jamk.fi

CSS-sovelluskehykset & Angular

Vertaileva tutkimus CSS-sovelluskehyksistä

Melina Tallqvist

Opinnäytetyö
Marraskuu, 2019
Liiketalouden ala
Tradenomi (AMK), Tietojenkäsittelyn tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Tallqvist, Melina	Opinnäytetyö, AMK	Marraskuu 2019
	Sivumäärä 43	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
CSS-sovelluskehykset ja Angular
Vertaileva tutkimus CSS-sovelluskehyksistä

Tutkinto-ohjelma
Tietojenkäsittely

Työn ohjaaja(t)
Immonen, Jarkko

Toimeksiantaja(t)

Tiivistelmä

CSS-sovelluskehyksiä on paljon erilaisia ja ne kehittyvät jatkuvasti. GitHubissa suosituin CSS-sovelluskehys on saanut 128 000 tähteä, joka on lähes 85 000 tähteä enemmän kuin seuraavaksi suosituin. Haluttiin selvittää, millaisia ominaisuuksia CSS-sovelluskehyksillä on, soveltuvatko ne kehitykseen Angularin kanssa ja milloin niiden käyttö on paikallaan. Päädyttiin vertailemaan kolmea eniten GitHubissa tähtiä saanutta CSS-sovelluskehystä ja lisäksi puhdasta CSS:ää tämän selvittämiseksi.

Vertailua varten suunniteltiin ja toteutettiin Angularilla pieni SPA-sovellus, jonka käyttöliittymä rakennettiin kaikilla valituilla CSS-sovelluskehyksillä sekä CSS:llä, kaikilla tekniikoilla mahdollisimman samanlaisiksi. Osalla valituista CSS-ohjelmistokehyksistä on olemassa myös Angularille suunniteltu, riisuttu versio, mutta niitä ei vertailun selkeyttämiseksi käytetty. Vertailua varten suunniteltiin myös kriteeristö, johon peilaten CSS ja sovelluskehykset pisteytettiin.

Tutkimuksen ohjelmistokehysten vertailusta voidaan nähdä toteutettujen käyttöliittymien kautta, että vain Semantic UI -CSS-sovelluskehys osoittautui yhteensopimattomaksi Angularin kanssa. Kriteeristön pisteytysten kautta voidaan nähdä, että Bootstrap on hieman Materializea soveltuvampi käytettäväksi Angularin kanssa, mutta ei merkittävästi.

Avainsanat (asiasanat)

CSS, HTML, Web-teknologiat, Web-sovelluskehitys

Muut tiedot (salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Tallqvist, Melina	Bachelor's thesis	Month Year
		Language of publication:
	Number of pages 43	Permission for web publication: x

Title of publication

CSS Frameworks and Angular

a comparative study of CSS Frameworks

Degree programme

Business Information Technology

Supervisor(s)

Immonen, Jarkko

Assigned by

Abstract

There is a great number of different types of CSS frameworks, which develop constantly. GitHub's top-rated CSS framework has 128 000 stars, which is almost 85 000 stars more than the second popular one. Features of CSS frameworks and their use cases were investigated by researching and comparing the top frameworks on GitHub.

A small single-page application was designed and developed with Angular. A user interface for the application was built with all chosen CSS frameworks and also with pure CSS. Some of chosen frameworks have a version just for Angular, but those versions don't have all the features of the full versions, it was decided not to use them for the development of the application. For comparison purposes, a set of criteria was also designed to score CSS and CSS frameworks.

From the comparison it can be seen that with the created user interface only Semantic UI is not compatible with Angular. Through the chosen criteria it can be seen that Bootstrap is slightly more compatible to be used with Angular than Materialize.

Keywords/tags (subjects)

CSS, HTML, Web technologies, Web development

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

Sanasto

npm	Sovelluspakettien hallinta- ja asennustyökalu
Sovelluskehys	Tuote, joka tarjoaa valmiita ratkaisuja pohjaksi sovelluskehityksen nopeuttamiseksi.
Front-end	Sovelluksen asiakkaalle näkyvä puoli
GitHub	Lähdekoodin varastointipalvelu ja suurin ohjelmistokehittäjien yhteisöpalvelu.
jQuery	JavaScript-kirjasto, joka helpottaa HTML dokumentin manipulointia, eventtien hallintaa ja animaatioiden tekoa.
MPA-sovellus	Multi-page application. Jokainen muutos tekee uuden pyynnön ja palvelimelta lähetetään koko näkymä uudelleen.
Sass	Syntactically Awesome Style Sheet on CSS:n laajennuskieli, joka on täysin yhteensopiva kaikkien CSS-versioiden kanssa.
Sass-muuttuja	Sass-kielen tarjoama tapa säilyttää tietoa, jota voidaan uudelleen käyttää tyylitiedostossa.
Grid-systeemi	Ruudukkoon perustuva layout-työkalu
Back-end	Sovelluksen serverpuoli

1 Johdanto

Asiakkaat haluavat näyttäviä, moderneja, dynaamisia web-sivustoja ja mielellään nopeasti kehitettynä. Nykyään staattiset web-sivustot eivät riitä, vaan SPA-sovellukset kasvattavat suosiotaan.

CSS-sovelluskehykset voivat mahdollistaa nopeamman kehittämisen. CSS-sovelluskehykset on valmiiksi koodattu niin, että niillä toteutetut sivustot näyttävät samalta kaikilla selaimilla, tämä on paljon aikaa säästävä ominaisuus. Nämä sovelluskehykset tarjoavat myös valmiiksi tyyliteltyjä käyttöliittymäkomponentteja, jolloin niitä ei tarvitse itse tehdä aivan alusta asti.

Tässä tutkimuksessa selvitetään, kuinka suuri hyöty on käyttää CSS-sovelluskehystä verrattuna puhtaaseen CSS:ään sekä sitä, millaisia CSS-sovelluskehyksiä on ja miten ne eroavat toisistaan. Tutkimus tulee antamaan kokemusperäistä tietoa opiskelijoille sekä aloitteleville front-end-kehittäjille.

2 Tutkimusasetelma

2.1 Opinnäytetyön tavoitteet ja rajaukset

Opinnäytetyön tavoitteena on vertailla ja tutkia CSS-sovelluskehysten toimintaa ja soveltuvuutta Angular -ohjelmistokehyksellä rakennetun SPA-sovelluksen kanssa. Opinnäytetyö rajataan niin, että tutkitaan lähemmin kolmea GitHubin mukaan suosituinta CSS-sovelluskehystä, Bootstrapia, Semantic UI:ta ja Materializea, sekä vertailukohtana myös puhdasta CSS:ää ja sen Grid-ominaisuutta.

2.2 Tutkimuskysymykset

Tutkimuskysymyksillä saadaan opinnäytetyölle runko ja ne johdetaan tutkimusongelmasta. Hyvä opinnäytetyö vastaa tutkimuskysymyksiin ja näin myös tutkimusongelma saadaan ratkaistua (Kananen, 2008, 51.).

Tämän opinnäytetyön tutkimuskysymyksiksi valikoituivat seuraavat:

- Mitä ovat CSS-sovelluskehykset?
    - Millaisia ominaisuuksia CSS-sovelluskehyksillä on?
- Milloin kannattaa valita käyttöön CSS-sovelluskehys?

Ensimmäiseen tutkimuskysymykseen vastataan teoreettisessa viitekehyksessä. Toiseen kysymykseen vastataan empiirisessä osassa.

2.3 Tutkimus- ja kehittämismenetelmät

Opinnäytetyö toteutetaan kvalitatiivisena vertailevana tutkimuksena. Kvalitatiivisella tutkimuksella pyritään ilmiön syvällisempään ymmärtämiseen ja antamaan mielekäs, syvällinen sekä rikas kuvaus ja tulkinta ilmiöstä. (Kananen, 2008, 24.) Kanasen (2008, 25) mukaan on määritelty että, jollei tutkimus ole määrällistä tutkimusta, se on kvalitatiivista tutkimusta.

Vertailevassa tutkimuksessa tarkastellaan kohteita, jotka eroavat joillakin tavoin toisistaan ollen kuitenkin samaa lajia. Ensin päätetään vertailtava kohteet ja sitten ominaisuudet, joita vertaillaan kohteiden välillä. Tutkimuksen aikana vertailtavia ominaisuuksia voidaan lisätä. (Routio, 2007.)

Tässä opinnäytetyössä vertaillaan CSS-sovelluskehyksiä ja puhdasta CSS:ää toteuttamalla Angularilla pieni SPA-sovellus, jonka ulkoasu toteutetaan kaikilla valituilla CSS-sovelluskehyksillä sekä puhtaalla CSS:llä, pyrkien kaikilla tekniikoilla mahdollisimman lähelle etukäteen suunniteltua lopputulosta. Tutkimukseen valitaan kolme GitHubin tähtien mukaan suosituinta CSS-sovelluskehystä.

3 Web-teknologiat

3.1 HTML ja CSS

HTML

HTML, Hypertext Markup Language, on verkkosivujen rakentamiseen tarkoitettu standartoitu kieli (Kurniawan, 2015, 1). Tim Barners-Lee keksi ensimmäisen version HTML:stä vuonna 1991 ja julkaisi HTML 2.0 standardin 1995. Vuonna 1997 tammikuussa julkaistiin HTML 3.2 ja saman vuoden joulukuussa HTML 4.0 W3C:n suositukseksi. (Kurniawan, 2015, 2.) Viimeisin versio, HTML5, on kehitetty W3C:n ja WHATWG:n yhteistyönä. Ensimmäinen julkinen versio siitä julkaistiin vuonna 2008 WHATWG:n toimesta. Vuonna 2012 W3C ja WHATWG eriyttivät HTML5:n kehittämisen. W3C keskittyi työstämään lopullista standardia ja WHATWG jatkuvaan kehitykseen jatkuvin parannuksin. W3C julkaisi virallisen suosituksensa HTML5:stä vuonna 2014. (A look back at HTML5 2017.)

HTML-koodi kertoo internet-selaimelle, millainen sivu rakenteeltaan on. HTML:n avulla selain osaa näyttää ja ladata kuvat, tekstit ja muut elementit oikein. HTML:ää käytetään sivuston luurangon, rakenteen luomiseen. (HTML 2018.) HTML-dokumentti koostuu kahdesta pääelementistä: *head* elementti sisältää dokumentin metadatan, *body* elementti sisältää kaiken sen, mikä halutaan näkyvän websivulla (Basic structure of an HTML document n.d.).

Kuvio 1. HTML-dokumentin rakenne (Basic structure of an HTML document n.d.)

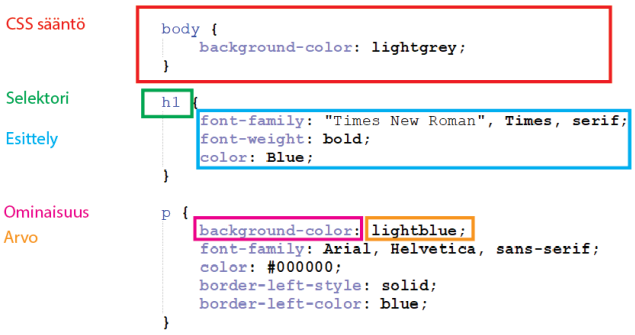
CSS

CSS, Cascading Style Sheet, on web-sivujen sommitteluun tarkoitettu tyylinmäärittelyformaatti (CSS, n.d.). Håkon Wium Lie julkaisi ensimmäisen luonnoksen CSS:tä 1994 (CSS, 2017). Vuonna 1996 Netscape julkaisi CSS:n kanssa kilpailevan JavaScript-pohjaisen JSSS-standardin, Microsoft taas ilmaisi tukevansa CSS:ää. Joulukuussa W3C valitsi CSS:n viralliseksi standardiksi. (Forsström 2016.) CSS 2 julkaistiin vuonna 1997, mutta sai W3C:n virallisen suosituksen vasta vuonna 2011. Jo vuonna 1998 aloitettiin CSS 3:n kehittäminen (Gasston, 2014, 2). Osa CSS 3:n moduuleista on saanut W3C:n suosituksen (CSS current status n.d.).

CSS:ää käytetään websivun ulkoasun määrittelyyn. Sillä muotoillaan tyylittely, mukaan lukien sommittelu myös eri kokoisille näyttöpäätteille. Erillisellä CSS-tiedostolla on mahdollista muokata useaa HTML-tiedostoa samaan aikaan. (CSS introduction n.d.)

Kuvio 2. Kuvio 1:n HTML-esimerkin sivu ilman CSS:ää

CSS-tiedostoon kootaan CSS-sääntöjä, jotka määrittelevät ulkoasun. CSS-sääntö sisältää selektorin ja esittelyn, joka taas sisältää ominaisuuden ja sen arvon. (CSS syntax, n.d.)



Kuvio 3. CSS-tiedoston rakenne ja CSS-syntaksi (CSS syntax, n.d.)

Kuvio 4. Kuvio 1:n HTML-sivu, kun siihen on lisätty Kuvio 3:n CSS-esimerkin säännöt

3.2 JavaScript ja TypeScript

JavaScript

JavaScriptin kehitti Netscapen ohjelmoija Brendan Eich. Vuonna 1995 hän loi kymmenessä päivässä scriptikielen, tuolloin nimeltään Mocha. Julkaisun myötä sen nimi vaihdettiin LiveScriptiksi. Myöhemmin nimi muutettiin jälleen, tuolloin Java oli ohjelmointikielten kuuma nimi, joten markkinointikikkana uudeksi nimeksi tuli mediasek-sikäs JavaScript. (JavaScriptin lyhyt historia n.d.)

Tärkeä askel JavaScriptin kasvussa oli siirtyminen Ecma Internationaliin. JavaScriptin virallinen nimi muuttui, sillä Oracle Corporation omistaa JavaScript tuotemerkin. Nimi

on nykyään ECMAScript, mutta JavaScript nimitys on jäänyt yleisenä nimityksenä puhekieleen. JavaScript sai ensimmäisen standardin vuonna 1997. (Punchatz 2017.)

Aluksi, ja vielä noin kymmenen vuotta sitten, JavaScriptiä käytettiin vain asiakaspuolen kielenä. Sitten tuli AJAX, web-tekniikka, joka on back-end-tapa käyttää JavaScriptiä XML-kielellä. Se mahdollisti nopeammin reagoivat web-sivustot, esimerkiksi sivusto lataa sivulle uutta sisältöä päivittämättä koko sivua. Myös JavaScript JSON-tiedostoformaatti nousi suosituimmaksi tavaksi siirtää tietoa. (Birmir 2018.) Vuonna 2009 Rayan Dahl loi Node.js:n, joka mahdollisti JavaScriptin käytön server-puolen kielenä ja johti ”JavaScript everywhere” -buumiin, nyt ohjelmoijille riittää yksi kieli sovellusten rakentamiseen (Punchatz 2017).

TypeScript

TypeScript on alun perin Microsoftin kehittämä avoin ohjelmointikielen määrittely, joka rakentuu JavaScript perustalle lisäten siihen uusia ominaisuuksia, kuten vahvan tyypityksen (Tarvainen 2016). ECMAScript 6:ssa on jo uusia ominaisuuksia. Esimerkiksi olio-orientoituneita ominaisuuksia, kuten luokat. Mutta se, että kaikki suuret selaimet saavat integroitua nämä uudet ominaisuudet, vie aikaa. TypeScript vastaa tähän ongelmaan tarjoamalla mahdollisuuden kirjottaa ohjelmaan noita ominaisuuksia, sillä se kääntää kirjoitetun koodin tavalliseksi, kaikkien selainten ymmärtämäksi JavaScript-koodiksi. (What is TypeScript and Why Should You Use it? 2017.)

3.3 CSS-sovelluskehykset

CSS-ohjelmistokehykset ovat paketteja, joiden sisällä on valmiita rakennelmia kansioita, tiedostoja ja standardisoitua koodia. Sovelluskehysten tarkoitus on tarjota valmiita, yleisiä rakenteita, joiden avulla web-kehittäjä säästää aikaa työssään. (What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design n.d.) Internettiä selataan paljon pienillä näyttöpäätteillä, vuonna 2018 verkkosivustojen vierai-

luista 58 % tehtiin mobiililaitteilla (Enge, 2019). Tällä hetkellä responsiivisuus on verkkosivuston tärkeimpiä ominaisuuksia. Muun muassa se on helppoa ja nopeaa toteuttaa CSS-ohjelmistokehyksen avulla. (What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design n.d.)

Responsiivinen web-suunnittelu tarkoittaa sivuston sovittamista usealle eri näyttökoolle. Responsiivisesti suunniteltu websivusto mukautuu automaattisesti näytön koon mukaan. Responsiivinen sivusto toteutetaan käyttämällä mukautuvaa ruudukkoa (fluid grid) ja elementtien kokoa ei määritetä pikseleinä, vaan suhteessa toisiin elementteihin, kuten prosentteina. Mobiililaitteet tuovat eteen erityisesti huomionarvoiseksi miettiä kuvien koon ja klikattavat elementit. Mobiililaitteilla selatessa helposti ongelmaksi muodostuu latausnopeus. Kuvien on hyvä olla riittävän pieniä ja grafiikoiden maltillisia, joilloin sivuston lataus on nopeampaa. Mobiililaitteissa on usein kosketusnäytöt, jolloin klikkaillaan hiiren sijaan sormea käyttäen. Jos klikattavat elementit ovat pieniä, ei sormella ole niihin yhtä helppoa osua, kuin hiireä käyttäen. (Pilon, 2018)

CSS-Sovelluskehys sisältää tavallisesti HTML-, CSS- ja JavaScript-tiedostoja, jotka pitävät yhtenäisen muotoilun ja typografian kaikilla sovellusalueilla. Yleensä CSS-sovelluskehykset tarjoavat helpon koodin ylläpidon ja responsiivisuuden lisäksi valmiiksi tyyliteltyjä komponentteja, kuten napit, valikot ja lomakkeet, sekä joukon UI-elementtejä, kuten fontteja ja ikoneita. (Saxena, 2017.)

CSS-sovelluskehykset sopivat hyvin verkkosivustojen prototyyppien tekemiseen. Toimiva staattinen HTML-prototyyppi ei ole loppukäyttäjälle niin abstrakti kuin tulosteet tai PDF-tiedostot, jolloin testausympäristössä heidän on helpompi antaa palautetta ja pienet virheet selviävät paremmin. (Maduro, 2018)

CSS-sovelluskehyksiä on paljon. Hyvän käsityksen niiden suosittuudesta saa Githubin tähdistä. Tässä opinnäytetyössä tutkitaan lähemmin kolmea Githubin tähtien mukaan suosituinta CSS-sovelluskehystä, Bootstrapia, Semanticia ja Materializea, mutta esittelen myös useamman listan kärkipään CSS-sovelluskehyksen lyhyesti.

Taulukko 1. Githubin mukaan suosituimpia CSS-ohjelmistokehyksiä (CSS frameworks n.d.)

Sovelluskehys	Github-tähdet	Ominaisuudet
Bootstrap	128 059	Mobile first, responsiivisuus, HTML-, CSS- ja JavaScript-Sovelluskehys
Semantic UI	43 175	Käyttää lyhenteiden sijaan selkeää englanninkieltä, UI-komponenttiSovelluskehys
Materialize	34 175	Material Designiin perustuva CSS-sovelluskehys
Bulma	30 507	Flexboxiin perustuva moderni CSS-sovelluskehys
Foundation	27 696	Nopeasti toteutettavat prototyypit ja koodin tuotto
Pure CSS	19 173	Setti pieniä, responsiivisia CSS moduuleita, sopii kaikenlaisiin projekteihin
Skeleton	16 095	Erittäin yksinkertainen, mobiiliystävällinen boilerplate
Uikit	13 274	Kevyt ja modulaarinen sovelluskehys, nopeiden webkäyttöliittymien luomiseen

Bootstrap

Bootstrap on suosituin CSS- sovelluskehys. Se on tarkoitettu responsiiviseen, mobile first-projektien kehitykseen. Se tarjoaa omat Sass-muuttujat, grid-systeemin, valmiit komponentit ja jQuery-pluginit tehokkaaseen ja nopeaan kehitystyöhön. (Bootstrap

n.d.) Bootstrapin on alkujaan kehittänyt Twitterin designer ja ohjelmoija vuonna 2010. Tuolloin se tunnettiin nimellä Twitter Blueprint. Ensimmäinen julkaisu oli elokuussa 2011. Sen jälkeen Bootstrap on kirjoitettu uudelleen kolmesti: V2:n lisättiin responsiivisuutta varten vaihtoehtoinen tyylitiedosto, V3:n tuli responsiivisuus oletuksena sekä mobile first-lähtökohta, Bootstrap 4:n tärkeimmät arkkitehtuuriset muutokset olivat siirtyminen Sass:iin ja CSS:n flexboxiin. (Bootstrap documentation n.d.) Opinnäytetyön tutkimuksen toteutusosassa keskitytään Bootstrap 4:ään.

Semantic UI

Semantic UI on responsiivisesti suunniteltu UI-sovelluskehys. Semantic UI käyttää luokissaan ja nimeämiskäytänteissään lyhenteiden sijaan kuvailevaa, luonnollista englanninkieltä. Semantic UI tarjoaa uniikkeja ominaisuuksia, kuten feedi- ja kommentti UI-komponentit. Se myös jättää mahdollisuuden komponenttien kustomointiin ja tyylittelyyn käyttäjälle, valmiit tyylit ovat neutraaleja ja minimaalisia. (Gerchev 2014.)

Materialize

Materialize on Material Designiin perustuva sovelluskehys. Se on Googlen suunnittelema ja kehittämä. Se käyttää tavallista CSS:ää ja tarvitsee toimiakseen jQuery-JavaScript-kirjaston. Materialize tarjoaa perinteisten käyttöliittymäkomponenttien lisäksi paranneltuja, uusia ja trendikkäitä komponentteja, jotka kaikki noudattavat Material Design -konseptia. (What is Materialize CSS n.d.)

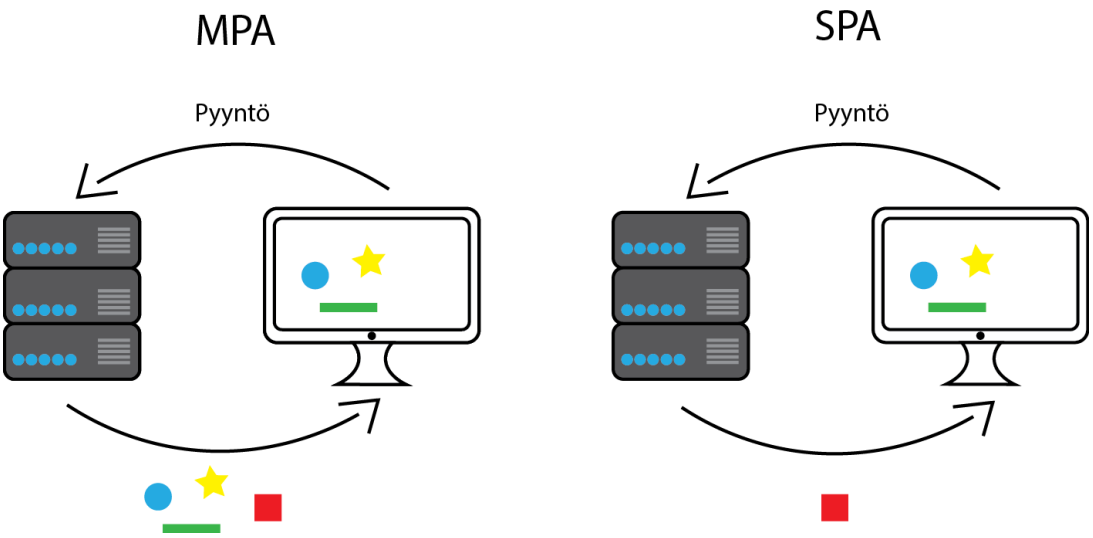
3.4 Angular

Angular on Googlen front-end-sovelluskehys sovellusten rakentamiseen HTML:llä ja TypeScriptillä. Angular soveltuu web-, mobiili- ja työpöytäsovellusten kehittämiseen. (Angular n.d.)

Angular sisältää valinnaisia- ja core-toimintoja TypeScript-kirjastoina. Kirjastot tarjoavat useita nykyaikaisia ominaisuuksia, kuten datan sitominen, reititys ja animaatiot. Angular tarjoaa myös tyyliohjeen, jossa on ohjeita, kuinka sovellus kannattaa järjestellä ja rakentaa. (Angular n.d; VanToll 2017.)

3.5 SPA

SPA (Single Page Application) on yhden sivun web-sovellus, jossa kaikki tieto ja toiminnallisuus ladataan kerralla selaimeen (Halme 2018). MPA-sovelluksissa sovellus lataa aina uuden pyynnön tullessa kaiken uudelleen serveriltä. Monesti sovelluksissa on elementtejä, kuten navigaatio, footer ja logot, jotka pysyvät samoina, vaikka sivua vaihdettaisiin. SPA-sovellus lataa uuden pyynnön tultua vain ne sivun osat, jotka vaihtuvat. Tämä tekee latauksesta nopeampaa ja käyttäjäystävällisempää, sillä haettavan informaation määrä serveriltä on huomattavasti pienempi. (Lawson 2018).



Kuvio 5. MPA- ja SPA- sovellusten toiminta havainnollistettuna.

4 Tutkimuksen toteutus

4.1 Arviointikriteerit

Arvioinnissa pisteytetään tähdin, viisi tähteä (★★★★★) on maksimipisteet.

Taulukko 2. Arviointikriteerit

Dokumentaation taso	Dokumentaation laajuus ja selkeys
Dokumentaation taso	Dokumentaation laajuus, selkeys ja informatiivisuus
Ajankäyttö	Kuinka paljon aikaa kului testisovelluksen käyttöliittymään
Responsiivisuus	Ovatko komponentit responsivisia
Yhteisö	Foorumit, sosiaalinen media, GitHub
Yhteensopivuus Angularin kanssa	Kuinka helposti yhdistettävissä Angularin kanssa

4.2 CSS-sovelluskehysten komponentit

Valittujen CSS-sovelluskehysten tutkiminen aloitettiin vertailemalla niiden dokumentaatioista kehysten tarjoamia komponentteja.

Taulukko 3. Komponenttien vertailua valittujen CSS-sovelluskehysten välillä

Komponentti	Component	Bootstrap 4	Semantic UI	Materialize
Arviointi	Rating		x	
Askeleet	Step		x	
Collapse	Collapse	x		x
Feedi	Feed		x	
Kommentit	Comments		x	
Ikonit	Icons		x	x
Karuselli	Carousel	x		x
Kortit	Cards	x	x	x
Latausmerkki	Loader		x	x
Leivänmurut	Bredcrumb	x	x	x
Lomakkeet	Forms	x	x	x
Menupalkki	Nav/Menu bar	x	x	x
Scrollspy	Scrollspy	x		x
Sivumenu	Sidebar		x	x
Sivutus	Pagination	x	x	x
Statistiikka	Statistic		x	
Toast	Toast	x		x
Tooltips	Tooltips	x		x
Upotus	Embed	x	x	
Välilehdet	Tabs	x	x	x

4.3 Testisovellus

Testisovelluksen käyttöliittymä toteutettiin kaikilla valituilla CSS-sovelluskehyksillä sekä puhtaalla CSS:llä. Sovellus itsessään toteutettiin erittäin yksinkertaisena. Esimerkiksi sovellukselle ei tehty lainkaan backendia ja esimerkkidata on kovakoodattua

frontendissa. Myös useimmat toiminnot toteutettiin vain käyttöliittymäelementteinä.

Testisovellus on teemaltaan reseptisovellus, johon tehdään navigaatio, jossa linkkien lisäksi on kirjautumislomake. Etusivulle tehdään rekisteröintilomake ja kuvagalleria. Etusivun lisäksi tehdään kaksi alasivua. Toinen alasivu on sivu reseptin lisäämistä varten, jolta löytyy lomake, johon voi kirjoittaaa reseptin sekä lisätä tiedoston. Toisella alasivulla on listattuna reseptit kahdella erilaisella näkymällä, kortti- ja taulukkonäkymänä, joiden välillä voi vaihdella vaihtopainikkeella.

CSS-versioon päädyttiin etsimään valmiita kolmannen osapuolen tekemiä Angularille tarkoitettuja moduuleita käyttöliittymään. Vaatimuksena oli, että moduulit ovat toteutettu puhtaalla CSS:llä ja madollisimman vähäisin muotoiluin. CSS-sovelluskehyksien kohdalla käytettiin vain sovelluskehyksen tarjoamia komponentteja, jotta voidaan paremmin esittää, mitä sovelluskehys oikeasti tarjoaa ja mihin sillä pystyy.

4.4 Angular-projekti

Angular-projektia varten tulee Angular CLI olla asennettuna. Se asennetaan globaalisti seuraavalla komennolla:

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

Angular CLI, Angular Command Line Interface, on työkalu Angular-projekteja varten. Se tarjoaa apuvälineitä projektin luomiseen ja kehitystyöhön, esimerkiksi testaamiseen, buildaamiseen ja käyttöönottoon.

Angular-projekti luodaan Angular CLI:a hyödyntäen seuraavalla komennolla:

```
ng new my-app
```

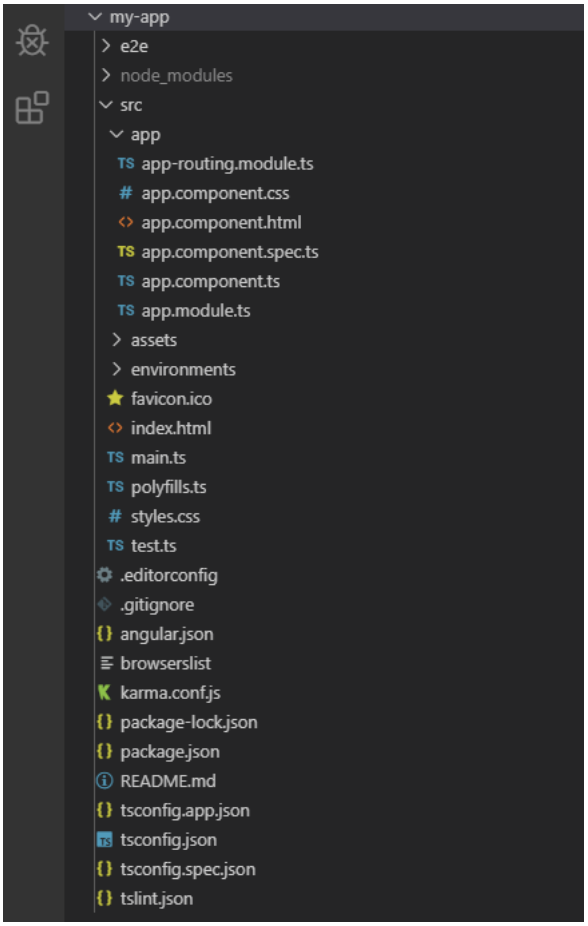
Angular CLI kysyy, lisätäänkö projektiin Angular routing. Angular router on erillinen paketti, eikä se tule Angular coren mukana.

```
PS C:\Users\melin\Test_app> ng new my-app
? Would you like to add Angular routing? (y/N)
```

Seuraavaksi Angular CLI kysyy, millä formaatilla tyylitiedostot luodaan.

```
PS C:\Users\melin\Test_app> ng new my-app
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS   [ https://sass-lang.com/documentation/syntax#scss                ]
  Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less   [ http://lesscss.org                                             ]
  Stylus [ http://stylus-lang.com                                         ]
```

Tämän jälkeen Angular-projektin runko on valmiina. Tässä projektissa otettiin Angular routing ja CSS-tyylitiedosto käyttöön.

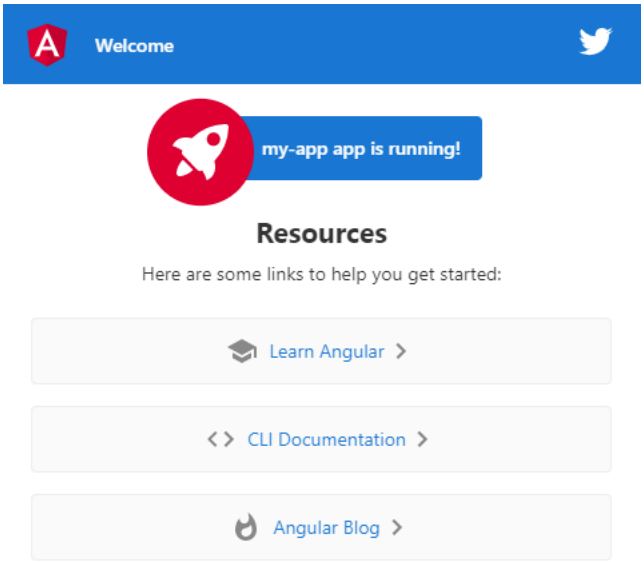


Kuvio 6. Angular-projektin kansiorakenne

Angular-projekti voidaan nyt ajaa seuraavalla komennolla:

```
ng serve
```

Nyt verkko-osoitteessa http://localhost:4200 on juuri aloitettu Angular-projekti käynnissä.



Kuvio 7. Osoitteessa http://localhost:4200 näkymä selaimessa, kun Angular-projekti on onnistuneesti luotu ja käynnistetty

Angular CLI auttaa myös uusien komponenttien luomisessa. Uuden komponentin voi luoda seuraavalla komennolla:

```
ng generate component index
```

Tämä komento luo uuden kansion *index* ja sen sisälle kaikki komponentin tarvitsemat tiedostot, *index.component.html, index.component.spec.ts, index.component.ts* ja *index.component.css.* Lisäksi se päivittää *app.module.ts* tiedoston, root moduulin, lisäten juuri luodun index-komponentin importteihin ja root moduulin määrittelyssä esiteltäväksi.

4.5 Testisovelluksen käyttöliittymän versio CSS:lla

CSS-versiota tehdessä päädyttiin käyttämään kahta kolmannen osapuolen tekemää npm:lla jaossa olevaa valmista komponenttia: ng-sidebar ja NgSimpleSlideshow. Molempiin tyylit on toteutettu CSS:llä ja sidebarin tyylit ovat todella vähäiset, joten ne

koettiin sopiviksi tähän testiprojektiin. Valmiiden komponenttien käyttäminen koettiin järkeväksi tässä projektissa, sillä suunnitelman vaatimien navigoinnin ja gallerian rakentaminen itse aivan alusta alkaen olisi ollut todella työlästä. Myöskin työelämässä käytetään valmiita komponentteja ja koodinosia, mikäli sopivia on olemassa ja saatavilla, eikä koodata kaikkea itse työajan säästämiseksi ja tuottavuuden maksimoimiseksi.

ng-sidebar ja NgSimpleSlideshow

NgSimpleSlideshow ja ng-sidebar asennettiin projektiin npm:lla terminaalissa projektikansion juuressa seuraavilla komennoilla:

```
npm install --save ng-sidebar

npm install --save ng-simple-slideshow
```

NgSimpleSlideshow ja ng-sidebar ovat Angular moduuleita, jolloin ne pitää rekisteröidä projektin root moduulin importissa, app.module.ts tiedostossa.

```
import { SidebarModule } from 'ng-sidebar';
import { SlideshowModule } from 'ng-simple-slideshow';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    SlideshowModule,
    SidebarModule.forRoot(),
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

```
export class AppModule { }
```

ng-sidebar moduuli vaatii rekisteröinnin kutsumalla forRoot() metodia. Sen avulla moduuli ja providerit saadaan ladattua globaalisti.

ng-sidebar navigaatio

Sivusta avattavaa navigointia varten ng-sidebar moduulia käyttäen tarvitsi laittaa AppComponent luokkaan ainoastaan seuraava koodinosa:

```
export class AppComponent {
  title = 'oppari-app';
  private _opened: boolean = false;
  private _toggleSidebar() {
    this._opened = !this._opened;
  }
}
```

Navigointi ja muu sisältö rakennettiin app.component.html-tiedostossa <ng-sidebar-container> tagien sisälle.

```
<ng-sidebar-container style="height: 100vh">
  <ng-sidebar class="sidebar"
      [(opened)]="_opened"
      mode="push"
      autoCollapseWidth=500>
      <a routerLink="/index" routerLinkActive="active">Home</a>
      <a routerLink="/recipes">Recipes</a>
      <a routerLink="/addrecipe">Add Recipe</a>
      <div class="login">
        <form>
          <div class="form-group">
            <label for="username">Username</label>
            <input type="text" class="form-control" id="username">
          </div>
          <div class="form-group">
            <label for="password">Password</label>
            <input type="text" class="form-control" id="password">
          </div>
          <input type="submit" value="Submit">
        </form>
      </div>
  </ng-sidebar>

  <div ng-sidebar-content class="topbar">
    <button class="navbutton" (click)="_toggleSidebar()">&#9776;</button>
    <div class="navi">
      <a routerLink="" routerLinkActive="active">Home</a>
      <a routerLink="/recipes">Recipes</a>
      <a routerLink="/addrecipe">Add Recipe</a>
      <div class="form-group">
        <label for="username" class="nav-label">Username</label>
        <input type="text" class="form-control" id="username">
      </div>
      <div class="form-group">
        <label for="password" class="nav-label">Password</label>
        <input type="text" class="form-control" id="password">
      </div>
      <input type="submit" value="Submit" class="mini-submit">
    </div>
    <router-outlet></router-outlet>
  </div>
</ng-sidebar-container>
```

Kuvio 8. app.component.html

CSS-tyyleissä määritettiin oletustyylinä sivusta avattavan navigaation tyylit.

```
:host {
    display: block;
    background-color: #ffffff;
}
:host >>> .ng-sidebar {
    background-color: white;
    font-size: 1em;
    padding-left: 0.5em;
    padding-right: 2em;
}
.topbar {
    width: 100%;
    height: 3.5em;
    background-image: linear-gradient(to right, #65b31c, #52b02f, #3fac3e, #28a74a, #00a354);
}
.navbutton {
    border: none;
    background-color: transparent;
    color: white;
    font-size: 2em;
    margin: 0.5% 0% 0% 1%;
}
.navi {
    display: none;
}
```

Kuvio 9. app.component.css

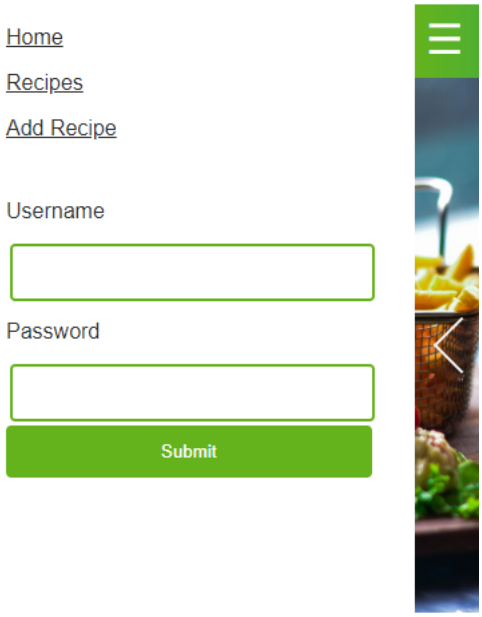
Media querylla asetettiin tyylit 950 px ja sitä leveämpiä näyttöjä varten.

```
@media screen and (min-width: 950px) {
    .navbutton {
        display: none;
    }
    .navi {
        display: grid;
        grid-template-columns: 7% 10% 13% 30% 30% 10%;
        justify-content: start;
    }
    input[type=text] {
        border: 2px solid #65b31c;
        margin: 0;
        padding-right: 0;
        width: 60%;
        margin-top: 1%;
    }
    .form-group {
        display: inline;
        margin: 0;
        padding: 0;
    }
```

Kuvio 10. app.component.css, CSS-tyylit 950 px ja sitä leveämmille näytöille

Lopputuloksena kaksi erilaista navigaatiota. Leveällä näytöllä navigaatio on sivun yläreunassa lineaarisesti. Alle 950 px näytöllä sivulta aukeavan navigaation saa esille painamalla yläreunan hampurilaisikonia.

Kuvio 11. Navigointipalkki 950 px tai leveämmällä näytöllä



Kuvio 12. Navigointi avattuna alle 950 px leveällä näytöllä

Vaihtopainike CSS:lla

Reseptien listausnäkymän vaihtopainikkeen toteuttamiseen kirjoitettiin melko paljon CSS-koodia. Toteutukseen löytyi useita ohjeita tutoriaaleista ja alan foorumeilta.

```
<div class="toggleButton">
  Card view
  <label class="switch">
    <input type="checkbox" [(ngModel)]="checkBoxValue" [ngModelOptions]="{standalone: false}">
    <span class="slider round"></span>
  </label>
  Table view
</div>
```

Kuvio 13. recipes.component.html

```css
.toggleButton {
    margin-top: 5%;
}
.switch {
    position: relative;
    display: inline-block;
    width: 60px;
    height: 34px;
}
.switch input {
    opacity: 0;
    width: 0;
    height: 0;
}
.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #ccc;
  -webkit-transition: .4s;
  transition: .4s;
}
.slider:before {
  position: absolute;
  content: "";
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  -webkit-transition: .4s;
  transition: .4s;
}
input:checked + .slider {
  background-color: #65b31c;
}
input:focus + .slider {
  box-shadow: 0 0 1px #65b31c;
}
input:checked + .slider:before {
  -webkit-transform: translateX(26px);
  -ms-transform: translateX(26px);
  transform: translateX(26px);
}
.slider.round {
  border-radius: 34px;
}
.slider.round:before {
  border-radius: 50%;
}
```

Kuvio 14. recipes.component.css

Kuvio 15. CSS:lla toteutettu vaihtopainike

4.6 Testisovelluksen käyttöliittymän versio Bootstrapilla

Bootstrapin lisääminen Angular-projektiin

Ensin asennettiin Bootstrap, jQuery ja Popper.js npm:n avulla terminaalissa projektikansion juuressa seuraavilla komennoilla.

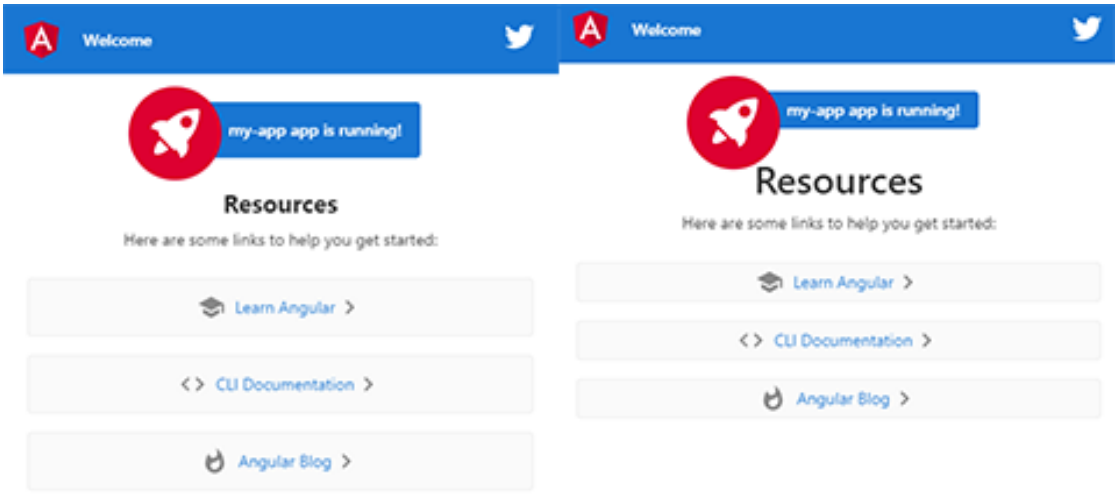
```
npm install bootstrap

npm install jquery popper.js
```

Tämän jälkeen projektin juuresta löytyvään angular.json tiedostoon muokattiin ”style”- ja ”script”-polkuihin arvoiksi node_modules-kansiossa olevien Bootstrap-, jQuery- ja Popper.js-kirjastojen polut.

```
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "node_modules/jquery/dist/jquery.min.js",
  "node_modules/popper.js/dist/umd/popper.min.js",
  "node_modules/bootstrap/dist/js/bootstrap.min.js"
],
```

Kuvio 16. Kirjastojen polut angular.json tiedostossa.

Kun Bootstrapin asennus on onnistunut ja Angular-sovelluksen serveri käynnistetään selaimessa, näkyviin tuleva tervehdysviesti on saanut Bootstrap-tyylittelyn.



Kuvio 17. Angular-sovelluksen tervehdys alkuperäisenä ja Bootstrapin tyylittelyillä.

Bootstrap tarjoaa useita erilaisia valmiita käyttöliittymäkomponentteja, joiden laittaminen projektiin on verrattain helppoa. Tässä projektissa käytettiin valmiina komponentteina navigaatiokomponenttia, form-komponentteja ja korttikomponenttia.

Bootstrap-navigaatio

Bootstrap tarjoaa valmiin komponentin navigaatiota varten. Komponentti on valmiiksi responsiivinen, joten CSS-koodia ei tarvitse kirjoittaa välttämättä lainkaan, vain sen verran kuin halutaan muokata navigaation sisältöä. TypeScriptiä ei tarvitse kirjoittaa yhtään. HTML-koodi rakennettiin Bootstrapin dokumentaation mukaan ja lisättiin oman projektin tarvitsemat elementit.

```
<nav class="mynavbar navbar navbar-expand-lg navbar-light bg-light">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
  aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" routerLink="">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="/recipes">Recipes</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="/addrecipe">Add Recipe</a>
      </li>
    </ul>
    <form class="form-inline">
      <div class="form-row">
        <div class="form-group col-auto">
          <label for="username">Username</label>
          <input type="username" class="form-control" id="uname" placeholder="Username">
        </div>
        <div class="form-group col-auto">
          <label for="password">Password</label>
          <input type="password" class="form-control" id="password" placeholder="Password">
        </div>
        <button type="submit" class="mybutton btn btn-primary">Submit</button>
      </div>
    </form>
  </div>
</nav>
<router-outlet></router-outlet>
```

Kuvio 18. app.component.html

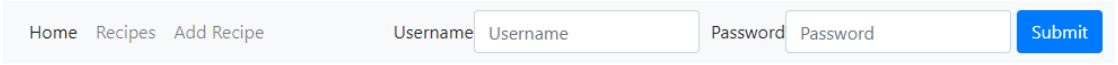
Tässä projektissa oli tarpeen ylikirjoittaa valmiin navigaation tyyliä. Suunnitelman mukaan leveän näytön navigaatiopalkin taustaväriksi laitettiin vihreä liukuväri, fontti laitettiin valkoiseksi ja submit-nappi laitettiin läpinäkyväksi. Mobiiliystävällisen version taustaväriksi asetettiin valkoinen, fonttiväriksi musta ja submit-napin väriksi vihreä.

```
.mynavbar {
    background-image: linear-gradient(
        to right, #ffffff, #ffffff, #ffffff, #ffffff, #ffffff);
    color: #000000;
}
.mynavbar a {
    color: #000000;
}
.mynavbar a:active {
    color: #000000;
}
.mybutton {
    border: none;
    background-color: #65b31c;
    color: #ffffff;
}
input[type=username] {
    margin-left: 5px;
    margin-right: 5px;
}
input[type=password] {
    margin-left: 5px;
}
```

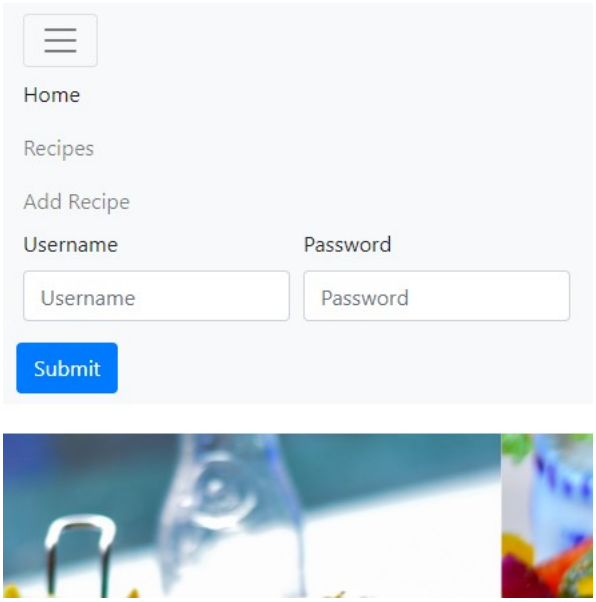
```
@media screen and (min-width: 992px) {
    .mynavbar {
        background-image: linear-gradient(
            to right, #65b31c, #52b02f, #3fac3e, #28a74a, #00a354);
        color: #ffffff;
    }
    .mynavbar a {
        color: #ffffff;
    }
    .mynavbar a:active {
        color: #ffffff;
    }
    .mybutton {
        color: #ffffff;
        background-color: transparent;
    }
}
```

Kuvio 19. app.component.css

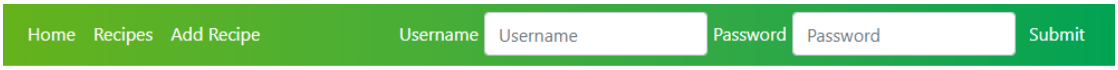
Ilman CSS-ylikirjoitusta Bootstrapin oletuksena on napeissa Bootstrapin ”primary”-väri, sininen. Navbarin taustaväri on harmaa ja fontit tummempaa ja vaaleampaa harmaata, koska Navbarille on annettu sen tyylin määrittelevä Bootstrapin luokka ”navbar-light”.



Kuvio 20. Bootstrapin Navbar ennen CSS-ylikirjoitusta

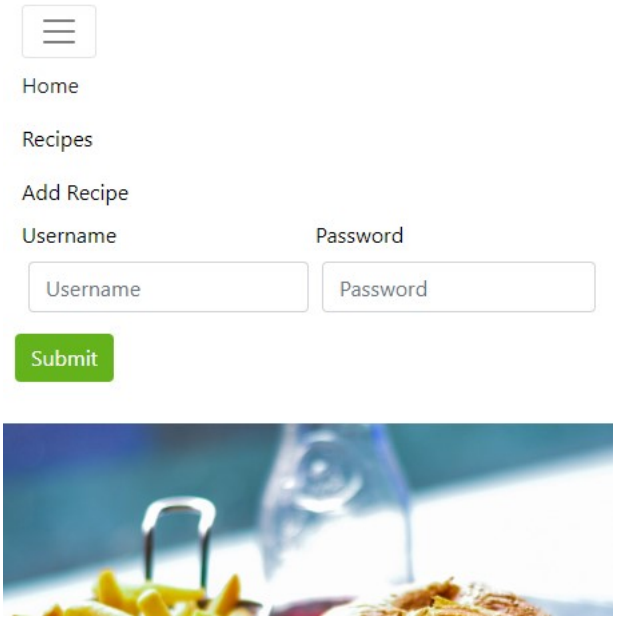


Kuvio 21. Mobiiliystävällinen Bootstrapin Navbar ennen CSS-ylikirjoitusta



Kuvio 22. Bootstrapin Navbar CSS-ylikirjoitusten jälkeen

Bootstrapin Navbar-komponenttiin on sisäänrakennettuna vaihtajaominaisuus, jonka avulla tietyssä pysäytyspisteessä navigaatiopalkki muuttuu mobiiliystävälliseksi, ham-purilaisikonista ylhäältä alas aukeavaksi valikoksi.



Kuvio 23. Mobiiliystävällinen Bootstrapin Navbar CSS-ylikirjoituksen jälkeen

Bootstrap vaihtopainike

Bootstrapin vaihtopainikkeen tyylin muuttaminen osoittautui erittäin hankalaksi, eikä siinä onnistuttu.

```
<div class="mySwitch">
  <h6 class="mySwitchItem">Card view</h6>
  <div class="custom-control custom-switch mySwitchItem">
    <input type="checkbox" class="custom-control-input" id="customSwitch1" [(ngModel)]="checkBoxValue" [ngModelOptions]="{standalone: false}">
    <label class="custom-control-label myLabel" for="customSwitch1"></label>
  </div>
  <h6 class="mySwitchItem">Table view</h6>
</div>
```

Kuvio 24. recipes.component.html

Kuvio 25. Bootstrap vaihtopainike

4.7 Testisovelluksen käyttöliittymän versio Semantic UI:lla

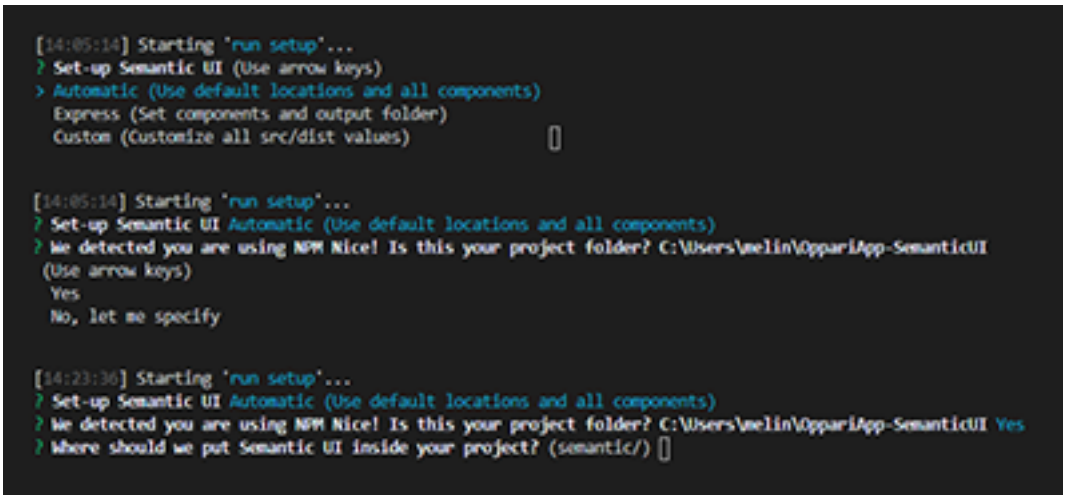
Semantic UI:n lisääminen Angular-projektiin

Semantic UI tarjoaa mahdollisuuden koota itse oman kirjastonsa, jonne sisällyttää vain tarvittavat komponentit. Oma kirjasto käännetään koontiversioksi Gulp-komentorivityökalulla. Gulp tulee olla globaalisti asennettuna. Semantic UI vaatii myös jQueryn. Gulp ja jQuery asennettiin terminaalissa npm:lla seuraavilla komennoilla:

```
npm install --global gulp-cli

npm install jquery --save
```

Semantic UI lisättiin projektiin npm:lla terminaalissa projektikansion juuressa seuraavalla komennolla:

```
npm install semantic-ui --save
```



Kuvio 26. Semantic UI:n asennuksen valinnat

Asennuksen jälkeen navigoitiin terminaalissa semantic-kansioon ja ajettiin gulpin build-komento. Käännöksen jälkeen angular.json-tiedostoon lisättiin ”styles”- ja

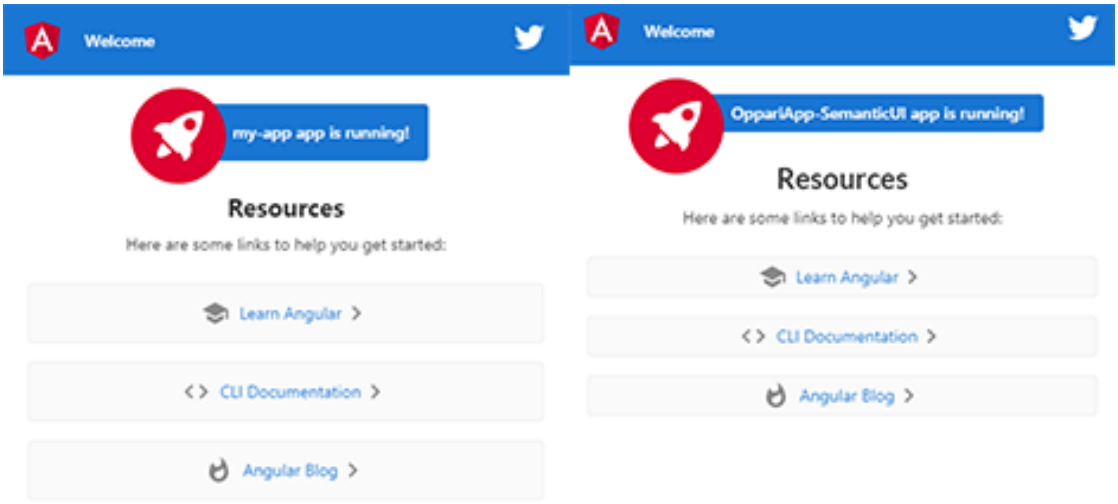
”scripts”-kohtiin minifioitujen Semantic UI:n CSS- ja JS-tiedostojen ja jQueryn JS-tiedoston polut.

```
cd semantic

gulp build
```

```
"styles": [
  "src/styles.css",
  "semantic/dist/semantic.min.css"
],
"scripts": [
  "semantic/dist/semantic.min.js",
  "./node_modules/jquery/dist/jquery.min.js"
]
},
```

Kuvio 27. angular.json

Kun SemantiUI:n asennus on onnistunut ja Angular-sovelluksen serveri käynnistetään selaimessa, näkyviin tuleva tervehdysviesti on saanut Semantic UI-tyylittelyn.



Kuvio 28. Angular-sovelluksen tervehdys alkuperäisenä ja Semantic UI:n tyylittelyillä.

Semantic UI navigaatio

Semantic UI ei tarjoa valmiina komponenttina vastaavasti responsiivista navigaatiota kuin Bootstrap. Vastaavan voi toteuttaa yhdistelemällä kaksi erilaista navigaatiota ja

lisäämällä toiminnallisuutta jQuery-koodilla. Tähän projektiin edellä mainittua ratkaisua toteutettaessa huomattiin, ettei jQueryn lisääminen projektiin onnistunut. Ratkaisua yritettiin hakea lisäämällä jQuery projektiin usein eri tavoin, mutta mikään ratkaisu ei tuottanut haluttua tulosta.

Semantic UI vaihtopainike

Myös vaihtopainike tarvitsi toimiakseen jQuery koodia, joten sitäkään ei saatu toimimaan.

4.8 Testisovelluksen käyttöliittymän versio Materializella

Materializen lisääminen Angular-projektiin

Materialize tarvitsee Angularin kanssa käytettäväksi erillisen lisätuen, angular2-materialize, sekä Materialize tyypitykset, jonka saa lisättyä projektiin nmp:lla. Tarvittavia ovat myös jQuery ja Hammer.js. Ne lisättiin projektiin seuraavilla komennoilla:

```
npm install materialize-css@next --save

npm install angular2-materialize –save

npm install --save @types/materialize-css


npm install jquery –save

npm install hammerjs –save
```

Projektiin lisättyjen Materializen, jQueryn ja Hammer.js:n CSS- ja JS-tiedostojen polut lisättiin angular.json-tiedostoon ”architech” > ”build” > ”options” sekä ”architech” > ”test” >”options” konfiguraatioden sisälle kuvan mukaisesti.

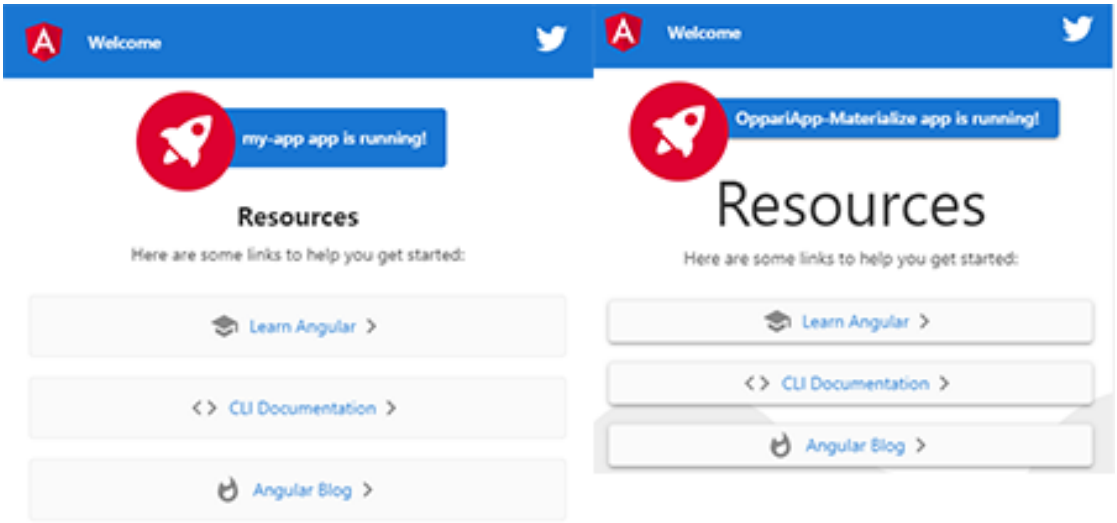
```
"styles": [
  "src/styles.css",
  "node_modules/materialize-css/dist/css/materialize.css"
],
"scripts": [
  "node_modules/materialize-css/dist/js/materialize.js",
  "node_modules/jquery/dist/jquery.js",
  "node_modules/hammerjs/hammer.js"
]
```

Kuvio 29. angular.json

Materialize tyypitykset lisättiin tsconfig.app.json-tiedostoon seuraavasti.

```
"types": ["materialize-css"]
```

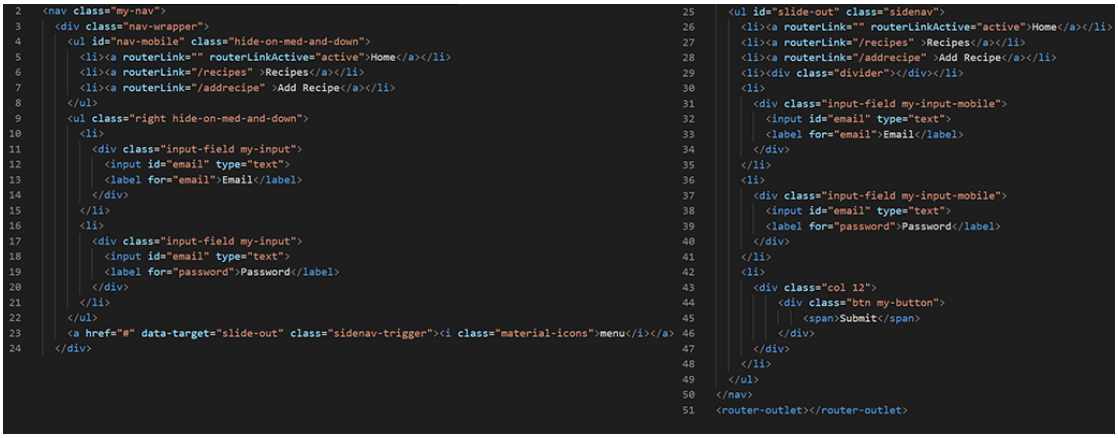
Kun Materializen asennus on onnistunut ja Angular-sovelluksen serveri käynnistetään, selaimessa näkyy Materializen tyylittelemä Angularin tervehdys.



Kuvio 30. Angular-sovelluksen tervehdys alkuperäisenä ja Materializen tyylittelyillä.

Materialize navigaatio

Materializen navigaatio toteutettiin yhdistämällä Materializen Navbar ja Sidenav dokumentaation mukaisesti.



```
<nav class="my-nav">
  <div class="nav-wrapper">
    <ul id="nav-mobile" class="hide-on-med-and-down">
      <li><a routerLink="" routerLinkActive="active">Home</a></li>
      <li><a routerLink="/recipes" >Recipes</a></li>
      <li><a routerLink="/addrecipe" >Add Recipe</a></li>
    </ul>
    <ul class="right hide-on-med-and-down">
      <li>
        <div class="input-field my-input">
          <input id="email" type="text">
          <label for="email">Email</label>
        </div>
      </li>
      <li>
        <div class="input-field my-input">
          <input id="email" type="text">
          <label for="password">Password</label>
        </div>
      </li>
    </ul>
    <a href="#" data-target="slide-out" class="sidenav-trigger"><i class="material-icons">menu</i></a>
  </div>
  <ul id="slide-out" class="sidenav">
    <li><a routerLink="" routerLinkActive="active">Home</a></li>
    <li><a routerLink="/recipes" >Recipes</a></li>
    <li><a routerLink="/addrecipe" >Add Recipe</a></li>
    <li><div class="divider"></div></li>
    <li>
      <div class="input-field my-input-mobile">
        <input id="email" type="text">
        <label for="email">Email</label>
      </div>
    </li>
    <li>
      <div class="input-field my-input-mobile">
        <input id="email" type="text">
        <label for="password">Password</label>
      </div>
    </li>
    <li>
      <div class="col 12">
        <div class="btn my-button">
          <span>Submit</span>
        </div>
      </div>
    </li>
  </ul>
</nav>
<router-outlet></router-outlet>
```

Kuvio 31. app.component.html

Navigaatio vaati myös hieman koodia app.component.ts-tiedostoon toimiakseen. Dokumentaatiosta löytyi alustuskoodi JavaScriptina, mutta suoraan kopioi-liitä ei käynyt vaan alustuskoodi piti muuntaa toimimaan Angularin kanssa. Alustuskoodi Materializen dokumentaatiossa oli seuraavanlainen:

```
document.addEventListener('DOMContentLoaded', function() {
          var elems = document.querySelectorAll('.sidenav');
          var instances = M.Sidenav.init(elems, options);
});
```

Alustuskoodi piti laittaa ngOnInit() ja setTimeOut() -funktioiden sisälle.

```typescript
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    setTimeout( () => {
      const elem = document.querySelector('.sidenav');
      const instance = M.Sidenav.init(elem);
    }, 0);
  }
}
```

Kuvio 32. app.component.ts

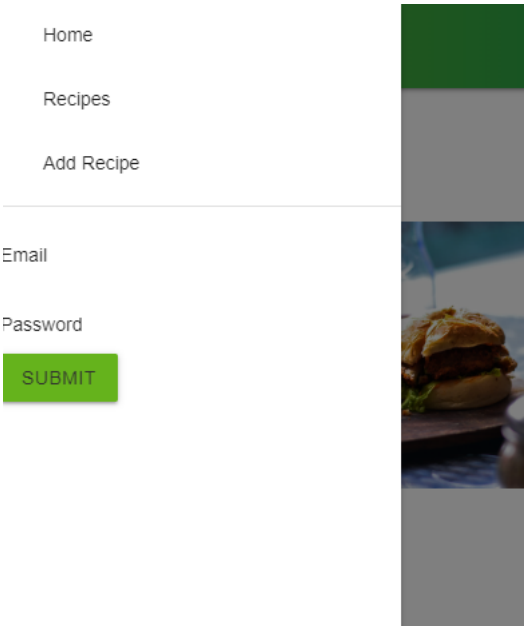
Materialize noudattaa Googlen Material Designia ja väripaletteja on useampia valittavana. Valmiista paleteista ei löytynyt suunnitelmaan sopivia vihreän sävyjä, joten tyylejä piti yli kirjoittaa CSS:lla. Mobiiliystävällinen Sidebar oli valmiiksi pohjaväriltään valkoinen, joten tarpeen oli vaihtaa ainoastaan Navbarin taustaväri, fonttiväri sekä tekstikenttien väri.

```css
.my-nav {
    background-image: linear-gradient(to right, #65b31c, #52b02f, #3fac3e, #28a74a, #00a354);
}
.my-input label{
    color: #ffffff;
}
.my-input-mobile label {
    color: rgba(0, 0, 0, 0.87);;
}
.input-field input:focus + label {
    color: #65b31c;
}
.input-field input:focus {
    border-bottom: 1px solid #65b31c;
    box-shadow: 0 1px 0 0 #65b31c;
}
.hide-on-med-and-down .input-field input:focus + label {
    color: #ffffff;
}
.hide-on-med-and-down .input-field input:focus {
    border-bottom: 1px solid #ffffff;
    box-shadow: 0 1px 0 0 #ffffff;
}
.my-button {
    background-color: #65b31c;
    color: #ffffff;
}
```

Kuvio 33. app.component.css

Tuloksena oli hyvin samanlaiset navigaatiot kuin suunnitelmassa. Materializen tekstikentät ovat tyylitelty erilaisiksi kuin suunnitelmassa, mutta niitä ei lähdetty muuttamaan.

Kuvio 34. Materializen Navbar



Kuvio 35. Materializen mobiiliystävällinen Sidenav

Materialize vaihtopainike

Materializen vaihtopainikkeen tyyli oli helposti vaihdettavissa suunnitelman mukaiseen väriin.

```
<div class="switch">
        Card view
    <label>

        <input type="checkbox" [(ngModel)]="checkBoxValue" [ngModelOptions]="{standalone: false}">
        <span class="lever"></span>

    </label>
    Table view
</div>
```

Kuvio 36. recipes.component.html

```
.switch label input[type=checkbox]:checked+.lever {
    background-color: #65b31c;
}

.switch label input[type=checkbox]:checked+.lever:after {
  background-color: #65b31c;
}
```

Kuvio 37. recipes.component.css

Kuvio 38. Materialize vaihtopainike

5 Johtopäätökset

5.1 Vertailun tulokset

Taulukko 4. CSS-sovelluskehysten pisteytys

Kriteeri	Bootstrap	Semantic UI	Materialize	CSS
Dokumentaation taso	★★★★★	★★★★	★★★★	★★★★★
Ajankäyttö	★★★★	Ei arvioitu	★★★★	★★★
Responsiivisuus	★★★★★	★★★	★★★★★	★★★★★
Yhteisö	★★★★	★★★	★★★	Ei arvioitu
Yhteensopivuus Angularin kanssa	★★★★★	★	★★★★	★★★★★

Dokumentaation taso

Kaikilta sovelluskehyksiltä löydettiin kattavat dokumentaatiot. Komponenteista oli esiteltynä useita erilaisia versioita hyvin koodinäyttein. Bootstrapilla todettiin olevan hieman selkeämpi dokumentaatio kuin Materializella ja Semantic UI:lla. Komponent-

teja ei ollut jaettu useisiin eri kategorioihin, jolloin tarvittavien komponenttien löytäminen oli helpompaa ja nopeampaa. CSS:n kohdalla tarkasteltiin W3Schoolsin CSS-tutoriaalia. Tutoriaalista löytyi kattavasti, miten CSS:llä toteutetaan responsiivista web designia sekä tietoa ja ohjeita CSS gridistä.

Ajankäyttö

Semantic UI:n kohdalla jätettiin arvioimatta tämä kriteeri, sillä kaikkia testisovellukseen suunniteltuja ominaisuuksia ei voitu toteuttaa. Testisovelluksen toteuttamiseen Bootstrapilla kului aikaa noin puoli työpää, mutta lisäksi aikaa kului pari tuntia, kun vaihtopainikkeen värin vaihtamista yritettiin selvittää. Toteutukseen Materializella kului vajaa työpäivä. Bootstrapin lisääminen projektiin oli nopeampaa ja yksinkertaisempaa kuin Materializen. CSS:llä toteutettuun testisovellukseen aikaa kului yksi työpäivä. Lisäksi puoli työpäivää kului sopivien Angularin kanssa yhteensopivien valmiiden navigaatio- ja karusellikomponenttien etsimiseen.

Responsiivisuus

Bootstrapin ja Materializen komponentit toimivat responsiivisesti erittäin hyvin. Semantic UI:n mobiiliystävällistä navigaatiota ei saatu toimimaan lainkaan, eikä indexsivun lomaketta saatu toimimaan kunnolla. Responsiivisuuden toteuttaminen CSS:llä onnistui hyvin.

Yhteisö

Kaikilta sovelluskehyksiltä löydettiin Twitter-tili. Huomattiin, ettei yksikään sovelluskehys ollut erityisen aktiivinen julkaisija Twitterissä. Kaikilta löydettiin myös GitHubi repositoryt. Semantic UI:n GitHubin readme:ssä huomattiin olevan suora linkki StackOverFlown sivuille kysymyksiin, joihin oli liitetty Semantic UI asiasanana. Sovelluskehysten web-sivuja tutkittaessa kävi ilmi, että Semantic UI:lla ja Materializella on käytössä Gitter chathuone. Semantic UI:lla chatissa on hieman yli 3000 käyttäjää, Materializella lähes 11 000. Bootstrapilla käytössä on Slack, jossa heidän työtilassaan on lähes 30 000 käyttäjää. Kuitenkaan kenenkään chatpalveluissa keskustelu ei ollut

kovinkaan vilkasta, eikä kirjoituksia ole edes päivittäin. Bootstrapille annettiin yksi tähti enemmän siitä, että he käyttävät Slackia. Slack on useilla ohjelmistokehittäjillä jo valmiiksi käytössä ja Slackissa on mahdollista luoda kanavia eri aiheista, jolloin keskustelua on helpompi seurata. CSS:n kohdalla tätä kriteeriä ei arvioitu, mutta suositulta CSS-tricks.com sivustolta löytyi foorumi, jossa keskustelua havaittiin olevan päivittäin.

Yhteensopivuus Angularin kanssa

Bootstrapin yhteensovittamisessa Angularin kanssa ei ollut mitään ongelmia ja Bootstrapin lisääminen Angulariin onnistui helposti, eikä asentaa tarvinnut mitään muuta kuin dokumentaatiossa mainitut. Materializen saaminen toimimaan Angular-projektissa vaati angular2-materialize ja @types/materialize-css asennuksen, joita ei Materializen dokumentaatiossa manittu. Semantic UI:ta ei saatu sovitettua yhteen, sillä jQuerya ei saatu otettua käyttöön TypeScript-tiedostossa. CSS:n kanssa ei ollut lainkaan yhteensopivuusongelmia.

Yhteenveto

Tulosten perusteella voidaan todeta, että Bootstrap ja Materialize soveltuvat hyvin Angularin kanssa käytettäviksi. Sen sijaan tutkimusten tulosten perusteella Semantic UI:n käyttöä Angularin kanssa ei voida suositella. Sovelluskehysten tarjoamat komponentit helpottavat huomattavasti työtä, mutta ajankäytöllisesti tämän kokoisen sovelluksen kehittämissä ei ollut kovin suurta eroa siinä, käytettiinkö sovelluskehystä vai vain CSS:ää.

Materialize tarjoaa hyviä komponentteja, jotka noudattavat Material Design tyyliohjetta, joten sovelluksesta saa helposti tyylikkään näköisen, mikäli ei itse halua tyylin suunnitteluun erityisesti käyttää aikaa, tai tavoitteena on tehdä staattinen prototyyppi tai demo. Kaikki testatut sovelluskehykset sopivat hyvin prototyyppien tekemiseen, sillä niillä saa nopeasti hyvännäköisiä, toimivia komponentteja, jolloin prototyypin esittely ja testaus on mielekästä.

5.2 Pohdinta

Tutkimuksen tavoitteena oli selvittää, miten kolme GitHubin tähtien mukaan suosituinta CSS-ohjelmistokehystä on yhteensopivia Angularin kanssa ja milloin niiden käyttö on paikallaan.

Tutkimuksen tuloksena saatiin tietoa, miten CSS-ohjelmistokehykset soveltuvat Angular-projekteihin ja millaista hyötyä on käyttää sovelluskehyksiä, eikä ainoastaan CSS:ää. Kuten jo tietoperustaosassa kävi ilmi, että CSS-sovelluskehykset ovat hyviä työkaluja prototyyppien tekemiseen, todettiin se myös tutkimuksen toteutusta tehdessä.

Tutkimustyön aikana tuli vastaan haasteita ja paljon uutta opittavaksi. Käytetyt webteknologiat olivat ennestään hieman tuttuja ja kokemustakin niistä jonkin verran, mutta tietoperustaosasta saatiin silti paljon uutta tietoa. CSS-sovelluskehykset olivat myöskin hieman tuttu aihe jo, mutta valituista sovelluskehyksistä vain Bootstrap oli entuudestaan tuttu. Semantic UI ja Materialize täysin vieraita, joten uutta tietoa ja taitoa kertyi tutkimuksen aikana.

Tutkimuksella on jatkotutkimuspotentiaalia, sillä tässä tutkimuksessa ei otettu huomioon sovelluskehysten kehitystä ja tulevia ominaisuuksia. Tutkimuksen vertailussa ei myöskään otettu huomioon sovelluskehysten Angularille kehittämiä versioita. Näiden versioiden ottaminen mukaan vertailuun, olisi antanut vahvemman pohjan tutkimukselle ja sen tuloksille. Myöskin testisovellus oli verrattain pieni eikä ominaisuuksia ollut kovin montaa. Isommalla ja enemmän ominaisuuksia sisältävällä testisovelluksella olisi käytetty laajemmin erilaisia komponentteja, jolloin sovelluskehysvertailu olisi ollut tarkempi ja validimpi. Jatkotutkimusmahdollisuutena on myös vertailla CSS-sovelluskehyksiä muiden front-end-sovelluskehysten, kuten Reactin tai Vue.js:n kanssa. Tutkimuksen tuloksia voivat hyödyntää opiskelijat ja aloittelevat kehittäjät valitessaan teknologioita käyttöliittymän toteutukseen.

Lähteet

A look back at HTML5. 2017. Blogikirjoitus Fasthosts:n sivuilla 17.2.2017. Viitattu 10.10.2018. https://www.fasthosts.co.uk/blog/websites/look-back-html5

Angular. N.d. Angularin oma dokumentaatio. Viitattu 18.10.2018. https://angular.io

Basic structure of an HTML document. N.d. Raj Singh:n ylläpitämä websivu web-tekniikoista. Viitattu 16.10.2018. http://www.scriptingmaster.com/html/basic-structure-HTML-document.asp

Birnir A. 2018. Should You Learn JavaScript? The answer is YES. 17.6.2018. Viitattu 28.10.2018. https://skillcrush.com/2014/04/10/learn-javascript/

Bootstrap documentation. N.d. Bootstrapin websivun dokumentaatio. Viitattu 17.10.2018. https://getbootstrap.com/docs/4.1/getting-started/introduction/

Bootstrap. N.d. Bootstrapin websivu. Viitattu 17.10.2018. https://getbootstrap.com

CSS current status. N.d. World Wide Web Consortiumin standardit kokoava sivu. Viitattu 11.10.2018. https://www.w3.org/standards/techs/css#w3c_all

CSS frameworks. N.d. Sitecaken kokoama lista Githubin mukaan suosituimmista CSS frameworkeistä. Viitattu 17.10.2018. https://sitecake.com/resources/css-frameworks.html

CSS introduction. N.d. W3School CSS tutoriaali. Viitattu 16.10.2018. https://www.w3schools.com/css/css_intro.asp

CSS syntax. N.d. MDN web docs sivusto. Viitattu 16.10.2018. https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Syntax

CSS. 2017. Sanakirja Computer Hopen sivuilla 11.10.2017. Viitattu 11.10.2018. https://www.computerhope.com/jargon/c/css.htm

CSS. N.d. Internettermistö sanakirja TechTerms. Viitattu 11.10.2018. https://techterms.com/definition/css

Enge E. Where is the Mobile vs. Desktop Story Going? Artikkeli Perficient Digitalin sivustolla liittyen ”Mobile vs Desktop Usage”-tutkimukseen. 11.4.2019. Viitattu 18.11.2019. https://www.perficientdigital.com/insights/our-research/mobile-vs-desktop-usage-study

Forsström M. CSS:n lyhyt historia. Fraktion blogi 18.10.2016. Viitattu 11.10.2018. https://fraktio.fi/blogi/cssn-lyhyt-historia/

Forsström M. JavaScriptin lyhyt historia. Fraktorin blogi 1.10.2015. Viitattu 12.10.2018. https://fraktio.fi/blogi/javascriptin-lyhyt-historia/

Gasston P. 2014. Book of CSS 3 : A Developer's Guide to the Future of Web Design. San Francisco: No Starch Press

Gerchev I. 2014. Introducing: Semantic UI Component Library. 7.2.2014. Sitepoint sivuston artikkeli. Viitattu 18.10.2018. https://www.sitepoint.com/introducing-semantic-ui-component-library/

Halme A. 2018. Mikä on single-page app ja mihin sitä käytetään? 10.4.2018. Viitattu 22.10.2018. https://citydevlabs.fi/single-page-app/

HTML. 2018. Sanakirja Computer Hopen sivuilla 3.8.2018. Viitattu 10.10.2018. https://www.computerhope.com/jargon/h/html.htm

Kananen, J. 2008. Kvali. Kvalitatiivisen tutkimuksen teoria ja käytänteet. Jyväskylä: Jyväskylän yliopistopaino.

Kurniawan, B. 2015. HTML : A Beginner's Tutorial. Brainy Software.

Lawson K. 2018. What is a Single Page Application and why do people like them so much? 19.7.2018. Viitattu 22.10.2018. https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html

Maduro H. 2018. Rapid Prototyping in Bootsrap 4.0. 7.6.2018. Blogikirjoitus Epiphyn verkkosivuilla. Viitattu 22.11.2019. https://epiphycorp.com/2018/06/07/rapid-prototyping-in-bootstrap/

Pilon A. What is Responsive Web Design? Artikkeli Small Busines Trends sivustolla. 22.5.2018. Viitattu 18.11.2019. https://smallbiztrends.com/2013/05/what-is-responsive-web-design.html

Punchatz C. 2017. How JavaScript Became the Dominant Language of the Web. 7.8.2017. Viitattu 28.10.2018. https://www.lform.com/blog/post/how-javascript-became-the-dominant-language-of-the-web

Routio P. 2007. Vertailu. 3.8.2017. Viitattu 18.10.2018. http://www2.uiah.fi/projects/metodi/072.htm

Saxena R. 2017. A Beginner's Guide to CSS Front End Frameworks. ZipBoard projektihallintajärjestelmän Web-ohjelmointiaiheinen blogi. 23.8.2017. Viitattu 17.10.2018. https://blog.zipboard.co/a-beginners-guide-to-css-front-end-frameworks-8045a499456b

Tarvainen J. 2017. Mikä on TypeScript? 30.7.2016. Viitattu 28.10.2018. https://symfony.fi/artikkeli/mika-on-typescript

VanToll T. 2017. What is Angular? 18.1.2017. Artikkeli Telerik Developer Network sivustolla. Viitattu 18.10.2018. https://developer.telerik.com/topics/web-development/what-is-angular/

What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design. N.d. Awwwards designereille ja web-ohjelmoijille suunnatun verkkolehden artikkeli. Viitattu 17.10.2018. https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html

What is Materialize CSS. N.d. Javatpointin Materialize tutoriaali. Viitattu 22.10.2018. https://www.javatpoint.com/what-is-materialize-css

What is TypeScript and Why Should You Use it? 2017. Blogikirjoitus web-markkinointiin, -ohjemointiin ja -designiin suunnatussa WakeFly-blogissa 29.8.2017. Viitattu 28.10.2018. https://www.wakefly.com/blog/what-is-typescript-and-why-should-you-use-it/