Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut Teknologiat

2017

Juho Keski-Rahkonen

MONIALUSTAINEN VERKKOSOVELLUS ANGULAR 2 OHJELMAKEHYSTÄ KÄYTTÄEN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ AMK | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka

2017 | 20

Juho Keski-Rahkonen

MONIALUSTAINEN VERKKOSOVELLUS ANGULAR 2 OHJELMAKEHYSTÄ KÄYTTÄEN

Nykyään on useita erilaisia ohjelmistokehyksiä, ja niiden käyttö eroaa toisistaan aika paljon. Suosituimmat näistä ovat Reactjs ja Angular. Angular-ohjelmistokehyksestä on hiljattain tullut versio 2, jonka mukana tuli paljon uudistuksia. Ennen julkaisua, Angular 2 -testiversiot olivat erittäin vaikeakäyttöisiä.

Opinnäytetyön tarkoituksena oli selvittää, kuinka helppoa on toteuttaa Angular 2 -sovellus joka myöhemmin muutetaan Ionic 2 -sovellukseksi. Näin luotiin sekä nettisivusto että mobiilisovellus samalla koodipohjalla.

Näiden sovellusten ohella tuotettiin palvelinsovellus Node.js-alustalla. Opinnäytetyössä käytettiin Amazon Web Services -pilvipalveluita verkkosovelluksen ja palvelinsovelluksen käyttöönottoon.

Kehittämisen aikana huomattiin, että Angular 2 -ohjelmistokehyksen mukana tulevalla Angular-cli-komentorivityökalulla pystytään luomaan Angular 2 -sovellus helposti ja nopeasti, verrattuna aikaisempiin Angular-versioihin, jolloin ei ollut kyseistä komentorivityökalua. Samalla komentorivityökalulla voitiin myös luoda esimerkiksi lisää komponentteja sovellukseen. Angular-cli-komentorivityökalulla voitiin myös pystyttää testauspalvelin omalle tietokoneelle, jotta pystyttiin testaamaan ja kehittämään sovellusta selaimessa.

Ionic 2 -ohjelmistokehyksen mukana tulee oma Ionic-cli-komentorivityökalu, joka nopeuttaa Ionic 2 -sovelluksen kehitystä. Kehityksessä kuitenkin huomattiin, että siitä puuttuu joitain Angular-cli-työkalun ominaisuuksia. Näin katsottiin paremmaksi tehdä ensin Angular 2 -sovelluksessa jokin ominaisuus ja sitten kopioida se Ionic 2 -sovellukseen.

Opinnäytetyön tarkoituksena oli tehdä sekä Android- että iOs-alustoille sovellukset. Mutta kävi ilmi että iOs-alusta vaatii kehittäjältä oman Mac-tietokoneen sekä iPhone-puhelimen. Ilman näitä iOs-alustalle ei voitu kehittää mobiilisovellusta.

Android-sovellusta kehittäessä huomattiin, että Android Studio käyttää eri SDK-versioita kuin Cordova. Tämä loi ongelmia mobiilisovelluksen testauksessa. Tästä syystä on parempi tehdä sovellus valmiiksi käyttämällä vain selainta, ja vasta lopuksi testata sovellus puhelimessa.

ASIASANAT:

Angular 2, Ionic 2, Verkkosovellus, Mobiilisovellus, Node.js

BACHELOR´S | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2017 | 20

Juho Keski-Rahkonen

MULTIPLATFORM WEB APPLICATION USING ANGULAR 2

This thesis documents the making of a web application that utilizes the Angular 2 framework, and how such an application could be changed into a mobile application that utilizes the Ionic 2 framework.

The purpose of this thesis was to determine how easy it is to develop an Angular 2 application which will then be migrated into an Ionic 2 application. In this way, we could create both a web application and a mobile application with a single codebase.

Along with these applications, a server application was created using the Node.js platform. For deployment, Amazon's Web Services were used to host web and server applications.

During development, it was noticed that the Angular 2 framework comes with a command line tool, that can be used to create an Angular 2 application with ease. The same command line tool was also used to generate more components for the application, and to host a local development server, which allowed testing and development of the application within the browser.

It emerged that the Ionic 2 framework had a similar command line tool, however it was not as robust as Angular 2 had. Thus, it was determined that it was better to first develop a feature on Angular 2, and then copy and paste it to Ionic 2.

After the development work had been completed, it was time for testing. It was planned to have two mobile applications, for Android and iOs. However, the documentation revealed that development for iOs requires the developer to own a Mac computer from Apple, and an iPhone. Without these, the iOs application could not be developed.

While making the Android application, the Android Studio apparently had version differences with the Cordova framework. This made testing the Android application very hard. It was concluded that it is better to first develop the application using the browser and only after development is finished, test it on a mobile phone.

KEYWORDS:

Angular 2, Ionic 2, Web application, Mobile application, Node.js

SISÄLTÖ

KUVAT

KUVIOT

KÄYTETYT LYHENTEET

JS	Javascript on selain-puolella käytetty ohjelmointikieli
TS	Typescript on Javascriptin superset, eli Javasript sisältyy Typescriptiin.

1 JOHDANTO

Nykyään nettisivustoja on monenlaisia. Enimmät sivustoista on tehty vanhanaikaisella PHP-kielellä, ja siitä johdettujen ohjelmistokehyksien, esimerkiksi Wordpressin, avulla. Tätä tekniikkaa käytetään vielä nykyäänkin nettisivustojen kehittämiseen. Syynä tähän on helppous ja saatavuus. Suurin osa kehittäjistä on niin sanotusti kasvanut PHP:n parissa, ja nämä kehittäjät hyödyntävät taitojaan kehittäen nettisivustoja.

Vaikkakin nettisivujen tekeminen vanhoilla ja varmoilla työkaluilla on helppoa, sitä ei kuitenkaan suositella nykypäivänä. Nykypäivän työkalujen ohella PHP:ta kutsutaan ”kaksipäiseksi vasaraksi”, ja syystäkin. Laatu on nykypäivänä tärkeämmässä roolissa myös nettisivujen kohdalla, verrattuna vuoteen 1994, kun PHP keksittiin (Atwood 2010).

Lähivuosina esiin on noussut uusia ohjelmistokehyksiä, jotka käyttävät vaihtoehtoisia menetelmiä nettisivustojen kehityksessä. Googlen kehittämä Angular-ohjelmistokehys on yksi suosituimmista.

Tässä raportissa käydään läpi, kuinka Angular 2 -ohjelmistokehyksellä tuotetaan verkkosovellus ja kuinka verkkosovellus muutetaan yhteensopivaksi Ionic 2 -ohjelmistokehykselle. Tällöin saadaan sekä nettisivusto että mobiilisovellus samalla koodipohjalla.

Sovelluksen ohella on myös kehitetty palvelinohjelmisto käyttäen Googlen kehittämää Node.js-alustaa. Tämä käsittelee sovelluksessa olevien tietojen käsittelyn ja tallentamisen. Myös käyttäjätilien hallinta ja todentaminen käsitellään tällä pavelimella.

Projektin kautta halutaan nähdä, kuinka helppoa on muuttaa Angular 2 -ohjelmakehyksellä tehty verkkosovellus mobiilisovellukseksi Ionic 2 -ohjelmistokehyksen avulla. Myös mobiilisovelluksen suorituskykyä halutaan testata projektin kautta.

2 TEKNISEN ALAN TAUSTA

Projektissa kehitettiin verkkosovellus käyttäen pääosin Javascript-pohjaisia teknologioita. Verkkosovelluksiin kuuluu aina HTML ja CSS. Projektissa kehitettiin myös mobiilisovellus Android-alustalle.

HTML eli “HyperText Markup Language” on pienin rakennuspala nettisivussa. HTML määrittää sisällön nettisivuille. HTML:n ohella käytetään myös CSS- ja Javascript-teknologioita (Mozilla Developer Network 2017a).

CSS eli ”Cascading Style Sheets” määrittää miten HTML-elementit näytetään ruudulla. CSS voi sijaita omassa CSS-tiedostossa, ja se voi määrittää useiden eri sivujen HTML-elementtien tyylit. Tämä säästää paljon aikaa kehittämisessä (W3Schools 2017).

Javascript on ohjelmointikieli, jota käytetään nettisivuissa antamaan toiminnallisuutta. JS on kevyt tulkkikieli, eli koodi luetaan ja suoritetaan ajon aikana. JS:llä on oma standardi, ECMAScript. Nykyisin versio tästä standardista on ECMAScript2018, mutta uusin mitä verkkoselaimet tukevat on ECMAScript 5.1 (Mozilla Developer Network 2017b).

Typescript on Microsoftin kehittämä vapaan lähdekoodin tyypitetty superset JS:lle. TS sisältää JS:n ja myös uudistuksia ECMAScript2015:stä. Typescriptin tarkoitus on tuoda olio-ohjelmoinnin periaatteita verkkoselain kehitykseen. Tuotantovaiheessa Typescript käännetään Javascriptille. Käännettäessä kääntäjä luo tarvittavat ECMAScript2015 ominaisuudet, jotta myös selaimet pystyvät suorittamaan käännettyä koodia (Microsoft 2017).

Android on Google:n kehittämä käyttöjärjestelmä mobiililaitteille. Android perustuu Linux-käyttöjärjestelmään, ja se on suunniteltu pääosin kosketuslaitteille. Androidin lähdekoodi on vapaasti saatavilla Googlen julkaisemana. Uusin Android versio on 7.0 eli ”Nougat” (Wikipedia 2017).

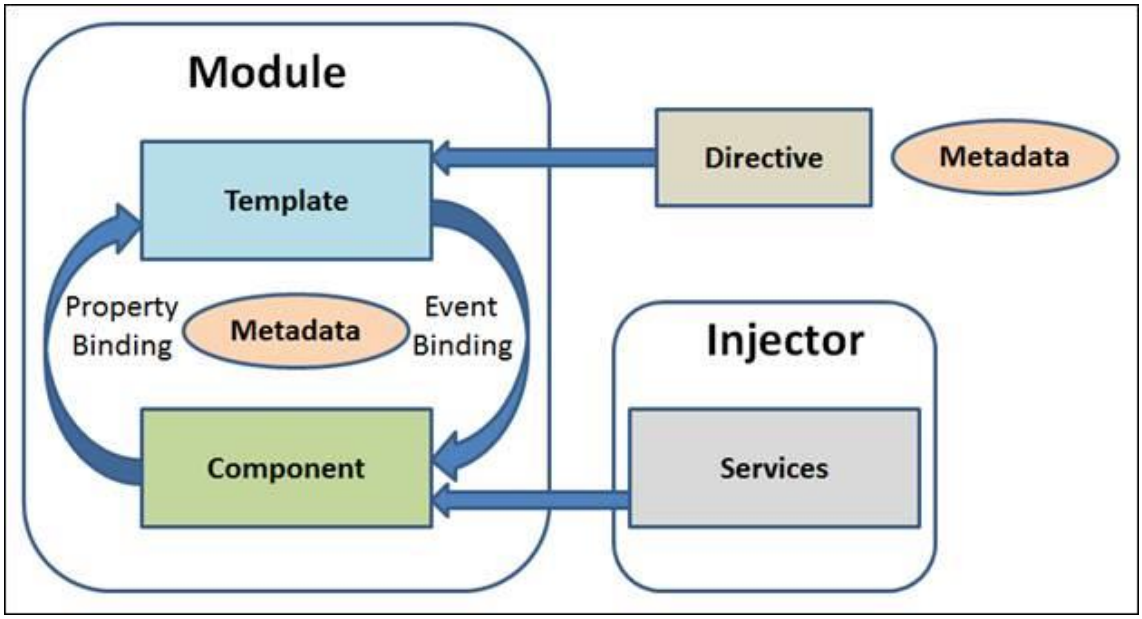
3 TEKNISEN TOTEUTUKSEN TAUSTA

Projektissa kehitettiin verkkosovellus käyttäen Angular 2 -ohjelmakehystä. Tällä ohjelmakehyksellä tehty sovellus siirrettiin Ionic 2 -ohjelmakehykselle yhteensopivaksi. Verkkosovelluksen ohella kehitettiin palvelinsovellus, joka käyttää Node.js-alustaa.

3.1 Angular 2 -ohjelmakehyksen tausta

Angular on Googlen kehittämä ohjelmistokehys yksi-sivuisille sovelluksille. Angular 2 on kehitetty TS:llä. Angularilla pystyy rakentamaan sovelluksia sekä selaimeen että mobiilille (Angular 2017).

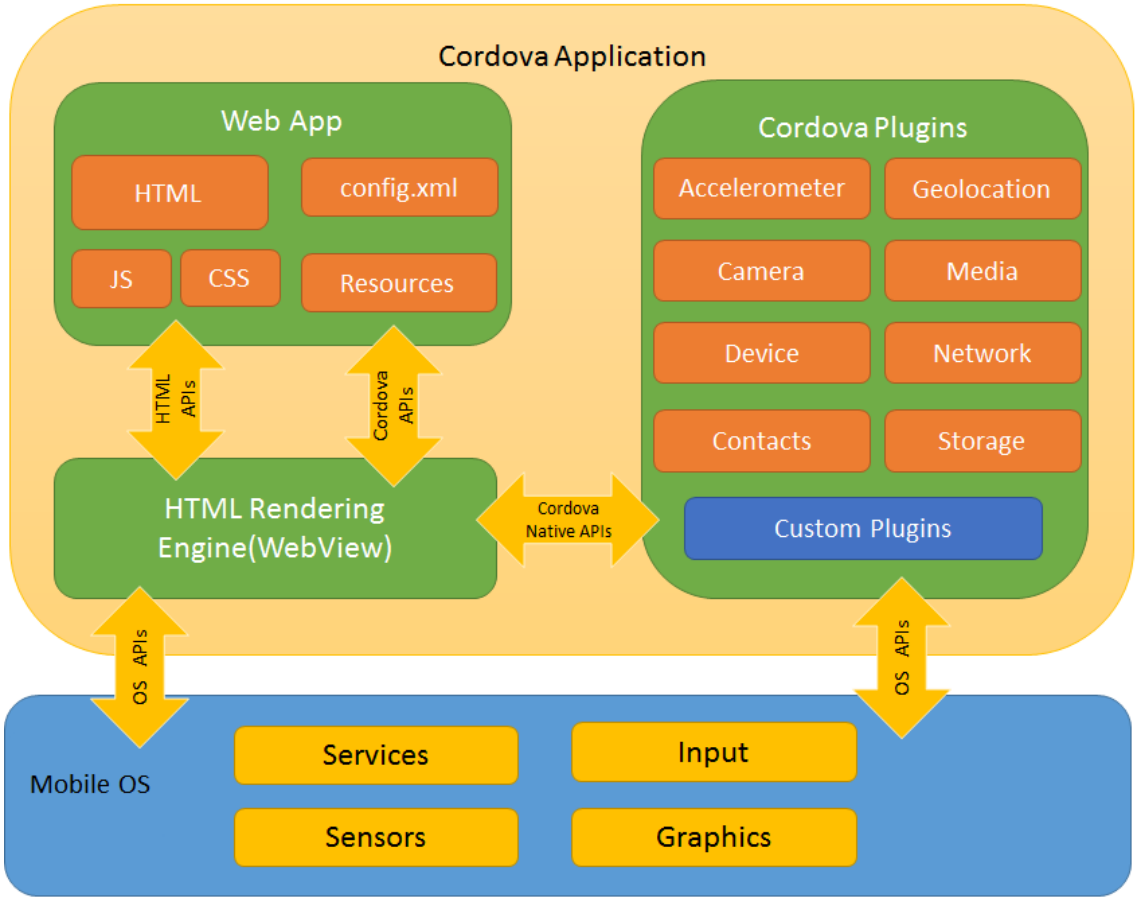
Angular 2 -sovelluksessa on 8 eri pääkomponenttia. Näistä tärkeimmät ovat moduulit, komponentit ja palvelut. Kuvio 1 näyttää yksinkertaistettuna Angular 2 -sovelluksen arkkitehtuurin. Komponentit sisältävät näkymän logiikan, ja palvelut jakavat funktioita ja/tai tietoa eri komponenteille. Moduulit ovat säiliöitä komponenteille ja palveluille (Trivedi 2016).



Kuvio 1. Angular 2 -sovelluksen arkkitehtuuri (Trivedi 2016).

3.2 Cordovan tausta

Cordova on Apachen kehittämä vapaan lähdekoodin ohjelmistokehys mobiilisovelluksille. Cordova pystyy luomaan mobiilisovelluksia käyttämällä verkkosivu-teknologioita hyödyksi. Valmis sovellus käyttää hyödyksi mobiililaitteen verkkoselainta, jossa verkkosovellus näytetään (Kuvio 2). Cordova tuo myös mobiililaitteille ominaiset ominaisuudet kuten esimerkiksi kiihtyvyysanturin verkkosovellukselle käytettäväksi (Apache 2017).



Kuvio 2. Cordova-sovelluksen arkkitehtuuri ja miten Cordova toimii mobiilikäyttöjärjestelmän kanssa (Apache 2017).

3.3 Ionic 2 -ohjelmakehyksen tausta

Ionic 2 on Googlen kehittämä ohjelmistokehys mobiilisovelluksille. Ionic käyttää Cordovaa pinnan alla, jolloin saadaan kaikille mobiilialustoille kehitettyä sovellus. Ionic myös perustuu Angulariin, jolloin Angular-sovelluksen koodia pystytään käyttämään

myös Ionic-sovelluksissa. Tästä syystä voidaan käyttää samaa lähdekoodia monessa eri sovelluksessa, ja saadaan sama sovellus monelle eri alustalle hyvin helposti (Ionic 2017).

3.4 Node.js:n tausta

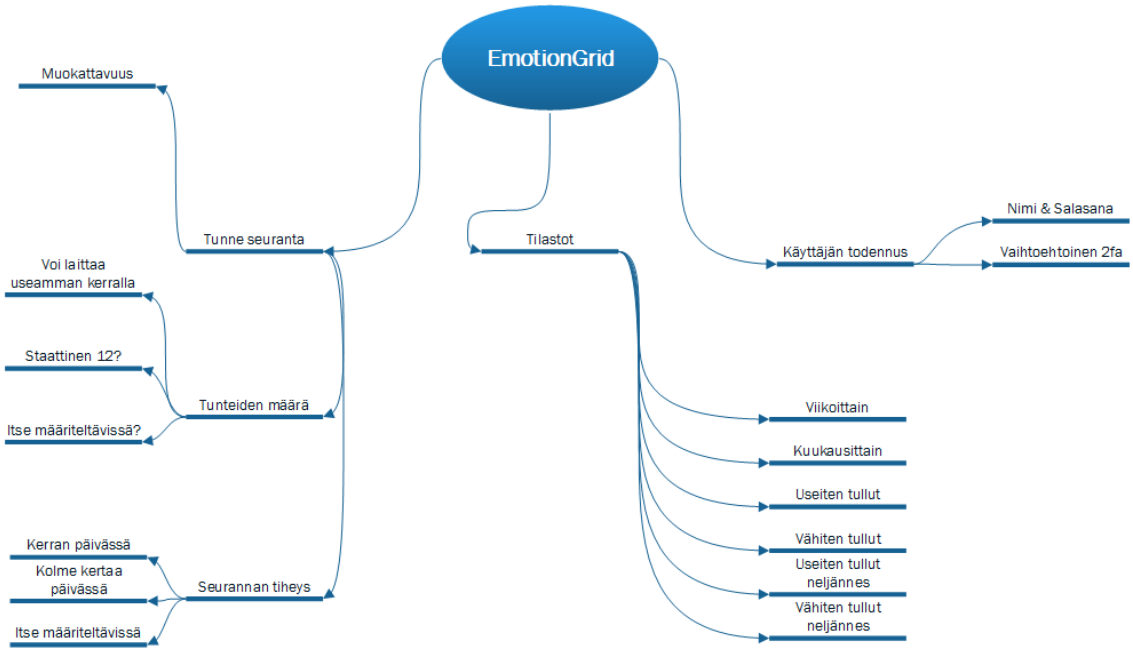
Node.js on Googlen kehittämä alusta palvelinsovelluksille. Node.js pystyy ajamaan JS-sovelluksia palvelimena tai jopa työpöytäsovelluksina. Se ei käytä tietokoneiden monilangoitusta hyödyksi, vaan ajaa kaiken yhdessä langassa. Node.js-sovellus on täysin tapahtuman-pohjainen, ja jos tapahtumia ei ole, sovellus on lepotilassa. Tämä on ideaali tapa toteuttaa palvelinsovellus (Node.js Foundation 2017).

Node.js käyttää Googlen kehittämää V8-moottoria, jolla pystytään ajamaan hyvin suorituskykyistä Javascript koodia. V8 on rakennettu C++ -kielellä, jonka ansiosta se on hyvin nopea. V8-moottoria käytetään myös Googlen Chrome-selaimessa, sekä monissa muissa sulautetuissa sovelluksissa (Google 2017).

4 SOVELLUSVAATIMUKSET

Jotta sovellusta voidaan tarkastella lähemmin, on sovellukselle määriteltävä vaatimukset. Nämä vaatimukset täytettyään sovellus on valmis. Kuitenkin täytyy muistaa, että raportissa käsiteltävä sovellus ei ole julkiseen tuotantoon tarkoitettu, vaan pääasiallisesti näyttämään, kuinka sovellus tuotetaan Angular 2 -ohjelmistokehyksellä ja kuinka Ionic 2 -ohjelmistokehystä käytetään.

Esimerkkisovellus on pääsääntöisesti tarkoitettu tunnetilojen tallentamiseen ja niiden pitkäaikaiseen tarkkailuun. Samat periaatteet täytyy toimia sekä selaimessa että mobiilisovelluksessa. Sovelluksessa täytyy pystyä myös tarkastelemaan tallennettuja tunnetiloja pitkällä aikanäkymällä. Kuviossa 3 näkyy sovelluksesta tehty miellekartta, jonka mukaan sovellus on tehty.



Kuvio 3. Miellekartta sovelluksen toiminnasta. Sovelluksella on kaksi päätoimintoa. Tunteiden tallennus ja tunteiden tilastointi.

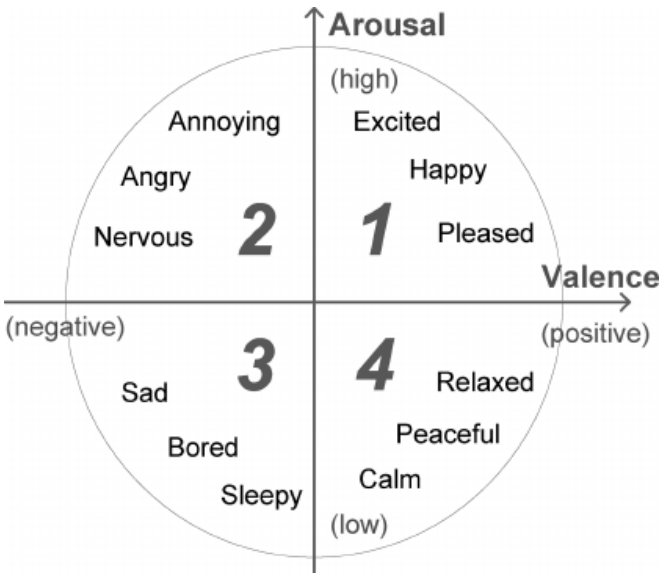
4.1 Sisäänkirjautuminen

Sovelluksessa hyödynnetään käyttäjätilejä, joten sovelluksessa täytyy olla luotettava sisäänkirjautuminen. Sen pitää tukea salausta, jotta käyttäjän salasana ei paljastuisi väliintulohyökkäyksissä. Myös uloskirjautuminen täytyy olla mahdollisimman perusteellinen, jotta selaimen välimuistiin ei jää mitään mikä paljastaisi käyttäjän tietoja.

Myös kaksi-vaiheinen todennus on mahdollinen kehittää sovellukselle. Kaksi-vaiheinen todennus hyödyntää mahdollista mobiililaitetta. Kirjautuessaan käyttäjän täytyisi todentaa kirjautuminen mobiililaitteen kautta. Tämä olisi vaihtoehtoinen tapa kirjautua, ja käyttäjä pystyy itse valitsemaan tämän kirjautumistavan.

4.2 Tunteiden tallennus

Sovelluksen päätarkoitus on tallentaa käyttäjän vallitseva tunnetila. Käyttäjä pystyy valitsemaan ennalta määritellyistä kahdestatoista tunnetilasta. Kuva 1 näyttää miten nämä tunteet eroavat toisistaan. Tämä perustuu tunteiden luokitteluun valenssin ja kiihottumistason avulla. (Stangor 2010)



Kuva 1. Valenssi-kiihottumis tunneavaruus (Yang & Chen 2012).

Tunteita pitää pystyä tallentamaan sekä yksittäin että useamman kerralla. Tallentamisen jälkeen käyttäjä näkee tilastoja meneillään olevasta kuukaudesta, ja mikäli käyttäjällä on katkeamaton sarja päiviä jolloin käyttäjä on tallentanut tunteitaan.

4.3 Tilastollisuus

Kun tunnetiloja on tallennettu pitkän aikaa, pitää käyttäjän pystyä näkemään, miltä hänestä on tuntunut pitkällä aikavälillä. Mikä on ollut käyttäjän useimmiten tullut tunnetila? Mikä tunnetila on tullut vähiten? Muun muassa näihin kysymyksiin tilastoilla pyritään vastaamaan. Tunnetiloista voidaan myös piirtää viiva kuvan 2 osoittamassa ympyrässä tunnetilojen havainnollistamiseksi. Tunnetilat pitää saada näkyviin viikko- ja kuukausinäkymissä.

4.4 Ilmoitukset

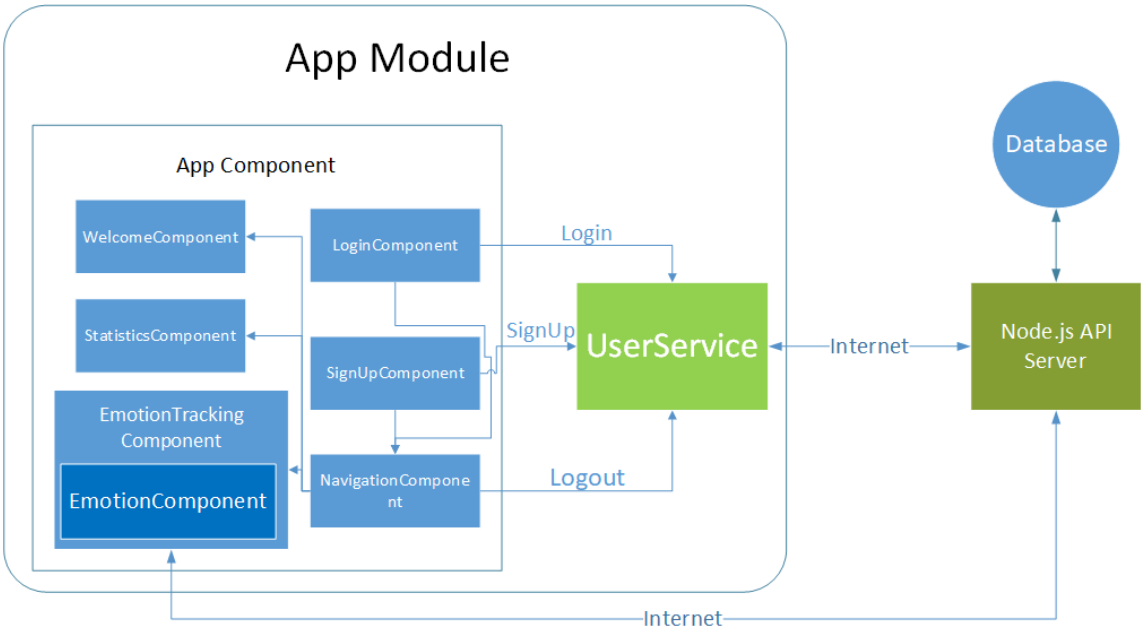
Sovelluksen pitäisi antaa ilmoituksia käyttäjälle esimerkiksi muistuttaakseen käyttäjää tallentamaan tunnetilansa. Sekä viikoittain että kuukausittain pitäisi tulla myös ilmoituksia siitä miten käyttäjällä on mennyt kulunut aika, mikäli käyttäjä näin haluaa.

Mobiililaitteessa sovellus voi antaa mobiililaitteen päänäkymään viestin muistuttaakseen käyttäjää tunnetilan tallentamisesta. Käyttäjä voi myös itse määrittää, milloin nämä muistutukset tulevat. Käyttäjä voi myös määrittää, kuinka tiheään hän haluaa tallentaa tunnetilojaan, esimerkiksi joko päivittäin tai kolme kertaa päivässä.

5 ANGULAR 2 -SOVELLUKSEN TOTEUTUS

Angular 2 -ohjelmakehyksellä kehitetty sovellus sisältää moduuleja, komponentteja ja palveluita. Sovellus voi koostua yhdestä tai useammasta moduulista. Moduuli sisältää yhden tai useamman komponentin ja tarvittaessa myös palveluita. Toteutuksessa käytettiin vain yhtä moduulia, joka sisältää useamman komponentin ja yhden palvelun.

Angular 2 -sovelluksissa on aina tietynlainen rakenne. Kuviosta 4 käy ilmi, miten toteutuksen arkkitehtuuri toimii. Sovelluksessa on kaksi yhteyspistettä palvelimen kanssa.

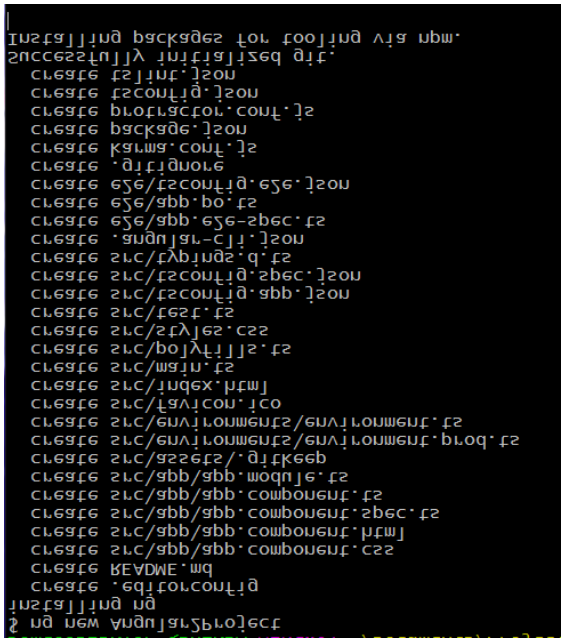


Kuvio 4. Toteutuksen arkkitehtuuri.

5.1 Angular-cli-komentorivityökalu ja sen käyttö

Angular 2 -ohjelmakehyksen ohella on kehitetty komentorivityökalu Angular 2 -sovellusten kehitystä varten. Angular-cli työkalulla voidaan helposti aloittaa projekti tai luoda esimerkiksi komponentteja. Kun projekti aloitetaan, komentorivityökalu luo kaiken valmiiksi projektin tekemiseen. Kuvassa 2 projekti on aloitettu kansioon

”Angular2Project”. Komentorivityökalu luo myös useita asetus-tiedostoja, sekä valmiit koodin testausympäristöt.



Kuva 2. Angular-cli:n new-komento suorittaa projektin luomisen annettuun kansioon.

Angular-cli pystyy myös rakentamaan projektin kehittämistä tai jopa tuotantoa varten, käyttäen hyväksi Webpack-moduulisitojaa. Tämän prosessin myötä kaikki projektin tiedostot esikäsitellään toimitusta varten.

Kehitystä varten Angular-cli pystyy luomaan kehityspalvelimen kehittäjän tietokoneelle. Kun kehittäjä muokkaa tiedostoja, kehityspalvelin päivitetään automaattisesti peilaamaan kehittäjän tekemiä muutoksia.

Komponenttien tai minkä tahansa muun elementin luonnissa Angular-cli tekee myös yksikkötestaus-tiedoston kyseisen elementin lisäksi. Tämä helpottaa yksikkötestien kirjoittamista.

5.2 Palvelut

Angular 2 -sovelluksessa palvelut ovat keskeisessä roolissa. Ne antavat komponenteille yleisiä toimintoja mitkä ei ole järkevää säilöä yksittäisen komponentin sisälle. Tuotannossa käytettiin yksittäistä palvelua järjestämään kaikki käyttäjää koskevat

toiminnot komponenteille. Esimerkiksi sisään- ja uloskirjautuminen onnistuu palvelun kautta Kuvan 3 osoittamalla tavalla.

```
export class LoginComponent {

  private username;
  private password;

  constructor(private user : UserService) { }

  private login() {
    this.user.login(this.username, this.password);
  }
}
```

Kuva 3. Sisäänkirjautuminen palvelun kautta. Käyttäjänimi ja salasana tulevat komponenttiin sidotusta HTML-palasesta.

5.3 Komponenttien sisään- ja ulosmenevä kommunikaatio

Isossa sovelluksessa, jossa on paljon komponentteja, tietoa täytyy pystyä välittämään komponentilta toiselle. Tähän löytyy monta eri menetelmää. Ne eroavat toisistaan hyvin paljon mutta lopputulos on sama.

5.3.1 Alemmalta komponentilta emokomponentille

Yksi tapa suorittaa kahden komponentin välinen kommunikaatio tapahtuu julkisen funktiokutsun kautta. Mikäli yksi komponenteista sisältyy toiseen komponenttiin, emokomponentti voi tällöin julistaa julkisen funktion alemman komponentin käytettäväksi. Alempi komponentti voi saada referenssin emokomponenttiin riippuvuusinjektion kautta, jolloin komponentti voi kutsua emokomponentin julkista funktiota lähettääkseen tietoa Kuvan 4 osoittamalla tavalla.

```
constructor(private emotionTracking : EmotionTrackingComponent) { }

toggle() {
  this.selected = !this.selected;
  this.emotionTracking.toggle(this.emotion.name);
}
```

Kuva 4. Riippuvuusinjektion kautta saadaan referenssi ylempään komponenttiin.

5.3.2 Yksisuuntainen sidos

Yleisin tapa saada tai lähettää tietoa on käyttää yksisuuntaista sidosta. Tämä pitää esittää HTML-koodin sisällä, ja sillä pystytään sitomaan mikä tahansa tapahtuma tai ominaisuus koodiin. Jos esimerkiksi halutaan sitoa nappulan painamiseen jokin funktio, se tapahtuu yksisuuntaisella sidoksella.

Tämä toimii sekä natiivien HTML-elementtien kanssa, että Angular 2 -komponenttien välillä. Mutta komponenttien välinen kommunikaatio voi tapahtua vain emokomponentin ja alemman komponentin välillä. Komponenttien välillä käytetään @Input ja @Output -koristeita, jotka sisältyvät Angular 2 -ohjelmakehykseen. Vain alemman komponentin täytyy käyttää näitä koristeita. @Input-koriste tuo suoraan emokomponentin muuntujan alemman komponentin käytettäväksi, mutta @Output-koristetta käytettäessä pitää aiheuttaa tapahtuma, johon emokomponentti sitoo funktion. Kuva 5 näyttää miltä yksisuuntaiset sidokset näyttävät HTML-koodissa. Mikäli koodin muuntuja ”selected” on tosi, niin elementti saa HTML-luokan ”selected” itseensä.

```
[class.selected]="selected"
(click)="toggle()"
```

Kuva 5. Hakasulkeet tarkoittavat ominaisuuden tai @Input-koristeeseen sitomista muuntujaan tai funktioon, ja kaarisulkeet tarkoittavat tapahtuman sitomista funktioon.

5.3.3 Palvelun kautta

Mikäli monet eri komponentit, jotka eivät sisälly mitenkään toisiinsa, tarvitsevat tiedonvälitystä, pitää käyttää Angular 2 -palvelua. Palvelun käytössä on suositeltavaa käyttää Havaittavia. Tällöin komponentin ja palvelun kontekstit eivät mene sekaisin, ja komponentti saa oman kutsutakaisin-funktion sidottua Havaittavaan.

Palvelulla voidaan esimerkiksi välittää palvelimelta tulevia tietoja moneen eri komponenttiin tarvittaessa. Palvelu myös hoitaa tarvittavat esikäsittelyt palvelimelta tulevaa tietoa varten. Täten komponenttien ei tarvitse hoitaa tiedonkäsittelyä, vaan komponentit saavat suoraan sen mitä tarvitsevat.

5.4 Käyttäjän todennus

Projektia tehdessä tärkein asia, mitä pitää miettiä, on käyttäjän todennus. Nykypäivänä mikään ei ole täysin turvallinen, mutta kompromissejä täytyy tehdä. Sivustossa kirjaudutaan käyttäjänimellä ja salasanalla, ja palvelin palauttaa käyttäjä-avaimen. Käyttäjä-avaimella todennetaan tulevat kyselyt, eikä käyttäjän tarvitse laittaa käyttäjätunnuksiaan useampaan kertaan. Uloskirjautuessa käyttäjä-avain nollataan palvelimelta.

Salasanoja lähetettäessä käytetään MD5-salausta, jotta väliintulohyökkäykset eivät toimisi. Palvelimella salasanaa vastaanotettaessa käytetään vielä SHA-256-salausta salasanaa tallennettaessa. Suositeltavaa olisi suolan käyttö, jotta saadaan lisävarmuus salasanan turvaamiseen palvelimella.

6 IONIC 2 -SOVELLUKSEN TOTEUTUS

Ionic 2 -ohjelmakehys eroaa Angular 2 -ohjelmakehyksestä hyvin vähän. Ionic 2 -ohjelmakehys lisää mobiilisovelluskehityksessä käytettyjä termejä ja rakenteita. Ionic 2 -ohjelmakehys käsittelee komponentteja pääosin sivuina, mutta myös alakomponentteja voi tehdä. Navigaatioon tarkoitettu sivupalkki on erillinen HTML-rakenne, ja se täytyy pitää mukana jokaisessa sivussa. Muuten Ionic 2 -ohjelmakehyksessä toimii kaikki samat asiat kuin Angular 2 -ohjelmakehyksessä.

Kehityksessä kohdennettiin ainoastaan Android-alustalle, sillä iOs-alusta vaatisi kehittäjältä Apple:n Mac-tietokoneen sekä iPhone-puhelimen.

6.1 Ionic-cli-komentorivityökalu ja sen käyttö

Ionic 2 -ohjelmakehyksen mukana tuleva Ionic-cli-komentorivityökalu on verrannollisesti heikko Angular-cli-komentorivityökaluun verrattuna. Ionic-cli-työkalu ei pysty luomaan mitään muuta kuin uusia sivuja. Kuitenkin Ionic-cli-työkalulla pystyy aloittamaan uuden projektin erittäin helposti. Siinä on jopa erilaisia valmiiksi tehtyjä pohjia, mistä voi valita. Projektiin valittiin pohja, jossa oli valmiiksi tehty sivuvalikko navigaatiota varten. Kuvassa 6 näkyy Ionic-cli:n automaattisesti tehty sivuvalikko HTML-muodossa. Koodiin on lisätty oma uloskirjautumisnappula.

```
<ion-menu [content]="content">
  <ion-header>
    <ion-toolbar>
      <ion-title>Menu</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <ion-list>
      <button menuClose ion-item *ngFor="let p of pages" (click)="openPage(p)">
        {{p.title}}
      </button>
      <button menuClose ion-item (click)="logout()">
        Log out
      </button>
    </ion-list>
  </ion-content>

</ion-menu>
```

Kuva 6. Ionic-cli:n luoma navigaatio-paneeli. Tähän on lisätty oma uloskirjautumis-nappula.

Uuden sovelluksen tekeminen on haastavaa Ionic-cli-työkalua käytettäessä. On täten helpompaa tuoda Angular 2 -sovellus Ionic 2 -ohjelmakehykseen ja muuttaa se yhteensopivaksi. Ionic-cli-työkalu on erittäin hyödyllinen esimerkiksi kehitysvaiheessa, kun halutaan testata sovellusta. Työkalulla pystytään luomaan samanlainen kehityspalvelin kuin Angular-cli-työkalun kanssa.

Ionic-cli-työkalu pystyy myös rakentamaan sovelluksen .apk-muotoon, jonka pystyy tuomaan esimerkiksi puhelimeen. Tällä menetelmällä saadaan tuote-valmis sovellus rakennettua, kun kehitysvaihe on ohi.

6.2 Angular 2 -sovelluksen tuonti

Ionic 2 -sovellus käyttää erilaista kansiorakennetta kuin Angular 2 -sovelluksessa. Pääsovellus on siirretty app-kansioon, ja sivut ovat omassa pages-kansiossa. Mitään muuta eroa ei juurikaan ole. Palvelut menevät providers-kansioon, mikä tosin vaikeuttaa tiedostopolkujen kirjoittamista.

Komponentteja tuodessa huomattiin, että komponenttien tyylitiedostojen tiedostomuoto ei ole yhteensopiva Ionic 2 -ohjelmakehyksen kanssa. Tyylitiedostot käyttivät .sass-tiedostomuotoa, joten se täytyi muuttaa .scss-tiedostomuotoon. Syytä tähän ei löydetty.

Muita vaikeuksia ei juurikaan ollut. Käyttäjä-palvelu toimii pelkällä kopioinnilla ja liittämisellä, mutta palvelun käyttämä http-moduuli ei ollut automaattisesti liitettynä projektissa. Tämä vaati HtppModule-moduulin lisäämistä päämoduulin tiedostoon. Tämän jälkeen käyttäjätodennus toimi ilman mitään ongelmia.

6.3 Android Studion käyttö testausvaiheessa

Jos kehitysvaiheessa halutaan testata esimerkiksi puhelimella tai puhelinsimulaattorilla, täytyy asentaa Android Studio -kehitysympäristö. Valitettavasti tämä tuo ongelmia testaamiseen. Ionic 2 -ohjelmakehyksen käyttämä Cordova vaatii eri versiot Android-SDK:sta kuin Android Studio. Tämä luo yhteensopivuusongelmia Cordovan ja Android Studio:n välille. Android Studio:lla voidaan kätevästi simuloida sovellusta sovellukselle tarkoitetussa ympäristössä, mutta Cordovalla pystytään kehittämään sovellusta selaimen avulla.

Yksi ratkaisu on pysyä Cordovalla ja kehittää pelkän selaimen avulla. Testauksessa voi käyttää .apk-tiedostomuotoa ja tuoda se testipuhelimeen testattavaksi. Tämä ei ole tietenkään paras vaihtoehto, mutta toimiva. Joissain selaimissa on kehitystyökaluissa toiminto, jolla voidaan simuloida puhelimen ruudun kokoa. Täten saadaan hyvä kehitysympäristö.

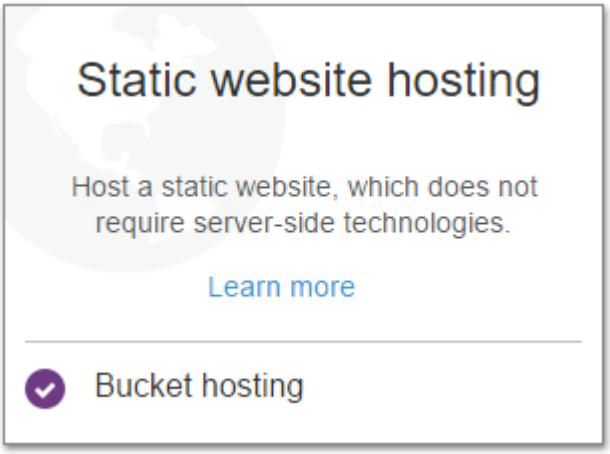
7 TESTAUS

Testauksella tässä tilanteessa tarkoitetaan miten sovellus toimisi tuotanto-vaiheessa. Verkkosovelluksen ja palvelinsovelluksen jakoon on hyödynnetty Amazonin tarjoamia pilvipalveluita. Mobiilisovelluksen testauksessa on käytetty yhteensopivaa mobiililaitetta.

7.1 Amazon Web Services -pilvipalvelu

Verkkosovellus ja Palvelinsovellus erotettiin toisistaan, laittamalla ne eri palvelimille. Täten palvelinsovellus pystyy toimimaan API-palvelimena tasapuolisesti sekä mobiilikäyttäjille että selainkäyttäjille, eikä selainkäyttäjät kuormita palvelinsovellusta niin paljon.

Verkkosovellus on jaossa S3-palvelussa staattisena verkkosivuna. Verkkopalvelin pystyy keskittymään verkkosovelluksen jakoon eikä tarvitse suorittaa erillisiä palvelinsovelluksia. Kuva 7 näyttää painikkeen Staattisen nettisivun aktivointiin S3-palvelussa.



Kuva 7. Staattisen sivun tarjoaminen ei vaadi palvelimenpuoleisia teknologioita.

Palvelinsovellus käyttää EC2-palvelua. Tämä antaa kehittäjälle käyttöön oman Linux-palvelimen, jossa pitää itse käydä asentamassa tarvittavat sovellukset. Palvelinsovellukseen käytettiin PM2-sovellusta, joka on saatavilla NPM-moduulijärjestelmästä. PM2-sovelluksella pystytään tarjoamaan Node.js-sovelluksia,

mutta sillä ei pystytä kiinnittymään porttiin 80. API-palvelimena porttia 80 ei tarvita. Mikäli näin halutaan, täytyy asentaa joko Apache- tai Nginx-palvelinohjelmisto.

Amazon Web Services -pilvipalvelu tarjoaa ilmaiskokeilua vuoden verran. Testaamiseen ei tällöin vaadita rahallista osuutta. Ilmaiskokeilu voi myös loppua jos liikennettä tulee liikaa. Tällöin myös tuotanto-vaiheessa voidaan käyttää ilmaiskokeilua hyväksi, mikäli pysytään käyttörajan alapuolella.

7.2 Mobiilisovelluksen testaus

Ainoastaan Android-alustalle tuotettiin mobiilisovellus, joten testaamiseen vaaditaan Android-mobiililaite. Testivaiheessa ei huomattu mitään eroja verkkosovellukseen. Testisovellus toimi ongelmitta ja suorituskyky oli korkea.

8 PÄÄTELMÄT

Projekti sujui ilman suurempia ongelmia. Angular 2 -sovelluksen kehittäminen on sujuvaa, ja vastaan tuleviin ongelmiin löytyy aina ratkaisu suuren yhteisön ansiosta. Ionic 2 -komentorivityökalu antoi hyvät pohjat mobiilisovelluksen tekoon, sillä automaattisesti luodun pohjan ansiosta saatiin sivupalkki navigaatiota varten.

Angular-cli-komentorivityökalu toi hyvin paljon valmiiksi tehtyä automatisointia kehitykseen. Komponenttien, palvelujen ja moduulienkin luomiseen työkalu toimi täydellisesti. Työkalun tekemä tuotantoon tarkoitettu sovellus olisi voinut olla tiheämmin pakattu, sillä työkalu tekee kuusi erillistä .js-tiedostoa eikä pakkaa niitä yhdeksi tiedostoksi. Tähän täytyy käyttää erillistä sovellusta, mikäli näin haluaa tehdä.

Mobiilisovellus oli liian pieni suorituskyvyn testaamiseen. Olisi vaadittu paljon isompi sovellus, jotta olisi saatu järkevä näkymä hybridisovelluksen suorituskykyyn. Joka tapauksessa mobiilisovellus toimi loistavasti.

Yllätyksenä oli Android Studion yhteensopivuusongelma Cordovan kanssa. Tämä vaikeutti huomattavasti mobiilisovelluksen kehitystä. Täytyi kokoajan päivittää Android SDK -versiota, jotta pystyi testaamaan ja kehittämään.

Loppujen lopuksi verkkosovelluksen kehitys on yksinkertaista Angular 2 -ohjelmakehyksellä, ja Angular 2 -sovelluksen tuominen Ionic 2 -ohjelmakehykseen ei myöskään tuota suurempia ongelmia. Tulevaisuudessa suositellaan Angular 2 -ohjelmakehyksen käyttöä sekä pieniin että isompiin verkkosovelluksiin. Myös mobiilisovelluksiin tullaan käyttämään Ionic 2 -ohjelmakehystä.

LÄHTEET

Angular 2017. Viitattu 26.4.2017 https://angular.io/

Apache 2017. Architecture overview of Cordova platform. Viitattu 20.4.2017 https://cordova.apache.org/docs/en/latest/guide/overview/index.html

Atwood, J. 2012. The PHP Singularity. Viitattu 20.4.2017 https://blog.codinghorror.com/the-php-singularity/

Google 2017. Chrome V8. Viitattu 26.4.2017 https://developers.google.com/v8/

Ionic 2017. Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular. Viitattu 26.4.2017 https://ionicframework.com/

Microsoft 2017. Typescript - JavaScript that Scales. Viitattu 26.4.2017 https://www.type-scriptlang.org/

Mozilla Developer Network 2017a. HTML. Viitattu 26.4.2017 https://developer.mozilla.org/en-US/docs/Web/HTML

Mozilla Developer Network 2017b. JavaScript. Viitattu 26.4.2017 https://developer.mozilla.org/en-US/docs/Web/JavaScript

Node.js Foundation 2017. About. Viitattu 26.4.2017 https://nodejs.org/en/about/

Stangor, C. 2010. "10.1 The Experience of Emotion" kirjassa Introduction to Psychology. v1.0 painos. Washington D.C. Flat World Knowledge, L.L.C. Sivulta 523 eteenpäin. Saatavilla myös https://ocw.mit.edu/ans7870/9/9.00SC/MIT9_00SCF11_text.pdf

Trivedi, J. 2016. Basic Architecture of Angular 2 Applications. Viitattu 26.4.2017. http://www.c-sharpcorner.com/article/basic-architecture-of-angular-2-applications/

Wikipedia 2017. Android Operating System. Viitattu 26.4.2017 https://en.wikipedia.org/wiki/Android_(operating_system)

W3Schools 2017. CSS. Viitattu 26.4.2017 https://www.w3schools.com/css/css_intro.asp

Yang, Y. & Chen, H.H. 2012, "Machine recognition of music emotion: A review", ACM Transactions on Intelligent Systems and Technology (TIST), vol. 3, no. 3, pp. 40.