LAPIN AMK
Lapland University of Applied Sciences

POROMERKIN PIIRTÄMINEN JA TUNNISTAMINEN MOBIILISOVELLUKSESSA

Horneman Jonna

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2023

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Jonna Horneman	Vuosi	2023
Ohjaaja	Aku Kesti		
Toimeksiantaja	Paliskuntain yhdistys		
Työn nimi	Poromerkin piirtäminen ja tunnistaminen mobiilisovelluksessa		
Sivu- ja liitesivumäärä	32 + 6		

Opinnäytetyön tarkoituksena on luoda prototyyppi mobiilisovelluksesta, johon käyttäjä voi piirtää poromerkin ja etsiä piirtämänsä poromerkin omistajan tiedot. Opinnäytetyön tavoitteena on tuoda helpotusta porotalouden työskentelytapoihin ja selvittää, onko mahdollista kehittää piirtosovellus, joka tunnistaa käyttäjän piirtämät poromerkit. Opinnäytetyön toimeksiantajana toimii Paliskuntain yhdistys.

Sovelluksen frontend luodaan Flutterilla ja backend Pythonilla. Prototyyppiä voi käyttää Android- ja iOS-pohjaisilla laitteilla.

Opinnäytetyössä tutustutaan poromerkkeihin, Pythonin REST API:iin ja kuvavertailuun sekä Flutterin valikoituihin widgetteihin. Lisäksi käydään läpi sovelluksen suunnittelu- ja tekovaiheet sekä sovelluksen toiminta.

Opinnäytetyön tuloksena syntyi toimiva poromerkkipiirtohakusovellus. Sovelluksella käyttäjä voi piirtää sormella tai kosketusnäyttökynällä poromerkkiin erilaisia tekoja. Sovellus vertaa käyttäjän piirtämää poromerkkiä vertailukuviin ja palauttaa kuvaan sopivat poromerkkivastaavuudet, joista käyttäjä valitsee oikeat kuvat. Tämän jälkeen ohjelma palauttaa mahdollisen omistajan tiedot. Prototyyppiä voidaan jatkossa hyödyntää esimerkiksi porotalouden työskentelytapojen digitalisoimisessa.

Avainsanat	mobiilisovellukset, ohjelmointi, poronhoito
Muita tietoja	Opinnäytetyö sisältää esittelyvideon.

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Degree Programme in Information
and Communication Technology
Bachelor of Engineering

Author Jonna Horneman Year 2023
Supervisor Aku Kesti
Commissioned by Reindeer Herders' Association
Subject of thesis Mobile Application for Drawing and Identifying Reindeer Earmarks
Number of pages 32 + 6

The purpose of the thesis study was to create a prototype of a mobile application where the user can draw a reindeer earmark and search for information about the owner of that earmark. Also, the aim of the thesis study was to bring relief to the working methods of reindeer husbandry and to find out if it is possible to develop a drawing application that recognizes the reindeer earmarks drawn by the user. The was commissioned by The Reindeer Herders' Association.

The frontend of the application was created with Flutter and the backend with Python. Reindeer earmarks, Python's REST API and image comparison, as well as Flutter's selected widgets were studied. The design and creation phases of the application, as well as the operation of the application were described.

As a result of the thesis study, a functional reindeer earmark search application was created. The prototype can be used on Android and iOS-based devices. With the application the user can draw different deeds on the reindeer earmark with a finger or touch-screen pen. The application compares the reindeer earmark drawn by the user with comparison pictures and returns the corresponding reindeer earmarks suitable for the picture, from which the user chooses the right pictures. After this, the program returns the potential owner's information. In future the prototype can be used, for example, in digitizing the working methods of reindeer husbandry.

Key words mobile applications, programming, reindeer husbandry
Special remarks The thesis contains an introductory video.

SISÄLLYS

ALKUSANAT

Kiitos Paliskuntain yhdistykselle, että sain olla mukana mielenkiintoisessa projektissa. Erityiskiitos Aarre Jortikalle kehittävän palautteen antamisesta työn eri vaiheissa.

Kiitos Aku Kuntsille mentoroinnista ja auttamisesta sovelluksen kanssa. Kiitos myös ohjaajalleni Aku Kestille ja perheelleni, jotka olivat tukena opinnäytetyön teossa.

KÄYTETYT LYHENTEET JA TERMIT

android	mobiililaitteiden käyttöjärjestelmä
API	Application Programming Interface, ohjelmointirajapinta
backend	sovelluksen osa, joka toimii palvelimen puolella käyttäjältä näkymättömissä (Simmons 2023)
Dart	ohjelmointikieli
Flask	Pythonin verkkokehys
Flutter	mobiilisovelluskehys
frontend	sovelluksen osa, joka suoritetaan asiakaspuolella (Simmons 2023)
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla
iOS	mobiililaitteiden käyttöjärjestelmä
MSE	Mean Square Error, keskimääräinen neliövirhe kuvavertailussa
poromerkki	poron korvamerkki
REST	Representational State Transfer, palvelinpuolen rajapinnan arkkitehtuurimalli
resurssi	esimerkiksi tiedosto tai palvelu (Sanastokeskus 2012)
SOAP	Simple Object Access Protocol, tietoliikenneprotokolla
sovelluslogiikka	ohjelman toiminnan kannalta oleelliset osat sisältävä logiikka
teko	viilto poromerkissä
URI	Uniform Resource Identifier, internetissä olevan resurssin yksilöivä tunnus (Sanastokeskus 2012)

1 JOHDANTO

Opinnäytetyön aiheena on Poromerkin piirtäminen ja tunnistaminen mobiilisovelluksessa. Ollessani työharjoittelussa Frostbit-ohjelmistolaboratoriossa olin mukana luomassa Vasama-hankkeelle offline-mobiilisovellusta, jolla käyttäjä voi etsiä tietoja poromerkeistä ja niiden omistajista. Vasama-hanke on Euroopan aluekehitysrahaston (EAKR) rahoittama hanke, ja sen tarkoituksena on selvittää poronhoidon työmenetelmiä ja tuoda niiden ongelmakohtiin ratkaisuja uuden tutkimustiedon ja teknologian avulla (Työ- ja elinkeinoministeriö 2022). Alun perin ajatuksena oli, että olisin tähän sovellukseen liittyen tehnyt myös opinnäytetyöni, mutta koska sovelluksen kehittämisellä oli melko tiukka aikataulu, ei sitä otettu aiheekseni.

Vasama-hankkeessa mukana olevat Frostbitin työntekijät keskustelivat hankkeen toisen osapuolen Paliskuntain yhdistyksen kanssa ja sieltä nousi mielenkiinto testata piirtosovellusta, johon esimerkiksi poroerotuksessa mukana oleva käyttäjä voisi piirtää poromerkin ja sovellus etsisi vastaavan poromerkin omistajan tiedot. Päätettiin siis, että alan tehdä opinnäytetyönäni prototyyppiä tällaisesta sovelluksesta ja kokeilemaan, onko tällainen sovellus mahdollista toteuttaa. Aluksi tarkoitus oli, että toimeksiantajana olisi ollut Vasama-hanke, mutta hankkeen toimeksiantajaksi muuttui joulukuussa 2022 Paliskuntain yhdistys.

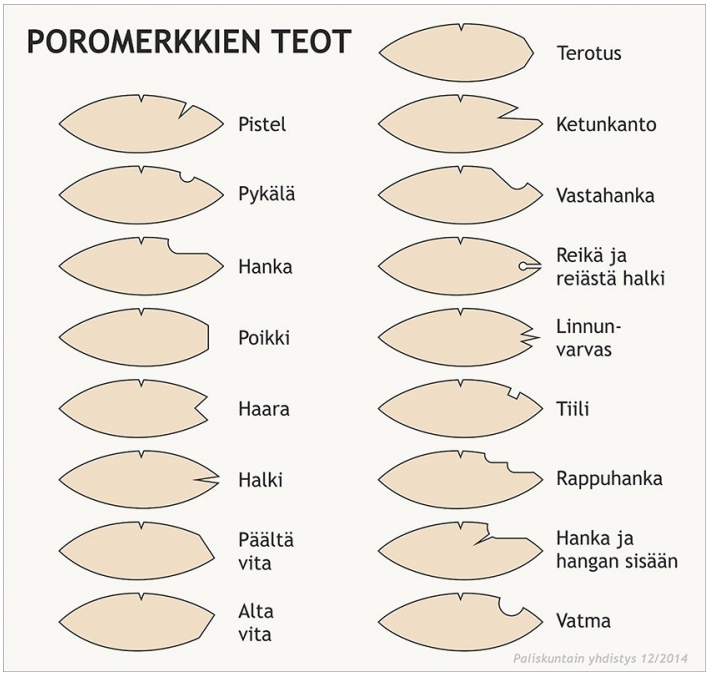
Opinnäytetyön tarkoituksena on luoda prototyyppi mobiilisovelluksesta, johon käyttäjä voi piirtää poromerkin ja etsiä piirtämänsä poromerkin omistajan tiedot. Opinnäytetyön tavoitteena on tuoda helpotusta porotalouden työskentelytapoihin ja selvittää, onko mahdollista kehittää piirtosovellus, joka tunnistaa käyttäjän piirtämät poromerkit.

Aihe on ajankohtainen, sillä digitalisoituminen vaikuttaa porotalouteen enenevässä määrin. Erityisesti nuoremman sukupolven poronhoitajilla digitaalinen tekniikka on jo luontevasti läsnä poronhoidon arjessa. Digitalisoitumisen kehittäminen ja tukeminen voi avata merkittävät kehitysnäkymät koko poronhoidon alalle. (Lapin yliopisto 2017, 2–3.)

2 PALISKUNTAIN YHDISTYKSEN ESITTELY JA POROMERKIT

Paliskuntain yhdistys toimii porotalouden ohjaus-, neuvonta- ja asiantuntijaorganisaationa. Yhdistyksen jäseniä ovat kaikki paliskunnat (54 kappaletta). Yhdistyksen tehtävät on säädetty poronhoitolaissa. Tehtävänä on toimia paliskuntien yhdyssiteenä, kehittää porotaloutta ja poronhoitoa, edistää poronhoidon tutkimusta, poronhoitoa koskevaa koetoimintaa ja poronjalostusta ja suorittaa muut sille määrätyt tehtävät. Yhdistys toteuttaa näitä tehtäviä muun muassa julkaisemalla Poromies-lehteä ja muuta tietomateriaalia, neuvomalla poronhoitoon liittyvissä kysymyksissä, järjestämällä koulutuksia, ylläpitämällä Kutuharjun koeporotarhaa ja toimimalla viranomaisena poromerkkiasioissa. (Paliskuntain yhdistys 2023a.)

Kaikilla poroilla on omistaja, ja kaikki porot merkitään poromerkillä, joka on virallisesti rekisteröity poron omistajalle. Yhdellä omistajalla voi olla useita poromerkkejä. Poromerkki koostuu viilloista eli teoista. Jokainen poromerkki on yhdistelmä erilaisia tekoja. Teot voidaan kuvata sekä piirroksin että sanallisesti (Kuvio 1). Poromerkki luetaan poron oikean korvan kärjestä tyveen päin edeten. Paliskuntain yhdistys ylläpitää poromerkkirekisteriä, jossa on noin 12 000 eri poromerkkiä. (Paliskuntainyhdistys 2023b.)



Kuvio 1. Poromerkkien teot (Paliskuntain yhdistys 2023b)

3 PYTHON BACKEND ALUSTANA

Backend on se sovelluksen osa, joka toimii palvelimen puolella käyttäjältä näkymättömissä. Backend sisältää sovelluslogiikkaa sekä tarvittaessa yhteyden tietokantaan. (Simmons 2023.) Hyviä backend-ohjelmointikieliä on useita. Valittu kieli vaikuttaa siihen, kuinka hyvin sovellus toimii, kuinka helppoa se on päivittää ja millaista tietokannan hallinta on. Ennen backendin valintaa tulee ymmärtää, mitä tuleva sovellus vaatii, miten sitä käytetään ja miten ohjelmointikielen valinta vaikuttaa käyttöliittymän käyttäjiin. Suosittuja backend-ohjelmointikieliä ovat esimerkiksi PHP, Python ja Java. (Gomez 2023.)

Tähän opinnäytetyöhön backend-ohjelmointikieleksi valikoitui Python sen helppokäyttöisyyden ja laajojen ominaisuuksien vuoksi. Se oli myös tutuin ohjelmointikieli opinnäytetyön tekijälle. Python on myös yleisimmin käytetty ohjelmointikieli koneoppimisen sovelluksissa (Gomez 2023). Python on avoimen lähdekoodin olio-ohjelmointikieli (Python 2023). Python ei ole alustariippuvainen, ja se tukee useita erilaisia järjestelmiä (Gomez 2023). Se on suosittu sen sisäänrakennetun tietorakenteiden, yksinkertaisuuden ja nopeuden vuoksi. Python tukee erilaisia kirjastoja ja moduuleja, jotka auttavat ohjelmoinnissa. (Python 2023.) Se on myös tulkattava kieli, eli koodi kääntyy automaattisesti ennen ohjelman suorittamista. Pythonia käytetään erityisesti verkko-ohjelmoinnissa. (Jakonen 2021, 7.) Se ei ole ehkä paras valinta mobiilisovellusten kehittämiseen, koska se saattaa olla joiltain toiminnoiltaan hieman hidas, mutta monet ohjelmoijat käyttävät sitä varsinkin prototyyppien rakentamiseen (Gomez 2023).

3.1 REST API ja Pythonin Flask

Ohjelmointirajapinta (API) on rajapinta, jonka välityksellä verkko-ohjelmiston eri osat keskustelevat keskenään. Ohjelmointirajapintoja on kahdenlaisia: palvelin- ja asiakasrajapinnat. Palvelinrajapinnassa palvelin tarjoaa yhden tai useamman resurssin (URI), jonka avulla asiakas voi olla vuorovaikutuksessa palvelimen kanssa. Verkkorajapintojen kyselyt tehdään HTTP-protokollan avulla. HTTP-protokollassa on useita erilaisia metodeja, mutta niistä tärkeimmät ovat GET-, POST-, PUT- ja DELETE-metodit. GET-metodilla pyydetään tietoa resurssista, POST-metodilla lähetetään dataa resurssille, PUT-metodilla tehdään muutoksia

resurssien dataan ja DELETE-metodilla voidaan poistaa resurssien dataa. HTTP-pyyntöjen onnistumisesta kertovat erilaiset tilakoodit, kuten 200, joka tarkoittaa, että pyyntö on onnistunut. (Jakonen 2021, 2–3.)

Yksi suosittu API on REST. REST on palvelinpuolen rajapinnan arkkitehtuurimalli, jonka tarkoituksena on parantaa rajapintojen suorituskykyä, lisätä luotettavuutta ja skaalautuvuutta sekä yksinkertaistaa niiden toteutusta. REST-rajapinnoissa URI-merkkijonot toimivat resurssien yksilöintinä. (Jakonen 2021, 3, 5.) REST-sovellusliittymien yksi etu on, että ne tarjoavat paljon joustavuutta. REST pystyy käsittelemään monenlaisia pyyntöjä ja palauttamaan erilaisia tietomuotoja, koska sen tietoja ei ole sidottu resursseihin tai menetelmiin. (MuleSoft 2023.)

Toinen suosittu API on SOAP, joka käyttää XML:ää käyttöjärjestelmissä käynnissä olevien prosessien todentamiseen, valtuutukseen ja viestintään. SOAP antaa mahdollisuuden kutsua ja vastaanottaa vastauksia ohjelmointikielestä tai alustasta riippumatta. (Kulkarni 2021.) Toisin kuin REST, SOAP on protokolla ja sitä käytetään enemmän korkeamman turvallisuustason sovelluksissa. SOAP käyttää URI:n sijasta palvelurajapintaa ja on rajoittunut käyttämään vain XML:ää. (Upwork 2021.) Tähän opinnäytetyöhön valikoitui REST API sen joustavuuden vuoksi.

Pythonissa suosituimpia verkkokehyksiä ovat Django ja Flask. Django sopii parhaiten suurten ja monimutkaisten verkkosovellusten kehittämiseen. Flask taas on kevyt ja hyvin laajennettavissa oleva, joten se soveltuu parhaiten pienten sovellusten käyttöön. (InterviewBit 2022.) Siitä huolimatta, että Flask on kevyt, sillä on paljon ominaisuuksia, joiden avulla voidaan toteuttaa hyvin laajoja ohjelmistoja. Flaskia voi laajentaa useilla lisäosilla ja kirjastoilla. Vaikka Flaskia ei ole varsinaisesti suunniteltu REST-rajapintojen kehittämiseen, se täyttää REST-mallin vähimmäisvaatimuksen ja näin ollen on käyttökelpoinen myös REST API -rajapintana. (Jakonen 2021, 8.) Tämän opinnäytetyön verkkoviitekehykseksi valikoitui Flask, koska se on helppokäyttöinen, vaatii vähän koodia ja sopii näin pienelle sovellukselle erinomaisesti.

3.2 Kuvavertailu Pythonilla

Python on erittäin hyvä ohjelmointikieli kuvien vertailemiseen, koska se sisältää useita kuvien prosessointikirjastoja. Kuvia voi vertailla useilla eri metodeilla, ja niihin on olemassa valmiita kirjastoja. Useimmiten kuvia vertaillaan niiden pikselien eri ominaisuuksien vertailulla. Isommissa kuvavertailuissa käytetään nykyään yhä enemmän tekoälypohjaisia kirjastoja. Seuraavaksi käydään lyhyesti läpi kirjastoja, joita kokeilin sovelluksen tekovaiheessa. DeepImageSearch-kirjasto valikoitui lopulta käytettäväksi kirjastoksi tässä työssä, koska testauksissa se tunnisti parhaiten poromerkkien eroavaisuudet ja näin antoi tarkimmat vastaavuudet piirretylle merkille.

Pythonin **ImageHash-kirjasto** vertailee kuvia kuvanhajautuseron mukaan (Rendyk 2022). Kirjaston kuvanhajautusalgoritmeja ovat keskimääräinen hajautus, havaintohajautus, gradienttihajautus ja wavelet-hajautus. Nämä kaikki algoritmit vertailevat kahden kuvan pikseleiden eroja eri tekniikoilla. (Petrov 2016.)

Esimerkiksi keskimääräisen hajautuseron saa selville, kun kaksi kuvaa muutetaan ensin harmaansävyiseksi, kuvakoot pienennetään esimerkiksi 8 x 8 pikseliin. Tämän jälkeen ohjelma laskee 64 pikselin keskiarvon ja tarkastaa kaikki kuvan pikselit, ovatko ne suurempia kuin keskiarvo. ImageHash-ero on eri arvojen lukumäärä kahden kuvan välillä. Mitä lähempänä luku on nollaa, sitä samankaltaisempia kuvat ovat. (Rendyk 2022.)

OpenCV-kirjastolla voi suorittaa kuvankäsittely- ja tekoälytehtäviä erilaisilla menetelmillä. Tässä sovelluksessa kokeilin seuraavia OpenCV:n tarjoamia kuvavertailu vaihtoehtoja: MSE ja Template Matching. MSE vertaa kahden kuvan pikseliarvojen keskimääräistä neliövirhettä. Mitä pienempi on kuvien keskimääräinen neliövirhearvo, sitä samankaltaisempia kuvat ovat keskenään. (Khan 2022.) Template Matchingissa etsitään tekoälyn avulla pienemmän mallikuvan sijainti suuremmasta kuvasta vertailemalla kuvien pikseleitä (Great Learning 2022).

SentenceTransformer-kirjasto vertailee kuvia vektoreiden ja tekoälyn avulla (PyPi 2022). Kirjastolla on laaja kokoelma valmiiksi koulutettuja malleja, jotka mahdollistavat erilaiset tehtävät (Chiusano 2022). Kuvan tunnistuksessa kuvat

muutetaan vektoreiksi ja kirjasto etsii ne tiheysalueet kuvasta, jotka ovat vastaavia verrokkikuvan kanssa. Kuville annetaan pistemäärä 0–1,0, jossa 1,0 tarkoittaa, että kuvat ovat identtiset keskenään. (PyPi 2022.) Kirjastolla voi vertailla myös esimerkiksi tekstien samankaltaisuutta tai upotuksia lauseessa (Chiusano 2022).

DeepImageSearch-kirjasto on tekoälypohjainen kuvahakukone. Se sisältää syväsiirto-oppimisominaisuuksia ja perustuu puupohjaiseen vektorisoituun hakutekniikkaan. (PyPi 2021.) Sen ominaisuuksia on muun muassa nopea ja tarkka hakutulos. Sitä suositellaan käytettäväksi erityisesti Python-pohjaisissa sovelluksissa tai API:ssa. Hyviä käyttöesimerkkejä ovat esimerkiksi verkkokaupat, sosiaalinen media tai muut alustat, joissa halutaan toteuttaa kuvahakua. Kirjasto on yhteensopiva Windows- ja Linux-järjestelmän kanssa. (TechyNilesh 2021.)

Kirjasto sisältää kolme tärkeää luokkaa: LoadData, Index ja SearchImage. LoadDatalla ladataan kuvatiedostot esimerkiksi kansiosta tai CSV-tiedostosta. Jotta kuvien haku olisi nopeampaa, kirjasto tallentaa vertailukuvien metatiedot paikalliseen [meta-data-files/]-kansioon. Tämä tapahtuu Index-luokan avulla. Itse kuvien vertailu tapahtuu SearchImage-luokan get_similar_images-metodin avulla. Halutessa haetut vastaavuudet voidaan näyttää myös suoraan näytöllä SearchImage().plot_similar_images -metodilla. (TechyNilesh 2021.)

4 FLUTTER FRONTEND ALUSTANA

Frontend tarkoittaa sitä sovelluksen osaa, joka suoritetaan asiakaspuolella, kuten esimerkiksi web-selain. Frontend sisältää käyttäjän näkemän käyttöliittymän ja siihen liittyvän sovelluslogiikan. (Simmons 2023.) On olemassa useita erilaisia mobiilisovelluskehyksiä. Mobiilikehys on ohjelmistojen luontialusta, joka sisältää esimerkiksi työkaluja, ohjelmistoja, kääntäjiä ja ohjelmointirajapintoja. Nykyiset mobiilisovelluskehykset tarjoavat useita sisäänrakennettuja etuja, kuten nopeus, tehokkuus ja virheettömyys. Suosittuja mobiilisovelluskehyksiä ovat esimerkiksi React Native, Flutter, Xamarin ja Swiftic. (Technostacks 2023.) Tähän opinnäytetyöhön mobiilisovelluskehykseksi valikoitui Flutter sen nopeuden, monipuolisuuden ja helppouden vuoksi.

Flutter on Googlen kehittämä avoimen lähdekoodin ohjelmistopaketti, jolla voi rakentaa moninaisia sovelluksia helposti ja nopeasti (Flutter 2023a). Flutterissa ohjelmointikielenä käytetään Dartia, joka on olio-orientoitunut ohjelmointikieli. Dart on suunniteltu erityisesti käyttöliittymän rakentamisen tarpeisiin. (Dart 2023.) Flutter toimii kaikissa laitteissa, sillä sen koodi käännetään ARM- tai Intel-konekoodiksi sekä JavaScriptiksi automaattisesti (Flutter 2023a).

Flutterilla tehty käyttöliittymä rakentuu erilaisista widgeteistä. Erilaisia widgettejä on todella paljon, ja niitä voi myös itse luoda. Flutterilla on esimerkiksi materiaalin suunnitteluun käytettäviä widgettejä, joilla voi rakentaa sovelluksia. Näitä on esimerkiksi MaterialApp, jonka sisällä on esimerkiksi Navigator-widgetin, jolla eri näyttöjä voi reitittää keskenään. Flutterin peruswidgettejä ovat muun muassa Text (teksti), Row (rivi), Colum (sarake) ja Container (sisältö). (Flutter 2023d.) Seuraavaksi käydään läpi tämän opinnäytetyön prototyyppiin käytetyt tärkeimmät widgetit ja niiden toiminta sekä Flutterin Dio-paketti.

CustomPaint-widgetin avulla voidaan luoda kangas, jolle voi piirtää. Piirtäminen tapahtuu suorakulmion sisällä olevia koordinaatteja hyödyntäen. Suorakulmio alkaa origosta ja kattaa tietyn kokoisen alueen. (Flutter 2023b.) CustomPaintin alaluokka on CustomPainter, jolla määritetään piirtämismenetelmät, esimerkiksi kynän ominaisuudet tai minkä koordinaattien väliin piirretään esimerkiksi viiva tai kuvio.

GestureDetector-widget tunnistaa erilaisia eleitä, kuten napsautuksen, koskettuksen ja vetämisen (Flutter 2023c). Kun halutaan tunnistaa käyttäjän kosketus ruutuun, käytetään OnPanDown-, onPanUpdate- ja OnPanEnd-ominaisuuksia ja määritetään niille haluttu toiminta, kuten koordinaattipisteiden koonti taulukkoon.

Screenshot-widgetillä voidaan tallentaa haluttu widget/widgetit kuvina. Screenshot-widget on hyvin helppo ja yksinkertainen käyttää. Sillä voi kuvakaapata myös widgetit, joita ei näytetä näytöllä. (Dreamsoft Innovations 2023.)

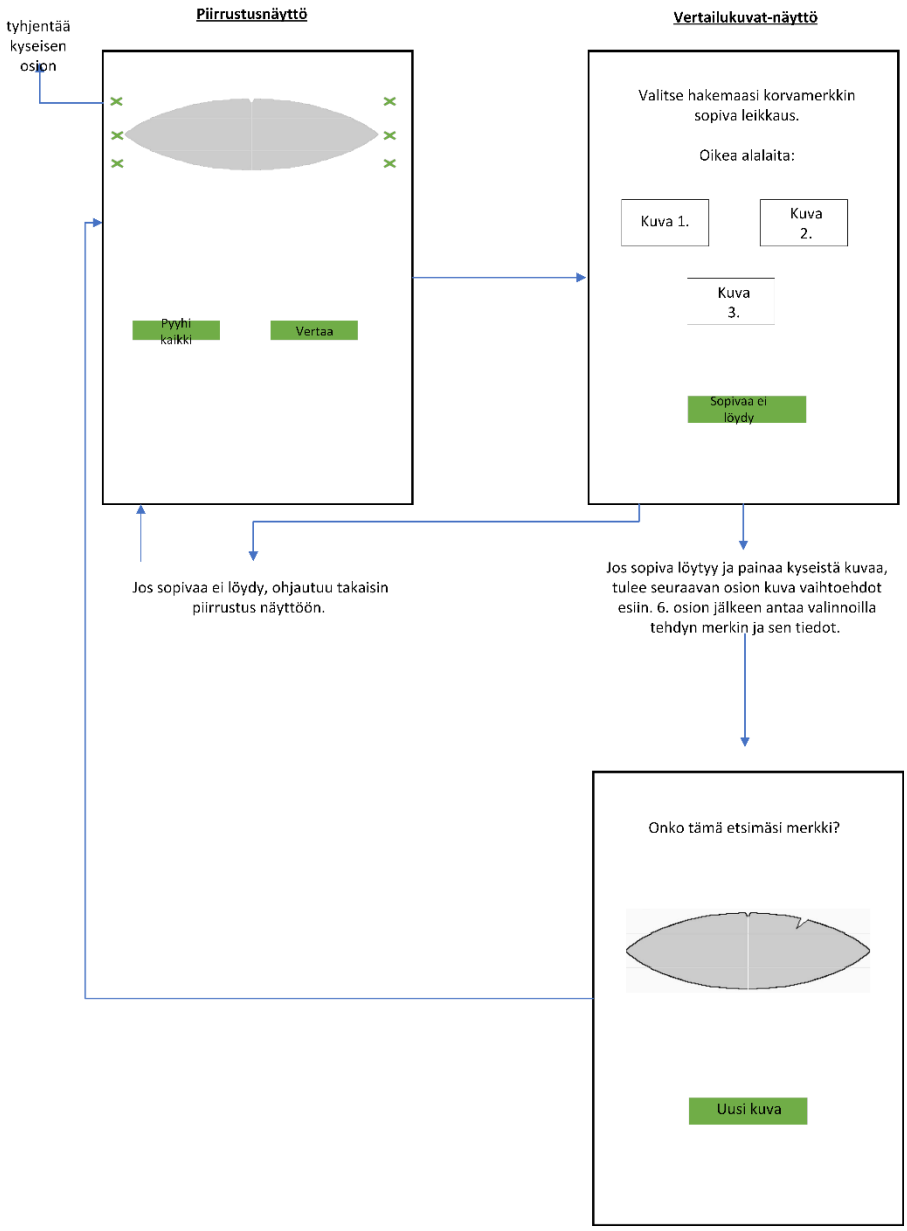
ListViewBuilder on widgetti, jonka avulla voi luoda luettelon kirjoittamatta koodia uudelleen ja uudelleen. Widget luo automaattisesti vieritettävän lineaarisen joukon sen child-widgettejä, esimerkiksi Card-widgettejä tai Image-widgettejä. Widgettiin määritellään itemCount, joka määrittää, kuinka monta kertaa itemBuilder luo widgetin uudelleen. (GeeksforGeeks 2022.)

Dio-paketti on tehokas HTTP-asiakas Flutterille. Dio tukee muun muassa pyyntöjen peruuttamista, tiedoston lataamista ja aikakatkaisua. Diolla on useita ominaisuuksia, joiden avulla voidaan lähettää erilaisia kutsuja API:lle. (Flutter.cn 2023.)

5 MOBIILISOVELLUKSEN SUUNNITTELU JA TOTEUTTAMINEN

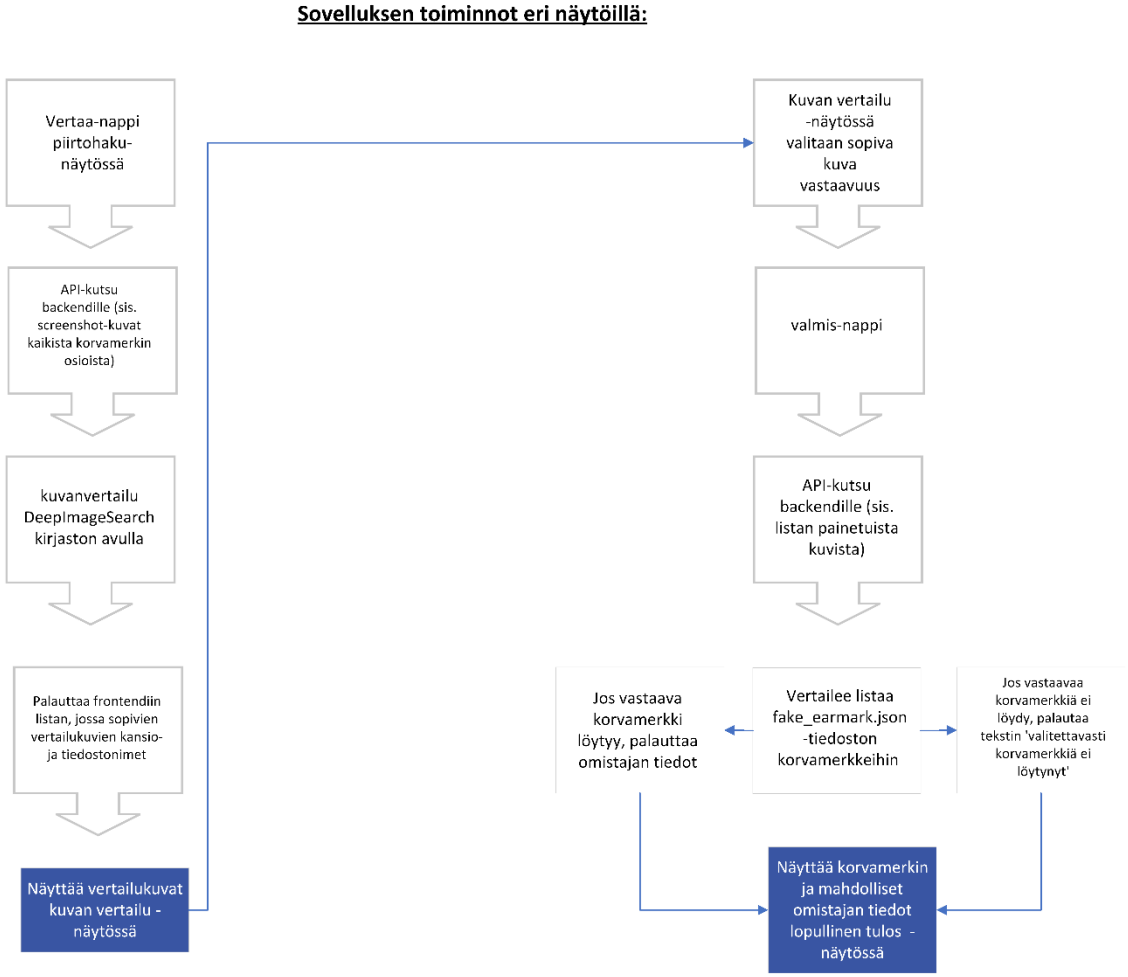
Opinnäytetyön tarkoituksena on kehittää piirtohakusovellus, jolla voidaan piirtää poromerkki ja hakea sen avulla poron omistajan tiedot. Tämäntyyppistä poromerkkihakusovellusta ei ole vielä olemassa, joissa on mukana piirto-ominaisuus. Paliskuntain yhdistyksellä olemassa olevaa poromerkkirekisteriä voi käyttää webselaimella sekä mobiililaitteella. Rekisteristä voi etsiä poromerkin perusteella omistajan tiedot, mutta jokaisen osion teot valitaan valmiista katalogista, ei piirtämällä. Myös Norjasta löytyy vastaavanlainen rekisteri ja siihen myös mobiilisovellus. Vasama-hankkeen myötä on tarkoituksena kehittää prototyyppi offline-mobiilisovelluksesta, jota voi käyttää myös paikoissa, joissa ei välttämättä puhelinverkkoyhteys toimi.

Sain melko vapaat kädet, miten lähteä toteuttamaan sovellusta. Sovelluksen backend luotiin Pythonilla, koska se oli itselleni tutuin ja on helppokäyttöinen. Frontend luotiin Flutterilla. Flutteria käytettiin, koska se käy sekä Android- että iOS-laitteille. Aikatauluksi sovittiin, että opinnäytetyö on valmis viimeistään huhtikuussa 2023. Koska kyseessä on prototyyppi, en panostanut ulkonäöllisiin piirteisiin kovinkaan paljon, vaan keskityin siihen, että sovelluksen toiminta on kunnossa. Piirsin aluksi itselleni hahmotelman siitä, miltä sovellus tulisi näyttämään (Kuvio 2) ja lähdin sen jälkeen työstämään sovellusta.



Kuvio 2. Hahmotelma sovelluksesta

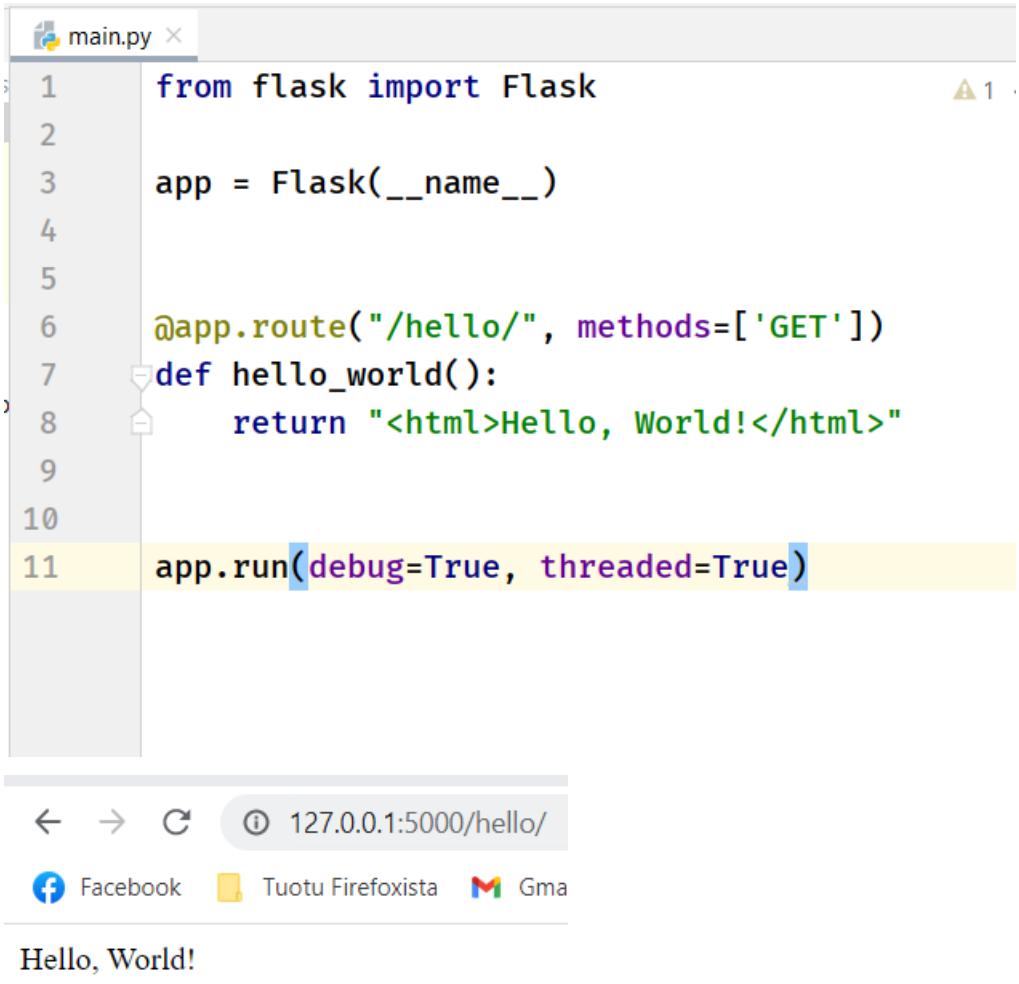
Lopullisessa sovelluksessa on neljä eri näyttöä: aloitus/info-, piirtohaku-, kuvien vertailutuloksien valinta- ja lopullinen hakutulosnäyttö. Sovelluksen tärkeimmät toiminnot aktivoituvat piirtohaku- ja kuvien vertailutuloksien valinta -näytöissä (Kuvio 3).



Kuvio 3. Sovelluksen ominaisuudet ja toiminnot eri näytöissä

5.1 Backendin toteutus Flaskilla

Sovelluksen tekeminen aloitettiin backendistä. Aluksi luotiin Python-projekti backend_earmark_drawing_app ja siihen Flask. Flaskin toimivuus testattiin hello_world()-funktiolla, jossa käytettiin GET-metodia (Kuvio 4).



Kuvio 4. Flaskin luominen ja testaus hello_world()-funktiolla

Testauksen jälkeen luotiin resurssi /data ja testattiin vastaanottaako se kuvan Flutterilta. Seuraavassa kuviossa 5 on kuvattu koodi, jolla edellä mainittu testaus suoritettiin.

```python
@app.route("/data", methods=['POST'])
def post_image():
    #vastaanottaa kuva tiedoston flutterilta
    f = request.get_data()

    print(
        f
    )

    return 'ok'
```

Kuvio 5. Resurssin testauksessa käytetty koodi

Kun testaukset oli suoritettu, alettiin rakentaa sovelluksen oikeita resursseja. Luotiin palvelimen resurssit /images ja /earmarks (Kuvio 6) ja niiden toiminnot.

```python
@app.route("/images", methods=['POST'])
def post_image():
    #vastaanottaa kuva tiedoston flutterilta
    print ('vastaanotti tiedoston')
    dataList = []

    f = request.get_data()
    flutterList = literal_eval(f.decode('utf-8'))

    imagePath = flutterList[0]['path']
```

```python
@app.route("/earmarks", methods=['POST'])
def post_earmarkDetails():



    # flutterEarmarkData = [{"bottomEnd":"j","bo

    flutterEarmarkData = []
    endOfEarmarkPath = []
    answer = ''
    f = request.get_data()
```

Kuvio 6. Palvelimen resurssit

Lopullisessa sovelluksessa resurssi /images vastaanottaa sovelluksesta listan, jossa on kunkin kuuden poromerkkikuvaosion kansion nimi sekä kuva bittijonona. Ennen kuvavertailua kuvia täytyy muuttaa. Ensin kuvan bitit muutetaan kuvaksi ja niiden koko muutetaan vastaamaan vertailukuvien kokoa. Listan kuvakansion nimeä käytetään apuna määritettäessä oikeat mitat kullekin osiolle. Lisäksi käytetään Canny-funktiota, joka tunnistaa kuvan reunat (Kuvio 7).

Kuvio 7. Kuva Canny-funktion jälkeen

Tämän jälkeen käyttäjän piirtämä poromerkkiosio tallennetaan Canny-muotoisena. Sitten kuvaa vertaillaan compareImages-funktiossa DeepImageSearch-kirjaston avulla 346:een ohjelmalle aiemmin syötettyyn poromerkkiosion vertailukuvaan. Lopulta resurssi palauttaa takaisin sovellukselle listan, jossa on viiden parhaan kuvavastaavuuden kansion ja kuvatiedoston nimi.

Resurssi /earmarks vastaanottaa sovelluksesta listan, jossa on niiden kuvien kansiot ja tiedostonimet, joita käyttäjä painaa sovelluksen kuvavertailun tulokset-osiossa. Lista käydään läpi ja sitä verrataan fake-earmark.json-tiedoston (Kuvio 8) poromerkkitietoihin, josta saadaan poromerkin omistajan tiedot. Paliskuntain yhdistyksellä on olemassa oleva poromerkkien tietokanta, mutta koska tämä sovellus on prototyyppi, ei sitä yhdistetä tässä vaiheessa tietokantaan ollenkaan. Tämän vuoksi käytetään toimivuuden testaamiseksi tiedostoa, jossa on keksityt poromerkit ja niiden omistajat.

Kuvio 8. Ote fake_earmarks.json-tiedostosta

5.2 Kuvavertailu ja CompareImages-funktio

Sovelluksen ohjelmoinnin aikana testattiin useita eri Pythonin kuvavertailukirjastoja. Testit osista kuvantunnistusmenetelmistä on nähtävillä liitteenä (Liite 1). Ongelmaksi osoittautui se, että vertailukuvat olivat niin samankaltaisia toistensa kanssa, että esimerkiksi perinteiset pikselin, histogrammin tai vektorien tunnistamistekniikat eivät antaneet luotettavia tuloksia. Testattiin myös mahdollisuutta, että ohjelma tunnistaisi ainoastaan viiltojen muodot (ympyrä, kolmio jne.) ja sen avulla muodostaisi poromerkin. Tässä kuitenkin ongelmana oli, että muutamat teot ovat samanmuotoisia keskenään, vain erikokoisia ja osa teoista taas on sen verran erikoisen muotoisia, että muotojen tunnistus ei toimisi. Tekoälykirjastoista OpenCv:n Template Matching -kirjasto toimi välttävästi, kun koneelle syötettiin malliksi tekojen osioiden kuvat. Onneksi lopulta löytyi DeepImageSearch-kirjasto, joka testien perusteella toimi melko hyvin ja valikoitui lopulta tähän sovellukseen käytettäväksi.

Kun sopiva kuvavertailukirjasto oli löytynyt, luotiin backendille uusi tiedosto ja sinne funktio compareImages. Ensimmäisellä käyttökerralla ladattiin haluttujen vertailukuvien metatiedot Index-metodin avulla paikalliseen kansioon (Kuvio 9).

```
def compareImages(imagePath):

    # ladataan kuvat, joihin vertaillaan

    image_list = LoadData().from_folder([f'{imagePath}'])


    image_list = LoadData().from_folder(['images/teko1',
    'images/teko2','images/teko3',
    'images/teko4',
    'images/teko5','images/teko6',
    ])



        # jos uusia kuvakansioita, pitää ladata ensin ja hyväksyä tiedoston uudelleen kirjoittaminen:
    Index(image_list).start()
```

Kuvio 9. Vertailukuvien metatietojen lataaminen

Tämän jälkeen kirjaston metodeja käyttäen etsittiin kuvavastaavuudet poromerkkien osioille. Kirjaston avulla oli mahdollista itse määritellä, kuinka monta parasta vastaavuutta se antaa kullekin osiolle. Päädyttiin etsimään kaikille kuville viisi parasta vastaavuutta, joista funktio palauttaa kuvan kansion ja kuvatiedoston nimen listamuotoisena frontendiin (Kuvio 10).

```
flutter: [{folder: teko5, path: -.png}, {folder: teko5, path: p.png}, {folder: teko5, path: o.pn
g}, {folder: teko5, path: q.png}, {folder: teko5, path: qaa.png}]
```

Kuvio 10. Backendin frontendille lähettämä lista viidestä vastaavasta kuvasta

5.3 Aloitus/infonäyttö

Aloitus/infonäyttö on todella yksinkertainen. Näyttöä varten luotiin MainDrawer()-widget, jossa näkyy sovelluksen sisällys (Kuvio 11).

Kuvio 11. Maindrawer()-widget

Lisäksi luotiin Container-widget, jonka sisällä ovat aloitussivun info sekä nappi, jolla voi siirtyä piirtohaku-näyttöön. Kuviossa 12 on esitetty, miltä aloitussivun näyttö näyttää emulaattorissa.

Kuvio 12. Aloitus/infonäyttö

5.4 Piirtohakunäyttö

Piirtohakunäytöstä löytyy valmis tyhjä poromerkki, joka on jaettu kuuteen osaan. Jako osiin on tehty sen vuoksi, että olemassa olevassa tietokannassa poromerkki on myös osissa. Erilaisia poromerkkejä on olemassa yli 12 000, mutta yksittäisessä osiossa vaihtoehtoja on vain noin 60 kappaletta, mikä helpottaa ja nopeuttaa poromerkkien etsintää. Poromerkkiin piirretään etsittävät teot joko sormella tai kosketusnäyttökynällä. Poromerkkiä saa suurennettua piirtämisen helpottamiseksi hiirellä tai sormilla. Näytössä on myös kaksi nappia, joista toisella voi pyyhkiä poromerkin piirrokset ja toinen nappi käynnistää kuvavertailun. Osiot on myös mahdollista pyyhkiä yksi osio kerrallaan vieressä olevasta rastista. I-napista pääsee lukemaan ohjeet sovelluksen käyttämiseen.

Piirtohakunäytön tärkein ominaisuus on poromerkin viiltojen piirtomahdollisuus. Siinä tärkeimmät widgetit ovat GestureDetector ja CustomPaint. GestureDetector tunnistaa näytön liikkeet käyttäjän piirtäessä (Kuvio 13).

```
child: GestureDetector(
  onPanDown: (details) {
    setState(() {
      points.add(details.localPosition);
    });
  },
  onPanUpdate: (details) {
    setState(() {
      points.add(details.localPosition);
    });
  },
  onPanEnd: (details) {
    setState(() {
      points.add(null);
    });
  },
  child: CustomPaint(
```

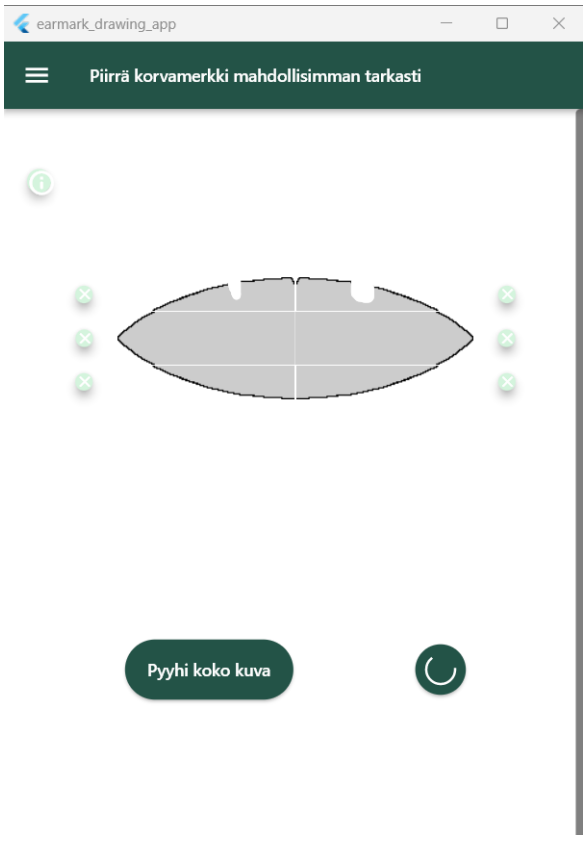
Kuvio 13. GestureDetector-widget

CustomPaint-widget mahdollistaa itse piirtämisen ja antaa ’kynälle’ tarvittavat ominaisuudet. Kuvion 14 koodissa on määritelty ’kynän’ asetukset.

```
class MyCustomPainter extends CustomPainter {
  List<Offset?> points;

  MyCustomPainter({required this.points});
  @override
  void paint(Canvas canvas, Size size) {
    Paint canvasBackround = Paint()..color = Colors.amber;
    //canvasBackround koko:

    Rect rect = Rect.fromLTWH(0, 0, size.width, size.height);
    canvas.drawRect(rect, canvasBackround);

    Paint paint = Paint();
    paint.color = Colors.black;
    paint.strokeWidth = 2.0; //viivan paksuus
    paint.isAntiAlias = true; //viiva selkeämpi
```

Kuvio 14. CustomPaint-widgetin ’kynän’ asetukset koodissa

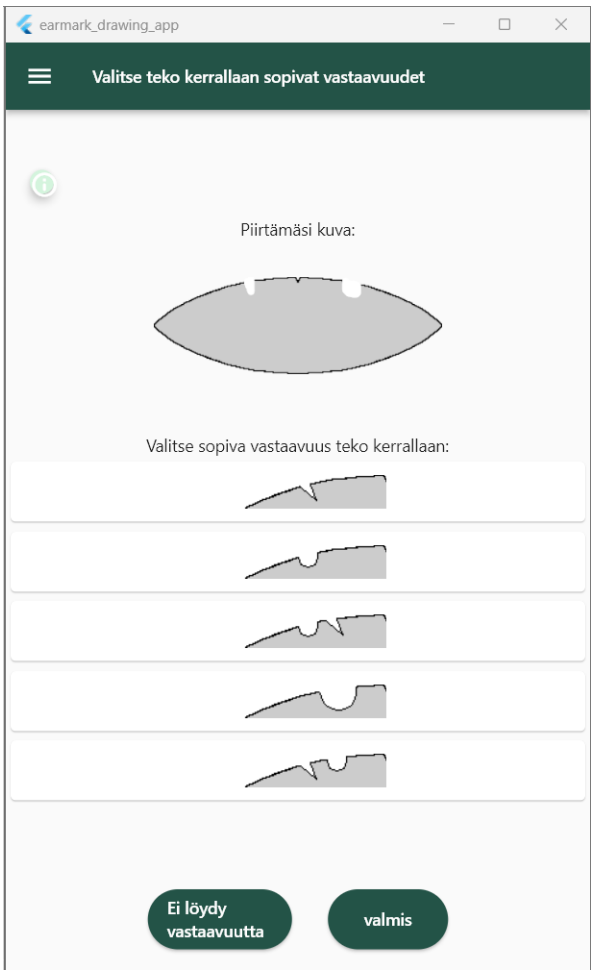
Kun käyttäjä on piirtänyt poromerkin teot (Kuvio 15), hän painaa vertaa-nappia, jolloin jokaisesta poromerkin osiosta otetaan näyttökuva Screenshot-widgetillä. Otetut näyttökuvat lähetetään bitteinä backendille vertailuun Dio-paketin avulla.



Kuvio 15. Käyttäjän piirtämät teot piirtohaku-näytössä

5.5 Kuvavertailun tuloksien valinta -näyttö

Kuvavertailun tuloksien valinta -näytöllä (Kuvio 16) on ylhäällä käyttäjän piirtämä poromerkki, jota on vertailtu. Sen alla ovat jokaisesta osiosta backendiltä tulleet viisi parasta vastaavuutta. Käyttäjän tulee valita jokaisesta osiosta sopiva vastaavuus. Mikäli vastaavuutta ei löydy, painetaan Ei löydy vastaavuutta -nappia, joka palauttaa käyttäjän takaisin piirtohakunäytölle. Käyttäjä voi muokata haluamaansa osiota ja yrittää sen jälkeen uudelleen. Mikäli osion vastaavuus löytyy, painetaan tätä kuvaa ja näytölle ilmestyy seuraavan osion parhaat vastaavuudet. Kun kaikista osioista on löytyneet vastaavuudet, painetaan valmis-nappia, joka lähettää käyttäjän valitsemien kuvien tiedot backendille Dio-paketin avulla ja siirtää käyttäjän seuraavalle näytölle.

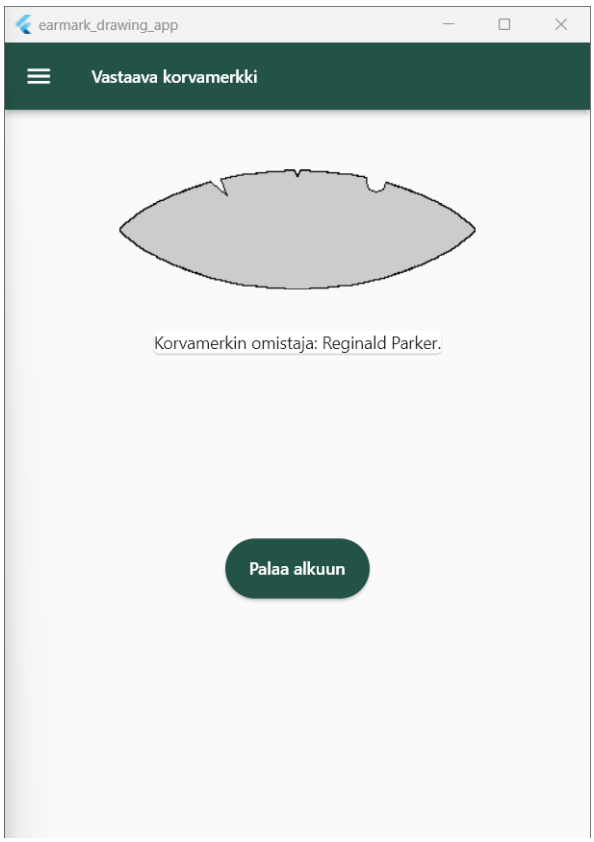


Kuvio 16. Vertailukuvien valinta näytöllä

Koska kuvien vertailu tapahtuu backendillä, tässä näytössä ei tarvinnut käyttää erikoisempia widgettejä. Backendiltä palautuvat kuvatiedot tulevat listana, joka käydään läpi ListViewBuilder-widgetin avulla.

5.6 Lopullinen tulos -näyttö

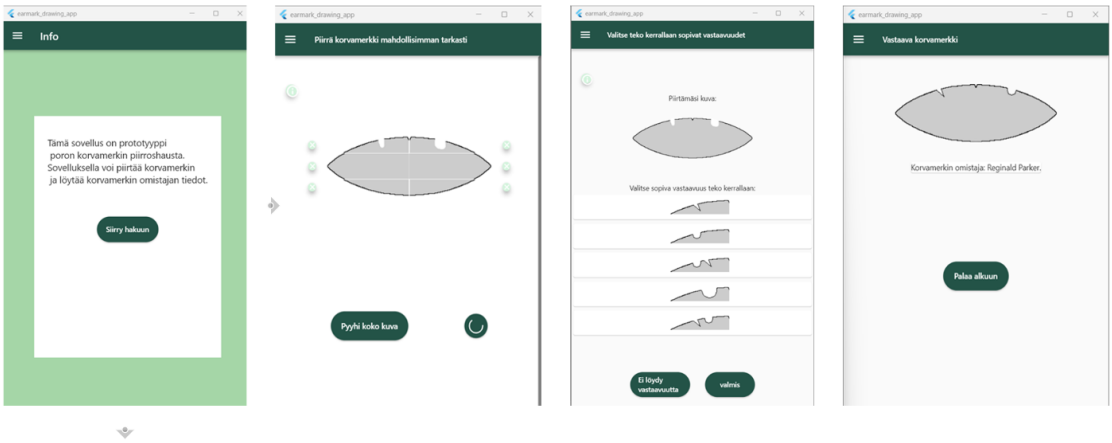
Lopullisessa tulos -näytössä näkyy käyttäjän valinnoista muodostunut poromerkki ja sen omistajan tiedot, mikäli tämä poromerkki löytyy backendin käyttämästä fake_earmarks.json-tiedostosta (Kuvio 17). Mikäli omistajan tietoja ei löydy, näytölle tulee teksti: ’Valitettavasti hakemaasi poromerkkiä ei löytynyt’. Lisäksi on nappi, josta pääsee takaisin piirtohaku-näyttöön. Myöskään tässä näytössä ei tarvinnut käyttää erikoisempia widgettejä.



Kuvio 17. Lopullinen tulos -näyttö

5.7 Valmis sovellus

Sovellus (Kuvio 18) valmistui aikataulun mukaisesti. Se esitettiin toimeksiantajalle, joka oli tyytyväinen sovelluksen ominaisuuksiin ja toimivuuteen. Ainoastaan kuvavertailu voisi olla heidän mielestään nopeampi. DeepImageSearch-kirjasto, jota käytetään kuvavertailuun tallentaa vertailukuvien metatiedot paikalliseen kansioon ja käy kaikki kansion kuvat (346 kuvaa) läpi jokaisen poromerkkiosion kohdalla, vaikka yhtä osiota vastaa vain noin 60 kuvaa. Yritin, jos olisi mahdollista rajata käytävien vertailukuvien määrää ja näin nopeuttaa kuvavertailua, mutta se ei ollut mahdollista, koska vertailukuvien metatiedot ovat tallennettuna binäärimuodossa yhteen tiedostoon. Nyt kaikkien kuuden poromerkkiosion kuvien vertailu kestää noin minuutin verran. Ulkonäöllisiin asioihin ei tässä prototyypissä panostettu kovinkaan paljon, sillä todennäköisesti sovellusta ei käytetä tässä muodossaan tulevaisuudessa, vaan siitä mahdollisesti yhdistetään osia jo olemassa oleviin/kehitettäviin sovelluksiin.



Kuvio 18. Valmis sovellus

6 POHDINTA

Opinnäytetyön tekeminen oli mielenkiintoinen oppimisprosessi. Sovelluksen luominen itse alusta lähtien oli minulle aivan uutta ja Flutter oli minulle myös kokonaan uusi ohjelmistopaketti, vaikka ehdin työharjoitteluni aikana siihen jonkin verran tutustua. Ohjelmoinnista minulla ei ollut aiemmin muuta kokemusta, kuin mitä koulussa on pieniä tehtäviä tehty eri ohjelmointikielillä.

Koska tällaista sovellusta ei ole aiemmin tehty, vaati erityisesti kuvavertailu paljon testailuja eri kirjastoilla. Välillä meinasi epätoivo iskeä, että mikään kuvavertailu ei anna tarpeeksi hyviä tuloksia ja sovellus ei toimi ollenkaan. Onneksi sitten lopulta löytyi tarpeeksi hyvä kirjasto kuvavertailuun. Tulevaisuudessa kuitenkin, mikäli sovellus halutaan ottaa käyttöön, voisi olla järkevämpää kouluttaa oma tekoäly juuri poromerkkejä varten, eikä käyttää valmista kirjastoa, jota ei pysty itse muokkaamaan halutunlaiseksi. Tämä auttaisi myös sovelluksen kuvavertailun nopeuteen.

Tätä prototyyppiä ei testattu opinnäytetyön teon aikana oikealla laitteella. Sitä on kokeiltu ainoastaan erilaisilla emulaattoreilla tietokoneella, eikä esimerkiksi kosketusnäytöllisellä tabletilla. Jatkossa siis sovellusta tulisi testata ennen mahdollista käyttöönottoa. Koska sovellusta ei liitetty nyt olemassa olevaan poromerkkitietokantaan, sen testaamiseen tarvitaan myös tietokantayhteys.

Kaiken kaikkiaan prototyyppi onnistui mielestäni hyvin siihen nähden, että opinnäytetyön alussa ei ollut tietoa, onko mahdollista kehittää edes sovellusta, joka tunnistaisi poromerkit. Kuvavertailu kuitenkin voisi olla nopeampi ja siinä ei tarvitsisi käydä jokaisen osion kohdalla kaikkia yli 300 kuvaa läpi. Olisi ollut myös mielenkiintoista, mikäli sovellus olisi pystytty liittämään oikeaan tietokantaan ja sitä olisi voinut testata vielä paremmin. Uskon, että sovelluksesta on Paliskuntain yhdistykselle jatkossa hyötyä. Sitä voi edelleen jatkokehittää erityisesti kuvavertailun osalta. Lisäksi sovelluksen tai sen ominaisuuksia voi mahdollisesti ottaa jossain vaiheessa käyttöön esimerkiksi Vasama-hankkeessa kehitettävien offline-sovellusten yhtenä ominaisuutena.

LÄHTEET

Chiusano, F. 2022. Two minutes NLP — Sentence Transformers cheat sheet. Viitattu 15.2.2023 https://medium.com/nlplanet/two-minutes-nlp-sentence-transformers-cheat-sheet-2e9865083e7a.

Dart 2023. Paint your UI to life with Dart VM's instant hot reload. Viitattu 13.2.2023 https://dart.dev/.

Dreamsoft Innovations 2023. Screenshot 1.3.0. Viitattu 13.2.2023 https://pub.dev/packages/screenshot.

Flutter 2023a. Build apps for any screen. Viitattu 13.2.2023 https://flutter.dev/.

Flutter 2023b. CustomPaint class. Viitattu 13.2.2023 https://api.flutter.dev/flutter/widgets/CustomPaint-class.html.

Flutter 2023c. GestureDetector class. Viitattu 13.2.2023 https://api.flutter.dev/flutter/widgets/GestureDetector-class.html.

Flutter 2023d. Introduction to widgets. Viitattu 13.2.2023 https://docs.flutter.dev/development/ui/widgets-intro.

Flutter.cn 2023. Dio 5.0.0. Viitattu 13.2.2023 https://pub.dev/packages/dio.

GeeksforGeeks 2022. Listview.builder in Flutter. Viitattu 15.2.2023 https://www.geeksforgeeks.org/listview-builder-in-flutter/.

Gomez, J. 2023. Best Back-end Languages: How to Choose the Perfect One? Viitattu 15.2.2023 https://www.koombea.com/blog/best-back-end-languages-how-to-choose-the-perfect-one/.

Great Learning 2022. OpenCV Tutorial: A Guide to Learn OpenCV in Python. Viitattu 8.2.2023 https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/.

InterviewBit 2022. Flask Vs Django: Which Python Framework to Choose? Viitattu 15.2.2023 https://www.interviewbit.com/blog/flask-vs-django/.

Jakonen, S. 2021. Python Flask -kehyksen soveltuminen REST-rajapinnan kehittämiseen. Tampereen yliopisto. Informaatioteknologian ja viestinnän tiedekunta. Kandidaatintyö. Viitattu 15.2.2023 https://trepo.tuni.fi/bitstream/handle/10024/136134/JakonenSami.pdf?sequence=2&isAllowed=y.

Khan, S. A. 2022. How to compare two images in OpenCV Python? Viitattu 8.2.2023 https://www.tutorialspoint.com/how-to-compare-two-images-in-opencv-python.

Kulkarni, A. 2021. API vs REST API Simplified: 6 Critical Differences. Viitattu 15.2.2023 https://hevodata.com/learn/api-vs-rest-api/.

Lapin yliopisto 2017. Porotalouden digitalisoituminen. Loppuraportti. Viitattu 23.3.2023 https://www.lapinkeino.fi/wp-content/uploads/2019/05/PANTA-Porotalouden-digitalisoituminen.pdf.

MuleSoft 2023. What is a RESTful API? Viitattu 15.2.2023 https://www.mulesoft.com/resources/api/restful-api.

Paliskuntain yhdistys 2023a. Paliskuntain yhdistys. Viitattu 8.2.2023 https://paliskunnat.fi/py/organisaatio/paliskuntain-yhdistys/.

Paliskuntain yhdistys 2023b. Poromerkit. Viitattu 8.2.2023 https://paliskunnat.fi/py/neuvonta/poromerkit-ja-pilttarekisterit/.

Petrov D. 2016. Wavelet image hash in Python. Viitattu 8.2.2023 https://fullstackml.com/wavelet-image-hash-in-python-3504fdd282b5.

PyPi 2021. Deep Image Search - AI-Based Image Search Engine. Viitattu 8.2.2023 https://pypi.org/project/DeepImageSearch/.

PyPi 2022. Sentence Transformers: Multilingual Sentence, Paragraph, and Image Embeddings using BERT & Co. Viitattu 8.2.2023 https://pypi.org/project/sentence-transformers/.

Python 2023. What is Python? Executive Summary. Viitattu 8.2.2023 https://www.python.org/doc/essays/blurb/.

Rendyk 2022. Beginner’s Guide to Image and Text Similarity. Viitattu 8.2.2023 https://www.analyticsvidhya.com/blog/2021/05/beginners-guide-to-image-and-text-similarity/.

Sanastokeskus 2012. Termiharava. Viitattu 1.3.2023 http://www.terminfo.fi/sisalto/termiharava-136.html.

Simmons, L. 2023. Front-End vs. Back-End: What’s the Difference? Viitattu 17.3.2023 https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/.

Technostacks 2023. Most Popular Mobile App Development Frameworks For App Developers. Viitattu 15.2.2023 https://technostacks.com/blog/mobile-app-development-frameworks/.

TechyNilesh 2021. Deep Image Search – AI-Based Image Search Engine. Viitattu 15.2.2023 https://github.com/TechyNilesh/DeepImageSearch?ref=morioh.com&utm_source=morioh.com.

Työ- ja elinkeinoministeriö 2022. Euroopan aluekehitysrahaston (EAKR) rahoittaman hankkeen kuvaus. Viitattu 13.2.2023 https://www.eura2014.fi/rrtiepa/projekti.php?projektikoodi=A78732.

Upwork 2021. SOAP vs. REST: A Look at Two Different API Styles. Viitattu 15.2.2023 https://www.upwork.com/resources/soap-vs-rest-a-look-at-two-different-api-styles.

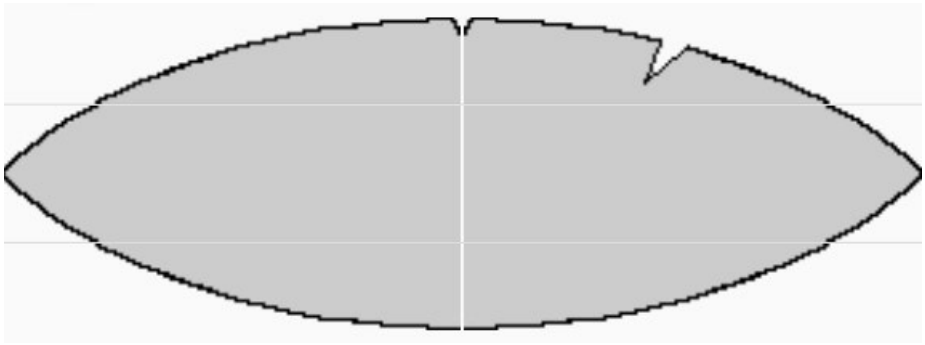
LIITE

Liite 1. Kuvavertailun testituloksia eri menetelmillä

Liite 1 1(6)

image_compare: ^1.1.2 paketin PerceptualHash-metodi:

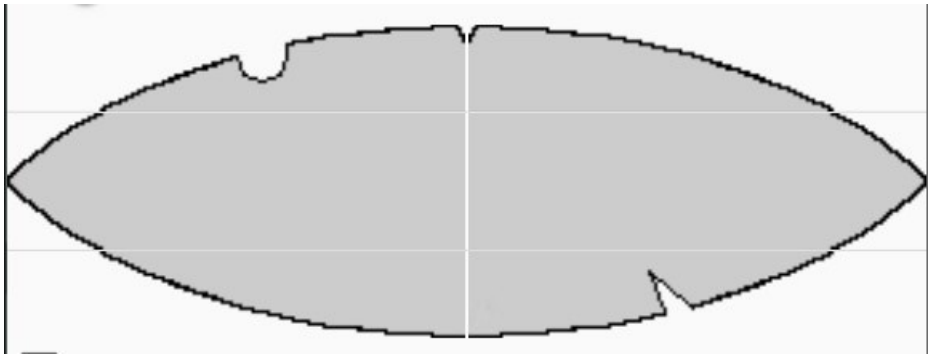
Vertailukuvat:



Earmark1

Earmark2

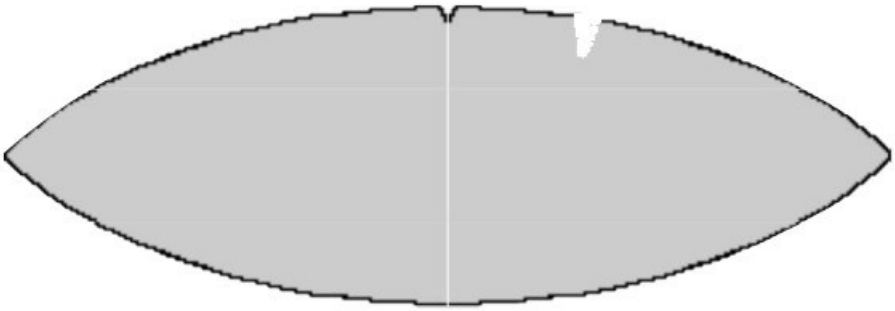
Liite 1 2(6)



Earmark3

Testi1:

Testattava kohde:

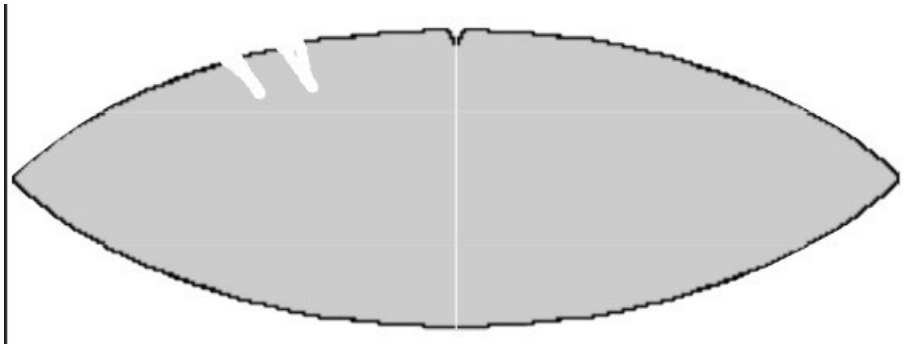


Vähiten eroavaisuuksia:

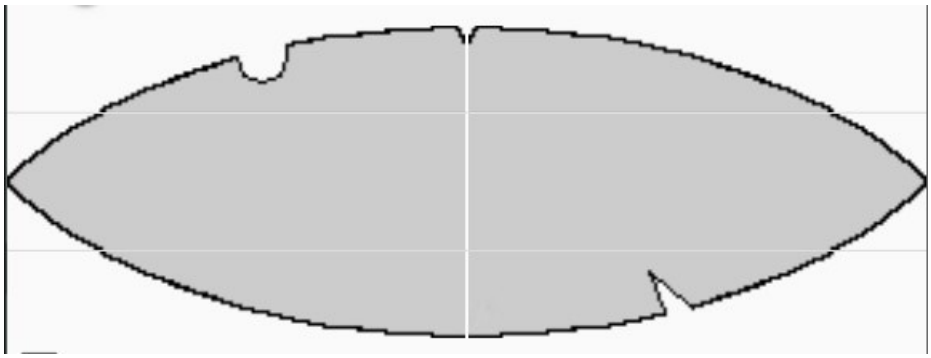
Testi2:

Testattava kohde:

Liite 1 3(6)

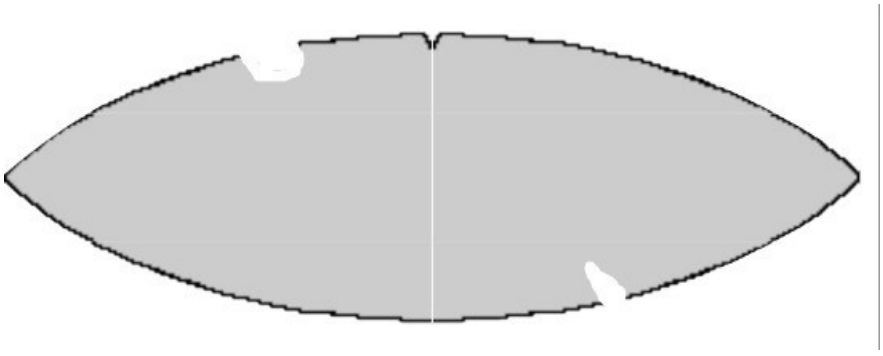


Vähiten eroavaisuuksia:



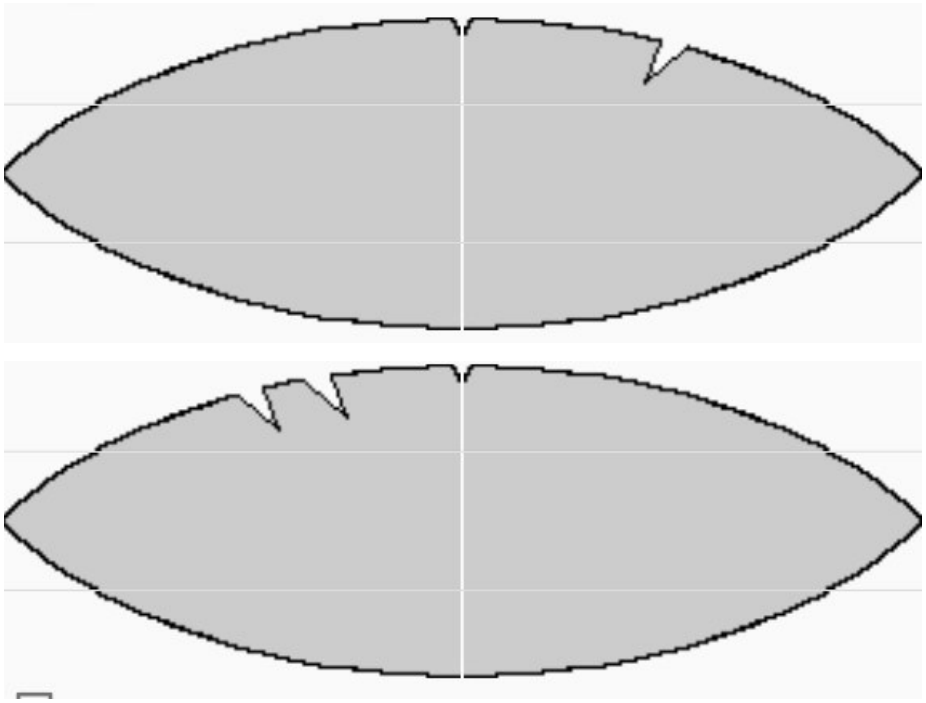
Testi3:

Testattava kohde:



Vähiten eroavaisuuksia:

Liite 1 4(6)



Johtopäätös: Ei tarjoa oikeita vertailukuvia.

Python Sentence-transformers-kirjastolla tehdyt vertailutulokset:

Vertailu 1:

Piirretty kuva:

Verrattiin 67 tietokannassa olevaan kuvaan ja tuloksena antoi 96,122 % vastaavuudella seuraavan kuvan:

Kuitenkin oikea kuva, mitä etsittiin, olisi ollut noiden 67 kuvan listassa seuraava kuva:

 95.745 % vastaavuus (sijalla 14.)

Liite 1 5(6)

Vertailu 2:

Piirretty kuva:

Verrattiin 67 tietokannassa olevaan kuvaan ja tuloksena antoi 96,332 % vastaavuudella seuraavan kuvan:

Kuitenkin oikea kuva, mitä etsittiin, olisi ollut noiden 67 kuvan listassa seuraava kuva:

 95.894 % vastaavuus (sijalla 15.)

Python cv2-kirjastolla MSE (keskimääräinen neliövirhe) testaus:

Vertailu 1:

Piirretty kuva:

Verrattiin 67 tietokannassa olevaan kuvaan ja tuloksena antoi, että vähiten eroavaisuutta (arvo lähimpänä 0) on seuraavalla kuvalla:

 eroavaisuus 8,313.

Kuitenkin oikea kuva, mitä etsittiin, olisi ollut noiden 67 kuvan listassa seuraava kuva:

 eroavaisuus 11.846

Liite 1 6(6)

Vertailu 2:

Piirretty kuva:

Verrattiin 67 tietokannassa olevaan kuvaan ja tuloksena antoi, että vähiten eroavaisuutta (arvo lähimpänä 0) on seuraavalla kuvalla:

 eroavaisuus 25.556.

Kuitenkin oikea kuva, mitä etsittiin, olisi ollut noiden 67 kuvan listassa seuraava

kuva:

 eroavaisuus 29.303