DYNAAMISTEN LINKKIEN JA QR-KOODIEN HYÖDYNTÄMINEN MOBIILISOVELLUKSEN KEHITYSPROSESSISSA

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus
kevät, 2023

Ahonen Ari-Jussi

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus — Tiivistelmä

Tekijä Ahonen Ari-Jussi — Vuosi 2023

Työn nimi Dynaamisten linkkien ja QR-koodien hyödyntäminen mobiilisovelluksen kehitysprosessissa

Ohjaaja Ojaniemi Pentti

TIIVISTELMÄ

Tämän opinnäytetyön tavoitteena oli selvittää kuinka dynaamisia linkkejä ja niistä generoituja QR-koodeja on mahdollista luoda ohjelmallisesti sekä hyödyntää mobiilisovelluksen kehitysprosessissa. Tavoitteena oli sovelluksen käyttäjämäärän mahdollisimman nopea kasvattaminen sekä sovelluksen käyttäjäystävällisyyden kehittäminen. Opinnäytetyön toimeksiantajana toimi Calevala Interactive Oy.

Opinnäytetyön tietopohja koostuu QR-koodin historian, ominaisuuksien ja mahdollisuuksien esittelystä. Lisäksi työssä kartoitettiin QR-koodin generointiin sopivia erilaisia ohjelmointikieliä. Opinnäytetyön teoriaosuudessa tutustuttiin myös dynaamisten linkkien ominaisuuksiin ja etuihin sekä etsittiin erilaisia alustoja dynaamisen linkkien hallintaan. Tämän lisäksi työssä esiteltiin teoreettista taustaa Python- ja JavaScript-ohjelmointikielistä, REST-arkkitehtuurimallista sekä React Native Framework -kehitysympäristöstä. Opinnäytetyö on tyypiltään toiminnallinen opinnäytetyö.

Kehitysprojektin tuloksena havaittiin QR-koodien ja dynaamisten linkkien soveltuvan erittäin hyvin osaksi mobiilisovelluksen kehitysprosessia. Firebasen dynaamisten linkkien sekä QR-koodien luominen ohjelmallisesti koettiin helpoksi toteuttaa. Työn tuloksena syntyneen ohjelman integrointiin asiakkaan alustalle löydettiin useita eri vaihtoehtoja ja toimeksiantaja oli tyytyväinen kehittämistyön tuloksiin.

Avainsanat QR-koodi, dynaaminen linkki, mobiilisovellus, Google Firebase, Python

Sivut 53 sivua ja liitteitä 2 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology Abstract

Author Ahonen Ari-Jussi Year 2023

Subject Utilization of dynamic links and QR codes in a mobile application development process

Supervisor Ojaniemi Pentti

ABSTRACT

The goal of this thesis was to find out how dynamic links and QR codes can be created programmatically and utilized in the development process of a mobile application. The goal was to increase the number of users of the application as quickly as possible and to develop the user-friendliness of the application. The client of the thesis was Calevala Interactive Oy.

The knowledge base of the thesis consists of a presentation of the history, features and possibilities of the QR code. In addition, the work mapped various programming languages suitable for QR code generation. In the theory part of the thesis, the characteristics and advantages of dynamic links and various platforms for managing dynamic links were also introduced. In addition to this, the work presented the theoretical background of the Python and JavaScript programming languages, the REST architectural model, and the React Native Framework development environment. The thesis is a functional thesis.

As a result of the development project, QR codes and dynamic links were found to be very well suited as part of the mobile application development process. Creating Firebase dynamic links and QR codes was found to be relatively easy to implement programmatically. Several different options were found for the integration of the resulting program into the client's platform and the client was satisfied with the results of the development work.

Keywords QR code, dynamic link, mobile application, Google Firebase, Python

Pages 53 pages and appendices 2 pages

Sanasto

QR-koodi	Quick Response code. Kaksiulotteinen kuviokoodi, johon on koodattu informaatiota.
Dynaaminen linkki	Ohjelmoitava linkki, joka tunnistaa käytössä olevan käyttöjärjestelmän sekä siihen liitetyn sovelluksen asennuksen tilan ja toimii tämän mukaan halutulla tavalla.
Syvälinkki	Mobile deep link. Linkki, joka johtaa suoraan tiettyyn sisältöön tai paikkaan mobiilisovelluksessa.
Google Firebase	Googlen alusta, joka tarjoaa erilaisia tuotteita sovelluskehityksen avuksi. Tarjoaa myös URL-osoitteita dynaamisten linkkien ylläpitoon.
API	Application Programming Interface. Ohjelmointirajapinta, jonka avulla eri ohjelmat tai ohjelmat ja laitteet voivat keskustella keskenään.
REST	Representational state transfer. Ohjelmointirajapintojen toteuttamiseen tarkoitettu arkkitehtuurimalli (tai -tyyli).
URL	Uniform Resource Locator. Verkkosivuston tai tiedoston sijainti internetissä.
HTML	Hypertext Markup Language. Verkkosivujen kirjoittamiseen kehitetty merkintäkieli. HTML:n avulla kuvataan nettisivun rakenne sekä sivun sisältämä sisältö.
CSS	Cascading Styling Sheets. Verkkosivuille kehitetty tyylisivu. CSS:n avulla voidaan määritellä monipuolisesti verkkosivujen ulkoasuun liittyviä ominaisuuksia ja rakenteita.

PHP	PHP: Hypertext Preprocessor. Alun perin vuonna 1995 julkaistu verkkopalvelimille suunniteltu skriptikieli, jota käytetään verkkosivujen toteuttamiseen.
CodeIgniter	PHP-ohjelmointikielelle suunniteltu avoimen lähdekoodin sovelluskehys, jonka avulla voidaan toteuttaa verkkosovelluksia.
Java	Sun Microsystemsin kehittämä teknologiaperhe ja ohjelmistoalusta. Javaan kuuluu myös laitteistoriippumaton oliopohjainen ohjelmointikieli.
JavaScript	Netscapen kehittämä, pääasiassa verkkoympäristössä käytettävä oliopohjainen ohjelmointikieli.
AJAX	Asynchronous JavaScript And XML. Tekniikka, jonka avulla voidaan siirtää tietoa selaimen ja palvelimen välillä ilman, että koko verkkosivua täytyy ladata uudelleen.
Python	Guido van Rossumin alun perin kehittämä oliopohjainen ohjelmointikieli.
Flask	Python ohjelmointikielen ympäristö, joka mahdollistaa web-sivujen kehittämisen Pythonin avulla.
WSGI	Web Server Gateway Interface. Python -ohjelmointikielen standardi web-sovellusten kehittämiseen.
ReactJS	JavaScript-ohjelmointikielelle kehitetty kirjasto, jonka avulla voidaan kehittää verkkokäyttöliittymiä.
JSX	JavaScript XML. JavaScript-ohjelmointikielen syntaksin laajennusosa, joka mahdollistaa JavaScript-koodin kirjoittamisen suoraan Reactilla kirjoitettuun koodiin.
React Native	Avoimen lähdekoodin ympäristö natiivien mobiilisovellusten kehittämiseen Android- ja iOS-alustoille.

Android	Googlen ylläpitämä ja kehittämä käyttöjärjestelmä, jota käytetään nykyään mm. älypuhelimissa, tablettitietokoneissa ja älytelevisioissa.
iOS	Applen kehittämä ja ylläpitämä käyttöjärjestelmä, joka on käytössä Applen valmistamissa älypuhelimissa, tablettitietokoneissa ja älytelevisioissa.
HTTP	Hypertext Transfer Protocol. Protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
PaaS	Platform as a service. Pilvipalveluiden malli, jossa koko sovellusalusta tarjotaan asiakkaalle ulkoistettuna palveluna.

Sisällys

Komennot, kuvat, ohjelmakoodit ja taulukot

Liitteet

1 Johdanto

Vuoden 2022 lopulla pelkästään Applen ja Googlen sovelluskaupoissa oli yhteensä yli viisi miljoonaa mobiilisovellusta saatavilla. Yksittäisen sovelluksen esiin nostaminen tästä valtavasta massasta on usein erittäin haasteellista. Onkin löydettävä keinoja, joilla uuden sovelluksen asentaminen ja käyttäminen on tehty loppukäyttäjille mahdollisimman yksinkertaiseksi, nopeaksi ja tehokkaaksi. Dynaamiset linkit ja QR-koodit tarjoavat osaltaan ratkaisun tähän.

Tämän opinnäytetyön tarkoituksena on selvittää, kuinka dynaamisia linkkejä ja niistä generoituja QR-koodeja voidaan luoda ja hallita sekä hyödyntää mobiilisovelluksen kehitysprosessissa. Tavoitteena on sovelluksen käyttäjämäärän mahdollisimman nopea kasvattaminen erityisesti sen julkaisun jälkeen, sovelluksen käyttäjäystävällisyyden kehittäminen sekä käyttäjien sitouttaminen. Opinnäytetyössä pyritään selvittämään, kuinka QR-koodeja luodaan ohjelmallisesti ja mikä olisi tässä tapauksessa paras vaihtoehto ohjelmointikieleksi. Työssä selvitetään myös, kuinka voidaan luoda ja hallita dynaamisia linkkejä, jotka ohjaavat käyttäjän suoraan haluttuun sisältöön käytössä olevan laitteen tai alustan mukaan. Lopuksi selvitetään, miten nämä ominaisuudet voidaan integroida asiakkaan sovelluksen hallinta-alustalle.

Työn lopputuloksena syntyy Python-ohjelmointikielellä toteutettu selainpohjainen ohjelma, joka generoi mukautetun QR-koodin, jonka sisältönä on dynaaminen linkki. Dynaaminen linkki on luotu asetuksilla, jotka ohjaavat linkin käyttäjän aina haluttuun sisältöön kulloinkin käytössä olevan alustan mukaan. Kun loppukäyttäjä skannaa QR-koodin tai klikkaa linkkiä, ohjataan hänet joko haluttuun sisältöön sovelluksessa tai suoraan paikkaan, josta sovelluksen voi ladata omalle laitteellensa. Tämän lisäksi työn tuloksena syntyy myös Android-pohjainen mobiilisovellus, jossa demonstroidaan, kuinka dynaamista linkkiä ja sen sisältöä voidaan ohjelmallisesti käsitellä.

Opinnäytetyön toimeksiantajana toimii Calevala Interactive Oy. Yritys on kehittämässä uutta digitaalista alustaa ja tarvitsee tämän työn lopputulosta avuksi uusien käyttäjien hankinnassa

ja sitouttamisessa. Tavoitteena on tehdä sovelluksen käyttö, asennus sekä sovelluksessa julkaistuihin kampanjoihin osallistuminen mahdollisimman vaivattomaksi käyttäjille.

Opinnäytetyön tutkimuskysymykset ovat

- Mikä olisi paras vaihtoehto ohjelmointikieleksi QR-koodin luomiseen?
- Mikä olisi paras alusta dynaamisen linkin luomiseen ja ylläpitoon?
- Kuinka dynaamisia linkkejä voidaan luoda ohjelmallisesti?
- Miten dynaamisen linkin sisältöä voidaan käsitellä mobiilisovelluksessa?
- Miten nämä ominaisuudet voidaan integroida olemassa olevaan sovellukseen?

2 QR-koodi

QR-koodien suosio on kasvanut voimakkaasti erityisesti viimeisten viiden vuoden aikana älypuhelinten ja nopeampien internetyhteyksien yleistymisen myötä. Myös COVID-19 pandemian vaikutuksesta äkillisesti nopeutunut yhteiskunnan digitalisaatio on myötävaikuttanut omalta osaltaan QR-koodien kasvavaan suosioon. Älypuhelimien kamerasovelluksissa on QR-koodin skannausominaisuus usein jo sisäänrakennettuna. QR-koodin skannaus säästää käyttäjältä aikaa ja vaivaa sekä osaltaan varmistaa sen, että käyttäjä suorittaa halutun prosessin loppuun asti. (Garg, 2018)

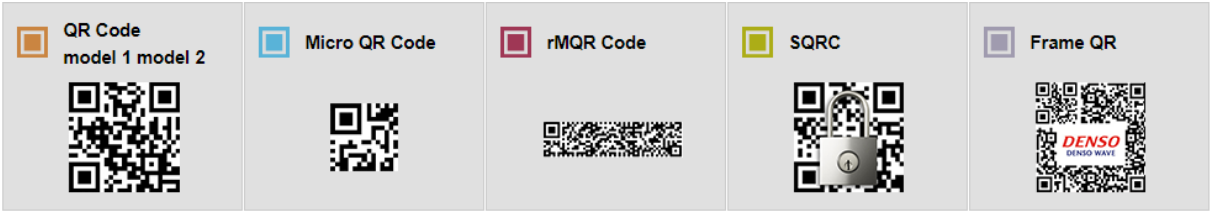
2.1 QR-koodin historia

QR-koodi kehitettiin Japanissa vuonna 1994. Sen kehitti Denso Wave niminen yhtiö, joka toimii Toyotan tytäryhtiönä. Alun perin QR-koodin kehityksen perustana oli se, että japanilaiset asiakkaat halusivat mahdollisuuden lisätä viivakoodiin myös japanilaisia Kanji- ja Kana-merkkejä. Tavallinen viivakoodi voi sisältää vain 20 kirjainta tai numeroa, joten Denso Wave perusti tiimin kehittämään uutta viivakoodi-mallia, johon mahtuisi tarvittava määrä tietoa. QR-koodin kehityksestä vastasi aluksi kahden hengen tiimi, jonka johtoon nimettiin Masahiro Hara. QR-koodin kehitystyö kesti noin puolitoista vuotta. Lopputuloksena kehitettiin koodi, johon mahtui tarvittava määrä tietoa ja joka voitiin myös lukea yli 10 kertaa nopeammin kuin aikaisemmat viivakoodit. QR-koodin nimi tuleekin englanninkielisistä sanoista **Quick Response** (nopea vaste). (*Denso Wave*, 2023)

2.2 Erityyppisiä QR-koodeja

Denso Wave on kehittänyt tähän mennessä viisi erityyppistä QR-koodia erilaisiin käyttötarkoituksiin (Kuva 1).

Kuva 1 Erityyppisiä QR-koodeja. (*Denso Wave*, 2023)



Model 1 on ensimmäinen kehitetty QR-koodimalli. **Model 2** on tästä uudempi, päivitetty versio, jossa QR-koodin sisältämän datan maksimimäärää kasvatettiin ja koodiin lisättiin myös suoristuskuviot sekä virheenkorjausominaisuus. **Micro QR** on pienikokoinen QR-koodi, jonka maksimikoko on vain 17x17 moduulia. Se on suunniteltu mahtumaan pieneen tilaan, mutta se voi sisältää maksimissaan vain 35 merkkiä. **rMQR** on suorakaiteen muotoinen QR-koodi, joka on suunniteltu erityisesti kapeaan tilaan. Tämän tyyppiseen QR-koodiin mahtuu maksimissaan 361 merkkiä tietoa. **SQRC** on puolestaan QR-koodi, joka koostuu sekä julkisesta että yksityisestä datasta. Yksityinen data on salattu suojausavaimen avulla ja sitä voi lukea ainoastaan tällaisen koodin lukemiseen suunnitellulla laitteella tai ohjelmistolla. Julkisen puolen datan voi taas lukea tavallisella QR-koodinlukijalla**. Frame QR** on uuden sukupolven QR-koodi. Siinä on keskellä tyhjä alue, jota voi käyttää vapaasti esimerkiksi oman kuvan tai tekstin lisäämiseen. Alueen muoto sekä QR-koodin värit ovat vapaasti muokattavissa. (*Creative Safety Supply*, 2023; *Denso Wave*, 2023)

Model 2 on tällä hetkellä ylivoimaisesti käytetyin malli QR-koodista. Tämä johtuu siitä, että Denso Wave on antanut avoimeen käyttöön Model 2 -tyyppisen QR-koodin spesifikaatiot. Tässä opinnäytetyössä keskitytäänkin tähän yleisimpään QR-koodityyppiin ja sen ominaisuuksiin.

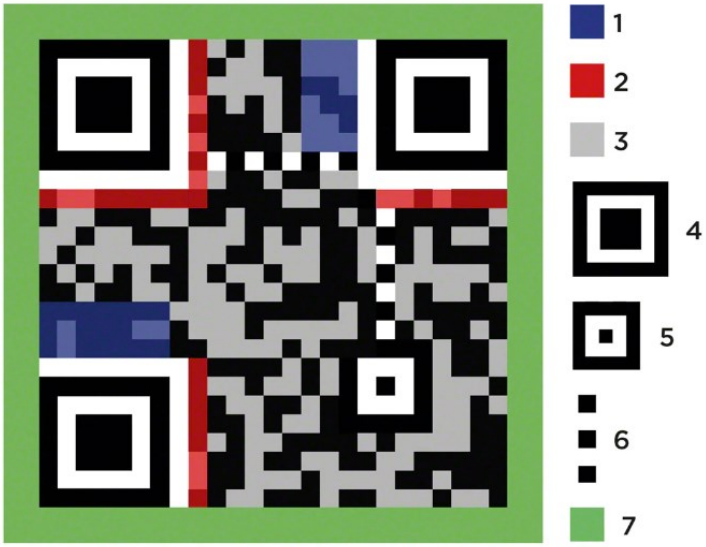
2.3 QR-koodin rakenne

QR-koodi on kaksiulotteinen viivakoodi eli se sisältää tietoa sekä pysty- että vaakasuunnassa. QR-koodin sisältämä data koostuu vuorottelevista, yleensä mustista ja valkoisista pisteistä, joita kutsutaan myös nimellä **moduuli**. Musta moduuli vastaa bittiä 1 ja valkoinen bittiä 0. Yhteen QR-koodiin voi teoriassa mahtua yli seitsemän tuhatta numeroa tai yli neljä tuhatta kirjainta. Käytännössä merkkejä täytyy olla kuitenkin huomattavasti vähemmän, jotta QR-koodia olisi mahdollista lukea esimerkiksi älypuhelimen kameralla. Mitä enemmän informaatiota QR-koodi pitää sisällään, sitä tiheämmäksi koodi tulee ja samalla myös sen fyysinen koko kasvaa. (*Creative Safety Supply*, 2023; *Denso Wave*, 2023)

QR-koodin kuvion rakenne koostuu seitsemästä eri osiosta (Kuva 2 ja Taulukko 1). Kohdistus- ja suoristuskuvioiden avulla QR-koodinlukija pystyy nopeasti lukemaan koodin oikein. Lisäksi

QR-koodin rakenteeseen on varattu erityiset alueet, joiden sisällä määritellään QR-koodin versio, virheenkorjauksen taso sekä koodissa käytetty maskikuvio. Näiden alueiden ulkopuolelle jäävä alue sisältää varsinaisen QR-koodin sisältämän datan sekä virheenkorjauskoodit. (Mecalux, 2021; *Denso Wave*, 2023)

Kuva 2 QR-koodin rakenne. (Mecalux, 2021)



Taulukko 1 QR-koodin rakenteen eri osiot.

1	Versioinformaatio. Määrittää käytetyn QR-koodin version.
2	Muotoinformaatio. Määrittää virheenkorjauksen tason ja käytössä olevan maskikuvion. Tämän tiedon järjestelmä lukee ensimmäisenä purkaessaan koodia.
3	Varsinainen data eli informaatio, jonka koodi sisältää. Määritellään vaihtelevien tummien ja vaaleiden pisteiden eli moduulien avulla.
4	Kohdistuskuviot. Kuviot QR-koodin kulmissa, joiden avulla QR-koodinlukija osaa kohdistaa skannauksen oikein.
5	Suoristuskuviot. Määrittävät sen, että QR-koodinlukija osaa tulkita koodin sisällön oikeinpäin, vaikka se luettaisiin missä kulmassa tahansa.
6	Ajoituskuviot. Vaakasuora ja pystysuora rivi vuorottelevia tummia ja vaaleita pisteitä, joiden avulla QR-koodinlukija määrittää data-alueen yksittäisen moduulin koon sekä rivien ja sarakkeiden lukumäärän.
7	Tyhjä tila. QR-koodin reunoilla oleva neljän moduulin levyinen tyhjä tila, joka on myös välttämätön osa koodia. Sen avulla varsinainen koodi eristetään ympäristöstään.

2.3.1 QR-koodin versio

QR-koodin versionumero ilmaistaan kokonaislukuna väliltä 1-40. Käytännössä se määrittää myös koodiin mahtuvan tiedon määrän ja samalla QR-koodin fyysisen koon. Version 1 koodi on kooltaan pienin, 21x21 moduulia ja version 40 koodi on suurin, kooltaan 177x177 moduulia. Käytännössä useimmin käytetyt QR-koodit ovat versioita 1–7, jotta niiden lukeminen koon puolesta onnistuisi järkevästi. (*Creative Safety Supply*, 2023; *Denso Wave*, 2023)

2.3.2 QR-koodin virheenkorjaustaso

QR-koodin muotoinformaatio sisältää tiedon QR-koodin **virheenkorjaustasosta**. QR-koodinlukija käsittelee aina tämän informaation ensimmäisenä, jotta se osaisi tulkita koodia oikein. QR-koodille on määritelty neljä eri virheenkorjauksen tasoa: **L, M, Q** ja **H** (Taulukko 2). Virheenkorjauksen taso määrittelee sen, kuinka suuri osuus QR-koodin datasta voi olla vahingoittunut koodin edelleen ollessa luettavissa. (*Denso Wave*, 2023; QRStuff.com, 2011)

Taulukko 2 QR-koodin virheenkorjaustasot.

Virheenkorjaustaso	Suurin sallittu vahingoittuneen datan osuus
L	7 %
M	15 %
Q	25 %
H	30 %

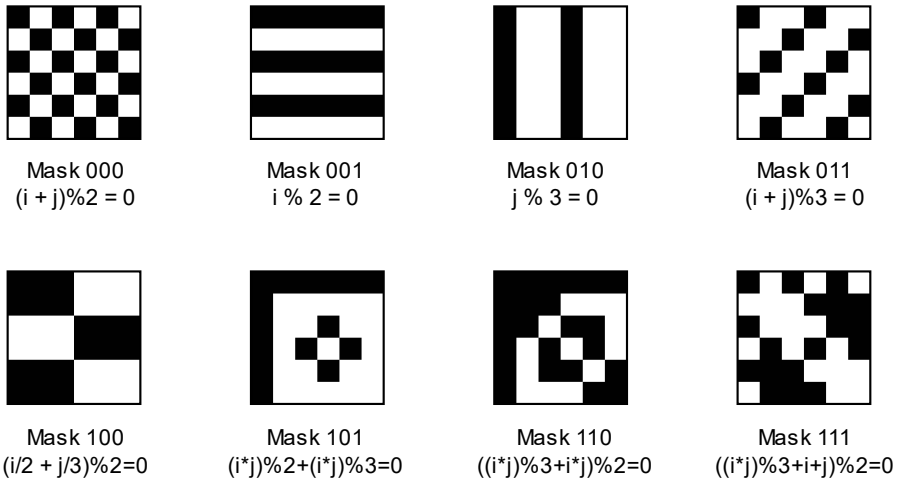
QR-koodin virheenkorjauskyky toteutetaan lisäämällä koodin dataan niin sanottu Reed-Solomon-koodi. Se on virheenkorjaukseen kehitetty matemaattinen koodi, jonka kehittivät Irving Reed ja Gustave Solomon Yhdysvalloissa 1960-luvulla. Virheenkorjauksen tason nostaminen parantaa QR-koodin virheenkorjauskykyä, mutta samalla se myös kasvattaa QR-koodin fyysistä kokoa sekä koodin tiheyttä, koska virheenkorjauskoodin dataa tarvitaan silloin myös enemmän. Yleisimmin QR-koodeissa on käytetty virheenkorjaustasoa M. Virheenkorjauskyky mahdollistaa esimerkiksi kuvan tai tekstin lisäämisen suoraan QR-koodin päälle. Lisätyn informaation koko täytyy kuitenkin suhteuttaa oikein QR-koodin kokoon ja

käytettyyn virheenkorjaustasoon nähden, jotta QR-koodiin koodattu data olisi vielä luettavissa. (QRStuff.com, 2011; Wicker & Bhargava, 1999, s. 1)

2.3.3 QR-koodin maskikuvio

QR-koodin muotoinformaatio sisältää myös tiedon koodin käyttämästä niin sanotusta **maskikuviosta**. Maskikuvion avulla QR-koodi pyritään muuntamaan mahdollisimman helposti luettavaksi. Sen tarkoituksena on varmistaa, ettei QR-koodiin muodostu isoja valkoisia alueita (0-bittejä), jotka vaikeuttavat koodin tulkitsemista. QR-koodin standardi tukee kahdeksaa erilaista maskikuviota (Kuva 3). Kaikki niistä noudattavat jotain erilaista matemaattista kaavaa. Kyseisessä kaavassa **i** vastaa aina kyseisen moduulin **rivinumeroa** ja **j** vastaa **sarakenumeroa**. Saman informaation sisältävä QR-koodi saattaa siis näyttää erilaiselta riippuen maskikuviosta, jota siinä on käytetty. (Thonky.com, 2023)

Kuva 3 QR-koodin erilaiset maskikuviot. (*QR-code mask patterns | Wikimedia*, 2011)

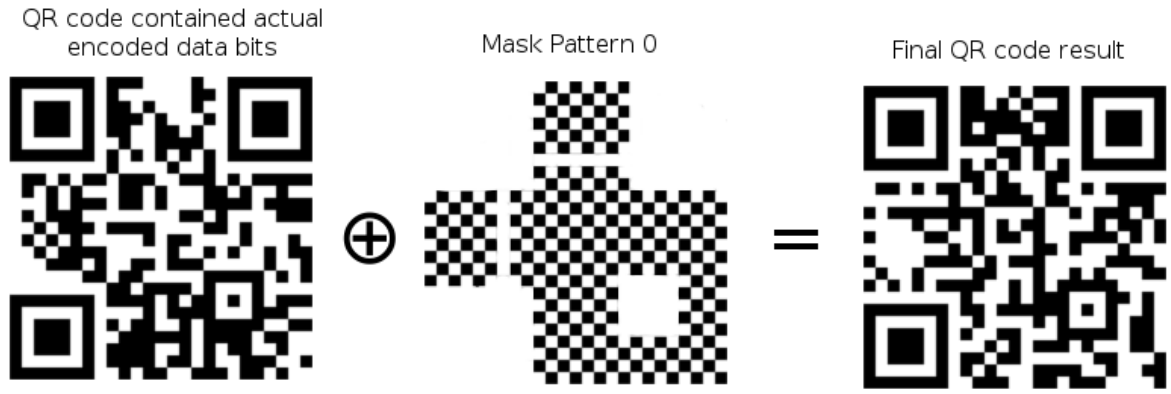


QR-koodinlukija tulkitsee jokaisen data-alueen pisteen koordinaattien osalta kaavaa siten, että jos kaava antaa tulokseksi nollan, on kyseinen piste luettaessa käännettävä vastakkaiseksi. Jos tietyssä koordinaatissa on musta piste eli bitti on 1 ja kaava antaa tulokseksi 0, oikea luettava bitti on silloin 0. Jos taas luettava bitti on 0 eli valkoinen ja

kaavan tulos on myös 0, oikea luettava bitti on silloin 1. Jos kaavan tuloksena ei ole 0, niin bitti luetaan sellaisenaan. (*QRazyBox - About qr-code*, 2023; Thonky.com, 2023)

Maskikuviota käytetään vain koodin varsinaisen data-alueen moduuleissa. Muut QR-koodin osa-alueet jätetään ennalleen. Jokaisen QR-koodin data-alueen pisteen ja maskikuvion vastaavan pisteen kesken suoritetaan looginen XOR-operaatio, eli jos molemmat pisteet ovat arvoltaan samoja (1 ja 1 tai 0 ja 0), on lopputuloksena 0. Jos taas pisteet ovat eriarvoisia (1 ja 0 tai 0 ja 1), on lopputuloksena 1. Kuva 4 on esitelty 0-tyypin maskikuvion käyttöä. Vasemmanpuoleinen QR-koodi on alkuperäinen malli, johon on koodattu haluttu informaatio. Keskellä on maskikuvio, jota käytetään ja oikealla on edellisistä yhdistetty lopullinen QR-koodi. (*QRazyBox - About qr-code*, 2023; Thonky.com, 2023)

Kuva 4 Esimerkki QR-koodin maskikuvion käytöstä. (*QRazyBox - About qr-code*, 2023)



2.4 QR-koodin käyttökohteet ja mahdollisuudet

Kun ensimmäisen QR-koodin kehitystyö oli saatettu loppuun vuonna 1994, se otettiin käyttöön Toyotan tehtailla Japanissa. Sen avulla tunnistettiin ja seurattiin auton eri osia nopeasti liikkuvilla liukuhihnoilla. Vuosien saatossa QR-koodin suosio on kasvanut räjähdysmäisesti erityisesti älypuhelinten yleistymisen myötä. QR-koodit ovat tätä nykyä tuttuja lähes kaikille. Niihin saattaa törmätä muun muassa mainoksissa, uutisissa, käyntikorteissa tai erilaisissa käyttöohjeissa. Tyypillisimmin QR-koodilla pyritään välittämään jokin osoitelinkki tai toiminto käyttäjän mobiililaitteeseen. QR-koodin avulla voidaan

nopeasti jakaa erilaista tietoa, kuten esimerkiksi nettilinkkejä, linkkejä tiettyyn sovellukseen, tekstiä, käyntikortteja, kuvia, videoita, sijaintitietoja tai sosiaalisen median linkkejä. QR-koodin skannaamalla käyttäjä voi esimerkiksi nopeasti kirjautua langattomaan verkkoon, lähettää WhatsApp-, SMS-, tai sähköpostiviestejä, soittaa puheluita, antaa palautetta tai arvosteluja, lisätä merkintöjä kalenteriin tai maksaa vaikkapa PayPal-palvelulla. (QRcodeChimp, 2022)

QR-koodin avulla kaikista näistä toiminnoista voidaan tehdä käyttäjälle yksinkertaisempia ja helpompia. Erilaisten tuotteiden ja palveluiden markkinointi tehostuu, kun käyttäjä pääsee nopeasti tutustumaan aiheeseen skannaamalla mainoksen yhteyteen liitetyn QR-koodin. Samalla myös pyritään varmistamaan, että käyttäjä todella ohjautuu oikeaan paikkaan, esimerkiksi tiettyyn sovellukseen.

2.5 QR-koodin generointi ohjelmallisesti

QR-koodin generointiin on olemassa valmiita, ilmaisia ohjelmakirjastoja lähestulkoon kaikille yleisimmille ohjelmointikielille. Koska Model2-tyyppisen QR-koodin spesifikaatio on vapautettu yleiseen käyttöön, olisi mahdollista kirjoittaa QR-koodin generointiin täysin oma ohjelmansa ilman valmiita ohjelmakirjastoja. Tämä ei kuitenkaan tarjoa mitään varsinaista käytännön hyötyä ja kuluttaa aikaa sekä resursseja tarpeettomasti.

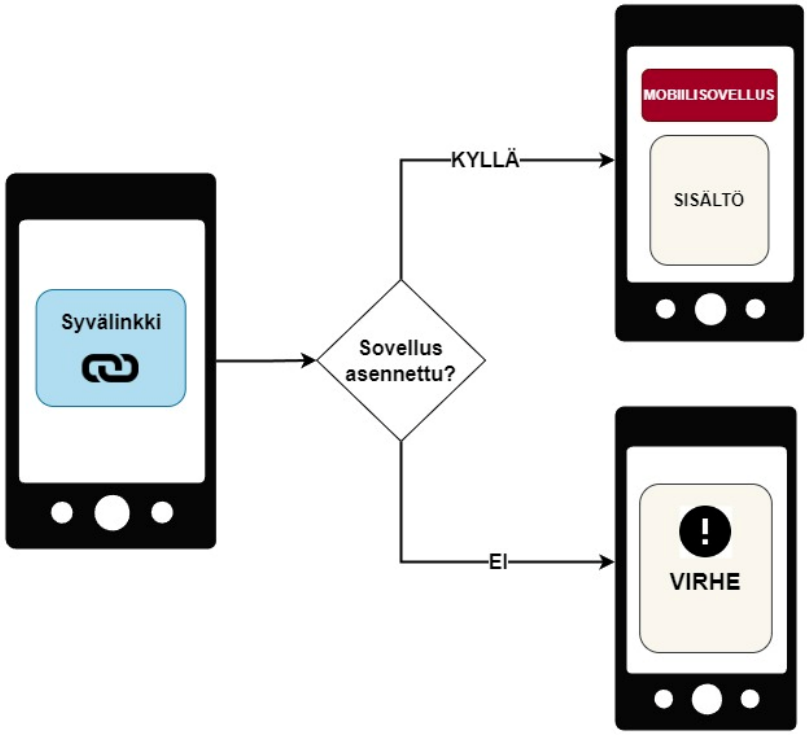
Kun QR-koodin luomista erilaisten ohjelmakirjastojen avulla tutkittiin, kiinnitettiin erityistä huomiota QR-koodin ulkonäön mukauttamiseen. QR-koodista saadaan paremmin massasta erottuva sen värejä ja pisteiden muotoja muokkaamalla sekä lisäämällä siihen esimerkiksi kuva tai jotain muuta informatiivista tietoa. Ohjelmointikielen valinnan pohjaksi testattiin eri ohjelmointikieliä sekä niiden ohjelmakirjastoja. Testauksen aikana kävi selväksi, että Python ohjelmointikielelle on kehitetty useita monipuolisia ohjelmakirjastoja QR-koodin luomiseen. Esimerkiksi Java-kielelle, joka on yksi maailman suosituimmista ohjelmointikielistä, on kehitetty yllättävän vähän ohjelmakirjastoja QR-koodin generointiin. Javan suosituin QR-koodin generointiin tarkoitettu ohjelmakirjasto on nimeltään ZXing, joka tarjoaa kuitenkin vain rajallisesti mahdollisuuksia QR-koodin personointiin. (*ZXing Project*, 2011/2023)

Pythonin ohjelmakirjastoista kaksi nousi ylitse muiden monipuolisuutensa ansiosta. Ohjelmakirjastot ovat nimeltään **qrcode** ja **Segno**, jotka molemmat sisältävät paljon valmiiksi sisäänrakennettuja työkaluja QR-koodin mukauttamiseen ja personointiin. Molemmat näistä kirjastoista olisivat sopineet hyvin QR-koodigeneraattorin kehittämiseen, mutta lopulta toteutukseen valittiin Segno-ohjelmakirjasto, koska se ei tarvitse toimiakseen mitään muita kolmannen osapuolen ohjelmakirjastoja. (*Comparison of Python QR Code libraries*, 2022)

3 Mobiilisovelluksen syvälinkit

Mobiilisovelluksen yhteydessä **syvälinkillä** (englanniksi deep link) tarkoitetaan linkkiä, joka ohjautuu johonkin tiettyyn sisältöön tietyssä sovelluksessa. Kun käyttäjä klikkaa mobiilisovellukseen kytkettyä linkkiä, avautuu sovellus ja sen sisältö suoraan käyttäjän näytölle, mikäli sovellus on asennettuna (Kuva 5). (Huang, 2020)

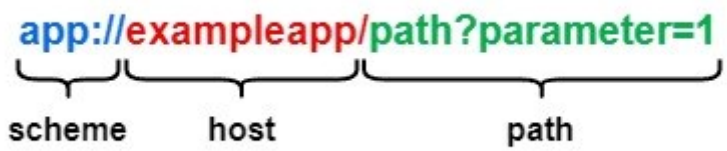
Kuva 5 Syvälinkin toiminta mobiilisovelluksessa.



3.1 Syvälinkit Android-sovelluksessa

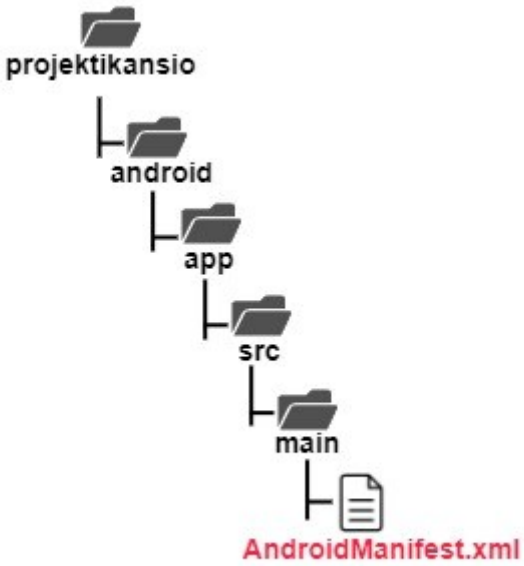
Android käyttöjärjestelmässä tiettyyn sovellukseen kytketyistä syvälinkeistä käytetään myös nimitystä **App Links**. Linkki on käytännössä merkkijono, jolla identifioidaan tietyn resurssin sijainti. Tällaista merkkijonoa kutsutaan nimellä **URI** (Unified Resource Identifier). Android-sovelluksen syvälinkin URI koostuu kolmesta eri osasta (Kuva 6 Android-sovelluksen syvälinkin rakenne). Näistä käytetään nimityksiä **scheme, host** ja **path**. (*App Manifest Overview*, 2023)

Kuva 6 Android-sovelluksen syvälinkin rakenne.



Syvälinkki määritellään sovelluksen **AndroidManifest.xml**-tiedostossa, joka on pakollinen tiedosto jokaisessa Androidin ohjelmistoprojektissa. Tiedoston avulla annetaan Android käyttöjärjestelmälle ja Google Play -palvelulle tärkeää tieto sovelluksesta, kuten tietoa sovelluksen eri komponenteista, käyttöoikeuksista sekä sovelluksen laitteisto- ja ohjelmistovaatimuksista. Tiedosto sijaitsee Android-projektin lähdekoodin juuritason kansiossa (Kuva 7). Esimerkiksi React Native -projektissa tiedosto löytyy kansiosta **/android/app/src/main/** (*App Manifest Overview*, 2023; Reidt, 2022)

Kuva 7 AndroidManifest.xml -tiedoston sijainti.



Syvälinkin toiminta toteutetaan määrittelemällä AndroidManifest.xml -tiedostoon niin sanottu **intent-filter-elementti**. Tämän elementin sisällä määritellään linkille sallitut toiminnot ja kategoriat sekä linkin URI-osoitteen attribuutit. (Ohjelmakoodi 1). Tässä tapauksessa sovellus käynnistyy, jos klikattava syvälinkki on muodossa **app://exampleapp/parameter/**. Yksi sovellus voi sisältää useita erilaisia intent-filter-elementtejä. (*Create Deep Links to App Content*, 2023)

Ohjelmakoodi 1 Intent-filter-elementti.

```
53    <intent-filter android:autoVerify="true">
54
55      <action android:name="android.intent.action.VIEW" />
56      <category android:name="android.intent.category.DEFAULT" />
57      <category android:name="android.intent.category.BROWSABLE" />
58
59      <data android:scheme="app"
60            android:host="exampleapp"
61            android:path="/parameter/" />
62
63    </intent-filter>
```

3.2 Syvälinkin heikkoudet

Syvälinkin suurin ongelma piilee siinä, ettei se osaa ohjata käyttäjää eteenpäin, jos kyseistä sovellusta ei olekaan asennettu käyttäjän laitteelle. Tässä tapauksessa käyttäjälle näytetään ainoastaan jonkinlainen virheilmoitus. Tämä ei ole käyttäjäystävällisyyden tai jatkuvuuden kannalta hyvä asia. Tämän lisäksi Applen iOS-käyttöjärjestelmässä syvälinkkien URI-osoitteen muoto on erilainen kuin Android-järjestelmässä. Tämä aiheuttaa ongelmia, koska käytännössä molemmille järjestelmille täytyy muodostaa omanlaiset linkkinsä. Yksi ratkaisu tähän ongelmaan on käyttää tavallisen syvälinkin sijaan dynaamista linkkiä.

4 Dynaamiset linkit

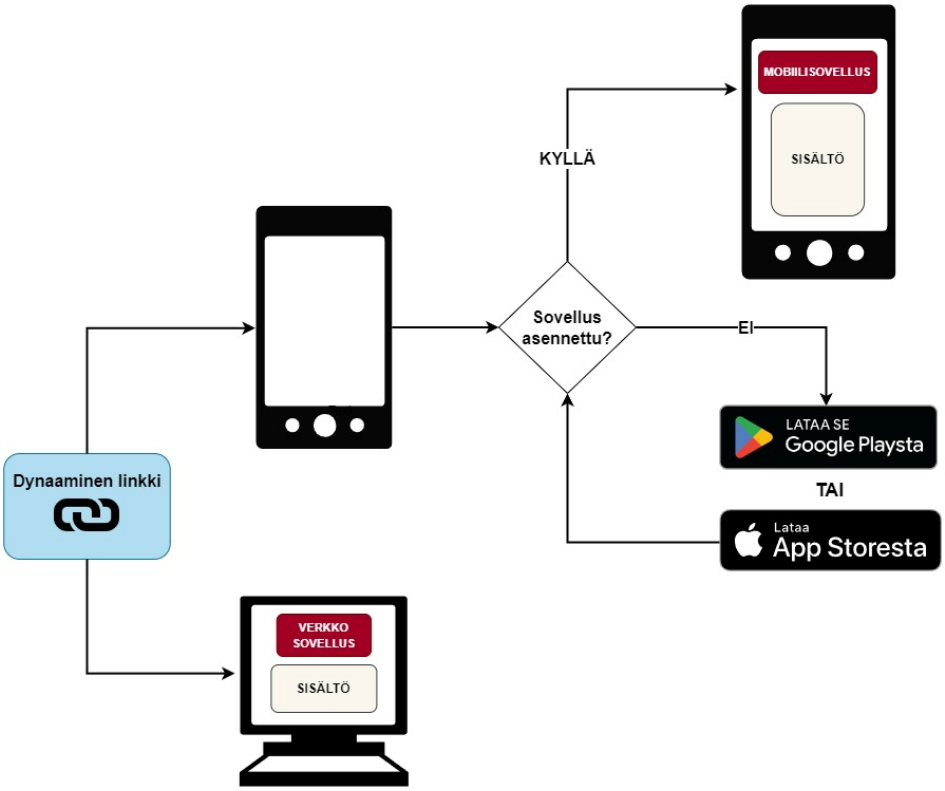
Erilaiset klikattavat linkit ovat tuttuja melkeinpä kaikille, jotka ovat jollain tavalla olleet tekemisissä tietokoneiden tai mobiililaitteiden kanssa. Internetin ja laitteiden kehitys toi ensin mukanaan hyperlinkit, jotka ohjasivat käyttäjän erilaisten sisältöjen pariin verkkosivuilla. Aikaisemmin linkit ovat olleet usein vain staattisia eli ne ohjaavat käyttäjän aina samaan paikkaan, mutta myös linkkien teknologia on kehittynyt ajan myötä. Nykyään on olemassa myös **dynaamisia** linkkejä, jotka voidaan ohjelmoida ohjaamaan käyttäjiä samasta linkistä erilaisiin sisältöihin.

4.1 Dynaamisen linkin toimintaperiaate

Dynaamisella linkillä tarkoitetaan ohjelmoitavaa linkkiä, joka voidaan määritellä toimimaan eri tilanteissa eri tavoilla. Dynaaminen linkki näyttää ulospäin kaikille käyttäjille samanlaiselta, mutta ohjaa käyttäjän eri paikkaan sen mukaan, miltä alustalta se on avattu. Jos käyttäjällä on linkin asetuksissa määritelty sovellus asennettuna, ohjataan käyttäjä sovellukseen, muuten käyttäjä ohjataan esimerkiksi suoraan paikkaan, josta sovellus voidaan asentaa laitteelle. (*Understanding Dynamic Links*, 2021)

Kuva 8 on esitelty dynaamisen linkin toimintaa. Kun käyttäjä klikkaa linkkiä mobiilialustalta, ohjataan hänet linkkiin ohjelmoituun mobiilisovellukseen ja sisältöön. Mikäli käyttäjällä ei ole sovellusta asennettuna, hänet ohjataan joko Google Play- tai App Store-palveluun sen mukaan, mikä käyttöjärjestelmä hänen mobiililaitteessaan on käytössä. Mobiilisovelluksen asennuksen jälkeen käyttäjä ohjautuu suoraan linkissä määriteltyyn sovellukseen ilman, että käyttäjän tarvitsee uudelleen klikata linkkiä. Jos käyttäjä puolestaan klikkaa linkkiä tietokoneella, ohjataan hänet tiettyyn sisältöön verkkosovelluksessa.

Kuva 8 Dynaamisen linkin toiminta.



Yksi dynaamisen linkin tuomista eduista on myös se, että mobiililaitteessa linkin sisältö säilyy sovelluksen asennusprosessin yli. Tämän ansiosta käyttäjä pystytään ohjaamaan vielä sovelluksen asennuksen jälkeenkin haluttuun sisältöön.

4.2 Alustoja dynaamisen linkin hallinnointiin

Yritykset tarjoavat dynaamisten linkkien luomisen ja hallinnointiin erilaisia alustoja. Nämä alustat sisältävät usein myös monia muita ominaisuuksia esimerkiksi linkkien jakamiseen ja käyttäjästatistiikan keräämisen sekä analysointiin. Alustat ovat lähes poikkeuksetta myös jollain tasolla maksullisia (Taulukko 3). Hinnat määräytyvät usein aktiivisten käyttäjien määrän mukaan. Hinnat eri palveluiden välillä vaihtelevat muutamista kymmenistä eurosta useisiin satoihin euroihin kuukaudessa. Kaikki yritykset ovat myös kehittäneet jonkin oman tavaramerkityn nimensä dynaamisille linkeille, mutta käytännössä ne kaikki kuitenkin toimivat samanlaisten periaatteiden mukaan. (AppsFlyer, 2023; Branch, 2023; GetSocial, 2023; *Google Firebase*, 2023)

Taulukko 3 Dynaamisten linkkien hallinnointiin soveltuvia alustoja.

Alusta	Verkkosivut	Maksullinen	Dynaamiset linkit
Get Social	https://www.getsocial.im/	Kyllä	Smart Link
Branch	https://www.branch.io/	Kyllä	NativeLink
Google Firebase	https://firebase.google.com/	Kyllä	Dynamic Links
AppsFlyer	https://www.appsflyer.com/	Kyllä	OneLink

Erilaisten alustojen hallinta on usein melko monimutkaista ja aikaa vievää. Kaikille alustoille on kehitetty paljon erilaisia ominaisuuksia sekä toimintoja, joista monet ovat itse dynaamisten linkkien hallinnoinnin kannalta tarpeettomia.

Googlen Firebase-alustan etuihin kuuluvat muun muassa dynaamisten linkkien hallinnoinnin maksuttomuus sekä hyvin toimiva ja selkeä REST-rajapinta. Rajapinnan avulla dynaamisten linkkien luominen ohjelmallisesti on sujuvaa lähestulkoon kaikilla yleisimmillä ohjelmointikielillä. Firebasen REST-rajapinnan rajoituksena on ainoastaan palvelinpyyntöjen rajaaminen viiteen pyyntöön sekunnissa ja 200 000 pyyntöön päivässä per IP-osoite, mikä tämän projektin puitteissa on täysin riittävä. Firebase on myös suhteellisen helppo integroida mobiilisovellukseen. Näiden ominaisuuksien vuoksi Firebase-alusta valittiin osaksi tämän opinnäytetyön käytännön osuuden toteuttamista. (*Google Firebase*, 2023)

5 Python ja React Native Framework

Tässä luvussa esitellään erilaisia opinnäytetyön käytännön osuuden toteutuksessa käytettyjä teknologioita. Ensin tutustutaan Python- ja JavaScript-ohjelmointikielten perusteisiin sekä Pythonille kehitettyyn Flask-ohjelmakirjastoon. Tämän jälkeen esitellään perusteita ohjelmointirajapintojen toteuttamiseen kehitetystä REST-arkkitehtuurimallista sekä mobiilisovellusten kehittämiseen soveltuvasta React Native -sovelluskehyksestä.

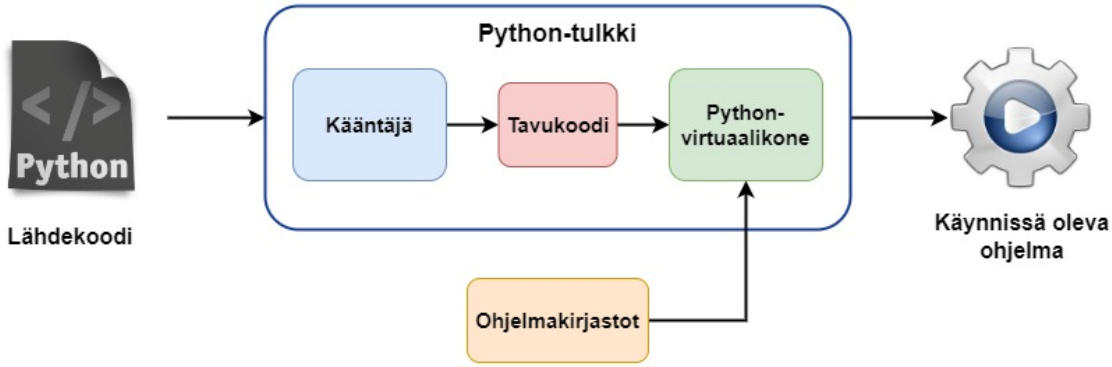
5.1 Python-ohjelmointikieli

Python on korkean tason, rakenteinen, oliopohjainen avoimen lähdekoodin ohjelmointikieli. Sen kehitti alun perin Guido Van Rossum vuonna 1991. Hän kehitti kielen yksinkertaisen ABC-ohjelmointikielen pohjalta. Van Rossum nimesi ohjelmointikielen englantilaisen 1970-luvun TV-sarjan ”Monty Python’s Flying Circus” mukaan. (Peltomäki, 2023, s. 6)

Pythonilla kirjoitetulle ohjelmalle ominaista ovat koodin selkeys ja helppolukuisuus. Pythonilla kirjoitettu koodi on hyvin jäsenneltyä ja suhteellisen helppoa ylläpitää. Pythonia suositellaankin usein ensimmäiseksi ohjelmointikieleksi, koska sen lähdekoodi on selkeämmin tulkittavissa verrattuna useisiin muihin ohjelmointikieliin. Pythonin etuihin kuuluu myös sen maksuttomuus, joten erillisille lisensseille ei ole tarvetta edes kaupallisessa käytössä. (Peltomäki, 2023, s. 6)

Python on niin sanottu tulkattava kieli. Python-kielellä kirjoitetun ohjelman suorittamiseksi käynnistettään erityinen **Python-tulkki**, joka lukee koodin ja toteuttaa sen käskyt. Python-tulkki täytyy asentaa järjestelmään ennen Python-ohjelman suorittamista. Python-tulkki kääntää lähdekoodin ensin **tavukoodiksi**, joka sitten suoritetaan **Python-virtuaalikoneessa** (Kuva 9). Tämän opinnäytetyön kirjoittamisen aikaan Pythonin uusin aliversio on 3.11.2, joka pohjautuu vuonna 2008 ensimmäistä kertaa julkaistuun **Python 3** -pääversioon. (Peltomäki, 2023, s. 7)

Kuva 9 Python-tulkin toiminta.



Ohjelmakirjasto on kokoelma, joka usein koostuu aliohjelmista, luokista ja/tai ohjelmista. Kirjastoja käytetään apuna tietokoneohjelmien modulaarisessa kehittämisessä. Usein riittää, että koodissa vain viitataan kirjastoon, jolloin se tulee automaattisesti sisällytettyä osaksi ohjelmaa. Ohjelmakirjaston sisältämät toiminnot ovat tällöin suoraan käytettävissä ohjelmassa. Tämä säästää aikaa ja lisää tehokkuutta, kun samaa koodia ei tarvitse kirjoittaa aina uudelleen eri ohjelmille. Pythonissa on tuki useille erittäin monipuolisille ohjelmakirjastoille, kuten esimerkiksi **Pandas, SciPy, Numpy, PyTorch, Flask ja MathPlotLib**. Osa ohjelmakirjastoista on Pythonissa sisäänrakennettuina ja osa on niin sanottuja kolmannen osapuolen kirjastoja, jotka vaativat erillisen asennuksen järjestelmään. (Peltomäki, 2023, s. 105)

Python-kielen syntaksissa koodirivien sisennys on merkittävässä roolissa. Python käyttää sisennystä määritelläkseen erillisen **koodilohkon**. Useimmissa muissa ohjelmointikielissä koodilohkojen rajaamiseen käytetään usein aaltosulkumerkkejä. Koodirivien perässä ei myöskään käytetä puolipistettä, kuten esimerkiksi Java- tai C-ohjelmointikielissä (Ohjelmakoodi 2). (Peltomäki, 2023, s. 14; ”Python Syntax”, 2023)

Ohjelmakoodi 2 Esimerkki Python-koodin rakenteesta.

```
i = 100
if i > 10:
    print("i on suurempi kuin 10")
else:
    print("i on pienempi kuin 10")
```

Pythonin ominaisuuksien ja monipuolisten kirjastojen ansiosta sitä käytetään usein esimerkiksi data-analytiikassa, koneoppimisessa, web-kehityksessä, automatisoinnissa, skriptauksessa, ohjelmistotestauksessa ja prototyyppien kehityksessä. (Dillemuth, 2023)

5.2 Flask

Flask on Python-ohjelmointikielelle kehitetty kevyt kirjasto, jonka avulla voidaan kehittää verkkosovelluksia. Flaskin kehitti Armin Ronacherin johtama Python harrastajien joukko nimeltä Pocoo. Sen ensimmäinen versio julkaistiin vuonna 2010. Nykyään Flaskin kehitystyöstä ja ylläpidosta vastaa ryhmä nimeltä The Pallets Projects. Flaskia kutsutaan usein myös **mikrosovelluskehykseksi**, koska siihen ei ole sisäänrakennettuina yleisiä täydellisen verkkosovelluskehyksen toimintoja, kuten käyttäjätilien hallintaa, käyttäjien autentikointeja, tietokantojen hallintaa tai syötteiden validointeja. Flaskin ydintoiminnot on haluttu pitää yksinkertaisina, mutta sen avulla kehitetyt sovellukset ovat kuitenkin helposti laajennettavissa isompaankin mittakaavaan. (*Welcome to Flask*, 2010; *What is Flask Python - Python Tutorial*, 2021)

Ronacher tiimeineen kehitti Flaskin perustaksi kirjaston nimeltä Werkzeug, jonka toiminta perustuu Pythonin WSGI-standardiin. WSGI on Pythonin standardi, joka määrittää sen, kuinka verkkopalvelin kommunikoi verkkosovelluksen kanssa ja kuinka verkkosovelluksia voidaan ketjuttaa yhteen palvelinpyyntöjä käsiteltäessä. WSGI on lyhenne englanninkielisistä sanoista Web Server Gateway Interface. (*Werkzeug*, 2023; *WSGI.org*, 2023)

Flaskilla toteutetun ohjelman rakenne on perustasolla hyvinkin yksinkertainen ja suhteellisen helppo ymmärtää. Ohjelmakoodi 3 on esitetty yksinkertainen verkkosovellus, joka on toteutettu Flask-kirjaston avulla.

Ohjelmakoodi 3 Flaskilla toteutetun ohjelman rakenne.

```
# tuodaan Python-ohjelmaan flask-kirjastosta Flask-luokka
from flask import Flask

# luodaan Flask-luokan ilmentymä eli olio nimeltä app
app = Flask(__name__)

# Kerrotaan Flaskille mitä osoitetta seuraava funktio vastaa ja millä metodeilla
# funktiota voidaan kutsua.
@app.route('/sayhello', methods=['GET'])
def hello_world():
    return 'Hello World!'
```

Jos ohjelma saa kutsun verkkosivulta **HTTP-protokollan GET**-metodilla palvelinpyynnön osoitteeseen **/sayhello**, palauttaa se merkkijonon ”Hello World!”, joka tulostuu verkkosivulle.

5.3 JavaScript-ohjelmointikieli

JavaScript on oliopohjainen komentosarja- eli **skriptikieli**. Sitä käytetään yleensä erilaisten dynaamisten toiminnallisuuksien toteuttamiseen verkkosivuilla. JavaScriptin avulla pystytään esimerkiksi muokkaamaan verkkosivujen HTML- ja CSS-koodeja. JavaScriptin kehitti alun perin Brendan Eich vuonna 1995. Hän kehitti ohjelmointikielen osana Netscape 2 -verkkoselaimen kehitystyötä. Vuonna 1997 JavaScript hyväksyttiin osaksi verkkosivujen standardeja. (Dickson, 2022)

JavaScriptin syntaksi pohjautuu C-ohjelmointikieleen, mutta siinä on vaikutteita myös Self- ja Scheme-ohjelmointikielistä. JavaScript sekoitetaan usein Java-ohjelmointikieleen, mutta ne ovat täysin eri ohjelmointikieliä. Java-sanan käyttö nimessä on peräisin Netscapen ja Javan kehittäneen Sun-yhtiön lisenssisopimuksesta sekä markkinointitaktiikasta, jonka tavoitteena oli markkinoida JavaScriptiä Javan apukielenä. JavaScript on kehittynyt ajan myötä ja nykyään sitä voidaan käyttää monipuolisemmin esimerkiksi palvelinpuolen ohjelmoinnissa. JavaScriptille on kehitetty myös lukuisia suosittuja sovelluskehyksiä, kuten **Angular, jQuery, React, Vue** ja **Node.js**. Monipuolisuutensa ansiosta JavaScript onkin noussut maailman

suosituimmaksi ohjelmointikieleksi sovelluskehittäjien keskuudessa. (Dickson, 2022; ”History of JavaScript”, 2022; *Statista*, 2022)

JavaScript on Pythonin tavoin niin sanottu tulkattava ohjelmointikieli eli se tarvitsee toimiakseen ympäristön, joka sisältää **JavaScript-tulkin**. Se tulkkaa ja suorittaa koodin rivi kerrallaan. Sittemmin tulkista on alettu käyttää myös nimeä **JavaScript-moottori**. JavaScript on rakenteeltaan **heikosti tyypitetty** ohjelmointikieli. Tämä tarkoittaa, että käytössä olevia tietorakenteita ei erikseen määritellä, vaan JavaScript pyrkii muuttamaan tyyppejä automaattisesti virheiden välttämiseksi. Tämä aiheuttaa myös ongelmia, koska koodissa olevien virheiden jäljittäminen vaikeutuu virheilmoitusten puuttuessa. JavaScriptin rinnalle onkin kehitetty vahvasti tyypitetty ohjelmointikieli nimeltä TypeScript. (Tarvainen, 2016)

5.4 React Native Framework

React Native on avoimeen lähdekoodiin perustuva sovelluskehys, jonka avulla voi kehittää natiiveja mobiilisovelluksia iOS- ja Android-alustoille. React Nativen kantavana ideana on ajatus siitä, että samalla koodipohjalla voidaan toteuttaa sovellus molemmille alustoille, eikä näin ollen tarvita kahta täysin erillistä ohjelmistoprojektia. React Nativen kehitti alun perin Meta (entinen Facebook) ja ensimmäinen virallinen versio siitä julkaistiin vuonna 2015. (Masiello & Friedmann, 2017, s. 45; Ruokangas, 2023)

React Native on rakennettu JavaScript-ohjelmointikielen ja sille kehitetyn ReactJS-kirjaston päälle. ReactJS-kirjaston avulla voidaan kehittää dynaamisia verkkokäyttöliittymiä sekä niiden komponentteja. React Native, kuten myös ReactJS, käyttää JSX-syntaksia, joka muistuttaa hyvin pitkälti verkkosivujen luomiseen käytettävää HTML-koodia. JSX on lyhenne sanoista JavaScript XML. Se mahdollistaa dynaamisen sisällön luomisen kirjoittamalla JavaScript-koodia suoraan React Native -koodiin aaltosulkeiden sisään. React Nativella kehitetty sovellus koostuu komponenteista, joiden avulla määritellään sovelluksen käyttöliittymän sisältö. React Native sisältää useita sisäänrakennettuja nimettyjä komponentteja, kuten **Text, Button, View** tai **Image**. Näistä komponenteista on mahdollista rakentaa myös omia mukautettuja komponentteja JavaScript funktioiden avulla. Tällaisen funktion on aina palautettava jotain return-lauseen avulla. Komponentin nimi alkaa aina isolla kirjaimella ja se kirjoitetaan koodissa **kulmasulkeiden (< >)** väliin. Komponentti on

myös aina suljettava **vinoviivalla (/)**. React Nativelle on myös saatavilla kattava määrä erilaisia kolmannen osapuolen kirjastoja, joiden avulla sovelluksen kehittäminen on tehokkaampaa. (*React Native · Learn Once, Write Anywhere*, 2023)

Ohjelmakoodi 4 on esitettynä yksinkertainen mobiilisovellus, joka on toteutettu React Native sovelluskehyksellä. Koodin alussa ohjelmassa tarvittavat kirjastot ja komponentit on tuotu käyttöön (rivit 1–2). Itse varsinainen sovellus on **App**-niminen komponentti (rivit 4–11) eli käytännössä JavaScript-funktio, joka palauttaa **View**-komponentin, jonka sisällä on **Text**-komponentti (rivi 7) ja mukautettu komponentti nimeltään **Hello** (rivi 8). Hello-komponentti (rivit 13–15) on puolestaan JavaScript-funktio, joka palauttaa Text-komponentin. Riveillä 17–31 on määritelty muuttuja nimeltään **styles**, joka on puolestaan React Nativen **StyleSheet**-komponentti. Sen avulla voidaan määritellä ohjelmassa käytettyjen komponenttien tyylejä, esimerkiksi värejä, fonttikokoja ja kohdistuksia. Kuin ohjelma ajetaan, tulostaa se puhelimen näytön keskelle kaksi erilaista tekstiä eri väreillä (Kuva 10).

Ohjelmakoodi 4 Esimerkki React Native ohjelmakoodista.

```
1  import React from 'react';
2  import {StyleSheet, Text, View} from 'react-native';
3
4  const App = () => {
5    return (
6      <View style={styles.container}>
7        <Text style={styles.blue}>Hello React Native World!</Text>
8        <Hello />
9      </View>
10   );
11 };
12
13 const Hello=()=>{
14   return <Text style={styles.red}>Hello, there!</Text>
15 }
16
17 const styles = StyleSheet.create({
18   container: {
19     flex: 1,
20     justifyContent: 'center',
21     alignItems: 'center',
22   },
23    blue: {
24     color: 'blue',
25     fontSize: 26,
26   },
27   red: {
28     color: 'red',
29     fontSize: 26,
30   },
31 });
32
33 export default App;
34
```

Kuva 10 React Native -esimerkkiohjelma käynnissä mobiililaitteessa.

React Native on noussut nopeasti yhdeksi suosituimmista mobiilisovellusten kehitysympäristöistä. Sen etuja ovat kustannustehokkuus, koodin uudelleenkäyttömahdollisuus, koodiin tehtyjen muutosten reaaliaikainen päivittyminen, monipuoliset kirjastot sekä laajan kehittäjäyhteisön tuki. Useat tunnetut yritykset ovat toteuttaneet React Nativen avulla omia mobiilisovelluksiaan. Näihin yrityksiin lukeutuvat esimerkiksi AirBNB, Skype, Tesla, Meta, Microsoft, Puma, Foreca, Discord, Pinterest, Klarna ja Walmart. (Brewster, 2023; Showcase · React Native, 2023)

5.5 REST

REST on ohjelmointirajapintojen toteuttamiseen tarkoitettu arkkitehtuurimalli tai -tyyli. REST on lyhenne englanninkielisistä sanoista Representational State Transfer. Arkkitehtuurimallin kehitti ja esitteli vuonna 2000 väitöskirjassaan Roy Fielding, joka tunnetaan myös yhtenä alkuperäisen HTTP-protokollan kehittäjistä. Verkkoselaimet ja palvelimet käyttävät HTTP-protokollaa keskinäiseen tiedonsiirtoon. (Kivisaari, 2016)

REST määrittelee sen, kuinka dataa voidaan hyödyntää ja siirtää paikasta toiseen. REST-rajapintojen avulla erilaiset sovellukset voivat ”kommunikoida” keskenään. REST pohjautuu

HTTP-protokollaan ja sen metodeihin, joista käytetyimpiä ovat **GET, POST, PUT, UPDATE, PATCH** ja **DELETE**. REST-rajapinnassa määritellään juuriosoite eli mistä osoitteesta sen resursseja voidaan käsitellä, resurssien esitysmuodon määrittelevä mediatyyppi (esimerkiksi HTML-, JSON- tai XML-tiedostomuoto) ja millä HTTP-protokollan metodilla resursseja voidaan käsitellä. (Kivisaari, 2016)

6 Kehitysprojekti - case DigiMoi

Opinnäytetyön käytännön osuus toteutetaan kehitysprojektina, jonka tarkoituksena on vastata toimeksiantajan tarpeisiin sekä toimia tukena uuden sovelluksen kehitysprosessissa. Tarkoituksena on kartoittaa erilaisia keinoja dynaamisten linkkien ja QR-koodien luomiseen sekä hallinnointiin ja valita tämän jälkeen projektiin parhaiten sopivat teknologiat. Kehitysprojekti aloitetaan asiakkaan tarpeiden kartoituksella ja lopuksi toteutetaan erilaisiin käytännön kokeiluihin, testauksiin ja vertailuihin perustuen.

Dynaamisten linkkien osalta kehitysprojekti rajattiin koskemaan ainoastaan Android-sovelluksia, koska linkitykset iOS-sovelluksiin vaativat sekä Applen laitteita että maksullista Apple-kehittäjätiliä ja näitä investointeja ei koettu tarpeellisiksi vielä tässä vaiheessa kehitysprosessia.

6.1 Tausta

Calevala Interactive Oy on kehittämässä uutta digitaalista alustaa nimeltään DigiMoi. DigiMoi koostuu verkkoselaimen kautta ohjattavasta hallintaliittymästä sekä loppukäyttäjille suunnatusta mobiilisovelluksesta. Hallintaliittymän kautta voidaan luoda erilaisia kampanjoita, joihin DigiMoin mobiilikäyttäjät voivat halutessaan osallistua mobiilisovelluksen kautta.

Kampanjoiden tarkoituksena on kerätä kollektiivisesti dataa loppukäyttäjiltä kuvien muodossa. Kampanjaan osallistuvia käyttäjiä voidaan palkita riippuen kampanjan ehdoista. Kampanja voidaan rajata esimerkiksi tiettyyn maantieteelliseen sijaintiin, jolloin määritellyn alueen ulkopuolella olevat käyttäjät eivät voi osallistua kampanjaan.

6.2 Tavoitteet

Asiakkaan tavoitteena on saada kehitettyä DigiMoin hallintaliittymään kampanjan hallintasivulle ominaisuus, jolla voidaan luoda kampanjakohtainen dynaaminen linkki ja linkistä edelleen QR-koodi. Kun käyttäjä skannaa tämän QR-koodin mobiililaitteellaan, avautuu DigiMoin mobiilisovellus ja sieltä kyseisen kampanjan tiedot käyttäjälle. Jos

käyttäjällä ei ole DigiMoin mobiilisovellusta asennettuna, ohjataan hänet suoraan paikkaan, josta mobiilisovelluksen voi ladata. Sovelluksen asennusprosessin jälkeen linkkiin upotetun kampanjan tiedot avautuvat käyttäjälle. QR-koodin avulla mobiilisovelluksen käytöstä ja asennuksesta halutaan tehdä nopeampaa ja yksinkertaisempaa. QR-koodia on lisäksi helppo hyödyntää DigiMoi-sovelluksen markkinoinnissa, käyttäjäkannan kasvattamisessa sekä jo olemassa olevien käyttäjien sitouttamisessa.

7 Google Firebase ja dynaamiset linkit

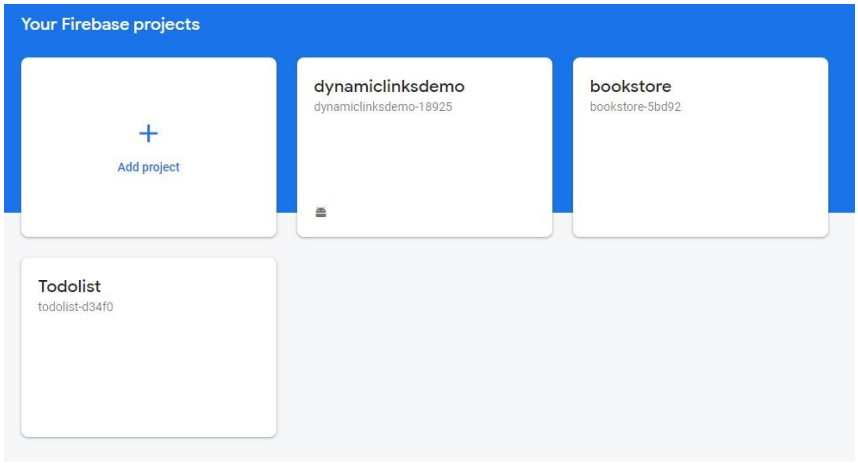
Firebase on Googlen ylläpitämä, erityisesti mobiilisovellusten kehittämiseen soveltuva alusta. Firebase toimii **PaaS**-mallilla (**Platform-as-a-Service**). Tämä tarkoittaa, että Firebase sisältää valmiina kaiken sovellusten kehittämisessä tarvittavan infrastruktuurin. Käyttäjän ei tarvitse tehdä mitään erillisiä asennuksia, koska kaikki tarvittava hoidetaan verkkoselaimen kautta pilvipalvelun avulla. Tämän ansiosta Firebase-projektien hallinta ei ole myöskään sidottu tiettyyn paikkaan tai aikaan. (Stevenson, 2018)

Firebase sisältää useita ohjelmistokehitykselle keskeisiä ominaisuuksia, kuten esimerkiksi autentikoinnin, tietoturvallisen käyttäjätietojen hallinnan, integraatiot eri alustoille, datan analysointityökalut sekä tietokantojen hallinnan. Firebasen käytön aloittaminen on ilmaista, mutta datamäärän kasvaessa palvelun käyttö muuttuu tietyiltä osin maksulliseksi. Firebase sisältää kuitenkin useita ominaisuuksia, jotka ovat aina ilmaisia niiden käyttömäärästä riippumatta. Dynaamiset linkit kuuluvat juuri tähän täysin ilmaiseen kategoriaan. (*Google Firebase*, 2023)

7.1 Firebase-projektin luominen

Uuden Firebase-projektin luomista varten käyttäjän on ensin luotava ensin itselleen Google-tili. Tilin luomisen jälkeen käyttäjä voi kirjautua tunnuksillaan Firebase-alustalle osoitteessa https://firebase.google.com/. Etusivun oikeasta yläkulmasta löytyvästä linkistä päästään suoraan Firebasen konsoliin, jossa käyttäjä voi luoda ja hallita omia projektejaan (Kuva 11).

Kuva 11 Firebasen konsolinäkymä.

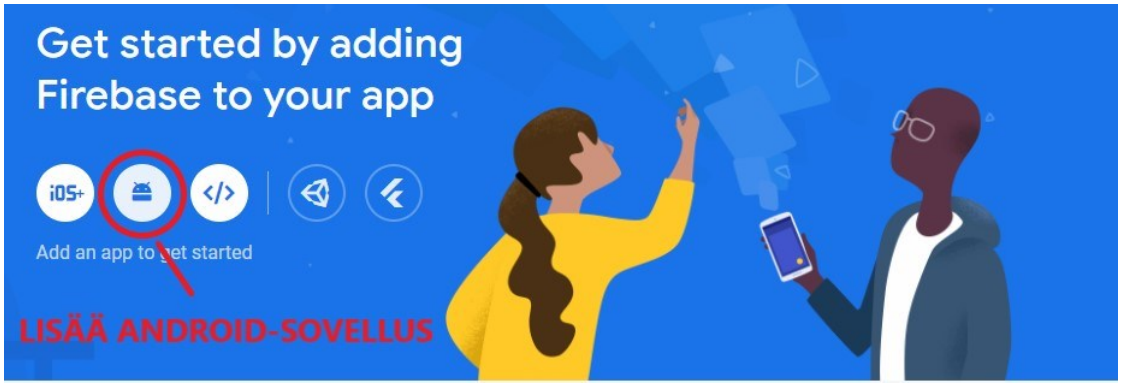


Uuden projektin luomisvaiheessa projektille luodaan myös uniikki tunnus sekä kytketään haluttaessa Googlen Analytics -palvelut käyttöön. Kun uusi Firebase-projekti on luotu, voidaan se valita klikkaamalla sitä Firebasen konsolinäkymästä.

7.2 Android-sovelluksen rekisteröinti Firebase-projektiin

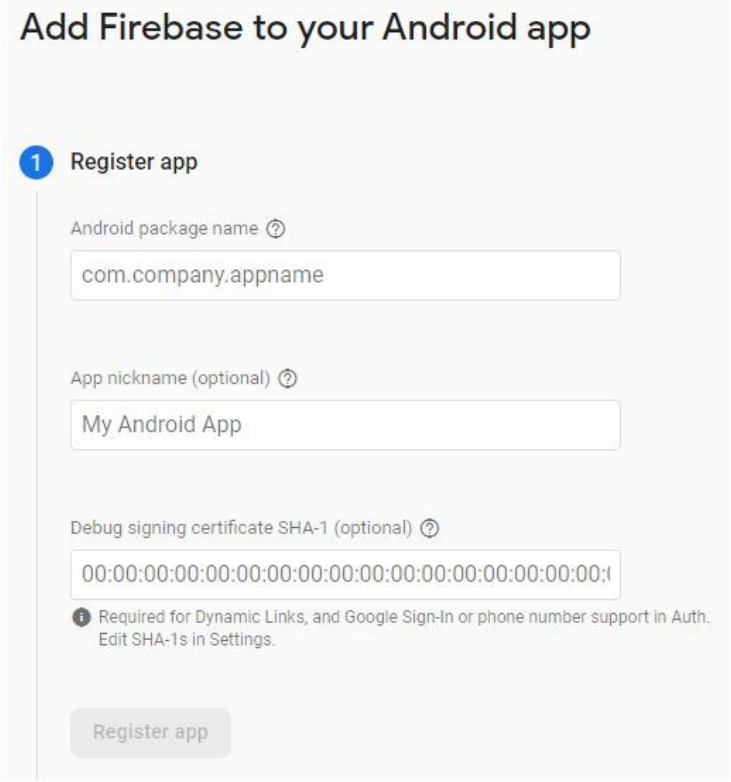
Kun Firebasen dynaamiseen linkkiin halutaan liittää jokin tietty Android-sovellus, on sovellus ensin lisättävä osaksi kyseistä Firebase-projektia. Android-sovelluksen rekisteröiminen Firebase-projektiin käynnistyy klikkaamalla Android-ikonia projektin hallintasivulta (Kuva 12).

Kuva 12 Android-sovelluksen lisääminen Firebase-projektiin (*Google Firebase*, 2023).



Tämän jälkeen avautuvassa näkymässä (Kuva 13) määritellään ensin sovelluksen uniikki tunniste, joka tunnetaan myös nimellä **Andoid package name** tai **applicationId**. Tunnus löytyy joko Android projektin AndroidManifest.xml -tiedostosta tai Google Play -palvelusta kyseisen sovelluksen URL-osoitteesta, jos sovellus on rekisteröity Google Play -palveluun. Sovelluksen URL-osoite Google Playssa on muotoa **https://play.google.com/store/apps/details?id=tämä.on.tunnus**. (*Firebase for Android*, 2023)

Kuva 13 Sovelluksen rekisteröinti Firebase-projektiin (*Google Firebase*, 2023).



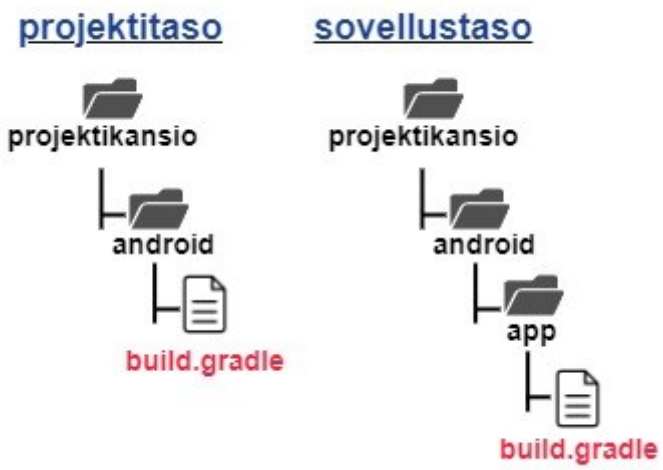
Tunnuksen lisäksi Android-sovellukselle voidaan määritellä oma nimi, joka näkyy vain Firebasen sisällä sekä niin sanotut **SHA-sormenjäljet**. SHA-sormenjälkiä käytetään tietotekniikassa tiedon oikeellisuuden sekä muuttumattomuuden varmentamiseen ja se luodaan erityisen matemaattisen algoritmin avulla. Sovelluksen nimimerkin ja SHA-sormenjälkien määritteleminen on oletusarvoisesti vapaaehtoista, mutta dynaamisten linkkien ja REST-rajapinnan toiminnan kannalta SHA-sormenjälkien määritteleminen on pakollista. Android-sovellukselle on määriteltävä **SHA-1**- ja **SHA-256**-sormenjäljet. Android-sovelluksen SHA-sormenjäljet voi selvittää esimerkiksi Javaan sisäänrakennetulla **keytool**-työkalulla tai Android-projektin **Gradle**-työkalun avulla. Kun vaaditut SHA-sormenjäljet on määritelty, voidaan sovellus rekisteröidä painamalla **Register app** -painiketta. (*React Native Firebase*, 2023)

7.3 Firebasen lisääminen Android-sovellukseen

Android-sovelluksen rekisteröinnin jälkeen Firebasen konsolista avautuu näkymä, josta voidaan ladata **google.services.json**-niminen tiedosto. Tämä tiedosto on Android-sovellukselle tarkoitettu Firebasen konfigurointitiedosto JSON-muodossa. JSON-tiedosto

täytyy kopioida Android-projektin **sovellustason** kansioon **(/android/app**). Tämän lisäksi Android-projektin **build.gradle**-tiedostoihin on tehtävä joitakin lisäyksiä, jotta **google-services**-laajennusosa voidaan ottaa käyttöön kyseisessä Android-projektissa. Android-projektissa on kaksi build.gradle-tiedostoa. Toinen niistä sijaitsee **projektitasolla** ja toinen **sovellustasolla** (Kuva 14). (*React Native Firebase*, 2023)

Kuva 14 Android-projektin build.gradle -tiedostojen sijainnit.



Sovellustason Gradle-tiedoston syntaksi riippuu käytettävästä kehitysympäristöstä sekä ohjelmointikielestä. **Virhe. Kirjanmerkin viittaus itseensä ei kelpaa.** jaOhjelmakoodi 6 on esitetty React Native -projektin Gradle-tiedostoihin tarvittavat muutokset.

Ohjelmakoodi 5 Lisäys projektitason Gradle-tiedostoon (*React Native Firebase*, 2023).

```
buildscript {
  dependencies {
    // ... other dependencies
    classpath 'com.google.gms:google-services:4.3.15'
    // Add me --- /\
  }
}
```

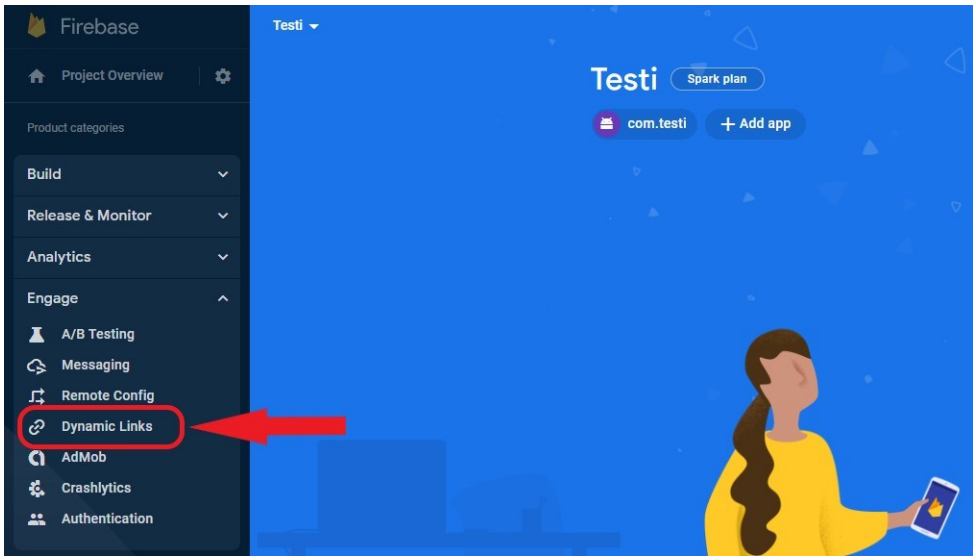
Ohjelmakoodi 6 Lisäys sovellustason Gradle-tiedostoon (*React Native Firebase*, 2023).

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services' // <- Add this line
```

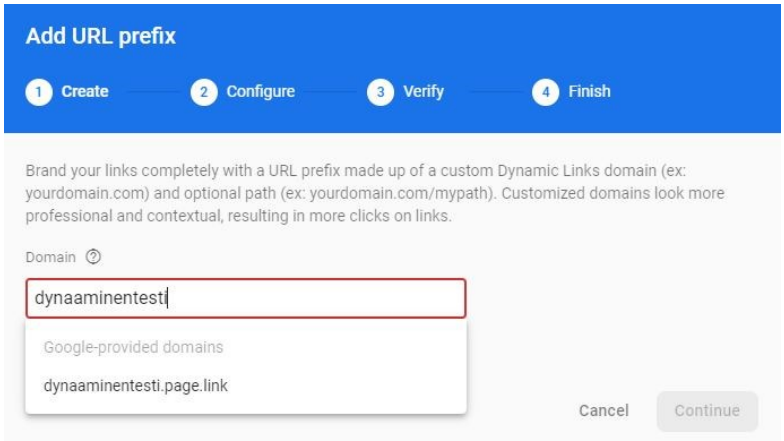
7.4 Dynaamisen linkin luominen Firebasen konsolissa

Dynaaminen linkki luodaan Firebase-projektiin vasemmalta löytyvän **Product categories** -valikon kohdasta **Engage** => **Dynamic Links** (Kuva 15). Tämän alta aukeavassa näkymässä (Kuva 16) määritellään dynaamisen linkin verkkotunnus eli **URL-osoitteen alkuosa**, joka on **uniikki** ja **pysyy aina samana** jokaisessa sen alle luodussa dynaamisessa linkissä. Firebase tarjoaa tähän omaa verkkotunnustaan, mutta käyttäjän on mahdollista määritellä tähän myös kokonaan oma verkkotunnus. Käytettävä verkkotunnus ei voi kuitenkaan olla mikä tahansa, vaan sen täytyy toimia Firebasen Hosting-palvelun alla. Erilaisia verkkotunnuksia (**URL prefix**) voi liittää yhteen Firebase-projektiin enintään kymmenen. (*Google Firebase*, 2023)

Kuva 15 Dynaamisen linkin lisääminen (*Google Firebase*, 2023).

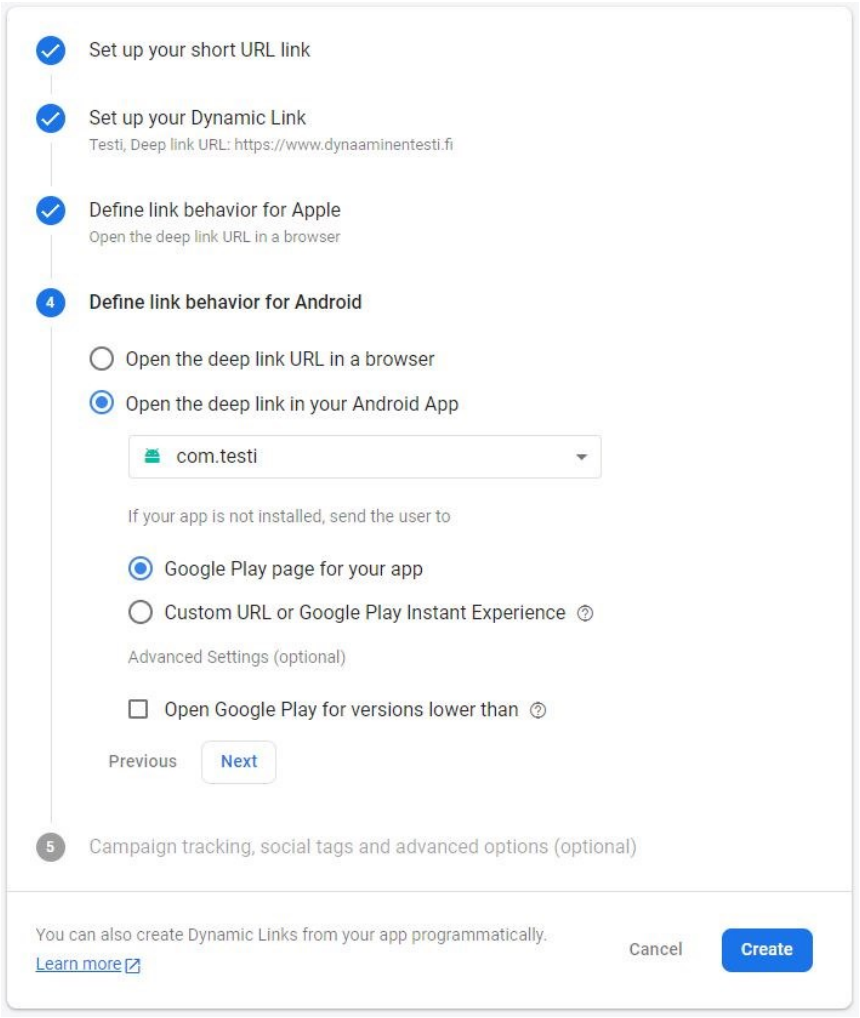


Kuva 16 Dynaamisen linkin URL-osoitteen määrittäminen (*Google Firebase*, 2023).



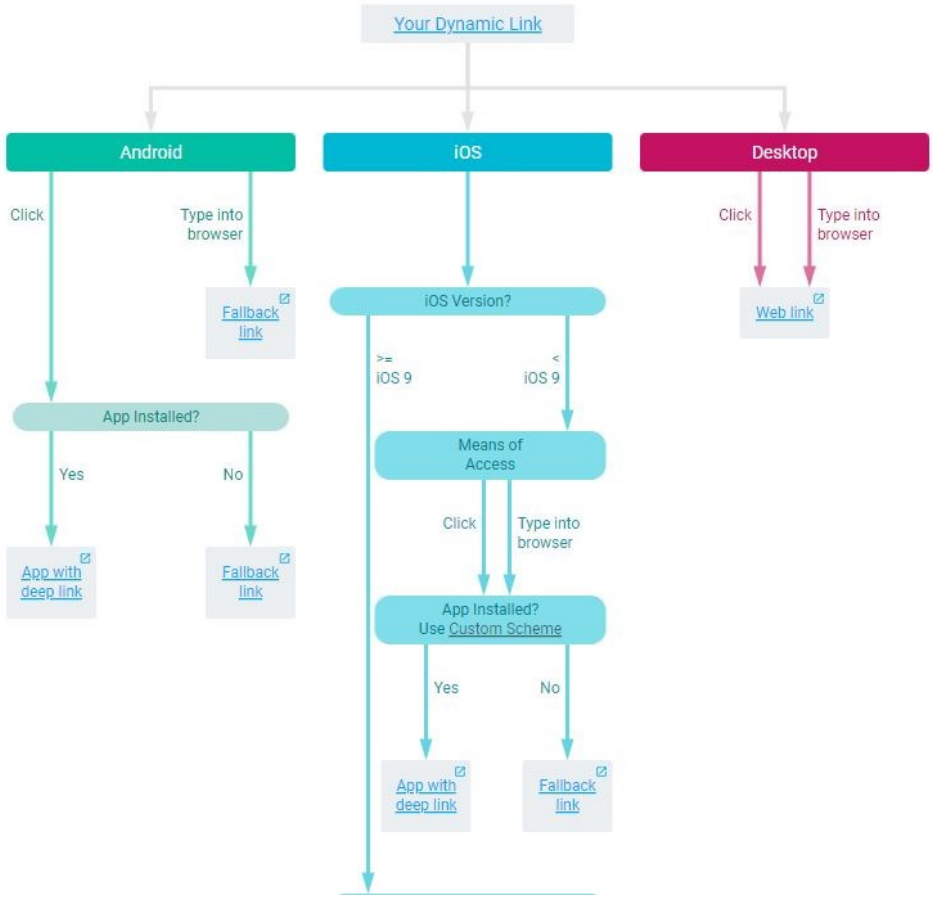
Kun dynaamisen linkin verkkotunnus on määritelty, voidaan aloittaa varsinaisen dynaamisen linkin luominen. Yhteen verkkotunnukseen voi luoda käytännössä rajattoman määrän dynaamisia linkkejä. Kun dynaaminen linkki luodaan, sille määritellään ensin niin sanottu lyhyt URL-osoite, jossa linkin verkkotunnuksen perään lisätään jokin tunniste. Näistä kahdesta osasta muodostuu dynaamisen linkin varsinainen käyttäjälle näkyvä URL-osoite (esimerkissä https://dynaaminentest.page.link/home). Tämän jälkeen määritellään dynaamisen linkin toiminta eri tilanteissa (Kuva 17). Ensimmäiseksi määritellään URL-osoite, johon dynaaminen linkki oletusarvoisesti ohjautuu. Sen jälkeen voidaan määritellä linkin toiminta erikseen Apple- ja Android-pohjaisille laitteille. Linkki voidaan määrittää avaamaan mikä tahansa sovellus, joka on rekisteröity samaan Firebase-projektiin. Dynaamiselle linkille voidaan myös määritellä esikatseluotsikko ja -kuva sosiaalista mediaa varten. (*Google Firebase*, 2023)

Kuva 17 Dynaamisen linkin luominen ja toiminnot (*Google Firebase*, 2023).



Kun kaikki dynaamisen linkin halutut toiminnot on määritelty, luodaan linkki painamalla **Create**-nappia ja dynaaminen linkki näkyy tämän jälkeen listauksessa. Dynaamisen linkin toimintaa voi tarkastella tarkemmin valitsemalla listauksesta rivin oikeasta reunasta löytyvän kolmen pisteen takaa avautuvasta valikosta kohdan **Link preview (debug)**. Tämä avaa visuaalisen kaavion dynaamisen linkin toiminnasta (Kuva 18). Tässä näkymästä löytyvät myös mahdolliset varoitukset dynaamisen linkin toimintojen virheistä. (*Google Firebase*, 2023)

Kuva 18 Visuaalinen kaavio dynaamisen linkin toiminnasta (*Google Firebase*, 2023).



8 QR-koodin generointiohjelma

QR-koodin generointiohjelma toteutetaan kokonaisuudessaan Python-ohjelmointikielellä. Kaikki osa-alueet dynaamisen linkin ja QR-koodin luomisesta aina verkkosovelluksen toimintoihin toteutetaan samassa Python-kooditiedostossa, jotta ohjelmisto pysyisi mahdollisimman kompaktina. Tämä helpottaa myös sovelluksen integrointia DigiMoin hallintaliittymään.

8.1 Pythonin ja tarvittavien kirjastojen asennus

Pythonilla kirjoitetun ohjelman ajamista varten järjestelmään täytyy ensin asentaa Python-tulkki. Uusimman version ohjelmasta voi ladata Pythonin kotisivuilta osoitteesta https://www.python.org. Windows-järjestelmässä Pythonin asennuksen tilan ja version voi tarkastaa Windowsin komentokehotteesta tai PowerShell-ikkunasta (Komento 1).

Komento 1 Pythonin asennuksen ja version tarkastus.

```
python --version
```

Pythonin asennuksen jälkeen järjestelmään on asennettava kaikki QR-koodigeneraattorin tarvitsemat ohjelmakirjastot. Asennettavat kirjastot ovat nimeltään **Flask, Segno** ja **Pillow**. Flask-kirjastoa käytetään verkkosovelluksen toteuttamiseen, Segno-kirjastoa varsinaisen QR-koodin luomiseen ja Pillow-kirjastoa kuvan muokkaukseen. Ohjelmakirjastojen asentaminen tapahtuu Pythonin mukana tulevan paketinhallintasovelluksen (**pip**) avulla (Komento 2). Tämän lisäsi lopullisessa ohjelmassa tarvitaan myös joitain Pythoniin sisäänrakennettuja kirjastoja.

Komento 2 Pythonin ohjelmakirjaston asentaminen.

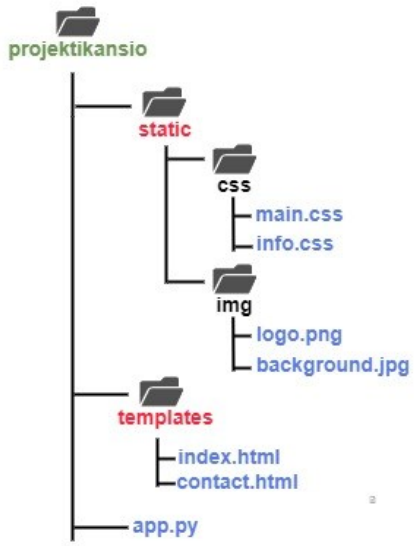
```
pip install Segno
```

8.2 QR-koodigeneraattorin verkkosovellus

QR-koodigeneraattorin verkkosovelluspuolen kehittämiseen valitsin Pythonin **Flask**-kirjaston, koska se vaikutti suhteellisen yksinkertaiselta ja mielenkiintoiselta. Flask sisältää myös sisäänrakennetun kehityspalvelimen, jonka avulla verkkosovellusta on helppo ajaa ja testata paikallisesti omalta tietokoneelta. Minulla ei ollut mitään aiempaa kokemusta Flaskista, joten sain samalla myös mahdollisuuden kasvattaa omaa tietotaitoani.

Kun verkkosovellusta kehitetään Flaskin avulla, on projektin hakemistorakenteen noudatettava ennalta määrättyä mallia (Kuva 19). Flask olettaa, että projektin päähakemistossa on kaksi tietyn nimistä alihakemistoa, **static** ja **templates**. Static-hakemisto sisältää kaikki verkkosovelluksen käyttämät materiaalit, kuten esimerkiksi kuvat, CSS- ja JavaScript-tiedostot. Templates-hakemisto puolestaan sisältää sovelluksen käyttämät HTML-tiedostot. Python-tiedostot sijaitsevat projektin juuritason hakemistossa. (*Flask Templates*, 2023)

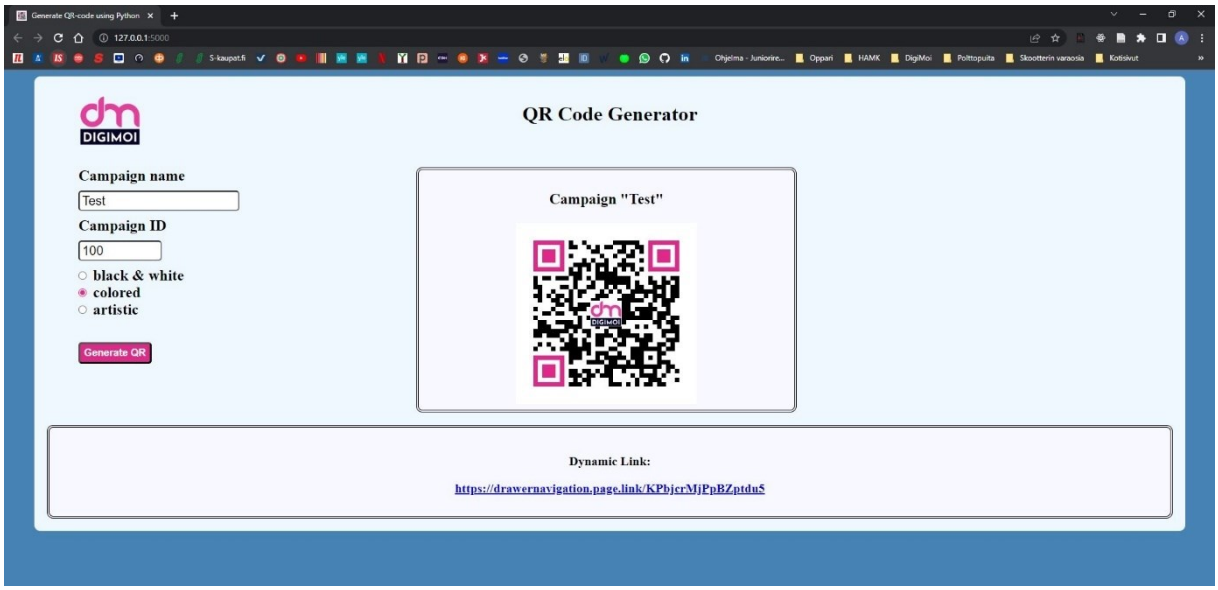
Kuva 19 Esimerkki Flask-projektin hakemistorakenteesta.



QR-koodigeneraattorin verkkosovellus koostuu kolmesta päätiedostosta: **HTML-tiedostosta** itse verkkosivun luomiseen, **CSS-tiedostosta** verkkosivun ulkoasun määrittelemiseen ja **Python-tiedostosta**, jonka avulla haetaan tarvittavat parametrit verkkosivulta, luodaan dynaaminen linkki ja QR-koodi sekä välitetään halutut tiedot takaisin verkkosivulle.

Kun sovellus käynnistetään, avautuu verkkoselaimesta QR-koodigeneraattorin käyttöliittymä eli index.html-sivu (Kuva 20). Käyttäjä voi syöttää ohjelmaan kampanjan nimen ja ID-tunnisteen sekä valita luotavan QR-koodin tyypin. Käyttäjän painaessa nappia, lomakkeen tiedot lähetetään Python-ohjelmalle, jonka jälkeen tietojen avulla luotu QR-koodi sekä dynaaminen linkki palautuvat takaisin verkkosivulle. Käyttöliittymän ulkoasu ja muotoilut on määritelty erillisessä CSS-tyylitiedostossa.

Kuva 20 QR-koodigeneraattorin käyttöliittymä.



Ohjelmakoodi 7 Esimerkki QR-koodigeneraattorin lähdekoodista.

```
# aluksi luodaan uusi Flask-luokan objekti nimeltään app
app = Flask(__name__)

# jos GET-metodilla kutsutaan palvelimen juuriosoitetta, ohjataan käyttäjä index.html-sivulle
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

# jos POST-metodilla kutsutaan palvelimelta osoitetta generateQR, suoritetaan generateQR-funktio
@app.route('/generateQR', methods=['POST'])
def generateQR():
    # alustetaan muuttujat
    filename = ''
    campaign_name = ''
    shortLink = ''

    # 1. PARAMETRIEN HAKU VERKKOSIVUN LOMAKKEELTA
    # 2. KAMPANJAKOHTAISEN KANSION LUOMINEN. JOS KANSIO ON JO OLEMASSA SE TYHJENNETÄÄN
    # 3. DYNAAMISEN LINKIN LUOMINEN.
    # 4. QR-KOODIN LUOMINEN + DIGIMOIN LOGON LIITTÄMINEN QR-KOODIIN.
    # 5. QR-KOODIN TALLENTAMINEN KUVATIEDOSTOKSI. TIEDOSTONIMEEN LISÄTÄÄN ID-NUMERO JA AIKALEIMA.

    # palautetaan tarvittavat muuttujat JSON-objektina takaisin HTML-sivulle
    return jsonify({'filename' : filename, 'campaign': campaign_name, 'shortLink' : shortLink})

# käynnistetään Flaskin kehityspalvelin
if __name__ == '__main__':
    app.run(debug=True)
```

Ohjelmakoodi 7 on esitetty QR-koodigeneraattorin verkkosovelluspuolen lähdekoodi pelkistettynä. Kun palvelimen IP-osoitetta kutsutaan selaimesta **GET**-metodilla, ohjataan käyttäjä Flaskin **render_template** -funktion avulla index.html -sivulle (rivit 12–14). Kun verkkosivulta kutsutaan **POST**-metodilla palvelimen osoitetta /**generateQR**, suoritetaan funktio, joka aloittaa dynaamisen linkin sekä QR-koodin luomisprosessin (rivit 24–28). Prosessin alussa haetaan verkkosivun lomakkeelta käyttäjän syöttämät tiedot kampanjasta sekä luotavan QR-koodin tyyppi (mustavalkoinen, värillinen tai taiteellinen). Seuraavaksi luodaan palvelimelle kampanjakohtainen hakemisto ja luodaan dynaaminen linkki Firebasen REST-rajapinnan avulla. Dynaamiseen linkkiin liitetään parametrina kampanjan id-numero, jonka avulla kampanja on yksilöitävissä. Tämän jälkeen dynaamisesta linkistä luodaan halutunlainen QR-koodi. Prosessin loppuvaiheessa luotu QR-koodi tallennetaan palvelimelle kuvatiedostona kampanjan id-numerolla sekä aikaleimalla varustettuna. Lopuksi QR-koodin kuvatiedoston täydellinen polku, kampanjan nimi sekä luodun dynaamisen linkin lyhyt URL-osoite palautetaan takaisin verkkosivulle JSON-muodossa (rivi 31).

8.3 Dynaamisen linkin luominen Pythonilla

Kun dynaamisia linkkejä halutaan luoda ohjelmallisesti, on ensin saatava pääsy Firebasen **REST**-rajapintaan. Tätä varten tarvitaan uniikki **API-avain** (**Web API Key**). Avaimen löytää Firebasen konsolista projektin hallintasivulta **Project settings** -valikon **General**-välilehdeltä. REST-rajapintaa voidaan kutsua tietystä URL-osoitteesta HTTP-protokollan **POST**-metodilla. API-avain liitetään parametrina URL-osoitteen perään (Kuva 21). (*Create Dynamic Links with the REST API | Firebase Dynamic Links*, 2023; *Google Firebase*, 2023)

Kuva 21 Firebasen REST-rajapinnan kutsuminen.

```
https://firebasedynamiclinks.googleapis.com/v1/shortLinks?key=api_key
```

Dynaaminen linkki voidaan luoda ohjelmallisesti siten, että sille rakennetaan ensin URL-osoite eli merkkijono, josta käytetään myös nimitystä pitkä linkki (**long link**). Dynaamisen linkin verkkotunnus on ensin määriteltävä Firebasen konsolissa, kuten kappaleessa 7.4 on esitetty. Tämän jälkeen verkkotunnuksen perään määritellään parametreina kaikki

dynaamisen linkin halutut toiminnot. Parametreja voi olla useita ja niille kaikille on määritelty tietty nimi tai lyhenne. Taulukko 4 sisältää tärkeimpiä dynaamisen linkin parametreja. (*Create Dynamic Links with the REST API | Firebase Dynamic Links*, 2023)

Taulukko 4 Dynaamisen linkin parametreja.

Parametrin nimi	Toiminto
link	Varsinainen linkki, joka halutaan avata sovelluksessa. Tähän linkkiin on mahdollista lisätä omia parametreja.
apn	**Android package name**. Sen Android-sovelluksen nimi tai tunnus, jota Android-käyttöjärjestelmässä käytetään linkin avaamiseen.
afl	**Android fallback link**. Osoite, mihin käyttäjä ohjataan Android-laitteessa, mikäli sovellusta ei ole asennettu.
ibi	**iOS bundle ID**. Sen iOS-sovelluksen tunnus, jota käytetään iOS-käyttöjärjestelmässä linkin avaamiseen.
isi	**iOS store ID**. Sen sovelluksen tunnus App Storessa, johon käyttäjä ohjataan iOS-laitteessa, mikäli sovellusta ei ole asennettu.
ifl	**iOS fallback link**. Osoite, mihin käyttäjä ohjataan iOS-käyttöjärjestelmässä, mikäli sovellusta ei ole asennettu.
ofl	**Other fallback link**. Osoite, mihin käyttäjä ohjataan, jos käytössä on jokin muu käyttöjärjestelmä, kuin Android tai iOS.

Pitkää linkkiä muodostettaessa kaikki sen parametrit määritellään yhteen merkkijonoon peräkkäin (Kuva 22). Kun pitkä linkki on määritelty, voidaan siitä luoda dynaaminen linkki lähettämällä se JSON-tiedostomuodossa Firebasen REST-rajapintaan. Rajapinnasta saadaan takaisin vastaus, joka sisältää luodun dynaamisen linkin tiedot JSON-tiedostomuodossa. Firebase luo dynaamiselle linkille automaattisesti uniikin loppuosan, joka yhdessä aiemmin määritellyn dynaamisen linkin verkkotunnuksen kanssa muodostavat niin sanotun lyhyen linkin (**short link**). Tämä lyhyt linkki on varsinainen loppukäyttäjälle näkyvä linkki. Ohjelmakoodi 8 on esitetty dynaamisen linkin luominen Python-ohjelmointikielellä. (*Create Dynamic Links with the REST API | Firebase Dynamic Links*, 2023)

Kuva 22 Esimerkki pitkän linkin rakenteesta.

```
https://your_subdomain.page.link/?link=your_deep_link&apn=android_package_name&afl=android_fallback_link&ibi=ios_bundle_id&isi=app_store_id&ifl=ios_fallback_link&ofl=other_fallback_link
```

Ohjelmakoodi 8 Dynaamisen linkin luominen Pythonilla.

```
url = 'https://firebasedynamiclinks.googleapis.com/v1/shortLinks?key=<your_api_key>'

longLink = '<your_long_link>'
linkObj = {
    "longDynamicLink" : longLink
}

#lähetetään JSON-objekti POST-metodilla Firebasen REST APIn URL-osoitteeseen
response = requests.post(url, json = linkObj)

# muutetaan saatu vastaus eli html response JSON-objektiksi ja haetaan siitä
# shortLink-muuttujan arvo, joka on dynaamisen linkin lyhyt URL-osoite
shortLink = response.json()['shortLink']
```

8.4 QR-koodin generointi Pythonilla

Ohjelmakoodi 9 on esitetty esimerkki QR-koodin generoinnista Pythonin **Segno**-ohjelmakirjaston avulla. Koodin alussa (riveillä 1–2) tarvittavat ohjelmakirjastot otetaan käyttöön import-komennolla. Tämän jälkeen luodaan **qrcode**-niminen objekti Segno-kirjaston **make**-funktion avulla (rivi 4). Funktion ainoa pakollinen parametri on QR-koodin sisältämä informaatio eli tässä tapauksessa tavallista tekstiä sisältävä merkkijono. Tämän lisäksi QR-koodille voidaan erikseen määritellä parametreina muun muassa virheenkorjaustaso, versionumero ja käytettävä maskikuvio. Jos nämä arvot jätetään määrittelemättä, Segno käyttää oletuksena virheenkorjauksen tasoa M, määrittelee versionumeroksi pienimmän mahdollisen version, johon annettu informaatio mahtuu sekä määrittelee algoritmin avulla parhaimman mahdollisen maskikuvion. Kun qrcode-objekti on ensin luotu, voidaan siitä tallentaa Segnon **save**-metodin avulla halutunlainen versio (rivi 5). Tässä QR-koodille voidaan määritellä haluttu tiedostomuoto, koodin eri alueiden värit, skaalaus sekä esimerkiksi koodin reuna-alueen leveys. (*Segno documentation*, 2022)

Ohjelmakoodi 9 Esimerkki QR-koodin generoinnista Pythonilla.

QR-koodia voidaan personoida lisäämällä siihen informaatiota, kuten esimerkiksi yrityksen logo. Python-ohjelmassa tämä onnistuu esimerkiksi **Pillow**-kirjaston **Image**-moduulin avulla. QR-koodin sekä logon kuvatiedostot ladataan ensin muistiin sekä asetetaan muuttujien arvoiksi Image-moduulin **open**-funktiota käyttäen (rivit 7–8). Tämän jälkeen QR-koodin kuva muunnetaan RGB-muotoon, jotta sen käsittely onnistuu Pillow-kirjaston funktioilla (rivi 10). Seuraavaksi määritellään sen pisteen koordinaatit, johon logo lisätään QR-koodin päälle. Tähän pisteeseen sijoittuu tässä tapauksessa logon vasen yläkulma. Koordinaatit lasketaan QR-koodin ja logon pituuksien ja leveyksien avulla (rivit 12–13). Kuvien pituudet ja leveydet saadaan suoraan Image-tyyppisten muuttujien **length-** ja **width**-arvoista. Kun koordinaatit on selvitetty, lisätään logo QR-koodin päälle Image-moduulin **paste**-funktion avulla (rivi 15). (*Pillow Image Module*, 2023)

9 Dynaamisen linkin käsittely mobiilisovelluksessa

Kun mobiilikäyttäjä skannaa QR-koodin, hän ohjautuu suoraan QR-koodiin upotettuun dynaamiseen linkkiin liitettyyn sovellukseen. Dynaamiseen linkkiin on myös sisällytetty yksilöivä tieto siitä kampanjasta, jota varten linkki on alun perin luotu. Tässä tapauksessa tämä yksilöivä tieto on kampanjakohtainen id-numero tai -tunniste. ID-tunniste voidaan liittää merkkijonoon osaksi dynaamisen linkin **link-parametria** (Taulukko 4). Tähän parametriin määritellään se varsinainen syvälinkki, jonka mobiilisovellus avaa. Link-parametrin täytyy olla Firebasen määritelmien mukaan URL-osoite HTTP- tai HTTPS-muodossa. Link-parametri voisi olla näin ollen määritelty esimerkiksi muotoon: **https://www.testi.fi/campaign?id=123**, jossa viimeinen lukusarja (123) vastaa kampanjan id-numeroa. Mobiilisovelluksen koodissa dynaamisesta linkistä luetaan ensin linkin URL-osoite muuttujaan, jonka jälkeen id-numero puretaan sen sisältä esimerkiksi leikkaamalla merkkijono poikki oikeasta kohdasta. Kun id-numero on saatu purettua, sovellus ohjaa käyttäjän oikean kampanjan tietoihin. (*Google Firebase*, 2023)

9.1 React Native -kehitysympäristön rakentaminen

React Native -kehitysympäristöä rakennettaessa järjestelmässä täytyy olla asennettuna vähintään Taulukko 5 esitetyt ohjelmat/työkalut. Kun kaikki tarvittava on asennettu, voidaan uusi React Native -projekti alustaa Windowsin komentokehotteesta Node.js:n paketinhallintatyökalun (**npx**) avulla (Komento 3). (*React Native · Learn Once, Write Anywhere*, 2023)

Taulukko 5 React Native -ympäristön vaatimat asennukset.

Ohjelman/ alustan nimi	Mihin tarkoitettu?	Latauspaikka
Java JDK	Java Development Kit on kehitysympäristö, joka sisältää työkalut Java-ohjelmien kirjoittamiseen.	https://www.oracle.com/java/technologies/downloads/

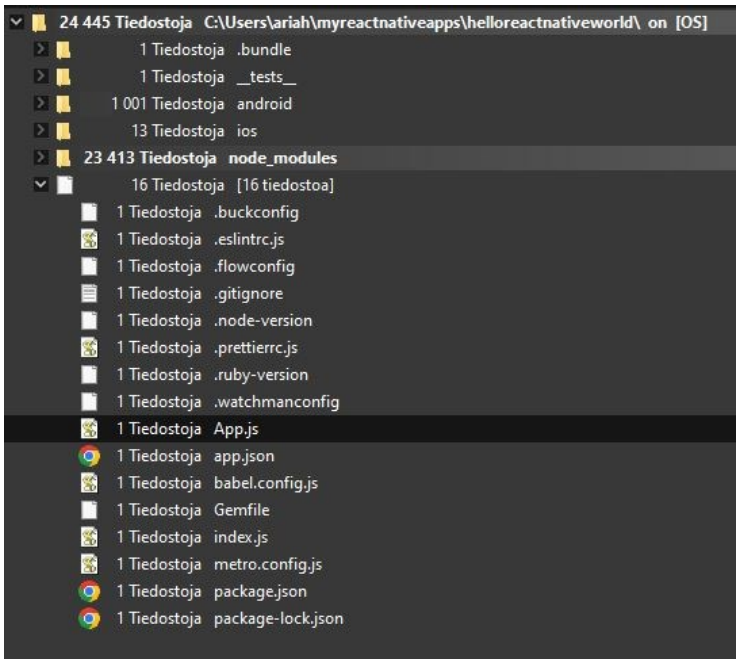
Node.js	Selaimesta riippumaton JavaScript-tulkki JavaScript-ohjelmien ajamiseen.	https://nodejs.org/en/download
Android Studio (Android SDK + Android Emulator)	Android-käyttöjärjestelmän virallinen ohjelmointiympäristö. Sisältää emulaattorin Android-laitteiden mallintamiseen tietokoneella.	https://developer.android.com/studio
Visual Studio Code tai jokin muu IDE-ohjelmisto	Ohjelmisto ohjelmakoodin kirjoittamiseen ja ajamiseen.	https://code.visualstudio.com/

Komento 3 React Native -projektin luominen

```
npx react-native init helloworld
```

Komento luo React Native -projektille tarvittavan hakemistorakenteen sekä lataa ja asentaa sinne kaikki tarvittavat tiedostot (Kuva 23). Projektihakemiston juuresta löytyy tiedosto nimeltä **App.js**, joka suoritetaan oletusarvoisesti, kun ohjelma ajetaan. Tämä JavaScript-tiedosto sisältää varsinaisen suoritettavan React Native -koodin. (*React Native · Learn Once, Write Anywhere*, 2023)

Kuva 23 React Native -projektin hakemistorakenne.



9.2 Firebasen asennus React Native -projektiin

Kun halutaan käsitellä Firebasen dynaamisia linkkejä React Native -sovelluksessa, on projektiin ensin asennettava Firebasen tarvitsemat moduulit. Ne voidaan asentaa järjestelmään **Yarn**-paketinhallintasovelluksen avulla. Jos järjestelmässä ei ole Yarn-sovellusta asennettuna, se täytyy ensin asentaa komentokehotteesta Node.js:n **npm**-ohjelman avulla (Komento 4). Tämän jälkeen sovellus on vielä liitettävä Firebasen konsolissa osaksi Firebase-projektia. Kappaleessa 7.3 on esitetty kaikki tähän liittyvät toimenpiteet. (*React Native Firebase*, 2023)

Komento 4 Yarn-sovelluksen ja Firebase-moduulien asennus

```
npm install --global yarn
yarn add @react-native-firebase/app
yarn add @react-native-firebase/dynamic-links
```

9.3 Dynaamisen linkin käsittely React Nativessa

Kun kaikki tarvittavat ohjelmat ja moduulit on asennettu, voidaan dynaamisten linkkien käsittelyä varten luoda ohjelma (Ohjelmakoodi 10). Dynaamisten linkkien käsittelyä varten täytyy aluksi ottaa käyttöön dynamicLinks-moduuli (rivi 4). Riveillä 8–9 alustetaan kaksi Reactin tilamuuttujaa id-numerolle ja dynaamisen linkin URL-osoitteelle. Riveillä 15–30 on määritelty JavaScript-funktio, jonka sisällä noudetaan dynaamisesta linkistä sen sisältämä URL-osoite (rivi 23), josta puolestaan saadaan purettua id-numero (rivi 24). Lopuksi URL-osoite ja id-numero asetetaan tilamuuttujien arvoiksi (rivit 25 ja 26), minkä jälkeen niitä voidaan käsitellä halutulla tavalla. Esimerkkiohjelmassa niiden arvot tulostetaan mobiililaitteen ruudulle (Kuva 24).

Ohjelmakoodi 10 Esimerkki dynaamisen linkin käsittelystä ohjelmakoodissa.

```
4  import dynamicLinks from '@react-native-firebase/dynamic-links';
5
6  const dynamicLink=()=>{
7
8    /* määritellään tilamuuttujat id:lle ja link-parametrille */
9    const [id, setId] = useState(null);
10   const [url, setUrl] = useState(null);
11
12   /*  funktio, joka tutkii onko sovellus avattu dynaamisen linkin kautta.
13       Jos linkki on olemassa luetaan se muistiin ja puretaan sen sisältä ensin
14       linkin URL ja siitä puolestaan id-numero*/
15   const getDynamicLink = () => {
16     let linkUrl = "";
17     let id = "";
18     useEffect(() => {
19       dynamicLinks()
20         .getInitialLink()   /* kutsutaan dynamicLinks-moduulin getInitialLink-funktiota */
21         .then(link => {     /* odotetaan, että saadaan vastaus getInitialLink-funktiolta*/
22           if (link){ /* jos linkki löytyy, suoritetaan tämä koodiblokki */
23             linkUrl  = link.url;  /* haetaan dynaamisen linkin url */
24             id = linkUrl.split('?id=')[1];  /* id-numero on linkin url-osoitteessa ?id= -merkkijonon perässä */
25             setId(id);     /*  asetetaan id ja linkUrl tilamuuttujien arvoiksi  */
26             setUrl(linkUrl);
27           }
28         });
29       }, []);
30   };
```

Kuva 24 Id-numero ja dynaamisen linkin URL-osoite mobiililaitteessa

Tässä esimerkkiohjelmassa parametrit ainoastaan puretaan dynaamisen linkin sisältä ja tulostetaan käyttäjälle. Asiakkaan mobiilisovelluksessa logiikka toimisi siten, että ohjelma tutkisi ensin sen, onko se avattu dynaamisen linkin kautta. Jos dynaaminen linkki löytyy, puretaan sen sisältä kampanjan id-numero ja navigoidaan sen jälkeen kyseisen kampanjan tietoihin. Jos dynaamista linkkiä ei ole olemassa, ohjataan käyttäjä ohjelman pääsivulle.

10 Johtopäätökset ja pohdinta

Tässä luvussa arvioidaan kehitysprojektina toteutetun sovelluksen kehitystyötä ja lopputulosta. Luvussa pohditaan myös erilaisia vaihtoehtoja QR-koodigeneraattorin integroimiseksi asiakkaan jo olemassa olevaan sovellukseen.

10.1 QR-koodigeneraattorin kehitysprosessi

Google Firebasen dynaamisen linkin sekä QR-koodin luominen ohjelmallisesti oli suhteellisen suoraviivainen ja yksinkertainen prosessi. Kun Firebase-projekti ja dynaamisen linkin verkkotunnus oli ensin luotu Firebaseen, oli dynaamisten linkkien luominen Googlen REST-rajapinnan kautta helppoa. Dynaamisten linkkien osalta eniten haasteita projektille toivat Applen kehittäjätilin maksullisuus sekä kehitysympäristön rajoittaminen vain Applen laitteille. Näistä syistä dynaamisen linkin toimintaa ei pystytty testaamaan iOS-ympäristössä.

QR-koodin luominen ohjelmallisesti valmiiden ohjelmakirjastojen avulla oli myös melko yksinkertaista. QR-koodien osalta eniten haasteita projektille toivat sopivan ohjelmointikielen sekä ohjelmakirjaston valinta. Python-ohjelmointikieli sekä Segno- ja Flask-ohjelmakirjastot osoittautuivat erittäin monipuolisiksi ja helposti hallittaviksi työkaluiksi projektin toteuttamiseen.

10.2 QR-koodigeneraattorin integrointi DigiMoin verkkosovellukseen

DigiMoin käyttäjä- sekä kampanjatietoja hallitaan verkkosovelluksella, jota käytetään verkkoselaimen kautta. Verkkosovellus on toteutettu **PHP**-ohjelmointikielellä **CodeIgniter** -sovelluskehyksen avulla. CodeIgniter on avoimen lähdekoodin sovelluskehys, jolla voidaan kehittää dynaamisia verkkosivuja. (Pedamkar, 2019)

Helpoin tapa QR-koodigeneraattorin integroimiseen DigiMoin alustalle on asentaa jo valmis Flask-ohjelma erillisenä palveluna verkkopalvelimelle ja kutsua palvelua suoraan verkkoselaimesta. Ehtona tälle on tietysti, että palvelimella on mahdollista ajaa Python-koodia. Mikäli DigiMoin palvelimella ei ole tukea Pythonille, on QR-koodigeneraattori mahdollista asentaa myös täysin eri palvelimelle. DigiMoin kampanjanhallintasivulla voisi

olla esimerkiksi nappi, jota painamalla kyseisen kampanjan nimi ja id-numero lähetetään QR-koodigeneraattorille. Prosessin jälkeen QR-koodigeneraattori palauttaa luodun QR-koodin kuvatiedostona takaisin DigiMoin verkkosivulle.

Toinen tapa QR-koodigeneraattorin toteuttamiseen, on luoda se kokonaan PHP-ohjelmointikielellä. Tässä tapauksessa koodi olisi mahdollista sisällyttää suoraan DigiMoin verkkosovelluksen lähdekoodiin. PHP-koodi suoritetaan Pythonin tapaan palvelinpäässä (engl. backend), josta prosessin jälkeen tarvittavat tiedot lähetetään takaisin selainpuolelle (engl. frontend).

Kolmas mahdollisuus on luoda QR-koodi suoraan selaimessa ilman varsinaista palvelinpuolella suoritettavaa koodia. Tämä voidaan toteuttaa JavaScript-ohjelmointikielen ja AJAX-tekniikan avulla. Niillä pystytään toteuttamaan dynaamisen linkin luomiseen tarvittavat HTTP-palvelinpyynnöt Firebasen REST-rajapintaan, minkä jälkeen QR-koodi voidaan luoda dynaamisesti jonkun JavaScriptille kehitetyn QR-koodikirjaston avulla.

11 Yhteenveto

Opinnäytetyöprosessi ja sen ohessa toteutettu ohjelmistoprojekti sujuivat mielestäni hyvin. Asetettuihin tutkimuskysymyksiin saatiin riittävän kattavat vastaukset dynaamisten linkkien ja QR-koodien toiminnan sekä niiden ohjelmallisen luomisen osalta.

Suurimmat ongelmat opinnäytetyön kirjoittamisessa syntyivät mielestäni siitä, kuinka selittää asioita ymmärrettävästi lukijalle. Tämän ongelman osalta päädyin käyttämään runsaasti kuvia tekstin tukena. Myös Applen kehitysympäristön ja laitteiston rajoitukset toivat omat haasteensa kehitysprojektille.

Opinnäytetyön kehitysprojektin lopputuloksena syntyneestä ohjelmasta on jatkossa toimeksiantajalle aidosti hyötyä. Dynaamisten linkkien avulla asiakkaan toiveet QR-koodin toiminnallisuudesta saatiin ratkaistua.

Opinnäytetyöprosessin kautta sain runsaasti uutta tietoa QR-koodin sekä dynaamisten linkkien toiminnasta. Dynaamiset linkit sekä niiden toiminta eivät olleet minulle aiemmin lainkaan tuttuja. Kehitysprojektin aikana sain myös paljon uutta käytännön kokemusta Python-ohjelmointikielestä sekä verkkosovelluksen palvelinpuolen ja käyttöliittymän yhteensovittamisesta. Aikaisempi kokemukseni Pythonista oli lähinnä perusteiden tasolla. Lisäksi opin itselleni täysin uuden Pythonin sovelluskehyksen, Flaskin käyttöä. Flask osoittautui omasta mielestäni hyväksi ja helposti sisäistettäväksi työkaluksi verkkosovellusten kehittämiseen.

QR-koodin generointiohjelman integroimiseen asiakkaan ohjelmaan löydettiin useita eri vaihtoehtoja. Haluttu ominaisuus voidaan toteuttaa joko täysin omana palvelunaan tai vaihtoehtoisesti sisällyttää suoraan asiakkaan sovelluksen lähdekoodiin.

Itse uskon vahvasti siihen, että tulevaisuudessa QR-koodit sekä dynaaminen linkitys tulevat nousemaan vieläkin suurempaan rooliin sovellusten kehitysprosesseissa.

Lähteet

A Complete Guide to QR Codes. (2023). https://www.creativesafetysupply.com/articles/a-complete-guide-to-qr-codes/

App manifest overview. (2023). Android Developers. https://developer.android.com/guide/topics/manifest/manifest-intro

AppsFlyer. (2023). *AppsFlyer | Make good data-driven choices*. AppsFlyer. https://www.appsflyer.com/

Branch. (2023). *Branch | Mobile Growth & Attribution Platform for Enterprises and Brands*. Branch. https://www.branch.io/

Brewster, C. (2023). *15 Examples of Successful Companies Using React Native in 2023 | Trio Developers*. https://www.trio.dev/blog/companies-use-react-native

Comparison of Python QR Code libraries. (2022). https://segno.readthedocs.io/en/stable/comparison-qrcode-libs.html

Create Deep Links to App Content. (2023). Android Developers. https://developer.android.com/training/app-links/deep-linking

Create Dynamic Links with the REST API | Firebase Dynamic Links. (2023). Firebase. https://firebase.google.com/docs/dynamic-links/rest

Dickson, B. (2022, toukokuuta 20). *A Brief History of JavaScript*. DEV Community. https://dev.to/dboatengx/history-of-javascript-how-it-all-began-92a

Dillemuth, J. (2023). *Mihin Python-ohjelmointikieltä käytetään?* Tieturi. https://www.tieturi.fi/blogi/mihin-python-ohjelmointikielta-kaytetaan/

Firebase for Android. (2023). Firebase. https://firebase.google.com/docs/android/setup

Flask Templates. (2023). https://python-adv-web-apps.readthedocs.io/en/latest/flask3.html

Garg, G. (2018, syyskuuta 12). QR Code Statistics 2022: Latest Numbers on Global Usage. *Scanova Blog*. https://scanova.io/blog/qr-code-statistics/

GetSocial. (2023). *GetSocial | User acquisition and engagement solution*. GetSocial. https://www.getsocial.im/

Google Firebase. (2023). Firebase. https://firebase.google.com/

Huang, Z. (2020). *A complete guide to mobile app deep linking*. https://www.adjust.com/blog/dive-into-deeplinking/

JavaScript Institute. (2022). *JavaScript Institute*. https://www.javascriptinstitute.org/javascript-tutorial/history-of-javascript/

Kivisaari, T. (2016). *API:t ovat modernin integraatiostrategian ydin*. https://blog.digia.com/rest-api

Masiello, E., & Friedmann, J. (2017). *Mastering React Native: Leverage Frontend Development Skills to Build Impressive IOS and Android Applications with React Native*. Packt Publishing, Limited. http://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=4787004

Mask patterns are used to break up solid blocks in a QR code. (2011, syyskuuta 13). https://commons.wikimedia.org/wiki/File:QR_Code_Mask_Patterns.svg

Mecalux. (2021). *QR codes in logistics: Speed and flexibility*. https://www.mecalux.com/blog/qr-codes-logistics

Most used languages among software developers globally 2022. (2022). Statista. https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

Pedamkar, P. (2019, kesäkuuta 15). What is CodeIgniter? | How IT Work | Scope & Skill | Feature & Advantage. *EDUCBA*. https://www.educba.com/what-is-codeigniter/

Peltomäki, J. (2023). *Python 3 -ohjelmoinnin perusteet*. Books on Demand.

Pillow Image Module. (2023). Pillow (PIL Fork). https://pillow.readthedocs.io/en/stable/reference/Image.html

Python Syntax. (2023). *Python Tutorial - Master Python Programming For Beginners from Scratch*. https://www.pythontutorial.net/python-basics/python-syntax/

QRazyBox—About qr-code. (2023). https://merricx.github.io/qrazybox/help/getting-started/about-qr-code.html

QRcodeChimp. (2022). QR Code Use Cases: 17 Top Applications for Business, Marketing, and More. *Free QR Code Generator Online*. https://www.qrcodechimp.com/qr-code-use-cases/

QRcode.com ／DENSO WAVE. (2023). https://www.qrcode.com/en/

QRStuff.com. (2011, joulukuuta 14). QR Code Error Correction. *QRStuff.Com*. https://blog.qrstuff.com/general/qr-code-error-correction

React Native · Learn once, write anywhere. (2023). https://reactnative.dev/

React Native Firebase. (2023). React Native Firebase. https://rnfirebase.io/

Reidt, T. (2022). *What is the Android manifest file?* https://emteria.com/learn/android-manifest-file

Ruokangas, J. (2023). *React Native: Kaikki mitä olet aina halunnut kysyä*. https://www.fraktio.fi/blogi/react-native-kaikki-mita-olet-aina-halunnut-kysya

Segno documentation. (2022). https://segno.readthedocs.io/en/latest/

Showcase · React Native. (2023). https://reactnative.dev/showcase.html

Stevenson, D. (2018, lokakuuta 25). What is Firebase? The complete story, abridged. *Firebase Developers*. https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

Tarvainen, J. (2016). *Mikä on TypeScript?* https://symfony.fi/artikkeli/mika-on-typescript

Thonky.com. (2023). *Data Masking—QR Code Tutorial*. https://www.thonky.com/qr-code-tutorial/data-masking

Understanding Dynamic Links (i.e. Programmable Deeplinks). (2021). BDK. https://bdk.crisp.help/en-us/article/understanding-dynamic-links-ie-programmable-deeplinks-11tbkf8/

Welcome to Flask—Flask Documentation (2.2.x). (2010). https://flask.palletsprojects.com/en/2.2.x/

Werkzeug. (2023). Pallets. https://palletsprojects.com/p/werkzeug/

What is Flask Python—Python Tutorial. (2021). https://pythonbasics.org/what-is-flask-python/

Wicker, S. B., & Bhargava, V. K. (1999). *Reed-Solomon Codes and Their Applications*. John Wiley & Sons.

WSGI.org. (2023). https://wsgi.readthedocs.io/en/latest/index.html

Zxing/zxing. (2023). [Java]. ZXing Project. https://github.com/zxing/zxing (Original work published 2011)

Liite 1: Aineistonhallintasuunnitelma

Kehitysprojektin aikana pidetään päiväkirjaa (aineisto), johon kerätään teknistä tietoa projektista. Tämä tieto analysoidaan opinnäytetyötä varten. Päiväkirjaa säilytetään tekijän tietokoneen C-asemalla, ja siitä tehdään säännöllisesti varmuuskopioita ulkoiselle kovalevylle. Päiväkirjaa säilytetään C-asemalla ainakin vuoden verran opinnäytetyön valmistumisesta.

Kehitysprojektin aikana pidetyistä kokouksista pidetään pöytäkirjoja, jotka säilytetään tekijän tietokoneen C-asemalla sekä varmuuskopioidaan ulkoiselle kovalevylle.

Opinnäytetyöaineiston jatkokäyttö työn valmistumisen jälkeen

Tämän opinnäytetyön tuloksena syntyneet aineisto ja tulokset ovat Calevala Interactive Oy:n omaisuutta, eikä niitä luovuteta kolmansille osapuolille. Opinnäytetyön tekijä säilyttää oman kopionsa aineistosta vuoden ajan opinnäytetyön hyväksymispäivästä lukien, jotta opinnäytetyön tulokset voidaan tarvittaessa varmistaa. Tämän jälkeen tekijä hävittää oman kopionsa aineistosta. Kaikki aineiston säilytykseen sekä sen hävittämiseen liittyvät toimenpiteet tehdään hyvää tietoturvaa noudattaen.