Health Advisor

Dokumentacija

Verzija projekta: M2
Verzija dokumentacije: XM2A4

XV. Gimnazija, Zagreb, 2019.

Autor: Mihael Turina

Mentor: Nada Klepić, prof.

Sadržaj

Uvod ---------- 3

Minimalna sistemska konfiguracija ---------- 4

Preporučena sistemska konfiguracija ---------- 4

Upute za instalaciju ---------- 5

Upute za korištenje ---------- 6

Kako skenirati ---------- 11

Planovi za budućnost ---------- 12

Tehničke informacije ---------- 13

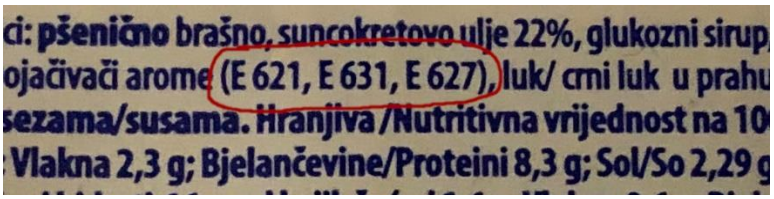
- Odabir tehnologija ---------- 13
- Struktura projekta ---------- 14
- O podacima u bazi podataka ---------- 15
- Sustav ocjenjivanja aditiva i proizvoda ---------- 16
- Izvorni kod ---------- 17

Uvod

Ja sam Mihael Turina i autor sam ovoga projekta. Imam 15 godina, pohađam 1. razred XV. Gimnazije u Zagrebu i bavim se programiranjem u Pythonu, C++ i Dartu zadnjih 8 godina. Dosad sam pisao samo servise i serverske aplikacije, pa sam htio naučiti pisati za mobilne platforme. Naučio sam Dart i u njemu sam napisao ovaj projekt.

Projekt koji se razvio u Health Advisor pokrenut je u 7. mjesecu 2018. jer sam bio frustriran E –brojevima. To su europske oznake za prehrambene aditive, a sastoje se od velikog slova „E“ i tri ili četiri znamenke. Problem je što neki označavaju bezopasne ili čak korisne aditive, dok drugi označavaju vrlo štetne aditive. Oni su svugdje, no jedini (realističan) način za znati što znače je sve ih pamtiti, što nije praktično niti pouzdano. Stoga sam istražio postoje li aplikacije koje rješavaju moj problem. Otkrio sam da postoje aplikacije koje ispisuju podatke o E –brojevima, no traže ručno upisivanje koda. To mi se nije svidjelo jer mi za to ne treba specijalizirana aplikacija, nego to mogu i tražiti na internetu. Zaključio sam da ću ja napraviti ***bolje*** rješenje. Budući da sam vidio demonstraciju modernih OCR i Machine Learning engina koja me jako impresionirala, odlučio sam koristiti rješenje na bazi fotografiranja ambalaže proizvoda. Plan projekta je dovršen u 11. mjesecu, a izrada je počela sredinom 12. mjeseca u programskome jeziku koji je tek izašao – Googlovom Dartu.

Health Advisor je mobilna aplikacija za procjenu štetnosti prehrambenih aditiva. Health Advisor omogućuje korisniku brzo, jednostavno i pouzdano prepoznavanje takozvanih „E - brojeva“ na popisu sastojaka prehrambenog proizvoda. Korisnik fotografira popis sastojaka kamerom mobilnoga uređaja, a zatim Health Advisor pročita sadržaj slike i ispiše popis svih „E - brojeva“ te njihova svojstva kao što su štetnost, namjena i kemijski sastav. U slučaju da aplikacija ne može prepoznati E – brojeve sa slike, korisnik može koristiti ručnu pretragu koda.



Slika 1[1]: Dio naljepnice prehrambenog proizvoda (Tuc krekera). Vidljivi su E -brojevi, ali njihovo značenje nije pojašnjeno (sva 3 su jako štetna, a vi to ne biste niti znali da ne pretražujete značenje. Health Advisor rješava taj problem.).

[1] Slike 1 i od 3 do 12 su snimljene na Android uređaju (Sony Xperia Z5 Compact), dok su slike od 13 do 18 snimljene na iOS uređaju (iPhone 8). Slika 2 je snimljena na stolnom računalu sa Windowsima.

Minimalna sistemska konfiguracija

- Mobilni uređaj sa kamerom (preporučeno 8 megapiksela ili više)
- Operativni sustav Android 5.0 ili iOS 9.0 (ili noviji)
- 15 MB ili više prostora na mediju za pohranu

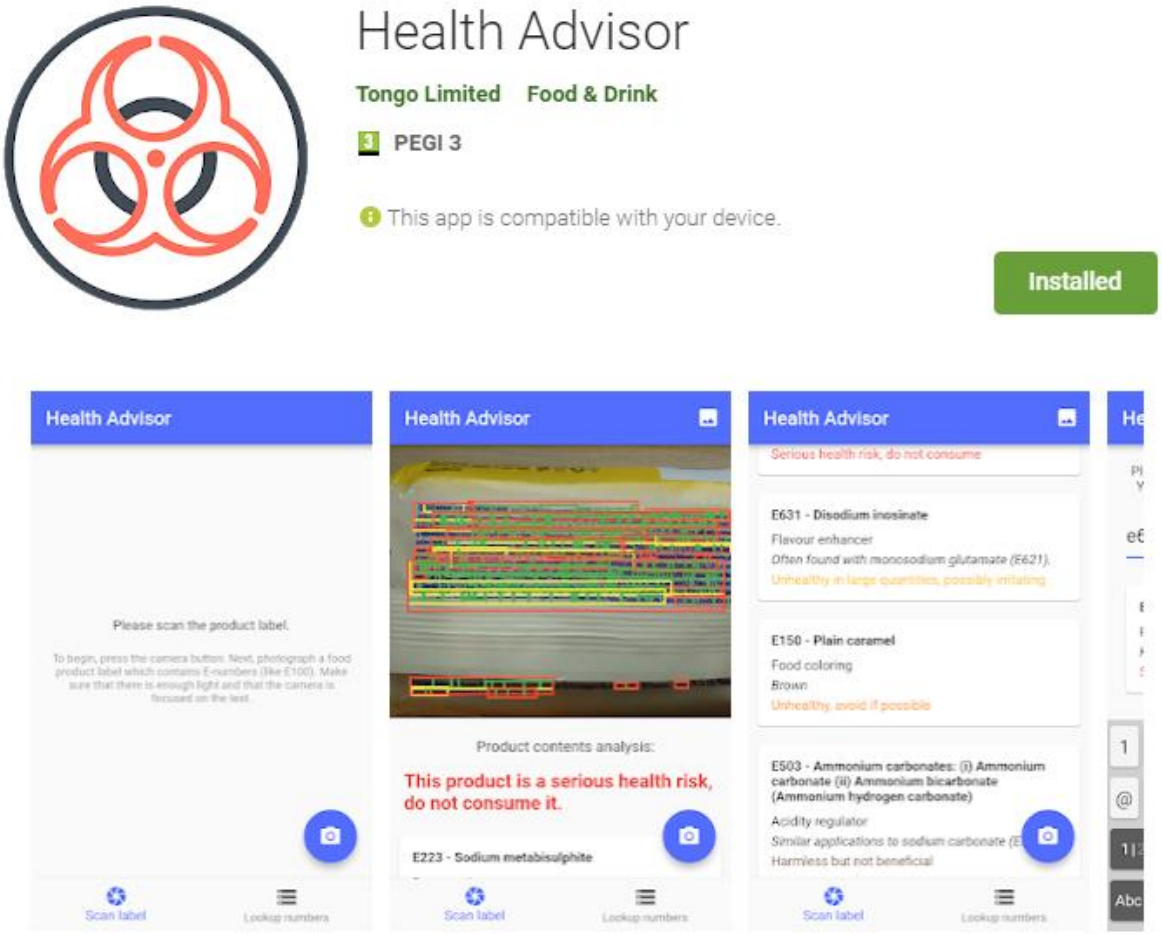
Preporučena sistemska konfiguracija

- Mobilni uređaj sa kamerom od 12 megapiksela
- Operativni sustav Android 9.0 (Pie) ili iOS 12 (ili noviji)
- 15 MB ili više prostora na mediju za pohranu
- Geekbench 4 score od 5000 ili više

Moderni uređaji su poželjni jer imaju kvalitetniju kameru te brže i glađe izvršavaju aplikaciju.

Upute za instalaciju

Aplikaciju je moguće besplatno preuzeti sa Google Play Storea pod nazivom Health Advisor Beta. U budućnosti će biti postavljena i na Apple App Store. Aplikacija je trenutno besplatna, ali će se vjerojatno u budućnosti naplaćivati oko 2 USD. Dolje: Slika 2: Health Advisor na Google Play Storu.



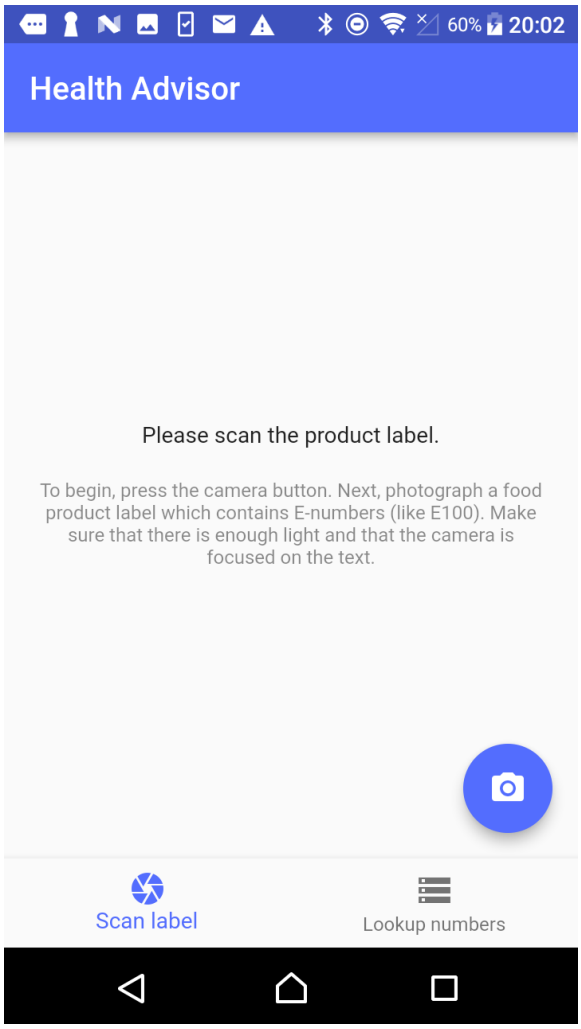
Have you ever found yourself in a store, looking at the label of a food product, seeing certain "codes", such as E300, E951 or E621, and wondering, what do they mean? These "E-numbers" actually stand for food additives. But how do you know if they are good or bad? In fact, one of these is very healthy and the other two are potentially carcinogenic! To find out which is which, use Health Advisor!

Health Advisor works by recognizing all E-numbers on the label and giving you a report on their danger/benefit levels, as well as an assessment of the product as a whole. Usage is simple, just photograph the food label and Health Advisor will do the rest of the work for you! And in case the photographic mode fails, you can always search for the code yourself on the manual lookup screen.

In order to ensure the best possible results, use Health Advisor in a well-lit environment, place your device camera as close as you can to the food label, and focus the image.

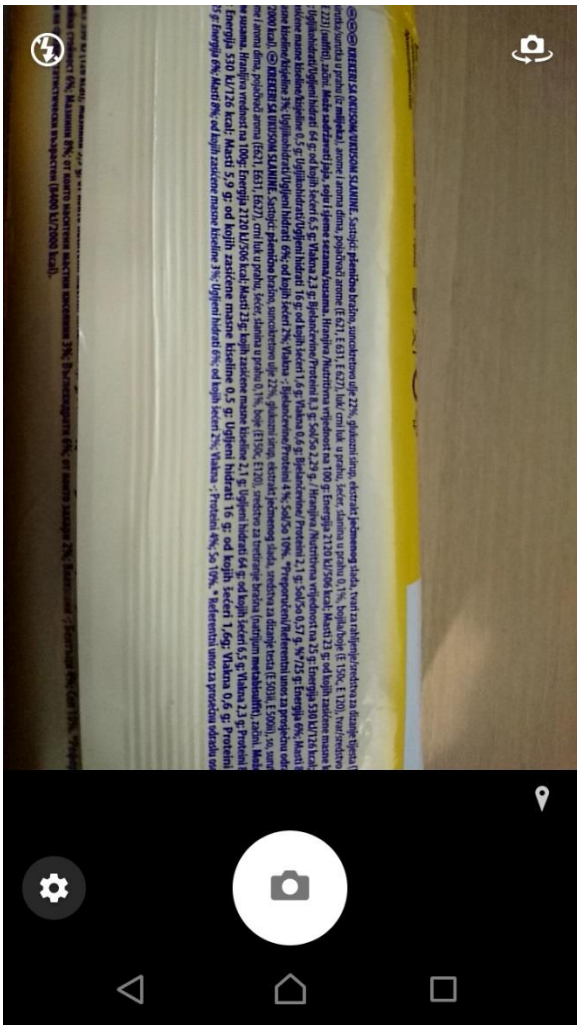
Upute za korištenje

Nakon pokretanja aplikacije će se prikazati početna stranica vidljiva na Slici 3.

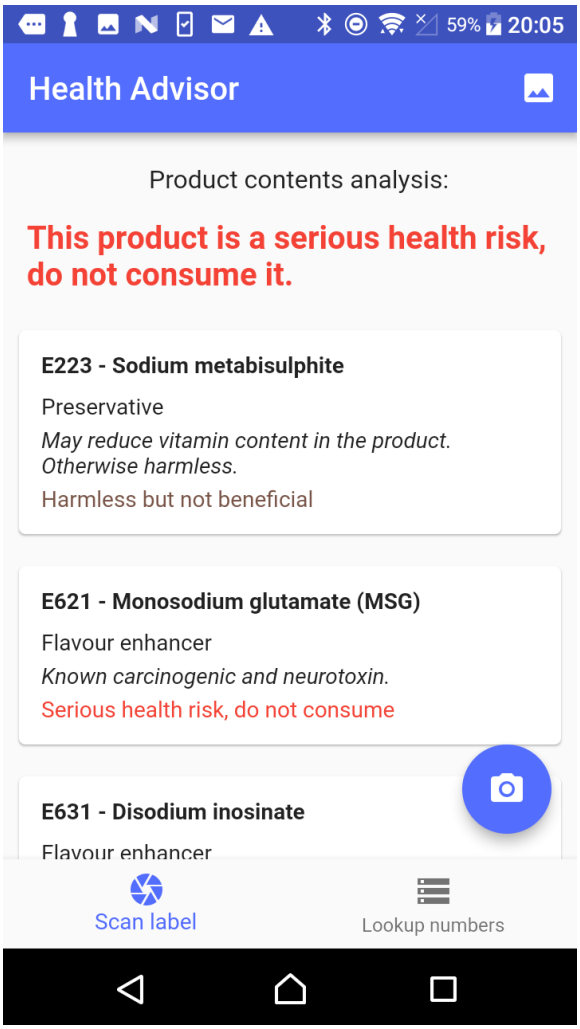


Slika 3: Početna stranica aplikacije

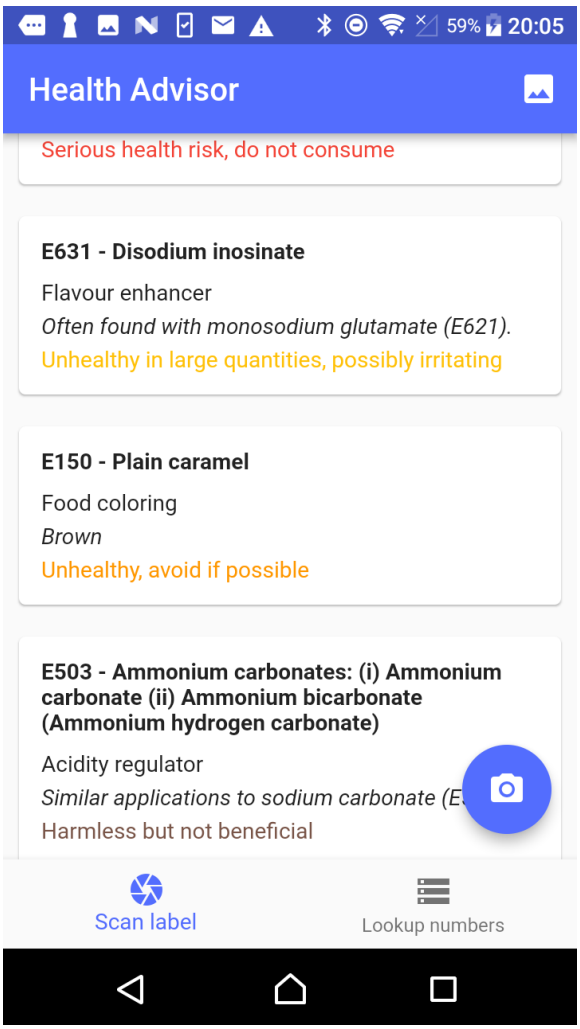
Aplikacija govori korisniku da fotografira ambalažu proizvoda te mu daje upute za uspješno prepoznavanje. Kad korisnik pristisne gumb, otvara se sučelje kamere vidljivo na Slici 4. Kad je zadovoljan slikom, korisnik pritišće gumb za snimanje i aplikacija radi ostatak posla. Kad je gotova, ispisuje podatke vidljive na Slikama 5 i 6. U gornjem desnom kutu se pojavljuje gumb sa ikonom slike. Pritiskom na taj gumb prikazuje se obrađena fotografija. Ponovni pritisak skriva fotografiju (gumb je toggle).



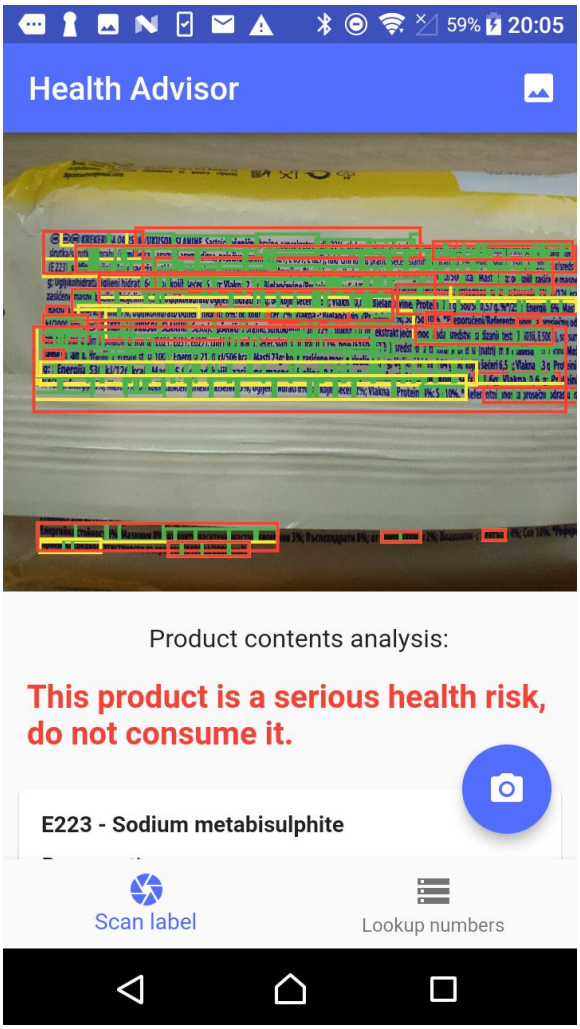
Slika 4: Sučelje kamere



Slika 5: Ispis rezultata analize



Slika 6: Prikaz kartica s podacima



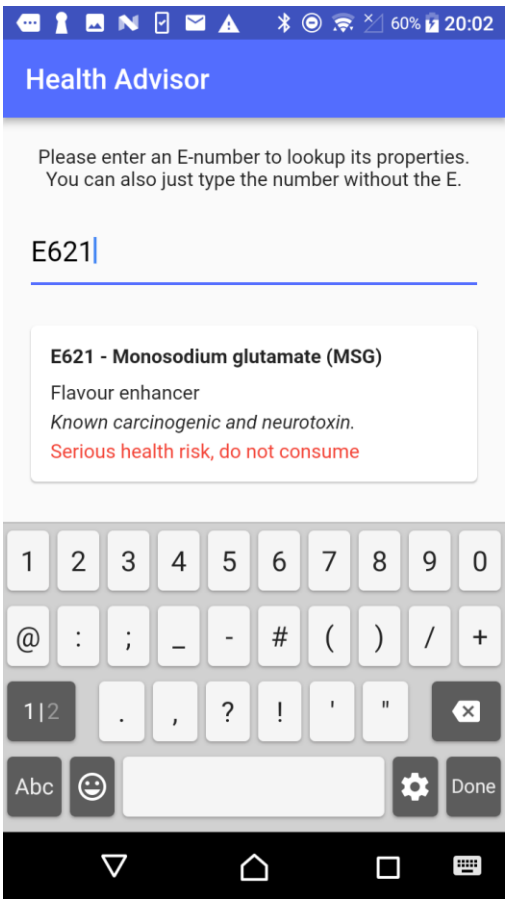
Slika 7: Vidljiva je obrađena fotografija

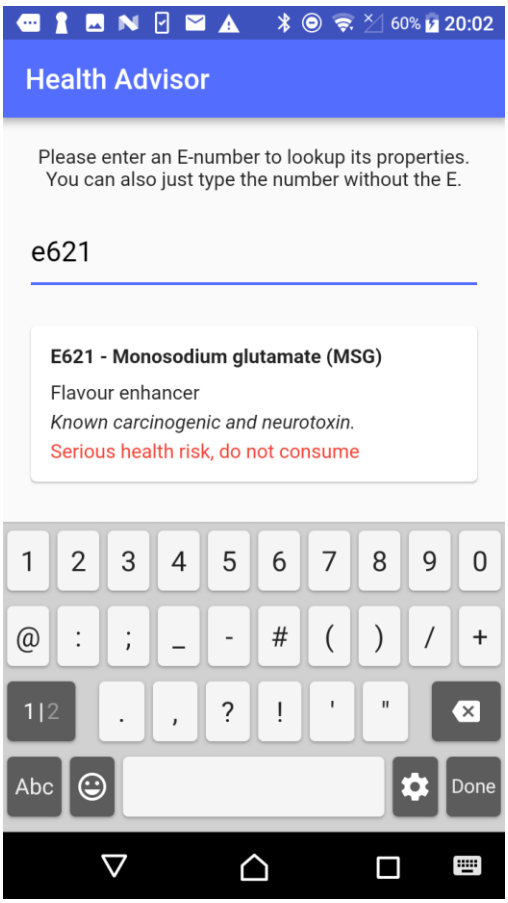
U slučaju da aplikacija ne može prepoznati E-brojeve sa ambalaže, korisnik može ručno unijeti E-brojeve na za to predviđenoj stranici te će ih zatim aplikacija pretražiti i ispisati podatke o njima. Sučelje je prikazano na Slikama 8, 9, 10, 11 i 12 na sljedećim stranicama.

Ponovnim pritiskom na gumb kamere korisnik može ponovno skenirati isti ili novi proizvod.

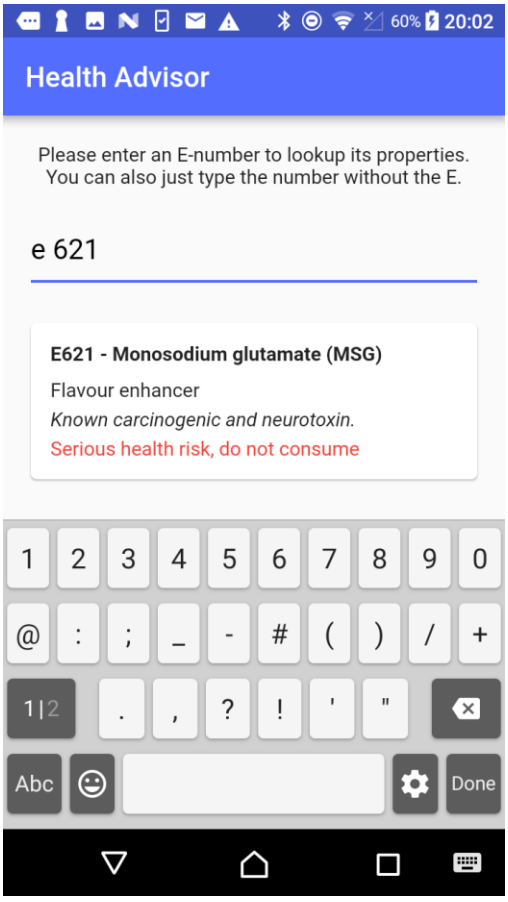
Na starijim uređajima aplikacija može raditi sporo te se čini kao da se „zamrznula“ ili da se ništa ne događa. To nije problem, samo je potrebno pričekati da ponovno počne reagirati te ne gasiti aplikaciju.

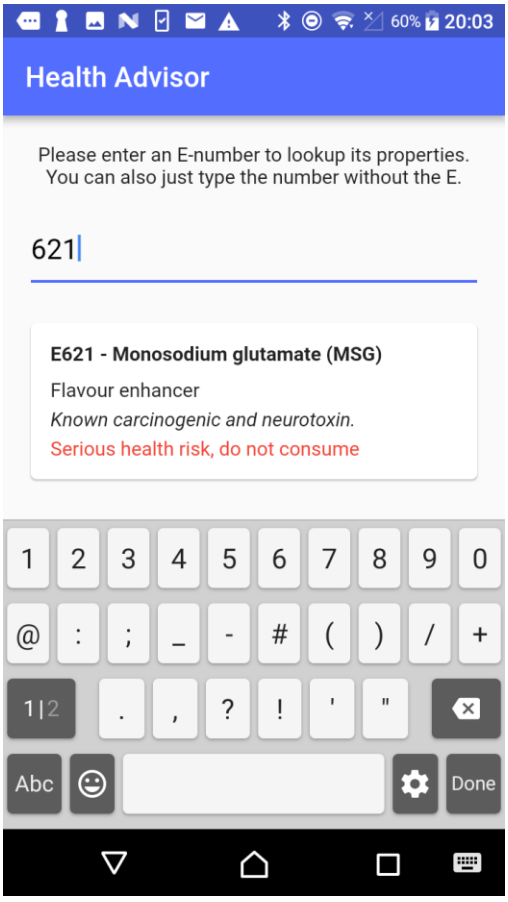
Slika 8: Sučelje za ručni unos brojeva





Slike 9, 10, 11 i 12: Prikazana su 4 različita načina za unos broja, te ispisana kartica s podacima.

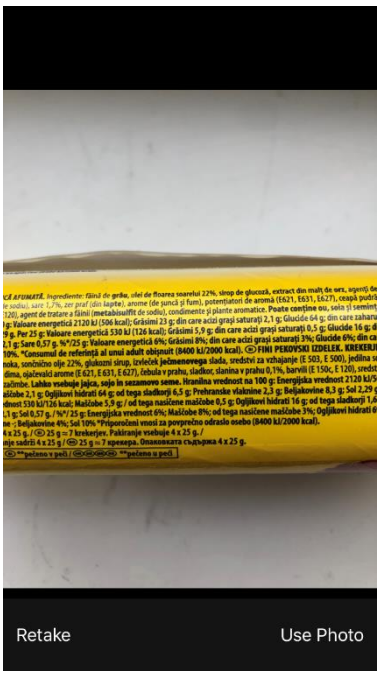


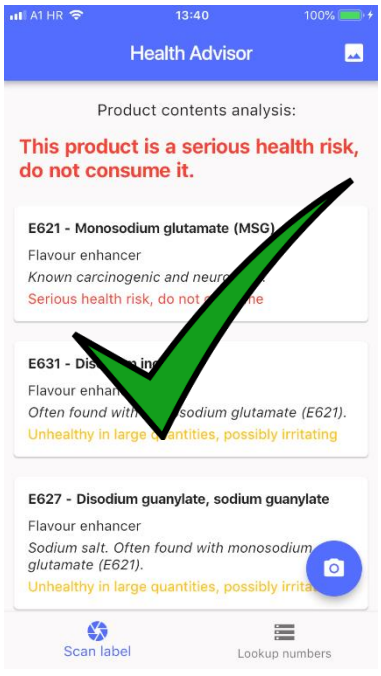


Kako skenirati

Kada sam podijelio primjerke aplikacije svojim „beta testerima“, primijetio sam da često ne znaju kako točno skenirati ambalažu proizvoda, te skeniraju bar-kod, serijski broj ili tablicu prehrambene vrijednosti umjesto popisa sastojaka. Takav način je pogrešan, te je ispravno fotografirati popis sastojaka (ako sadrži E – brojeve. Na nekim proizvodima su aditivi označeni kemijskom formulom ili trgovačkim nazivom, a ne E – brojem.). Ispravan i neispravan način skeniranja prikazani su Slikama 13, 14, 15 i 16:

Slike 13, 14, 15 i 16: Ilustracija ispravnog i pogrešnog načina fotografiranja ambalaže





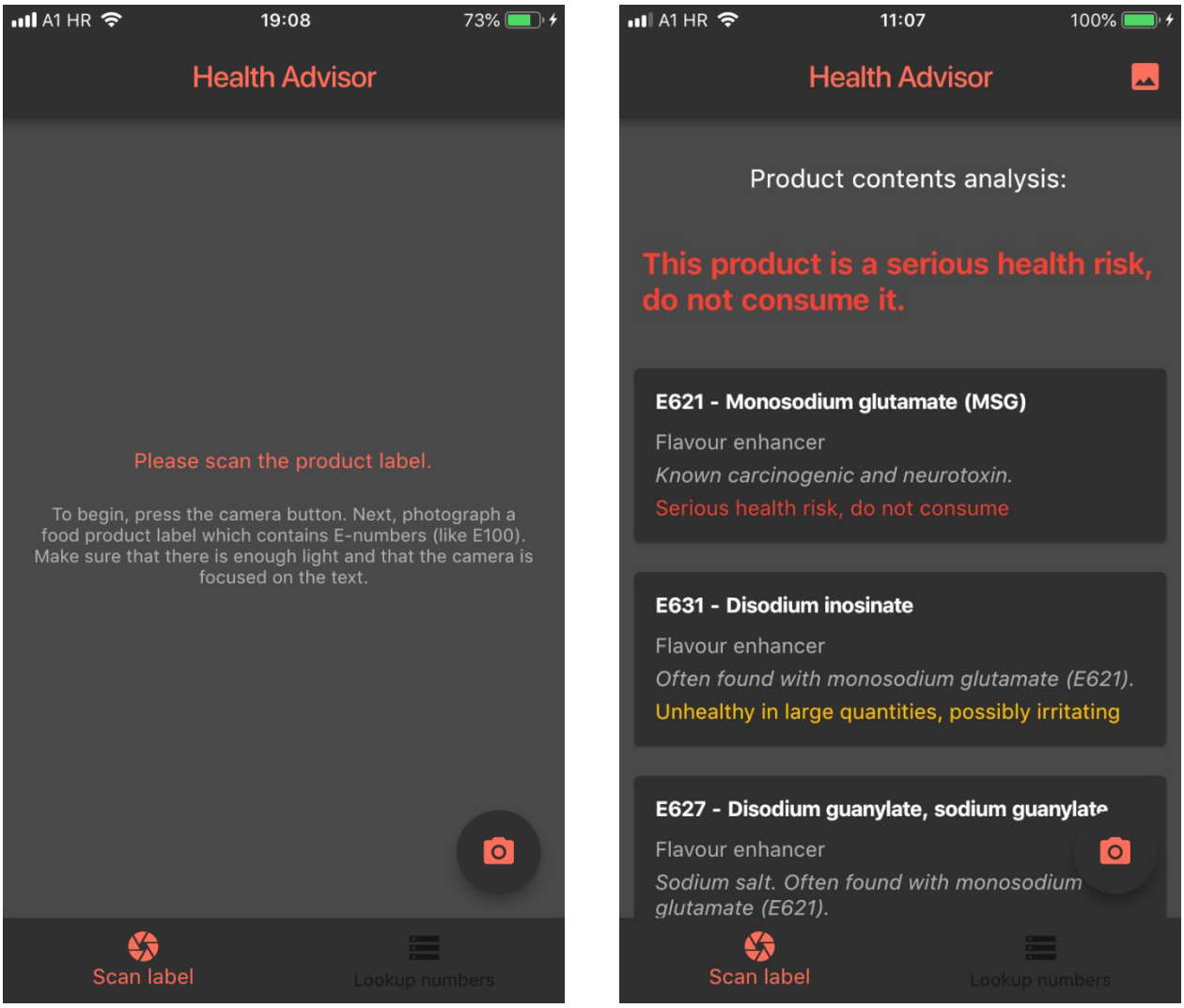
Planovi za budućnost

U planu je novi dizajn korisničkog sučelja aplikacije. Aplikacija će najvjerojatnije imati svijetlu i tamnu temu. Tamna tema je trenutno u eksperimentalnoj formi te je vidljiva na Slikama 17 i 18, dok je svijetla tema u vrlo ranoj fazi razvoja te je nije moguće ovdje prikazati.

Mod za ručnu pretragu će imati napredniji algoritam za pretraživanje koji će moći ispisivati više od jednog rezultata te će biti moguća pretraga aditiva po imenu.

Iako je verzija aplikacije za iOS potpuno funkcionalna, još nije postavljena na App Store, ali biti će u skoroj budućnosti.

U planu je i mogućnost generiranja crash reportova i slanje istih na privatni server radi analize i uklanjanja bugova. Ovaj feature trenutno nije izrazito potreban jer je aplikacija vrlo stabilna.



Slike 17 i 18: Prototip buduće tamne teme za aplikaciju

Tehničke informacije

Odabir tehnologija

Aplikacija je napisana u Flutter frameworku koji koristi jezik Dart. Dart je sličan Javi, ali je napravljen za optimalan rad s Flutterom. Glavna prednost Fluttera je ta da se aplikacija napisana u njemu bez ikakvih promjena može koristiti i za Android i za iOS. Razlog zašto je korišten Flutter a ne druge, slične tehnologije kao što je React Native, je jednostavan – nisam znao za njih.

Sav razvoj je obavljen u JetBrains-ovom Android Studiu gdje se Flutter koristi kao plugin. Kompajliranje za Android vrši sam Android studio, a kompajliranje za iOS se vrši preko Xcodea koji se poziva iz Android Studia. Dart kod se kompajlira u nativni kod za tu platformu, što znači da se za iOS kompajlira u C++, a za Android u Javu. Takav način rada optimizira aplikaciju i omogućava joj da automatski koristi sistemske resurse kao što su tipkovnica i ikone bez ikakvog podešavanja.

Za OCR (Optical Character Recognition, odnosno algoritam za skeniranje teksta sa slike) i algoritam za crtanje pravokutnika na slici je korišten ML Kit sa Googlovog Firebasea. Izvorno sam pokušao koristiti Tesseract OCR ali sam imao problema s njim te nije radio dobro. Nisam ni pomišljao o pisanju vlastitog OCR-a jer bi mi za to trebale godine i iznimno velika stručnost, a čak i kad bih uspio, moj sustav bi vjerojatnije bio lošiji od tuđih jer se OCR tehnologija razvija i unaprijeđuje preko 30 godina od strane korporacija koje imaju mnogo više resursa nego ja. Odlučio sam koristiti Firebase jer je to pouzdan servis sa dobrom podrškom, a jednostavan je za korištenje. Još jedan faktor za Firebase je taj da se sigurno može koristiti u komercijalnim aplikacijama. Problem sa ML Kit OCR-om je što zahtijeva točno određenu orijentaciju slike, pa je potrebno reorijentirati sliku pomoću EXIF podataka nakon što je aplikacija snimi, što je neoptimalno rješenje.

Za bazu podataka je korišten SQLite engine. Razlog tome su moja pozitivna prethodna iskustva s njim, te činjenica da SQLite znam koristiti a druge engine ne. Razlog zašto baza podataka nije na Firebaseu je taj što bi aplikacija u tom slučaju zahtijevala vezu na internet, što je samo faktor nepouzdanosti u slučaju gdje nije nužno potreban.

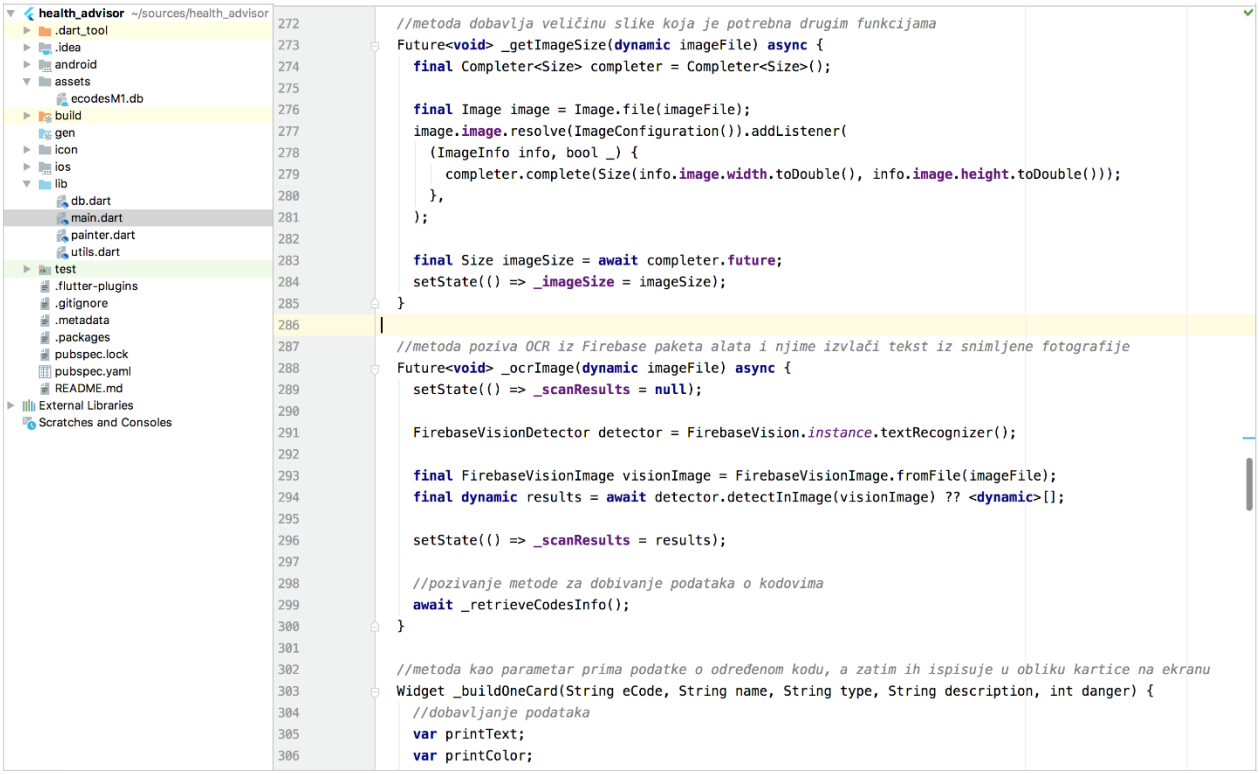
Za uploadanje aplikacije i popratnih materijala na Play Store je korišten sustav Fastlane koji automatizira proces i osigurava od grešaka.

Korišteni su paketi sqflite za upravljanje bazom podataka, image_picker za sučelje kamere, flutter_icons za automatiziranu promjenu ikona aplikacije i flutter_exif_rotation za reorijentaciju fotografije nakon fotografiranja, prema EXIF podacima.

Korištene su i moje vlastite Python 3 skripte za obradu i upisivanje podataka s interneta u bazu podataka.

Struktura projekta

Sav kod aplikacije se nalazi u datotekama main.dart, db.dart, utils.dart i painter.dart. Main se bavi korisničkim sučeljem i agregiranjem podataka iz ostalih datoteka, db vrši operacije na bazi podataka, Utils sadrži sustav za ocjenjivanje i algoritam za izvlačenje E – brojeva iz teksta, dok Painter sadrži algoritam za crtanje oznaka na fotografiji. Te datoteke se nalaze u mapi lib. U mapi assets se nalazi baza pod imenom ecodesM1.db. U mapama android i ios se nalaze datoteke nužne za rad aplikacije na tim platformama. U mapi icon se nalaze ikone aplikacije. Važna je i datoteka pubspec.yaml koja se nalazi u rootu projekta, a popisuje sve potrebne assete za rad aplikacije, uključujući librarye, ikone i bazu podataka.



Slika 19: Screenshot iz Android Studia, vidljiva je struktura projekta te dio koda iz main.dart

O podacima u bazi podataka

Podaci o aditivima koje predstavljaju E – brojevi su dobavljeni sa velikog broja web stranica. Među glavnima su Wikipedia, Food Info (food-info.net), Noshly (noshly.com) i Health Line (healthline.com). Informacije su dobavljane sa više izvora, uključujući i izvorne znanstvene radove. Informacije nisu sve nužno potpuno točne, ali stvaraju generalnu sliku o aditivu. Izvorni popis brojeva je dobavljen sa stranice na Wikipediji en.wikipedia.org/wiki/E_number. Kao što je vidljivo, taj popis je iznimno nepotpun te je zato dopunjavan informacijama sa drugih stranica.

U bazi podataka svaki aditiv ima sljedeće podatke o sebi:

- Svoj E-broj po kojemu ga se identificira
- Ime – najčešće trgovački naziv ili formula kemijskog spoja, često i sa informacijama o svrsi
- Tip aditiva – njegova svrha
- Opis – uključuje detaljne informacije o zdravstvenim učincima i nuspojavama, status u pojedinim državama, vrste proizvoda u koje se aditiv stavlja te uvjete u kojima može biti zdrav/štetan
- Ocjenu štetnosti (rating) – Cijeli broj između [-3 i 3]. 0 je neutralno, veće od 0 je nepoželjno, a manje od 0 je poželjno. Više o sustavu ocjenjivanja na sljedećoj stranici. Aplikacija ovu ocjenu prevodi u poruku koju ispisuje obojanu odgovarajućom bojom.

Ti podaci se koriste za izradu kartica koje se ispisuju nakon skeniranja ili ručne pretrage.

E-brojevi, nazivi i tipovi aditiva su programski preuzeti sa već spomenute Wikipedia stranice pomoću Python skripti. Opisi i ocjene (najvažniji dio) su svi ručno sastavljeni i upisani u bazu. Taj dio obrade podataka se ne može automatizirati jer isključivo ovisi o ljudskoj procjeni i analizi.

	code	name	type	description	danger
	F...	Filter	Filter	Filter	F...
127	E321	Butylated hydroxytoluene (BHT)	Antioxidant	Antioxidant in fats and fatty products. Can cause liver damage, allergic reactions...	2
128	E322	Lecithin	Emulsifier	Found in every living organism as a part of the cell wall. Harmless.	0
129	E323	Anoxomer	Antioxidant	Synthetic antioxidant. Non-digestible. Harmless.	0
130	E324	Ethoxyquin	Antioxidant	Used in pet foods and fish feed. It was found to be unsafe and it is slowly phase...	1
131	E325	Sodium lactate	Acidity regulator	Sodium salt of lactic acid (E270). Same properties.	0
132	E326	Potassium lactate (antioxidant)	Acidity regulator	Potassium salt of lactic acid (E270). Same properties.	0
133	E327	Calcium lactate	Acidity regulator	Calcium salt of lactic acid (E270). Same properties.	0
134	E328	Ammonium lactate	Acidity regulator	Used in skin lotions. Harmless.	0
135	E329	Magnesium lactate	Acidity regulator	Used as a magnesium supplement. Do not consume if allergic.	0
136	E330	Citric acid	Acid, acidity regulator	Found in most fruits. Used to stabilize jams and jellies. Harmless.	0
137	E331	Sodium citrates (i) Monosodium citrate (ii) Disodium citrate (iii) Sodium citrate (trisodium citrate)	Acidity regulator	Sodium salts of citric acid (E330). Harmless.	0
138	E332	Potassium citrates (i) Monopotassium citrate (ii) Potassium citrate (tripotassium citrate)	Acidity regulator	Potassium salts of citric acid (E330). Used as antacid. Harmless.	0
139	E333	Calcium citrates (i) Monocalcium citrate (ii) Dicalcium citrate (iii) Calcium citrate (tricalcium citrate)	Acidity regulator, firming agen...	Calcium salts of citric acid (E330). Harmless.	0
140	E334	Tartaric acid (L(+)-)	Acid	Present in many fruits, grapes in particular. Not metabolized, therefore harmless.	0
141	E335	Sodium tartrates (i) Monosodium tartrate (ii), Disodium tartrate	Acidity regulator	Sodium salts of tartaric acid (E334). Harmless.	0
142	E336	Potassium tartrates (i) Monopotassium tartrate (cream of tartar) (ii) Dipotassium tartrate	Acidity regulator	Potassium salts of tartaric acid (E334). Harmless.	0
143	E337	Sodium potassium tartrate	Acidity regulator	Sodium and potassium salt of tartaric acid (E334). Harmless.	0
144	E338	Orthophosphoric acid	Acid	Commonly used as an additive in cola. Harmless.	0
145	E339	Sodium phosphates (i) Monosodium phosphate (ii) Disodium phosphate (iii) Trisodium phosphate	Antioxidant	Source of phosphates, which are essential for normal organism function.	-2
146	E340	Potassium phosphates (i) Monopotassium phosphate (ii) Dipotassium phosphate (iii) Tripotassium phosphate	Antioxidant	Source of phosphates, which are essential for normal organism function.	-2
147	E341	Calcium phosphates (i) Monocalcium phosphate (ii) Dicalcium phosphate (iii) Tricalcium phosphate	Anti-caking agent, firming agent	Source of phosphates, which are essential for normal organism function.	-2
148	E342	Ammonium phosphates: (i) monoammonium phosphate (ii) diammonium phosphate	Antioxidant	Source of phosphates, which are essential for normal organism function.	-2
149	E343	Magnesium phosphates (i) monomagnesium phosphate (ii) Dimagnesium phosphate	Anti-caking agent	Source of phosphates, which are essential for normal organism function.	-2
150	E344	Lecithin citrate	Acidity regulator	Salt of lecithin (E322) and citric acid (E330). Harmless.	0
151	E345	Magnesium citrate	Acidity regulator	Used as a laxative. Prevents kidney stones. No other effects.	0
152	E349	Ammonium malate	Acidity regulator	Known to cause allergic reactions.	1
153	E350	Sodium malates (i) Sodium malate (ii) Sodium hydrogen malate	Acidity regulator	Sodium salts of malic acid (E296). Same properties.	0
154	E351	Potassium malate	Acidity regulator	Potassium salt of malic acid (E296). Same properties.	0
155	E352	Calcium malates (i) Calcium malate (ii) Calcium hydrogen malate	Acidity regulator	Calcium salts of malic acid (E296). Same properties.	0
156	E353	Metatartaric acid	Emulsifier	Metabolized in the body to tartaric acid (E334). Harmless.	0

Slika 20: Dio baze podataka prikazan u browseru za bazu

Sustav ocjenjivanja aditiva i proizvoda

Svaki aditiv ima svoju ocjenu koja je cijeli broj od [-3 do 3]. Te ocjene su ovdje pojašnjene:

- -3 – Vrlo zdravo. Najčešće vitamini ili tvari koje imaju mnogo dobrih svojstava a niti jedno loše, barem u količinama koje se nalaze u hrani. Poruka koja se ispisuje u aplikaciji za ovu ocjenu je: Very healthy, seek intake.
- -2 – Zdravo. To su tvari koje imaju više dobrih svojstava, ali nisu osobito izražena ili tvar ima i poneko loše svojstvo. Općenito, unos takvih tvari je preporučen, ali u umjerenim količinama. Poruka je: Healthy.
- -1 – Korisno. Tvari koje imaju poneko dobro svojstvo, ali to svojstvo nije osobito izraženo ili je korisno samo u određenim situacijama. Tvari s ovom ocjenom mogu imati i loša svojstva, ali su ona ipak slabija nego dobra svojstva. Poruka: Beneficial, possibly over time.
- 0 – Neutralno. Tvar ili ne utječe na organizam ili ima slaba dobra i slaba loša svojstva. U drugom slučaju, opis aditiva će reći više detalja o tim svojstvima. Poruka: Harmless but not beneficial.
- 1 – Blago nepoželjno. Može izazivati alergijske reakcije, spriječavati unos korisnih tvari ili izazivati blage probavne smetnje. Poruka: Unhealthy in large quantities, possibly irritating.
- 2 – Nezdravo. Može uzrokovati debljanje, probleme s hormonima, probavne i srčane bolesti, jake alergijske reakcije te čak trovanje kod osjetljivih osoba. Poruka: Unhealthy, avoid if possible.
- 3 – Iznimno štetno. Kancerogeni, otrovi, halucinogeni, ostale iznimno štetne tvari. Uzimanje je jako riskantno. Poruka: Serious health risk, do not consume.

Kod fotografske analize slike se zbrajaju sve ocjene štetnosti. Zatim se taj broj boduje po sljedećoj skali i tako se određuje štetnost cijelog proizvoda. U slučaju da proizvod sadrži barem jedan aditiv sa ocjenom 3, automatski se smatra iznimno štetnim i ispisuje se poruka This product is a serious health risk, do not consume it. Slijedi skala:

- < -3 – Proizvod je vrlo zdrav. Preporučeno ga je konzumirati. Poruka: This product is very healthy.
- <-1, -3] – Proizvod je zdrav. Potpuno ga je sigurno konzumirati te je u umjerenim količinama čak i preporučeno. Poruka: This product is moderately healthy.
- [1, -1] – Proizvod je zdravstveno neutralan i siguran. Poruka: This product is safe for consumption.
- <1, 3] – Proizvod je blago nezdrav. Unos nije štetan u manjim količinama, ali u većima nije preporučen. Poruka: This product should not be consumed in larger quantities.
- > 3 – Proizvod je nezdrav. Nije ga poželjno često uzimati te ga je najbolje općenito izbjegavati. Poruka: This product is unhealthy, consumption is not recommended.

U slučaju da ne postoje podaci o štetnosti određenog aditiva, bit će smatran neutralnim.

Izvorni kod

Slijedi čitavi izvorni kod projekta.

main.dart:

```
//osnovni paketi
import 'dart:async';
import 'dart:io';

//vanjski paketi
import 'package:firebase_ml_vision/firebase_ml_vision.dart';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:flutter_exif_rotation/flutter_exif_rotation.dart';

//drugi dijelovi aplikacije
import 'db.dart';
import 'utils.dart';
import 'painter.dart';


//pokretanje aplikacije
void main() => runApp(
    MaterialApp(
        debugShowCheckedModeBanner: false,
        home: MainPage(),
        theme: ThemeData(primaryColor: Colors.indigoAccent)
    )
);

//početna stranica
class MainPage extends StatefulWidget {
  @override
  _MainPageState createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> {
  PageController _pageController;
  int _page = 0;

  //slijede funkcije za upravljanje mijenjanjem stranica
  @override
  void initState() {
    super.initState();
    _pageController = new PageController();
  }

  @override
  void dispose() {
    super.dispose();
    _pageController.dispose();
  }
```

```
void navigationTapped(int page) {
    _pageController.animateToPage(page, duration: const Duration(milliseconds:
300), curve: Curves.ease);
  }

  void onPageChanged(int page) {
    setState(() {
      this._page = page;
    });
  }

  @override
  //sastavljanje početne stranice
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomPadding: false,
      body: PageView(
        children: [
          _HomePage(),
          _ManualScanPage(),
        ],
        onPageChanged: onPageChanged,
        controller: _pageController,
      ),
      bottomNavigationBar: Theme(
        data: Theme.of(context).copyWith(
        ),
        child: BottomNavigationBar(//traka za biranje stranica
          items: [
            BottomNavigationBarItem(//stranica za skeniranje
                icon: Icon(Icons.camera),
                title: Text("Scan label", style: TextStyle(),
                )),
            BottomNavigationBarItem(//stranica za ručni unos
                icon: Icon(Icons.storage),
                title: Text("Lookup numbers", style: TextStyle(),
                )),
          ],
          onTap: navigationTapped,
          currentIndex: _page,
        ),
      ),
    );
  }
}

//stranica za ručni unos
class _ManualScanPage extends StatefulWidget {
  @override
  _ManualScanPageState createState() => _ManualScanPageState();
}

class _ManualScanPageState extends State<_ManualScanPage> {
  TextEditingController _searchController = TextEditingController();
  Map _data = {};

  _ManualScanPageState() {
    _searchController.addListener(() {
      doSearch(_searchController.text);
    });
  }
```

```
  //metoda za pretragu upisanog pojma
  void doSearch(String query) async {
    if (query.isEmpty) {
      setState(() {
        _data = {};
      });
      return;
    }

    //obrada stringa
    String queryFinal;
    String query2 = query.toUpperCase();
    String query3 = query2.replaceAll(RegExp(r'\s'), "").trim();
    if (query3[0] != "E" && query3[0] != "e") {
      queryFinal = "E"+query3;
    } else {
      queryFinal = query3;
    }

    //upit u bazu
    var data = await getDataFromDatabase(DefaultAssetBundle.of(context),
queryFinal);
    if (data != null) {
      print("Data found: $_data");

      setState(() {
        _data = data;
      });

    } else {
      setState(() {
        _data = {};
      });
    }
  }

  //metoda obrađuje podatke iz baze i poziva metodu za izgradnju kartice
  Widget unpackDbData (Map data) {
    dynamic homePageState = _HomePageState();
    String name = data['name'];
    String code = data['code'];
    String type = data['type'];
    String description = data['description'];
    int danger = data['danger'];
    Card card = homePageState._buildOneCard(code, name, type, description,
danger);
    return card;
  }
```

```
  //sastavljanje stranice
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomPadding: false,
        appBar: AppBar(
          title: Text("Health Advisor"),
        ),
        body: Padding(
          padding: EdgeInsets.all(20),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              //natpis s uputama
              Text("Please enter an E-number to lookup its properties. You can
also just type the number without the E.", textAlign: TextAlign.center,),
              SizedBox(height: 20),
              Row(
                children: <Widget>[
                  Expanded(
                    //polje za unos teksta
                    child: TextField(
                      autofocus: true,
                      style: TextStyle(fontSize: 20, color: Colors.black),
                      controller: _searchController,
                      //primjer upita
                      decoration: InputDecoration(hintText: "e.g. E621 or
e621")
                    ),
                  )
                ],
              ),
              SizedBox(height: 20),
              _data != null && _data.isNotEmpty ? unpackDbData(_data) :
Container(),
            ],
          ),
        )
    );
  }
}

//stranica za skeniranje
class _HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => new _HomePageState();
}

class _HomePageState extends State<_HomePage> {
  //deklariranje globalnih varijabli klase
  bool showImage = false;
  dynamic _imageFile;
  Size _imageSize;
  dynamic _scanResults;
  List _codesData;
```

```
  //metoda dobiva sirovi tekst OCR-an sa slike, iz njega izvlači E-kodove i
vraća podatke o njima
  Future<void> _retrieveCodesInfo() async {
    if (_scanResults == null) {
      return;
    }

    //izvlačenje teksta iz blokova i pohranjivanje na listu textList
    var textList = new List();
    for (TextBlock block in _scanResults.blocks) {
      for (TextLine line in block.lines) {
        textList.add(line.text);
      }
    }

    //filtriranje E-kodova iz teksta
    var eCodes;
    if (textList.length>0) {
      eCodes = filterM2A1(textList);
    } else {
      eCodes = new List();
      eCodes.add("No text detected");
    }

    //dobivanje podataka o E-kodovima iz baze i pohranjivanje podataka na
listu
    var codeData = List();
    for (String eCode in eCodes) {
      var data = await getDataFromDatabase(DefaultAssetBundle.of(context),
eCode);
      if (data != null) codeData.add(data);
    }

    //vraćanje rezultata
    setState(() => _codesData = codeData);
  }

  //metoda dobavlja sliku, njezinu veličinu te poziva OCR
  Future<void> _getAndScanImage() async {
    FocusScope.of(context).requestFocus(new FocusNode());
    //osnovne varijable
    setState(() {
      _imageFile = null;
      _imageSize = null;
    });

    //otvara se kamera te kad korisnik snimi fotografiju, ona se posprema u
imageFile
    dynamic imageFile = await ImagePicker.pickImage(source:
ImageSource.camera);
    if (Platform.isAndroid) {
      imageFile = await FlutterExifRotation.rotateImage(path: imageFile.path);
    }

    //zbog greške u libraryu, moguće je da aplikacija puca kod fotografiranja

    _imageFile = imageFile;
```

```
      //dobavljanje veličine slike i pozivanje OCR-a
    if (imageFile != null) {
      _getImageSize(imageFile);
      _ocrImage(imageFile);
    }

    setState(() => _imageFile = imageFile);
  }

  //metoda dobavlja veličinu slike koja je potrebna drugim funkcijama
  Future<void> _getImageSize(dynamic imageFile) async {
    final Completer<Size> completer = Completer<Size>();

    final Image image = Image.file(imageFile);
    image.image.resolve(ImageConfiguration()).addListener(
      (ImageInfo info, bool _) {
        completer.complete(Size(info.image.width.toDouble(),
info.image.height.toDouble()));
      },
    );

    final Size imageSize = await completer.future;
    setState(() => _imageSize = imageSize);
  }

  //metoda poziva OCR iz Firebase paketa alata i njime izvlači tekst iz
snimljene fotografije
  Future<void> _ocrImage(dynamic imageFile) async {
    setState(() => _scanResults = null);

    FirebaseVisionDetector detector =
FirebaseVision.instance.textRecognizer();

    final FirebaseVisionImage visionImage =
FirebaseVisionImage.fromFile(imageFile);
    final dynamic results = await detector.detectInImage(visionImage) ??
<dynamic>[];

    setState(() => _scanResults = results);

    //pozivanje metode za dobivanje podataka o kodovima
    await _retrieveCodesInfo();
  }
```

```
//metoda kao parametar prima podatke o određenom kodu, a zatim ih ispisuje u
obliku kartice na ekranu
  Widget _buildOneCard(String eCode, String name, String type, String
description, int danger) {
    //dobavljanje podataka
    var printText;
    var printColor;
    if (danger != null) {
      List dataList = dangerList(danger);
      printText = dataList[0];
      printColor = dataList[1];
    }

    //sastavljanje kartice
    return Card(
        margin: EdgeInsets.fromLTRB(0, 10, 0, 10),
        child: Padding(
            padding: EdgeInsets.all(14),
            child: Column(crossAxisAlignment: CrossAxisAlignment.start,
children: <Widget>[
              Padding(
                  padding: EdgeInsets.only(bottom: 10),
                  //broj i naziv aditiva
                  child: Text("$eCode - $name", style: TextStyle(fontWeight:
FontWeight.bold))),
              Text("$type"),//tip aditiva
              Visibility(
                child:
                Padding(
                    padding: EdgeInsets.only(top: 5),
                    child: Text(description ?? '', style: TextStyle(fontStyle:
FontStyle.italic))//opis aditiva
                ),

                visible: description != null,
              ),
              Visibility(
                child:
                Padding(
                    padding: EdgeInsets.only(top: 5),
                    child: Text("$printText", style: TextStyle(color:
printColor))//ocjena štetnosti
                ),

                visible: danger != null,
              ),
            ])));
  }
```

```
//metoda sastavlja sadržaj stranice sa rezultatima analize slike
  _buildContent() {
    var scoreList = new List();
    if (_imageSize == null || _scanResults == null) {
      return Container(child: Center(child: Text('Text scan in
progress...')));
    }

    //sastavljanje liste kartica s podacima
    var cards = new List<Widget>();
    if (_codesData != null) {
      for (Map map in _codesData) {
        cards.add(_buildOneCard(map['code'], map['name'], map['type'],
map['description'], map['danger']));
        scoreList.add(map['danger']);
      }
    }

    //ispis rezultata bodovanja proizvoda
    String rating = dangerLabel(scoreList)[0];
    Color ratingColor = dangerLabel(scoreList)[1];
    final title = Text("Product contents analysis:", style:
TextStyle(fontSize: 16));
    final analysis = Text("$rating", style: TextStyle(color: ratingColor,
fontSize: 20, fontWeight: FontWeight.bold));

    //definiranje Containera, odnosno sadržaja stranice
    return Container(
      child: ListView(children: <Widget>[
        _codesData != null && _codesData.isNotEmpty ?
          Column(
            children: <Widget>[
              //postavljanje elemenata
              showImage ? _buildImage(): Container(),
              Padding(padding: EdgeInsets.only(top: 20, left: 12), child:
title),

              //ispis ocjene proizvoda
              Align (
                alignment: Alignment.topCenter,
                child: Padding(
                  padding: EdgeInsets.only(
                    top: 15,
                    bottom: 10,
                    left: 15,
                    right: 15
                  ),
                  child: analysis
                ),
              ),

              //ispis liste sa karticama
              Padding(
                padding: EdgeInsets.fromLTRB(10, 4, 10, 4),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: cards
                )
              )
            ],
          )
```

```
        //poruka o grešci u slučaju da niti jedan kod nije pronađen
        : Container(padding: EdgeInsets.only(top: 150), child: Center(child:
Text('No E-numbers detected in product label'))),
    ]));
  }

  //metoda stvara završnu sliku i vraća je u obliku widgeta koji vraća metodi
_buildContent
  Widget _buildImage() {
    //pozivanje algoritma za iscrtavanje oznaka na fotografiji
    final painter = CustomPaint(painter: TextDetectorPainter(_imageSize,
_scanResults));

    //stvaranje Containtera koji sadrži sliku
    return Container(
        constraints: BoxConstraints(
          minHeight: 400,
          minWidth: 400
        ),
      decoration: BoxDecoration(
        color: Colors.orange.shade100,
          image: DecorationImage(image: Image.file(_imageFile).image, fit:
BoxFit.fill)
      ),
      child: _imageSize == null || _scanResults == null ? Center(child:
Text('Scanning label...')) : painter
    );
  }

  @override
  //metoda uzima sadržaj stranice koji joj dobavlja _buildContent i sastavlja
gotovu stranicu od njega
```

```
 Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomPadding: false,
      appBar: AppBar(title: Text('Health Advisor', style: TextStyle(color:
Colors.white, )), actions: <Widget>[
        //definiranje gumba za togglanje prikaza slike
        _imageSize != null ? IconButton(
            icon: Icon(Icons.image, color: Colors.white),
            onPressed: () {
              setState(() {
                showImage = !showImage;
              });
            })
            :Container()//u slučaju da fotografija još nije snimljena, gumb se
niti ne pojavljuje
      ]),
      //natpis koji se prikazuje u slučaju da fotografija još nije snimljena
      body: _imageFile == null ? Center(
        child: Padding (
          padding: EdgeInsets.only(left: 20, right: 20),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text("Please scan the product label."),
              Padding(padding: EdgeInsets.only(top: 20),),
              Text("To begin, press the camera button. Next, photograph a food
product label which contains E-numbers (like E100). Make sure that there is
enough light and that the camera is focused on the text.", textAlign:
TextAlign.center, style: TextStyle(fontSize: 12, color: Colors.black45),)
            ],
          ),
        ),
      )
      : _buildContent(),
      //stvaranje gumba za uključivanje kamere
      floatingActionButton: FloatingActionButton(onPressed: _getAndScanImage,
child: Icon(Icons.photo_camera, color: Colors.white), backgroundColor:
Colors.indigoAccent,));
  }
}
```

db.dart:

```
import 'dart:io';

import 'package:flutter/services.dart';
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

//funkcija za operacije s bazom podataka, kao parametre prima context podatke
i string koda koji treba pretražiti u bazi podataka
Future getDataFromDatabase(AssetBundle rootBundle, String eCode) async {
  //deklariranje varijabli
  var databasesPath = await getDatabasesPath();
  var path = join(databasesPath, "tmp_bazaM2.db");
  Database db;

  //otvaranje privremene baze
  try {
    db = await openDatabase(path, readOnly: true);
  } catch (e) {
    print("Error $e");
  }

  if (db == null) {
    //kopiranje trajne baze u radni direktorij radi zaštite od modificiranja
    ByteData data = await rootBundle.load(join("assets", "ecodesM1.db"));
    List<int> bytes = data.buffer.asUint8List(data.offsetInBytes,
data.lengthInBytes);
    await new File(path).writeAsBytes(bytes);

    //otvaranje baze
    db = await openDatabase(path, readOnly: true);
  }

  //TODO pretraga po imenu
  //upit na bazu i popisivanje rezultata
  String sql = "select * from codes where code = ?"; //databinding štiti od
sql injectiona
  List result = await db.rawQuery(sql, [eCode]);
  print (result);

  //vraćanje rezultata ako postoji odgovor na upit
  return result != null && result.isNotEmpty ? result.first : null;
}
```

utils.dart:

```
import 'dart:core';

import 'package:flutter/material.dart';

//filter za E-kodove, kao parametar prima listu stringova, vraća E-kodove koji se nalaze u tim stringovima
List filterM2A1 (List original) {
  //deklariranje potrebnih varijabli
  var finalList = new List ();
  var eIndexes = new List ();
  String currentElement;
  String line;
  String line2;
  String eStr;
  String finalStr;
  int index;

  //iz svake linije teksta se miču razmaci, na kraj se dodaje string za popunjavanje (zbog mogućnosti greške s indeksom), te se indeksiraju sva velika slova E
  for (line in original) {
    line2 = line.replaceAll(" ", "");
    currentElement = line2+"xxxxx";
    if (currentElement.contains("E")) {
      for (int counter = 0; counter < currentElement.length; counter++) {
        if (currentElement[counter] == "E") {
          eIndexes.add(counter);
        }
      }
      //izdvajaju se kodovi iz stringa, provjerava se jesu li važeći, ako jesu, stavljaju se na listu za izlaz
      for (index in eIndexes) {
        eStr = currentElement.substring(index+1, index+4);
        if (int.tryParse(eStr) != null) {
          finalStr = currentElement.substring(index, index+4);
          finalList.add(finalStr);
        } else if (int.tryParse(eStr.substring(0, 2)) != null) {
          finalStr = currentElement.substring(index, index + 3);
          finalList.add(finalStr);
        }
      }
    }
    //čišćenje liste prije novog okreta petlje
    eIndexes.clear();
  }
  //uklanjanje duplikata s liste
  return finalList.toSet().toList();
}

//funkcija za sastavljanje poruka o štetnosti aditiva, koristi se kod sastavljanja kartica s podacima
List dangerList (int danger) {
  //deklariranje varijabli
  List endList = new List();
  String endString = "";
  Color endColor = Colors.black;
```

```
  if (danger == null) {
    return null;

  } else if (danger == -3) {
    endString = "Very healthy, seek intake";
    endColor = Colors.lightBlue;

  } else if (danger == -2) {
    endString = "Healthy";
    endColor = Colors.lightGreen;

  } else if (danger == -1) {
    endString = "Beneficial, possibly over time";
    endColor = Colors.green;

  } else if (danger == 0) {
    endString = "Harmless but not beneficial";
    endColor = Colors.brown;

  } else if (danger == 1) {
    endString = "Unhealthy in large quantities, possibly irritating";
    endColor = Colors.amber;

  } else if (danger == 2) {
    endString = "Unhealthy, avoid if possible";
    endColor = Colors.orange;

  } else if (danger == 3) {
    endString = "Serious health risk, do not consume";
    endColor = Colors.red;
  }

  endList.add(endString);
  endList.add(endColor);
  return endList;
}

//funkcija na temelju bodova opasnosti pojedinih E-kodova procjenjuje je li
proizvod zdrav ili štetan
List dangerLabel (List intList) {
  int score = 0;
  var element;
  //u slučaju da ocjenjivanje ne uspije, aplikacija će ispisati da ne može
ocijeniti proizvod
  String returnString = "Insufficient data to evaluate this product.";
  Color returnColor = Colors.black;
  var returnList = new List();
```

```
  //zbrajanje bodova
  for (element in intList) {
    //u slučaju da barem jedan kod ima ocjenu štetnosti 3, proizvod se
automatski procjenjuje kao nezdrav
    print (element);
    if (element == null) {
      element = 0;
    } else if (element == 3) {
      returnString = "This product is a serious health risk, do not consume
it.";
      returnColor = Colors.red;
      returnList.add(returnString);
      returnList.add(returnColor);
      return returnList;

    } else {
      score = score + element;
    }
  }

  //ocjenjivanje proizvoda
  if (score <= 1 && score >= -1) {
    returnString = "This product is safe for consumption.";
    returnColor = Colors.brown;

  } else if (score < -1 && score >= -3) {
    returnString = "This product is moderately healthy.";
    returnColor = Colors.green;

  } else if (score < -3) {
    returnString = "This product is very healthy.";
    returnColor = Colors.lightGreen;

  } else if (score > 1 && score <= 3) {
    returnString = "This product should not be consumed in larger quantites";
    returnColor = Colors.amber;

  } else if (score > 3) {
    returnString = "This product is unhealthy, consumption is not
recommended.";
    returnColor = Colors.orange;
  }

  returnList.add(returnString);
  returnList.add(returnColor);
  return returnList;
}
```

painter.dart:

```
import 'package:firebase_ml_vision/firebase_ml_vision.dart';
import 'package:flutter/material.dart';

//funkcija za iscrtavanje oznaka teksta na obrađenoj slici, zapravo je
modifikacija importane funkcije
class TextDetectorPainter extends CustomPainter {
  TextDetectorPainter(this.absoluteImageSize, this.visionText);

  final Size absoluteImageSize;
  final VisionText visionText;

  //modifikacija postojeće metode paint
  @override
  void paint(Canvas canvas, Size size) {
    final double scaleX = size.width / absoluteImageSize.width;
    final double scaleY = size.height / absoluteImageSize.height;

    //podešavanje veličine slike
    Rect scaleRect(TextContainer container) {
      return Rect.fromLTRB(
        container.boundingBox.left * scaleX,
        container.boundingBox.top * scaleY,
        container.boundingBox.right * scaleX,
        container.boundingBox.bottom * scaleY,
      );
    }

    //podešavanje opcija za način bojanja
    final Paint paint = Paint()
      ..style = PaintingStyle.stroke
      ..strokeWidth = 2.0;

    //bojanje blokova teksta u crveno
    for (TextBlock block in visionText.blocks) {
      //bojanje linija u žuto
      for (TextLine line in block.lines) {
        //bojanje razmaka između riječi u zeleno
        for (TextElement element in line.elements) {
          paint.color = Colors.green;
          canvas.drawRect(scaleRect(element), paint);
        }

        paint.color = Colors.yellow;
        canvas.drawRect(scaleRect(line), paint);
      }

      paint.color = Colors.red;
      canvas.drawRect(scaleRect(block), paint);
    }
  }

  //potvrda da se slika nije promijenila, ako je, ponovno se boja
  @override
  bool shouldRepaint(TextDetectorPainter oldDelegate) {
    return oldDelegate.absoluteImageSize != absoluteImageSize ||
        oldDelegate.visionText != visionText;
  }
}
```