellusten käyttöliittymän rakentamiseen. |
| Angular | Googlen kehittämä frontend:issa käytettävä sovelluskehys. |
| Vue.js | Frontend:issä käytettävä sovelluskehys. |
| CLI | Command Line Interface. Komentorivikäyttöliittymä. |
| API | Application Programming Interface. Rajapinta. |
| SPA | Single Page Application. Yhden sivun sovellus. Sovellus on yhdessä div-tagissa index.html:n sivulla. |
| Node.js | JavaScript runtime, joka on kehitetty Chromen V8 JavaScript-moottorin päälle. |
| NPM | Node Package Manager. Node.js:n pakettienhallintaohjelma. |
| TypeScript | Microsoftin kehittämä JavaScriptin superset. |
| CRUD | Create, Read, Update ja Delete. Luonti, lukeminen päivittäminen ja poistaminen. |
| UI | User Interface. Käyttöliittymä. |
| Backend | Serveri eli palvelinpuoli websovelluksesta. |

| | |
|---|---|
| Full Stack | Websovellus, jossa molemmat puolet frontend ja backend. |

# 1 Johdanto

Insinöörityöni aiheena on React-, Angular- ja Vue.js Javascript-sovelluskehysten ja kirjastojen vertailu. Reactista, Angularista ja Vue.js:stä on tullut lähiaikoina kolme suosituinta frontendissä käytettävää sovelluskehystä. Vue.js on näistä kolmesta teknologiasta uusin, ja Vuen suosio on koko ajan kasvamassa. Tällä hetkellä React on näistä suosituin. Vue.js toisena ja Angular kolmantena.

Reactilla, Angularilla ja Vuella rakennetaan SPA (Single Page Application) -sovelluksia. Insinöörityössäni vertailen näiden sovelluskehysten eroja. Angularissa yksi iso ero Reactiin ja Vue.js:ään on, että Angular käyttää Javascriptin sijasta TypeScriptiä. Teen myös lähes samanlaisen sovelluksen kaikilla näillä kolmella teknologioilla. Vertailen myös näitä kolmea eri sovellusta toisiinsa. Lähteenä käytin enimmäkseen Reactin(reactjs.org), Angularin(angular.io) ja Vuen(vuejs.org) virallisia dokumentaatioita.

En pane insinöörityössäni paremmuusjärjestykseen sovelluskehityksiä vaan vertailen sovelluskehysten ja kirjastojen eroja ja yhtäläisyyksiä. React ja Vue.js varsinkin ovat mielestäni monelta osin aika samankaltaisia. Angularin ensimmäinen versio on AngularJS, joka on eri sovelluskehys eikä enää yhteensopiva uuden Angular(2+) sovelluskehyksen kanssa. Angular eroaa myös siinä, että Angular-komponentit ovat kolmessa eri tiedostossa ja Reactin ja Vuen vain yhdessä.

## 2 JavaScript

JavaScript on tulkittu(”interpreted”) ohjelmointikieli eli skriptikieli. JavaScriptin on kehittänyt Brendan Eich. JavaScript pyörii normaalisti selaimessa, mutta myös nykyään palvelimella Node.js:n avulla. Javascript mahdollistaa myös mobiilisovellusten kehityksen käyttämällä esimerkiksi React Nativea tai NativeScriptiä. JavaScriptillä voi kehittää myös työpöytäsovelluksia Electron-sovelluskehyksellä.

Javascriptin standardi on ECMAScript. Vuodesta 2012 kaikki nykyaikaiset selaimet tukevat täysin ECMAScript 5.1:tä. ECMA International julkaisi 17. kesäkuuta 2015 ECMAScriptin kuudennen pääversion, jota kutsutaan virallisesti ECMAScript 2015, mutta sitä kutsutaan myös nimellä ECMAScript 6 tai ES6. ES6:ssa tuli paljon uusia ominaisuuksia kuten let- ja const-muuttujat ja nuolifunktiot. Siitä lähtien ECMAscriptistä on julkaistu vuosittain uusi versio. Tällä hetkellä ECMAScript 2020 on ECMAScritin uusin versio. [1]

JavaScriptille on kehitetty paljon sovelluskehyksiä ja kirjastoja. Frontendissä käytetyistä suositummat ovat tällä hetkellä React, Angular ja Vue.js.

# 3 React

React on Facebookin kehittämä Javascript-kirjasto käyttöliittymien rakentamiseen. React on tällä hetkellä suosituimpia frontendissä käytettäviä teknologioita. Reactia saatetaan usein kutsua sovelluskehykseksi ja on verrattavissa sovelluskehyksiin kuten Angular ja Vue.js.

React käyttää virtual DOM:ia. Virtuaalinen DOM on ohjelmointikonsepti, jossa käyttöliittymän ideaali tai virtuaali esitys pidetään muistissa ja synkronoidaan todellisen DOM:in kanssa kirjastolla kuten ReactDOM.

Koska React on kirjasto Reactin lisäksi React-sovelluksessa käytetään usein myös muita kirjastoja. Tilanhallintaan voidaan käyttää esimerkiksi Reduxia. HTTP-pyyntöihin voidaan käyttää esimerkiksi Axios-kirjastoa tai Fetch API:a. React-sovelluksen voi luoda esimerkiksi Create React App -työkalulla. Create React App on CLI-työkalu, jolla voi luoda React-sovelluksia. Reactille on myös oma DevTools (kuva 1). Create React App käyttää Webpackia, mutta ei vaadi käyttäjältä mitään konfigurointia. Sisältää ”hot reload” -ominaisuuden eli aina, kun tallentaan, sivu päivittyy automaattisesti eikä sivua tarvitse ladata uudelleen.


Kuva 1. React DevTools.

## 3.1 Reactin historia

Reactin kehitti alun perin Facebookin työntekijä Jordan Walke. Facebook alkoi 2010 käyttämään XHP:tä PHP-koodissa. XHP teki frontend-koodista ymmärrettävämpää ja auttoi välttämään cross-site-scripting-hyökkäyksiä. Jordan Walke julkaisi Reactin varhaisen prototyypin nimeltä ”FaxJS”. Facebook-Ads oli 2012 kasvanut isoksi ja koodia oli vaikea hallita ja Facebook halusi siihen hyvää ratkaisua. Reactin alkuperäinen julkinen julkaisu oli 29. toukokuuta 2013. [2.]

## 3.2 Reactin ominaisuudet

### 3.2.1 JSX

JSX on JavaScriptin syntaksilaajennus. Sitä suositellaan käyttämään Reactin kanssa, vaikka Reactia on myös mahdollista käyttää myös ilman JSX:ää. JSX tuottaa React-elementtejä. React yhdistää sovelluksen merkkauskielen ja logiikan samaan komponenttiin. JSX mahdollistaa minkä tahansa JavaScript-lausekkeen laittamisen kaarisulkeiden sisään. JSX käyttää camelCasea, joten esimerkiksi class on className ja background-color on backgroundColor. [3.]

Toisin kuin selaimen DOM-elementit, React-elementit ovat objecteja, joiden luominen on halpaa. React käyttää Virtual DOM:ia ja  Reactissa React DOM huolehtii DOM:in päivittämisestä. [4.]

### 3.2.2 Komponentit, props ja state

Komponenttien avulla voidaan jakaa käyttöliittymä itsenäisiksi, uudelleen käytettäviksi ja erillisiksi osiksi. Reactissa komponentit voi luoda joko luokkakomponenttina tai funktiokomponenttina. Komponenttien nimet alkavat Reactissa aina isolla kirjaimella.

Yleensä React-sovelluksissa on yksi App-komponentti, jonka sisällä muut komponentit ovat. Komponenteille voidaan määritellä propseja, jotka ovat lyhenne propertieseista. Propsit ovat vain lukutyyppisiä. [5.]

Reactissa normaalisti vain luokkana luodussa komponentissa voidaan käyttää statea. Hooks-ominaisuutta käyttämällä myös funktiokomponentissakin voi käyttää statea. State esitellään aluksi luokkakomponentin konstruktorissa. Tilaa ei saa muuttaa suoraan vaan sitä pitää muuttaa setState-funktiolla. Tilaa näkyy ainoastaan sille komponentille, jossa se on määritelty. Usein kuitenkin muiden komponenttien pitää tietää toisen komponentin tila. Silloin tila voidaan laittaa propsina alaspäin muille komponenteille. Reactissa on käytössä siis yksisuuntainen data virta(one way data flow) eli yhdensuuntainen tiedonsitominen(one way databinding). [6.]

```
import React, { Component } from 'react'

export default class Form extends Component {
  constructor(props) {
    super(props)
    this.state = {
      value: ''
    }
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  handleChange(e) {
    this.setState({ value: e.target.value });
  }

  handleSubmit(e){
    e.preventDefault();
    console.log(this.state.value);
  }

  render() {
    return (
      <div>
        <form onSubmit={this.handleSubmit}>
          <label>
            <input type="text" value={this.state.value} on-
Change={this.handleChange} />
          </label>
          <input type="submit" value="Submit" />
        </form>
      </div>
    )
  }
}
```

Esimerkkikoodi 1. React-luokkakomponenttiesimerkki, jossa käytetään statea. Esimerkissä esitellään aluksi luokan konstruktorissa state Käyttäjän antama arvo logataan konsoliin.

```
import React, { Component } from 'react'
import ListComponent from './components/ListComponent'
export default class App extends Component {
  constructor(props) {
    super(props)
    this.state = {
       items: [
         {
           id: 1,
           name: 'name1'
         },
         {
           id: 2,
           name: 'name2'
         }
       ]
    }
  }
  render() {
    return (
      <div>
        <ListComponent  items={this.state.items} />
      </div>
    )
  }
}
import React from 'react'
export default function ListComponent(props) {
    const items = props.items;
    const listItems = items.map((item) =>
        <div>{item.name}</div>
    );
    return (
        <div>
            {listItems}
        </div>
    )
}
```

Esimerkkikoodi 2. React-funktiokomponenttiesimerkki. App-komponentin state items laitetaan propsina ListComponentille.Lista käydään ES6-map-funktiossa läpi.

### 3.2.3 Lifecycle-metodit

- render. Render on ainut pakollinen Lifecycle-metodi Reactin luokkakomponentissa. Palauttaa React-elementtejä, jotka yleensä luodaan JSX:llä.

- componentDidMount. Kutsutaan heti, kun komponentti on asennettu (asetettu puuhun). Voidaan käyttää esimerkiksi, kun haetaan dataa palvelimelta HTTP-pyynnöllä GET.

- componentWillUnmount. Kutsutaan välittömästi ennen komponentin irrottamista ja tuhoamista.

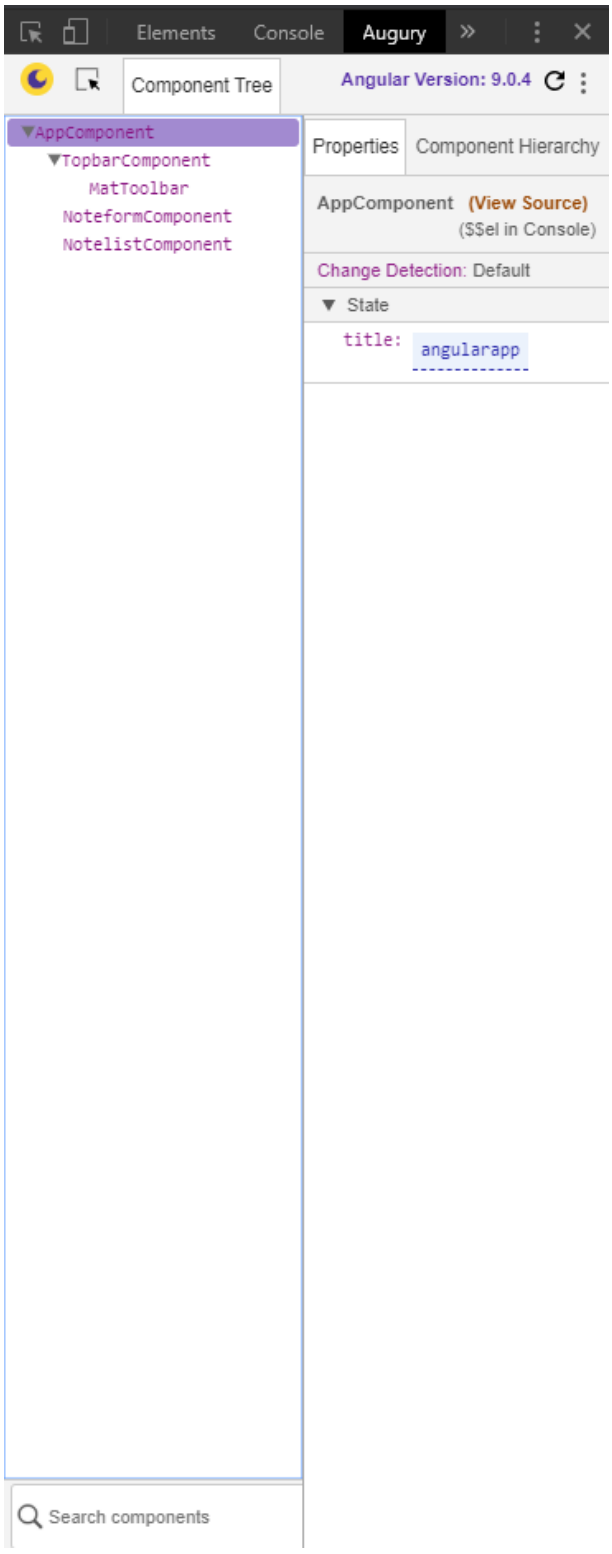
### 3.2.4 Hooks

Hooksit lisättiin Reactiin Reactin versiossa 16.8. Hookseilla voidaan käyttää Reactin state-ominaisuutta ja joitakin muita ominaisuuksia ilman luokkakomponentteja. Ennen hookseja statea voitiin käyttää vain luokkakomponenteissa. Hooksit mahdollistavat siis staten käytön funktiokomponenteissa. Hooksit ovat täysin valinnainen Reactin ominaisuus eikä luokkakomponenttien poistamista Reactista olla suunnittelemassa. [7.]

## 4 Angular

Angular on Googlen kehittämä sovelluskehys. Angular on kokonaan uusi sovelluskehys samoilta tekijöiltä, jotka tekivät AngularJS:n. Nimi Angular ilman JS:ää viittaa Angular versio 2 tai uudempiin ja AngularJs versioon 1. Angular on all-in-one ratkaisu sovelluksen frontendiin. Angular sisältää myös oman HttpClient-kirjaston HTTP-pyyntöjen tekemi-

seen. Reactiin ja Vue.js pitää käyttää jotakin erillistä kirjastoa HTTP-pyyntöjen tekemiseen kuten Axios tai Fetch api. Angularin uusin versio on tällä hetkellä 8. Angulariin on saatavilla Angular Material UI-komponenttikirjasto. Angularille on myös on Augury Dev-Tools (kuva 2). Angularille on myös Angular CLI-työkalu, jolla voi esimerkiksi luoda uusia Angular-projekteja, käynnistää kehityspalvelimen ja luo uusia komponentteja ja palveluja.




Kuva 2. Angularin Augury DevTools.

## 4.1 Angularin historia

Ennen AngularJS:n julkaisua Misko Hevery kehitti sivuprojektia. Projektin oli tarkoitus tehdä websovellusten kehittämisestä helpompaa. Tällä sivuprojektille tuli myöhemmin nimeksi AngularJS. Angular, koska HTML:ssä käytetään < ja > -merkkejä. AngularJS julkaistiin avoimen lähdekoodin projektina vuonna 2010. AngularJS on erillinen sovelluiskehys Angularista, eivätkä sovelluskehykset ole toistensa kanssa yhteensopivia. AngularJS on Angularin versio 1 ja AngularJS:n versio on aina 1.x.[8.]

## 4.2 Angularin ominaisuudet

### 4.2.1 Komponentit

Angularissa kuten myös Reactissa ja Vuessa sovelluksen käyttöliittymä rakennetaan komponenteista. Angularin komponentti koostuu kolmesta tiedostosta. Ensimmäinen on HTML-tiedosto, toinen Typescript ja kolmas CSS-tyylitiedosto.

```
import { Component, OnInit } from '@angular/core';
import { NoteService } from "../note.service";
@Component({
  selector: 'app-notelist',
  templateUrl: './notelist.component.html',
  styleUrls: ['./notelist.component.css']
})
export class NotelistComponent implements OnInit {
  notes;
  constructor(private noteService: NoteService) { }
  ngOnInit(): void {
    this.notes = this.noteService.getNotes();
  }
  deleteNote(id): void {
    this.noteService.deleteNote(id);
  }
}
```

Esimerkkikoodi 3. Angular-komponentin TypesScript-tiedosto.

```
<div *ngFor="let note of notes">
  <mat-card>
    Title: {{ note.title }} <br />
    Content: {{ note.content }} <br />
    <button (click)="deleteNote(notes.indexOf(note))" mat-raised-but-
ton color="warn">Delete</button>
  </mat-card>
</div>
```

Esimerkkikoodi 4. Angular-komponentin HTML-tiedosto.

### 4.2.2 Servicet ja Dependency Injection(DI)

Angularissa service on instanssi luokasta, jota voi käyttää missä tahansa sovelluksessa käyttämällä Angularin Dependency Injectionia. Servicet ovat paikka, jossa voidaan jakaa dataa sovelluksen eri osiin Angular-sovelluksessa. Angularin Dependency Injectionilla voidaan injectoida service komponentille @Injectable-decoratoria käyttämällä.

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class ItemService {
  items = [];

  createItem(item) {
    this.items.push(item);
  }

  getItems() {
    return this.items;
  }

  deleteItems() {
    this.items = [];
    return this.items;
  }

  constructor() { }

}
```

Esimerkkikoodi 5. Angular service -esimerkki.

### 4.2.3 Template-syntaksi

Angularin template-syntaksi laajentaa HTML:n ja JavaScriptin ominaisuuksia. Angularilla on käytössä rakenteellisia direktiivejä. Rakenteelliset direktiivit alkavat ”*”-merkillä. Rakenteelliset direktiivit muuttavat DOM:ia. Yleensä lisäämällä, poistamalla tai muuttamalla elementtejä, joissa rakenteellisia direktiivejä käytetään.Template-syntaksiin kuuluu myös interpolointi kahdella kaarisululla, ominaisuuksien sitominen hakasuluilla ja tapahtumasidonta tavallisilla suluilla. [9.]

Rakenteellisia direktiivejä:

- *ngFor. For-silmukka. `<div *ngFor="let item of items"> <h3> {{ item.name }} </h3> </div>`
- *ngIf. If-lause. `<div *ngIf="ehto">Sisältö renderöidään, jos ehto on totta. </div>`

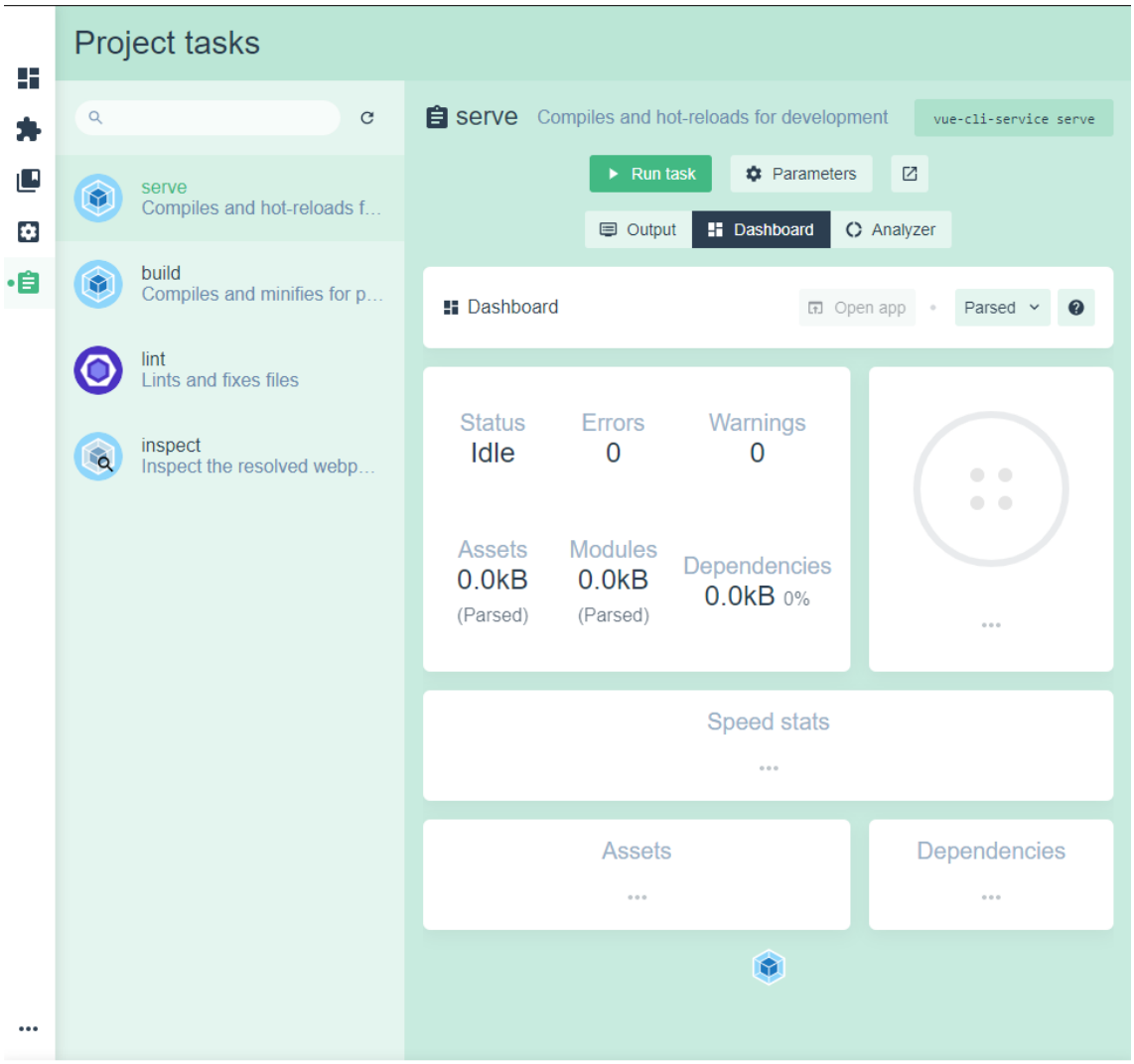
### 4.2.4 TypeScript

TypeScript on Microsoftin kehittämä superset JavaScriptille. Kaikki JavaScript on myös validia TypeScriptiä. TypeScript kääntyy JavaScriptiksi. TypeScriptissä pystytään määrittelemään muuttujille tyypit. TypeScriptissä myös voidaan käyttää luokkia, olioita ja rajapintoja samankaltaisesti kuin oliokielissä kuten Java ja C#.

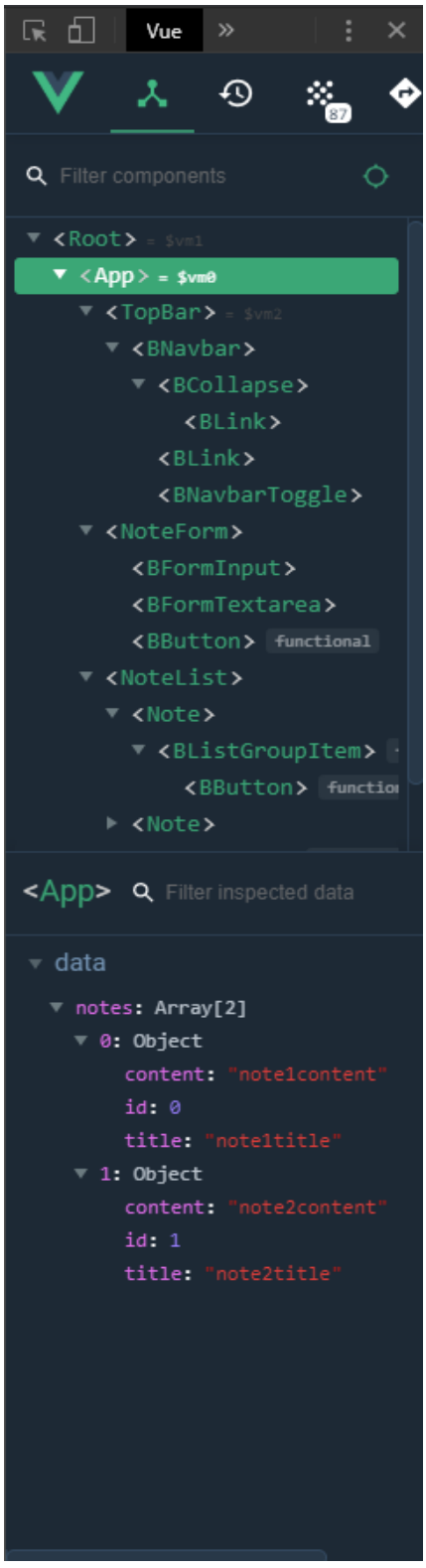
## 5 Vue.js

Vue.js on avoimen lähdekoodin Javascript-sovelluskehys käyttöliittymien rakentamiseen. Vue.js:llä kehitetyt sovellukset ovat yhden sivun sovelluksia (SPA=Single Page Application). Vue.js käyttää virtual DOM:ia. Vue.js työkaluja on esimerkiksi Vue CLI (Command Line Interface) ja Vue Devtools. Vue Router on Vuen virallinen reititin (router).

Vue.js:lle on tilanhallintaan tarkoitettu kirjasto Vuex. Vuex on samankaltainen kuin usein Reactin kanssa käytetty kirjasto Redux. Vue-projekti on mahdollista tehdä käyttämällä Vue CLI -työkalua. ”Vue create projektinnimi” luo projektin. Vuelle on myös olemassa graafinen työkalu Vue ui, jolla voi luoda projekteja (kuva 3) ja käynnistää kehityspalvelimen. Vue.js on myös mahdollista lisätä html-sivun headiin script-tagilla. Vuelle on myös oma DevTools (kuva 4).




Kuva 3. Graafinen Vue UI -työkalu.




Kuva 4. Vuen DevTools

## 5.1 Vuen historia

Vue.js:n on alun perin luonut Evan You. Nyt Vuea ylläpitävät hän sekä monet muut. Evan You loi Vuen sen jälkeen, kun hän oli työskennellyt Googlella käyttäen AngularJS:ää. You piti Angularin databindingista ja siitä, että DOM:iin ei tarvinnut koskea itse.

Angular toi myös paljon ylimääräisiä käsitteitä, jotka pakottivat rakentamaan koodin Angularin haluamalla tavalla. Se tuntui liian raskalta käyttötapaukselle, joka hänellä oli tuolloin.

You ajatteli, että voisi ottaa Angularista ominaisuudet, joista hän piti ja luoda jotakin kevyttä ilman kaikkia Angularin ominaisuuksia. [10.]

## 5.2 Vuen ominaisuudet

### 5.2.1 Vue-instanssi

Kaikki Vue-sovellukset alkavat luomalla Vue-instanssi Vue-funktiolla.[11.]

```
import Vue from 'vue'
import App from './App.vue'

Vue.config.productionTip = false

new Vue({
  render: h => h(App),
}).$mount('#app')
```

Esimerkkikoodi 6. Vue-CLI:llä luodussa projektissa new Vue luo vue-instanssin ja renderöi komponentin App.

### 5.2.2 Template-syntaksi

Vue käyttää HTML-pohjaista template-syntaksia. [12.]

Vuella voidaan käyttää direktiivejä:

- v-text. Päivittää elementitn textContentin. `<span v-text="msg"></span>` tai `<span>{{msg}}</span>`
- v-html. Päivittää elementin innerHTML:ää.
- v-show. Muuttaa css:n display-propertya trueksi tai falseksi.
- v-if. Renderöi elementin ehdollisesti. `<div v-if="true"> Elementti näkyy </div>`
- v-else. Edellesessä elementissä pitää olle v-if tai v-else-if.
- v-else-if. Edellesessä elementissä pitää olle v-if tai v-else-if.
- v-for. For-silmukka. `<div v-for="item in items"> {{ item.text }} </div>`
- v-on. Lisää event listenerin elementtiin. `<button v-on:click="doSomething"></button>`
- v-bind. Mahdollistaa atribuutin tai komponentin propin sitomisen elementtiin.
- v-model. Mahdollistaa kahden suuntaisen tiedon sitomisen input-elementin tai komponentin välillä.
- v-pre. Ohittaa kääntämisen elementille ja elementin sisäisille elementeille. `<span v-pre>{{ this will not be compiled }}</span>`

- v-cloak. Direktiivi pysyy elementissä siihen asti kun Vue-instanssi lopettaa kääntymisen. Yhdistettynä CSS-sääntöihin, kuten [v-cloak] { display: none }, tätä direktiiviä voidaan käyttää piilottamaan kääntymättömien kahden kaarisulun sidonta, kunnes Vue-instanssi on valmis. `<div v-cloak>{{ message }}</div>.`

- v-once. Renderöi elementin ja komponentin vain kerran.[13.]

Vue.js-direktiiveille on myös lyhenteitä.

### 5.2.3 Komponentit

Vue.js:ssä kuten myös Reactissa ja Angularissa käyttöliittymä rakennetaan komponenteista. Vue.js:n komponentit koostuvat kolmesta osasta, jotka ovat Template, script ja style. Templateen menee HTML. Toiminnallisuus menee scriptiin ja tyylit styleen. Vue-komponentti on yhdessä.vue-tiedostossa. Komponentissa siis on kolme tagia: <template>, <script> ja <style>. Style-tagin sisälle pitää laittaa scoped-sana, mikäli haluaa css:n vaikuttavan vain samaan komponenttiin. Komponentit ovat uudelleen käytettäviä Vue-instansseja, joilla on nimi. [14.]

```
<template>
  <div id="app">
    <TopBar />
    <NoteForm v-on:create-note="createNote" />
    <NoteList v-bind:notes="notes" v-on:del-note="deleteNote" />
  </div>
</template>
<script>
import TopBar from "./components/TopBar";
import NoteList from "./components/NoteList";
import NoteForm from "./components/NoteForm";

export default {
  name: "App",
  components: {
    TopBar, NoteList, NoteForm
  },
  data() {
    return {
      notes: [
        {
          title: "First Note Title", content: "First Note Content"
        },
        {
          title: "Second Note Title", content: "Second Note Content"
        }
      ]
    };
  },
  methods: {
    deleteNote(i) {
      this.notes.splice(i, 1);
    },
    createNote(newNote) {
      this.notes = [...this.notes, newNote];
    }
  }
};
</script>
<style>
</style>
```

Esimerkkikoodi 7. Vue-komponentti

### 5.2.4 Lomakkeet

Vuessa voidaan käyttää v-model-direktiiviä, jolla luodaan kahdensuuntainen tiedon sitominen lomakkeisiin.

```
<input v-model="msg">

<p>Message is: {{ msg }}</p>
```

## 6 Sovellus Reactilla, Angularilla ja Vuella ja sovellusten vertailu

Teen sovelluksena muistiosovelluksen Angularilla, Vuella ja Reactilla. Sovelluksen olisi tarkoitus olla lähes samanlaiset ja vertailla sovellusten ja Reactin, Angularin ja Vuen eroja. Sovellus tulee olemaan mahdollisimman yksinkertainen muistiosovellus. Käytän kovakoodattua dataa sovelluksissa, koska keskityn tässä työssä vain frontendiin.

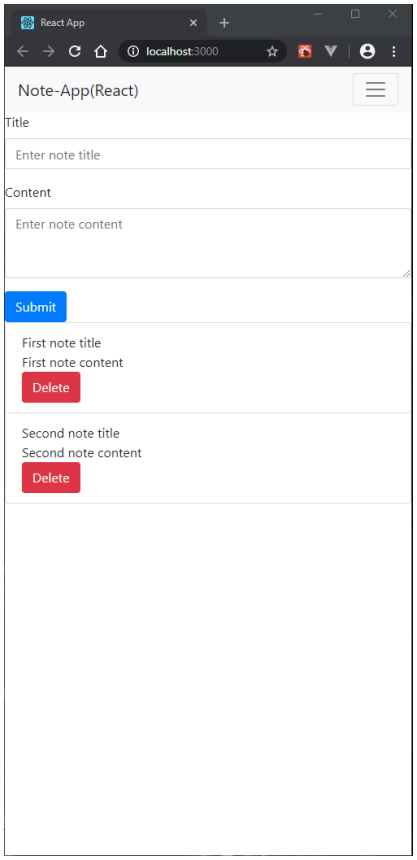
Sovelluksessa on CRUD (Create, Read, Update ja Delete) toimenpiteet, joilla muistiinpanoja voi lisätä, lukea, päivittää ja poistaa. Koska näissä sovelluksissa on vain frontend mukana niin data on vain UI:ssa ja poistuu, kun käyttäjä lataa sivun uudelleen.

Isommassa full stack -sovelluksessa olisi myös backend mukana ja frontend tekisi sinne http-pyyntöjä. Backendissä olisi myös tietokanta, johon tieto tallentuisi, ja se olisi useimmiten REST API ja ohjelmointikielenä voitu käyttää vaikka Javaa, Pythonia, PHP, C# tai JavaScriptiä (Node.js).

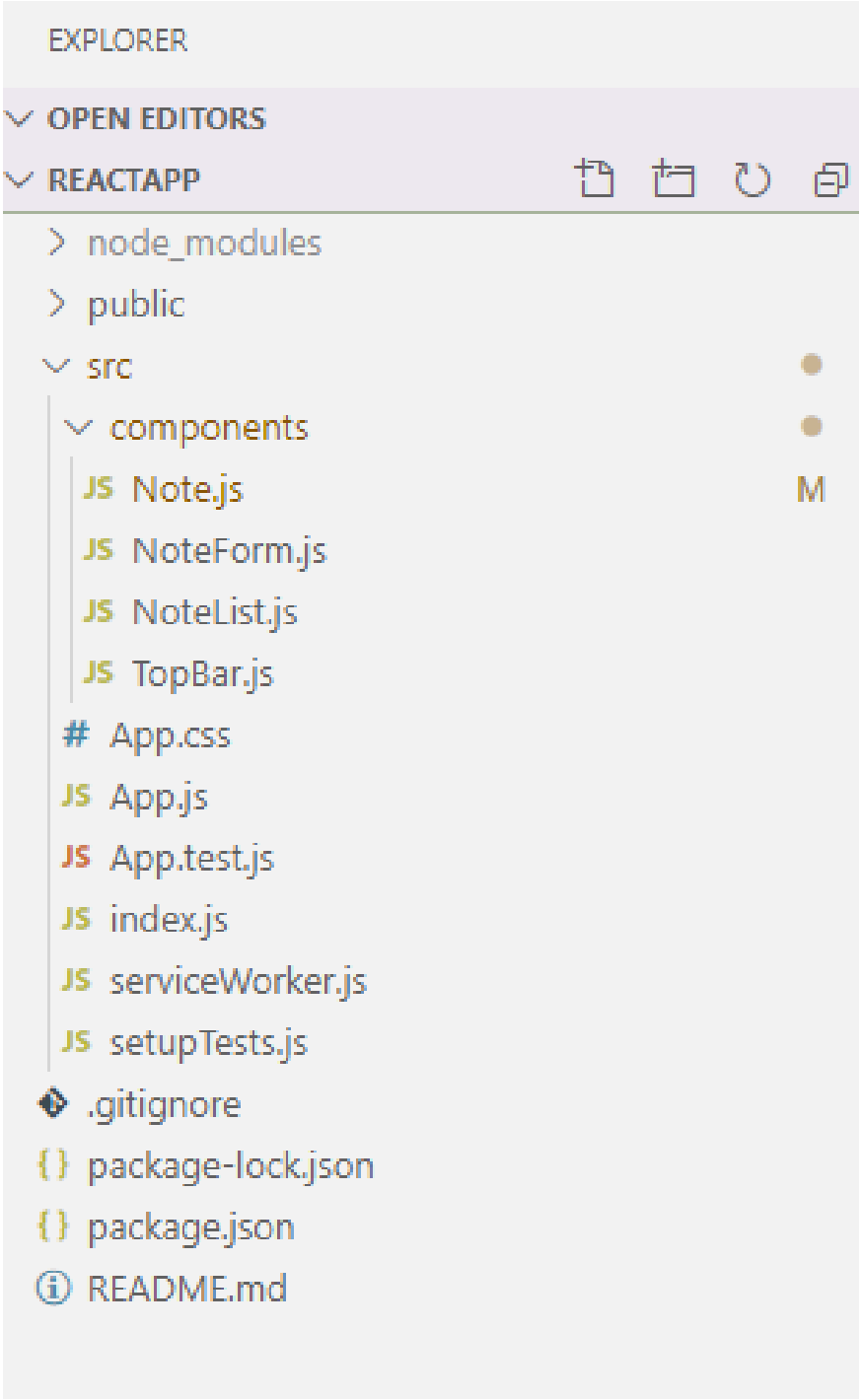
Otin myös Bootstrapin mukaan, jotta sovellus olisi responsiivinen ja toimisi hyvin työpöydän lisäksi myös mobiilissa. Käytin Bootstrappia Reactin ja Vue.js:n sovelluksissa ja niihin oli saatavilla React Bootstrap- ja BootstrapVue-kirjastot, jotka tekivät Bootstrapin käytön Reactilla ja Vue.js:llä helpommaksi. Angularilla on oma Bootstrapin kaltainen kirjasto Angular Material, joka sopii hyvin käytettäväksi Angularin kanssa. Käytin Angularilla siksi Angular Materialia. Koodieditorina käytän Microsoftin Visual Studio Codea.

## 6.1 React-sovellus

React-sovelluksessa käytin Reactin lisäksi Bootstrappia ja react-bootstrappia (kuva 5). React-sovelluksessa on viisi komponenttia (kuva 6). Ensimmäisenä App-komponentti, jonka sisällä muut komponentit ovat. Yläpalkkikomponentti TopBar on tehty käyttäen Bootstrapin NavBaria. Uusia muistiinpanoja lisätään NoteForm-komponentilla. NoteList-komponentti on Bootstrapin ListGroup, jonka sisällä yksittäinen muistiinpano on yksi Note-komponentti. Muistiinpanot ovat App-komponentin statessa taulukko, jonka sisällä on olioita. Muistiinpanot viedään propsina NoteList- ja Note-komponenteille.



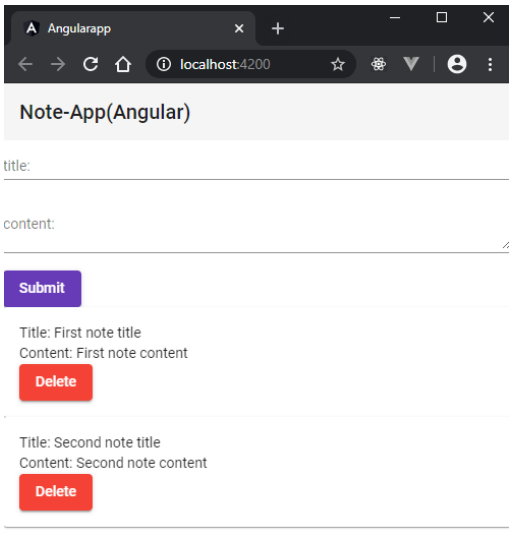

Kuva 5. Kuva React-sovelluksesta.



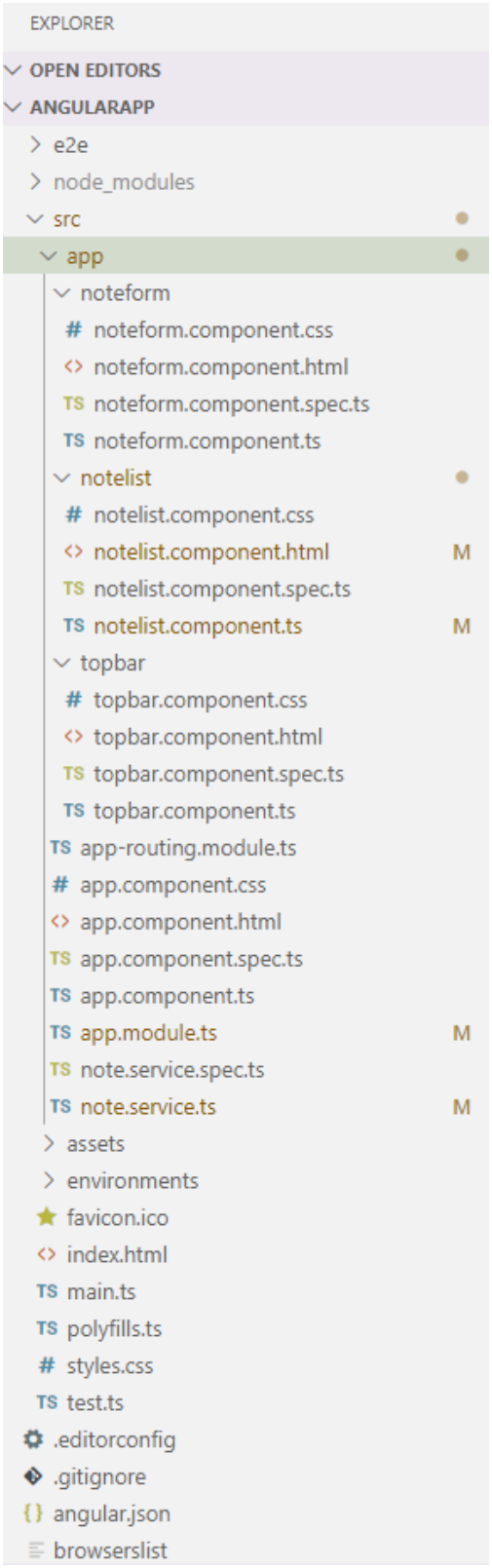

Kuva 6. React-projektin tiedostot

## 6.2 Angular-sovellus

Käytin Angular-sovelluksessa Angular Material UI -komponenttikirjastoa (kuva 7). Sovelluksessa on neljä komponenttia (kuva 8). Ensimmäinen komponentti on app-komponentti, jonka sisällä muut komponentit ovat. Yläpalkkikomponentti top-bar on tehty käyttäen Angular Materialin mat-toolbaria. Lomakekomponentin nimi on noteform ja noteform-komponentista voidaan lisätä uusia muistiinpanoja. Muistiinpanot tulevat näkyviin notelist-komponenttiin. Angular-sovelluksessa dataa jaetaan komponenttien välillä käyttäen serviceä.



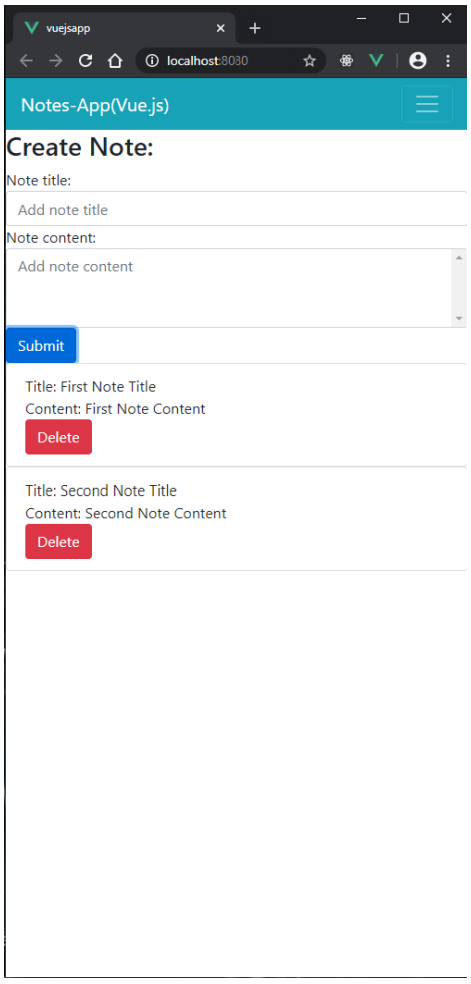

Kuva 7. Kuva Angular-sovelluksesta.



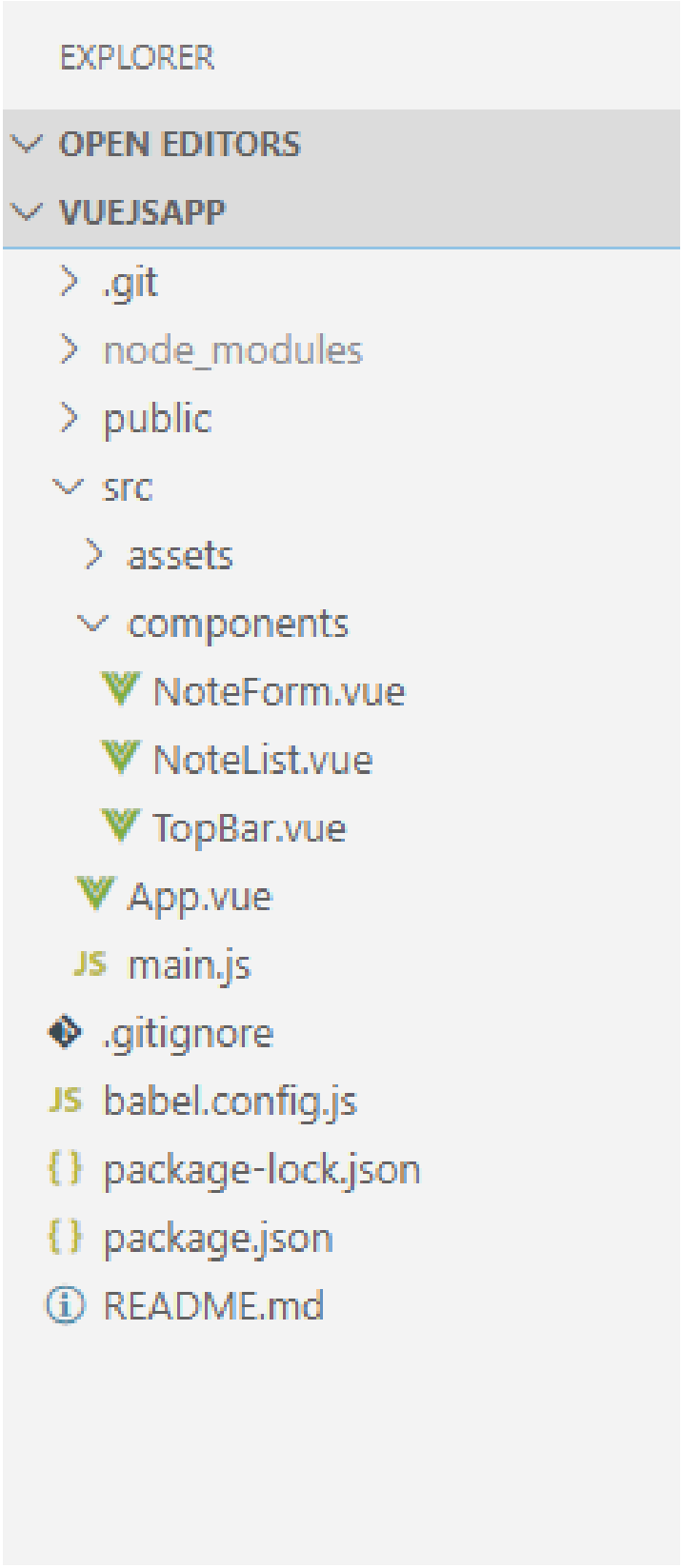

Kuva 8. Angular-projektin tiedostot.

## 6.3 Vue.js-sovellus

Vue.js-sovelluksessa käytin lisäksi Bootstrappia ja Bootstrap-vuea (kuva 9). Sovelluksessa on yhteensä neljä komponenttia (kuva 10). Ensimmäinen komponentti on App-komponentti, jonka sisällä muut komponentit ovat. Yläpalkin komponentti TopBar on tehty käyttäen Bootstrapin NavBaria. Muistiinpanoja voidaan lisätä NoteForm-komponentilla. Kun muistiinpanoja on lisätty, niin ne näkyvät NoteList-komponentissa. Muistiinpanot viedään NoteList-komponentille propsina käyttämällä v-bind-direktiiviä. App-komponetissa on data-funktion sisällä taulukko notes. App-komponentissa on methods-kohdassa metodit createNote ja deleteNote.




Kuva 9. Kuva Vue.js-sovelluksesta




Kuva 10. Vue.js-projektin tiedostot.

```
<div v-for="note in notes" v-bind:key="notes.indexOf(note)">
      <b-list-group>
        <b-list-group-item>
          Title: {{note.title}}
          <br />
          Content: {{note.content}}
          <br />
          <b-button v-on:click="$emit('del-note',notes.in-
dexOf(note))" variant="danger">Delete</b-button>
        </b-list-group-item>
      </b-list-group>
    </div>
```

Esimerkkikoodi 8. Vue.js:n v-for direktiivillä muistiinpanot käydään for-silmukassa läpi. Kun käyteään v-for-direktiiviä pitää elementeille antaa oma key käyttämällä v-bind:key.

```
data() {
    return {
      title: "",
      content: ""
    };
  }
```

Esimerkkikoodi 9. NoteForm-komponentin data-metodissa title,content ja id.

```
createNote(e) {
      e.preventDefault();
      const newNote = {
        title: this.title,
        content: this.content
      };
      this.$emit("create-note", newNote);
      this.title = "";
      this.content = "";
    }
```

Esimerkkikoodi 10. Metodi createNote, jolla lisätään muistiinpanoja notes-taulukkoon ja tyhjennetään title- ja content-lomakkeet.

## 6.4 Sovellusten ja tekniikoiden vertailu

Suurimpia eroja Reactissa, Vuessa ja Angularissa on, että komponentit Reactissa ja Vuessa ovat yhdessä tiedostossa ja Angularissa kolmessa. React ja Vue myös käyttävät Virtual DOM:ia ja Angular ei. Angular myös käyttää TypeScriptiä ja React ja Vue JavaScriptiä. On kuitenkin myös mahdollista käyttää TypeScriptiä Reactin ja Vuen kanssa, mutta oletuksena Reactissa ja Vuessa on käytössä JavaScript. Angular on myös koko ratkaisu sovelluksen frontendiin. Angular vaatii enemmän aikaa opetella kuin React ja Vue.js.

Angularilla ja Vuella voidaan myös käyttää direktiivejä ja Reactilla ei. React on näistä kolmesta tällä hetkellä suosituin ja Reactille on eniten työpaikkoja. Työpaikkoja toiseksi eniten tällä hetkellä olisi Angularille. Vue.js on vielä niin uusi, että työpaikkoja ei vielä paljoa löydy, mutta Vuen suosio on koko ajan nousussa. Google Trendsin mukaan React on suosituin, Angular toisena ja Vue.js kolmantena (kuva 11). Npm Trendsin mukaan React on suosituin, Vue.js toisena ja Angular kolmantena (kuva 12).

Reactilla ja Vuella tila pidetään yleensä ylimmässä komponentissa ja alemmat komponentit vastaanottavat tiedon propseina. Reactilla ja Vuella usein tarvitsee ottaa käyttöön tilanhallintakrjasto, kun sovellus kasvaa isoksi. Reactilla käytetään useimmiten Reduxia ja Vuella Vuexia. Angularilla on myös mahdollista käyttää tilanhallintakirjastoa, mutta niitä harvoin käytetään Angularin kanssa. Angularissa datan jakaminen komponenttien kesken onnistuu serviceillä. Mielestäni, koska Angularissa on käytössä servicet, joilla voidaan jakaa dataa komponenttien kesken ei tilanhallintakirjastojen käytölle Angularissa pitäisi kovin usein olla tarvetta. Kaikille kolmelle on olemassa selaimessa käytettävät DevToolsit.

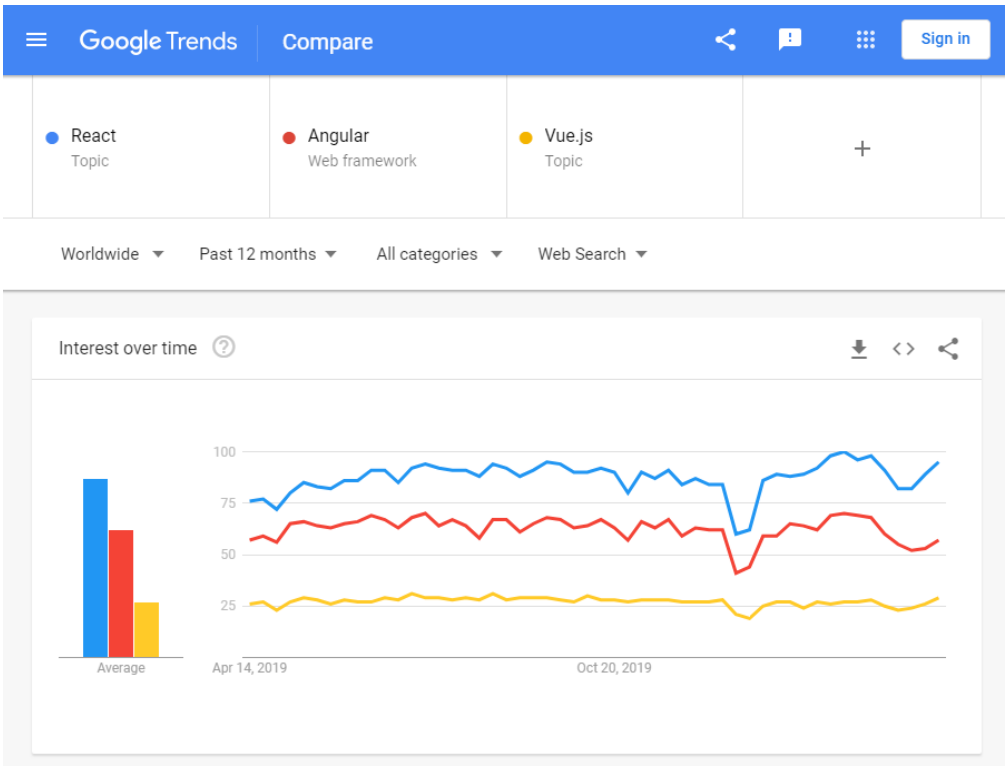
Kaikilla kolmella kehitetään SPA-sovelluksia eli sovellus yhdessä div-tagissa index.html-sivulla. SPA-sovelluksen kehittäminen on huomattavasti hankalampaa vain JavaScriptiä käyttäen eli ilman sovelluskehyksiä ja kirjastoja.

Taulukko 1. Reactin, Angularin ja Vuen ominaisuuksien vertailua taulukossa.

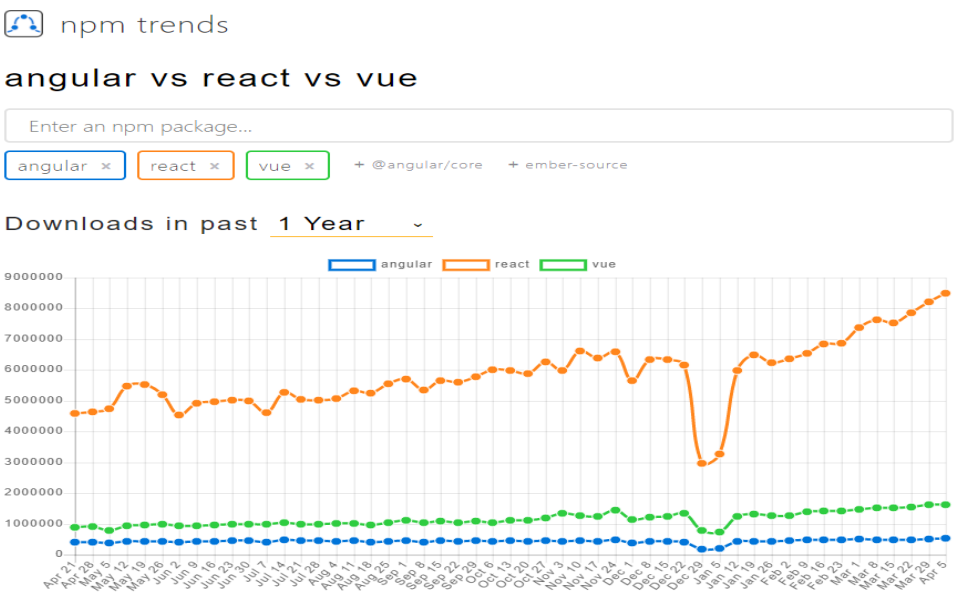
| | **React** | **Angular** | **Vue.js** |
|---|---|---|---|
| Kirjasto | Kyllä | Ei | Ei |
| Sovelluskehys | Ei | Kyllä | Kyllä |
| SPA(Single Page Application) | Kyllä | Kyllä | Kyllä |
| PWA(Progressive Web Application) | On mahdollista kehittää PWA-sovelluksia. | On mahdollista kehittää PWA-sovelluksia. | On mahdollista kehittää PWA-sovelluksia. |
| Komponentit | Kyllä | Kyllä | Kyllä |
| JSX | Kyllä | Ei | Ei oletuksena(on mahdollista ottaa käyttöön. JSX:n käyttö Vuessa ei kovin yleistä.) |

| | | | |
|---|---|---|---|
| Virtual DOM | Kyllä | Ei | Kyllä |
| Oma Http-pyyntöihin tarkoitettu kirjasto | Ei (Voidaan käyttää esim. Fetch API tai Axios) | Kyllä (Angular sisältää HttpClientin) | Ei (Voidaan käyttää esim. Fetch API tai Axios) |
| Direktiivit | Ei | Kyllä | Kyllä |
| Typescript | Oletuksena JavaScript(TypeScriptin voi ottaa myös käyttöön, mutta käyttö valinnaista) | Kyllä | Oletuksena JavaScript(TypeScriptin voi ottaa myös käyttöön, mutta käyttö valinnaista) |
| Komponentit yhdessä tiedostossa | Kyllä | Ei(Komponentit ovat kolmessa tiedostossa) | Kyllä |
| Servicet | Ei | Kyllä | Ei |
| Suuren yrityksen kehittämä | Kyllä(Facebook) | Kyllä(Google) | Ei(Evan You) |

| | | | |
|---|---|---|---|
| Oma käyttöliittymäkomponenttikirjasto | Ei(Voidaan käyttää esim. Bootstrap.) | Kyllä(Angular Material) | Ei(Voidaan käyttää esim. Bootstrap.) |
| Data binding | Kyllä. Yhdensuuntainen (one way data-binding). | Kyllä. Kahdensuuntainen (two way data-binding). | Kyllä. Kahdensuuntainen (two way data-binding). |
| Tilanhallinta | Voidaan käyttää esim. Redux tai Reactin oma Context API. | Sisältää servicet niin erilliselle tilanhallintakirjastolle ei pitäisi olla niin paljoa tarvetta. | Vuex |
| CLI | Kyllä. Create React App | Kyllä. Angular CLI. | Kyllä. Vue CLI. |
| DevTools | Kyllä | Ei virallisia DevToolseja. Voidaan käyttää esim. Augury | Kyllä |




Kuva 11. Reactin, Angularin ja Vue.js:n suosion vertailua Google Trendsissä vuoden ajalta maailmanlaajuisesti[https://trends.google.com/trends/].




Kuva 12. Reactin, Angularin ja Vue.js:n vertailua Npm Trendsissä. Npm-pakettien lataukset vuoden ajalta. [https://www.npmtrends.com/angular-vs-react-vs-vue].

# 7 Yhteenveto

Yhteenvetona voisi sanoa, että kaikilla kolmella: Reactilla, Angularilla ja Vue.js:llä voi rakentaa moderneja SPA-sovelluksia. On oma mielipide, mitä näistä kolmesta pitää suosikkina. Eroina on kuitenkin, että komponenttit Reactissa ja Vuessa ovat yhdessä ja Angularissa kolmessa tiedostossa. Angular myös käyttää TypeScriptiä. On myös mahdollista käyttää TypeScriptiä Reactin ja Vue.js:n kanssa. React ja Angular ovat myös isojen yritysten kehittämät ja Vue.js ei.

Sovelluksia, jotka tein olivat todella yksinkertaisia ja niitä voisi paljon jatkokehittää. Tällä hetkellä tieto on kovakoodattua eikä tallennu minnekään. Jos haluaisi, että tiedot tallentuvat voisi sovellukseen tehdä palvelinpuolen ohjelmoinnin, vaikka Node.js:llä ja käyttää vaikka MongoDB-tietokantaa. HTTP-pyyntöjä voisi Reactilla ja Vuella tehdä käyttäen Fetch API:a tai Axios-kirjastoa. Angular sisältää oman HTTP-client-moduulin, jolla voi tehdä HTTP-pyyntöjä.

Mielestäni React ja Vue.js ovat mielestäni näistä kolmesta helpoimmat oppia. Vue.js ehkä hieman helpompi kuin React. Angular on vaikein oppia, koska Angular on koko ratkaisu sovelluksen frontendiin ja sisältää paljon ominaisuuksia. Virtual DOM on myös käytössä Reactissa ja Vuessa. Tästä syystä Reactin ja Vue.js:n suorituskyky on hieman Angularia edellä.

## Lähteet


1 MDN web docs. 2019. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Luettu 17.2.2020.

2 The History of React.js on a Timeline. RisingStack. <https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/>. Luettu 17.2.2020.

3 Introducing JSX. 2020. Verkkoaineisto. Facebook. <https://reactjs.org/docs/introducing-jsx.html>. Luettu 31.01.2020.

4 Rendering Elements. 2020. Verkkoaineisto. Facebook. <https://reactjs.org/docs/rendering-elements.html>. Luettu 31.01.2020.

5 Components and Props. 2020. Verkkoaineisto. Facebook. <https://reactjs.org/docs/components-and-props.html>. Luettu 31.01.2020.

6 State and Lifecycle. 2020. Verkkoaineisto. Facebook. <https://reactjs.org/docs/state-and-lifecycle.html>. Luettu 31.01.2020.

7 Introducing Hooks. 2020. Verkkoaineisto. Facebook. <https://reactjs.org/docs/hooks-intro.html>. Luettu 7.5.2020.

8 The History of Angular. Medium. 2019. Verkkoaineisto. Medium. <https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7>. Luettu 17.2.2020.

9 Getting Started with Angular. 2020. https://angular.io/start. Verkkoaineisto. Luettu 11.4.2020.

10 Between the Wires: An interview with Vue.js creator Evan You. Verkkoaineisto. FreeCodeCamp. https://www.freecodecamp.org/news/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4/. Luettu 20.4.2020.

11 The Vue Instance. 2020. Verkkoaineisto. Evan You. <https://vuejs.org/v2/guide/instance.html>. Luettu 02.02.2020.

12 Template Syntax. 2020. Verkkoaineisto. Evan You. <https://vuejs.org/v2/guide/syntax.html>. Luettu 02.02.2020.

13 Directives. 2020. Verkkoaineisto. Evan You. <https://vuejs.org/v2/api/#Directives>. Luettu 02.02.2020.

14 Components Basics. 2020. Verkkoaineisto. Evan You. <https://vuejs.org/v2/guide/components.html>. Luettu 02.02.2020.