LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

# VISION HUNT -MOBIILISOVELLUS

LAHDEN AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikka (Insinööri AMK)
Ohjelmistotekniikka
Syksy 2018
Ville Teräväinen

**Tiivistelmä**


| Tekijä(t) | Julkaisun laji | Valmistumisaika |
|---|---|---|
| Teräväinen, Ville | Opinnäytetyö, AMK | Syksy 2018 |
| | Sivumäärä<br>38 | |


Työn nimi

**Vision Hunt Mobiilisovellus**

Tutkinto

AMK Insinöörin tutkinto


Tiivistelmä

Opinnäytetyön tavoitteena on tutkia Google Cloud -pilvipalvelusta löytyviä tekoälyrajapintoja ja kehittää niitä hyödyntävä Android-sovellusprototyyppi. Prototyyppi nimeltään Vision Hunt kehitettiin mukana CitiCAP-hankkeessa, joka on Lahden kaupungin kehittämä hanke. CitiCAP-hankkeessa suunnitellaan ja testataan asukkaiden henkilökohtaista päästökauppaa Lahden kaupungissa. Päästökauppaa käydään puhelinsovelluksella, joka käyttää puhelimen sensoreita tunnistamaan eri liikkumismuodot. Prototyyppi suunniteltiin puhelinsovellusta varten kolmannen osapuolen palveluksi, jota käyttäjät voivat vaihtoehtoisesti lisätä sovellukseen.

Vision Huntin kehitystä varten tutkittiin Google Cloud pilvipalvelusta löytyviä tekoälyrajapintoja ja palveluita. Google Cloud Vision API tekoälyrajapintaa käytetään toteuttamaan prototyyppiin ominaisuus, jolla analysoidaan sisältöä kuvista. Tätä tekoälyrajapintaa tutkitaan opinnäytetyössä, parin muun Google Cloudissa olevan rajapinnan lisäksi. Tämän lisäksi tutkitaan Microsoft Azuren ja Amazon Web Servicen vastaavat tekoälyrajapinnat. Vision Hunt prototyyppi toteutetaan Android-alustalle Java pohjaisena sovelluksena. Android-alustalle kehittäminen käydään läpi opinnäytetyön Android-osuudessa.

Vision Hunt prototyypistä esitellään sen tavoitteet, kehitys, toiminta ja ohjelmointi. Työssä esitellään Vision Huntin arkkitehtuuri ja mitä sovelluksella on tarkoitus tehdä. Jatkokehitys osuudessa esitellään seuraavan kehitysversion suunnitelmia, tavoitteita ja arkkitehtuuria.



Asiasanat

CitiCAP, Pilvipalvelu, Tekoälyrajapinta

**Abstract**


| Author(s) | Type of publication | Published |
|---|---|---|
| Teräväinen, Ville | Bachelor's thesis | Autumn 2018 |
| | Number of pages | |
| | 38 | |


Title of publication

**Vision Hunt Mobile Application**

Name of Degree

Bachelor's Thesis in software engineering


Abstract

The objective of the bachelor's thesis was to study artificial intelligence APIs found in the Google Cloud Platform and to develop an Android application prototype which uses AI APIs. The prototype named Vision Hunt was developed as part of the CitiCAP project, which is a project developed by the city of Lahti. The CitiCAP project is about developing and testing a personal carbon trade marketplace in the city of Lahti. The carbon trading is done with a phone application, which uses the sensors of the phone to identify different modes of travel. The prototype was created as a third party service to the phone application, which could be an optional addition to the application.

For Vision Hunt's development, research was done on the different AI APIs found in the Google Cloud Platform. Google Cloud Vision API was used to create a function in the prototype, which is used to analyze the contents of pictures. This AI API was studied in the thesis, in addition to a few others found in the Google Cloud Platform. In addition, the same kind of AI APIs found in Microsoft Azure and Amazon Web Services were explored. The Vision Hunt prototype was developed for the Android platform as a Java-based application. Developing for the Android platform was explored in the Android part of the thesis.



Keywords

CitiCAP , Cloud Platform, artificial intelligence API

SISÄLLYS

# 1 JOHDANTO

CitiCAP-hanke on Lahden kaupungin ylläpitämä hanke, jossa suunnitellaan ja testataan asukkaiden henkilökohtaista liikkumisen päästökauppaa Lahden kaupungissa. Päästökauppaa käydään mobiilisovelluksella, joka käyttää puhelimen sensoreita tunnistamaan käyttäjän eri liikkumismuodot. Hankkeeseen kuuluu myös älypyörätien rakennus matkakeskuksesta Apilakadulle. CitiCAP-hanke on toiminnassa vuodesta 2017 vuoden 2020 loppuun asti. Hanketta toteuttamassa ovat Lahden Kaupunki, Lappeenrannan ja Lahden teknillinen yliopisto (LUT), Lahden ammattikorkeakoulu, Lahden seudun kehitys LADEC sekä viisi liikkumisenalueella työskentelevää yritystä: Mattersoft, Moprim, Infotripla, Good Sign ja Future Dialog. Hanketta rahoitetaan pääosin EU:n Urban Innovatice Action -ohjelmasta. (Lahti 2018.)

Opinnäytetyön aiheena on luoda prototyyppi Android-sovelluksesta, jota voitaisiin käyttää kolmannen osapuolen palveluna CitiCAP-hankkeen puhelinsovelluksessa. Prototyypin toteutusta varten tutustutaan erilaisiin pilvipalveluista löytyviin tekoälyrajapintoihin ja siihen, miten niitä voidaan hyödyntää prototyypissä. Opinnäytetyössä tutkitaan Google Cloud -pilvipalvelusta löytyvää Cloud Vision API tekoälyrajapintaa, joka on kuvan tunnistusta varten käytettävä rajapinta pilvipalvelussa. Samaan aiheeseen liittyen esitellään muita Googlen pilvipalvelusta löytyviä tekoälyrajapintoja ja kahden toisen suuren pilvipalvelutarjoajan, Microsoftin ja Amazonin omia tekoälyrajapintoja.

Kesän 2018 aikana kehitetty prototyyppi Vision Hunt on peli Android-alustan puhelimille. Android on tällä hetkellä käytetyin älypuhelimien käyttöjärjestelmä, jonka toimintaa esitellään opinnäytetyössä (MyBroadBand 2018). Prototyypin ohjelmointi tehtiin Java ohjelmointikielellä, käyttäen Android Studio ohjelmaa sen ohjelmointiin. Vision Hunt pelissä käyttäjän tarkoituksena on kuvata puhelimen kameralla pelin pyytämiä asioita, jotka Vision Hunt lähettää analysoitavaksi pilvipalvelun tekoälykomponentille. Tehtävinä voi olla kuvata tavanomaisia asioita kuten puu tai järvi, joita pelaajan tulee etsiä kuvattavaksi. Tavoitteena oli kehittää sovellus, jota voi ruveta jatkokehittämään prototyyppi vaiheen jälkeen.

Viimeisessä osiossa käydään läpi Vision Huntin jatkokehitystä ja esitellään mahdollisia ideoita, miten sovellusta voitaisiin viedä eteenpäin. Vision Huntin kehitys kesti noin neljä kuukautta, josta ensimmäinen kuukausi koostui suunnittelutyöstä ja alustojen tutkimisesta. Loppuaika keskittyi prototyypin toteuttamiseen ja testaamiseen. Jatkokehitys alkoi syksyllä 2018, jolloin Vision Huntille luotiin uusi arkkitehtuuri, joka käsittää mobiilisovelluksen ja Cloud Vision API rajapinnan lisäksi, myös Firebase pohjaisen tietokannan.

# 2 STATE OF THE ART

## 2.1 Pilvipalveluiden tekoälyteknologiat

Erilaisten tietotekniikan toteutuksien siirtäminen pilvipalveluihin on kovaa vauhtia yleistymässä ja suurimmilla palveluntarjoajilla on monenlaisia tekoälypalveluita tarjolla heidän pilvi alustoissa. Opinnäytetyössä tutustutaan Google Cloudin -pilvipalvelu alustaan ja siitä löytyviin tekoälyrajapintoihin.

Google Cloudista löytyvät tekoälykomponentit on toteutettu helposti käyttöönotettavilla ohjelmointirajapinnoilla (API). Pilvestä löytyvät tekoälyrajapinnat ovat valmiiksi koulutettuja tekoälymalleja, joita käytetään valmiin ohjelmointirajapinnan avulla. Valmiiksi koulutettu tekoälymalli tarkoittaa, ettei tekoälymallia tarvitse erikseen kouluttaa tai ohjelmoida tekemään haluttua toimintoa. Toiminto voi esimerkiksi olla kuvasta erilaisten objektien tunnistus, jonka tekemiseen tekoälymalli on valmiiksi koulutettu. Valmiiksi koulutettuja tekoälymalleja tarjotaan näiden rajapintojen kautta, mutta lisäksi on myös mahdollista kouluttaa omien tarpeiden mukainen tekoälymalli, jos tarvitaan tietynlaista toimintaa, jota ei valmiissa rajapinnassa ole.

Opinnäytetyön prototyypissä käytetään valmiiksi koulutettua tekoälyrajapintaa, Google Cloud Vision API:a, koska se vastasi kaikkia prototyypin tarpeita ja sen toteuttaminen oli huomattavasti yksinkertaisempaa kuin uuden tekoälymallin luominen. Cloud Vision API on kuvien analysointia varten oleva tekoälyrajapinta, jolla saadaan kuvasta poimittua siitä löytyviä asioita ja tekstiä.

Ennen prototyypin kehittämistä suoritettiin vertailua eri pilvipalvelu alustojen välillä, josta lopulta päädyttiin käyttämään Google Cloud alustaa ja siitä löytyvää Cloud Vision API:a. Microsoftin Computer Vision API ja Amazonin Rekognition API täyttivät projektin prototyypin vaatimukset ja ovat ominaisuuksiltaan hyvin samanlaisia Googlen Cloud Vision API:n kanssa. Loppujen lopuksi päädyttiin käyttämään Googlen palveluita, koska niiden käyttö oli ennestään tuttua. Prototyypin olisi voinut toteuttaa millä tahansa näistä pilvipalvelu alustoista ja käytettävää tekoälyrajapintaa on mahdollista vaihtaa tarvittaessa, ilman että prototyypin koodia tarvitsisi muokata paljon.

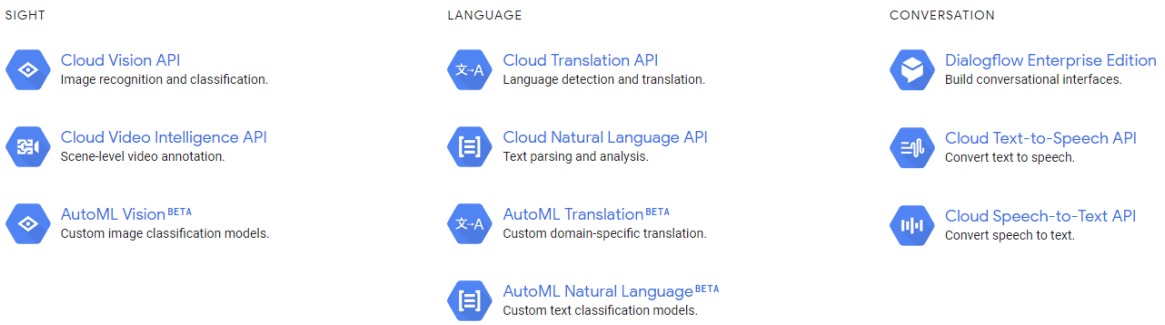
## 2.2 Google Cloudin tekoälyrajapinnat

Google Cloud on Googlen kehittämä pilvipalvelualusta. Tässä osiossa käydään läpi Google Cloudista löytyviä pilvilaskenta-, koneoppiminen- ja tekoälypalveluita. Pilvilaskenta

tarkoittaa IT-resurssien siirtämistä pilvialustalle, jolloin esimerkiksi datan varastointi, verkkoyhteydet ja datan laskenta tapahtuvat ulkoisilla pilvipalvelu tarjoajien laitteistolla pilvessä. Tällöin ei tarvita omia laitteistoja suorittamaan näitä toimenpiteitä, vaan kaikki voidaan halutessa tehdä ulkoisesti. (Google Cloud 2017a.)

Näin pystytään vähentämään laitteistosta aiheutuvia kuluja niin hankinnan kuin huollon suhteen, jonka lisäksi pilvessä tapahtuvan toiminnan ketteryys ja monipuolisuus auttavat keskittämään palvelun käyttäjän resurssit tehokkaammin. Laitteistosta aiheutuvia kuluja on myös helpompi seurata, kun hinnoittelu määräytyy käytön mukaan. Data on myös saatavilla tarvittaessa kaikkialla, eikä ohjelmistoja tarvitse erikseen asentaa, määrittää tai päivittää.

Koneoppiminen on alakategoria tekoälystä ja käytännössä tarkoittaa ohjelmiston opettamista suorittamaan jotain tiettyä tehtävää, ilman erillistä ohjelmointia tai sääntöjen luontia. Koneoppiminen perustuu luotuihin algoritmimalleihin, jotka on opetettu tunnistamaan yhtäläisyyksiä ja kuvioita tekstistä, kuvista tai puheesta. Pilvessä sijaitsevat tehokkaat ja lukuisat laskentaresurssit auttavat tekemään mahdollisimman tehokasta koneoppimista. (Google Cloud 2017b.) Koneoppimista on esimerkiksi kuvien lajittelu niiden kategorioiden mukaan, avainsanojen etsiminen pitkistä dokumenteista tai sähköposteista tai asiakkaalle henkilökohtaisten suositusten tekeminen. Googlen pilvipalvelussa on mahdollista käyttää valmiiksi koulutettuja koneoppimismalleja, tai kouluttaa tarvittaessa omaa tarvetta vastaava malli. Koneoppimispalveluita on mahdollista käyttää joko avoimen lähdekoodin kirjastojen kautta, tai käyttää pilvestä löytyviä palveluita tai rajapintoja.




KUVA 1. Google Cloudin tekoälyrajapinnat (Google Cloud 2018c)

Kuvassa esitetyt tekoälyrajapinnat tekevät seuraavaa (KUVA 1): Cloud Vision API on kuvan analysointia varten ja siitä löytyvän sisällön jakaminen erilaisiin kategorioihin. Lisää Cloud Vision API:sta kerrotaan kohdassa 2.3, jossa käydään siitä löytyviä ominaisuuksia läpi tarkemmin. Cloud Video Intelligence API on toiminnaltaan hyvin samanlainen Cloud Vision API:iin verrattuna ja sitä voidaan käyttää analysoimaan videoita ja kategorisoimaan niistä löytyvä sisältö eri tunnisteisiin, kuten vaikka ihminen, pyörä ja auto. Tämän lisäksi

voidaan tunnistaa, milloin videossa muuttuu tapahtumapaikka, onko siinä aikuissisältöä ja kääntää videon sisältö englannin kielelle. (Google Cloud 2018a.)

Cloud Translation API on eri kielien havaitsemiseen ja kääntämiseen tarkoitettu rajapinta. Sillä voidaan lukea ja kääntää annettu sana tai lause valitulle kielelle ja sen avulla voidaan tunnistaa tekstin alkuperäinen kieli siinä tapauksessa, jos sitä ei ennestään tiedetä. Rajapinnalle voidaan suoraan lähettää HTML dokumentti käännöstä varten, jos halutaan kääntää kerralla suurempi määrä tekstiä. Tällä hetkellä Translation API tukee yli sataa eri kieltä. (Google Cloud 2018d.)

Cloud Natural Language API käyttää koneoppimista hyväkseen, analysoidakseen tekstistä sen rakenteen ja tarkoituksen. Esimerkiksi sillä voidaan ymmärtää tekstistä mitä asiakas tarkoittaa keskustellessaan chat botin kanssa tai miten johonkin tuotteeseen suhtaudutaan. Rajapintaa voidaan yhdistää toisiin rajapintoihin kuten esimerkiksi Cloud Speech-to-Text API:n, jolloin voidaan muuttaa ensiksi asiakkaan puhe tekstiksi, joka analysoidaan Natural Language API:lla.

Cloud Text-to-Speech API ja Speech-to-Text API ovat nimensä mukaan rajapintoja, jolla käännetään puhetta tekstiksi tai toisinpäin. Molemmat rajapinnat tunnistavat automaattisesti käytetyn kielen ja pystyvät tuottamaan käännöstä reaaliajassa. Käännökseen voidaan myös suoraan tehdä asiattoman sisällön filtteröinti, ja puheentunnistusta voidaan suorittaa myös audiolle, jossa on paljon taustamelua. Puhetta voidaan tuottaa yli kolmellakymmenellä eri äänellä, jotta saadaan mahdollisimman luonnollisen kuuloinen käännös.

Dialogflow Enterprise Edition on keskustelukäyttöliittymien luontiin soveltuva palvelu, jota tarjotaan osana Google Cloudista löytyviä tekoälypalveluita. Sillä voidaan rakentaa koneoppimiseen perustuvia keskustelukäyttöliittymiä nettisivuille, mobiilisovelluksiin, eri viestintä alustoihin ja IoT laitteisiin. Dialogflow:ia voidaan yhdistää eri API:hin luomaan esimerkiksi mahdollisimman aidon tuntuinen asiakaspalvelu toiminto, yhdistäen Cloud Natural Language API:a asiakkaan tarpeen ymmärtämiseen, jolloin voidaan Dialogflown koneoppimista hyödyntäen saada mahdollisimman oikea vastaus tämän kysymykseen.

Viimeisenä osana ovat AutoML Vision, AutoML Translation ja AutoML Natural Language palvelut, jotka ovat omien koneoppimismallien koulutukseen tarkoitettuja palveluita niiden nimeä vastaaviin API:hin. Näiden avulla voidaan esimerkiksi saada Cloud Vision API tunnistamaan tiettyä ominaisuutta kuvasta ja antamaan niille oikeat tunnisteet. Esimerkiksi tätä kautta voidaan kouluttaa tekoälymalli, jolla voidaan tunnistaa oman yrityksen tuotteita ja logoja, jos Cloud Vision API ei ennestään pysty niitä tunnistamaan. Koneoppimismallien koulutus perustuu niille syötettävien tietojoukkojen läpikäymiseen, jolla koulutetaan malli tunnistamaan tiettyjä ominaisuuksia kuvasta. Tietojoukoilla mallille opetetaan, mikä kuva

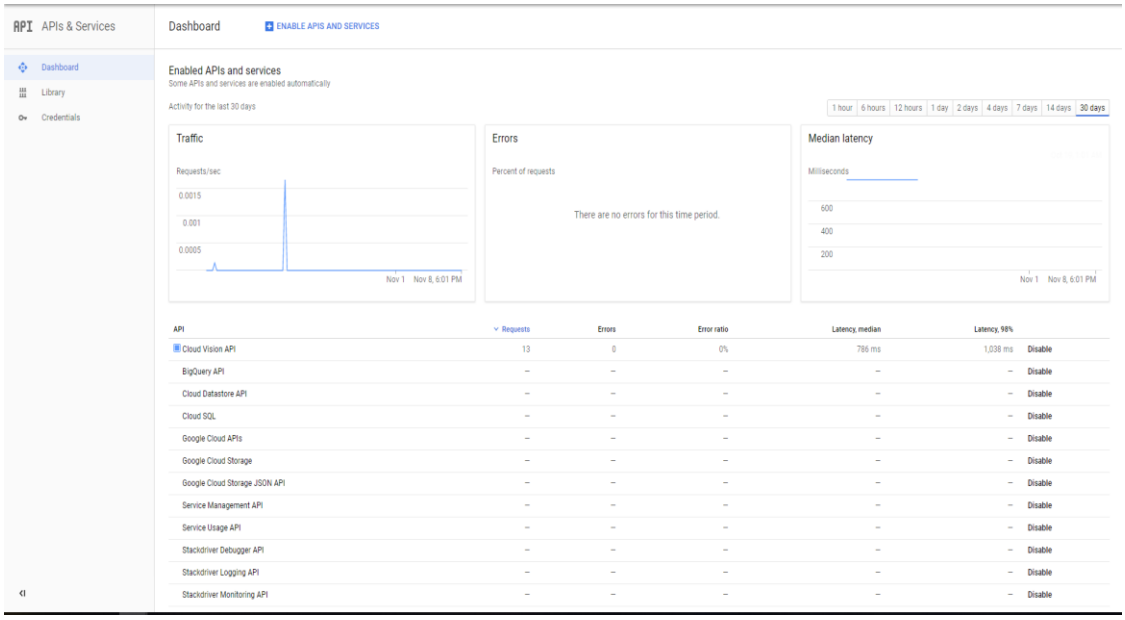
sisältää minkäkin halutun tunnisteen. Tämän avulla malli kehittyy lopulta ymmärtämään uusista kuvista, löytyykö jotain tiettyä koulutettua asiaa vai ei.

Kuvassa 1 on Google Cloudissa tällä hetkellä löytyvät tekoälyrajapinnat. Niitä voidaan käyttää Google JSON/REST API:n avulla, jolla voidaan käyttää pilvestä löytyviä rajapintoja helppokäyttöisillä HTTP kutsuilla. API:ja voidaan myös kutsua asiakaskirjastojen avulla, jolloin niitä voidaan käyttää halutulla ohjelmointikielellä. API:n rajapinnan käyttöä varten riittää yhden asiakaskirjaston lataaminen, sillä se tarjoaa rajapinnan kaikkien API:n käyttöä varten.

Mobiilisovelluksia varten on mahdollista käyttää Firebase mobile alustaa, jolla sovellukseen saadaan SDK:lla asiakasohjelman toiminnallisuus. Tämän avulla voidaan käyttää Cloud API:a suoraan mobiilisovelluksesta. On myös mahdollista ohjelmoida oma toiminnallisuus suoraan sovellukseen, jolla käytetään API palvelun alemman tason palveluita. Joka tapauksessa kaikki pilvestä löytyvät API:t käyttävät tavallista JSON/REST rajapintaa, jota voidaan käyttää periaatteessa millä tahansa HTTP asiakaskirjastolla.

Jotkin API:t voivat myös käyttää gRPC API rajapintaa. gRPC on Googlen kehittämä avoimen lähdekoodin alusta, joka on alustasta ja kielestä riippumaton RPC (remote procedure call) järjestelmä. Sen sijaan, että käytetään JSON over HTTP:tä kommunikoimaan REST rajapinnan kanssa, gRPC:tä käyttävät API asiakasohjelmat voivat myös käyttää protokollabuffereita ja gRPC over HTTP2 kommunikoimaan RPC rajapinnan kanssa. Jos API tukee gRPC:tä, voidaan luoda millä tahansa gRPC:tä tukevalla ohjelmointikielellä asiakaskirjastoja. gRPC:n käyttö on sovellusten suorituskyvyn kannalta tehokkaampaa, vaikkakin se on yhä melko uusi ja siksi sen yhteensopivuus ja dokumentointi ei vielä olekaan kovin kattava.

Jokaista API:a varten on saatavilla kattava dokumentaatio, jolla pääsee alkuun API:n käytössä eri alustoille. Google Cloudin konsolin kautta on mahdollista seurata omien projektien API:en käyttöä, näkemällä niiden käytön määrän, virheiden määrän ja viiveen. Suurin osa API:sta on maksullisia perustuen siihen, paljon niitä käytetään. Google Cloud konsolin kautta on mahdollista seurata paljon eri API:ja on käytetty ja kutsuttu eri projekteissa ja paljon siitä koituu kuluja. Vaikka eri API:lla on erilaiset hinnastot, perustuu niiden hinnoittelu käytön määrään, eli maksat vain siitä mitä käytät. Monien API:en käyttö on maksutonta tiettyyn pisteeseen asti, joten jos projekti ei käytä tiettyä API:a paljoa, ei siitä välttämättä koidu kuluja ollenkaan. Kuvassa 2 näkyy Google Cloud konsolin näkymä, jossa nähdään mitä API:a on käytetty, sen liikenne, virheet ja viiveen mediaani (KUVA 2).




KUVA 2. Google Cloud Konsoli

Jonkin tietyn API:n voi ottaa käyttöön omaa projektia varten käyttämällä Google Cloud Platform konsolia (GCP Console), joka on hallintatyökalu, jota käytetään verkkosivun kautta. GCP:llä valitaan projekti, ja sille voidaan antaa käyttölupa joihinkin API:hin, tai joidenkin käyttö voidaan halutessa poistaa. Jos API:n käyttö on maksullista, on projektille sallittava laskutusasetukset ennen kuin maksullista API:a voidaan käyttää. Jos maksullisen API:n ilmainen käyttöraja ylittyy, ruvetaan ylimenevästä käytöstä laskuttamaan. Saman GCP konsolin kautta voidaan helposti seurata omien projektien API:en käyttöä ja nähdä onko niistä koitunut kuluja. Myös API:en toimintaa voi tutkia konsolin kautta: konsolista voi esimerkiksi nähdä, mikä palvelu API:a käyttää tai milloin toiminta tapahtuu.

Halutessa voidaan käyttää Stackdriver palvelua analysoimaan API:en käyttöä tarkemmin. Stackdriver on Googlen monitorointityökalu, jolla voidaan analysoida erilaisia lokitiedostoja, datan käyttöä ja tapahtumia omista projekteista. Stackdriveria voi myös käyttää muiden pilvipalveluiden kanssa, eli Google Cloud Platformin käyttö ei ole pakollista.

## 2.3 Cloud Vision API

Cloud Vision API on Googlen kehittämä tekoälyrajapinta kuvien sisällön analysointia varten. Cloud Vision API on REST rajapintaa käyttävä API, joka perustuu valmiiksi koulutettuihin koneoppimismalleihin, joiden avulla kuvien sisältö voidaan luokitella erilaisiin kategorioihin. Sillä voidaan myös tunnistaa kasvoja ja ilmeitä, sekä lukea ja tunnistaa tekstiä. (Google Cloud 2018b.)

Vision API:n voidaan ottaa yhteys monella tapaa, alla on kuvattu yksi tapa tehdä se. Kuvasta haetaan erilaisia tunnisteita käyttämällä Label Detection ominaisuutta, jolla saadaan lista kuvasta tunnistetuista eri objekteista. Näitä voi olla mm. puu, patsas, järvi ja auto.

POST pyyntö luodaan Vision API:iin ja tähän identifioidaan käyttämällä prototyypille ominaista API avainta, joka liitetään ’annotate’ pyynnön perään. Tähän liitetään mukaan JSON objekti, joka näyttää samanlaiselta, kuin kuva 3:

```
{
  "requests":[
    {
      "image":{
        "content":"/9j/7QBEUGhvdG9...image contents...eYxxxzj/Coa6Bax//Z"
      },
      "features":[
        {
          "type":"LABEL_DETECTION",
          "maxResults":1
        }
      ]
    }
  ]
}
```

KUVA 3. Esimerkki POST pyyntö JSON objektista

Kuvassa 3 image tarkoittaa kuvaa konvertoituna base64 enkoodatuksi kuva merkkijonoksi (KUVA 3). Featuret ovat eri analyyseja tai niiden parametreja, mitä kuvalla halutaan suorittaa. Tässä tapauksessa LABEL_DETECTION on erilaisten kuvasta löytyvien asioiden tunnistamista ja jakamista eri kategorioihin.

Vastauksena tähän Vision API lähettää AnnotateImageResponse JSON vastauksen, joka sisältää kaikki tunnisteet mitä kuvasta löytyi. Tällainen vastaus näyttää samanlaiselta, kuin KUVA 4. Mid (machine-generated identifier) tarkoittaa kuvassa olevan tunnisteen merkintää Googlen Knowledge Graph:ssa, eli sillä voidaan tunnistaa tunnisteen asia kielestä riippumatta. Description on tunnisteen kuvaus, eli lyhyesti mikä asia on. Score on arvio, millä todennäköisyydellä kuvasta löytyy kyseinen tunniste. Arvio sijoittuu välille 0-1, jossa nolla on epätodennäköistä ja 1 on todennäköistä.

```
{
  "responses": [
    {
      "labelAnnotations": [
        {
          "mid": "/m/0bt9lr",
          "description": "dog",
          "score": 0.97346616
        },
        {
          "mid": "/m/09686",
          "description": "vertebrate",
          "score": 0.85700572
        },
        {
          "mid": "/m/01pm38",
          "description": "clumber spaniel",
          "score": 0.84881884
        },
        {
          "mid": "/m/04rky",
          "description": "mammal",
          "score": 0.847575
        },
        {
          "mid": "/m/02wbgd",
          "description": "english cocker spaniel",
          "score": 0.75829375
        }
      ]
    }
  ]
}
```


KUVA 4. Esimerkki JSON response:sta

Label detection:in lisäksi Vision API:sta löytyy monta muuta erilaista toimintoa, jolla kuvasta voidaan saada tietoa. Seuraavaksi on esitelty muutama tällainen ominaisuus ja selitetty lyhyesti näiden ominaisuuksien toiminta.

Optical character recognition tunnistaa ja poimii koneella luotua tekstiä kuvasta. Tällä hetkellä optical character recognition tukee 56 eri kieltä ja automaattisesti tunnistaa käytetyn kielen. Tämän kanssa hyvin samanlainen on Handwriting recognition, jolla voidaan tunnistaa käsin kirjoitettua tekstiä. Handwriting recognition on vielä beta vaiheessa.

Logo Detection tunnistaa erilaisia tunnettuja tuotteiden logoja kuvasta. Sillä voidaan paikantaa logon paikka kuvassa, jonka lisäksi tunnistetaan logolle nimi ja entity ID, jota voidaan käyttää referenssinä Googlen Knowledge Graph tietopankissa.

Beta vaiheessa oleva Object localizer pystyy identifioimaan objektin paikan kuvassa ja kyseisen objektityypin määrän. Esimerkkinä on kuva, jossa pyörän renkaat on tunnistettu pyörän sisälle. Objektille annetaan bounds tiedot, eli luodaan muoto sen ympärille antaen

alun ja lopun XY koordinaatit. Kuvassa 5 nähdään, miten löydetyt objektit ympäröidään laatikoilla, jotka kertovat objektien koordinaatit kuvassa (KUVA 5).


KUVA 5. Object localizer:n rajaamat alueet havainnollistettuna (Google Cloud 2018e)

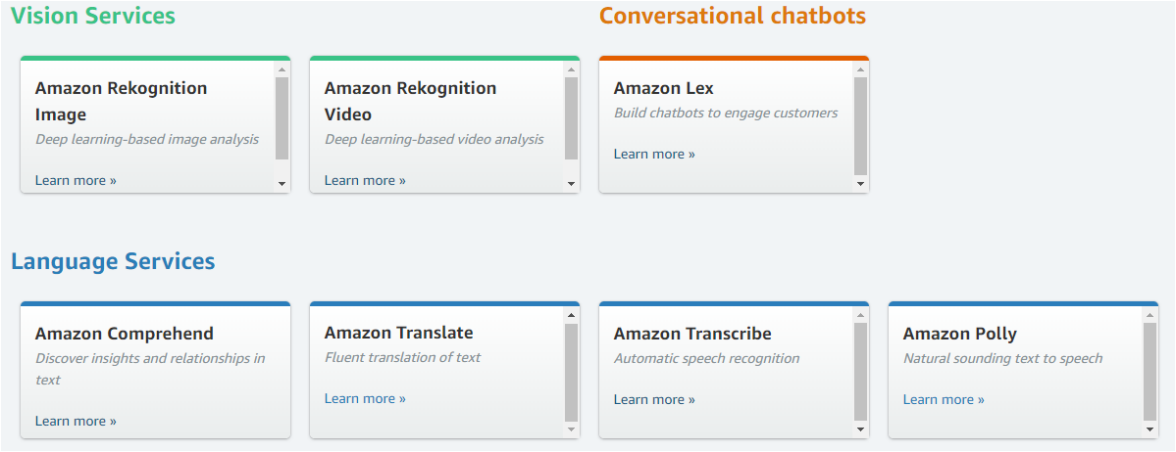
Landmark detection pystyy tunnistamaan erilaisia luonnollisia ja ihmisen luomia maamerkkejä kuvasta. Sillä voidaan liittää maamerkille koordinaatit latitudi ja longitudi muodossa maamerkin paikannusta varten. Joillekin maamerkeille on mahdollista saada Google Maps kuva. Jokaisella maamerkillä on nimi, analyysin tarkkuus ja alue johon maamerkki määritetään kuvassa.

Face detection tunnistaa kuvasta ihmisen kasvoja ja niissä olevia erilaisia tunnetiloja. Sillä voidaan tunnistaa, jos ihmisen päässä on jokin esine kuten hattu tai kypärä. Face detection tunnistaa kasvoilta löytyviä asioita kuten silmiä tai kulmakarvoja, ja pystyy paikantamaan niiden paikan kuvassa. Erilaisten tunnetilojen analyysille suoritetaan todennäköisyys arviointi, jolla voidaan arvioida, miten todennäköisesti jokin tunnetila esiintyy kasvoilla. Tällä hetkellä sillä ei voi tunnistaa ihmistä kasvojen perusteella.

## 2.4 AWS ja Microsoft Azure

Myös Amazon ja Microsoft tarjoavat tekoälyrajapintoja ja palveluita pilvipalvelu alustoillaan. Kun opinnäytetyötä varten valittiin pilvipalvelu alustaa, tehtiin tätä ennen vertailua näiden kolmen alustan välillä. Vertailun tuloksena päädyttiin lopulta valitsemaan Google Cloud. Näiden kolmen alustan tarjoamissa tekoälyrajapinnoissa on paljon yhtäläisyyksiä, joten suurimmiksi valintakriteereiksi opinnäytetyön alustan valinnassa oli Googlen Cloudin aikaisempi käyttökokemus.

### 2.4.1 Amazon Web Services




KUVA 6. AWS:n tekoälyrajapinnat (Amazon Web Services 2018)

Kuvassa 6 on esitetty Amazonin tarjoamat tekoälyrajapinnat (KUVA 6.) Amazon Rekognition Image ja Amazon Rekognition Video ovat kuvan ja videon analyysia varten käytettävät tekoälyrajapinnat. Amazon Rekognition on toiminnaltaan hyvin samanlainen Googlen Cloud Vision API:n kanssa. Se tunnistaa kuvasta objekteja, paikan ja aktiviteetin. Siinä on myös kasvontunnistus ja kasvojen analysointi toiminnot. Videoita analysoidessa voidaan seurata objektien liikerataa. Rekognition pystyy myös tunnistamaan sisällön aikuissisällöksi tai epäasialliseksi ja antamalla analysoidulle asialle sen mukaisen tunnisteen. Tämän lisäksi Rekognition pystyy tunnistamaan tekstin kuvista, konekirjoitetun ja käsinkirjoitetun. (Amazon Web Service 2018.)

Amazon Lex on palvelu, jolla voidaan rakentaa keskustelu-käyttöliittymiä sovelluksiin käyttämällä joko ääntä tai tekstiä. Siinä on syväoppimis toiminnallisuus, jolla voidaan tunnistaa puhetta, muuttaa puhetta tekstiksi ja tunnistaa puheesta sävy. Tämän avulla voidaan rakentaa käyttöliittymiä, jossa käyttäjän kanssa voidaan keskustella esimerkiksi chatbotin avulla. Esimerkiksi Amazonin Alexa pohjautuu samaan tekoälyteknologiaan kuin Amazon Lex.

Kielipohjaisiin tekoälypalveluihin kuuluvat seuraavat: Amazon Comprehend tekoälyn avulla saadaan tulkittua tekstistä tarkoituksellisuutta ja miten tekstissä esiintyvät asiat liittyvät toisiinsa. Amazon translate tekoälyllä voidaan kääntää tekstiä toiselle kielelle. Amazon Transcribe tunnistaa puheen ja pystyy muuntamaan sen tekstiksi. Viimeisenä Amazon Polly tekoälypalvelu muuttaa tekstiä luonnollisen kuuloiseksi puheeksi.

### 2.4.2 Microsoft Azure

Microsoft Azure:n tekoälyrajapinnat voidaan jakaa viiteen kategoriaan: Vision, Speech, Language, Knowledge ja Search (Microsoft Azure 2018). Vision palveluihin kuuluu Computer Vision joka on Googlen Cloud Vision:ia vastaava palvelu kuvien analysointiin. Siinä on kuvan luokittelu, aktiviteetin analysointi, maamerkkien tunnistus ja optisten sekä käsin kirjoitettujen merkkien tunnistus. Muita Vision pohjaisia palveluita ovat

- Video Indexer videoiden analysointia varten
- Face kasvojen analysointia varten
- Content Moderator kuviin ja videoihin rajoitusten luomista varten
- Custom Vision kuvien analysointimallien tekoa varten.

Speech kategoriaan kuuluvat Speech to Text ja Text to Speech, jotka nimensä mukaisesti muuntavat tekstiä tai puhetta vastakkaiseen formaattiin. Speaker Recognition taas pystyy tunnistamaan äänen perusteella henkilön ja sitä voi esimerkiksi käyttää ääni verifikaatiota varten. Viimeisenä on Speech Translator API, jolla voidaan muuntaa puhetta toiselle kielelle reaaliajassa.

Language kategoriaan kuuluvat Text Analytics, joka on tekstin analysointia varten. Translator Text joka on kielen tunnistamista ja kääntämistä varten. Bing Spell Check joka on oikeinkirjoituksen tarkistamista varten. Language Understanding, joka on kielessä kontekstin ymmärtämistä varten. Content Moderator, joka on sama kuin Vision kategoriassa oleva.

Knowledge kategoriaan kuuluu tällä hetkellä vain QnA Maker, jolla voidaan luoda sisällön perusteella kysymyksiin vastaava botti. Tätä voidaan käyttää sovelluksessa esimerkiksi luomaan käyttäjien kysymyksiin vastaava botti, joka analysoi käyttäjän antaman tekstin ja hakee sitä parhaiten vastaavan osuman vastaukseksi. Tätä voidaan helposti yhdistää toisiin API:hin, esimerkiksi puheen ymmärtämisen kanssa voidaan luoda puheen mukaan vastauksia hakeva botti.

Search kategoriaan kuuluvat tiedon etsintään perustuvat palvelut. Näitä ovat: Bing Web Search, Custom Search, Video Search, Image Search, Local Business Search, Bing Visual Search, Entity Search, News Search ja Autosuggest. Kaikki pohjautuvat Bing hakukoneeseen ja niitä voidaan käyttää JSON kutsulla, jolle annetaan halutun etsinnän parametrit.

# 3 ANDROID

## 3.1 Android-alusta

Android on Googlen kehittämä mobiilialusta, joka on tällä hetkellä eniten käytetyin älypuhelimien käyttöjärjestelmä. Alusta käsittää pääasiassa kosketusnäytölliset älylaitteet, kuten älypuhelimet ja tabletit. Myös monet muut laitteet käyttävät Android-pohjaista järjestelmää, esimerkiksi Android TV televisiot. Android pohjautuu Linux Kernel käyttöjärjestelmään (Android Developer 2018a). Kuvassa 7 on visualisoituna tärkeimmät Android-alustan komponentit (KUVA 7). Linux kernel tarjoaa toiminnot ja ajurit erilaisiin komponentteihin, kuten säikeisiin ja alemman tason muistinhallintaan.

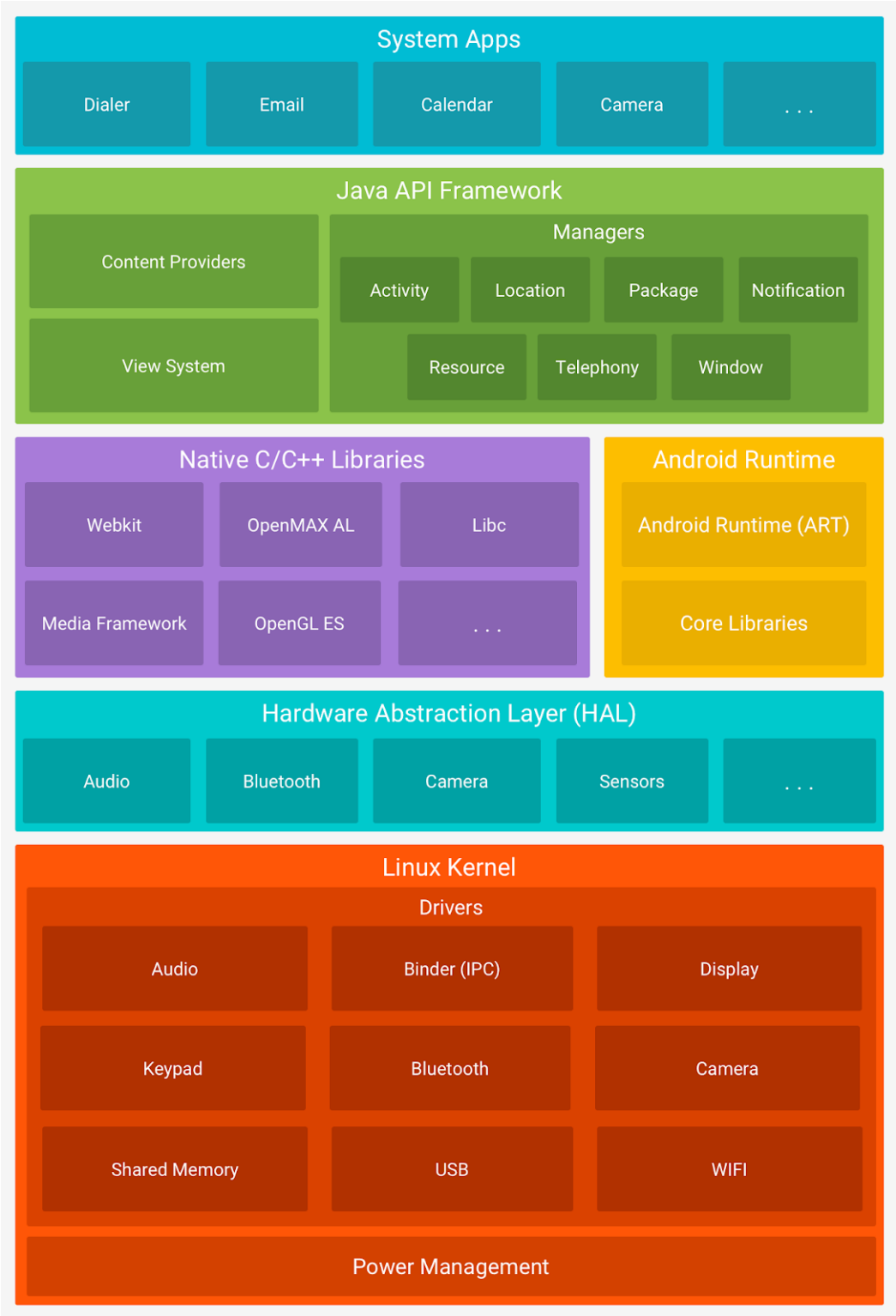
Hardware Abstraction Layer (HAL) tarjoaa standardit käyttöliittymät laitteiston ominaisuuksien käyttämiseen ylemmille tasoille. HAL koostuu monista kirjastomoduuleista, joista jokainen implementoi rajapintaa jollekin tietylle laitteiston komponentille. Kun jokin framework API luo kutsun käyttääkseen jotain tiettyä laitteen osaa, Android-järjestelmä lataa kirjastomoduulin sille laitteisto komponentille.

Android Runtime on uudemmissa Android-versioissa oleva prosessi, jota jokainen sovellus suorittaa omana instanssinaan. Android Runtime on tehty suoritettavaksi useilla virtuaalikoneilla laitteissa, joissa on pieni määrä muistia. Tämä onnistuu suorittamalla DEX tiedostoja, jotka ovat Androidille suunniteltuja ja optimoituja tavukoodi formaatin tiedostoja. ART siis kääntää sovellusten bytecoden natiiviksi ohjeiksi, jonka laitteen ajonaikainen ympäristö myöhemmin suorittaa. Android sisältää myös joukon ydin kirjastoja, jotka mahdollistavat suurimman osan Java ohjelmointikielen käytettävyydestä jota Java API framework käyttää.

Android-alusta tarjoaa myös Java framework API:t tuomaan toimivuutta sovelluksille, jotka vaativat C tai C++ kirjastoja toimiakseen. Monet Androidin ydinjärjestelmän komponenteista ja palveluista vaativat natiivit kirjastot C:tä tai C++:lta.

Java API Framework sisältää kaikki Android-käyttöjärjestelmän ominaisuudet Java kielellä kirjoitettujen API:en kautta. Nämä API:t toimivat sovellusten luomisessa helpottamaan ja yksinkertaistamaan ydinosien, modulaaristen järjestelmä komponenttien ja palveluiden uudelleenkäyttöä. Tähän kuuluu View järjestelmän käyttö sovelluksen käyttöliittymän luomista varten, resource manager resurssien hallintaan, notification manager tiedotusten hallintaan, activity manager sovellusten elinkaaren hallintaan ja content providers auttamaan sovelluksia jakamaan ja käyttämään dataa muiden kautta.

Androidissa on myös valmiina sovelluksia tiettyjä toimintoja varten. Näitä ovat esimerkiksi sähköposti, tekstiviesti, kalenteri, internet selain ja kontakti sovellukset. Nämä sovellukset antavat jakavat myös toiminnallisuuksia toisille sovelluksille, esimerkiksi kutsua kameraso-vellusta toisen sovelluksen sisältä ottamaan kuvaa kameralla sovellusta varten.




KUVA 7. Android-alustan tärkeimmät komponentit (Android developer 2018a)

## 3.2 Android-ohjelmointi

Android-alustalle sovelluksia voi ohjelmoida monella eri kielellä. Tällä hetkellä Java, Kotlin ja C++ toimivat natiivien Android-sovellusten ohjelmointikielinä. Opinnäytetyön tarkastelun aiheena oleva projekti tehtiin Javalla.

Android-sovelluksen juuressa on manifest tiedosto, joka kertoo järjestelmälle mitä komponentteja sovellus sisältää. Tähän tiedostoon määritellään kaikki käyttöoikeudet, mitä sovellus käyttää. Tällaisia ovat esimerkiksi puhelimen internet yhteyden käyttö, ja mitä ohjelmisto tai laitteistoja sovellus käyttää, kuten puhelimen kamera. Manifest tiedostoon määritellään sovelluksen API (Application Programming Interface) taso, joka määrittää, mitä ohjelmointirajapintoja sovellus käyttää. Käytetyt API kirjastot linkitetään myös manifest tiedostoon. Kuvassa 8 on esitetty Android-sovelluksen manifest tiedosto (KUVA 8).

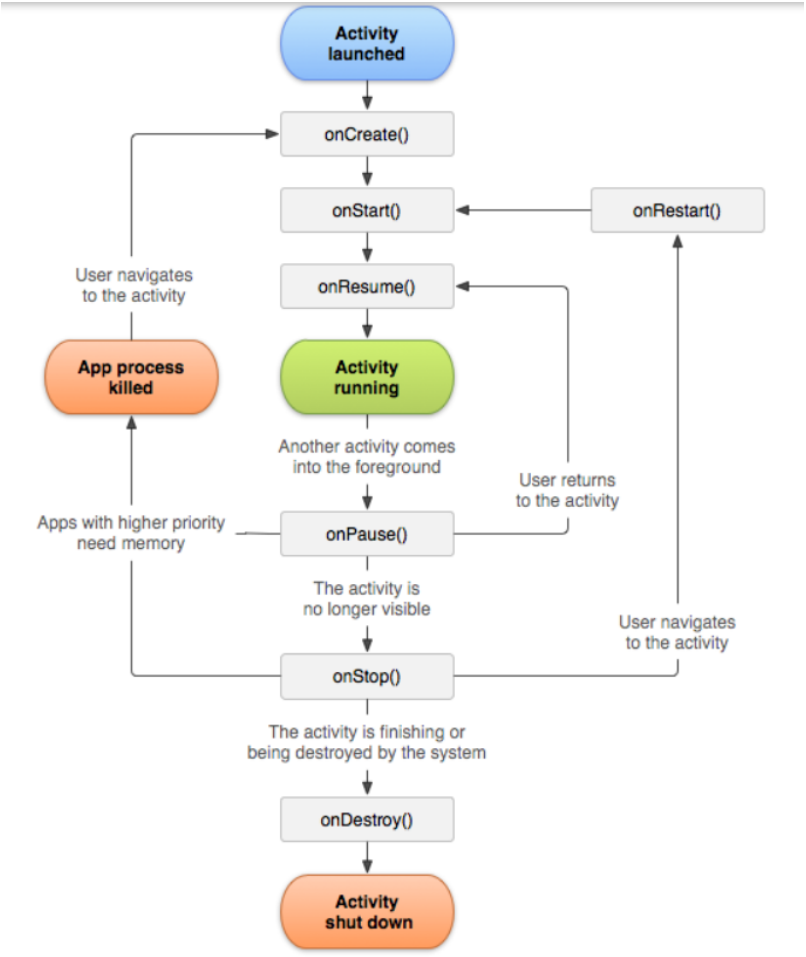
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="teravainen.imagegameapp">
    <!-- camera, storage and gps permissions -->
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-feature android:name="android.hardware.location.gps" />
    <uses-permission android:name="android.hardware.camera2.full" />

    <application
        android:name=".GlobalVariables"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Vision Hunt"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity" />
        <activity android:name=".TakePictureActivity" />
        <activity android:name=".PicTaken" />
        <activity android:name=".bottomNavigation" />
        <activity
            android:name=".SplashActivity"
            android:label="Vision Hunt"
            android:theme="@style/SplashTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".CameraDemo"></activity>
        <provider
            android:authorities="teravainen.imagegameapp.fileProvider"
            android:name="android.support.v4.content.FileProvider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/file_paths"></meta-data>
        </provider>
    </application>
</manifest>
```

KUVA 8. Android-sovelluksen manifest tiedosto

Natiivit Android-sovellukset omaavat neljä erilaista komponenttityyppiä, jotka toimivat käyttäjän ja järjestelmän tapoina käyttää sovellusta. Aktiviteetit näkyvät sivuina käyttäjälle ja ovat tapa esittää sisältöä käyttäjälle. Käyttöliittymä rakennetaan aktiviteettien ympärille, jossa esimerkiksi voidaan tehdä eri toiminnot erillisille aktiviteeteille, joiden välillä käyttäjä navigoi. Aktiviteetit ovat riippumattomia toisistaan, joten kaksi erillistä aktiviteettia voivat johtaa kolmanteen aktiviteettiin ilman, että olisivat tietoisia toisistaan. Yksinkertaisesti aktiviteetit ovat sivuja, jossa esitetään käyttäjälle sisältöä käyttöliittymän muodossa. Aktiviteeteilla on tietynlainen elinkaari aktiviteettipinossa. Kun aktiviteetti käynnistetään, siirtyy se pinossa ylimmäiseksi ja siitä tulee käynnissä oleva aktiviteetti. Edellinen aktiviteetti jää pinossa sen alle, ja ei tule esille ennen kuin uusi aktiviteetti suljetaan.




KUVA 9. Android-aktiviteetin elinkaari (Android developer 2018b)

Aktiviteeteilla on myös monta muuta eri tilaa. KUVA 9 on kuvattu aktiviteetin eri tilat. Kun järjestelmä luo aktiviteetin, kutsutaan onCreate() metodia. Kun aktiviteetti ollaan tuhoamassa, kutsutaan onDestroy() metodia. OnStart() kutsulla aktiviteetti tulee näkyväksi käyttäjälle, jonka suoritettua se siirtyy Resumed-tilaan. Started-tilan aikana aktiviteetti on olemassa, vaikkei käyttäjä sitä ruudulla näkisikään tai pysty tekemään sen kanssa mitään.

OnResume() kutsun jälkeen aktiviteetti on näkyvissä käyttäjälle aina kutsuun onPause() asti. Aktiviteetti voi mennä edestakaisin onResume() ja onPause() välillä, esimerkiksi kun puhelimen ruutu sammutetaan ja laitetaan takaisin päälle. OnStop() kutsu suoritetaan, kun aktiviteetin päälle aukeaa uusi aktiviteetti tai kun aktiviteetti on suorittanut tehtävänsä ja sitä ollaan pian tuhoamassa. (Android Developer 2018b.)

Palvelut ovat komponentteja, joita suoritetaan taustalla, jotta sovellus voi suorittaa jonkinlaisen tehtävän ilman, että tämän tarvitsee olla aktiivisena sovelluksena. Palvelut eivät tarjoa käyttöliittymää. Käyttäjä voi tehdä jotain muuta eri sovelluksessa tai aktiviteetissa, samalla kun sovellus tekee tehtäväänsä taustalla. Palvelu voi pyytää järjestelmää pitämään palvelun elossa siihen asti, kunnes se on saanut tehtävän tehtyä. Usein käyttäjä voi olla tietämätön, että jokin palvelu suoritetaan taustalla, jos siitä ei ole mitään audio tai visuaalista vihjettä. Tarvittaessa järjestelmä voi kuitenkin sulkea taustalla olevan palvelun, jos järjestelmä tarvitsee muistia johonkin toiseen tehtävään.

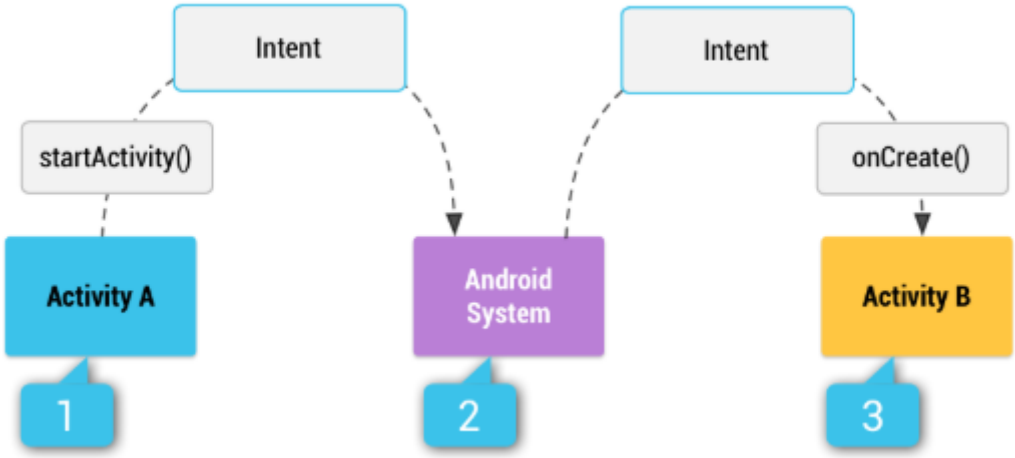
Viestien vastaanottaja on komponentti, jolla järjestelmä voi viestittää sovellusta tapahtumista, jotka tapahtuvat sovelluksen ulkopuolella, esimerkiksi silloin kun sovellus ei ole päällä. Tällä voidaan toteuttaa hälytys sovellukseen tietylle ajalle ilman, että sovelluksen tarvitsee olla päällä koko tätä aikaa. Järjestelmä voi myös viestittää sovellukselle muutoksista, kuten jos näyttö sammutetaan, osaa sovellus reagoida tähän.

Sisällöntuottajat hallitsevat yhteistä sovellus tietojoukkoa, johon sovellukset voivat päästä käsiksi. Sen kautta muut sovellukset voivat hakea ja muokata dataa, jos sisällön tuottaja sallii sen. Järjestelmälle sisällön tuottajat ovat tapa sovellukselle julkaista dataa, jota voidaan jakaa toisille sovelluksille.

Näiden neljän komponentin avulla Androidille pystyy luomaan sovelluksia, jotka käyttävät hyväksi järjestelmästä valmiiksi löytyviä osia. Esimerkiksi sovellukseen ei tarvitse ohjelmoida omaa kamera funktiota, vaan voidaan käyttää Androidista löytyvää kamerasovellusta tekemään tämä funktio, jonka sovellus avaa omaan käyttöön. Android-alustalla on paljon valmiita toimintoja, joita voidaan käyttää hyväksi omissa sovelluksissa. Näitä hyödyntämällä jokaista toimintoa ei tarvitse luoda alusta asti, vaan voidaan hyödyntää olemassa olevaa toimintoa.

Sovellukset usein käyttävät koodin ja edellä mainittujen asioiden lisäksi myös paljon muita resursseja, kuten kuvia, audio tiedostoja, ja erilaisia visuaalisiin asioihin liittyviä tiedostoja. XML tiedostoihin asetetaan erilaisia tyyleihin liittyviä asioita, kuten animaatioita, värejä, tyylejä, menuja ja layouteja. Näillä voidaan myös määrittää laitekohtaisia vaihtoehtoisia resursseja, kuten kieliä ja eri näytön koolle suunnattuja resursseja.

Intent-objektit ovat viestintä objekteja, joilla voidaan pyytää toimintoja eri sovellus komponenteilta. Kolme yleisintä käyttöä intent-objekteille ovat: Aktiviteetin aloittaminen, tämä tehdään esimerkiksi antamalla intent startActivity() kutsulle. Jos halutaan saada jokin tulos aktiviteetilta, voidaan käyttää startActivityForResult() kutsua. Toinen käyttötapa on palvelun aloittaminen. Tämä voidaan tehdä käyttämällä palvelut luokan metodia. StartService() metodille annetaan intent-objekti, jolla kuvataan haluttu palvelu ja siihen liittyvä data. Kolmas käyttö on viestin lähettäminen antamalla intent-objektille sendBroadcast() tai sendOrderedBroadcast() metodi. Näin voidaan lähettää viesti esimerkiksi muille sovelluksille. (Android Developer 2018c)




KUVA 10. Aktiviteetin aloittaminen intent-objektilla (Android developer 2018c)

Kuvassa 10 kuvataan miten Activity A luo intent-objektin ja lähettää sen startActivity() metodille (KUVA 10). Android-järjestelmä etsii kaikista sovelluksista intent-objektia vastaavan intent filtterin. Kun sellainen löytyy, järjestelmä aloittaa sitä vastaavan aktiviteetin, kutsumalla sen onCreate() metodia ja lähettämällä sille intent-objektin.

## 3.3 Android-puhelimen työkalut

Android-alustan puhelimet sisältävät monia erilaisia työkaluja, joita voi käyttää hyväkseen ohjelmoidessa sovelluksia Android-alustalle. Kameraa ei tarvitse ohjelmoida sovellukseen tyhjästä, vaan voidaan käyttää puhelimesta valmiiksi löytyvää kamerasovellusta itse tehdyn sovelluksen sisällä. Monet Googlen tekemät palvelut tarjoavat ohjelmointirajapinnan, jota voidaan hyödyntää sovellusten toteuttamisissa. Esimerkiksi Google Maps integrointia varten on tarjolla Google Maps SDK, joka lisätään haluttuun ohjelmointialustaan ja Google Maps API avain, jonka avulla tehdään kutsuja Google Maps API:n. Google tarjoaa Google

Mapsia varten kattavan dokumentaation, jolla ohjelmistokehittäjät voivat aloittaa kehityksen alustaa varten.

Nykyiset Android-puhelimet sisältävät paljon erilaisia sensoreita, joita voidaan käyttää hyväksi sovellusten toteuttamiseen. Taulukossa 1 on esitetty eri Android-versioiden sensorit (TAULUKKO 1). Sensoreilla voidaan muun muassa mitata liikettä, puhelimen asentoa, paikannusta, kiihtyvyyttä, valoisuutta, lämpötilaa, painetta ja monia muita asioita. (Android Developer 2018d) Puhelimen SDK version ja laitteiston mukaan joitain sensoreita ei välttämättä ole mahdollista käyttää. Sensoreiden dataa voi käyttää pyytämällä lupaa siihen sovelluksen manifest tiedostossa ja sovelluksessa niitä käytetään sensori palvelun kautta luomalla SensorManager luokka.

TAULUKKO 1. Eri Android-versioiden sensorit

| Sensor | Android 4.0 (API Level 14) | Android 2.3 (API Level 9) | Android 2.2 (API Level 8) | Android 1.5 (API Level 3) |
|---|---|---|---|---|
| TYPE_ACCELEROMETER | Yes | Yes | Yes | Yes |
| TYPE_AMBIENT_TEMPERATURE | Yes | n/a | n/a | n/a |
| TYPE_GRAVITY | Yes | Yes | n/a | n/a |
| TYPE_GYROSCOPE | Yes | Yes | n/a[1] | n/a[1] |
| TYPE_LIGHT | Yes | Yes | Yes | Yes |
| TYPE_LINEAR_ACCELERATION | Yes | Yes | n/a | n/a |
| TYPE_MAGNETIC_FIELD | Yes | Yes | Yes | Yes |
| TYPE_ORIENTATION | Yes[2] | Yes[2] | Yes[2] | Yes |
| TYPE_PRESSURE | Yes | Yes | n/a[1] | n/a[1] |
| TYPE_PROXIMITY | Yes | Yes | Yes | Yes |
| TYPE_RELATIVE_HUMIDITY | Yes | n/a | n/a | n/a |
| TYPE_ROTATION_VECTOR | Yes | Yes | n/a | n/a |
| TYPE_TEMPERATURE | Yes[2] | Yes | Yes | Yes |

Android-sovelluksissa pystyy dataa tallentamaan monella eri tavalla sovelluksen vaatimusten mukaisesti. Sisäinen tiedostomuisti on sovelluksen yksityisiä tiedostoja varten, joihin ei ulkopuolisilla ole tarkoitus päästä käsiksi. Ulkoinen tiedostomuisti on julkisia tiedostoja varten. Sharedpreferences luokkaan voidaan tallettaa yksinkertaista dataa kuten muuttujia, joka tapahtuu avain-arvo pareille. Datan voi noutaa avain-arvolla ja avain-arvo vastaa muuttujalle asetettua arvoa. Sovelluksissa on myös mahdollista tallentaa dataa sisäisiin tietokantoihin, kuten esimerkiksi prototyypissä käytettävään Room-tietokantaan.

# 4 PROTOTYYPPI

## 4.1 Prototyypin tavoitteet

Opinnäytetyön tavoitteena oli luoda Android-alustalle sovellusprototyyppi, joka käyttää hyväkseen pilvipalvelusta löytyvää tekoälyrajapintaa. Sovellus ideoitiin peliksi, joka käyttäisi Google Cloudista löytyvää kuvantunnistus tekoälyrajapintaa, Cloud Vision API:a tunnistaakseen kuvasta löytyviä asioita. Pelissä ideana on, että käyttäjä lähtee etsimään tehtävissä pyydettyjä asioita, kuvaa ne puhelimensa kameralla ja tekoälyrajapinta tunnistaa, löytyykö kuvasta pyydettyä asiaa vai ei. Prototyypille annettiin nimi Vision Hunt, joka perustuu Cloud Vision API:in ja asioiden metsästämiseen.

Ennen Vision Huntin kehitystä, suoritettiin eri pilvipalvelualustojen ominaisuuksien vertailua ja niistä löytyvien tekoälykomponenttien tutkimista. Näiden perusteella kehittyi kuva siitä, minkälaisia toimintoja oli mahdollista toteuttaa ja mitä eri ohjelmistokomponentteja tulisi käyttää. Alustan vaatimuksena oli Android-käyttöjärjestelmän puhelimet.

Pelissä pelaaja saisi tehtäviä, joista voisi valita minkä haluaa suorittaa.Tehtävät sisältävät kuvattavan asian, vaikeustason ja pistemäärän, minkä sen suorituksesta saa. Tehtävät kestäisivät vain tietyn aikaan, esimerkiksi maanantaista klo 00.00 sunnuntaihin klo 23.59 asti. Tehtäviä voisi olla monta aktiivisena samaan aikaan, ja niitä voisi saada lisää tietyn väliajoin. Miten vaikea tehtävän asia on löytää, sitä enemmän pisteitä tehtävästä saa. Tätä varten tarvitsi kartoittaa lista asioita, mitä pelissä etsittäisiin ja miten niitä löytyy normaalista kaupunkiympäristöstä.

Vision Huntin tarvitsisi pystyä ottamaan kuvia puhelimen kameralla. Sovelluksen täytyi pystyä ottamaan yhteys Vision API:n, joten toimiva verkkoyhteys oli pakollinen vaatimus. Tämän lisäksi sovellukseen piti luoda logiikkaa, jolla se tulkitsi saamansa analyysin. Logiikan kehitys Vision Huntissa perustui pelkästään tunnistamaan asiat kuvasta, eikä sitä miten todennäköisesti jokin asia löytyy tai miten paljon siitä on kuvassa.

Vision Huntissa pelin tehtävät ovat sovelluksen omassa muistissa tallennettuna puhelimeen, mutta tarkoituksena oli toteuttaa taustajärjestelmä, joka vastaisi tehtävien luomisesta ja hallinnasta. Jatkokehitys osiossa tätä käydään läpi tarkemmin. Prototyyppi käyttää lokaalia Room Persistence Library:a tietokannan luomiseksi. Tietokannassa voidaan lukea, lisätä, poistaa ja muokata tehtäviä tarpeen mukaan.

Koska Vision Huntin kehityksessä ei ollut mukana graafikkoa, on käyttöliittymä tehty yksinkertaiseksi ilman erillisiä animaatioita tai grafiikkoja. Toiminnallisuus oli etusijalla prototyypin kehityksessä, joten ulkoasu on tämän takia yksinkertainen. Vision Huntin grafiikat ja animaatiot ovat Android Studiosta löytyviä valmiita tai itse luotuja.

## 4.2 Prototyypin ominaisuudet

Vision Hunt kehitettiin teknologia kokeiluna Android-puhelimia ja tabletteja varten. Tarkoitus oli kehittää peli prototyypin ominaisuuksien ympärille, joka toimisi käyttäjille kannustimena lähteä liikkumaan ympäristöystävällisesti päivittäin. Pelillisyys sovellukseen tulisi tehtävien muodossa, joita käyttäjät saisivat tietyn väliajoin, kuten esimerkiksi kahdeksan tehtävää viikon alussa. Tehtävät vaatisivat käyttäjää kuvaamaan puhelimen kameralla sovelluksen kautta tiettyjä ulkoa löytyviä asioita ja sovellus käyttää Google Cloudista löytyvää kuvan analysointi palvelua tarkistaakseen löytyykö tehtävien vaatimaa asiaa kuvasta vai ei.

Sovellus vaatii toimiakseen verkkoyhteyden, sillä muuten kuvaa on mahdotonta saada analysoitua Google Cloudista löytyvällä Vision API:lla. Vision Hunt:iin oli myös suunniteltu taustajärjestelmä, joka vastaisi pelaajien käyttäjätileistä, tehtävien luomisesta sovellukseen ja tehtävien suoritusten merkitsemiseen. Prototyyppistä jätettiin kuitenkin taustajärjestelmä pois, sillä sen kehitys olisi vienyt liikaa aikaa. Vision Hunt prototyypissä tehtävien suoritus ja pisteytys tapahtuivat sovelluksen sisällä, eikä pelaajalla ollut erillistä käyttäjätiliä. Pelaajan eteneminen tallentuu puhelimen muistiin ja tehtävät olivat lokaalissa tietokannassa, jota pystyi muokkaamaan sovelluksen kautta.

Kamera ominaisuus sovellukseen toteutetaan käyttämällä Androidin omaa kamera sovellusta, jota käytetään kutsumalla sitä intent-objektilla. Kun käyttäjä ottaa kuvan, se tallennetaan sovelluksen luomaan kansioon ja käyttäjä voi nähdä kuvan, ennen kuin päättää haluaako lähettää sen analysoitavaksi. Kuva tallennetaan siten, että edellinen sovelluksella otettu kuva korvataan, ettei puhelimen muistia täytetä turhaan. Jatkokehityksessä mahdollisuutena oli tarjota käyttäjälle mahdollisuus säilyttää otetut kuvat, mutta testausvaiheessa kyseiselle ominaisuudelle ei ollut tarvetta.

Kun kuva lähetetään analysoitavaksi, luodaan kuvasta kopio, jonka laatua ja kokoa pienennetään. Tämä tehdään sen takia, että nykypuhelimien ottamat kuvat ovat niin laadukkaita, että niiden tiedostokoko on suuri. Suuri tiedostokoko aiheuttaa ongelmia kuvan lähettämisessä analysointia varten, koska sen lähetys kestää liian kauan. Tämä ongelma tuli vastaan testivaiheen alussa, jossa kuvan analysoinnissa meni usein yli 20 sekuntia. Ongelman paikallistamisen jälkeen prosessia saatiin nopeutettua huomattavasti johtaen

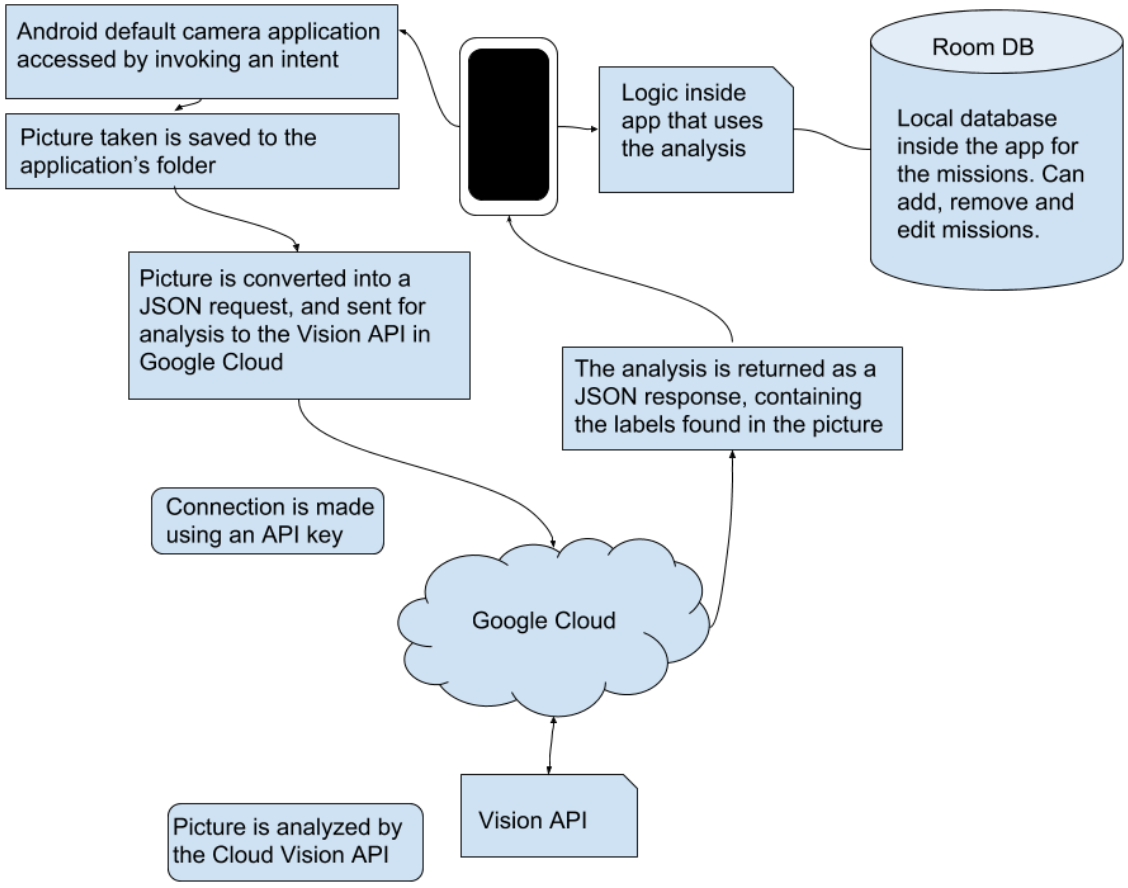
noin 2-4 sekunnin lähetysaikaan ilman, että kuvan analyysin tarkkuus kärsi. Nopeuteen vaikutti myös käyttölaitteen verkkoyhteyden nopeus, mutta tähän ei kehitetty ratkaisua prototyyppi vaiheessa.

Kuva liitetään JSON pyyntöön, joka sisältää kuvalle tehtävät analysointi-pyynnöt. Yhteys Vision API:n otetaan API avaimen avulla ja sovellusta varten on luotu Google Cloud projekti, joka antaa luvan sovellukselle käyttää Vision API:a. Vision API suorittaa analyysin lähetettyyn kuvaan annetuin parametrein, ja palauttaa analyysin JSON muotoisena vastauksena. Sovellus käy läpi kaikki löydetyt asiat analyysista, ja sen mukaan määrittää löytyikö tehtävän vaatimaa asiaa kuvasta vai ei. Jos asia löytyi, niin pelaaja saa ilmoituksen tästä ja hänelle annetaan pisteitä tehtävän pistemäärän mukaan. Sen jälkeen pelaaja voi lukea kuvasta löytyneet asiat tai palata takaisin päävalikkoon. Jos kuvasta ei löytynyt tehtävän vaatimaa asiaa, voi hän silti tarkastella analyysia ja nähdä mitä asioita kuvasta löytyi.

Vision Huntin päävalikosta näkee sillä hetkellä aktiiviseksi määritellyn tehtävän. Tehtävällä on vaikeustaso, pistemäärä, kuvattavan asian kuvaus ja tunniste, jota Vision API käyttää kategorioimaan kuvasta löytyviä asioita. Päävalikko sisältää painikkeet kamerasovelluksen avaamiseen, uuden tehtävän generoimiseen ja tehtävän automaattiseen suorittamiseen. Valikon lisäksi sovelluksessa on toimintojen testaamista varten tehtävien listaus, luominen, poisto ja muokkaus ominaisuudet eri välilehdellä. Välilehtien välillä voi liikkua alhaalta löytyvällä valikolla, tai sormella pyyhkäisemällä. Viimeinen sivu sisältää tietoja sivun, jossa on viimeksi otetusta kuvasta esikatselu kuva ja tallennettu analyysi. About sivu sisältää myös sovelluksen version ja lyhyen kuvauksen sovelluksesta.

## 4.3 Toiminta

Kuviossa 1 on Vision Huntin arkkitehtuuri kuvattuna (KUVIO 1). Android-sovelluksessa sijaitsee pelin logiikka ja tietokanta, kun taas Google Cloudissa sijaitsee kuvan analysointia varten Cloud Vision API rajapinta. Kuvat lähetetään Cloud Vision API:lle analysoitavaksi, joka analysoi kuvan ja palauttaa Vision Huntille tehdyn analyysin. Vision Hunt tulkitsee analyysin logiikallaan, jolloin käyttäjä voi suorittaa pelin tehtäviä. Suorittaessaan tehtävän pelaaja saa tehtävään merkityn pistemäärän verran pisteitä ja tehtävä merkitään suoritetuksi, ettei sitä voida suorittaa uudelleen.




KUVIO 1. Vision Hunt Prototyypin arkkitehtuuri

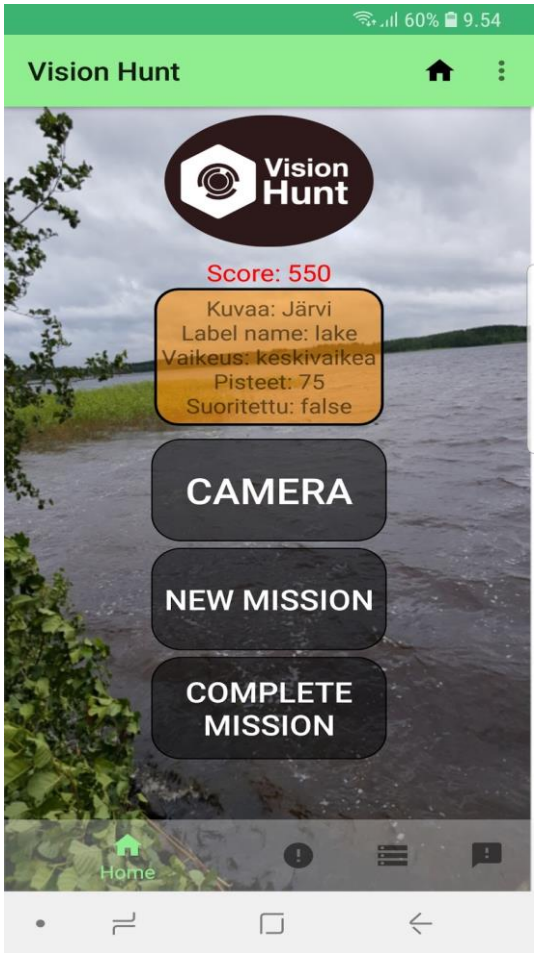
Pelissä on ideana kuvata tehtävissä pyydettyjä asioita puhelimen kameralla, ja täten saada tehtävistä pisteitä. Tehtävällä on kuvauksen kohde, kuvaus tehtävästä, vaikeustaso ja pistemäärä jonka sen suorittamisesta saa. Pelaajalla on kerrallaan yksi tehtävä, joka voi olla joko suoritettu tai suorittamatta. Pelaaja voi pyytää uuden tehtävän päävalikosta, jolloin hänelle arvotaan uusi satunnainen, eri niminen suorittamaton tehtävä tietokannasta. Tätä tehtävää varten pelaajan on otettava kuva tehtävän pyytämästä asiasta ja sovellus etsii pyydettyä asiaa saamastaan analyysista, merkiten tehtävän suoritetuksi, jos pyydetty asiaa löytyy.

Tehtävien tietokantaa voi muokata sovelluksen sisällä. Tietokannalla on lisäys, poisto ja muokkaus toiminnot tehtäville, jolla voidaan muokata, mitä tehtäviä tietokannassa on. Tärkein asia luodessa tietokantaan tehtäviä on tietää, minkä kuvauksen kohteen tehtävälle antaa sillä tätä arvoa vastaan suoritetaan vertailulogiikka. Kuvattava asia on siis oltava jokin Cloud Vision API:n tunnistama asia, jos halutaan luoda tehtävä, jonka voi suorittaa. Tätä varten luotiin esimerkki Excel lista asioista ja niiden käyttämistä kuvaus tunnisteesta, joita voitiin referoida tehtäviä luodessa.

## 4.4 Käyttöliittymä

Vision Huntin käyttöliittymä on yritetty toteuttaa normaalien Android-sovellusten standardien mukaisesti. Tämä tarkoittaa sitä, että käyttäjän eri toiminnoilla on oletusten mukaiset seuraukset ja että graafinen toteutus käyttää tunnettuja kuvia esittämään erilaisia toimintoja. Erilaiset valikot on toteutettu vastaamaan normaalien Android-sovellusten tyylejä, jotta sovellusta olisi mahdollisimman helppo käyttää.

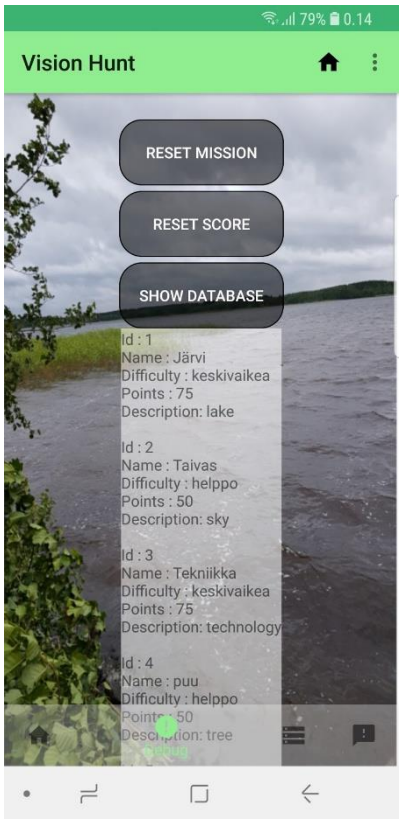
Päävalikko ja kaikki sen sisältö on toteutettu käyttämällä vain yhtä aktiviteettia. Aktiviteetti sisältää ViewPager-objektin, johon esitetään eri fragmentteja, ja widgetin, jolla toteutetaan sovelluksen alakulmaan neljän painikkeen alavalikko. Fragmentit esittävät sisältöä sivulle Viewpagerin-objektin kautta ja niiden välillä voidaan navigoida joko alavalikon painikkeiden kautta, tai pyyhkäisemällä sivulle.




KUVA 11. Vision Hunt prototyypin päävalikko

Ensimmäinen fragmentti sisältää päävalikon, ylläoleva KUVA 11 on ruudunkaappaus sovelluksen päävalikosta. Päävalikossa on sovelluksen logo, käyttäjän pisteet, aktiivinen tehtävä ja napit kamera funktiolle, uuden tehtävän generoinnille ja tehtävän manuaaliselle

suorittamiselle. Toinen fragment sisältää painikkeet aktiivisen tehtävän resetoimiselle, pisteiden resetoimiselle ja tietokannan esittämiselle samaan fragmenttiin. Kuvassa 12 on fragmentti 2 näkymä, jossa on tietokannasta löytyvät tehtävät listattuna show painikkeen-painamisen jälkeen (KUVA 12). Kolmas fragment sisältää tietokannan toiminnot, eli lisäys, poisto ja muokkaus optiot. Nappeja painamalla avataan toimintoja varten käytettävät kentät. Neljäs fragment sisältää tietoja sivun, jolla on sovelluksen tiedot, viimeisin analysoitu kuva ja kuvan analyysi.

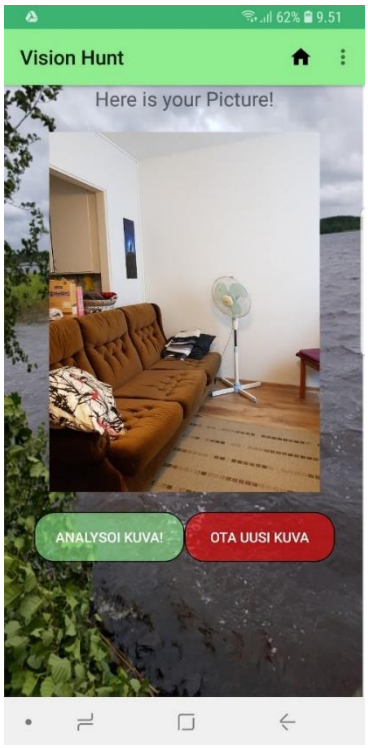



KUVA 12. Vision Hunt prototyypin fragment 2

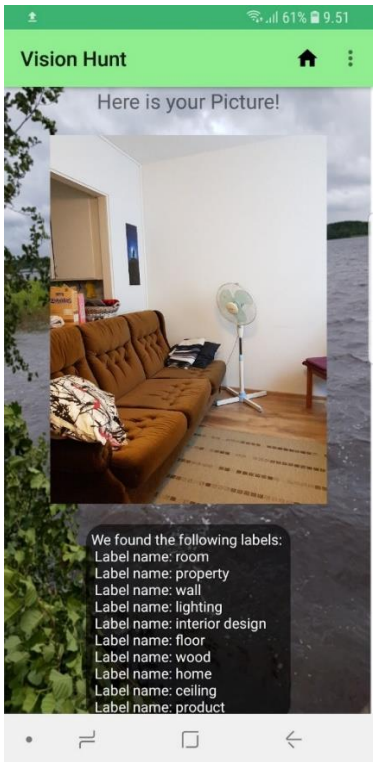
Kamera ominaisuus käyttää Androidin omaa kamera sovellusta, joten sen käyttöliittymä on sama kuin, mikä käyttäjän puhelimessa on. Kuvan otettua käyttäjältä varmistetaan, onko hän tyytyväinen kuvaan vai haluaako hän ottaa uuden. Tämän jälkeen sovellus avaa toisen aktiviteetin, jossa on kuvasta esikatselu ja napit kuvan analysointia ja pääsivulle palaamista varten. Painikkeiden väriteemat on valittu niin, että käyttäjälle on selkeää, kummalla edetään ja kummalla palataan sovelluksessa. Tästä esimerkki kuvassa 13 (KUVA 13).

Kun käyttäjä lähettää kuvan analysoitavaksi, avataan tälle lataus animaatio ja piilotetaan napit latauksen ajaksi. Analyysin vastaanotettua sovellus esittää kaikki kuvasta löydetyt asiat tekstikentässä, ja jos tehtävässä pyydetty asia löytyy, tulee käyttäjälle ilmoitus, joka

sisältää tehtävän kuvauksen kohteen ja pistemäärän. Kuvassa 14 on näkymä toisesta aktiviteetista, kun kuvasta on saatu analyysi (KUVA 14). Analyysia voi vierittää alaspäin tarvittaessa ja siinä näkyy kaikki kuvasta löytyneet tunnisteet.




KUVA 13. Vision Hunt prototyypin toinen aktiviteetti




KUVA 14. Vision Hunt prototyypin toinen aktiviteetti analyysin kanssa

## 4.5 Ohjelmointi

Vision Hunt prototyyppi on ohjelmoitu Java ohjelmointikielellä käyttäen Android Studio kehitysympäristöä. Prototyypissä käytetään Room Persistence Librarya luomaan lokaali tietokanta, jota käytetään tehtävien luomiseen ja tallentamiseen. Google Cloudin Vision API asiakaskirjastot on liitetty puhelin sovellukseen, jotta sovelluksella voidaan tehdä kutsuja Vision API:n.

Sovelluksen kaksi tärkeintä aktiviteettia ovat päävalikon aktiviteetti ja kamera aktiviteetti. Päävalikossa on toiminnot tehtävien hakemiselle, kameran avaamiselle ja tietokannan muokkaukselle. Kamera aktiviteetti sisältää kuvanottamis toiminnon, kuvan lähettämisen Vision API:lle ja kuvan esikatselun.

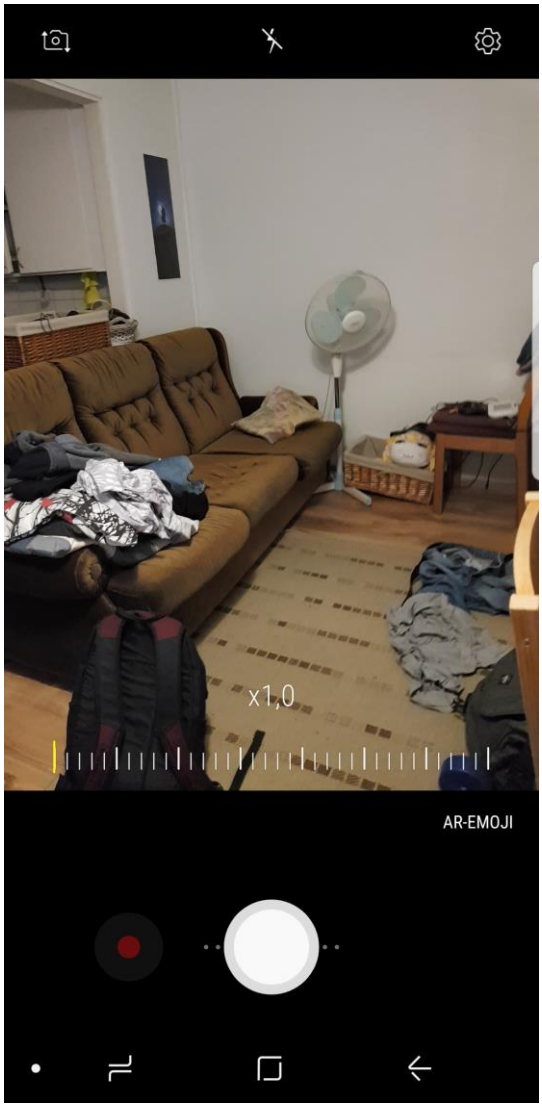
Päävalikko perustuu yhteen aktiviteettiin, johon ladataan sisältö käyttämällä viewpager-objektia, jolla ladataan eri fragmentteja aktiviteetille. Näin voidaan toteuttaa helposti monisivuinen valikko, jonka eri sivut pystyvät keskustelemaan ja jakamaan tietoja keskenään. Päävalikkoa voi navigoida alhaalla olevan valikon kautta tai kosketusnäytön avulla pyyhkäisy liikkeellä.

Tietokantaan lisätään objekteja käyttämällä sitä varten luotua Mission luokkaa. Mission luokan objektit ovat automaattisesti inkrementoituvia ja niillä jokaisella on nimi, vaikeustaso, pistemäärä ja kuvaus. Kun uusi tehtävä luodaan tietokantaan, annetaan sille nämä parametrit, jonka jälkeen ne lisätään tietokantaan käyttämällä MyDao rajapintaa, joka sisältää tietokannan muokkausta ja lukemista varten funktiot. Tietokannan eri objektit voidaan lukea fragmentilla kaksi, joka ottaa jokaisen Missions objektin tietokannasta, ja luo fragmentin tekstikenttään sitä vastaavan tehtävän luettavaksi.

Kun pelissä asetetaan jokin tehtävä aktiiviseksi tehtäväksi, eli siksi tehtäväksi, jota käyttäjä yrittää suorittaa, luodaan sitä varten uusi tehtävä luokan objekti. Tämä muodostetaan ottamalla tietokannasta uusi eriniminen Mission objekti ja tallentamalla sen arvot tehtävä luokan objektiin. Tehtävä luokan objekti tallennetaan puhelimen muistiin, jotta se pysyy tallessa, vaikka sovellus suljettaisiin välissä.

Tietokantatoiminnot ovat päävalikon kolmannella fragmentilla. Kolmannella fragmentilla voidaan suorittaa eri toimintoja tietokantaan. Tietyn toiminnon nappia painamalla avautuu sivulle syöttökentät ja lähetä nappi, johon käyttäjä voi tehdä haluamansa muutokset ja sitten julkaista ne tietokantaan. Tietokantaan voi lisätä uusi tehtävä, poistaa tehtävä tai editoida jotain siellä olevaa tehtävää.

Kamera on toteutettu käyttämällä Android-puhelimen oletus kamerasovellusta. Kamerasovellus avataan luomalla uusi intent-objekti kameran käyttämiselle, joka luodaan, kun avataan kamerafunktion aktiviteetti. Intent-objektia varten luodaan myös tiedostontuottaja, jolla kerrotaan, minne kamerasovelluksella otettu kuva tallennetaan ja ohjataan otettu kuva Vision Hunt sovellukseen. Seuraava kuva esittää sovellusta, kun kamera on aktiivisena (KUVA 15). Kuvat ottamisen jälkeen käyttäjälle aukeaa kamera aktiviteetti, jossa hän näkee esikatselun ottamastaan kuvasta ja hänellä on painikkeet kuvan analysointia ja uuden kuvan ottamista varten.




KUVA 15. Android-kamera jota kutsutaan intent-objektilla sovelluksessa

Kun käyttäjä painaa kuvan analysointi nappia, luodaan otetusta kuvasta lähetystä varten sopiva versio. Kuvan kopio pakataan ja skaalataan pienemmäksi, jonka jälkeen siitä luodaan JSON objekti. JSON objektiin asetetaan tyypiksi “LABEL_DETECTION” ja käsitelty

kuva liitetään objektiin Base64 koodattuna merkkijonona. Yhteys Vision API:n luodaan Vision Builder luokalla, jonka avulla voidaan myös luoda JSON pyyntö. Vision Builder käyttää sovellukselle tarkoitettua API avainta yhteyden luomiseen.

Kun JSON pyyntö on lähetettyä, saadaan vastauksena JSON vastaus Vision API:lta, jossa on jokainen kuvasta löydetty tunniste. Nämä arvot listataan aktiviteetissa olevaan tekstikenttään, jossa käyttäjä voi lukea niitä. Myös puhelimen muistiin tallennetaan kaikista viimeisin saatu analyysi, jota voi tarkistella päävalikon kautta tietoja fragmentissa. Jokaista analyysin arvoa verrataan sen hetkiseen aktiivisen tehtävän tunniste arvoon, ja jos jokin tunniste vastaa tätä, ilmoitetaan käyttäjälle hänen suorittaneen tehtävän näyttämällä hänelle ilmoitus, joka kertoo tehtävän pistemäärän. Käyttäjän pistemäärä päivittyy ja tallentuu puhelimen muistiin.

# 5 JATKOKEHITYS

## 5.1 Kaupallistaminen

Vision Hunt:iin oli suunnitteilla järjestelmä, jossa pelaajat voivat vaihtaa pelissä ansaitsemiaan pisteitä erilaisiin yritysten tarjoamiin palveluihin. Yritykset maksaisivat siitä, että heidän tarjouksensa voisivat olla pelin sisäisessä kaupassa. Vastalahjaksi yritykset pystyisivät tavoittamaan suuren käyttäjäkunnan pelin kautta. Tämä mahdollistaisi pelin kaupallistamisen niin, ettei pelaajalla tarvitsisi näyttää mainoksia tai peliin implementoida mikrotransaktioita.

Peliin olisi voinut myös toteuttaa tehtäviä, jossa pelaaja joutuu kuvaamaan jotain yrityksen tuotetta. Tämä kannustaa pelaajia vierailemaan yrityksen tiloissa ja ehkä käyttämään tämän tarjoamia palveluita. Tämän ominaisuuden toteuttamiseen olisi käytetty Vision API:sta löytyvää logon tunnistusta ja tekstin tunnistusta. Näillä voitaisiin saada kuvasta tarvittava tieto yritysten haluamien asioiden tunnistamiseen, kuten vaikka tietyn merkkisten tuotteiden muodossa. Tunnistamiseen tarvitsisi kehittää logiikka, jolla tunnistetaan tietynlaisia logoja. Vision API:n Logo Detection tunnistaa vain tiettyjä logoja, jotka ovat tunnettuja maailmalla. Vaikka sen tunnistamien asioiden lista kasvaa jatkuvasti, voi vähemmän tunnettujen suomalaisyritysten logojen tunnistus olla haastavaa. Vaikkei tunnistus logiikkaa ole vielä kehitetty sovellusta varten, on siihen liittyvää ideointia tehty.

Esimerkiksi, jos tarkoituksena on tunnistaa ‘Eldorado’-merkkinen banaani, voidaan joko yrittää tunnistaa logo suoraan, tai käyttää tekstin tunnistusta tunnisteen kanssa toteamaan, että kuvassa on banaaneja ja että kuvassa on teksti ‘Eldorado’. Toinen vaihtoehto tähän on monimutkaisempi, sillä se vaatii omien koneoppimismallien luontia tietyn asian tunnistamiselle. Tällöin voidaan suoraan luoda joko tunnistettava logo ‘Eldorado’ logosta, jolloin ei tarvita erillistä logiikkaa sovelluksessa tunnistukseen.

Joka tapauksessa mobiilisovelluksen ideana on tehdä rahaa jollain tavalla. Näin alkuvaiheessa on hyvin epäselvää, millä tavalla se käytännössä onnistuu. Yritysten kanssa yhteistyön tekeminen vaatii paljon aikaa, osaamista ja hyvän valmiin tuotteen. Kun prototyypistä päästään etenemään seuraavaan kehitysversioon, on myös helpompi kartoittaa niin käyttäjien kuin yritysten kiinnostusta Vision Hunt sovellukseen.

## 5.2 Ominaisuudet

Jatkokehitystä varten oli runsas lista asioita, joiden avulla pelistä voisi saada mielenkiintoisen pelaajalle ja viedä prototyyppi tasolle, joka vastaa laadukasta peliä josta käyttäjät voisivat kiinnostua. Kaikkia suunniteltuja ominaisuuksia ei luultavasti ehtisi saada ensimmäiseen versioon, mutta tarkoituksena oli lisätä ominaisuuksia versio päivityksissä.

Tehtäväpaketit ovat uusi tapa toteuttaa tehtävät peliin. Prototyypissä tehtävät olivat yksittäisiä ja niitä pystyi olemaan vain yksi aktiivisena kerralla. Jatkokehityksessä luotiin tehtäväpaketit, jotka olivat tehtäviä sisältäviä paketteja, joista pelaaja voisi valita tehtävän jota lähtee suorittamaan. Tehtäväpaketit kestäisivät määrätyn ajan, jonka jälkeen ne katoaisivat pelaajalta. Tämä kannustaa häntä suorittamaan mahdollisimman monta tehtävää. Tehtäväpaketteja voisi olla aktiivisena monta samaan aikaan ja ne voisivat alkaa ja loppua eri aikaan.

Yksi tällainen ominaisuus oli erilainen tapa suorittaa tehtäviä. Pelaaja saisi kuvattavan asian sijaan tehtävän, joka vaatisi hänen menevän tiettyyn GPS sijaintiin. Sovelluksessa näkisi kartalla, missä päin sijainti olisi ja pelaaja voisi kartan avulla matkustaa oikeaan paikkaan. Kun pelaaja saapuu oikeaan paikkaan ja avaisi pelin, saisi hän tehtävän suoritettua. Tämä kannustaa ihmisiä liikkumaan enemmän ja tätä ominaisuutta voisi yhdistää johonkin kuvaamistehtävään, esimerkiksi mene paikkaan X ja kuvaa asia Y.

Toinen idea oli suunnitella tehtäviä, jotka tulisivat kausittain. Kausittaiset tehtävät liittyisivät tiettyihin teemoihin, ja voisivat olla esimerkiksi joulun aikana jouluun liittyviä. Tällöin kuvattavat asiat liittyvät kauden teemaan. Kauden aikana voisi myös saada kausittaisia pisteitä, joita voisi käyttää pistekaupassa lunastamaan sen kauden tuotteita. Tällöin joka kausi olisi pelaajille mielenkiintoinen ja uuden kauden alkaminen tarkoittaisi uusia ostettavia asioita, joiden eteen pelata peliä.

Peliin tulisi myös mahdollisuus lisätä kaveriksi toisia käyttäjiä, jolloin voisi pisteiden keräämistä verrata toisiin käyttäjiin. Myös ideoita yhteiseen pelaamiseen oli esillä, mutta niitä ei ole vielä saatu kehitettyä. Esimerkiksi GPS tehtävissä voisi nähdä kaverinsa kartalla.

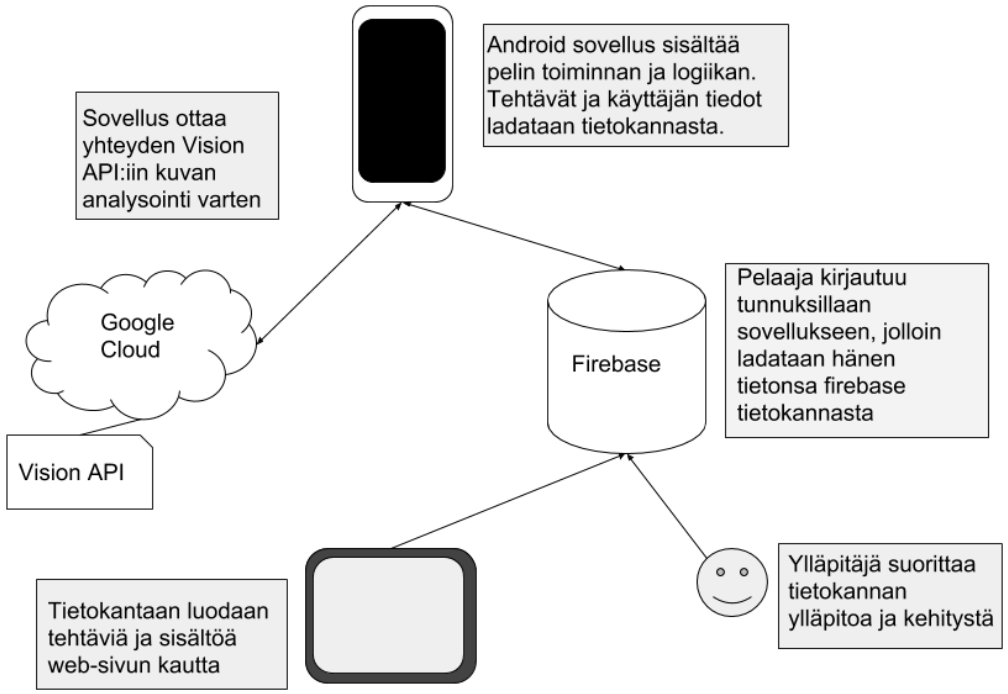
Myös erilaisia teemoja ja tapoja muokata sovelluksen ulkonäköä henkilökohtaiseksi pelaajalle oli suunnitteilla. Tämä tapahtuisi joko valmiiksi luoduilla väriteemoilla, joita voisi vaihtaa asetusten kautta, tai erilaisilla kosmeettisilla muutoksilla sovelluksen eri osiin. Esimerkiksi oman tunnuksen ulkonäköä voisi muokata erilaisten bannerien muodossa.

## 5.3 Taustajärjestelmä

Vision Hunt peliin suunnitteilla oli, että peliin kirjaudutaan tunnuksilla, johon tallentuu pelaajan eteneminen. Pelitunnusten hallintaa varten tarvitaan taustajärjestelmä, jota kautta voidaan tehdä kaikkea tunnusten hallintaan liittyvää. Myös pelin tehtävät toteutettaisiin taustajärjestelmän kautta, sillä prototyypin puhelimen oman muistin käytöllä esiintyy liikaa ongelmia pelin jatkuvasti päivittyvän sisällön kanssa ja montaa ominaisuutta ei voida toteuttaa. Taustajärjestelmä siis vastaisi ainakin pelitunnuksista, pelin sisällön luomisesta ja synkronoinnista. Pelaajan pisteiden päivityksen siirtäminen sovelluksesta pois on myös tärkeää, jotta pystytään tallentamaan ja seuraamaan pelaajien etenemistä, sekä estämään ja korjaamaan virheitä.

Aina puhelinsovelluksen käynnistyessä, synkronoi se taustajärjestelmän kanssa ja pelaaja kirjataan hänen pelitunnuksella sisään. Pelissä eteneminen on liitetty käyttäjätunnukseen, joten kaikki pisteet ja tehtävät siirtyvät pelitunnuksen mukana, jos esimerkiksi päätelaite vaihtuu. Pelitunnukset mahdollistavat myös toisien kanssa pelaamisen ja kausittaisen pistetilaston luomisen.

Tämänhetkinen taustajärjestelmän toteutus on suunnitteilla Firebase pohjaisena projektina. Firebase:lle toteutetaan tietokanta, jota ylläpitäjä voi operoida Firebase konsolin kautta. Pelin sisällön luojalla luodaan verkkosivu, jonka kautta voi suunnitella ja toteuttaa tehtäväpaketteja tietokantaan.




KUVIO 2. Vision Hunt jatkokehityksen arkkitehtuuri

Kuviossa 2 on havainnollistettu, minkälainen arkkitehtuuri on suunnitteilla Vision Huntin seuraavaan versioon (KUVIO 2). Puhelinsovellus toimii asiakasohjelmana, johon sisältö ladataan synkronoimalla tiedot serverin kanssa. Käyttäjä kirjautuu automaattisesti omilla pelitunnuksilla palveluun, jolloin hänelle ladataan hänen pelitunnuksille ominainen sisältö. Kuvien analysointi toteutetaan samalla tavalla kuin prototyypissä, mutta tehtävien suoritus päivitetään Firebase tietokantaan puhelimen oman muistin sijaan. Suurin osa Vision Huntin tämänhetkisestä toiminnasta siirtyy pois sovelluksesta Firebase tietokantaan. Tämä tekee sen, ettei sovellus ole riippuvainen tietylle laitteelle tallennetusta datasta, vaan kaikki henkilökohtainen sisältö kuten pelissä eteneminen ja tehtävät ladataan sovellukseen synkronoimalla tietokannan kanssa.

Arkkitehtuurissa on myös kaksi uutta käyttäjäroolia. Ylläpitäjä vastaa Firebase tietokannan hallinnasta ja pelitunnuksien ylläpidosta. Pelin sisällön luoja vastaa uusien tehtävien luomisesta peliin. Tämä tapahtuu verkkosivu käyttöliittymän kautta. Pelin sisällön luojan vastuulla olisi suunnitella ja toteuttaa erilaisia tehtäviä pelaajille, jotka julkaistaisiin Vision Hunt pelisovellukseen. Koska tehtävät toteutettaisiin määrätyn ajan kestävissä tehtäväpaketeissa, tulisi pelin sisällön luojan kehittää pelaajille mielenkiintoisia tehtävä paketteja tietyn väliajoin, jotta pelissä riittää pelattavaa.

Prototyyppi-vaiheessa backendia ei vielä ollut olemassa, joten käyttäjätilejä ei myöskään sovelluksessa ollut. Ylläpitäjän ja pelin sisällön luojan tehtävät liittyvät vasta tuleviin sovellus versioihin. Käyttäjäroolien tehtävät tulevat vielä tarkentumaan, kun Vision Huntin kehitys etenee. Myös backend serverin vaatimukset tulevat muuttumaan, kunhan jatkokehitys etenee. Kuviossa 2 esitetty arkkitehtuuri on toimiva teoriassa, mutta testauksen ja jatkokehityksen vaatimusten luonnin jälkeen voi ilmetä ongelmia jotka vaativat muutoksia arkkitehtuuriin. Suurin kysymys tällä hetkellä on miten Firebase tietokanta kestää kuormitusta, jos siihen tulee paljon liikennettä ja miten hyvin siinä olevaa dataa pystyy valvotusti muokkaamaan ilman että ongelmia ilmenee. Myös käyttäjätietokannan toteuttaminen on vielä epäselvä, koska Vision Huntille ei ole viimeistelty vaatimusmäärittelyä.

# 6 YHTEENVETO

Android-prototyyppi Vision Hunt saatiin kehitettyä valmiiksi kesän 2018 aikana. Suunnitteluvaihe ja Google Cloud -pilvipalvelun opetteluun ja käyttöönottoon meni noin kuukausi ja kehitysvaiheeseen noin kaksi ja puoli kuukautta. Lopputuloksena Vision Huntilla voidaan ottaa ja tallentaa kuva, jonka voi analysoida Google Cloud Vision API:lla. Vision Huntissa toteutuu myös pelillisyys suoritettavien tehtävien muodossa, josta saa hyvät lähtökohdat jatkokehitystä varten. Jatkokehitystä on pohdittu melko paljon ja ideoita, miten Vision Huntia viedä eteenpäin on paljon. Tavoitteena tulevaisuudessa olisi kehittää sovelluksesta 1.0 versio, jota pystyisi viemään käyttäjätesti vaiheeseen.

Pelillisyyttä prototyyppin ehdittiin kehittää melko vähän. Vaikka Vision Huntilla pystyy etsimään erilaisia asioita ja tunnistamaan ne, on pelillinen kokemus melko vaisu. Sellaisenaan Vision Hunt ei vielä sovellu käyttäjätestausta varten, joka on yhtenä tulevaisuuden tavoitteena. Toiminnaltaan Vision Hunt on kuitenkin täysin toimiva, ja siinä on toteutettu tärkeimmät ominaisuudet kuten Cloud Vision API:n käyttäminen, kuvan ottaminen ja pelilogiikan toiminta. Tärkein tulevaisuuden kehitysidea on saada sovelluksen tietokanta siirrettyä pois puhelinsovelluksesta pilveen, ja luoda tietokantaan käyttäjätilit jokaiselle pelaajalle jolla kirjaudutaan palveluun. Tämän jälkeen pelillisyyden kehitys on tärkeää, jotta Vision Hunt tuntuisi enemmän peliltä kuin kuvan analysointi sovellukselta.

Pilvipalveluiden tekoälyrajapintoihin päästiin opinnäytetyössä tutustumaan nopeasti, mutta aihe on melko uusi ja tulee kehittymään tulevaisuudessa paljon. Tekoälyn käyttäminen sovelluksissa muuttuu koko ajan helpommaksi ja sen käyttöä pyritään tekemään mahdollisimman yksinkertaiseksi, jotta mahdollisimman moni pystyisi käyttämään sitä. Pilvessä olevat tekoälypalvelut ovat helppokäyttöisiä ja niiden hyödyntäminen erilaisissa projekteissa onnistuu ilman kalliiden laitteiden hankintaa ja ylläpitoa. Pilvessä olevat tekoälyrajapinnat ja koneoppiminen ovat uusinta huipputekniikkaa ja niiden tulevaisuus näyttää lupaavalta. Tekoälyn demystifiointi helpottaa sen valjastusta arkikäyttöön, niin sovelluksissa kuin IoT (Internet of Things) ratkaisuissa.

Mielenkiintoiseksi tulee se, miten hyvin valmiiksi koulutetut tekoälymallit riittävät sovelluskehitykseen. Itse koulutetut tekoälymallit ovat varmastikin aluksi haastavia saada toimimaan halutulla tavalla, mutta niiden käytön opettelu on hyödyllinen taito osata tulevaisuudessa. Yksinkertaiselta tuntuvan tekoälypalvelun käyttö onnistui helposti ja palveluiden tarjonnan laajentuessa voi tulevaisuudessa olla todellakin paljon erilaisia tekoälyratkaisuja kaikenlaisissa sovelluksissa, niin mobiili sovelluksissa kuin IoT ratkaisuissa.

Nykyisten pilvipalveluiden palveluiden tarjonta on kattavaa ja edullista, jonka takia yritysten toiminnan siirtäminen pilveen kuulostaa yhä houkuttelevammalta. Omien fyysisten serverien hankinta ja ylläpito ei kuulosta kovin järkevältä, kun saman toiminnon voi siirtää pilveen, joka on usein halvempaa ja helpompaa yritykselle. Pilveen ei tarvitse kerralla siirtää tai toteuttaa kaikkea, vaan voidaan käyttää niin sanottua hybridi ratkaisua, jos kaikkea ei haluta siirtää pilveen.

# LÄHTEET

MyBroadBand 2018. The most popular operating system for desktop and mobile.[viitattu 14.11.2018] Saatavissa: https://mybroadband.co.za/news/software/258733-the-most-popular-operating-systems-for-desktop-and-mobile.html

Lahti 2018. CitiCAP – Älykkäitä liikkumisratkaisuja Lahdesta Eurooppaan [viitattu 8.11.2018]. Saatavissa: https://www.lahti.fi/palvelut/luonto-ja-ymparisto/citicap

Google Cloud 2017a. What is cloud computing? [viitattu 8.11.2018]. Saatavissa: https://cloud.google.com/what-is-cloud-computing/

Google Cloud 2017b. What is machine learning? [viitattu 8.11.2018]. Saatavissa: https://cloud.google.com/what-is-machine-learning/

Google Cloud 2018a. Cloud AI building blocks [viitattu 8.11.2018]. Saatavissa: https://cloud.google.com/products/ai/building-blocks/

Google Cloud 2018b. Cloud Vision [viitattu 8.11.2018]. Saatavissa: https://cloud.google.com/vision/

Google Cloud 2018c. AI Building Blocks [viitattu 27.11.2018] Saatavissa: https://cloud.google.com/products/ai/building-blocks/

Google Cloud 2018d. Cloud AI Products [viitattu 27.11.2018] Saatavissa: https://cloud.google.com/products/ai/

Google Cloud 2018e. Detect Multiple Objects [viitattu 27.11.2018] Saatavissa: https://cloud.google.com/vision/docs/object-localizer

Android developer 2018a. Platform Architecture [viitattu 8.11.2018]. Saatavissa: https://developer.android.com/guide/platform/

Android developer 2018b. Activity [viitattu 8.11.2018]. Saatavissa: https://developer.android.com/reference/android/app/Activity

Android developer 2018c. Intents and Intent Filters [viitattu 8.11.2018]. Saatavissa: https://developer.android.com/guide/components/intents-filters

Android developer 2018d. Sensors Overview [viitattu 8.11.2018]. Saatavissa: https://developer.android.com/guide/topics/sensors/sensors_overview

Amazon Web Services 2018. API-driven services bring intelligence to any application [viitattu 8.11.2018]. Saatavissa: https://aws.amazon.com/machine-learning/

Microsoft Azure 2018. AI Services[viitattu 8.11.2018] Saatavissa:
https://azure.microsoft.com/en-us/overview/ai-platform/