RESPONSIIVISEN MOBIILISOVELLUKSEN KÄYTTÖLIITTYMÄ

Case: Digiloop

LAHDEN AMMATTIKORKEAKOULU
Insinööri (AMK)
Tieto- ja viestintätekniikka
Kevät 2019
Lassi Hirvonen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Hirvonen, Lassi	Opinnäytetyö, AMK	Kevät 2019
	Sivumäärä	
	34	

Työn nimi

Responsiivisen mobiilisovelluksen käyttöliittymä
Case: Digiloop

Tutkinto

Insinööri (AMK), tieto- ja viestintätekniikka

Tiivistelmä

Opinnäytetyön tavoitteena oli luoda kierrätyssovellus Digiloop. Sovelluksen tarkoituksena on helpottaa kotitalouksista löytyvien akkujen, sähkö- ja elektroniikkaromun sekä tietoturvajätteen kierrätystä. Tavoitteen taustalla on lakiuudistus, joka pyrkii kasvattamaan kierrätykseen päätyvien jätteiden osuutta.

Sovellus toteutettiin verkkosovelluksena. Isoimman osan käyttäjistä uskotaan käyttävän sovellusta mobiililaitteella, joten opinnäytetyössä keskitytään paljon sovelluksen responsiivisiin ratkaisuihin. Käyttöliittymässä on hyödynnetty Material UI -kirjastoa, joka helpottaa responsiivisen toteutuksen luontia.

Sovelluksesta haluttiin myös mobiiliapplikaatio. Tämä toteutettiin luomalla sovelluksesta Apache Cordova hybridisovellus Androidille, jolloin verkkosovellusta voidaan suorittaa Cordovan avulla mobiilisovelluksena. Mobiiliapplikaation tarkoituksena oli helpottaa sovelluksen käyttöä poistamalla tarve verkkosovelluksen sivulla käymiseen.

Lopputuloksena saatiin luotua toimiva prototyyppi sovelluksesta. Sovelluksen kaikki olennaisimmat toiminnot saatiin toimimaan. Verkkosovellus on responsiivinen ja kykenee mukautumaan käyttäjän laitteen vaatimuksiin. Android-hybridisovellus saatiin toteutettua ja se jaettiin projektiin osallistuneille osapuolille.

Asiasanat

Apache Cordova, responsiivinen, Material UI

Abstract

Author(s)	Type of publication	Published
Hirvonen, Lassi	Bachelor's thesis	Spring 2019
	Number of pages	
	34	

Title of publication

A mobile application's responsive user interface
Case: Digiloop

Name of Degree

Bachelor of Engineering, Information and Communications Technology

Abstract

The goal of the thesis was to create a recycling application called Digiloop. The application aims to ease the recycling of battery, electrical and information security waste. The goal is based on a new law, which aims to increase the amount of waste that ends up recycled.

The application was implemented as a web application. Most of the users are expected to use the application on a mobile device, so the thesis focuses on the responsive implementations. The user interface uses the Material UI library, which helps in creating a responsive solution.

There was also a demand for a mobile application. This was done by creating an Apache Cordova hybrid application for Android, which allows using the web application as a mobile application. The goal of the mobile application was to make the software easier to use by removing the need to visit the web application's website.

As a result, a working prototype of the software was created. All of the application's most central functionalities work. The web application is responsive, and is able to adapt to the requirements of the user's device. The Android hybrid application was successfully created and shared with the parties involved in the project.

Keywords

Apache Cordova, responsive, Material UI

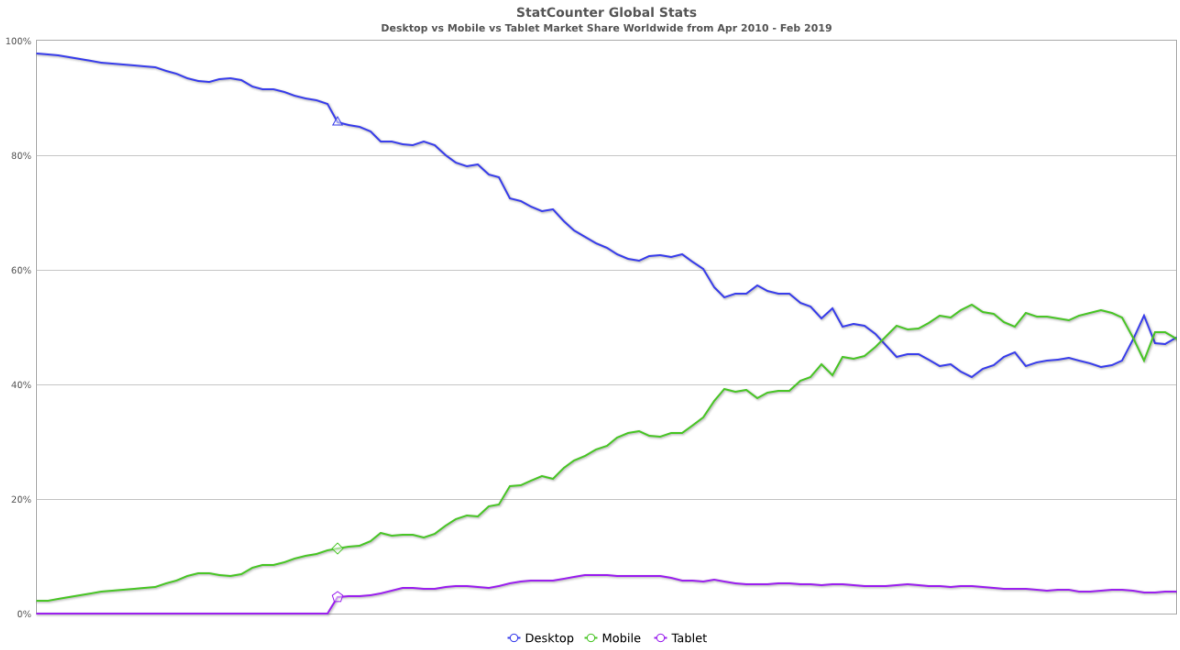
LYHENNELUETTELO

API	Application programming interface, ohjelmistorajapinta.
CLI	Command line interface, komentokehote.
CSS	Cascading style sheets, tyyliohjeet verkkosivulle.
HTTP	Hyper text transfer protocol, hypertekstin siirtoprotokolla.
NPM	Node packet manager.
REST	Representational state transfer.
SDK	Software development kit, sovelluskehitystyökalut jollekin tietylle alustalle.
SER	Sähkö- ja elektroniikkalaiteromu.

SISÄLLYS

1 JOHDANTO

Verkkosivustoja käytetään nykyään hyvin erilaisilla laitteilla. Perinteisen tietokoneen lisäksi verkkosivuja voidaan käyttää esimerkiksi puhelimilla, taulutietokoneilla ja älykelloilla. Kuten kuviosta 1 nähdään, etenkin mobiililaitteiden käyttäjämäärät ovat kasvaneet viimeisen vuosikymmenen aikana huomattavasti. Verkkosivustoista on syytä siis tehdä responsiivisia, jolloin niiden sisältö kykenee sopeutumaan kunkin laitteen vaatimusten mukaiseksi. Näin voidaan taata paras mahdollinen käyttökokemus laitteesta riippumatta.



KUVIO 1. Internetin käyttäjämäärät käyttölaitteittain (StatCounter 2019)

Responsiivista rakennetta voidaan toteuttaa sekä yksittäisillä CSS (cascading style sheets) -valinnoilla että laajemmin vaikuttavilla kirjastoilla ja rakennesuunnitteluilla. Esimerkiksi Material UI -kirjaston mukana tulee responsiivisia työkaluja, joiden käytöllä saadaan sivustosta responsiivisempi.

Opinnäytetyön tavoitteena on toteuttaa kierrätystä kehittävä sovellus Digiloop, joka toimii prototyyppinä esittelemään, miten jätteenkierrätyssovelluksen periaate toimisi käytännössä. Koska sovellusta oletetaan käytettävän hyvin erilaisilla laitteilla, on sovelluksen toimiva responsiivisuus keskeisimpiä tavoitteita.

Projektin kehitys aloitettiin syksyllä 2017. Sovelluksen kehitykseen osallistui yhteensä yhdeksän henkeä. Projektiin osallistui myös BLTK Banaanilaatikko Oy:n sekä Tramel Oy:n edustajat, jotka antoivat alan näkemystä jätteenkäsittelyn tarpeista.

Sovelluksesta on tarkoitus toteuttaa myös Android-hybridisovellus hyödyntäen Apache Cordova -kirjastoa. Hybridisovelluksen avulla palvelua voidaan käyttää mobiililaitteilla suoraan, jolloin palvelun käyttö helpottuu.

2 TOIMINTAYMPÄRISTÖN KUVAUS

Iso osa jätteistä jää kierrätyksen ulkopuolelle. Jätteitä voi kertyä ihmisten koteihin tai yritysten varastoihin. Niiden haltijoilla ei välttämättä ole tietoa, miten jätteet saataisiin takaisin kierrätyksen piiriin. Jätekierrätyskeskukset voivat myös olla kaukana, eikä kaikilla välttämättä ole kykyä kuljettaa jätteitään.

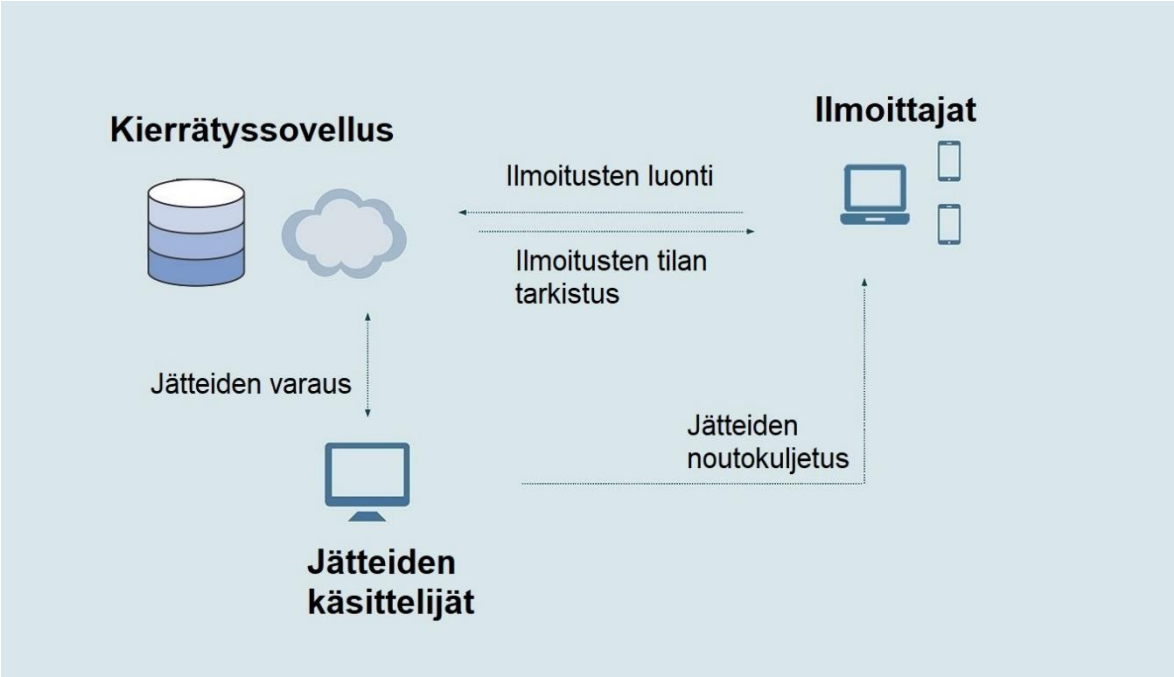
Hallituksen esityksen pohjalta Suomen jätelakia on uudistettu. Uuden lain on tarkoitus saada suurempi osuus jätteistä kierrätettyä ja kasvatettua tuottajan vastuuta jätteiden käsittelystä. Uusi laki astui voimaan 1. tammikuuta 2019. Uuden lain yhteydessä julkaistiin myös uusi valtakunnallinen jätesuunnitelma, jossa tarkennetaan jätehuollon tavoitteita. (Laki jätelain muuttamisesta 445/2018.)

SER-jätteiden osalta suunnitelmassa painotetaan kierrätetyn osuuden kasvattamista sekä SER-jätteen osuuden vähentämistä sekajätteessä. Tavoitteita lähestytään kierrätysmahdollisuuksien parantamisen sekä niistä tehokaan tiedottamisen kautta. Lisäksi suunnitelmalla pyritään pidentämään laitteiden käyttöikää. (Ympäristöministeriö 2016.)

Digiloop on kierrätyssovellusprojekti, jolla yritetään parantaa kierrätykseen päätyvän jätteen määrää. Sen tarkoituksena on luoda sovellus, jonka avulla yritykset sekä yksityishenkilöt kykenevät ilmoittamaan yrityksistään ja kodeistaan löytyvästä sähkö- ja elektroniikkaromusta. Sovellus välittää ilmoitukset jätteenkäsittely-yrityksille, jotka järjestävät jätteille noudon. Keskeisimpänä tavoitteena on mahdollistaa isomman osan SER-jätteiden päätymisestä kierrätykseen.

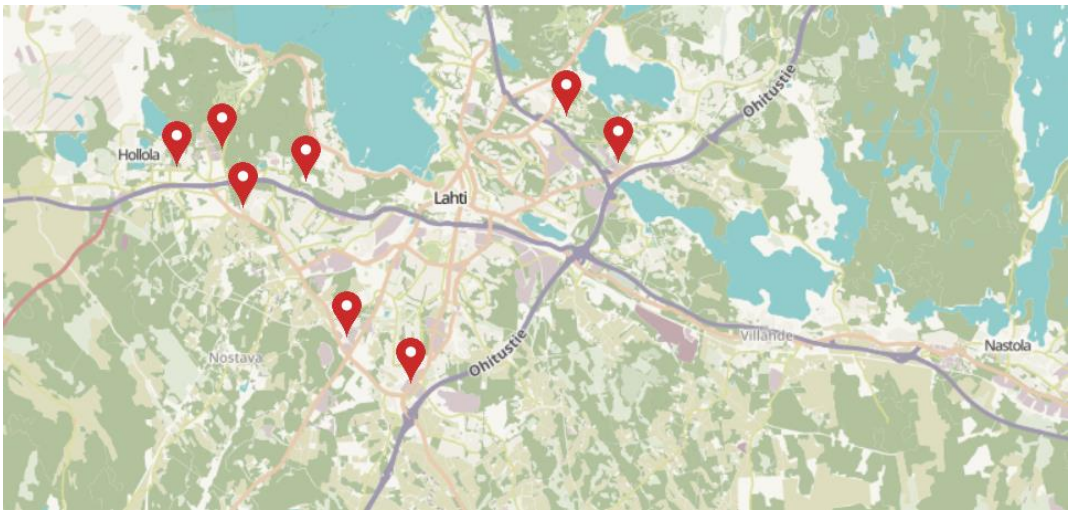
Jätteiden ilmoittamiseen käyttäjien uskotaan käyttävän enimmäkseen mobiililaitteita. Todennäköisesti osa ilmoituksista tehdään kuitenkin tietokoneilla, joten sovelluksen käyttö on mahdollistettava myös verkkoselaimella. Sovelluksen ilmoitusosa voidaan suunnitella ensisijaisesti mobiililaitteille, mutta sen on toimittava responsiivisesti myös tietokoneilla sekä taulutietokoneilla.

Kuviossa 2 nähdään sovelluksen keskeisimmät toiminnot sekä käyttäjäryhmät. Käyttäjillä on oltava kyky seurata ilmoitetun jätteen tilannetta sen edetessä kierrätysprosessissa. Sovellukseen on siis toteutettava selvästi määritellyt vaiheet jätteen kululle. Keskeisimpiä vaiheita ovat ilmoitusvaihe, jätteen varausvaihe, noutovaihe sekä valmis tila. Ilmoittajan on kyettävä seuraamaan ilmoittamiaan jätteitä ja jätteenkäsittelijän on nähtävä varaamiensa jätteiden tila sekä kaikki varattavissa olevat jätteet.



KUVIO 2. Sovelluksen käyttäjäryhmät

Jätteiden noudossa on huomattavasti tehokkaampaa yhdistää useiden jätteiden noudot yhteen. Jätteiden sijainneista voi muodostua kuvion 3 kaltaisia ryhmittymiä. Tätä edistääkseen jätteiden käsittelijälle on tarpeellista nähdä jätteiden sijainti helposti, jotta noutoon voidaan valita mahdollisimman paljon lähekkäin sijaitsevia jätteitä ja suunnitella optimaalinen keräysreitti. Kaikkein selkeimmin tämä tieto voidaan esittää kartalla, josta nähdään kaikki hakuehtojen kriteerit täyttävät jätteet alueella.



KUVIO 3. Jätteiden sijainnit voivat muodostaa ryhmiä

Sovelluksen käyttöliittymän täytyy toimia sujuvasti sovelluksen dataa hallitsevan serveri-osuuden kanssa. Sovellukseen tarvitsee siis määritellä toimivat sovellusrajapinnat, joiden kautta eri sovelluksen osat kykenevät jakamaan dataa. Digiloopissa rajapinta tulee toteuttaa REST (representational state transfer) -rajapintana.

3 KOMMUNIKOINTI

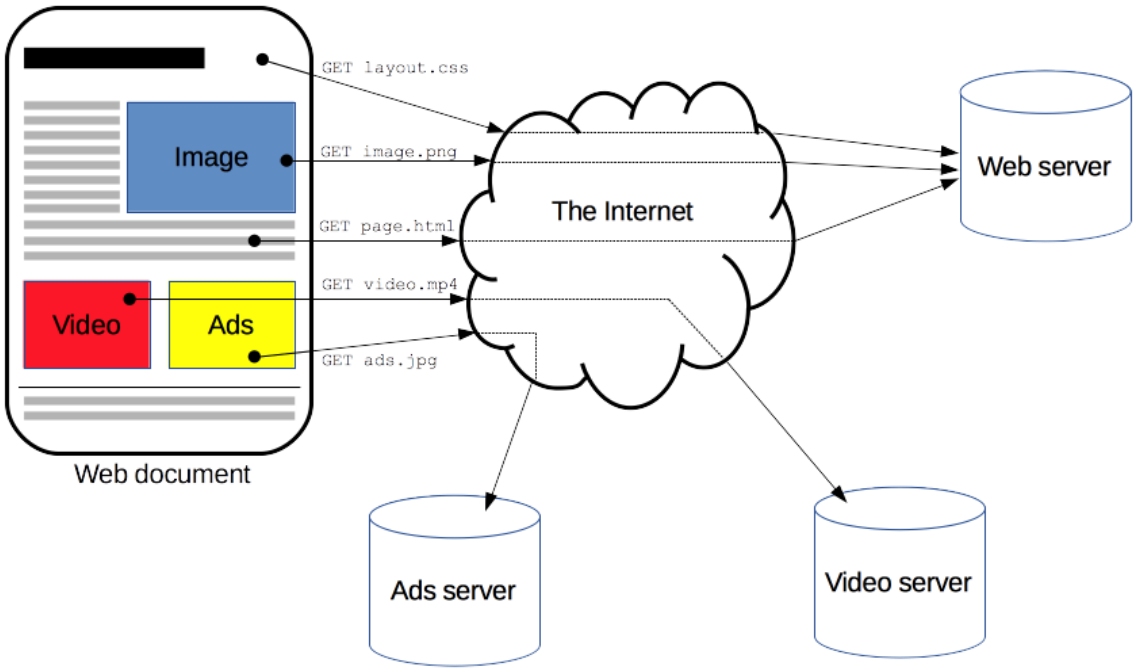
3.1 Verkkosivun kommunikointi

Verkkosivuston toiminta koostuu perinteisesti useasta eri tekijästä. Sivustoa käyttävää laitetta ja siinä suoritettavaa osuutta verkkosivusta sanotaan asiakkaaksi. Vastavuoroisesti toisessa päässä on palvelin, joka lähettää asiakkaan pyytämän verkkosivun, sekä hoitaa yhteydet esimerkiksi sivuston tietokantaan. (Mozilla 2019b.)

Asiakkaan ja palvelimen välinen kommunikointi tapahtuu tarkkaan määriteltyjen protokollien avulla. HTTP-protokolla varmistaa, että osapuolten lähettämät viestit ovat sellaisessa muodossa, että niitä kykenee vastapuoli ymmärtämään. TCP/IP-protokolla puolestaan määrittelee, miten osapuolten viestit kulkevat verkossa. (Mozilla 2019b.)

3.2 HTTP-protokolla

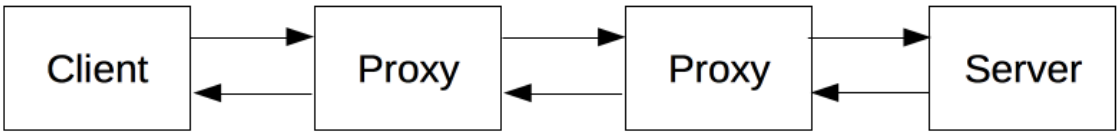
HTTP on protokolla, joka mahdollistaa resurssien, kuten HTML-dokumenttien, noudon. HTTP-pyynnöt lähtevät aina asiakkaalta palvelimelle, johon palvelin vastaa pyydetyllä resurssilla tai ongelmatilanteissa mahdollisella virhekoodilla. Kuviosta 4 nähdään, kuinka kokonainen HTML-dokumentti koostuu tyypillisesti useasta eri elementistä, jotka kaikki haetaan omilla HTTP-pyynnöillä (Mozilla 2019a.)



KUVIO 4. HTTP-protokollan avulla voidaan hakea sisältöä verkkosivulle (Mozilla 2019a.)

HTTP on asiakas-palvelin protokolla, eli kutsut lähtevät asiakkaalta ja niihin vastaa palvelin. Yleensä asiakkaana on verkkoselain, mutta se voi olla mikä vain, esimerkiksi verkkoa indeksoiva hakurobotti. Kommunikointi tapahtuu yksittäisinä viesteinä vaihtoehtoisen datavirran sijaan. (Mozilla 2019a.)

Asiakkaan ja palvelimen välissä kutsu voi edetä useamman eri välityspalvelimen kautta, kuten kuviossa 5 nähdään. Välityspalvelimet suorittavat kutsulle eri toimintoja, kuten toimivat välimuisteina tai yhdyskäytävinä. Välityspalvelimet saattavat muokata kutsua tai palvelimen vastausta tarpeen mukaan, mutta ne voivat myös ainoastaan välittää datan eteenpäin. (Mozilla 2019a.)



KUVIO 5. Asiakkaan ja palvelimen välillä voi olla useita välityspalvelimia (Mozilla 2019a.)

HTTP-protokolla on luonnostaan tilatonta. Kahden eri kutsu-vastaus parin välillä ei oletuksena ole yhteyttä. Istuntoja asiakkaan ja palvelimen välillä voidaan luoda HTTP-evästeiden avulla. Evästeet liitetään kutsuihin, jolloin palvelin kykenee tunnistamaan, mitkä kaikki kutsut ovat tulleet samalta asiakkaalta, ja kykenee ryhmittämään ne osaksi samaa sessiota. (Mozilla 2019a.)

HTTP-viestit kuljetetaan yhteyspohjaisen TCP-protokollan avulla. HTTP ei välttämättä vaadi yhteyttä, eli viestit voitaisiin myös kuljettaa yhteydettömän UDP-protokollan avulla. UDP-protokolla ei kuitenkaan ole yhtä luotettava kuin TCP-protokolla. (Mozilla 2019a.)

HTTP-viesteissä on omat formaatit kutsuille sekä vastauksille. Kutsut koostuvat aina vähintään kutsun metodista, haettavan kohteen polusta, HTTP-protokollan versiosta. Osa metodeista, kuten POST, voi lisäksi tarvita myös lähetettävän datan. Kutsuun voidaan myös liittää ylimääräisiä tietokenttiä, joita palvelin voi hyödyntää. Palvelimen vastaus koostuu palvelimen käyttämän HTTP-protokollan versiosta, kutsun onnistumisen tilakoodista, tilakoodia tarkentavasta tilaviestistä, sekä otsikoista. Vastauksessa voi olla mukana myös resurssi, jos kutsun tarkoituksena oli sellainen noutaa. (Mozilla 2019a.)

3.3 REST

REST on arkkitehtuurinen tyyli luoda standardoituja ohjelmistorajapintoja, joiden avulla järjestelmät kykenevät kommunikoimaan keskenään. Esimerkiksi kolmitasoarkkitehtuurillisessa ohjelmassa sisäistä kommunikointia varten palvelinpuolelle on luotu erinäisiä rajapintaosoitteita, joita voidaan kutsua sovelluksen käyttöliittymästä. (Codeacademy 2018.)

Ohjelmistorajapinnalla tarkoitetaan palvelinsovellukseen luotuja osoitteita. Palvelin kuuntelee rajapintaosoitteeseen tulevia kutsuja, suorittaa rajapintaan määritetyt funktiot sekä mahdollisesti palauttaa kutsujalle dataa. Rajapintoja voi käyttää sekä sovelluksen sisäiseen kommunikointiin, että mahdollistamaan sovelluksen käytön ulkopuolelta, esimerkiksi muista sovelluksista. (Gazarov 2016.)

REST järjestelmä on tilaton, eli kutsujan ja palvelimen ei tarvitse tietää toistensa tilasta tai rakenteesta mitään, jolloin sovelluksen rakenteesta saadaan modulaarisempi. REST kutsuun liitetään kaikki tarvittava tieto, jotta palvelin kykenee vastaamaan. Modulaarinen rakenne johtaa helpommin ylläpidettävään sovellukseen, jossa voidaan esimerkiksi muokata käyttöliittymää tai lisätä useampia eri käyttöliittymiä ilman, että rajapintaa tai palvelinosaa tarvitsee muuttaa. (Codeacademy 2018.)

REST järjestelmässä osapuolet eivät ole tietoisia toistensa datan tilasta. Kommunikaatio toimii, vaikka osapuolten aikaisemmat kutsut eivät olisi tiedossa. Jokainen kutsu on siis yksittäinen tapahtuma, johon ei vaikuta mikään kutsun ulkopuolinen tekijä. (Codeacademy 2018.)

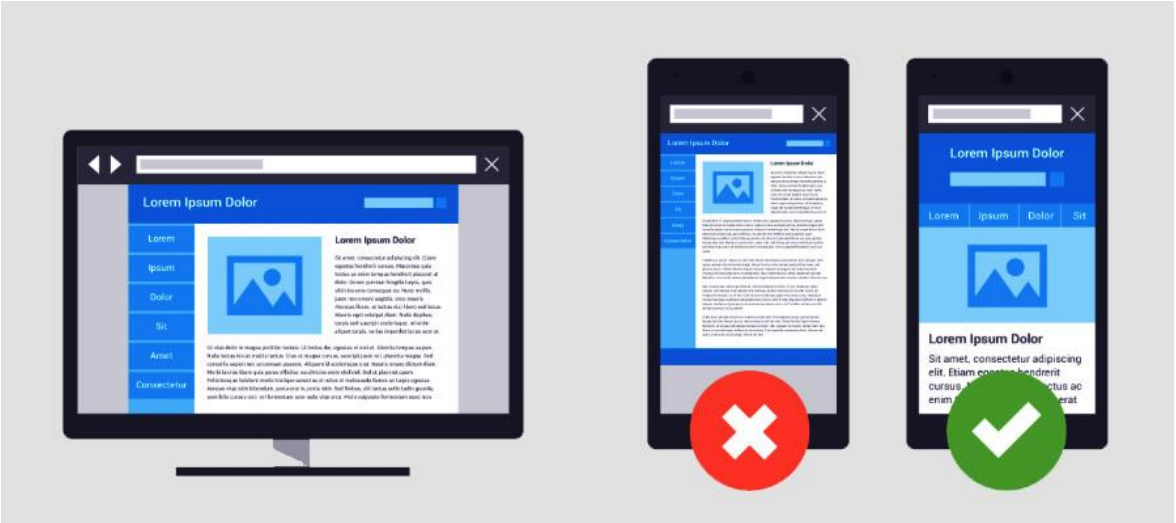
REST käyttää HTTP-metodeja kuten GET, POST, PUT ja DELETE vastaamaan CRUD (create, retrieve, update, delete) -operaatioita HTTP-kutsuissa. Eri metodeilla on omat käyttötarkoituksensa, esimerkiksi GET-metodilla voidaan hakea dataa ja POST-metodilla voidaan lähettää kutsussa dataa. (Codeacademy 2018.)

Palvelin vastaa kutsuihin aina vähintään tilakoodilla, josta selviää, onnistuiko REST-kutsun suorittaminen. Koodit jaotellaan viiteen ryhmään: tiedote, onnistuminen, uudelleenohjaus, käyttöliittymävirhe sekä palvelinvirhe.

4 RESPONSIIVISUUS WWW-PALVELUISSA

4.1 Responsiivisuus

Verkkosivustoa voidaan käyttää hyvin erilaisilla laitteilla. Vaihtuvia tekijöitä on muun muassa ruudun koko, asento sekä käytössä oleva verkkoselain. Kiinteästi rakennettu sivusto voi näyttää huomattavasti huonommalta, mikäli sitä käytetään erilaisissa olosuhteissa kuin mihin se oli alkujaan suunniteltu. Kiinteästi rakennettu sivusto näyttää kuvion 6 tapaan vain kutistetulta, kun sitä käytetään mobiililaitteella. Responsiivinen toteutus luo mobiililaitteelle omanlaisensa muotoilun.



KUVIO 6. Hyvä ja huono responsiivinen toteutus mobiilille (LocalOffers 2016.)

Responsiivisuus on tärkeää huomioida verkkosovelluksissa, joiden täytyy toimia käyttölaitteesta tai ympäristöstä riippumatta. Responsiivisuudella tarkoitetaan sovelluksen rakenteiden sekä sisällön dynaamista sopeutumista erilaisiin näyttökokoihin ja -asentoihin (Schade 2014).

Responsiivinen verkkosivusto saadaan useimmiten luotua CSS-tyylisääntöjen avulla. Puhtaasti visuaalisesta näkökulmasta, responsiivisuus voidaan toteuttaa hyödyntämällä kolmea tekniikkaa:

1. joustavaa, ruudukkopohjaista rakennetta
2. joustavaa mediaa
3. mediakyselyitä (Marcotte 2011).

Joustavilla rakenteilla sekä medialla sisältö sopeutuu hyvin pieniin muutoksiin ruudun koossa, kun taas mediakyselyillä voidaan erilaisille laiteko'oille luoda omat CSS-säännöt, jolloin sama sisältö saadaan näyttämään hyvältä hyvinkin erikokoisilla laitteilla.

4.1.1 Joustava rakenne

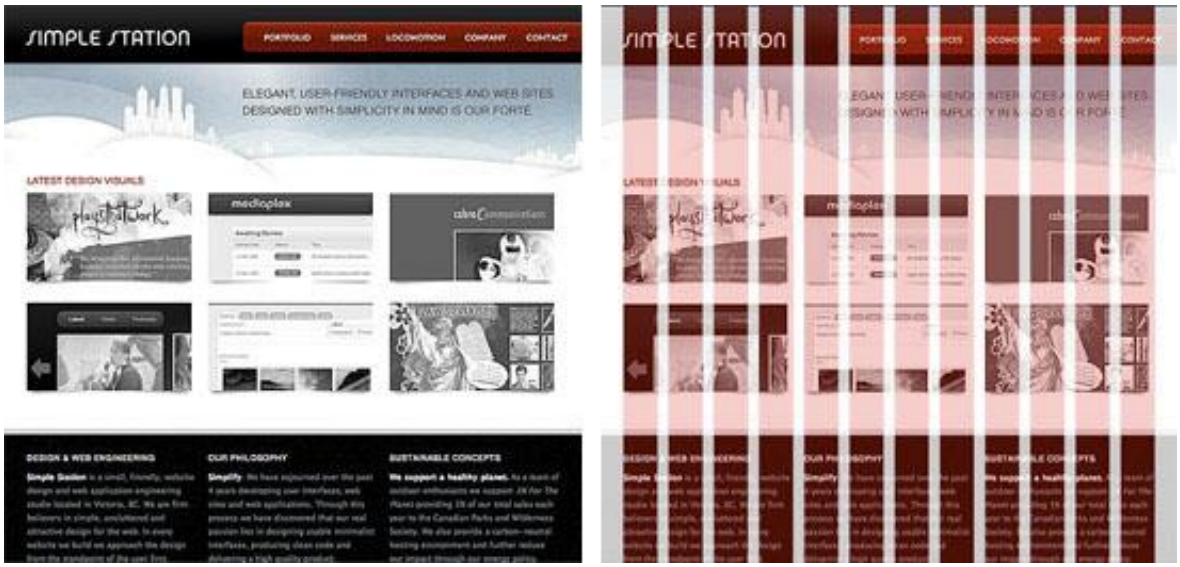
Sivuston joustavuuden kannalta keskeisin haittatekijä on pikseleinä määritellyt elementtien koot. Responsiivisuutta ajatellen tärkeintä on saada elementit skaalautumaan ruudun mukaisesti, eli elementtien koko on syytä määritellä suhteessa muihin elementteihin. (Marcotte 2011.)

Elementtien koon mukautuvuus voidaan helpoiten tehdä käyttämällä CSS:n suhteellisia yksiköitä, jotka on listattu taulukossa 1. Responsiivisuus elementeille voidaan saavuttaa parhaiten yksiköillä, jotka ovat riippuvaisia suoraan tai välillisesti selainikkunan koosta. Fonttikokojen responsiivisuudessa voidaan käyttää fonttikokoon suhteutettuja yksiköitä, jolloin riittää, että ylimmän tason fontti on määritelty mediakyselyillä.

TAULUKKO 1. CSS:n suhteelliset pituusyksiköt (W3Schools 2018a.)

Yksikkö	**Kuvaus**
em	Suhteessa elementin fonttikokoon
ex	Suhteessa elementin fontin x-korkeuteen
ch	Suhteessa fontin '0' -numeron leveyteen
rem	Suhteessa selaimen fonttikokoon
vw	Suhteessa prosenttiin selainikkunan leveydestä
vh	Suhteessa prosenttiin selainikkunan korkeudesta
vmin	Suhteessa prosenttiin selainikkunan pienemmästä ulottuvuudesta
vmax	Suhteessa prosenttiin selainikkunan isommasta ulottuvuudesta
%	Suhteessa ylemmän tason elementin kokoon

Ruudukkopohjaisessa rakenteessa sivuston rakenne jaetaan tasaisesti sarakkeisiin. Lisäksi sarakkeiden väliin jätetään tilaa marginaaleille. Elementtien sijoittelu sivulle perustuu jaettuihin sarakkeisiin, kuten esimerkkikuviossa 7. (Craig 2010.)



KUVIO 7. Sarakkeina rakennettu sivusto (Craig 2010.)

Sarakkeisiin jaetut elementit voidaan määritellä responsiivisissa yksiköissä, kuten prosenteissa. Alustava suunnitelma on tehty perustuen johonkin näyttökokoon, jolloin jokaiselle sarakkeelle voidaan määritellä leveys pikseleinä kyseisessä näyttökoossa. Prosentteja käytettäessä vertailukohde on aina ylemmän tason elementti, joten prosenttikoko saadaan jakamalla elementin haluttu pikselileveys ylemmän elementin pikselileveydellä. Täten saadaan elementtien välisistä kokosuhteista samat riippumatta siitä, mikä näytön koko on. (Marcotte 2011.)

Pienillä näyttöko'oilla elementeillä on hyvin vähän tilaa, joten etenkin reunojen koko kannattaa suhteuttaa pienemmäksi. Reunojen responsiivisuus toteutuu hyvin samaan tapaan kuin itse elementillä. Marginaaleissa prosentuaalinen osuus lasketaan ylemmän elementin leveydestä, mutta sisennyksissä prosentuaalinen osuus lasketaan elementin itsensä leveydestä (Marcotte 2011).

4.1.2 Joustava media

Toisin kuin helposti erilaisiin säiliöihin mukautuva teksti, joustamaton media voi helposti rikkoa sivuston responsiivisuuden. Sivustolla esiintyvät kuvat ja muu media yrittävät käyttää täyttä kokoaan riippumatta niitä ympäröivien elementtien koosta, ellei toisin määritellä. Tästä johtuen ilman erillisiä CSS-sääntöjä ne voivat helposti pilata muuten responsiivisen sivun joustavuuden.

’Max-width: 100%’ -CSS-säännöllä voidaan rajoittaa media korkeintaan ympäröivän elementin kokoiseksi, mikäli median oma leveys ylittää ympäröivän elementin leveyden (Marcotte 2011). Media saa samalla samat responsiiviset ominaisuudet kuin ympäröivä elementtikin, jolloin se kykenee mukautumaan ympäröivän säiliön kanssa ruudun koon muutoksiin. Median koko voi kuitenkin olla myös pienempi, mikäli sen alkuperäinen resoluution on pienempi kuin mitä ympäröivällä säiliöllä.

Toinen vaihtoehto on antaa medialle prosentuaalinen ’width’-sääntö, jolla median leveys säilyy suhteellisena osana ympäröivään säiliöön nähden. Erona ’max-width’-sääntöön ’width’-sääntö saattaa myös suurentaa mediaa sen alkuperäisestä resoluutiosta, mikäli ’width’-sääntö antaisi sille alkuperäistä suuremman resoluution. ’Width’ ja ’max-width’ -sääntöjä voidaan myös molempia hyödyntää, määrittämällä joustava ’width’ koko sekä kiinteä ’max-width’ maksimikoko. (W3schools 2018a.)

Joustavassa mediassa tarvitsee yleensä määrittää ainoastaan median leveys, sillä median korkeuden vakiosääntö määrittää sen automaattiseksi. Tämä tarkoittaa sitä, että media laskee korkeuden leveydestä, pyrkien aina säilyttämään kuvasuhteensa samana.

4.1.3 Mediakyselyt

Joustavilla elementeillä ja medialla saadaan sivuston sisältö pysymään samassa suhteessa näyttöön verrattaessa. Sivuston rakenne on kuitenkin suunniteltu ensisijaisesti tietyn kokoiselle sekä -asentoiselle näytölle. Elementtien suhteellisuus ei pelkästään riitä optimoimaan sivustoa etenkään erittäin suurille tai pienille näyttöko’oille.

Suunniteltua suuremmilla näyttöko’oilla sivuston sisältö on helposti liian leveällä. Etenkin tekstipainotteisen sisällön luettavuus kärsii, mikäli rivipituus kasvaa liikaa. Helposti luettavassa tekstissä pitäisi olla noin 45-75 merkkiä riviä kohden. Ratkaisuina liian leveälle tekstille on tekstin jakaminen sarakkeisiin, rivivälin kasvattaminen tai sisällön keskittäminen kapeammalle alueelle. Mikäli sisältöä kavennetaan, saadaan vaihtoehdoksi tuoda muuta sisältöä samaan tilaan, esimerkiksi siirtämällä alatunniste sivun reunaan. Liiallinen sisältö voi isollakin ruudulla olla sekavan oloinen. (Yates 2013.)

Pienillä ruuduilla ongelma on päinvastainen: tilaa on rajallisesti, joten sisältöä voidaan näyttää kerralla vähemmän. Mikäli pienille ruuduille sisällytetään kaikki sama sisältö kuin keskikokoisille ja isoille ruuduille, tulee sivusta näyttökokoon suhteutettuna hyvin pitkä, jolloin käyttäjän on huomattavasti vaikeampaa löytää sisältö, jota hän oli hakemassa.

Myös valikkorakenteet vievät pienillä näytöillä ison osan ruututilasta. Tämän takia pienillä ruuduilla avoimet valikot piilotetaan usein napin taakse, jolloin valikkoelementit eivät ole

näkyvillä, ennen kuin niitä tarvitaan. Aloitusruudussa on hyvä näyttää vain kaikkein olennaisimmat elementit, toissijaiset elementit on hyvä jättää valikkojen taakse tai alkunäkymän ulkopuolelle (Gove 2018).

Erilaisille laitteille saadaan räätälöityä omat CSS-säännöt mediakyselyillä. Mediakyselyillä tarkoitetaan sarjaa ehtoja, joiden avulla määritellään mitä CSS-sääntöjä käytetään. Ehtoina voidaan käyttää muun muassa selainikkunan sekä laitteen leveyttä ja korkeutta, laitteen orientaatiota, resoluutiota sekä mediatyyppejä. Mediatyypit kertovat, millaisella laitteella sisältöä käytetään. Esimerkkejä mediatyypeistä on muun muassa ’print’, eli tulostusnäkymä sekä ’screen’ eli perinteinen ruutunäkymä. (W3Schools 2018b.)

Mobiililaitteita varten on luotu mediatyyppi ’handheld’, mutta suurin osa mobiililaitteiden selaimista luokittelee itsensä ’screen’ -tyyppiseksi. Mediatyypeissä ei myöskään ole erillisiä tyyppejä taulutietokoneille, jotka ovat kokonsa puolesta puhelinten ja tietokoneiden välissä. Tämä voidaan huomioida lisäämällä ehtoihin mediatyypin lisäksi selainikkunan leveysehto. Kuviossa 8 on esimerkkejä erilaisista mediakysely-yhdistelmistä. (Marcotte 2011.)

```
/* big screens */
@media screen and (min-width: 1024px){
    body { width: 80%; }
}
/* mobile */
@media screen and (max-width: 480px){
    body { width: 100%; }
}
/* mobile in landscape */
@media screen and (max-width: 480px) and (orientation: landscape){
    body { width: 70%; }
}
/* printing */
@media print {
    body { width: 65%; }
}
```

KUVIO 8. Esimerkki mediakyselyistä

Mediakyselyt voidaan toteuttaa sekä yksittäisissä säännöissä, luomalla eri tilanteille sopivat säännöt, että luomalla esimerkiksi eri näyttöko’oille erilliset CSS-tiedostot, joista sopiva ladataan CSS-tiedoston tuontivaiheessa. Kyse on lähinnä siitä, halutaanko kaikki tyylit valita mediakyselyillä, vai tehdäänkö kyselyt ainoastaan tiettyihin tyyliominaisuuksiin.

Älypuhelinten alkuvuosina ei niitä usein otettu huomioon verkkosivujen suunnittelussa. Koska sivustot olivat hitaita sopeutumaan uusiin käyttäjiin, puhelinten selaimet sopeutuivat käyttämään verkkosivujen tietokoneversioita. Apple loi ensimmäisenä selaimen, joka näytti sivustot 960 pikseliä leveinä ja kutisti sen iPhonen 320 pikselin näyttöön. (Marcotte 2011.)

Applen selain aiheutti ongelman mediakyselyille, sillä sivustot luulevat 320 pikseliä leveitä näyttöjä 960 pikseliä leveiksi, jolloin käyttöön tulee väärät mediakyselyt. Puhelinten selaimien toiminta voidaan yliajaa ViewPort-metatagilla, jolloin selaimen koko voidaan määrittää vapaasti. Kuviossa 9 olevassa metatagissa määritellään selaimen leveys vastaamaan laitteen leveyttä sekä alustava suurennus yksinkertaiseksi.

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

KUVIO 9. ViewPort-metatietotagi (W3Schools 2018c.)

Mediakyselyissä määritellään erilaisia CSS-sääntöjä erilaisille laiteko'oille, mutta jokin laitetyyppi on aina valittava ensisijaiseksi. CSS-tiedostoissa aikaisemmin esiintyvät mediakyselyt ladataan aina ennen myöhemmin esiintyviä. Tästä seurauksena myöhemmin esiintyvät ehdot joutuvat lataamaan enemmän tietoa ennen kuin sivusto on valmis näytettäväksi.

Mobile First -ajattelumallissa ladataan aina ensin mobiilille kohdennetut mediakyselyt. Mobiililaitteilla on tietokoneisiin verrattuna usein heikompi suorituskyky sekä hitaampi verkkoyhteys. Lataamalla niille tarpeelliset tiedostot ensin, sivuston käyttönopeus paranee mobiililaitteilla, joilla se on kriittistä. (Poteet 2015.)

Mobile First -ajatusmallia voidaan myös laajentaa sivuston ominaisuuksien suunnitteluun. Mobiilinäkymät verkkosivuista on yleensä yksinkertaistettuja versioita sivuston kokonaisversiosta. Mikäli kokonaisversio suunnitellaan ensin, ongelmaksi saattaa muodostua sen karsiminen mobiiliyhteensopivaksi. Suunnittelemalla sivusto ensin mobiilin näkökulmasta, ja luomalla kokonaisversio sen pohjalta, saadaan välttämättömät ominaisuudet paremmin toimimaan kaikilla laitteilla.

4.2 Material UI

Material UI on avoimen lähdekoodin käyttöliittymäviitekehysprojekti, jossa Reactin elementit hyödyntävät Googlen Material Designia. Se on LESS CSS -esiprosessorin päälle

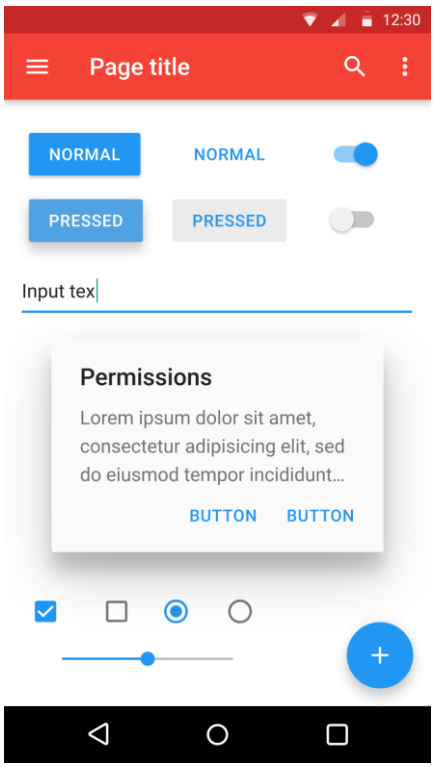
rakennettu, ja se mahdollistaa Material Designin käytön React elementteinä. (Foddai 2016.)

4.2.1 Material Design

Material Design on Googlen kehittämä muotokieli. Muotokielellä tarkoitetaan yleistä tyylisuunnitelmaa tai ohjeistusta, jolla tuotteelle saadaan yhtenäinen ulkonäkö. Siinä voidaan määritellä tuotteelle esimerkiksi väriskaaloja, muotoja, asetteluja tai tekstuureja.

Material Design on käytössä suurimmassa osassa Googlen sovelluksista, kuten Gmailissa, YouTubessa sekä Google Drivessä. Android sovelluksissa Material Designia voidaan hyödyntää käyttämällä Androidin andoid.support.design -kirjastoa (Android 2018a).

Material Design soveltuu hyvin perinteisempiin verkkosivustoihin ja -sovelluksiin, jotka hyödyntävät peruselementtejä, kuten nappeja ja valikoita. Sitä ei ole suunniteltu käytettäväksi esimerkiksi peleissä tai erikoisissa sovellusratkaisuissa. Koska Material Design on käytössä useimmissa Googlen sovelluksissa, sillä toteutetut sovellutukset saavat helposti saman Googleen assosioitavan ilmeen. Kuviossa 10 on esitelty erilaisia Material Designin elementtejä mobiilisovelluksessa. (Woodhead 2018.)

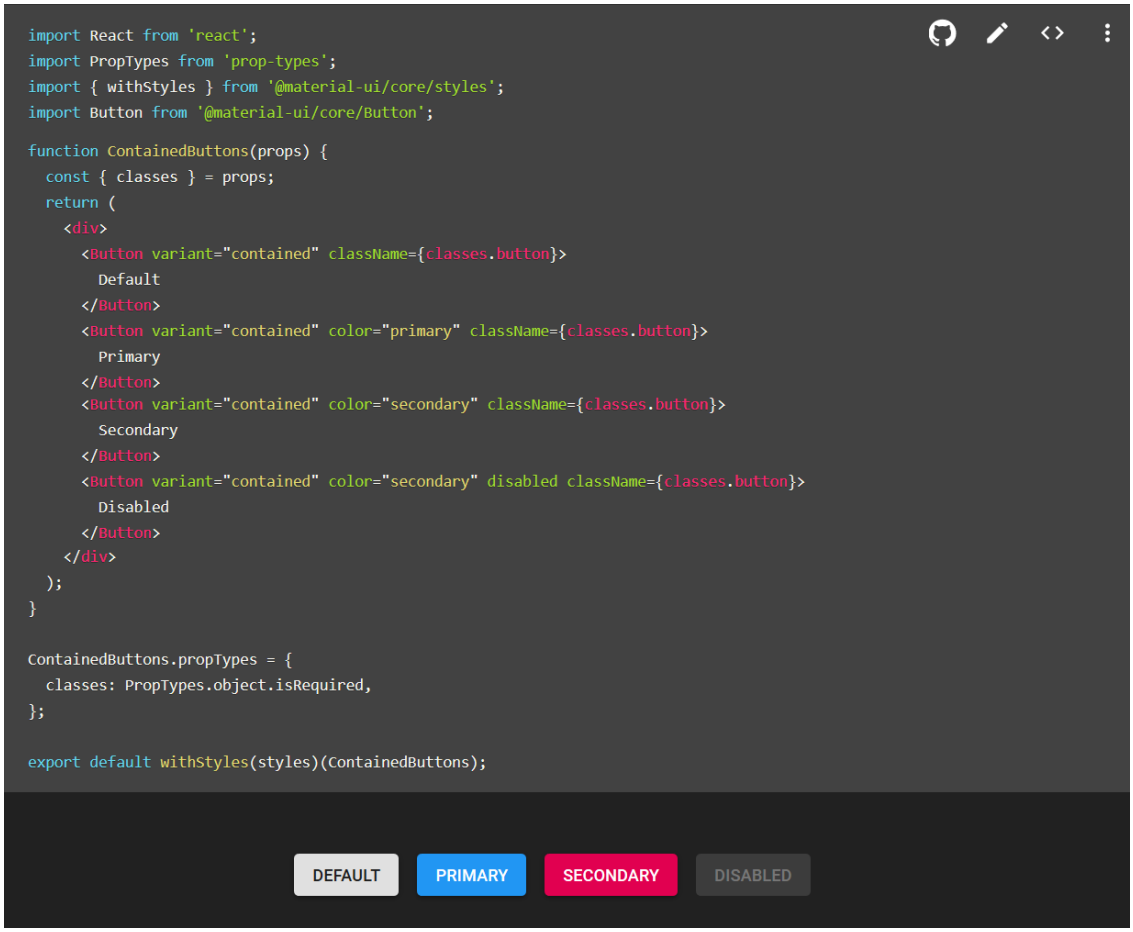


KUVIO 10. Material Designin elementtejä mobiilisovelluksessa (Wikimedia Commons 2015.)

Koska Material Design on pohjimmiltaan tyyliohjeistus, voidaan sitä toteuttaa hyvin erilaisilla tavoilla sovelluksissa. Yleisimmin sitä käytetään sovelluksissa kolmannen osapuolen kirjastojen, kuten Material UI:n avulla. Tyypillisimmin kirjastot toteuttavat Material Designia komponenttitasolla, eli ne tarjoavat Material Designissa määriteltyjä elementtejä käyttövalmiina.

4.2.2 Material UI React komponentteina

Material UI on komponenttitasolla toimiva Material Design kirjasto Reactille. Se siis tarjoaa Material Designin elementtejä valmiina React-komponentteina. Kuviossa 11 nähdään, miten Material UI:n button -komponenttia voidaan käyttää React-sovelluksessa. (Material-UI 2018a.)



KUVIO 11. Material UI button -komponentin käyttö Reactissa (Material-UI 2018c.)

Kaikki projektiin määritellyt Material UI -komponentit käyttävät projektinlaajuista Material UI:n teemaa, ellei sitä ole joillekin komponenteille yliajettu erillisellä CSS määrityksellä. Teema voidaan määritellä projektille vapaasti, ja sen avulla saadaan yhtenäinen ulkonäkö koko sovellukselle ilman monimutkaisia CSS-sääntöjä. (Material-UI 2018f.)

Yksittäisen elementin tyyli voi poiketa projektin yleisestä teemasta helpoiten yliajamalla sen CSS-ehdot. Sääntöjä voidaan yliajaa esimerkiksi luokkakohtaisesti tuomalla luokka withStyles -funktion avulla, tai elementin määrittelyssä niin sanotulla rivityylillä. (Material-UI 2018f.)

4.2.3 Responsiivisuus Material-UI:ssa

Material-UI:n avulla voidaan toteuttaa Material Designia sovelluksessa. Vaikka Material Designin keskeisin pääpaino ei ole responsiivisessa suunnittelussa, on se silti otettu huomioon. Täten myös Material UI:ssa on työkaluja responsiivisen sovelluksen toteutukseen.

Material UI tarjoaa responsiivisuuden toteutukseen komponentteja, joiden avulla Material Designin responsiivista ajattelua voidaan helpommin käyttää. Keskeisimpänä on Grid -komponentti sekä sen hyödyntämät murtumispisteet. Lisäksi Hidden -komponentilla voidaan luoda ehdollisia elementtejä, joita ei näytetä kaikille laiteko'oille.

Material UI:n Grid -komponentti mahdollistaa responsiivisen rakenteen hyödyntämisen sovelluksessa. Se käyttää Material Designin 12:n sarakkeen rakennetta, jossa sivuston rakenne jaetaan joustaviin sarakkeisiin. Sarakkeiden leveys määrittyy prosentuaalisesti ympäröivän säiliön mukaan. (Material-UI 2018d.)

Jokaiselle elementille määritetään, kuinka montaa saraketta kahdestatoista sen tulee käyttää. Jos esimerkiksi halutaan kaksi yhtä leveää elementtiä vierekkäin, voidaan molemmille määrittää leveydeksi kuusi saraketta. (Material-UI 2018d.)

Grid -komponenteille määritellään usein myös murtumapisteet, eli mediakyselyillä toteutetut ehdolliset rakenteet erikokoisille ruuduille. Material UI:ssa murtumapisteet on ennalta määritelty taulukon 2 mukaisesti, mutta niitä voidaan muokata projektikohtaisesti. Murtumapisteillä voidaan esimerkiksi laittaa normaalisti vierekkäin olevia elementtejä allekkain, kun sivustoa käytetään mobiililaitteilla, kuten kuviossa 12 on näytetty. (Material-UI 2018b.)

TAULUKKO 2. Material UI:n murtumapisteet (Material-UI 2018b.)

Nimi	**Pikselialue**
xs	>0 px
sm	>600 px
md	>960 px
lg	>1280 px
xl	>1920 px

KUVIO 12. Material UI:n responsiivinen ruudukko keskikokoisella ja pienellä näytöllä (Material-UI 2018b.)

Hidden -elementillä voidaan luoda sisältöä, joka näytetään vain joillakin näyttöko'oilla. Hidden -elementeille määritellään murtumispisteet, jolloin saadaan elementti pois näkyvistä, kun näyttökoko on suurempi tai pienempi kuin murtumispiste. Esimerkiksi 'smUp' -määritelmällä elementti kätketään, jos näyttökoko on 'small' tai suurempi. Vastaavasti 'mdDown' -ehdolla elementti kätketään, jos näyttökoko on 'medium' tai pienempi. Kolmantena ehtona on 'only', jolloin elementti kätketään vain, mikäli näyttö on ehdon kokoinen. (Material-UI 2018e.)

Hidden on hyödyllinen työkalu karsimaan epäoleellisia elementtejä ja ominaisuuksia pois, kun sivustoa käytetään pienellä laitteella. Pienillä näyttöko'oilla sisällön merkitys korostuu, sillä sisältöä mahtuu vähemmän. Tästä syystä on parempi piilottaa vähemmän merkittäviä elementtejä, jolloin käyttäjän on helpompi keskittyä olennaiseen.

5 APACHE CORDOVA

5.1 Toimintaperiaate

Apache Cordova on Apache Software Foundationin avoimen lähdekoodin viitekehys, jonka avulla voidaan luoda iOS, Android sekä Windows -alustoille mobiilisovelluksia. Se luo verkkosovelluksen ympärille säiliön, jonka kautta sovellusta käytetään. Säiliö mahdollistaa sovelluksen käytön mobiililaitteessa, sekä laiteen palveluiden käytön. (Apache Cordova 2018d.)

PhoneGap on Cordova projektin alkuperäinen nimi. PhoneGapin kehitti Nitobi vuonna 2009. Adobe osti Nitobin vuonna 2011, jolloin projektin koodikanta lahjoitettiin Apache Software Foundationille. Nykyään PhoneGap on Adoben ylläpitämä versio Cordovasta. (Ancheta 2016.)

Cordovan käyttöliittymä on verkkonäkymä. Se käyttää sovelluksen koodikantaa sellaisenaan, eikä käännä sitä laitteelle sopivalle kielelle. Sen sijaan Cordova luo sovelluksen ympärille käyttöjärjestelmäkohtaisen säiliön, joka näyttää varsinaista sovellusta verkkonäkymässä. (Ancheta 2016.)

Cordova-sovellukset ovat niin kutsuttuja hybridisovelluksia. Hybridisovelluksella tarkoitetaan sovellusta, jossa verkkosivusto muutetaan mobiilisovellukseksi. Hybridisovellukselle vaihtoehtoja ovat verkkosovellukset ja natiivisovellukset. Verkkosovellus selaimella suoritettava sovellus, joka sijaitsee verkossa. Natiivisovellus suoritetaan mobiililaitteella ja se on kirjoitettu mobiililaitteen natiivilla kielellä. (Rouse 2011.)

Cordovalla rakennetut sovellukset eivät ole yhtä nopeita kuin natiivisovellukset, johtuen sovelluksen verkkopohjaisesta rakenteesta. Cordova ei siis ole hyvä vaihtoehto, mikäli sovellus on muutenkin raskas. Cordova-sovellusten hyviä puolia on mm. helppo päivitettävyys. Cordova-sovelluksia on helppo tehdä, sillä ohjelmoijan tarvitsee osata kehittää ainoastaan verkkosovelluksia sen sijaan, että tarvitsisi osata kehittää kaikille eri mobiilialustoille, joihin sovellusta aiotaan jakaa.

5.2 Asennus ja käyttö

Cordova on jaossa npm (node packet manager) -pakettina. Se voidaan asentaa joko käyttöjärjestelmälaajuisena sovelluksena tai vain yksittäiselle projektille. Cordovaa varten on hyvä asentaa myös versionhallinta, sillä osa lisäosista on jaossa vain palvelun kautta. (Apache Cordova 2018 c).

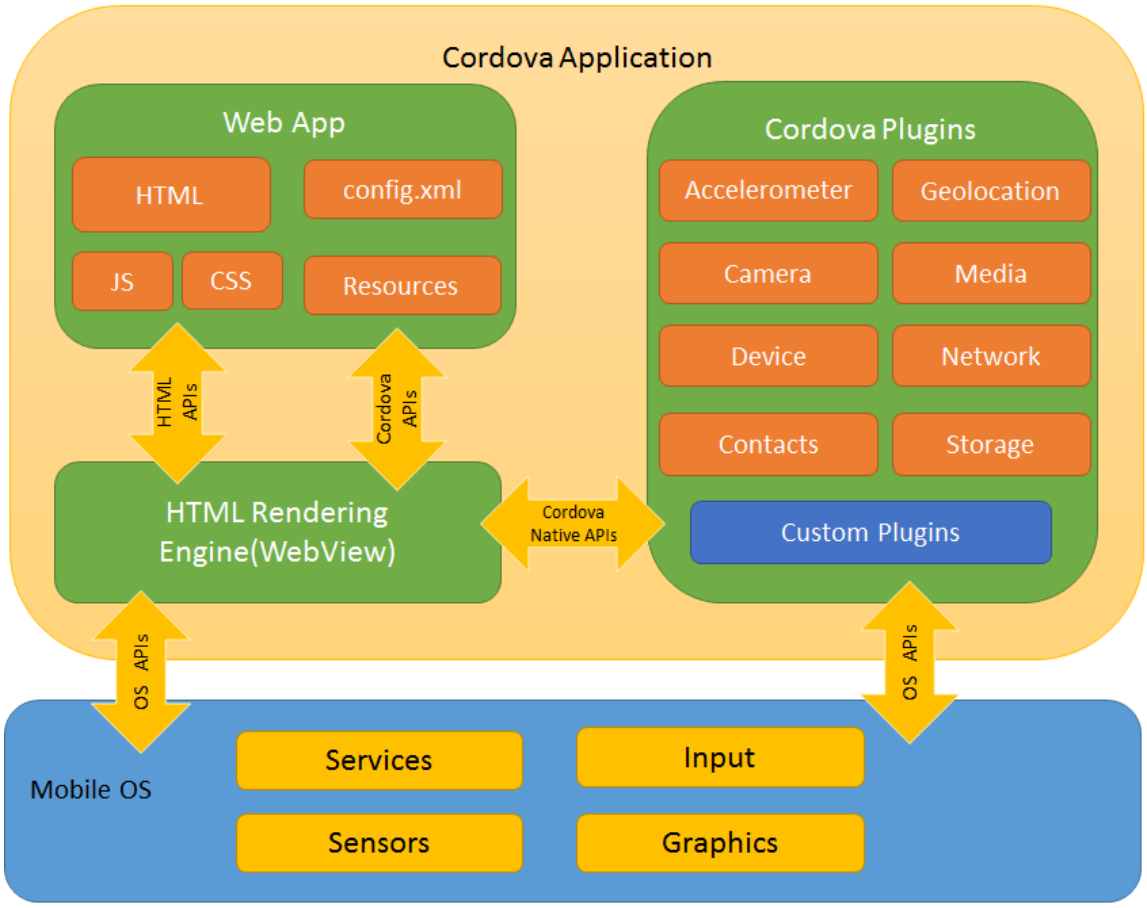
Cordova toimii täysin komentokehotteesta. Sen avulla voidaan mm. luoda uusia Cordova projekteja, lisätä sovelluksen tukemia kehitysalustoja, hallita lisäosia sekä rakentaa sovelluksesta asennuspaketti. Alustojen SDK:n (software development kit) mukana tulee usein myös välineet, joita Cordova voi käyttää sovelluksen emulointiin. (Apache Cordova 2018c.)

Verkkosovelluspohjasta voidaan luoda Cordova-sovellus ilman rakenteellisia muutoksia varsinaiseen sovellukseen. Cordova lisää ainoastaan tarvittavat tiedostot, joilla verkkosovellusta voidaan käyttää mobiilisovelluksena. Tämän takia muunnos verkkosovelluksesta hybridimobiilisovellukseksi voidaan tehdä missä tahansa kehityksen vaiheessa, esimeriksi jo valmiista verkkosovelluksesta.

5.3 Arkkitehtuuri ja lisäosat

Cordova-sovellusta käynnistettäessä Cordova lataa ensin sovelluksen aloitussivun, yleisimmin index.html:n. Sen jälkeen Cordova käynnistää sovelluksen WebView renderöintimoottorin. WebView on toteutettu kaikilla suurimmilla puhelinalustoilla: Androidilla on android.webkit.WebView luokka, iOS:illa on UIKit viitekehyksessä UIWebView luokka ja Windows puhelimilla WebView-luokka on osa Windows.UI.Xaml.Controls luokkia. (Saleh 2014.)

Cordovan WebView esittää sovellusta alustan oman WebView-järjestelmän kautta. Käyttöjärjestelmän kaikkien muiden ominaisuuksien käyttö tapahtuu Cordovan lisäosien kautta, jotka kutsuvat järjestelmästä vastaavia ominaisuuksia. Cordova-sovelluksen rakenne on esitetty kuviossa 13.

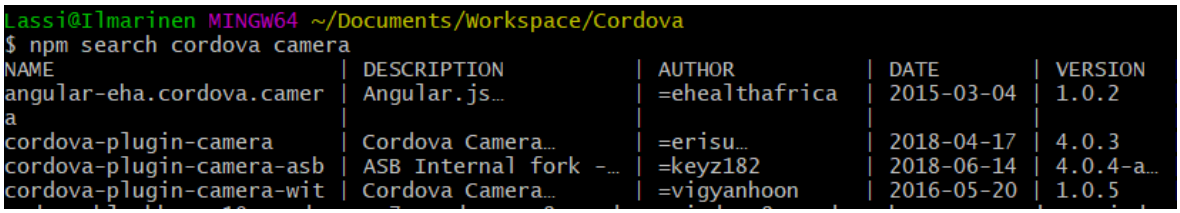


KUVIO 13. Cordova-sovelluksen arkkitehtuurillinen rakenne (Apache Cordova 2018d.)

Cordovan mukana tulee kokoelma lisäosia, joiden JavaScript-rajapintaa voidaan sovelluksesta kutsua. Lisäosat muuntavat kutsun käyttöjärjestelmäkohtaisesti natiiville kielelle, ja lähettävät sen eteenpäin järjestelmän rajapintoihin (Saleh 2014). Lisäosat toimivat siis eräänlaisina muuntimina, joilla laitteen ominaisuuksia voidaan kutsua verkkosovelluksesta lähetetyillä JavaScript-kutsuilla.

Cordova-sovellukseen voidaan lisätä myös kolmannen osapuolen lisäosia. Useimmat lisäosat ovat saatavilla joko npm-pakettina tai ne voivat olla jaossa Git-hakemistoissa. Mikäli tarvittavaa lisäosaa ei löydy valmiina Cordovasta tai kolmannen osapuolen tarjoamana, niin niitä voidaan luoda vapaasti tarpeen mukaan.

Lisäosat koostuvat kolmesta osasta: määritelmästä, sovelluksen JavaScript-rajapinnasta sekä eri alustojen natiivista liitännästä. Määritelmällä tarkoitetaan xml-tiedostoa, jossa määritetään lisäosan tukemat alustat sekä perustiedot lisäosalle. JavaScript-rajapinta vastaanottaa sovelluksesta tulleet kutsut ja natiivi liitäntä kutsuu laiteesta tarvittua ominaisuutta. (Saleh 2014.)



KUVIO 14. NPM-haku Cordovaan saatavista kameralisäosista

Lisäosien lisääminen sovellukseen tapahtuu helpoiten komentoriviltä. Sopivaa lisäosaa voi ensin etsiä, esimerkiksi “cordova plugin search camera” -komento avaa löydetyt kameralisäosat selaimeen. Mikäli tulokset halutaan komentoriville, NPM haku “npm search cordova camera” toimii paremmin. Kuviossa 14 on komentorivillä kameralisäosien haun ensimmäiset tulokset. (Apache Cordova 2018c.)

Löydetty lisäosa voidaan asentaa “cordova plugin add” -komennolla. Esimerkiksi Cordovan oman kameralisäosan asennus tapahtuu “cordova plugin add cordova-plugin-camera” -komennolla. Tämän jälkeen asennetut lisäosat voidaan tarkistaa “cordova plugin list” -komennolla. Asennettua lisäosaa voidaan asennuksen jälkeen käyttää suoraan. (Apache Cordova 2018c.)

Cordovassa voidaan myös tehdä sovellukseen alustakohtaisia eroavuuksia. Alustalle luodaan yksilöllisiä vastineita html, css ja JavaScript -tiedostoista, jotka korvaavat sovelluksesta vastineensa. Näitä korvaavia tiedostoja kutsutaan mergeiksi. (Apache Cordova 2018b.)

```
myapp/
|-- config.xml
|-- hooks/
|-- merges/
| | |-- android/
| | |-- windows/
| | |-- ios/
|-- www/
|-- platforms/
| |-- android/
| |-- windows/
| |-- ios/
|-- plugins/
  |--cordova-plugin-camera/
```

KUVIO 15. Cordova-sovelluksen tiedostorakenne (Apache Cordova 2018b.)

Cordova sovellusten kansiorakenne on kuvion 15 mukainen. Platforms kansiosta löytyy kullekin alustalle vaaditut alustakohtaiset koodit. Www-kansiossa on varsinainen sovellus, ja merges kansiossa siihen alustakohtaiset korvaavat tiedostot. (Apache Cordova 2018b.)

5.4 Rajapinnat ja puhelimen oikeudet

Jotta sovellus voi käyttää puhelimen ominaisuuksia tai käyttäjän dataa, sen on pyydettävä siihen lupa käyttäjältä. Näitä oikeuksia varten esimerkiksi Androidissa sovellus listaa tarvitsemansa oikeudet sovelluksen manifestissa. Sovelluksen manifesti on tiedosto, joka määrittelee sovelluksen tarvitsemat oikeudet. Oletuksena mikään sovellus ei voi suorittaa toimintoja, jotka voivat haitallisesti vaikuttaa muihin sovelluksiin, käyttöjärjestelmään tai käyttäjään. (Android 2018b.)

Cordovassa config.xml tiedostoon määritellään sovelluksen tarvitsemat oikeudet, joihin sovellus pyytää lupaa käyttäjältä. Cordova liittää config.xml tiedostossa määritellyt oikeuspyynnöt AndroidManifest.xml tiedostoon, jonka avulla oikeuksia voidaan pyytää käyttöjärjestelmältä. (Apache Cordova 2018b.)

Android sovellukset voivat käyttää muita puhelimen ominaisuuksia tai sovelluksia aikomusten avulla. Aikomukset ovat asynkronisia viestejä, joissa tarvittavan ominaisuuden käynnistystä pyydetään. Aikomuksilla voidaan siis esimerkiksi pyytää puhelimen kameran käyttöä tai pääsyä kuvagalleriaan. (Vogel 2016.)

Cordovassa sovelluslisäosilla on käytössään CordovaInterface-objekti, jonka avulla cordovasovellukset voivat käyttää Androidin aikomuksia. CordovaInterface sallii lisäosien aloittaa toimintoja sekä asettaa takaisinkutsu lisäosaan, kun aikomus palaa sovellukseen. (Apache Cordova 2018b.)

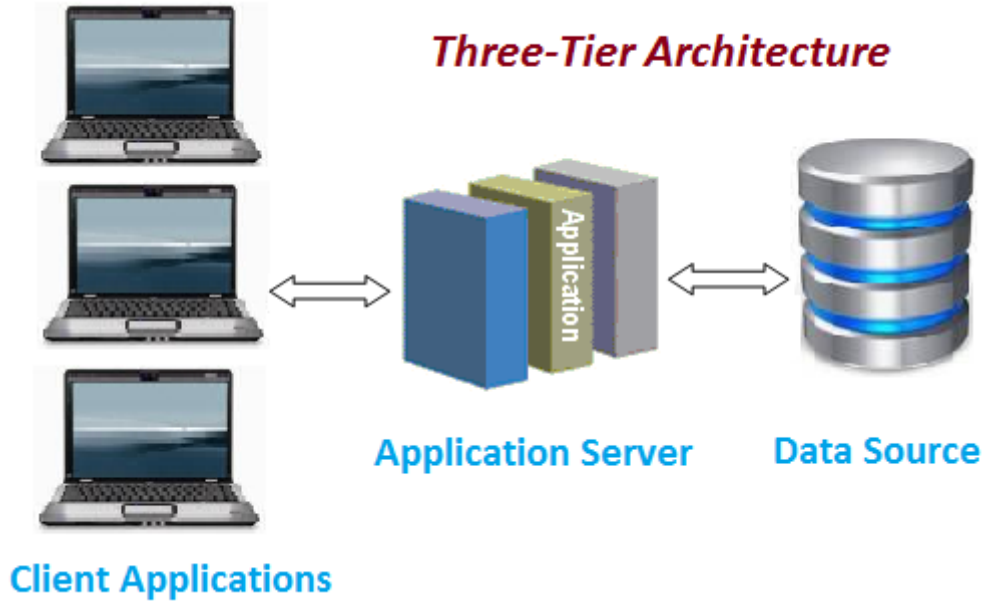
6 DIGILOOP

6.1 Yleiskuvaus

Digiloop on kierrätyssovellus, jonka avulla saadaan tehostettua yksittäisten akku-, SER- ja tietoturvajätteiden kierrätystä. Sen avulla yksityiset henkilöt sekä yritykset voivat ilmoittaa hallustaan löytyvistä jätteistä, jolloin kierrätysyritysten on helpompaa noutaa jätteet kannattavasti ja tehokkaasti.

6.2 Sovelluksen rakenne

Sovellus on toteutettu kolmitasoarkkitehtuurilla. Se koostuu Reactilla toteutetusta käyttöliittymästä, Node.js ja Expressillä toteutetusta palvelinpuolesta sekä MariaDB-tietokannasta. Kolmitasoarkkitehtuurin rakenne on esitetty kuviossa 16. Se on hyvin lineaarinen, jossa käyttöliittymä ja tietokanta eivät ole suorassa yhteydessä, vaan kaikki data niiden välillä liikkuu sovelluksen palvelinpuolen kautta.



KUVIO 16. Tyypillinen kolmitasoarkkitehtuurin rakenne (Software Testing Class 2013.)

Sovelluksen käyttöliittymä toteutettiin React.js-viitekehyksellä. Reactilla sovelluksen käyttöliittymän rakenteesta saatiin modulaarinen, jolloin sovelluksen kaikki osat ovat selkeästi komponentteina.

Käyttöliittymän datan hallinnan käytännöllisyys Reactissa kärsii samalla, kun sovelluksen koko kasvaa. Koska React tukee datan liikettä vain vanhemmalta lapselle tai lapselta vanhemmalle, dataa on vaikea siirtää kahden toisistaan kaukana olevan komponentin välillä. Tätä varten sovellukseen lisättiin Redux-kirjasto, joka luo sovelluksen datan tilasta tallenteen, jolloin dataa on helpompaa käsitellä eri komponenttien välillä.

Palvelinpuoli toteutettiin Node.js palvelimella sekä Express-viitekehyksellä. Palvelimen ja käyttöliittymän välinen tietoliikenne tapahtuu REST-rajapinnan avulla. Palvelinpuoleen luotiin rajapintaosoitteet, joita voidaan kutsua käyttöliittymästä.

Sovellus jakaantuu kahteen keskeiseen käyttäjäryhmään: jätteiden ilmoittajiin sekä jätteiden käsittelijöihin. Ilmoittajat kykenevät ilmoittamaan hallussaan olevat jätteet, sekä saavat tiedon, mikäli jokin käsittelijöistä on valmis sen noutamaan. Käsittelijät näkevät kaikki vapaana olevat ilmoitukset sekä pystyvät varaamaan niitä. Joissakin hahmotelmissa sovellukseen oli tarkoitus lisätä myös kolmas keskeinen ryhmä, joka olisi vastuussa jätteiden kuljetuksesta, mutta se on jätetty sovelluksen ensimmäisestä versiosta pois.

6.3 Kommunikointi

Käyttöliittymän ja palvelinpuolen välinen kommunikointi on toteutettu REST-kutsuilla. Käyttöliittymä kertoo palvelimelle tarvitsemansa tiedot tai ilmoittaa muutoksista palvelimelle REST-kutsulla. Kaikki tilanmuutokset lähtevät käyttöliittymästä, eli palvelimelta ei koskaan lähetetä mitään pyytämätöntä käyttöliittymään.

Käyttöliittymään määriteltiin erilliset funktiot kunkin REST-kutsun käyttämiseen. Niiden avulla kutsuja voitiin käyttää helpommin eri puolilta sovellusta, sekä saatiin sovelluksen rakenteesta modulaarisempi sekä helpommin ylläpidettävä.

```
function sendItemData(itemData) {
  let fd = new FormData();

  // need to use this due to the image requiring multipart/form-data
  for (let i = 0; i < itemData.length; i++) {
    fd.set(i + "category" , itemData[i].category)
    fd.set(i + "city" , itemData[i].city)
    fd.set(i + "description" , itemData[i].description)
    fd.set(i + "iscompany" , itemData[i].iscompany)
    fd.set(i + "latitude" , itemData[i].latitude)
    fd.set(i + "longitude" , itemData[i].longitude)
    fd.set(i + "pcs" , itemData[i].pcs)
    fd.set(i + "phone" , itemData[i].phone)
    fd.set(i + "pickupInstructions" , itemData[i].pickupInstructions)
    fd.set(i + "pickupaddr" , itemData[i].pickupaddr)
    fd.set(i + "size" , itemData[i].size)
    fd.set(i + "subCat" , itemData[i].subCat)
    fd.set(i + "weight" ,itemData[i].weight)
    fd.set(i + "zipcode" , itemData[i].zipcode)
    fd.set(i + "img" , itemData[i].image)
  }

  return axios({
    method: 'post',
    url: BASE_URL + '/itemAdd',
    data: fd,
    config: { headers: { 'Content-Type': 'multipart/form-data' } }
  })
    .then(response => response.data)
    .catch(function (error) {
      return error;
    });
}
```

KUVIO 17. Jäteilmoituksen palvelimelle lähettävä funktio

```
router.post('/itemAdd', middleware.wrap(async (req, res, next) => {
    let itemAdd = new itemadd(req.body, req.files)
    let msngImg = ['0']
    msngImg = await itemAdd.onlyMissingImg()

    let splittedArray = await misk.spliceArray(itemAdd.cleanArray(), 14)
    const query = 'INSERT INTO junk ( category, city, description, iscompany, latitude, longitude, pcs, itemphone, wishbo
    const secondary = [['date', Date.now()], ['datetoday', misk.dateToday()], ['owner', 1], ['userid', req.user.id]]

    await misk.createArray(splittedArray, secondary, req.files, sqldatahaku.querySql, query, msngImg)
    redis.setTimestamp();
    res.end();
}));
```

KUVIO 18. Palvelimen jäteilmoituksia vastaanottava rajapinta

Käyttöliittymästä lähetettyihin kutsuihin käytettiin Axios JavaScript-kirjastoa. Axios on lupauspohjainen kirjasto, jolla voidaan lähettää asynkronisia HTTP-kutsuja REST-rajapintoihin, sekä suorittaa CRUD-toimintoja (Eschweiler 2017). Kuviossa 17 on sovelluksen Axios-kutsu, jolla lähetetään uudet jäteilmoitukset palvelimelle. Kuviossa 18 on puolestaan näitä kutsuja vastaanottava rajapinta palvelimella.

Sovelluksen kannalta on tärkeää, että kaikki rajapinnat toimivat samalla periaatteella ja ovat rakenteeltaan modulaarisia. Rajapinnoista luotiin myös kattava dokumentaatio, josta selviävät käytössä olevat polut, vaadittu käyttäjätaso, odotetut parametrit sekä palautettavan datan tiedot.

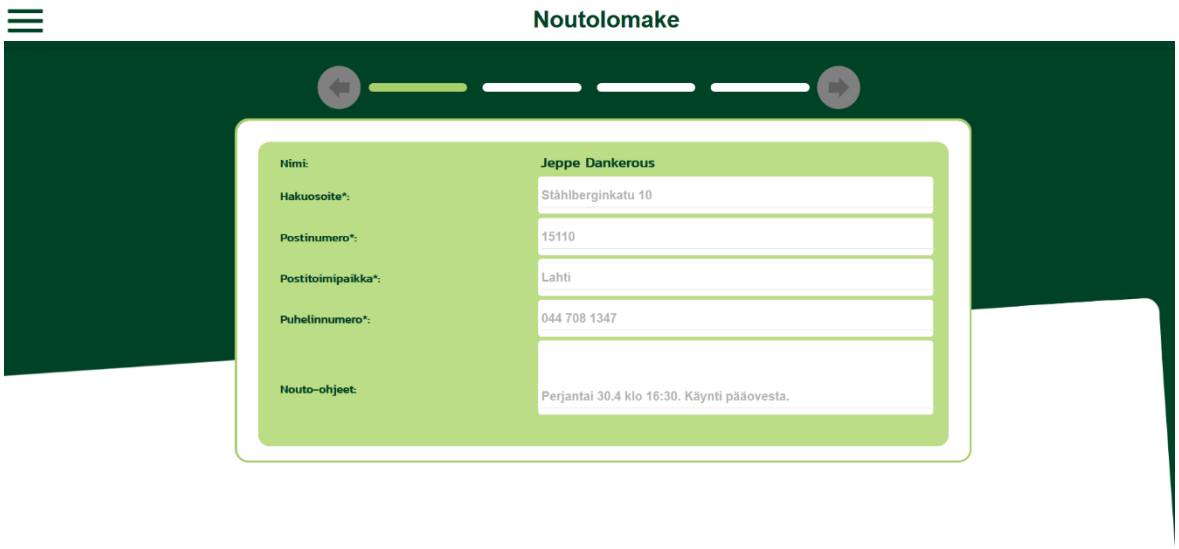
Sovelluksen kirjautumissessiot toimivat evästeillä. Onnistuneen kirjautumisen jälkeen palvelin antaa käyttöliittymälle sessioevästeen. Aina kun käyttöliittymä haluaa ladata mitään kirjautumisen vaativaa dataa, esimerkiksi käyttäjälistoja, lähetetään sessioeväste kutsun mukana. Palvelin tarkistaa evästeestä, onko kyseisellä käyttäjällä oikeutta pyytämäänsä dataan.

6.4 Responsiivinen toteutus

Koska sovellusta tullaan käyttämään hyvin erilaisissa tilanteissa ja erilaisilla laitteilla, on tärkeää, että se toimii laitteesta sekä käyttötarpeesta riippumatta hyvin. Ilmoituksia saatetaan tehdä puhelimilla tai taulutietokoneilla. Jätteiden käsittelijät käyttävät todennäköisimmin tietokoneita. Sovelluksen toteutuksessa pyrittiin huomioimaan erilaiset käyttötarpeet luomalla käyttöliittymästä mahdollisimman responsiivinen sekä priorisoimalla suunnittelu sopimaan kunkin käyttäjätyypin todennäköisesti yleisimmin käytettyyn laitteeseen sopivaksi.

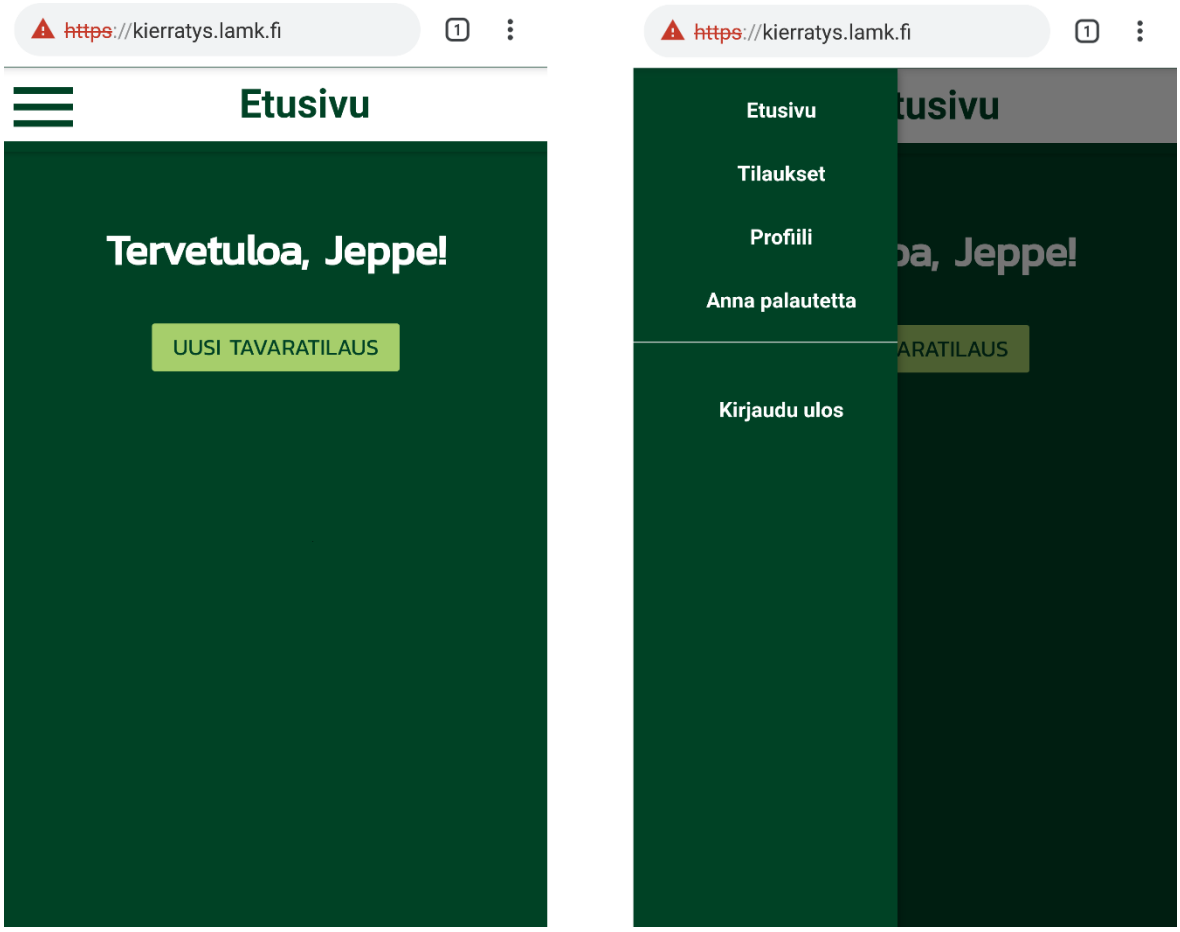
6.4.1 Mobiilikeskeinen suunnittelu

Suurin osa sovelluksen käyttäjistä on ihmisiä, joilta löytyy kerättävää jätettä. Näiden käyttäjien oletetaan suurimmalta osin käyttävän sovellusta mobiililaitteella, joten sitä pidettiin lähtökohtana sovelluksen suunnittelussa. Tämä yritettiin huomioida etenkin jätteiden ilmoitusosan rakenteissa.



KUVIO 19. Sovelluksen noutolomakkeen ensimmäinen sivu 1920x1080 kokoisella näytöllä

Sovelluksessa on vältetty turhien elementtien käyttöä, jolloin on ollut helpompaa toteuttaa sovelluksen mobiiliversio. Kaikki elementit sisältävät sovelluksen kannalta tärkeitä toimintoja. Isoilla ruuduilla käytettynä syntyy paljon tyhjää tilaa, mutta sitä ei nähty tarpeelliseksi täyttää tarpeettomilla elementeillä. Kuviossa 19 nähdään sovelluksen käyttöliittymä tietokoneen näytöllä. Sisältö on pyritty keskittämään mukavalle katseluleveydelle, jolloin ylimääräinen tila jää sivuille.



KUVIO 20. Sovelluksen valikko mobiilinäkymässä, piilotettuna sekä avattuna

Kaikki valikot päätettiin piilottaa erillisen valikkonapin taakse sovelluksen ilmoitusosassa. Kuviossa 20 nähdään sovelluksen valikko piilotettuna sekä avattuna. Ilmoittajilla oletetaan olevan hyvin vähän tarvetta käyttää valikkoa, sillä sovelluksen heille keskeisin osa: jätteiden ilmoitus, tapahtuu suoraan ilmoitusosan etusivulta.

6.4.2 Responsiivinen rakenne

Koska sovelluksen käyttöliittymä on suunniteltu mobiilikeskeiseksi, mediakyselyitä on käytetty lähinnä korjaamaan sovelluksen näkymää suurilla näyttöko'oilla. Esimerkiksi kuviossa 21 nähdään, kuinka jätteiden ilmoituskaavakkeen leveyttä on säädetty näytön leveyteen suhteutettuna kapeammaksi kuin, mitä se olisi mobiililaitteella käytettäessä.

```
566   @media (max-width: 1179px) {
567     .FrontPageContainer {
568       max-width: 80%;
569       font-weight: bold;
570       font-family: kanit;
571       margin: auto;
572       width: 98%;
573       min-height: 100vh;
574       align-content: center;
575     }
576   }
577   @media (min-width: 1280px) {
578     .FrontPageContainer {
579       max-width: 60%;
580       font-weight: bold;
581       font-family: kanit;
582       margin: auto;
583       width: 98%;
584       min-height: 100vh;
585       align-content: center;
586     }
587   }
588
```

KUVIO 21. Ilmoituslomakkeen leveys määritellään käyttäjän ruudun mukaan

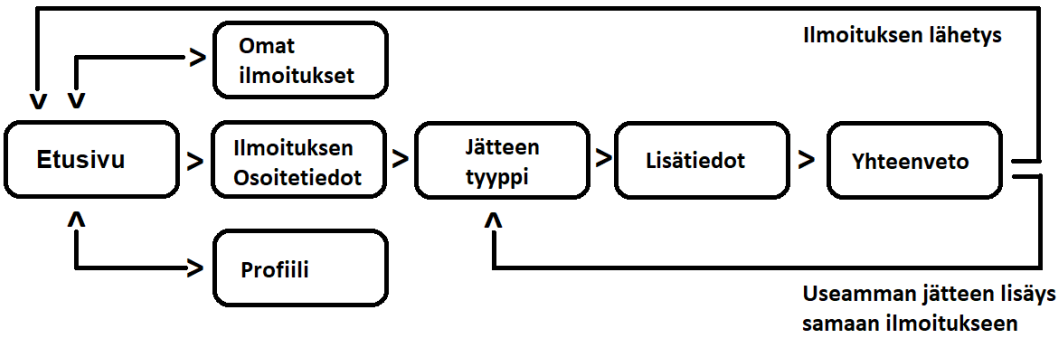
Sovelluksen responsiivisuutta edistää myös CSS-arvojen määrittely suhteellisina arvoina. Suurin osa elementtien ko'oista on määritelty säiliöelementtinsä suhteen. Osa arvoista on tosin määritelty elementissä käytetyn fonttikoon tai käyttäjän laitteen kokoon suhteutettuna.

6.5 Cordova-mobiilisovellus

Verkkosovelluksesta toteutettiin myös hybridi-mobiilisovellus rinnalle. Mobiilisovellus tehtiin Cordovan avulla. Sovellus toteutettiin ainoastaan Android laitteille, sillä sen koettiin olevan tärkein alusta sovellukselle. Windows ja iOS -versiot olisi kyetty myöskin luomaan, mutta sitä ei koettu tarpeelliseksi etenkään, kun mobiilisovellus ei ollut projektin kannalta tärkeä.

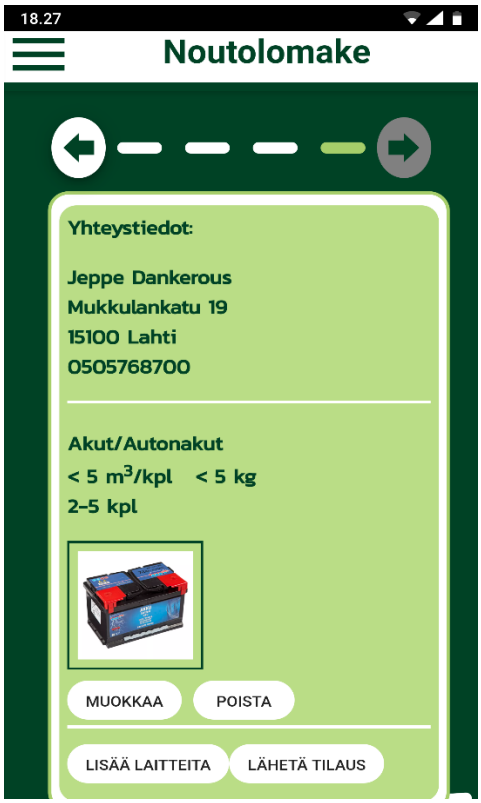
Sovellus tehtiin hybridisovelluksena. Sovelluksen koodikanta on sama kuin verkkosovelluksessa, johon on lisätty mobiilin tarvitsemat välikappaleet Cordovan avulla. Tämä helpottaa toteutusta huomattavasti, kun mobiilisovellusta ei tarvitse rakentaa alusta.

Sovellukseen on liitetty verkkosovelluksen internet-osoite, jota sovelluksen kautta käytetään. Tämän etuna on se, ettei sovellusta tarvitse ladata uudelleen tai päivittää, mikäli verkkosovellusta päivitetään. Huonona puolena mainittakoon se, että sovelluksen käyttö vaatii aina internet yhteyden, mutta sovellus luonteeltaan vaatisi sen muutenkin, joten siitä ei koidu ylimääräistä haittaa.



KUVIO 22. Sovelluksen rakenne loppukäyttäjälle

Sovelluksen rakenne on käyttäjän kannalta hyvin lineaarinen. Kuviosta 22 nähdään sovelluksen osat loppukäyttäjälle. Keskeisin osa sovelluksesta on uusien jätteiden ilmoituslomake. Kuviossa 23 nähdään ilmoituslomakkeen yhteenvetosivu, jossa voidaan lisätä muita jätteitä samaan ilmoitukseen, muokata lähtevää ilmoitusta tai lähettää ilmoitus palveluun. Muiden jätteiden lisääminen sekä muokkaaminen palauttaa käyttäjän jätteen tyypin -valintasivulle, josta käyttäjä voi edetä polkua jälleen eteenpäin.



KUVIO 23. Jätteenilmoituslomakkeen yhteenvetosivu mobiilisovelluksessa

7 YHTEENVETO

Työn tavoitteena oli luoda verkkosovelluksen prototyyppi, jonka avulla jätteiden kierrätystä kyettäisiin tehostamaan. Sovelluksen käyttöliittymän täytyi olla responsiivinen. Tavoitteena oli myös luoda siitä hybridimobiilisovellus.

Sovelluksesta saatiin ensimmäinen prototyyppi valmiiksi, jossa kaikki keskeisimmät ominaisuudet toimivat. Osa vähemmän tärkeistä ominaisuuksista jäi puutteellisiksi tai toimivat huonosti, mutta ne eivät olennaisesti vaikuta sovelluksen toimintaan.

Sovelluksen toteutuksen responsiivisuus oli tärkeää etenkin ilmoittajan osassa sovellusta. Siltä osin toteutuksen responsiivisuus on onnistunut, ja käyttöliittymää on mahdollista käyttää sujuvasti sekä mobiililaitteilla että tietokoneilla. Käsittelijöiden sekä ylläpitäjien käyttöliittymät toimivat huonosti mobiililaitteilla, sillä näiden käyttäjäryhmien oletetaan käyttävän sovellusta lähes puhtaasti tietokoneella, eikä sen osuuden responsiivista toteutusta siis pidetty tärkeänä.

Sovelluksen kannalta olisi ollut hyödyllistä, jos responsiivisuuden ja toimivuuden ongelmiin olisi mietitty ratkaisuja jo suunnitteluvaiheessa. Sovelluksen responsiiviset toteutukset on rakennettu pitkälti jälkikäteen, jolloin osa responsiivisista metodeista on vaikeaa toteuttaa. Esimerkiksi responsiivisen ruudukkorakenteen luominen myöhään projektissa olisi vaatinut liian isoja muutoksia sovelluksen rakenteeseen.

Hybridimobiilisovellus luotiin Androidille, mutta sitä ei jaettu esimerkiksi Play-kaupassa, sillä projektia ei ollut vielä tarkoitus jakaa julkisesti. Mobiilisovellusta jaettiin ainoastaan rajallisesti testikäyttöön projektissa mukana oleville tahoille. Kaupallistetussa versiossa sovelluksesta tehtäisiin todennäköisesti myös ainakin iOS-versio.

LÄHTEET

Ancheta, W. 2016. An Introduction to Cordova: Basics [viitattu 18.11.2018]. Envato tuts+. Saatavissa: https://code.tutsplus.com/tutorials/an-introduction-to-cordova-basics--cms-25146

Android 2018a. Material Design for Android [viitattu 1.12.2018]. Android documentation. Saatavissa: https://developer.android.com/guide/topics/ui/look-and-feel/

Android 2018b. Permissions overview [viitattu 27.12.2018]. Android documentation. Saatavissa: https://developer.android.com/guide/topics/permissions/overview

Apache Cordova 2018a. Android Plugin Development Guide [viitattu 10.2.2019]. Cordova documentation. Saatavissa: https://cordova.apache.org/docs/en/8.x/guide/platforms/android/plugin.html

Apache Cordova 2018b. Cordova Command-line-interface (CLI) Reference [viitattu 10.2.2019]. Saatavissa: https://cordova.apache.org/docs/en/8.x/reference/cordova-cli/index.html

Apache Cordova 2018c. Create your first Cordova app [viitattu 27.12.2018]. Saatavissa: https://cordova.apache.org/docs/en/latest/guide/cli/index.html

Apache Cordova 2018d. Overview [viitattu 26.10.2018]. Cordova documentation. Saatavissa: https://cordova.apache.org/docs/en/latest/guide/overview/

Codeacademy 2018 What is REST? [viitattu 17.1.2019]. Codeacademy. Saatavissa: https://www.codecademy.com/articles/what-is-rest

Code Realm 2018. Meet Material-UI—your new favorite user interface library [viitattu 7.11.2018]. freeCodeCamp. Saatavissa: https://medium.freecodecamp.org/meet-your-material-ui-your-new-favorite-user-interface-library-6349a1c88a8c

Craig, W. 2010. A Brief Look at Grid-Based Layouts in Web Design [viitattu 17.10.2018]. WebFX. Saatavissa: https://www.webfx.com/blog/web-design/a-brief-look-at-grid-based-layouts-in-web-design/

Eschweiler, S. 2017. Getting Started With Axios [viitattu 17.1.2019]. CodingTheSmartWay. Saatavissa: https://medium.com/codingthesmartway-com-blog/getting-started-with-axios-166cb0035237

Foddai, V. 2016. CSS Frontend Frameworks: The Best 10 for Modern Web Design [viitattu 2.12.2018]. Linkedin. Saatavissa: https://www.linkedin.com/pulse/css-frontend-frameworks-best-10-modern-web-design-valerio-foddai

Gazarov, P. 2016. What is an API? In English, please. [viitattu 17.1.2019]. freeCodeCamp. Saatavissa: https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82

Gove, J. 2018. What Makes a Good Mobile Site? [viitattu 1.11.2018]. Google. Saatavissa: https://developers.google.com/web/fundamentals/design-and-ux/principles/

Laki jätelain muuttamisesta 445/2018.

LocalOffers 2016. My Website looks bad on Mobile & Smartphone – A smart Solution [viitattu 21.3.2019]. Saatavissa: http://www.localofferstemplatewebsites.com/my-website-looks-bad-on-mobile-smartphone-a-smart-solution/

Marcotte, E. 2011. Responsive web design. A Book Apart.

Material Design 2018. Responsive layout grid [viitattu 7.11.2018]. Material Design. Saatavissa: https://material.io/design/layout/responsive-layout-grid.html

Material-UI 2018a. Basics [viitattu 18.12.2018]. Material-UI. Saatavissa: https://material-ui.com/layout/basics/

Material-UI 2018b. Breakpoints [viitattu 1.1.2019]. Material-UI. Saatavissa: https://material-ui.com/layout/breakpoints/

Material-UI 2018c. Buttons [viitattu 18.12.2018]. Material-UI. Saatavissa: https://material-ui.com/demos/buttons/

Material-UI 2018d. Grid [viitattu 1.1.2019]. Material-UI. Saatavissa: https://material-ui.com/layout/grid/

Material-UI 2018e. Hidden [viitattu 1.1.2019]. Material-UI. Saatavissa: https://material-ui.com/layout/hidden/

Material-UI 2018f. Themes [viitattu 1.1.2019]. Material-UI. Saatavissa: https://material-ui.com/customization/themes/

Mozilla 2019a. An overview of HTTP [viitattu 30.3.2019]. Mozilla Foundation. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Mozilla 2019b. How the Web works [viitattu 31.3.2019]. Mozilla Foundation. Saatavissa: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works

Poteet, C. 2015. An Introduction to Mobile-First Media Queries [viitattu 23.3.2019]. Sitepoint. Saatavissa: https://www.sitepoint.com/introduction-mobile-first-media-queries/

Rouse, M. 2011. hybrid application (hybrid app) [viitattu 1.1.2019]. TechTarget. Saatavissa: https://searchsoftwarequality.techtarget.com/definition/hybrid-application-hybrid-app

Saleh, H. 2014. Javascript Mobile Application Development. Packt Publishing.

Schade, A. 2014. Responsive Web Design (RWD) and User Experience [viitattu 20.9.2018]. Saatavissa: https://www.nngroup.com/articles/responsive-web-design-definition/

Software Testing Cass 2013. What is Difference Between Two-Tier and Three-Tier Architecture? [viitattu 27.12.2018]. Software Testing Class. Saatavissa: https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/

StatCounter 2019. Desktop & Mobile Screen Resolution Stats Worldwide [viitattu 29.3.2019]. StatCounter. Saatavissa: http://gs.statcounter.com/

Vogel, L. 2016. Android Intents – Tutorial [viitattu 1.2.2019]. Vogella. Saatavissa: http://www.vogella.com/tutorials/AndroidIntent/article.html

Wikimedia Commons 2015. Material Design.svg. Saatavissa: https://commons.wikimedia.org/wiki/File:Material_Design.svg

Woodhead, W. 2018. Should you use Material Design? [viitattu 13.12.2018]. Pilcro. Saatavissa: https://medium.com/pilcro/should-you-use-material-design-bfb596a04bae

W3Schools 2018a. CSS Units [viitattu 20.9.2018]. Saatavissa: https://www.w3schools.com/css/css_units.asp

W3Schools 2018b. Responsive Web Design - Media Queries [viitattu 21.9.2018]. Saatavissa: https://www.w3schools.com/css/css_rwd_mediaqueries.asp

W3Schools 2018c. Responsive Web Design - The Viewport [viitattu 21.9.2018]. Saatavissa: https://www.w3schools.com/css/css_rwd_viewport.asp

Yates, I. 2013. Life Beyond 960px: Designing for Large Screens [viitattu 17.10.2018]. Envato tuts+. Saatavissa: https://webdesign.tutsplus.com/articles/life-beyond-960px-designing-for-large-screens--webdesign-7348

Ympäristöminesteriö 2016. Valtakunnallinen jätesuunnitelma [viitattu 24.3.2019]. Ympäristöministeriö. Saatavissa: https://www.ym.fi/fi-FI/Ymparisto/Jatteet/Valtakunnallinen_jatesuunnitelma