Paikkatietojen hakeminen Google Places -ohjelmointirajapinnasta Android-sovelluksessa

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö
HAMK Visamäki
Tietojenkäsittelyn koulutusohjelma
2015
Toni Lavonen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Visamäki, Hämeenlinna

Tekijä Toni Lavonen **Vuosi** 2018

Työn nimi Paikkatietojen hakeminen Google Places ohjelmointirajapinnasta Android sovelluksessa

Työn ohjaaja/t Tommi Lahti, Lauri Salminen

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli perehtyä Google Places API for Android- ja Google Places API Web Service -teknologioihin ja verrata näiden kahden teknologian eroavaisuuksia. Käytännön osuudessa suoritettiin paikkatietojen hakeminen hyödyntäen toista näistä kahdesta teknologiasta.

Opinnäytetyön toimeksiantajana toimi Hämeen ammattikorkeakoulun Älykkäät palvelut -tutkimusyksikkö. Työn tarkoituksena oli edistää DigiTrail -hankkeessa kehitteillä olevaa DigiTrail-sovellusta, jonka avulla voitaisiin mainostaa Kanta-Hämeen pk-yrityksiä tarjoamalla sovelluksen käyttäjälle tietoa lähellä olevista paikoista.

Opinnäytetyön käytännönosuudessa käytiin läpi paikkatietojen hakeminen, joka käsitti URL-osoitteiden luomisen, HTTP-pyyntöjen muodostamisen, HTTP-vastauksen vastaanottamisen ja käsittelyn sekä haluttujen tietojen tulostamisen laitteen näytölle.

Opinnäytetyössä onnistuttiin vastamaan siinä esitettyihin tutkimuskysymyksiin ja sen tuloksista käy ilmi Google Places API for Android- ja Google Places API Web Service -teknologioiden merkittävimmät erot sekä miten paikkatietojen hakeminen suoritetaan käytännössä.

Avainsanat C#, Xamarin.Android, www-ohjelmointirajapinnat, mobiiliohjelmointi

Sivut 28 sivua, joista liitteitä 4 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Visamäki, Hämeenlinna

Author Toni Lavonen **Year** 2018

Subject Requesting Places Information from Google Places API in an Android Application

Supervisors Tommi Lahti, Lauri Salminen

ABSTRACT

The aim of the thesis was to study Google Places API Web Services and Google Places API for Android and compare the differences between them. In the practical part, location data was extracted using one of these two technologies.

The thesis was commissioned by the Häme University of Applied Sciences Intelligent Services research unit. The purpose of the thesis was to help develop the DigiTrail application developed in the DigiTrail project, which could eventually help promote Tavastia Proper's SMEs by providing the information about nearby places to the application user.

The practical part of the thesis examines retrieving of place information. This included URL creation, HTTP request formatting, HTTP response receiving and handling, as well as showing user the information on the screen.

The thesis managed to successfully answer to the research questions presented in it, and its results show the most significant differences between the Google Places API for Android and Google Places API Web Service technologies and how to request information about places in practice.

Keywords C#, Xamarin.Android, Web Services, mobile programming

Pages 28 pages including appendices 4 pages

SISÄLLYS

Sanasto

API	(Application Programming Interface) Ohjelmointirajapinta, jonka avulla ohjelma tarjoaa ominaisuutensa toisten ohjelmien käytettäväksi.
HTTP	(HyperText Transfer Protocol) Selainten ja palvelimien käyttämä siirtoprotokolla.
HTTPS	(HyperText Transfer Protocol Secure) HTTP ja TLS/SSL protokollien yhdistelmä. Käytetään tietojen suojattuun siirtoon internetissä.
JSON	(JavaScript Object Notation) Yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen.
Web Service	WWW -pohjainen ohjelmointirajapinta, jota palvelin tarjoaa muille ohjelmistoille Internet-pohjaisen protokollan yli.
XML	(Extensible Markup Language) Tekstipohjainen tiedon esitysmuoto, rakenteellinen kuvauskieli, jonka skeeman avulla voidaan kuvata itse tieto ja sen muoto.

1 JOHDANTO

Tämän opinnäytetyön toimeksiantajana toimi Hämeen ammattikorkeakoulun Älykkäät palvelut -tutkimusyksikö. Tutkimusyksikön DigiTrail-hankkeessa kehitetään retkeilyreittien monikanavaista näkyvyyttä ja saavutettavuutta. Hankkeen keskeisinä tavoitteina on rohkaista ja tukea Kanta-Hämeen pk-yrittäjiä kytkemään matkailupakettejaan osaksi alueen reitistöjä. Hanke tulee kestämään vuoden 2018 loppuun asti.

Hankkeessa kehittämiskohteina erityisesti ovat ympärivuotiset pyöräily- ja vaellusreitistöt sekä muut retkeilykohteet Kanta-Hämeen alueella. Hankkeen toteutuksessa keskitytään vahvasti digitaalisuuden hyödyntämiseen sisältöjen ja saavutettavuuden kehittämisessä yhteistyössä osallisten sidosryhmien kanssa. (DigiTrail).

DigiTrail on Google Android -käyttöjärjestelmälle kehitettävä mobiilisovellus, jonka tarkoituksena on tuoda näkyvyyttä reitistöille ja paikallisille yrityksille. Sovelluksen päätehtäviin kuuluu käyttäjän opastaminen valituilla luontopoluilla ja käyttäjän informoiminen lähellä olevista paikoista. (Liite 2.)

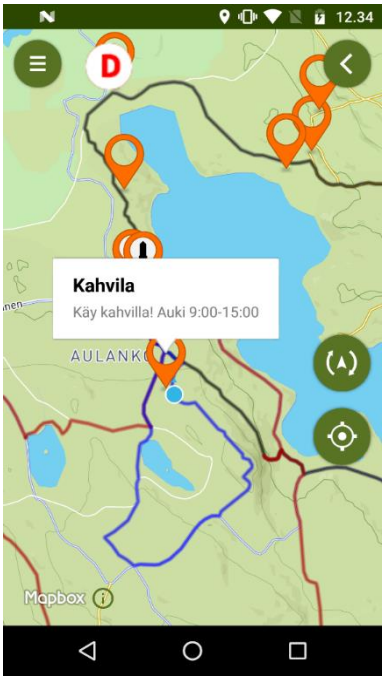
Sovellus on tätä opinnäytetyötä kirjoitettaessa aikaisessa kehitysvaiheessa. Sovelluksen kehitys oli aloitettu käyttäen Visual Studio 2017 ohjelmointiympäristöä ja Xamarin.Android laajennusta, joka mahdollistaa Android sovellusten kehittämisen C#-ohjelmointikielellä. Tästä syystä kehitystyötä tultaisiin jatkamaan samoilla työkaluilla eikä Android Studiolla, joka on puhtaasti Android sovellusten kehitykseen suunniteltu ohjelmointiympäristö.

Opinnäytetyön tekijällä on aikaisempaa historiaa toimeksiantajan kanssa. Tekijä on ollut kesäharjoittelijana vuonna 2017 toisen hankkeen parissa ja myöhemmin samana vuonna aloittanut työskentelyn DigiTrail-hankkeen parissa ICT-projektin muodossa.

Paikkatietojen kysely ohjelmointirajapinnoista oli yksi ICT-projektin tekemättä jääneistä tavoitteista. Aihe oli herättänyt mielenkiintoa, jonka takia tekijä ehdotti aihetta opinnäytetyöksi, johon toimeksiantaja oli suostunut. Samaan aikaan sovelluksen tietokantayhteyksien luomisesta suoritettaisiin opinnäytetyö tämän työn rinnalla, josta vastaisi Jerry Lehtisyrjä.

Toimeksiantajan toiveina ja tämän opinnäytetyön päätavoitteena on yritystietojen kyselyn implementointi DigiTrail-sovellukseen. Sovelluksen käyttäjän ollessa riittävällä etäisyydellä paikallista yritystä tai palvelua mainostavasta GPS-pisteestä, sovelluksen tulisi ilmoittaa käyttäjälle uudesta kohteesta, hakea tietoa paikasta ja tulostaa tietoa paikasta laitteen

näytölle tiedoille varatulle alueelle. Esimerkki ja ruutukaappaus sovelluksen kehitysversiosta Kuva 1.



Kuva 1. Paikkatietojen näyttäminen sovelluksessa

Paikkatietojen hakemiseen tultaisiin käyttämään hyödyksi Google Places ohjelmointirajapintoja, josta on olemassa kaksi eri versiota, Googles Places API for Android ja Google Places API Web Service. Opinnäytetyön toisena tavoitteena on verrata näitä kahta rajapintaa ja perustellusti päättää kumpaa ohjelmointirajapintaa hyödyntäen paikkatietojen kysely suoritettaisiin.

Opinnäytetyössä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

- Mikä on Google Places API for Android?
- Mikä on Google Places API Web Service?
- Miten Google Places API for Android ja Google Places Web Service eroavat toisistaan?
- Miten paikkatietojen kysely suoritetaan käytännössä?

2 OHJELMOINTIRAJAPINNAT JA WEB SERVICET

Opinnäytetyön keskisessä roolissa ovat ohjelmointirajapinnat ja web servicet. Ennen käytännön osuutta on hyvä ymmärtää mitä nämä kaksi asiaa ovat ja miten ohjelmistokehittäjä voi hyötyä niistä.

2.1 Ohjelmointirajapinnat

API (Application Programming Interface) eli ohjelmointirajapinta, on kokoelma komentoja, metodeja, protokollia ja objekteja joita ohjelmoija voi hyödyntää luodessaan sovelluksia tai kommunikoidessaan ulkoisen järjestelmän kanssa.

Ohjelmointirajapinnat tarjoavat kehittäjille komennot yleisimpiin operaatioihin, joten kehittäjän ei tarvitse kirjoittaa kaikkea tyhjästä. Eri käyttöjärjestelmille on omat ohjelmointirajapintansa. Nämä tarjoavat eri toimintoja, riippuen mille alustalle ne on kehitetty esimerkkinä mobiiliohjelmointirajapinnat, jotka yleisiä käyttöliittymäelementtejä kuten tekstikenttiä, ikkunoita ja painikkeita ruudun navigoimiseen.

Ohjelmointirajapinnat eivät rajoitu pelkästään sovelluksiin. Ohjelmointirajapinta voi olla myös verkkosivu, joka tarjoaa kehittäjälle pääsyn tiettyyn informaatioon. (TechTerms, API, 2016).

2.2 Web Servicet

Web Service on WWW-pohjainen ohjelmointirajapinta, joka on käytettävissä standardi verkkoprotokollien - HTTP tai HTTPS - kautta (TechTerms, Web Service, 2017). Web Servicet mahdollistavat tiedon vaihtamisen eri järjestelmien välillä verkon ylitse. Järjestelmien välillä ei tarvitse olla mitään yhteistä vaan ne voivat olla jopa eri alustoille kehitettyjä. (W3C Working Group Note, 2004.)

Web Servicen päätehtävä ei ole kommunikoida käyttäjän vaan toisten sovellusten kanssa. Web Servicen ja sovelluksen ei tarvitse olla samalta kehittäjältä vaan ne voivat erota niin arkkitehtuurillisesti kuin koodillisesti.

Kehittäjät voivat käyttää palvelimien rajapintojen komentoja ja metodeja datan kyselyyn. Riippuen rajapinnasta, data voidaan palauttaa useassa eri formaatissa, joista yleisimmät ovat XML ja JSON. (TechTerms, 2017.)

Suoraa suomennosta käsitteelle Web Service ei ole olemassa. Sanastokeskuksen mukaan yksi käännös olisi WWW-sovelluspalvelu (Sanastokeskus

TSK: Termipankki). Yleisesti www-sovelluspalveluista puhuttaessa käytetään kuitenkin termiä Web Service.

On hyvä huomioida, että Web Service voi viitata myös verkkopalveluihin, koska englanniksi käännös on sama. Tästä johtuen sekaannuksia tapahtuu tekstejä käännettäessä ja tästä syystä opinnäytetyön kirjoittaja katsoi parhaimmaksi selventää mitä termillä Web Service tarkoitetaan tässä työssä.

3 GOOGLE MAPS OHJELMOINTIRAJAPINNAT

Google ohjelmointirajapinnat ovat kokoelma Googlen kehittämiä WWW-ohjelmointirajapintoja, jotka mahdollistavat kommunikoinnin Googlen palveluiden kanssa kuten haku-, kartta- tai käännöspalvelut. Helpottaakseen kolmannen osapuolen sovellusten kehittäjien työtä Googlen ohjelmointirajapintojen kanssa, Google tarjoaa sovelluskirjastoja vähentääkseen tarvittavaa koodin määrää. (Google API Client Libraries.)

Google Maps ohjelmointirajapinnat ovat yksi osa Googlen ohjelmointirajapintakirjastoa. Google Maps ohjelmointirajapinnat tarjoavat eri tapoja yhdistää Google karttoja nettisivuihin, hakea tietoa Googlen karttapalveluista ja muokata karttanäkymää sovellukseen sopivaksi.

Riippuen tarpeista, yhtä tai useampaa ohjelmointirajapintaa voidaan käyttää. Google Maps sisältää tämän opinnäytetyön kirjoitushetkellä 16 eri ohjelmointirajapintaa. Kattavan listauksen Google Maps ohjelmointirajapinnoista löytyy liitteestä 2.

Käyttääkseen Googlen ohjelmointirajapintoja, kehittäjällä tulee olla API avain HTTP-pyyntöjen vahvistamiseksi. API avain toimii uniikkina tunnisteena, jonka avulla Google kykenee valvomaan kyselyiden määrää ja tarjoamaan analytiikkaa avaimen käyttömääristä.

Avaimen käyttöä voidaan halutessa rajoittaa. Tämä ei ole pakollista, mutta suositeltavaa, sillä rajoittamalla avaimen käyttökohteita voidaan taata, että avainta käytetään ainoastaan siihen tarkoitukseen, mitä varten se on luotu. (Google Maps APIs, FAQ.)

Google Maps ohjelmointirajapintojen käyttö on ilmaista, mutta rajoitettua. Normaalisti käyttöraja on 1000 pyyntöä 24 tunnin ajanjaksolla ja tunnistautuneilla 150 000 pyyntöä 24 tunnin ajanjaksolla. Tunnistautuminen tapahtuu aktivoimalla laskutuksen Google API konsolissa luottokorttitiedoilla, joita Google käyttää henkilöllisyyden tarkistuksessa. (Places API Web Service, Usage Limits and Billing.)

3.1 Google Places API for Android

Käyttämällä Google Places APIa kehittäjä voi luoda sovelluksia, jotka kykenevät tuomaan informaatiota laitteen lähellä olevista yrityksistä ja paikoista käyttämällä hyväksi Android laitteen paikannuspalvelua.

Määritteellä Place tarkoitetaan fyysistä paikkaa, jolla on nimi tai asiaa jonka voi löytää kartalta kuten yritys tai maantieteellinen kohde. APIssa kohdetta edustaa Place rajapinta. Tämä rajapinta sisältää informaatiota kuten paikan id, osoite, tyyppi ja maantieteellinen sijainti.

Paikan valintaan API tarjoaa useita sisäänrakennettuja työkaluja eli widget. Place picker UI widget antaa käyttäjän valita paikan interaktiiviselta kartalta. Autocomplete UI widget pyrkii arvaamaan mitä paikkaa käyttäjä etsii käyttäjän kirjoittaessa hakuehtoa. API tarjoaa myös tavan hakea listauksen lähellä olevista paikoista käyttäen laitteen viimeisintä sijaintia haussaan. (Places API for Android.)

3.2 **Google Places API Web Service**

Google Places API Web Service on Googlen ylläpitämä WWW-ohjelmointi rajapinta, joka palauttaa tietoja maantieteellisistä kohteista. Rajapinta palauttaa listan kohteista perustuen käyttäjän sijaintiin ja hakuehtoihin kuten hakusanoja, paikkojen yksilölliset tunnisteet tai tietty etäisyys sijainnista.

Haettaessa tiettyä kohdetta rajapinta palauttaa tarkempia tietoja kohteesta kuten yksikyiskohtaisempia tietoja ja arvosteluja sekä kohteen kuvien viittaukset, joilla voidaan suorittaa kuvahaku.

Rajapintaa käytetään HTTP-pyyntöjen kautta ja rajapinta palauttaa haun vastauksen JSON tai XML muodossa. Kaikki pyynnöt tulee käyttää HTTPS protokollaa ja niissä pitää olla sisällytettynä API avain. (Places API Web Service.)

3.3 **Google Places API for Android ja Web Servicen vertailu**

Google Places API for Android käyttää hyödykseen ohjelmointirajapinnan sisäänrakennettuja UI -elementtejä eli widget, tarjoten sovelluksen käyttäjälle interaktiivisen tavan paikkojen hakemiseen. Samalla API avustaa käyttää pyrkimällä arvaamaan, mitä paikkaa käyttäjä yrittää hakea.

Paikkojen tietojen haku ja tulostus suoritetaan kutsumalla ohjelmointirajapinnan metodeja ohjelmakoodissa. Ohjelmointirajapinta luo HTTP-pyynnöt, vastaan ottaa palvelun palauttaman vastauksen ja käsittelee sitä kutsuvien metodien parametrien mukaisesti. (Places API for Android).

Google Places API for Android vähentää tarvittavan koodin määrää, mutta paikkojen haku suoritetaan UI -elementtejä hyödyntämällä. DigiTrail sovelluksessa käyttäjän ei tarvitse hakea paikkaa tai paikan tietoja vaan kyselyt pitäisi suorittaa taustalla.

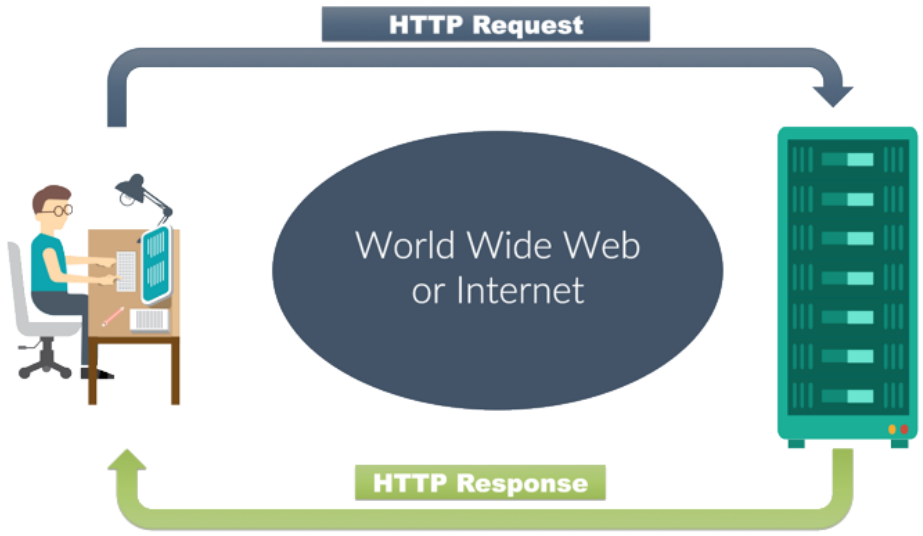
Google Places API Web Service tarjoaa samat palvelut kuin Google Places API for Android, mutta tietojen hakemisessa hyödynnetään HTTP viestien vaihtoa. Jokaiseen palveluun yhdistetään HTTP-pyynnön kautta johon palvelu palauttaa vastauksen. Pyynnöissä tulee olla sisällytettynä API avain pyynnön valtuuttamista varten. (Places API Web Service)

Kehittäjän tulee siis itse huolehtia kyselyiden muodostamisesta ja että kyselyissä käytettävät URL-osoitteet ovat oikein. Lisäksi kehittäjän tulee huolehtia, kyselyn lähettämisestä, vastauksen käsittelystä ja tiedon tulostamisesta käyttäjän luettavaksi. Toiminnot voitaisiin kuitenkin hoitaa taustalla. Tästä syystä käytännön osuudessa käytettäväksi teknologiaksi valittiin Google Places API Web Service.

4 HTTP -VIESTINTÄ WWW -OHJELMOINTIRAJAPINTOJEN KANSSA

WWW -pohjaisten ohjelmointirajapintojen käyttö tapahtuu HTTP -viestinnän avulla, jossa ohjelmointirajapinta vastaanottaa HTTP -pyynnön ja palauttaa HTTP -vastauksen. Opinnäytetyön käytännönosuus ei vaadi täyttä ymmärrystä mitä HTTP -viestien vaihdon aikana tapahtuu, mutta on hyvä ymmärtää pääpiirteittäin mistä osista pyynnöt muodostuvat.

Kuva 2 yksinkertaistettuna, asiakas tai asiakassovellus lähettää HTTP -pyynnön palvelimelle verkon välityksellä, johon palvelin vastaa palauttamalla HTTP -vastauksen.



Kuva 2. HTTP-pyyntö ja vastaus internetin välityksellä (HTTP Request and Response Over Web)

4.1 HTTP -pyyntö

HTTP -pyyntö muodostuu metodista, resurssin tunnisteesta ja protokolla versiosta. Metodi määrittää mitä pyynnön kohteena olevalle datalle halutaan tehdä. Yleisimpiä metodeja ovat hae (GET), lähetä (POST), päivitä (PUT) ja poista (DELETE). Tunniste eli URI (Uniform Resource Identifier) toimii palvelimella sijaitsevan resurssin uniikkina tunnisteena, johon metodia käytetään. Protokolla versio määrittää mitä protokollaa ja mitä versiota käytetään pyynnön lähetyksessä. (Hypertext Transfer Protocol -- HTTP/1.1.)

4.2 HTTP -vastaus

Palvelin vastaa HTTP -pyyntöön palauttamalla HTTP -vastauksen. Vastaus muodostuu protokolla versiosta, statuskoodista ja selitteestä sekä teksti-

muodossa olevasta vastauksesta, jos pyyntö oli onnistunut. Protokolla versio kuvaa palvelimen käyttämää protokollaa. Statuskoodin ja selitteen avulla kerrotaan käyttäjälle pyynnön onnistumisen tila. Jos pyynnölle löytyi vastausta, se palautetaan tekstimuodossa. (Hypertext Transfer Protocol -- HTTP/1.1.)

4.3 **HTTP -statuskoodit**

HTTP -vastauksen mukana saapuva statuskoodin avulla kerrotaan käyttäjälle, onnistuiko haku vai menikö jotain vikaan. Statuskoodit ovat kolme numeroisia ja ne luokitellaan viiteen luokkaan ensimmäisen luvun mukaan. Nämä luokat kuvaavat informaatiota (1xx), onnistumista (2xx), uudelleenohjausta (3xx), virhettä asiakkaan puolella (4xx) ja virhettä palvelin puolella (5xx). (Hypertext Transfer Protocol -- HTTP/1.1.)

5 KÄYTETTÄVÄT GOOGLE OHJELMOINTIRAJAPINTA PALVELUT

Tämän opinnäytetyön käytännön osuudessa käytettiin kolmea Google Places API palvelua: Place Search, Place Details ja Place Photos. Place Search palvelua käytetään paikkojen hakemiseen yleisellä tasolla ja se sisältää kaksi eri hakutoimintoa, Nearby Search ja Text Search.

Nearby Search vastaanottaa pakollisina parametreina API avaimen, sijainnin ja säteen. Hakua voidaan tarkentaa vaihtoehtoisilla parametreilla kuten avainsana, kieli, tyyppi ja onko paikka auki. Palvelu palauttaa listan paikoista, jotka ovat sijainnista lasketun kantaman sisäpuolella.

Text Search vastaanottaa pakollisina parametreina hakusanan ja API avaimen. Vaihtoehtoisina parametreina voidaan syöttää muun muassa aluekoodi, sijainti, säde, kieli, tyyppi ja hintaluokka. Palvelun palauttama vastaus sisältää listan paikoista, jotka vastaavat hakusanaa ja muita annettuja parametreja. (Places API Web Service, Place Search.)

Place Details palvelua käytetään tarkempien tietojen kyselyyn. Palvelu vastaanottaa pakollisina parametreina API avaimen ja paikka id. Paikka id on paikan yksilöivä tunniste, joka voidaan selvittää Place Search palvelun avulla. Vaihtoehtoisina parametreina voidaan syöttää kieli ja aluekoodi. Palvelu palauttama vastaus sisältää tarkempaa tietoa paikasta, kuin mitä Place Search palauttaa. (Places API Web Service, Place Details.)

Place Photos palauttaa kuvan viitettä vastaavan kuvan. Kuvan viite toimii kuvan id:nä Googlen tietokannassa. Nearby ja Text Search vastaukset sisältävät kustakin kohteesta yhden kuvan. Details Search vastaus sisältää listauksen kaikista paikan kuvien viitteistä.

Place Photo vastaanottaa pakollisina parametreina kuvan viitteen, kuvan maksimi korkeuden ja leveyden sekä API avaimen. Palvelun palauttaa viittausta vastaavan kuvan, jonka tyyppi määräytyy alkuperäisen kuvan mukaan. (Places API Web Service, Place Photos.)

6 KÄYTETTÄVÄT TYÖKALUT

6.1 Visual Studio 2017

Visual Studio 2017 on Microsoftin kehittämä ohjelmointiympäristö, joka mahdollistaa melkein minkä tahansa ohjelmakoodin tarkastelun ja editoimiseen Mac- ja Windows-tietokoneilla. Visual Studio kykenee automaattisesti analysoimaan ohjelmakoodia, jonka ansiosta Visual Studio kykenee täydentämään kehittäjän kirjoitusta ja huomauttamaan kehittäjää virheistä sekä tarjoamaan mahdollisia ratkaisuja. (Get started with Visual Studio 2017.)

6.2 Xamarin

Xamarin on vuonna 2011 perustettu ja vuonna 2016 Microsoftin ostama ohjelmistoyritys, joka tarjoaa ohjelmistokehittäjille työkalut, jotka auttavat heitä luomaan alustasta riippumattomia (cross-platform) mobiilisovelluksia. Sovellukset voivat omaa natiiveja ominaisuuksia, mutta silti niillä voi olla yhteinen koodipohja.

Xamarin on ladattavissa Visual Studion kanssa ilmaiseksi. Sen avulla ohjelmistokehittäjät voivat luoda Android-, iOS- ja Windows -sovelluksia käyttämällä Visual Studiota ja C# -ohjelmointikieltä. Toisin sanoen, Xamarin sisältää Android ja iOS SDKt muunnettuna C#:lle. Yhteisen ohjelmointikielen ansiosta, ohjelmistokehittäjän ei tarvitse erikseen opetella Java-, Objective-C- tai Swift- ohjelmointikieliä.

Ohjelmistokehittäjä voi lisätä toimintoja sovellukseensa lataamalla liitännäisiä. Tämä mahdollistaa suosituimpien backend -ominaisuuksien lisäämisen, kuten Microsoft Azure. Suurin osa liitännäisistä käyvät kaikille alustoille, mutta alustakohtaisia liitännäisiä löytyy myös. (What is Xamarin?.)

Voidakseen luoda Android-sovelluksia, ohjelmistokehittäjällä tulee olla Java ja Android SDKt sekä Visual Studion 2015 tai 2017 versio asennettuna. SDKt tarjoavat kääntäjän, emulaattorin ja muita työkaluja joita tarvitaan sovelluksen rakentamiseen, ajamiseen ja testaamiseen.

iOS-sovelluksia luodakseen. ohjelmistokehittäjän tulee omistaa Mac-tietokone, jossa on macOS. Sovelluksia voidaan luoda Visual Studiolla, mutta sovelluksen rakentaminen, ajaminen ja testaaminen vaativat Mac-tietokoneen lisenssisyistä.

Windows-sovellukset rakennetaan suoraan Visual Studiossa, joten Xamarin ei ole välttämätön. C#-koodi voidaan kuitenkin jakaa Windows, Android ja iOS alustojen kesken. (Understanding the Xamarin Mobile Platform.)

6.3 **C# -ohjelmointikieli**

C# on objektiorientoitunut kieli, jonka ansiosta ohjelmistokehittäjät voivat luoda suuren määrän erilaisia sovelluksia. C# voidaan käyttää moneen eri tarkoitukseen, kuten luomaan Windows sovelluksia, XML verkkopalveluja ja tietokantasovelluksia. (Introduction to the C# Language and the .NET Framework.)

Syntaksiltaan C# muistuttaa C, C++ ja Java ohjelmointikieliä. Ohjelmistokehittäjät, jotka osaavat jotain näistä kielistä voivat helposti siirtyä työskentelemään C#:la lyhyen ajan sisään. C# syntaksi yksinkertaistaa C++:n sekavuutta ja sisältää ominaisuuksia mitä Javasta ei löydy, kuten tyhjät arvotyypit, luettelointi, lambda ilmaukset ja suora muistiyhteys. (Introduction to the C# Language and the .NET Framework.)

6.4 **Android**

Alun perin Googlen kehittämä Linux pohjainen avoimen lähdekoodin Android on sovelluskokoelma, joka sisältää käyttöjärjestelmän, kolmannen osapuolen kuten laitevalmistajan ohjelmistoja ja Googlen kehittämät ohjelmistot. (Google Android: An Emerging Software Platform For Mobile Devices.)

Android käyttöjärjestelmä on tämän opinnäytetyön kirjoitushetkellä suosituin älylaitteissa käytettävä käyttöjärjestelmä. Android soveltuu puhelimien lisäksi muihinkin laitteisiin, kuten tabletteihin, älytelevisioihin, ja älykelloihin. Koska Android on avointa lähdekoodia ja ilmainen, laitevalmistajat voivat vapaasti muokata sitä. Google on lisensoinut joitain käyttöjärjestelmän ominaisuuksia, mutta jotkut valmistajat ohittavat nämä ja käyttävät vain ilmaista osuutta. (What is Google?.)

7 KÄYTÄNNÖN OSUUS

Opinnäytetyön käytännönosuudessa implementoitiin paikkatietojen haku DigiTrail-sovellukseen Google Places API Web Servicen avulla. Käytännönosuus sisältää Google API-avaimen luomisen ja käyttöönoton, http -pyynnön muodostamisen ja lähettämisen, http-vastauksen vastaanottamisen, vastaan otetun tiedon käsittelyn ja tulostamisen laitteen näytölle.

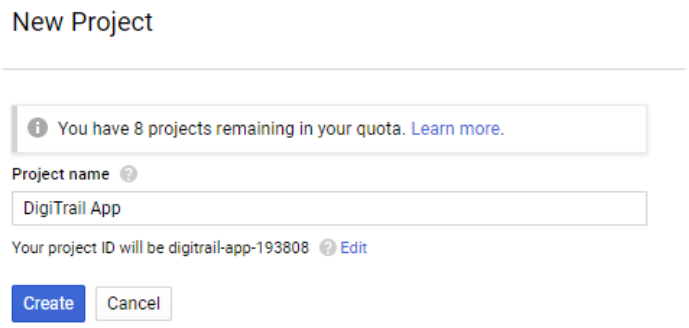
Käytännönosuudessa on sisällytettynä kuvakaappauksia ohjelmakoodista. On hyvä huomioida, että nämä kaappaukset ovat hyvin yksinkertaisia eivätkä sisällä juuri minkäänlaisia tarkistuksia tarvittavan tilan minimoimiseksi.

7.1 Google API avaimen käyttöönotto

Ennen kyselyiden suorittamista, projektia varten täytyi luoda Google API avain. Avaimen luomisen lisäksi API avaimeen piti lisätä APIt, joita sovelluksen käytössä tultaisiin tarvitsemaan.

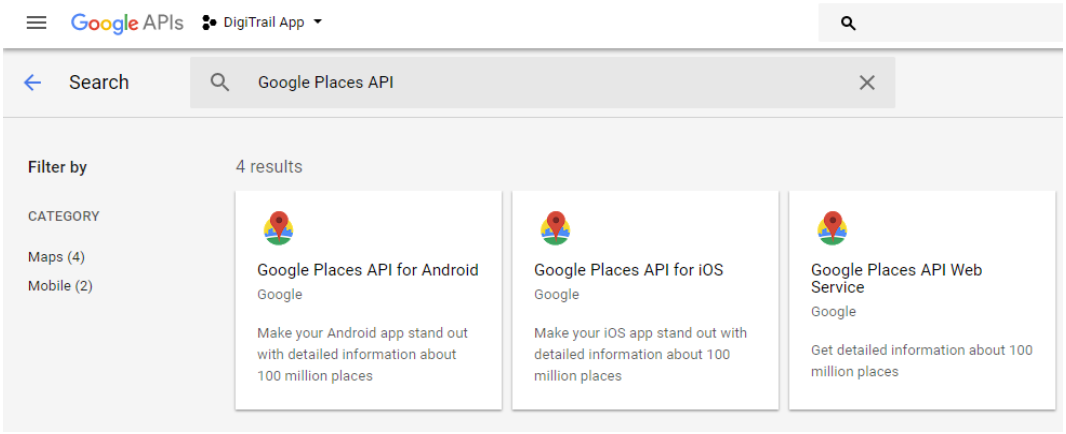
Google API avain luotiin Google API Console palvelussa, console.developers.google.com/apis. Palvelun käyttöä varten kirjauduin palveluun Google-tunnuksilla. API avainta varten loin uuden projekti resurssien hallinnasta. Annoin projektille nimeksi DigiTrail App.

On hyvä huomioida, että kirjauduin palveluun käyttämällä omia Google tunnuksiani, koska minulla ei ollut DigiTrail-hankkeen Google-tunuksia. API avaimen hallintaoikeudet tultaisiin siirtämään myöhemmin Älykkäät palvelut -tutkimusyksikölle.



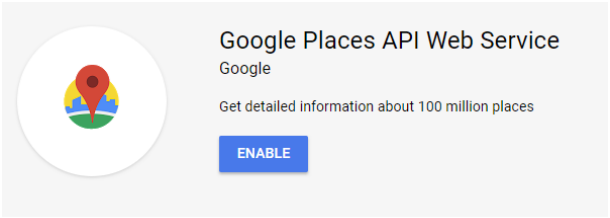
Kuva 3. Projektin luonti

Projektin luomisen jälkeen valitsin sovelluksessa käytettävät APIt. Sovelluksen tulisi hakea yritystietoja lähellä olevista yrityksistä, johon Google Places API Web Service sopisi hyvin. Hain kyseisen APIn suodattamalla kirjastoa hakukentän avulla. Kuva 4 suodatettu kirjasto.



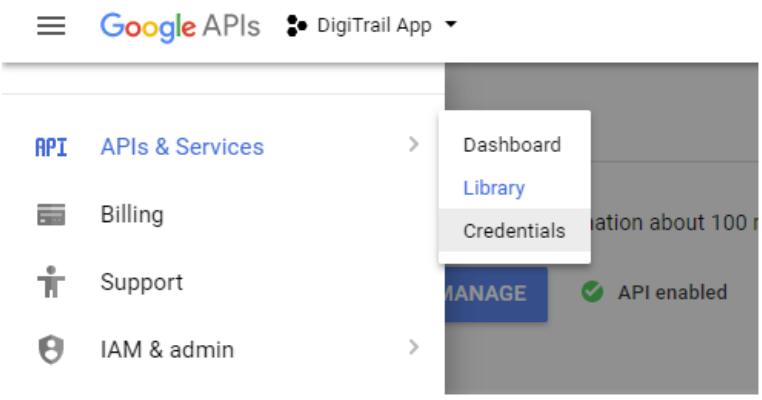
Kuva 4. Google Places API haku API kirjastosta

Valitsin käytettävän APIn, joka avasi kyseisen APIn infosivun. Tältä sivulta pystyin aktivoimaan APIn valitsemalla enable. Hetken odottelun jälkeen API oli aktivoitu ja sitä voitaisiin käyttää tällä avaimella. Jos sovelluksessa käytettäisiin muita APIa, niin ne voitaisiin aktivoida samalla. Tarvittaessa APIen määrää voidaan lisätä myös jälkikäteen.



Kuva 5. Google Places API Web Service aktivointi

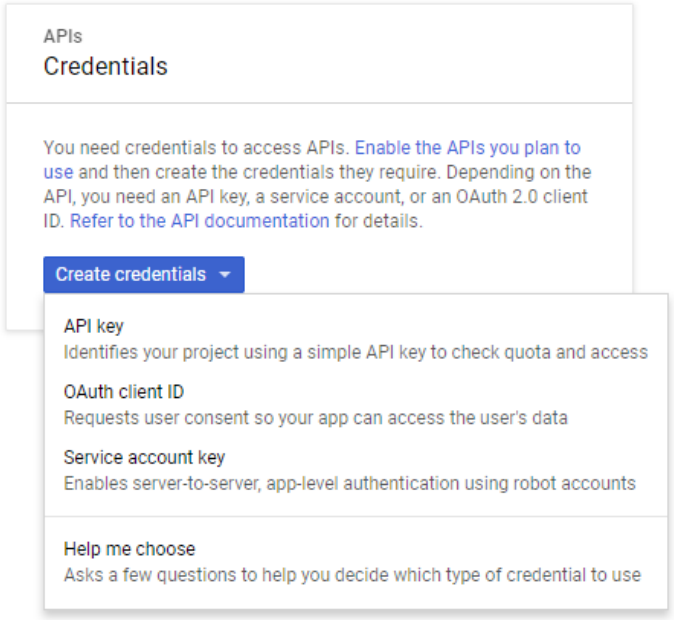
APIn aktivoimisen jälkeen loin avaimen käyttöoikeudet (credentials), jolla määriteltiin millä tavoin pyyntöjen vahvistaminen suoritetaan. Siirryin käyttöoikeuksien luomiseen valitsemalla APIs & Services ja credentials.



Kuva 6. Credentials näkymän valinta

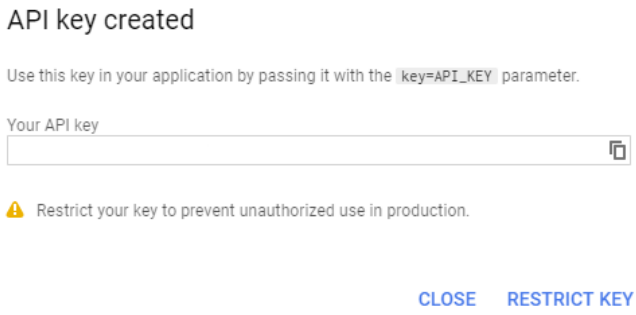
Aloitin käyttöoikeuksien luomisen valitsemalla create credentials. Käyttöoikeusien luomisessa valittiin millä tavoin API pyynnöt vahvistettaisiin. On

hyvä huomioida, että tavallisessa ja ei kaupallisessa käytössä pelkkä API avain riittää. Valitsin tässä tapauksessa API key vahvistustavaksi.



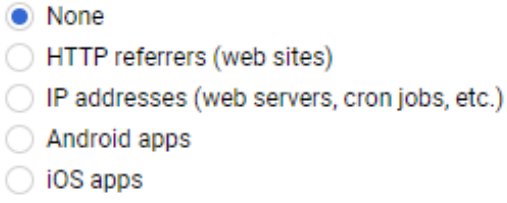
Kuva 7. API käyttöoikeuksien luonti

Käyttöoikeuksien luomisen jälkeen palvelu loi API avaimen automaattisesti. Avain oli kopioitavissa ja valmis käytettäväksi. Kuva 8 API avain on piilotettu tietosuojasyistä. Lopuksi valitsin close.



Kuva 8. Luotu API avain

Avaimen käyttöä voitaisiin tarvittaessa rajoittaa valitsemalla Restrict Key. Rajoituksia voitaisiin asettaa myöhemmin avaamalla avain muokattavaksi Credentials näkymässä. Kuva 9 valittavana olevat rajoitteet.



Kuva 9. API -avaimen rajoitteet

7.2 HTTP -pyynnön muodostaminen ja lähetys

HTTP-pyyntöjen muodostamista varten piti muodostaa URL-osoite, joka riippuu palvelusta, johon pyyntö osoitetaan. Lisäksi sallitut ja pakolliset parametrit muuttuvat palvelun mukaan.

Koska eri hakuvaihtoehtoja oli useita, päätin luoda jokaiselle haulle oman metodin, jotka muodostaisivat URL-osoitteen ja kutsuisivat yhteistä CreateHttpRequest metodia, joka vastaanottaisi URL-osoitteen, suorittaisi HTTP-pyynnön ja palauttaisi vastauksen.

Kuva 10 esimerkki julkisesta TextSearch metodista. Metodi vastaan ottaa kutsun yhteydessä lähetettävän hakutekstin (query), josta metodi ensin poistaa tyhjät välilyönnit ja korvaa ne + merkillä. Seuraavaksi metodi muodostaa URL-osoitteen, kutsuu CreateHttpRequest metodia, joka vastaan ottaa muodostetun URL-osoitteen. Lopuksi metodi vastaan ottaa CreateHttpRequest metodin palauttaman HTTP-vastauksen (result) ja palauttaa tämän vastauksen alkuperäiselle kutsujalle.

```
public async Task<string> TextSearch(string query) {

    query = query.Replace(" ", "+");

    var searchType = "textsearch";
    var output = "json";

    string url = "https://maps.googleapis.com/maps/api/place/"
        + searchType + "/" + output
        + "?query=" + query
        + "&key=" + API_KEY;

    var result = await CreateHttpRequest(url, output);
    return result;
}
```

Kuva 10. HTTP URL osoitteen muodostaminen

URL-osoitteen luonnin jälkeen muodostin HTTP-pyynnön käyttämällä HttpClient-luokkaa, joka on sisällytettynä .NET ja se on tuettuna .NET, Android ja iOS -alustoilla. On hyvä huomioida, että on olemassa muitakin tapoja luoda HTTP-pyyntöjä kuten Android Volley tai iOS NSURLSession, mutta nämä kaksi esimerkkiä ovat alustakohtaisia luokkia.

Lisäsin HttpClient-luokan käyttämällä System.Net.Http-nimiavaruutta. Muodostin HTTP-pyynnön luomalla HttpClient-luokasta uuden client nimisen olion. Seuraavaksi loin HttpResponseMessage-luokasta response nimisen olion. Seuraavaksi kutsuin client olion GetAsync metodia, jolle annoin

parametriksi aikaisemmin luodun HTTP URL osoitteen. GetAsync metodi suorittaa pyynnön parametrina annettuun HTTP URL osoitteeseen ja palauttaa HttpResponseMessage tyyppisen arvon, joka syötetään response olion arvoksi.

Asynkronisen verkkometodit kannattaa käsitellä try/catch -lohkon sisällä poikkeuksien käsittelyä varten. Avainsana await keskeyttää suorituksen, kunnes metodi on suoritettu loppuun. Kuva 11 koodinäyte.

```
private async Task<dynamic> CreateHttpRequest(string url, string output) {

    try {
        HttpClient client = new HttpClient();
        HttpResponseMessage response = await client.GetAsync(url);
```

Kuva 11. HTTP-pyynnön muodostaminen

Seuraavaksi tarkistin http-vastauksen mukana tulevan statuskoodin.
Jos statuskoodin tulos on onnistunut eli 200, suoritetaan vastauksen tallettaminen muuttujaan json tai photo riippuen mikä on haluttu ulostulo. Käytettävä metodi määräytyy output muuttujan arvon mukaan, jonka arvo määräytyy sen mukaan, mikä hakumetodi kutsui tätä metodia. Lopuksi metodi palauttaa vastauksen sisältävän muuttujan.

```
if (response.IsSuccessStatusCode) {
    switch (output) {
        case "json":
            var json = await response.Content.ReadAsStringAsync();
            return json;

        case "photo":
            var photo = await response.Content.ReadAsByteArrayAsync();
            return photo;
    }
```

Kuva 12. HTTP-vastauksen vastaanottaminen

7.3 HTTP-vastauksen käsittely

Teoriassa JSON olisi lukukelpoinen string tyyppisenä muuttujana, mutta se ei ole käytännöllistä. Haluttujen tietojen lukemisen ja tulostamisen helpottamiseksi JSON täytyisi ensin muuntaa objekteiksi. Tätä varten piti luoda JSON-malli, jonka jälkeen JSON-tiedosto voitaisiin muuntaa (deserialize) objekteiksi.

JSON-tiedostojen käsittelyyn on eri työkaluja kuten System.Json, joka on .NET sisäinen JSON-tiedostojen käsittelyyn kehitetty nimiavaruus. Tässä opinnäytetyössä kuitenkin päädyin käyttämään Newtonsoft kehittämää avoimen lähdekoodin Json.NET, koska se on suosittu ja sille löytyi runsaasti ohjeita.

7.3.1 JSON-mallin luominen

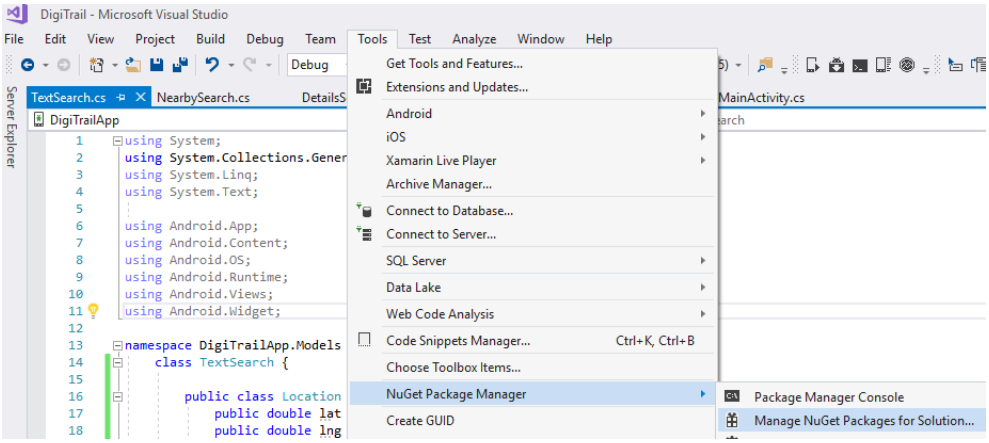
JSON-mallilla tarkoitetaan luokkarakennetta, jonka avulla JSON-tiedosto voidaan muuntaa objekteiksi ohjelmakoodissa. JSON-mallin tulee vastata HTTP-pyynnöllä saatua JSON tiedoston rakennetta. Liitteessä 3 JSON-malli, joka vastaa rakenteeltaan yhtä Google Places API Web Servicen lähettämistä JSON vastauksista liitteessä 2. Jokaista käytettävää palvelua kohtaan, tulee olla luotuna oma JSON-mallinsa.

Aloitin mallin luomisen luomalla RootObject-luokan, joka on päällimmäisin kerros. Toimiakseen mallin ylemmästä kerroksesta pitäisi aina viitata alempaan kerrokseen. Lisäksi kunkin kerroksen pitää sisältää JSON-tiedostossa kuvatut saman nimiset muuttujat, esimerkiksi liitteessä 3 RootObject sisältää html_attribution ja status muuttujat sekä viittauksen listaan Result luokkia. Result-luokka puolestaan sisältää omat muuttujansa ja viittauksen alempiin luokkiin.

Onnekseni löysin online työkalun json2csharp.com, joka on kehitetty luomaan JSON malleja C#:lle JSON-tekstistä tai JSON-URL:sta. Tämä nopeutti mallien luomista huomattavasti ja ehkäisi mahdollisten virheiden syntymiset.

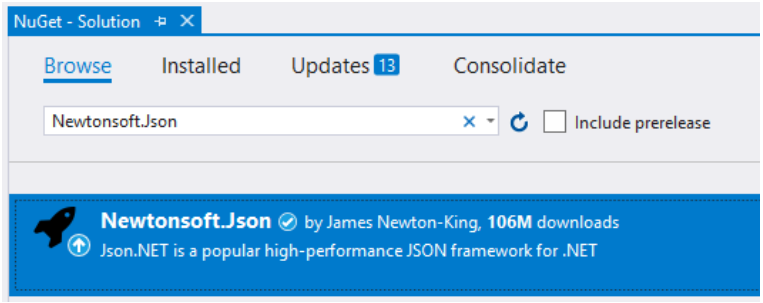
7.3.2 Newtonsoft.Json NuGet asennus

Lisäsin Json.NET -paketin projektiin NuGet -pakettien hallinnasta. Siirryin Pakettien hallintaan valitsemalla Tools, NuGet packet manager, jonka jälkeen Manage NuGet packages for Solution.



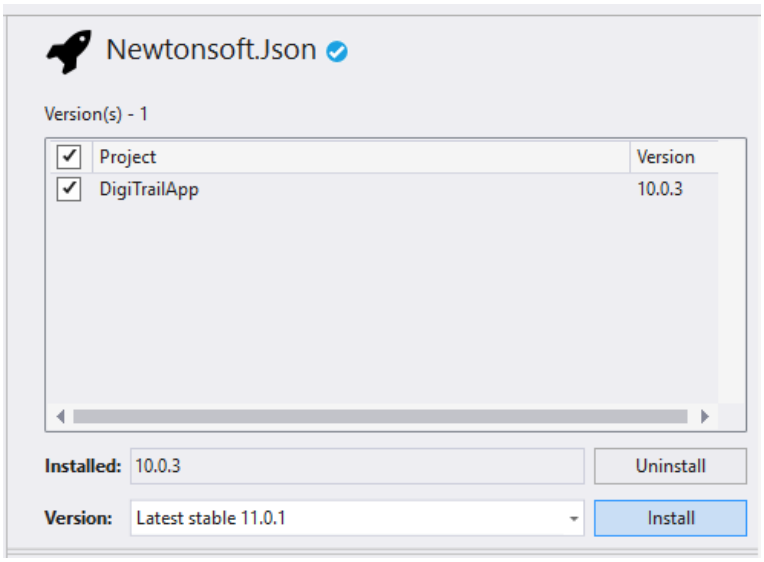
Kuva 13. NuGet pakettien hallinnan avaaminen

Suoritin haun Browse välilehdellä hakutermillä Newtonsoft.Json. Valitsin hakutermiä vastaava samanniminen NuGet-pakettin, joka oli haun ensimmäinen osuma.



Kuva 14. Newtonsoft.Json paketin hakeminen

Paketin valitsemisen jälkeen valitsin projektin johon NuGet paketi tultaisiin asentamaan. Valitsin viimeisimmän version, jonka jälkeen valitsin Install.



Kuva 15. Newtonsoft.Json paketin asennus

Paketin asennuksen aikana Visual Studio pyysi lupaa tehdä projektiin muutoksia, johon vastasin ok. Asennuksen päätyttyä pakettia pystyttäisiin käyttämään projektissa.

7.3.3 Vastauksen muuttaminen objekteiksi

Vastauksen muuntamista varten loin oman luokan, jonka nimesin JsonHandler. Loin luokalle neljä metodia: ParseTextResponse, ParseNearbyResponse ja ParseDetailsResponse. Kukin metodi vastaanottaisi string tyyppisen json nimisen muuttujan joka muutettaisiin objekteiksi. Lopuksi metodi palauttaisi rootObject olion, jonka avulla tietoja voitaisiin tulostaa laitteen näytölle. Kuva 16 esimerkki ParseNearbyResponse metodista.

```
static public NearbySearch.RootObject ParseNearbyResponse(string json)
{
    try {
        NearbySearch.RootObject rootObject = JsonConvert.DeserializeObject<NearbySearch.RootObject>(json);

        return rootObject;

    } catch (JsonException e) {
        Log.Error(tag, e.ToString());
    }

    return null;

}
```

Kuva 16. ParseNearbyResponse metodi

7.4 Lähellä sijaitsevien paikkojen hakeminen

Käytännönosuutta tehdessä sovellus oli vielä kovan kehitystyön alla. Loin paikkojen hakemisen testaamista varten oman painikkeensa joka tulisi näkymään sovelluksen kehitysversiossa. Sovelluksen kehitystyön jatkuessa paikkojen hakemisesta huolehtiva logiikka tultaisiin siirtämään järkevämpään kohtaan ohjelmakoodissa mutta itse logiikka tulisi pysymään samana.

Aloitin kutsumalla GetCurrentLocation metodia. Metodin palauttama laitteen sen hetkinen sijainti sijoitetaan muuttujan location arvoksi. Tarkistin muuttujan arvon tyhjän arvon varalta. Arvon ollessa tyhjä täytyisi tarkistaa voidaanko laitteen sijainti lukea onnistuneesti.

Kyselyä varten leveys- ja pituusasteet (latitude ja longitude) täytyy erottaa location muuttujasta ja asettaa omiin muuttujiinsa. Tämän lisäksi kyselyn muodostustavasta johtuen desimaalierottimet tulee korvata pisteellä pilkun sijasta. Lopuksi kutsutaan haluttua HttpRequest luokan sisältämää metodia joka saa parametreikseen leveys- ja pituusasteet sekä muut metodikohtaiset parametrit. Metodin palauttama vastaus sijoitetaan muuttujan json arvoksi. Kuva 17 esimerkki pyynnön alustuksesta.

```
var location = GetCurrentLocation();
if (location != null) {

    var lat = location.Latitude.ToString();
    lat = lat.Replace(",", ".");
    var lng = location.Longitude.ToString();
    lng = lng.Replace(",", ".");

    var json = await HttpRequest.NearbySearch(lat + "," + lng, 50, null);
```

Kuva 17. Nearby Search pyynnön alustus.

Viimeiseksi tarkistetaan json muuttuja tyhjän arvon varalta. Jos arvo ei ole tyhjä jatketaan kutsumalla JsonHandler-luokan parse-metodia, jonka palauttama arvo sijoitetaan rootobject olion. Kuvassa 18 esimerkki metodin kutsusta.

```
if (json != null) {
    var rootobject = JsonHandler.ParseNearbyResponse(json);
```

Kuva 18. Nearby Search vastauksen käsittely

7.5 Paikkatietojen tulostaminen laitteen näytölle

Paikkatietojen näyttämistä varten loin kustakin haussa löytyneestä kohteesta oman karttamerkkinsä. Kävin rootobject olion sisältämän results taulukon läpi hyödyntämällä foreach-lauseketta. Erotin kunkin paikan sijainnin pituus- ja leveysasteet joista muodostin muuttujan latLng. Tämän lisäksi erotin paikan nimen ja paikan tunnisteen. Lopuksi kutsuin metodia joka vastaanottaisi parametreina paikan sijainnin, tunnisteen ja nimen sekä karttamerkissä käytettävän kuvakkeen ja loisi sovelluksen kartalle karttamerkin mitä painamalla käyttäjä näkisi paikan nimen ja tunnisteen. Kuva 19 esimerkki karttamerkin luomisesta.

```
if (rootobject != null) {

    try {
        foreach (var place in rootobject.results) {

            var latLng = new LatLng(place.geometry.location.lat, place.geometry.location.lng);
            var iconFactory = IconFactory.GetInstance(this);
            var icon = iconFactory.FromResource(Resource.Drawable.markerDefault);

            AddMarkersInMap(latLng, place.place_id, place.name, icon);
```

Kuva 19. Karttamerkin luominen

Testasin ohjelmakoodin toimivuutta. Sovellus suoritti pyynnön Googlen palvelimelle, sai vastauksen ja loi karttamerkit vastauksen pohjalta. Liitteessä 4 ruudunkaappaus sovelluksesta ja avatusta karttamerkistä.

Tässä vaiheessa työ seisahtui koska paikkatietojen kyselyn kohtaa ohjelmakoodissa ei oltu suunniteltu, saati mitä tietoja haluaisimme näyttää käyttäjälle. Lisäksi sovelluksen tulisi pystyä toimimaan offline tilassa, joka tuottaisi ongelmia paikkatietojen hakemisen suhteen.

Pohjimmaisena ideana karttamerkkiä painettaessa toiseen kertaan suoritettaisiin uusi kysely käyttäen DetailSearch pyyntöä ja paikan tunnistetta. Tunnisteen avulla pyyntöä vastaava vastaus sisältäisi tarkempaa tietoa kohteesta, kuten aukioloaikoja, arvosteluja ja kuvien tunnisteita, joilla puolestaan voitaisiin hakea kohteen kuva. Tässä kohtaa työ kuitenkin jäi kesken, mutta se tultaisiin viimeistelemään joskus tulevaisuudessa. Liitteessä 4 kuvassa työn vaiheesta.

8 YHTEENVETO

Opinnäytetyössä saatiin selville mitä ovat ohjelmointirajapinnat ja www-sovelluspalvelut, miten ne toimivat ja miten niiden kanssa kommunikoidaan. Opinnäytetyössä perehdyttiin tarkemmin Googlen ohjelmointirajapintoihin ja www-sovelluspalveluihin. Käytännön työssä perehdyttiin paikkatietojen kyselyn suorittamiseen käytännössä

Opinnäytetyössä onnistuttiin vastamaan kaikkiin siinä esitettyihin tutkimuskysymyksiin. Teoriassa käsiteltiin Google Places API for Android- ja Google Places API Web Service -teknologioita sekä miten nämä kaksi teknologiaa eroavat toisistaan. Lisäksi pohdittiin kumpaa näistä kahdesta voisi hyödyntää käytännön työssä.

Käytännön työssä käytiin läpi, miten paikkatietojen kysely käytännössä toimii ja suoritettiin paikkatietojen kyselyn implementointi DigiTrail-sovellukseen. Käytännön työ sisälsi luokan kirjoittamisen URL-osoitteiden muodostamista ja HTTP-pyyntöjen lähettämistä varten, json-mallien luomisen, vastauksen käsittelyluokan kirjoittamisen, lähellä olevien paikkojen näyttämisen sovelluksessa ja haluttujen paikkatietojen tulostamisen laitteen näytölle.

Opinnäytetyötä tehdessä havaittiin Google Places API Web Servicen hyödyllisyys paikkatietojen kyselyssä. Ohjelmointirajapintaa käyttämällä säästettiin paljon aikaa mitä olisi kulunut oman tietokannan luomiseen paikkatietoja varten.

Opinnäytetyötä tehdessä tekijä ei kokenut mitään varsinaisesti merkittäviä onnistumisen hetkiä. Vaikeuksia ilmeni jonkin verran karttapohjalle lisättävien merkkien kohdalla. DigiTrail-sovelluksessa käytettävälle MapBox-karttapohjalle ei löytynyt hirveästi ohjeita toisin kuin Google Maps -karttapohjalle. Tästä johtuen tekijän piti karsia potentiaalista sisältöä pois opinnäytetyöstä. Lisäksi keskeneräisestä suunnittelusta johtuen työ piti keskeyttää.

Opinnäytetyö oli mainio tilaisuus tutustua uuteen aiheeseen ja hyödyntää oppimaansa käytännössä. Opinnäytetyön tekeminen tarjosi tekijällensä paljon kokemusta, jonka ansiosta tekijä pystyi kehittämään itseään mobiilikehittäjänä, sovelluskehittäjänä ja työnsä raportoijana.

LÄHTEET

DigiTrail. Viitattu 25.1.2018.
http://www.hamk.fi/tyoelamalle/hankkeet/digitrail/Sivut/default.aspx

Get started with Visual Studio 2017. Viitattu 31.4.2018
https://tutorials.visualstudio.com/vs-get-started/intro

Google Maps APIs, FAQ. Viitattu 25.1.2018.
https://developers.google.com/maps/faq

Google API Client Libraries Viitattu 1.2.2018.
https://developers.google.com/api-client-library/

HTTP Request and Response Over Web. Viitattu 19.2.2018
https://img.webnots.com/2013/06/HTTP-Request-and-Response-Over-Web-1.png

Introduction to the C# Language and the .NET Framework. Viitattu 20.1.2018.
https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Hypertext Transfer Protocol -- HTTP/1.1.
https://tools.ietf.org/html/rfc2616#section-4

Places API Web Service, Place Details. Viitattu 16.2.2018.
https://developers.google.com/places/web-service/details

Places API Web Service, Place Photos. Viitattu 16.2.2018.
https://developers.google.com/places/web-service/photos

Places API Web Service, Place Search. Viitattu 16.2.2018.
https://developers.google.com/places/web-service/search

Places API for Android. Viitattu 26.1.2018
https://developers.google.com/places/android-api/start

Places API Web Service. Viitattu 26.1.2018
https://developers.google.com/places/web-service/intro

Places API Web Service, Usage Limits and Billing. Viitattu 26.1.2018
https://developers.google.com/places/web-service/usage

Sanastokeskus TSK: Termipankki. Viitattu 13.2.2018
http://www.tsk.fi/tepa/fi/haku/verkkopalvelu

TechTerms, API, 2016. Viitattu 12.2.2018
https://techterms.com/definition/api

TechTerms, Web Service. Viitattu 13.2.2018
https://techterms.com/definition/web_service

Understanding the Xamarin Mobile Platform, Xamarin. Viitattu 30.1.2018
https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_1_-_understanding_the_xamarin_mobile_platform/

W3C Working Group Note, 2004. Viitattu 13.2.2018
https://www.w3.org/TR/ws-arch/

What is Google?, Marziah Karch. Viitattu 30.1.2018
https://www.lifewire.com/what-is-google-1616386

What is Xamarin? The Windows Club. Viitattu 30.1.2018
http://www.thewindowsclub.com/what-is-xamarin-and-cross-platform-mobile-development

Liite 1

DigiTrail Android-mobiilisovellus suunnitelma

DigiTrail Android-mobiilisovellus suunnitelma

Antti Juntunen

Sovelluksen kuvaus

DigiTrail Android-mobiilisovelluksella on tarkoitus tuoda näkyvyyttä reitistöille ja paikallisille yrityksille. DigiTrail sovellus opastaa käyttäjää liikkumaan valituilla luontopoluilla, sekä ilmoittaa läheisitä yrityksistä ja nähtävyyksistä.

Sovelluksen toiminnot

1. opastaa valitulle luontopolulle (Google Navigaattori)
2. näyttää luontopolut ja opastaa valituilla luontopolulla
3. näyttää tietoikkunan, kun käyttäjä on tarpeeksi lähellä nähtävyyttä tai yritystä (Yritysten tiedot Google places apilla?)
4. toimii myös taustalla, joten käyttäjä saa ilmoituksia, vaikka puhelin olisi taskussa
5. näyttää kuvaukset reiteistä, mahdollista katsoa videoita erillisestä linkistä
6. ilmoittaa akun varauksen ollessa heikko? ilmoittaa myös, jos on lähdössä pitkälle polulle ja akku ei ole täynnä? sekä huomauttaa käyttäjää, että sovellus vie paljon virtaa akusta
7. käyttäjää huomautetaan, jos GPS tarkkuus ei ole asetettu tarkimmalle asetukselle, heikolla tarkkuudella sovellus ei toimi toivotulla tavalla
8. kun käyttäjä on kulkenut reitin hän voi arvostella sen asteikolla 1-5.
9. näkymä palveluntarjoajien elämykset, joista käyttäjä voi tarkastella esim. melonta mahdollisuuksia
10. reiteistä reittikuvaus, korkeusero, matka, vaikeusaste, ym.
11. säätiedot kyseiselle reitille
12. sovelluksessa näkyy logot (EU ja HAMK)
13. suodatus mitä markkereita kartalla näkyy (esim. nuotiopaikat, vessat ym.)
14. sovelluksen tulosten jakaminen facebookkiin (kuljettu matka ym. tietoja)
15. reiteille eri teemoja (teemoitetuilta reiteiltä voi kerätä pisteitä ym.)
16. käyttäjä voi antaa kirjallista palautetta sovelluksesta

Haasteet

Haasteena on karttapohjan (openstreetmap) päivittäminen ja luontopolkujen ympäristön kartoittaminen, usealta reitistöltä puuttuu polkuja, metsää, taloja ym. lisäksi sovelluksessa käytettävä MapBox Android SDK on hieman puutteellinen C# ohjelmointikielelle.

Liite 2

Google Places API Web Service Response


```
{
   "html_attributions": [],
   "results": [
      {
         "formatted_address": "Visamäentie 35, A, 13100 Hämeenlinna, Finland",
         "geometry": {
            "location": {
               "lat": 60.976933,
               "lng": 24.47591
            },
            "viewport": {
               "northeast": {
                  "lat": 60.978283,
                  "lng": 24.477259
               },
               "southwest": {
                  "lat": 60.975582,
                  "lng": 24.47456
               }
            }
         },
         "icon": "https://maps.gstatic.com/mapfiles/place_api/icons/school-71.png",
         "id": "eb8fdaf8e4cd46b8a9f176931813dc0c4d70a795",
         "name": "HAMK Häme University of Applied Sciences",
         "photos": [
            {
               "height": 640,
               "html_attributions": [
                  "Photos are copyrighted by their owners"
               ],
               "photo_reference": "CmRaAAAAXouBqaJlHuDdODRIGiJ0IfXP_RSx_ZEQyemXCOjMkUX0LY3cyj
               "width": 960
            }
         ],
         "place_id": "ChIJCUIYGn9djkYRvUIAwdU0l-Y",
         "rating": 4.5,
         "reference": "CmRbAAAAblxnL-G-DiSU4G9pjUIPZBsBiy_a6d5NLs7f6gInvaLWpG484K0DHZ_dKTT20d
         "types": [
            "university",
            "point_of_interest",
            "establishment"
         ]
      }
   ],
   "status": "OK"
}
```

Liite 3

JSON model esimerkki

```
public class Location {
    public double lat { get; set; }
    public double lng { get; set; }
}

public class Northeast {
    public double lat { get; set; }
    public double lng { get; set; }
}

public class Southwest {
    public double lat { get; set; }
    public double lng { get; set; }
}

public class Viewport {
    public Northeast northeast { get; set; }
    public Southwest southwest { get; set; }
}

public class Geometry {
    public Location location { get; set; }
    public Viewport viewport { get; set; }
}

public class Photo {
    public int height { get; set; }
    public List<string> html_attributions { get; set; }
    public string photo_reference { get; set; }
    public int width { get; set; }
}

public class Result {
    public string formatted_address { get; set; }
    public Geometry geometry { get; set; }
    public string icon { get; set; }
    public string id { get; set; }
    public string name { get; set; }
    public List<Photo> photos { get; set; }
    public string place_id { get; set; }
    public double rating { get; set; }
    public string reference { get; set; }
    public List<string> types { get; set; }
}

public class RootObject {
    public List<object> html_attributions { get; set; }
    public List<Result> results { get; set; }
    public string status { get; set; }
}
```

Liite 4

Paikkatietojen näyttäminen sovelluksen kartalla