SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2751

ANALIZA OSOBNE POTROŠNJE SKENIRANJEM RAČUNA

Nimai Vadas

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2751

ANALIZA OSOBNE POTROŠNJE SKENIRANJEM RAČUNA

Nimai Vadas

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

DIPLOMSKI ZADATAK br. 2751

Pristupnik:	**Nimai Vadas (0036509003)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	prof. dr. sc. Krešimir Fertalj

Zadatak: **Analiza osobne potrošnje skeniranjem računa**

Opis zadatka:

Projektirati i izgraditi aplikaciju koja će korisniku omogućiti praćenje osobne potrošnje analizom računa i kupljenih proizvoda. Proučiti standarde za obilježavanje proizvoda te dostupna aplikacijska programska sučelja za pretraživanje proizvoda u bazama podataka na Internetu. Proučiti dostupne aplikacije za skeniranje računa. Ugraditi funkcionalnost za analizu računa optičkim prepoznavanjem znakova (optical character recognition, skraćeno OCR). Evidentiranje proizvoda s računa olakšati skeniranjem štapićastog koda s proizvoda te pretraživanjem baza podataka na Internetu. Omogućiti statističku obradu podataka s računa, dinamički prilagodljivim prikazom potrošnje po vremenskim razdobljima i kupljenim proizvodima.

Rok za predaju rada: 27. lipnja 2022.

SADRŽAJ

1. Uvod

Kupovina je svakodnevna životna pojava prosječnog čovjeka. Ljudi kupuju nužna i luksuzna dobra koja su im potrebna za njihovu egzistenciju ili uživanje. Također, većina ljudi ima određeni budžet koji je namijenjen za potrošnju kućanstva kojeg je poprilično teško pratiti zbog ubrzanog načina života. Postoje digitalni "pratitelji" potrošnje ugrađeni u bankovne aplikacije ukoliko potrošač plaća karticom, no takvi pokazatelji nisu lako opširni, a plaćanje gotovinom je nemoguće pratiti. Osim generalne potrošnje po npr. vremenskom razdoblju bilo bi korisno vidjeti i potrošnju po specifičnim proizvodima ili po trgovinama kako bi korisnici mogli vidjeti koliko su potrošili na hranu u jednom mjesecu ili na neku drugu kategoriju potrošnje koju sami stvore.

Praćenje takve potrošnje nije jednostavno jer korisniku oduzima mnogo vremena. Idealno rješenje za taj problem je skenirati račun iz trgovine i izvući informacije bitne za potrošnju kao što su datum, ukupan iznos i kupljeni proizvodi. Izvedba takvog rješenja nije trivijalna, no postoje alati i sustavi koji se mogu iskoristiti za lakšu realizaciju.

Stoga je u ovom radu razvijena mobilna aplikacija za praćenje osobne potrošnje u trgovinama koja korisniku omogućava skeniranje računa za brži i lakši unos podataka o svojim kupnjama, te stvaranje kategorija proizvoda za pregled potrošnje.

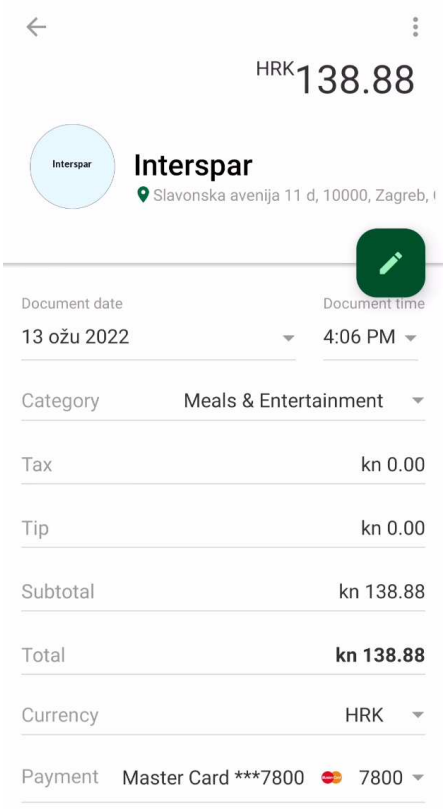
2. Analiza postojećih rješenja

Praćenje osobne potrošnje nije nova ideja te postoji mnoštvo aplikacija koje pokušavaju riješiti taj problem. Većinom se oslanjaju na korisnikov ručni unos podataka, a rijetko koje implementiraju skeniranje samog računa za dobivanje podataka. U nastavku je provedena osnovna analiza dvije aplikacije koje implementiraju skeniranje računa.

2.1. Veryfi Receipts OCR & expenses

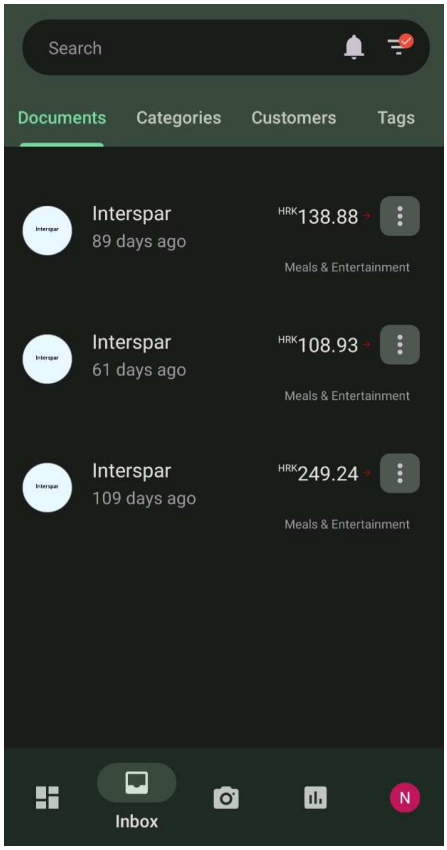
Veryfi je mobilna aplikacija za skeniranje računa i faktura pomoću OCR-a – optičkog prepoznavanja znakova (engl. *optical character recognition*) te izvlačenje informacija iz skeniranog dokumenta [11]. Primjer skeniranja računa prikazan je na slici 2.1. Aplikacija iz podataka dobivenih skeniranjem računa izvlači informacije o ukupnom iznosu, datumu i vremenu, porezu, načinu plaćanja, vrsti dokumenta, nazivu trgovine, broju računa i kategoriji troška (slike 2.2 i 2.3).

Slika 2.1: *Verify*: skeniranje računa

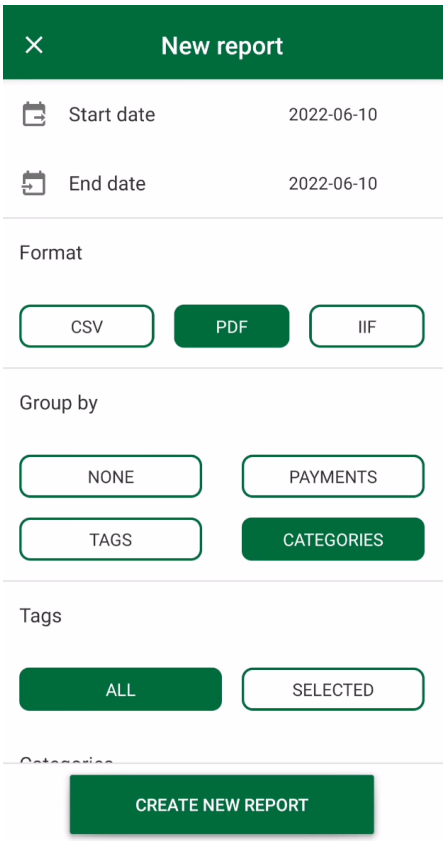


Slika 2.2: *Verify*: prikaz rezultata skeniranja

Aplikacija omogućuje praćenje ukupne potrošnje po vremenskim periodima i stvaranje izvješća o potrošnji po korisnički odabranim kategorijama kao na slici 2.4. Aplikacija ne prati kupljene proizvode te njih nije moguće naknadno dodati po računu.



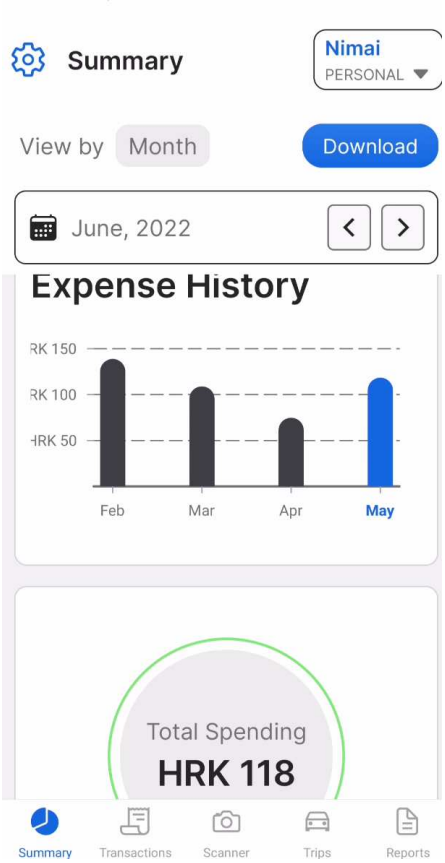
Slika 2.3: *Verify*: skenirani računi



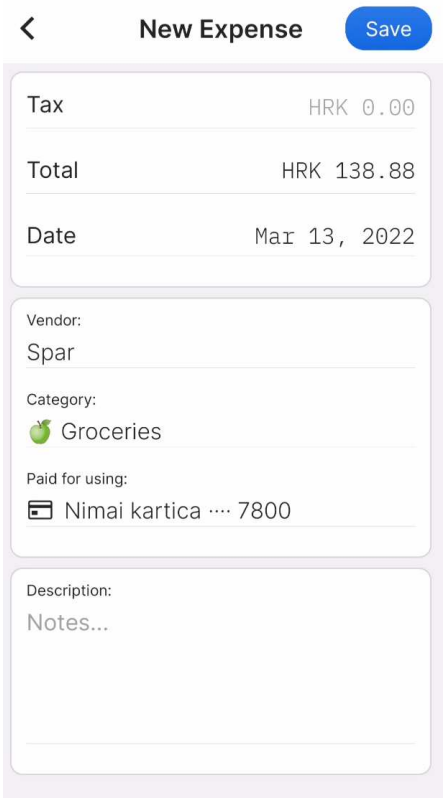
Slika 2.4: *Verify*: stvaranje izvješća

2.2. Receipt Scanner: Easy Expense

Easy Expense je mobilna aplikacija za praćenje osobne potrošnje koja pruža mogućnost skeniranja računa za lakše izvlačenje informacija [4]. Aplikacija iz skeniranog računa bilježi datum i ukupan trošak, a korisnik može dodati podatke o trgovini, kategoriji i opis računa. Korisnik može pregledavati svoje obavljene transakcije i račune koje je unio.

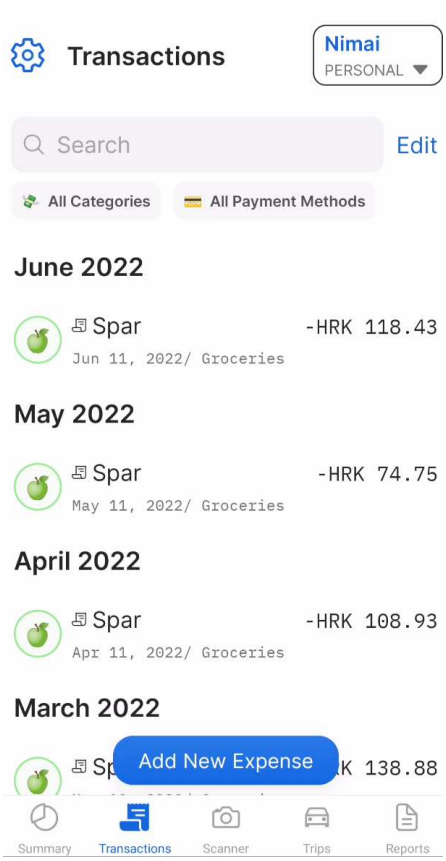


Slika 2.5: *Easy Expense*: prikaz mjesečne potrošnje



Slika 2.6: *Easy Expense*: prikaz rezultata skeniranja

Aplikacija pruža statistički uvid u potrošnu po vremenskim periodima, trgovinama ili kategorijama računa. Aplikacija također pruža mogućnost bilježenja putovanja i troškova tih putovanja prema unesenim vrstama koje zadaje korisnik.



Slika 2.7: *Easy Expense*: prikaz transakcija

Slika 2.8: *Easy Expense*: prikaz izvješća

3. Specifikacija zahtjeva

3.1. Funkcionalni zahtjevi

Aplikacija korisniku mora omogućiti sljedeće funkcionalnosti:

- ugrađeno slikanje računa za daljnje procesiranje,
- očitavanje teksta računa,
- izvlačenje informacija iz očitanog teksta računa o:
  - računu:
    - broj računa,
    - datum računa,
    - ukupan iznos na računu,
  - proizvodima:
    - skraćeni naziv proizvoda prikazan na računu,
    - količinu kupljenog proizvoda,
    - cijenu proizvoda,
- dodavanje informacija s računa koje nisu unesene obradom računa,
  - informacije o:
    - proizvodima,
    - trgovini,
    - računu,
- ispravljanje informacija o računu koje su krivo očitane i brisanje krivih informacija o proizvodima,
- brisanje unesenih podataka o računima,
- dopunjivanje naziva i slike proizvoda skeniranjem štapićastog koda (engl. *barcode*) na proizvodu,
- dodavanje vlastite slike proizvodu,

- upravljanje korisnički definiranim tipovima proizvoda za kategorizaciju ,
- dodjeljivanje definiranih tipova proizvodima,
- upravljanje podatcima o trgovinama,
- pregled očitanog teksta računa,
- pregled slike računa,
- pregled slike proizvoda,
- pregled svih kupnji nekog proizvoda po datumima
- pregled statistike o potrošnji po:
    - vremenskim periodima,
    - proizvodima,
    - vrstama proizvoda,
    - trgovinama.

3.2. Nefunkcionalni zahtjevi

- aplikacija treba moći obraditi račun u kratkom vremenu od nekoliko sekundi,
- korisničko sučelje aplikacije mora biti jednostavno i intuitivno za korištenje.

4. Analiza tehnologija za skeniranje i obrade dobivenih podataka

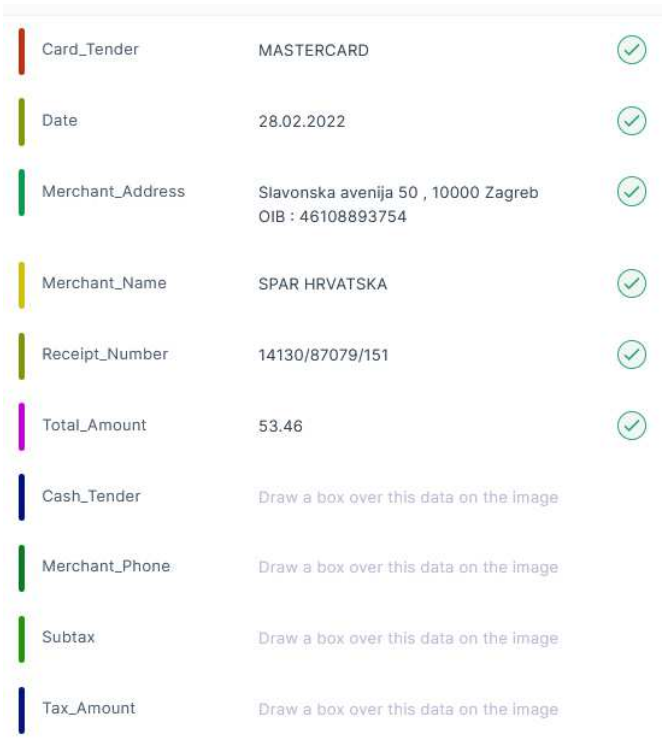
Postoji nekoliko načina za obraditi račun i dobiti tražene podatke iz njega. Postoje gotove usluge koje obrađuju račun i vraćaju tražene podatke. Osim gotovih usluga moguće je napraviti model dubokog učenja koji na temelju slike, ili tekstualnog zapisa računa klasificira podatke s varirajućim vjerojatnostima. Još jedan način za obradu i dobivanje podataka iz računa je korištenje kombiniranih načina za obrada teksta dobivenog optičkim prepoznavanjem znakova. U nastavku su navedeni načini obrađeni detaljnije.

4.1. Postojeće usluge

4.1.1. Nanonets Receipt OCR

Nanonets Receipt OCR je usluga tvrtke *Nanonets* koji koristi umjetnu inteligenciju i strojno učenje za ekstrakciju podataka iz određenih polja računa [7]. Usluga omogućava jednostavnu izgradnju prilagođenog OCR modela na temelju svojih podataka, u ovom slučaju slika računa. Postavljanje usluge je vrlo jednostavno i zahtjeva od korisnika da označi, tj. ispravi očitane podatke. Označavanje se radi grafički na samoj slici, a označenom polju se dodaje jedna od zadanih oznaka kao na slikama 4.1 i 4.2.

Slika 4.1: *Nanonets*: označavanje polja [7]



Slika 4.2: *Nanonets*: očitana polja [7]

Besplatna verzija usluge pruža mogućnost označavanja deset različitih oznaka: datum, ukupan iznos, naziv trgovine, adresa trgovine, broj računa, broj trgovca i još neke podatke. Nakon što korisnik dodijeli svoje primjere modelu može početi koristiti uslugu za automatsku obradu daljnjih računa. Postoji više načina za integraciju usluge, a neki od njih su slanje podataka na e-mail adresu, poziv na aplikacijsko sučelje i uvoz podataka preko *Google Drive* servisa. Rezultat upita moguće je primiti kao odgovor na poziv aplikacijskog sučelja ili je moguće spremiti rezultate na nekoliko drugih servisa kao što su baze podataka.

U sam tijek rada usluge moguće je uvesti i dodatne korake poboljšavanja podataka koji omogućuju različite promjene formata, pretraživanje baza podataka, kategorizaciju ili proizvoljan *Python* kod koji će se izvršavati nad podatcima. Moguće je uvesti i korak ručnog pregleda gdje jedan član tima provjerava valjanost podataka u odabranim poljima.

U besplatnoj *Starter* verziji usluge nije moguće očitati proizvode s računa dok *Pro* verzija pruža mogućnost čitanja tablica što bi se moglo iskoristiti za njihovo očitavanje. *Starter* verzija je također ograničena na obradu 100 računa na mjesec, a *Pro* verzija

podiže taj broj na 5000. Pošto *Starter* verzija nema mogućnost očitavanja proizvoda s računa, jedina opcija koju bi se moglo iskoristiti za ovaj projekt je *Pro* verzija, no ona se plaća 500 dolara na mjesec, a to nije prikladno za ovaj rad.

4.1.2. Microblink BlinkReceipt SDK

Tvrtka *Microblink* nudi komplet za razvoj programske podrške (engl. *software development kit, SDK*) *BlinkReceipt* koji omogućava razvoj sustava i aplikacija za skeniranje računa [6]. *Microblink* tvrdi da pomoću njihovog SDK-a moguće postići skeniranje računa s 98% točnosti bez ikakve poslužiteljske obrade podataka, već se sve izvodi lokalno na klijentu. Tvrde da iz računa očitavaju sve bitne podatke o trgovcu i kupnjama, uključujući i proizvode. Unatoč tome, nisam bio u mogućnosti isprobati sam SDK jer je za probnu verziju potrebno dogovoriti sastanak s tvrtkom.

4.2. Duboko učenje

Jedan od novijih načina za rješavanje problema ekstrakcije informacija iz računa je pomoću dubokog učenja. Prednost dubokog učenja je što može naučiti prepoznavati podatke direktno iz slike bez eksplicitnog postavljanja pravila. Također pruža mogućnost dobre generalizacije po različitim rasporedima podataka na računu ako postoje primjeri za učenje s tim rasporedima.

Postoji nekoliko znanstvenih radova koji obrađuju tematiku korištenja različitih načina dubokog učenja za izvlačenje informacija iz dokumenata [15], [13] ili specifično računa [12]. Sama implementacija takvog načina obrade računa nije trivijalna te je problem za sebe, uz to i izvan okvira ovog rada pa je suvišno ulaziti u detalje o teoriji.

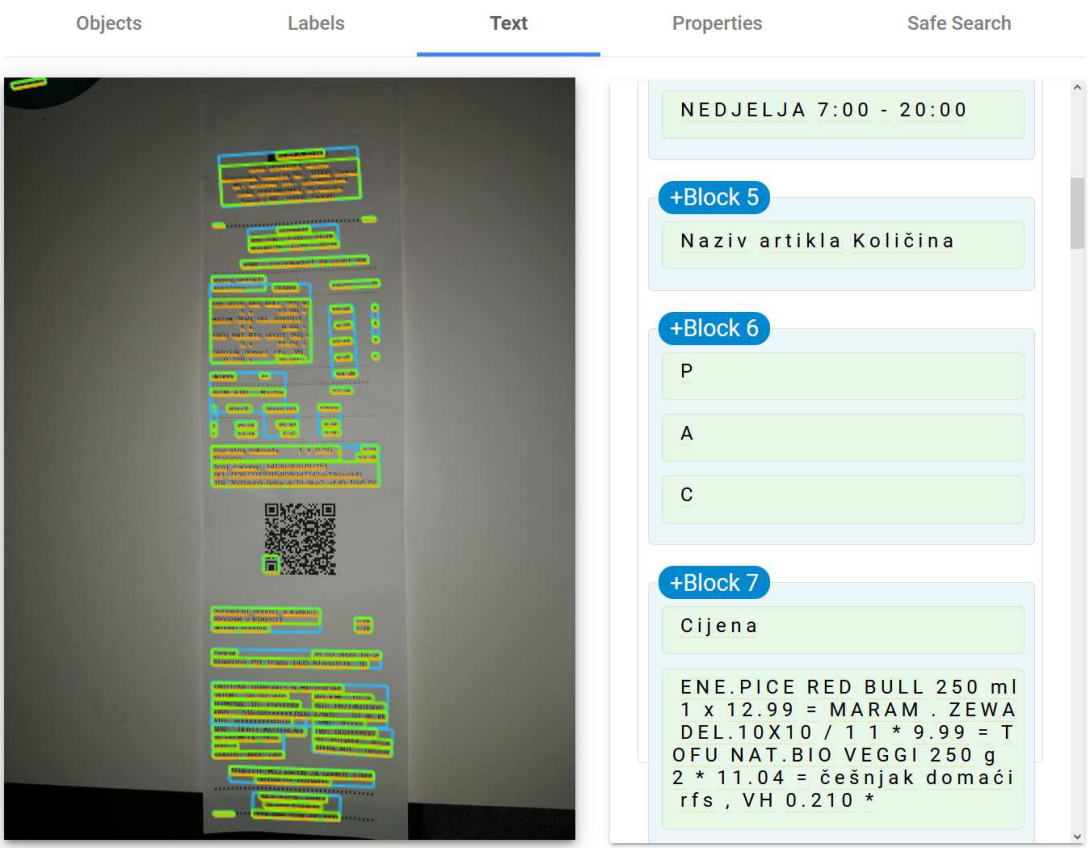
4.3. Optičko prepoznavanje znakova i daljnja obrada

Optičko prepoznavanje znakova (engl. *optical character recognition - OCR*) definira programske načine izvlačenja teksta iz slike. Koristi se za digitalizaciju papirnatih podataka s ciljem lakšeg skladištenja, uređivanja, pretraživanja, a i korištenja u daljnje svrhe.

Bez ulaženja u detalje izvedbe OCR-a, postoje mnogi servisi koji nude programska sučelja ili pakete koji vrše OCR. Jedan takav je *Tesseract*, besplatan paket otvorenog koda koji ima dugu povijest pružanja OCR-a [10]. *Tesseract* pruža relativno veliku mogućnost prilagođavanja, podržava očitavanje više od 100 jezika i podržava različite

formate slika. Također je moguće prilagoditi model treniranjem s novim skupom podataka. Iako je prilagodljiv i besplatan, očitavanje računa ne daje idealne rezultate s zadanim postavkama i modelom.

Još jedna usluga OCR-a je *Google Cloud Vision API*. Omogućava prepoznavanje teksta u dva načina, prvi je prepoznavanje teksta (*TEXT_DETECTION*) koji prepoznaje tekst na slikama i vrlo je robustan kada se radi o uvjetima slika, a drugi je prepoznavanje teksta u dokumentima (*DOCUMENT_TEXT_DETECTION*) koji je prilagođen za prepoznavanje teksta u dokumentima s puno teksta, npr. knjige. Primjer skeniranja računa prikazan je na slici 4.3. Usluga pruža relativno dobru kvalitetu očitavanja teksta, no velika mana je što se poziva udaljeno i obrada slike traje 10-15 sekundi, što nije idealno za brzu obradu.



Slika 4.3: Primjer skeniranja računa s *Google Cloud Vision API-jem* [5]

Treća usluga, koja se koristi u ovom radu je *ML Kit* koji pruža sličnu kvalitetu očitavanja kao i *Google Cloud Vision API*, a velike prednosti su što nije potreban udaljeni poziv ni veza na internet za očitavanje teksta i to što sama obrada traje svega par sekundi. Više o *ML Kitu* i izvedbi opisano je u kasnijem poglavlju.

Nakon očitavanja teksta nekom od usluga, za izvlačenje samih informacija mo-

guće je koristiti regularne izraze na temelju poznatih uzoraka u računu. Datum i broj računa je vrlo jednostavno očitati jer imaju jedinstven uzorak, pa je moguće oblikovati jednostavan regularni izraz (npr. za datum `"([\d]2\.[\d]2\.[\d]4)"`). Kad je blok teksta s informacijama o proizvodu dobro očitan, također je jednostavno dobiti te informacije pošto se uzorak ponavlja za svaki proizvod. Ovaj način nije idealan ni skalabilan, no za potrebe ovog rada je sasvim dovoljan.

5. Arhitektura rješenja

Ovaj rad je razvijen kao mobilna aplikacija za Android operacijski sustav, napisan je u programskom jeziku Kotlin, a razvijen u razvojnom okruženju Android Studio. Za razvoj aplikacije korištene su razne biblioteke iz skupa biblioteka Android Jetpack, kao i nekoliko vanjskih biblioteka. Korisničko sučelje izvedeno je skupom alata Jetpack Compose i korištena je baza podataka SQLite realizirana preko biblioteke Room. Razvoj same aplikacije okvirno se drži načela čiste arhitekture uz obrazac MVVM (*model-view-viewmodel*). Aplikacija je sačinjena od tri sloja: podatkovnog, prezentacijskog i sloja poslovne logike.

5.1. Razvoj Android aplikacija

Za razvoj aplikacija za Android operacijske sustave postoji mnogo preporuka i smjernica koje su opisane na službenim web stranicama [1]. Preporučena razvojna okolina je Android Studio i programski jezik Kotlin.

5.1.1. Android Studio

Android Studio je službena razvojna okolina za razvoj aplikacija na Android operacijskim sustavima. Baziran je na IntelliJ IDEA razvojnoj okolini, a ažuriraju ga Google i JetBrains. Pruža mnoge funkcionalnosti potrebne za razvoj Android aplikacija, a neke od njih su ugrađeno dopunjavanje i refaktoriranje programskog koda specifičnog za Android, ugrađena podrška za dodavanje biblioteka, ugrađeni emulator tj. virtualni Android uređaj za pokretanje i testiranje aplikacija, statičku analizu programskog koda s ciljem pronalaženja problema s performansama, kompatibilnosti i mnogim drugim. Iako je preporučeni programski jezik za razvoj u okolini Android Studio Kotlin, okolina podržava jezike kao što su Java, C++, a i dodavanje proširenja za podršku mnogih drugih [2].

Programski jezik Kotlin

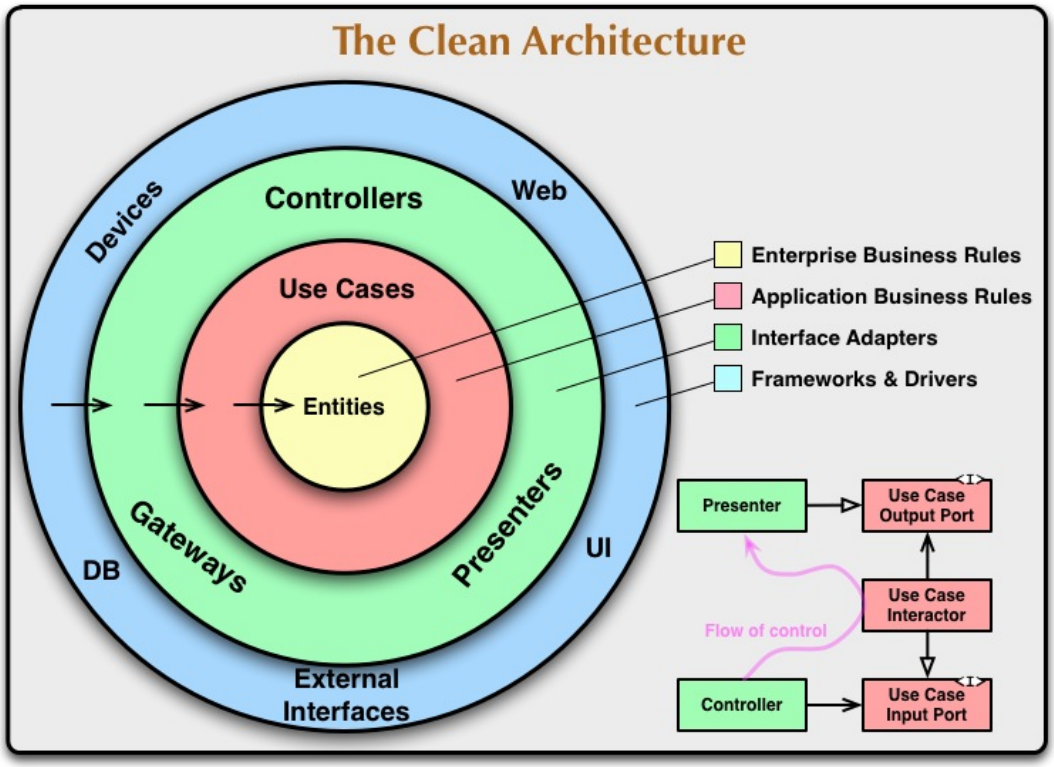
Kotlin je statički pisan programski jezik opće namjene i idealan je za razvoj mobilnih aplikacija na Android operacijskom sustavu zbog službene podrške Googlea, jednostavne i intuitivne sintakse, sigurnosti i integracije s postojećim JVM (*Java Virtual Machine*) bibliotekama.

5.1.2. Android Jetpack

Android Jetpack je skup komponenata, alata i smjernica koji omogućava razvoj modernih aplikacija za Android sustave prema najboljim praksama. Komponente dolaze u obliku biblioteka i međusobno su neovisne, iako su dizajnirane s međusobnim radom na umu. Biblioteke su službeno podržane i razvijene od strane Googlea i pružaju raznovrsne usluge, od upravljanja bazom podataka do dodavanja gotovih komponenti korisničkog sučelja.

5.1.3. Čista arhitektura i obrazac MVVM

Čista arhitektura (engl. *clean architecture*) je moderan način razvoja programske podrške koji je osmislio Robert C. Martin, a zasniva se na kružno slojevitosti sustava kod koje se vanjski slojevi oslanjaju prema unutarnjima, primjer na slici 5.1. Ključno pravilo ove arhitekture je da su slojevi ovisni isključivo prema sredini. Drugim riječima, unutarnji slojevi ne smiju referencirati ništa što se nalazi u vanjskim slojevima. Držanje načela ove arhitekture omogućava pregledniji, jednostavniji, brži i skalabilniji razvoj programske podrške, jednostavnu zamjenu komponenti i slojeva, te mogućnost testiranja individualnih slojeva, primarno poslovne logike, bez potrebe za uključivanjem ostalih vanjskih slojeva.



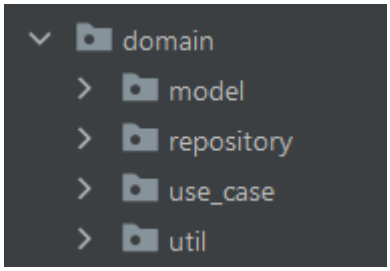
Slika 5.1: Primjer slojeva u čistoj arhitekturi [14]

Za razvoj aplikacija na Android sustavima postoji urođena podrška za obrazac MVVM zbog mnoštvo biblioteka koje pojednostavljuju razvoj pojedinih slojeva. Sam obrazac definira tri sloja: *model*, *view* i *viewmodel*. U ovom radu model predstavlja domenski model koji definira podatke, *view* predstavlja samo korisničko sučelje aplikacije i njegov opis. *ViewModel* sadrži podatke koji se prikazuju na korisničkom sučelju i bavi se logikom svih događaja koji dolaze sa korisničkog sučelja.

Aplikacija razvijena u ovom radu generalno se drži gore opisanih obrazaca, a okvirno je podijeljena na 3 sloja koji u sebi sadrže podslojeve. Ti slojevi su sloj domene, podatkovni sloj i prezentacijski sloj, a detaljnije su opisani u nastavku.

5.2. Sloj domene

Sloj domene je centralni sloj o kojemu ovise ostali. Sloj je podijeljen kao na slici 5.2, sadrži pakete *model*, *repository*, *use_case* i *util*.



Slika 5.2: Struktura sloja domene

Paket *model* sadrži podatkovne klase koji su ujedno i definicije entiteta, a ključni su za rad aplikacije s samim podatcima.

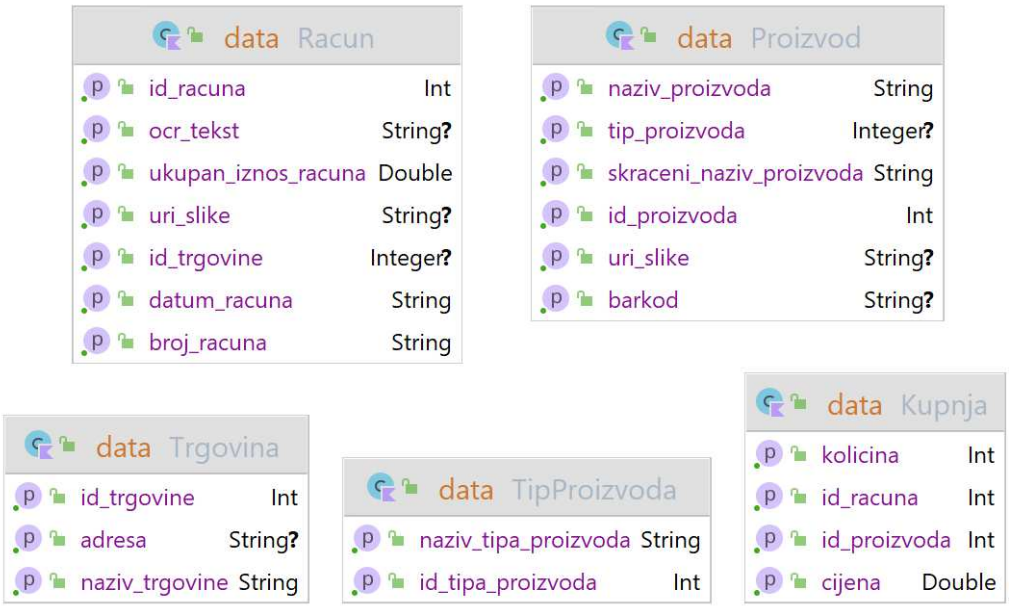
Paket *repository* sadrži programska sučelja koja definiraju funkcije za pristup podatcima.

Paket *use_case* sadrži svu poslovnu logiku aplikacije definiranu preko korisničkih sučelja.

Paket *util* sadrži razne klase i funkcije koje pomažu boljoj izvedbi ostalih funkcionalnosti.

5.2.1. Model

Model sadrži opise podatkovnih klasa koje modeliraju same entitete sustava. Te klase su *Racun*, *Proizvod*, *Trgovina*, *TipProizvoda* i *Kupnja*. One su centar arhitekture i ne moraju znati ništa o vanjskim slojevima aplikacije. Dijagram razreda prikazan je na slici 5.3.



Slika 5.3: Dijagram razreda modela

Klasa *Racun* modelira račun, a sadrži svojstva za id računa, očitani tekst računa, ukupan iznos, id trgovine, putanju datoteke slike računa, datum računa te broj računa.

Klasa *Proizvod* modelira proizvod, a sadrži svojstva za id proizvoda, naziv proizvoda, skraćeni naziv proizvoda, tip proizvoda, putanju datoteke slike proizvoda te barkod.

Klasa *Trgovina* modelira trgovinu, a sadrži svojstva za id trgovine, naziv i adresu trgovine.

Klasa *TipProizvoda* modelira tip proizvoda, a sadrži svojstva za id tipa proizvoda i za naziv tipa proizvoda.

Klasa *Kupnja* modelira kupnju nekog proizvoda, a sadrži svojstva za id računa, id proizvoda koji je kupljen, cijenu jednog proizvoda te količinu.

5.2.2. Repository

Repository sadrži programska sučelja koja definiraju funkcije za pristup podatcima.

Preko tih sučelja se odvija komunikacija poslovne logike sa samim podatkovnim slojem, neovisno o njihovoj implementaciji. Na taj način je moguće imati više implementacija sučelja bez da ovaj sloj mora znati išta o njima osim da vrše svoju funkcionalnost. To omogućava i jednostavnu zamjenu implementacije što je vrlo pogodno za testiranje ovog sloja ili za promjenu konkretnog načina izvedbe podatkovnog sloja.

Dijagram razreda je prikazan na slici 5.4.

Slika 5.4: Dijagram *repository* sučelja

Osnovne funkcije za dohvaćanje su *getAll* koja dohvaća listu podataka u obliku toka podataka (engl. *flow*) koji emitira nove podatke kad se oni promijene. Takav tok podataka je moguće motriti i automatski ažurirati podatke. Osim *getAll*, svako sučelje definira i funkcije za ažuriranje (engl. *update*), umetanje (engl. *insert*), brisanje (engl. *delete*) i dohvaćanje po id-u.

Osim gore navedenih funkcija sučelja definiraju i sljedeće funkcije:

– *RacunRepository* definira funkcije za dohvaćanje računa po id-u trgovine, dohvaćanje datuma i dohvaćanje toka podataka liste računa spojeno s trgovinama,

– *TipProizvodaRepository* definira funkciju za dohvaćanje liste tipova podataka,

– *TrgovinaRepostory* definira funkciju za dohvaćanje liste trgovina,

– *ProizvodRepository* definira funkcije za dohvaćanje liste proizvoda, ažuriranje slike proizvoda i brisanje slike proizvoda,

– *KupnjaRepository* definira funkcije za dohvaćanje liste kupnji po id-u računa, dohvaćanje liste kupnji po id-u proizvoda, dohvaćanje kupnji po tipu proizvoda, dohvaćanje toka podataka liste kupnje proizvoda jednog računa i dohvaćanje kupnji trgovine.

5.2.3. Slučajevi korištenja

Paket *use_case* sadrži klase koje implementiraju poslovnu logiku sustava. Svaka klasa predstavlja jedan slučaj korištenja i ima jednu funkciju *invoke*. *Invoke* je predefinirana *operator* funkcija svake klase u Kotlin jeziku koju je moguće 'nadjačati', tj. promijeniti definiciju. *Invoke operator* funkcija omogućava pozivanje funkcije samo preko imena klase, bez specificiranja imena same funkcije.

Klasa slučajeva korištenja ima mnogo, a većinom vrše jednostavnije poslove dohvaćanja, spremanja, ažuriranja i brisanja podataka. Osim tih, neke klase koje je bitno spomenuti su:

– *DownloadImage* koja vrši preuzimanje slike proizvoda,

– *ExtractProductInfoFromOCR* koja vrši izvlačenje informacija o proizvodima iz očitanog teksta računa,

– *ExtractRacunInfoFromOCR* koja vrši izvlačenje informacija o računu iz očitanog teksta računa,

– *GetFuzzyMatchProizvodKupnja* koja dohvaća informacije o proizvodu na temelju sličnosti skraćenog naziva proizvoda,

– *GetProizvodInfoFromBarcode* koja dohvaća informacije o proizvodu s udaljenog servisa pomoću barkoda,

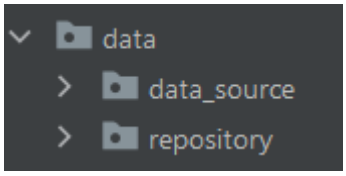
– *ParseBarcode* koja očitava barkod proizvoda sa slike,

– *ParseScanRacuna* koja očitava tekst računa sa slike.

Slučajevi korištenja također koriste pomoćne klase iz paketa *util*.

5.3. Podatkovni sloj

Podatkovni sloj sadrži dva paketa; *data_source* i *repository*.



Slika 5.5: Struktura podatkovnog sloja

Data_source sadrži definiciju baze podataka (BP) definiranu pomoću *Room* anotacije. Osim definicije BP, sadrži i sučelja anotirana s *@Dao* (engl. *data access object*). Ta sučelja povezuju SQL upite s definicijama funkcija.

Room

Room je biblioteka koja olakšava programsko definiranje baze podataka i upita. Iz definiranih entiteta i sučelja za pristup podatcima koji su anotirani s predefiniranim anotacijama *Room* generira SQLite bazu podataka i pripadajuće klase za izvršavanje upita. SQL upit se definira u anotaciji funkcije, a *Room* se pobrine za sve ostalo što oslobađa programera od pisanja tzv. *boilerplate* koda. Primjer jednostavnog SQL upita je na slici 5.6. U upitu se na jednostavan način mogu koristiti parametri funkcije.

```
@Query("SELECT * FROM racun WHERE id_racuna = :id")
suspend fun getRacunById(id: Int): Racun?
```

Slika 5.6: Primjer SQL upita u *Room* anotaciji

Paket *repository* ovog sloja sadrži implementacije sučelja koja definiraju funkcije za pristup podatcima sa slike 5.4. Svaka implementacija sadrži odgovarajuću instancu *DAO-a* te u implementaciji funkcija poziva odgovarajuće funkcije za dohvaćanje podataka.

5.3.1. Baza podataka

Slika 5.7: Dijagram baze podataka

Dijagram baze podataka prikazan je na slici 5.7. Baza podataka se sastoji od pet tablica, a njihove veze su prikazane na dijagramu.

Račun

Entitet *racun* predstavlja jedan račun, a sastoji se od sljedećih atributa:

- *id_racuna* – identifikacijski broj računa, ujedno je i primarni ključ,
- *broj_racuna* – broj računa, obavezan,
- *id_trgovine* – identifikacijski broj trgovine iz koje je račun,
- *ukupan_iznos_racuna* – ukupan iznos u kunama s računa, obavezan,
- *datum_racuna* – datum na koji je kupnja obavljena, obavezan,

- *ocr_tekst* – očitani tekst računa, obavezan,
- *uri_slike* – putanja do datoteke slike računa.

Trgovina

Entitet *trgovina* predstavlja trgovinu, a sastoji se od sljedećih atributa:

- *id_trgovine* – identifikacijski broj trgovine, ujedno i primarni ključ,
- *naziv_trgivine* – naziv trgovine koji joj korisnik zadaje, obavezan,
- *adresa* – adresa trgovine.

Kupnja

Entitet *kupnja* predstavlja kupnju nekog proizvoda s računa, a sastoji se od sljedećih atributa:

- *id_proizvoda* – identifikacijski broj proizvoda koji je kupljen,
- *id_racuna* – identifikacijski broj računa s kojeg je kupnja proizvoda,
- *kolicina* – količina kupljenoga proizvoda, obavezan,
- *cijena* – cijena jednog proizvoda koja je prikazana na računu, obavezan.

Proizvod

Entitet *proizvod* predstavlja kupljeni proizvod, a sastoji se od sljedećih atributa:

- *id_proizvoda* – identifikacijski broj proizvoda, ujedno i primarni ključ,
- *naziv_proizvoda* – korisnički proizvoljan naziv proizvoda, obavezan,
- *skraceni_naziv_proizvoda* – naziv proizvoda prikazan na samom računu, očitan s računa, obavezan,
- *barkod* – brojčani zapis barkoda s proizvoda,
- *uri_slike* – putanja do datoteke slike proizvoda,
- *tip_proizvoda* – identifikacijski broj tipa proizvoda kojeg je korisnik dodijelio proizvodu.

Tip proizvoda

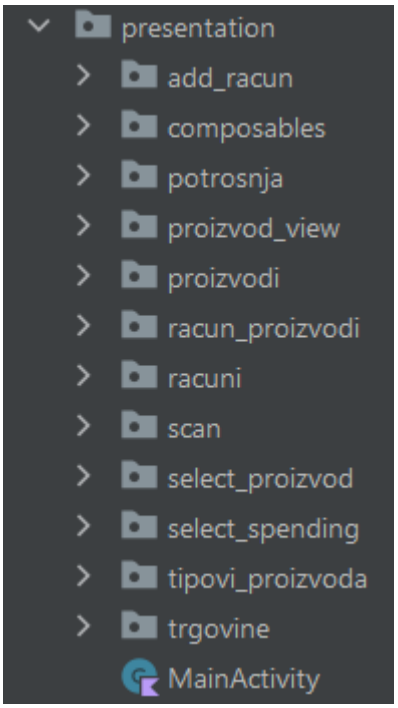
Entitet *tip_proizvoda* predstavlja korisnički zadan tip proizvoda koji služi za pregled potrošnje po proizvoljnim kategorijama, a sastoji se od sljedećih atributa:

– *id_tipa_proizvoda* – identifikacijski broj tipa proizvoda, ujedno i primarni ključ,

– *naziv_tipa_proizvoda* – naziv tipa proizvoda koji mu korisnik zadaje, obavezan.

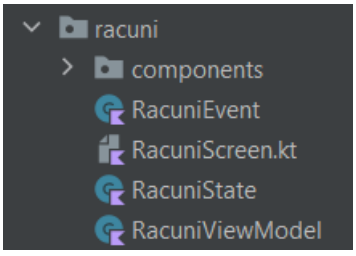
5.4. Prezentacijski sloj

Prezentacijski sloj aplikacije sadrži kod koji definira korisničko sučelje i koji se bavi dobavljanjem informacija s unutarnjih slojeva sustava. Struktura prezentacijskog sloja prikazana je na slici 5.8. Svaki paket sloja definira jedan zaslon aplikacije.



Slika 5.8: Struktura prezentacijskog sloja

Jedan paket sastoji se od četiri glavnih vrsta datoteka (slika 5.9). To su *Event*, *Screen*, *State* i *ViewModel*. Osim njih sadrži i paket *components* koji sadrži manje komponente koje se mogu ponavljati.



Slika 5.9: Datoteke jednog zaslona

Datoteka *State* sadrži definicju podatkovne klase koja definira atribute stanja podataka koji se prikazuju na korisničkom sučelju. To su podatci kao vrijednosti u tekstualnim poljima za unos, varijable za provjeru grešaka, liste podataka za prikaz i slično.

Datoteka *Event* definira sve takozvane događaje koje zaslon treba izkomunicirati prema *ViewModelu*. Drugim riječima, definira sve akcije koje korisnik može izvršiti na nekom zaslonu koje sa sobom povlače promjenu stanja podataka.

Datoteka *ViewModel* predstavlja klasu koja sadrži logiku koja upravlja događajima koji dolaze sa zaslona te inicijalizira stanja podataka. Osim toga, *ViewModel* emitira i određene događaje prema zaslonu, no on nema nikakve reference na zaslon, već zaslon sluša događaje određenog tipa koji dolaze.

Datoteka *Screen* modelira korisničko sučelje te prikazuje stanje podataka na zaslonu i poziva odgovarajuće događaje iz *ViewModela*. Samo korisničko sučelje definirano je programski što omogućava okolina *Jetpack Compose*.

Jetpack Compose

Jetpack Compose omogućava programsko oblikovanje i opisivanje korisničkog sučelja pomoću jezika *Kotlin*. To omogućuje jednostavnije oblikovanje korištenjem gotovih komponenti (npr. gumb, polja za unos itd.) i ubacivanjem podataka za prikaz direktno kod njihove deklaracije u programskoj datoteci.

5.5. Izvedba OCR-a

U ovom radu optičko prepoznavanje znakova (engl. *optical character recognition - OCR*) koristi se za dobivanje teksta računa koji se zatim dalje obrađuje. Za sam OCR se koristi *ML Kit*.

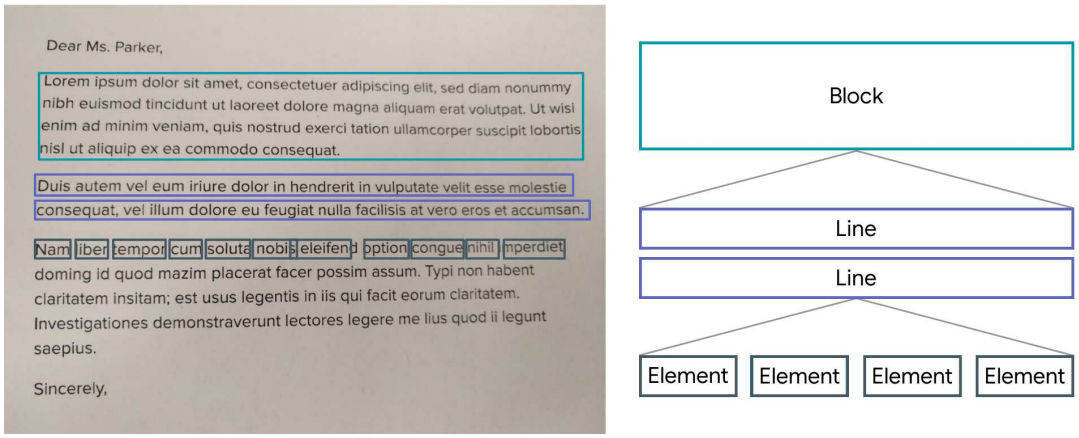
5.5.1. CameraX

Kako bi mogli obraditi sliku računa, prvo ju je potrebno dobiti. Za to se koristi biblioteka *CameraX* koja omogućava jednostavnu integraciju analiziranja i snimanje slike pomoću ugrađene kamere na mobilnom uređaju. Biblioteka nudi nekoliko slučajeva korištenja, a za potrebe ovog rada korišteni su *PreviewView* i *ImageCapture*. *PreviewView* omogućava korištenje elementa korisničkog sučelja koji prikazuje pregled slike s kamere na zaslonu uređaja. *ImageCapture* omogućava pohranjivanje slike u radnu memoriju uređaja koja se zatim može iskoristiti za daljnju obradu te pohraniti.

5.5.2. ML Kit

ML Kit je alat za razvoj koji omogućava korištenje gotovih API-ja baziranih na strojnom učenju. Alat pruža razne mogućnosti iz polja računalnog vida i obrade prirodnog jezika. Za OCR se koristi API za prepoznavanje teksta, koji iz dane slike izvlači tekst. Omogućava prepoznavanje teksta latiničnog, kineskog, korejskog, japanskog i devanagari pisma, no ovdje se koristi samo prepoznavanje latiničnog pisma.

Kao izlaz iz funkcije prepoznavanja dobije se tekst u blokovima gdje svaki blok sadrži linije, a svaka linija elemente (obično riječi). Primjer takve strukture prepoznavanja prikazan je na slici 5.10.



Slika 5.10: Prikaz strukture prepoznanog teksta [8]

Isječak koda koji obavlja poziv za prepoznavanja teksta nad ulaznom slikom prikazan je na slici 5.11. Potrebno je dobiti instancu prepoznavatelja teksta pozivom `getClient` nad klasom za prepoznavanje. Prepoznavanje teksta se zatim obavlja pozivom `process` metode koja prima sliku tipa `InputImage`, a pri uspješnom izvršavanju prepoznavanja teksta poziva se *callback* funkcija `onRecognitionComplete` s parametrom očitanog teksta. Očitani tekst se zatim obrađuje na već spomenut način, informacije se izvlače pomoću regularnih izraza kao što je opisano u poglavlju 4.3. Na tom primjeru programskog koda vidi se da je samo optičko prepoznavanje vrlo jednostavno ukomponirati u projekt, a sam proces obrade slike traje svega nekoliko sekundi.

```
val inputImage = imageProxy.image?.let { it: Image
    InputImage.fromMediaImage(it, imageProxy.imageInfo.rotationDegrees)
}

val recognizer = TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS)

inputImage?.let { it: InputImage
    recognizer.process(it).addOnSuccessListener { visionText ->
        val ocrText = visionText.text
        onRecognitionComplete(ocrText)
    }
}
```

Slika 5.11: Isječak koda za prepoznavanje teksta

Točnost optičkog prepoznavanja teksta ovisi o kvaliteti i uvjetima slike, no generalno je dovoljno dobro za daljnju obradu. Uspješnost izvlačenja informacija iz očitanog teksta je ovisna o točnosti očitavanja, a najproblematičnije informacije za dobiti su one o proizvodima. No ukoliko je slika dobro osvijetljena i kvalitetna, većina proizvoda, ako ne i svi, bi trebala biti uspješno očitana.

Osim prepoznavanja teksta *ML Kit* omogućava i skeniranje barkoda. Skeniranje funkcionira na isti način kao i prepoznavanje teksta, na odgovarajući API se preda slika i on vraća brojčanu vrijednost barkoda. U ovome radu se za skeniranje barkoda koristi standard EAN 13 koji definira 13-znamenkaste barkodove koji su na velikoj većini proizvoda.

5.6. Pretraživanje proizvoda pomoću barkoda

Servis za pretraživanje proizvoda po barkodu koji je korišten u radu je *Barcode Lookup* [3]. Servis omogućava pretraživanje proizvoda koji su u njihovoj bazi podataka po raznim podatcima, no ovdje je bitno pretraživanje prema barkodu. *Barcode Lookup* tvrde da imaju preko 450 milijuna barkodova u svojoj bazi podataka i time je to jedan od najbogatijih takvih servisa za pretragu proizvoda. Ako se proizvod koji pretražujemo nalazi u sustavu, za njega je moguće dobiti puno raznih informacija, npr. naziv, proizvođač, model, kategorija, opis, sastojci, veličina, slike i trgovine koje drže proizvod. Često su te informacije nepotpune, no za ovaj rad su bitni naziv proizvoda i njegova slika koji su gotovo uvijek u sustavu. Korištenje servisa se plaća, no moguće je uzeti probno razdoblje za isprobati ga.

5.6.1. Retrofit

Poziv na servis za pretraživanje proizvoda po barkodu izvršava se preko njegovog programskog sučelja na adresu `https://api.barcodelookup.com/v3/products` s postavljenim parametrom *barcode*. Taj poziv se iz programa može izvršiti pomoću biblioteke *Retrofit* [9]. Biblioteka omogućava slanje HTTP zahtjeva što je ovdje iskorišteno za dobivanje informacija o proizvodu. Definiranje sučelja za poziv prikazano je na slici 5.12. Pri slanju zahtjeva potrebno je definirati dva parametra, *key* i *barcode*. *Key* je korisnikov ključ za verifikaciju, a *barcode* je barkod proizvoda. U anotaciji funkcije treba biti definirana putanja do resursa.

```
@GET("/v3/products?formatted=y")
fun listProizvodi(@Query("key") key: String, @Query("barcode") barcode: String): Call<ProductsJSON>
```

Slika 5.12: Definiranje API sučelja za slanje zahtjeva

Programski kod koji vrši slanje zahtjeva na servis pretraživanja proizvoda po barkodu prikazan je na slici 5.13. Prvo je potrebno dobiti instancu *Retrofita* te zatim stvoriti servis za poziv na temelju sučelja koji ima metodu sa slike 5.12. Zahtjev se asinkrono poziva metodom `enqueue` koja zatim poziva *callback* kad dobije odgovor na zahtjev. Kod uspješnog dobivanja odgovora primljeni podatci se dalje koriste pozivom `onResponseSuccess` *callback* funkcije.

```
val retro = Retrofit.Builder()
    .baseUrl(URL_BARCODE_API)
    .addConverterFactory(GsonConverterFactory.create())
    .build()

val service = retro.create(GetProizvodFromBarcodeService::class.java)

val proizvodRequest = service.listProizvodi(API_KEY, barcode)

proizvodRequest.enqueue(object : Callback<ProductsJSON> {
    override fun onResponse(
        call: Call<ProductsJSON>,
        response: Response<ProductsJSON>
    ) {
        val proizvodi = response.body()
        onResponseSuccess(proizvodi)
    }

    override fun onFailure(call: Call<ProductsJSON>, t: Throwable) {
        Log.e( tag: "GetProizvodInfoFromBarcode",  msg: "Failed request!: ${t.message}")
    }
})
```

Slika 5.13: Isječak koda za poziv na udaljeni servis pretraživanja po barkodovima

Primjer dijela odgovora na zahtjev prikazan je na slici 5.14. Kao odgovor se dobiju informacije o proizvodu u JSON formatu. Podatke koje želimo iz odgovora je lako dobiti pomoću definiranja pomoćnih klasa koje imaju nazive atributa jednake kao i nazivi polja u JSON odgovoru. Klasa koja drži nama bitne podatke u ovom slučaju je *ProductsJSON*.

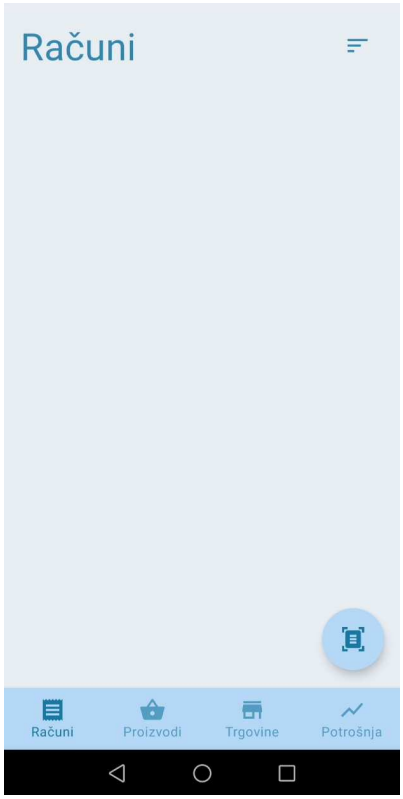
```
{
    "products": [
        {
            "barcode_number": "3850102314126",
            "barcode_formats": "EAN-13 3850102314126",
            "mpn": "0385010231412",
            "model": "",
            "asin": "",
            "title": "Kras Domacica Biscuits",
            "category": "Food, Beverages & Tobacco > Food Items > Grains, Rice & Cereal > Cereal & Granola",
            "manufacturer": "Kras",
            "brand": "Kras",
            "contributors": [],
            "age_group": "",
            "ingredients": "",
            "nutrition_facts": "",
            "energy_efficiency_class": "",
            "color": "",
            "gender": "",
            "material": "",
            "pattern": "",
            "format": "",
            "multipack": "",
            "size": "",
            "length": "",
            "width": "",
            "height": "",
            "weight": "",
            "release_date": "",
            "description": "Kras Domacica Biscuits.",
            "features": [],
            "images": [
                "https://images.barcodelookup.com/3181/31813313-1.jpg"
            ],
```


Slika 5.14: JSON odgovor na zahtjev za proizvodom

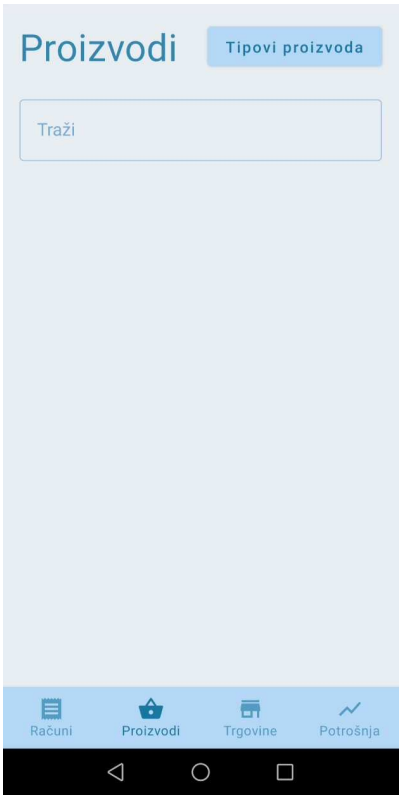
6. Funkcionalnosti aplikacije

6.1. Prvo pokretanje

Prilikom prvog pokretanja aplikacije korisnika dočeka prazan zaslon jer još nema nikakvih podataka. Korisnik svejedno može pregledavati različite zaslone, no neće biti nikakvih podataka. Zasloni koje može pregledavati su početni zaslon prikazan na slici 6.1, tj. zaslon s prikazom unesenih računa u aplikaciju, zaslon s popisom proizvoda koju su očitani, zaslon s trgovinama koje je korisnik upisao, te zaslon za prikaz potrošnje.



Slika 6.1: Početni zaslon



Slika 6.2: Zaslon s prikazom proizvoda

Navigacija među navedenim zaslonima obavlja se pomoću donje trake s gumbima za pripadajuće zaslone.

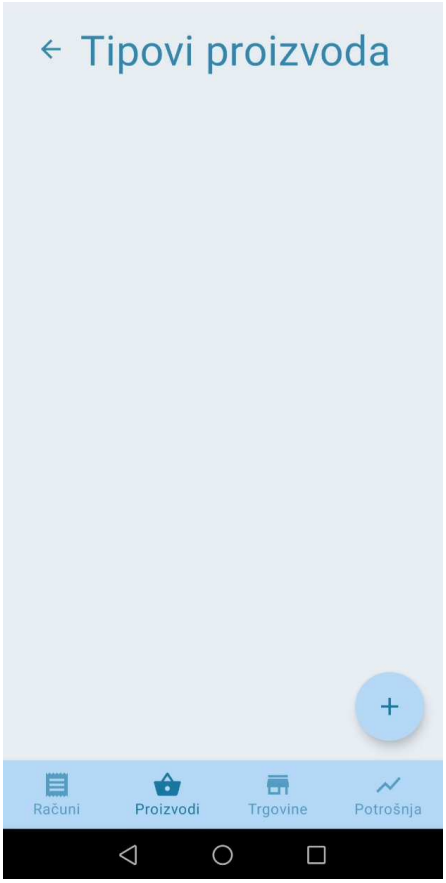
Na zaslonu s popisom proizvoda, koji je još prazan, postoji gumb za pregled tipova proizvoda 6.2. Dodirom na taj gumb prikazuje se zaslon s popisom tipova proizvoda, koji je također prazan.

6.2. Prvi koraci

Korisnik odmah može početi skenirati račune, no bilo bi dobro da prvo doda neke informacije u sustav. Jedna od stvari koju može dodati su tipovi proizvoda.

6.2.1. Dodavanje tipova proizvoda

Na zaslonu s listom tipova proizvoda postoji gumb za dodavanje novog u donjem desnom uglu kao na slici 6.3. Dodirom na taj gumb otvara se skočni prozor (engl. *dialog*) u koji je moguće upisati naziv novog tipa proizvoda (slika 6.4).



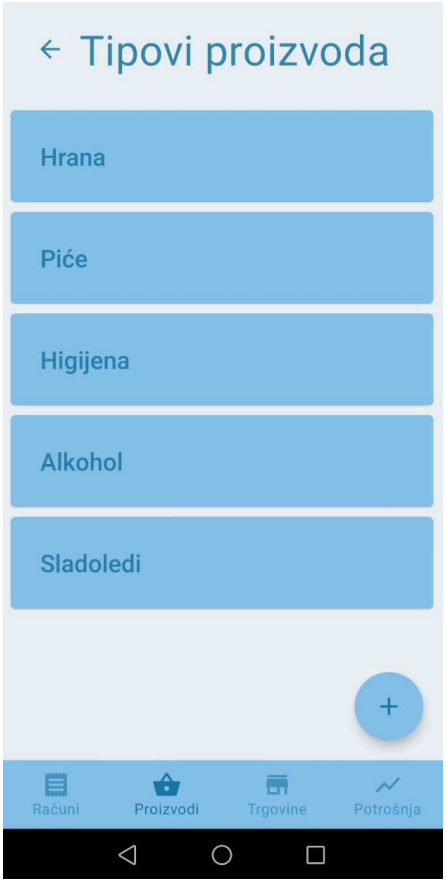
Slika 6.3: Zaslon s prikazom tipova proizvoda

Slika 6.4: Skočni prozor za dodavanje novog tipa proizvoda

Skočni prozor osim polja za upis naziva sadrži i dva gumba, jedan za spremanje novog tipa proizvoda u sustav, a drugi za odustajanje od kreiranja novog tipa proizvoda. Osim dodirom na gumb "Odustani", za odustajanje je moguće i dodirnuti zaslon izvan skočnog prozora. Kako bi došlo do uspješnog spremanja, naziv tipa proizvoda ne smije biti prazan.

Tipovi proizvoda koje korisnik dodaje u sustav su u potpunosti proizvoljni i služe za praćenje potrošnje. Nakon dodavanja nekoliko tipova zaslon s listom izgleda kao na slici 6.5.

Moguće je urediti postojeći tip proizvoda dodirom na karticu s njegovim nazivom, nakon čega se otvara skočni prozor za uređivanje (slika 6.6). Na tom skočnom prozoru se također nalazi i gumb za brisanje tipa proizvoda dodirom na kojeg se on u potpunosti briše iz sustava.



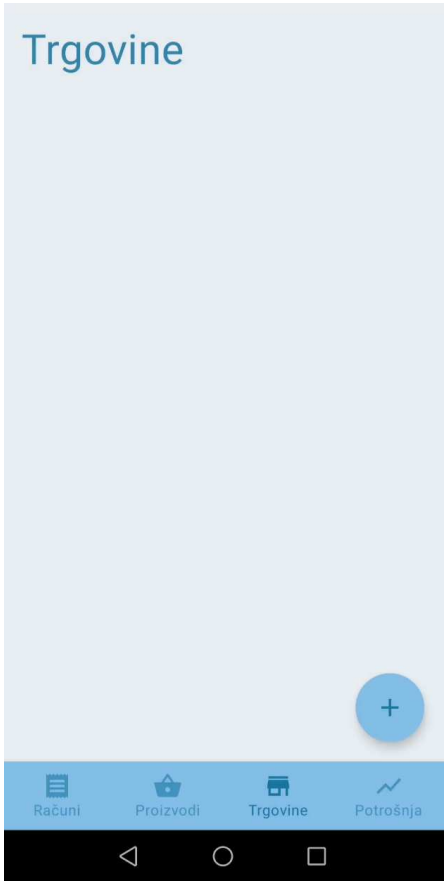
Slika 6.5: Zaslon s prikazom tipova proizvoda - s primjerima

Slika 6.6: Skočni prozor za uređivanje tipa proizvoda

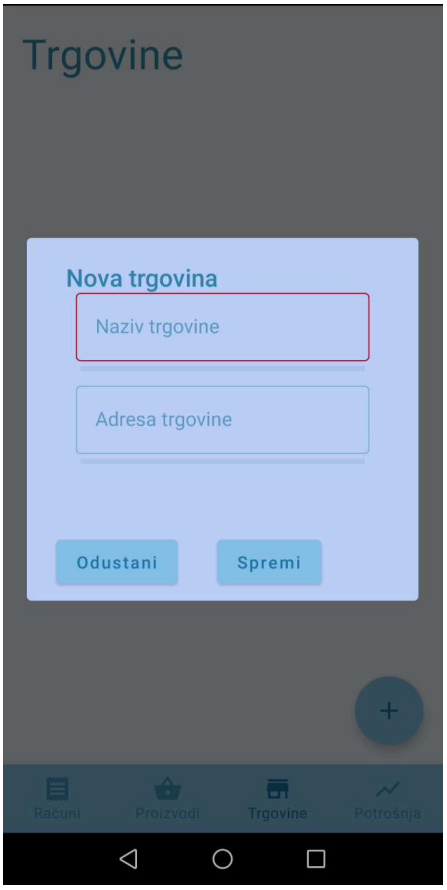
6.2.2. Dodavanje trgovina

Sad kad je dodano nekoliko tipova proizvoda po kojima korisnik želi pratiti potrošnju, bilo bi dobro dodati i nekoliko trgovina u kojima najčešće obavlja kupnje. Dodirom na gumb "Trgovine" na donjoj traci za navigaciju prikazuje se zaslon s listom trgovina koje je korisnik dodao u sustav. Zaslon je početno prazan, a nove trgovine moguće je dodati dodirom na gumb za dodavanje u donjem desnom uglu zaslona (slika 6.7).

Nakon dodira na gumb za dodavanje nove trgovine otvara se skočni prozor s dva polja za unos kao na slici 6.8.



Slika 6.7: Zaslon s prikazom trgovina



Slika 6.8: Skočni prozor za dodavanje nove trgovine

Prvo polje je za naziv trgovine, a drugo polje za adresu trgovine. Polje za naziv trgovine je obavezno i ne smije biti prazno, a ukoliko je prazno, označeno je crvenim obrubom. Ako korisnik pokuša spremiti trgovinu s praznim poljem za naziv, prikazat će mu se poruka upozorenja "Naziv trgovine ne smije biti prazan!" ispod samog polja za unos kao na slici 6.9. Kako korisnik počinje pisati naziv trgovine, ta poruka upozorenja se skriva (6.10). Nakon dodira na gumb "Spremi" trgovina se sprema u sustav.

Slika 6.9: Prikaz poruke upozorenja

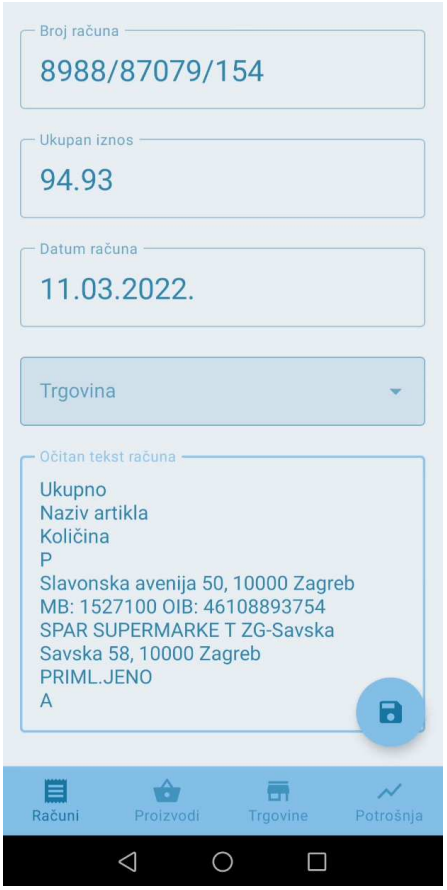
Slika 6.10: Pravilan upis podataka u polja

Uređivanje podataka o trgovinama obavlja se jednako kao i kod tipova proizvoda, dodirom na karticu s podacima trgovine na zaslonu trgovina. Potom se otvara skočni prozor za uređivanje podataka koji ujedno sadrži i gumb za brisanje.

6.3. Skeniranje računa

Nakon unosa gornjih informacija korisnik može krenuti sa skeniranjem prvog računa. Skeniranje se pokreće s zaslona za prikaz "Računi" dodirom da gumb u donjem desnom uglu. Nakon dodira na gumb otvara se zaslon s prikazom kamere. Na zaslonu se nalaze dva gumba, jedan za prekid radnje i jedan za okidanje slike (slika 6.11). Pritiskom na okrugli gumb za okidanje slike, uređaj snima sliku i šalje račun na obradu. Ukoliko nije slikan račun, podatci neće biti dobro očitani, no mogući je daljnji unos podataka ručno.

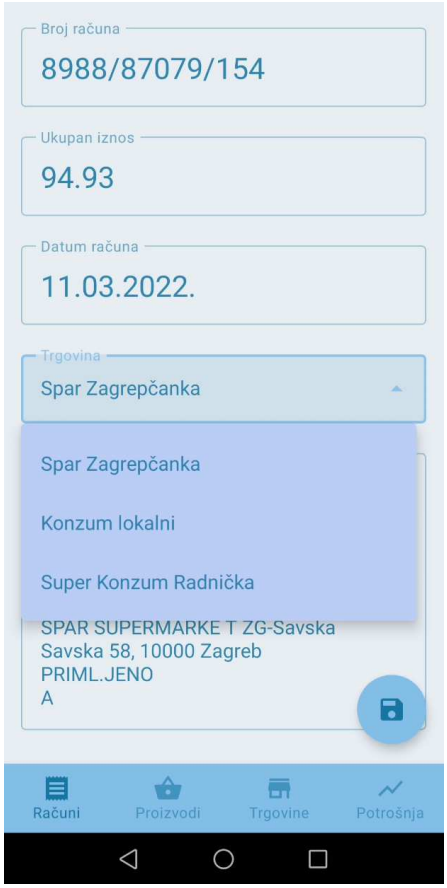
Slika 6.11: Prikaz zaslona s kamerom



Slika 6.12: Prikaz zaslona s učitanim podatcima o računu

Nakon obrade slike računa, prikazuje se zaslon z poljima za upis podataka o računu, od kojih su neka popunjena ako je očitavanje informacija iz slike bilo uspješno, dio zaslona je prikazan na slici 6.12. Polja koja se pokušaju očitati su broj računa, ukupan iznos, datum računa, tekst računa i podatci o proizvodima. Osim toga korisnik može odabrati trgovinu iz koje je račun dodirom na polje Trgovina, čime se otvara padajući izbornik sa svim trgovinama iz sustava kako je prikazano na slici 6.13.

Ispod gore navedenih polja nalaze se proizvodi koji su očitani s računa. Na slici 6.14 vidi se dio proizvoda. Podatci o proizvodima koji se očitavaju su skraćeni naziv proizvoda koji je prikazan na računu, količina kupljenog proizvoda i cijena proizvoda.

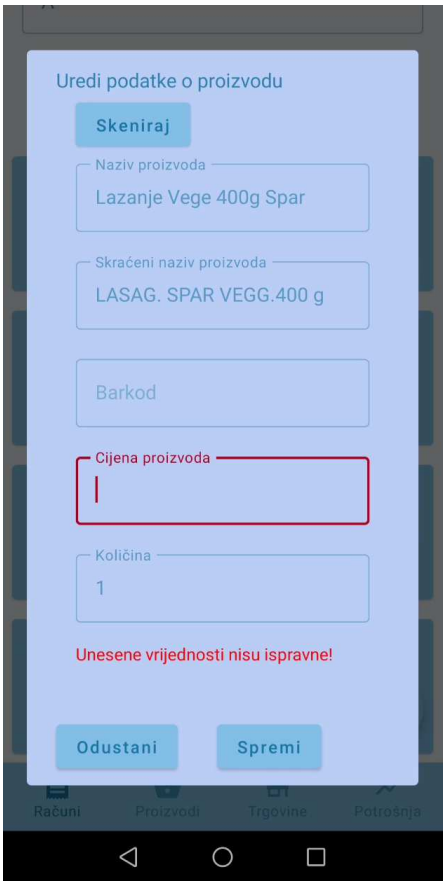


Slika 6.13: Prikaz padajućeg izbornika trgovina

Slika 6.14: Prikaz očitanih proizvoda s računa

Rijetko se događa da se svi podatci o proizvodima uspješno i pravilno očitaju sa slike, moguće zbog lošeg osvjetljenja, loše kvalitete računa ili još nekih utjecaja, pa su i neki podatci o proizvodima krivi. Na primjeru slike 6.14 krivo je očitana cijena proizvoda "LASAG. SPAR VEGG.400g" koja je na računu zapravo 18.99. Korisnik zbog toga može urediti podatke o proizvodima dodirom na karticu proizvoda nakon čega se otvara skočni prozor za uređivanje proizvoda prikazan na slici 6.15. Na taj način korisnik može promijeniti podatke o proizvodu koji su krivo očitani. Također može i promijeniti naziv proizvoda koji nije očitan, a služi za korisnikov opis proizvoda koji se razlikuje od skraćenog naziva očitanog s računa.

Slika 6.15: Prikaz skočnog prozora za uređivanje podataka o proizvodu



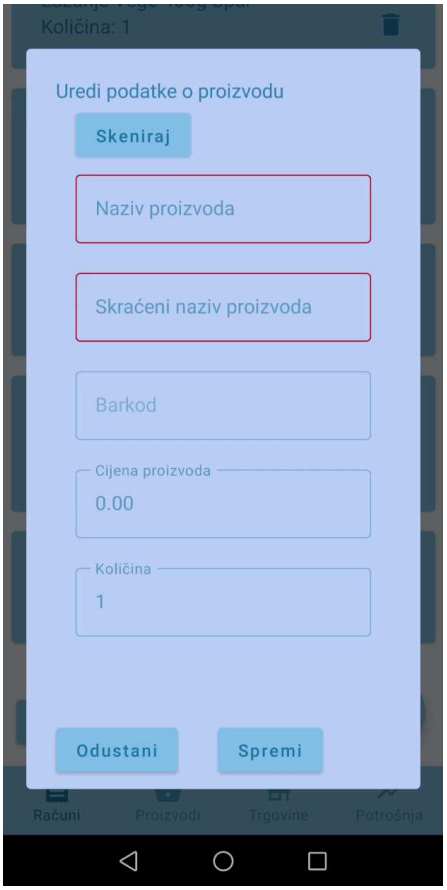
Slika 6.16: Prikaz poruke upozorenja o neispravnim vrijednostima

Kod uređivanja podataka o proizvodu obavezna polja su naziv proizvoda, skraćeni naziv proizvoda, cijena proizvoda i količina proizvoda. Ukoliko je neko od tih polja prazno, ili su u slučaju polja za cijenu proizvoda i količinu unesene vrijednosti neispravne, polja poprime crveni obrub, a ako korisnik pokuša spremiti podatke o proizvodu prikazati će mu se poruka upozorenja o neispravnosti unesenih vrijednosti (slika 6.16). Osim tih polja na prikazu se nalazi i polje za barkod proizvoda koje korisnik ne može urediti, već za unos vrijednosti u to polje postoji gumb "Skeniraj" koji se nalazi pri vrhu skočnog prozora. Dodirom na taj gumb otvara se zaslon s prikazom kamere identičan kao na slici 6.11. Korisnik tada treba okinuti sliku barkoda s proizvoda. Funkcionalnost očitavanja barkoda biti će pokrivena nešto kasnije. Dodirom na gumb "Spremi" podatci o proizvodu se ažuriraju u privremenoj memoriji i vide na prethodnom prikazu ispod skočnog prozora. Važno je napomenuti da nijedan novi proizvod još nije spremljen u sustav, što se događa tek kod spremanja računa.

Ponekad se može dogoditi da se neki tekst s računa krivo očita kao proizvod, a zapravo je besmislen podatak koji nije bitan korisniku. U tom slučaju moguće je mak-

nuti taj proizvod s liste dodirom na ikonu koja izgleda kao kanta za smeće, a nalazi se u donjem desnom uglu kartice proizvoda. Na slici 6.17 je tako primjer krivo očitanog podatka o proizvodu naziva "Povratna naknada". Korisnik isto tako može obrisati proizvod s liste za kojeg ne želi evidentirati kupnju iz bilo kojeg razloga.

Slika 6.17: Prikaz proizvoda i gumbova za dodavanje novih



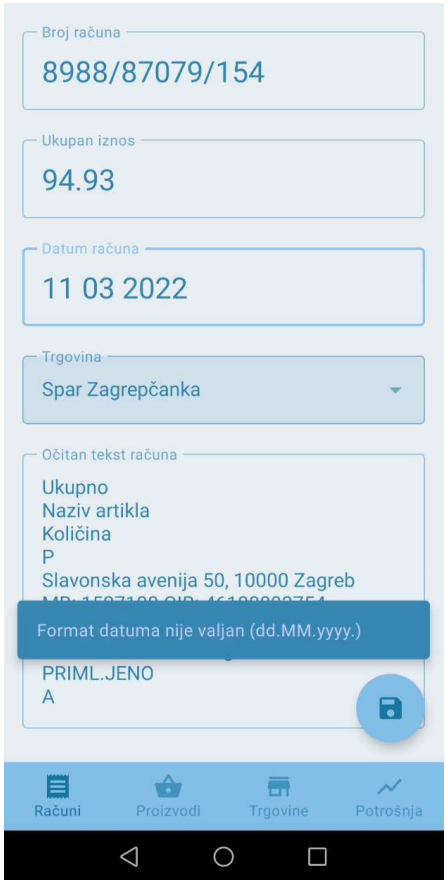
Slika 6.18: Prikaz skočnog prozora pri dodavanja novog proizvoda

Ukoliko pak neki proizvod nije očitan s računa, a korisnik ga želi evidentirati moguće ga je ručno dodati na listu dodirom na gumb "Dodaj novi" koji se nalazi pri dnu liste proizvoda (slika 6.17). Dodirom na taj gumb otvara se skočni prozor sličan onome za uređivanje proizvoda, samo s praznim poljima naziva, tj. s zadanim vrijednostima u slučaju cijene proizvoda i količine. Za dodavanje vrijede ista pravila validacije polja unosa kao i kod uređivanja proizvoda koje je prethodno opisano.

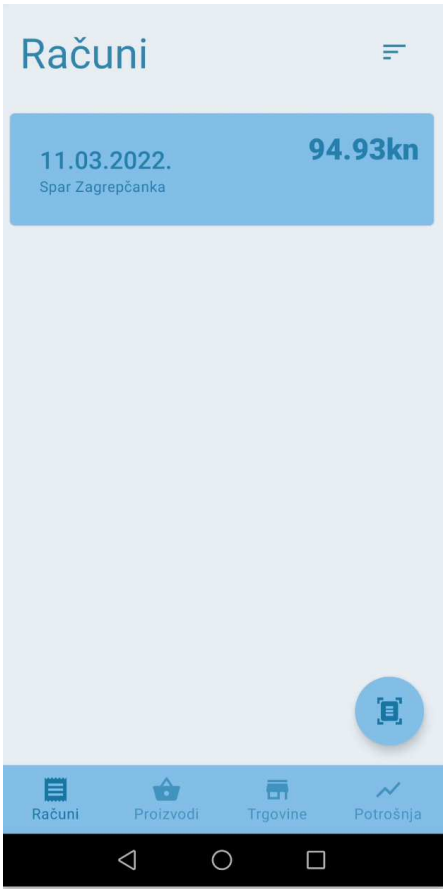
Ako u sustavu aplikacije već postoje proizvodi koji nisu očitani s računa, moguće ih je dodati dodirom na gumb "Dodaj postojeći", no trenutno još nema drugih proizvoda pa ćemo se na ovu funkcionalnost vratiti kasnije.

Nakon što je korisnik zadovoljan sa svim podatcima očitanim s računa, te podatke

može spremiti na lebdeći gumb za spremanje u donjem desnom uglu zaslona. Pri spremanju se provjerava valjanost podataka o računu i ako nisu ispravni javlja odgovarajuća poruka. Polja koja se provjeravaju su broj računa, koji ne smije biti prazan, ukupan iznos, koji treba biti valjan pozitivni broj te datum računa koji treba imati pravilan format oblika *dd.MM.yyyy.* Primjer prikazivanja poruke o nevaljanosti pri pokušaju spremanja je na slici 6.19.



Slika 6.19: Prikaz poruke o neispravnoj vrijednosti datuma računa



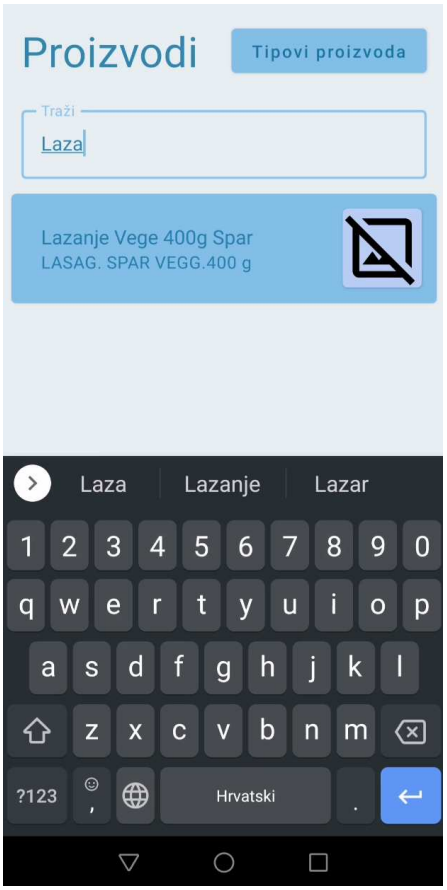
Slika 6.20: Prikaz nakon uspješnog spremanja računa

Nakon uspješnog spremanja računa i pripadajućih podataka, prikaz aplikacije se vraća na zaslon s računima te se prikazuje novo dodani račun na popisu (slika 6.20). Podatci na tom zasloni koji se prikazuju o pojedinom računu su datum računa, trgovina iz koje je račun te ukupan iznos računa.

6.4. Prikaz unesenih proizvoda

Nakon unosa podataka o prvom računu spremljeni proizvodi prikazuju se na zaslonu "Proizvodi". Proizvodi koje smo prethodno učitali s računa i spremili u sustav prikazani su na slici 6.21. Za svaki proizvod prikazuje se njegov naziv, skraćeni naziv koji piše na računu i slika proizvoda ako je unesena u sustav, inače se prikazuje ikonica da nema slike. Za većinu do sad unesenih proizvoda naziv i skraćeni naziv su jednaki jer ih nismo još promijenili.

Slika 6.21: Prikaz proizvoda



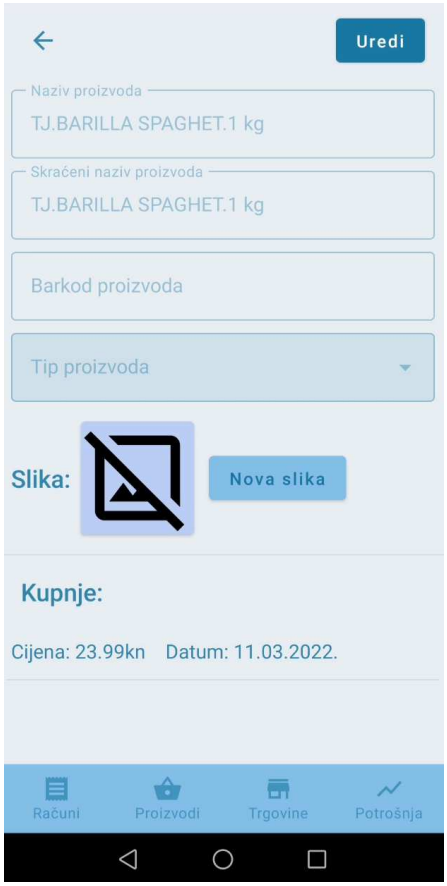
Slika 6.22: Prikaz primjera pretraživanja proizvoda

Pri vrhu zaslona s proizvodima također se nalazi polje za pretragu. Proizvodi se pretražuju unosom upita u to polje tijekom čega se lista proizvoda ažurira tako da pronalazi podudaranje upita s nazivom proizvoda. Primjer pretraživanja s upitom "Laza" prikazan je na slici 6.22, a rezultat upita je proizvod "Lazanje Vege 400g Spar".

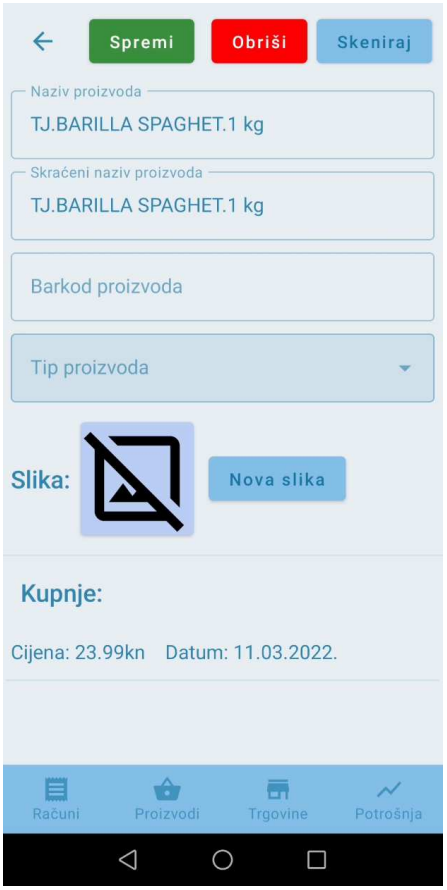
Dodirom na karticu proizvoda prelazi se na novi zaslon za prikaz informacija o tom proizvodu prikazan na slici 6.23.

6.5. Prikaz informacija o proizvodu

Zaslon za prikaz informacija o proizvodu sastoji se od polja za naziv proizvoda, skraćeni naziv proizvoda, barkod proizvoda i tip proizvoda. Ispod tih polja nalazi se slika proizvoda, ili ikonica da slike nema. Ispod slike nalaze se obavljene kupnje proizvoda, a za svaku je kupnju prikazan datum kupnje i cijena proizvoda.



Slika 6.23: Prikaz zaslona s informacijama o proizvodu

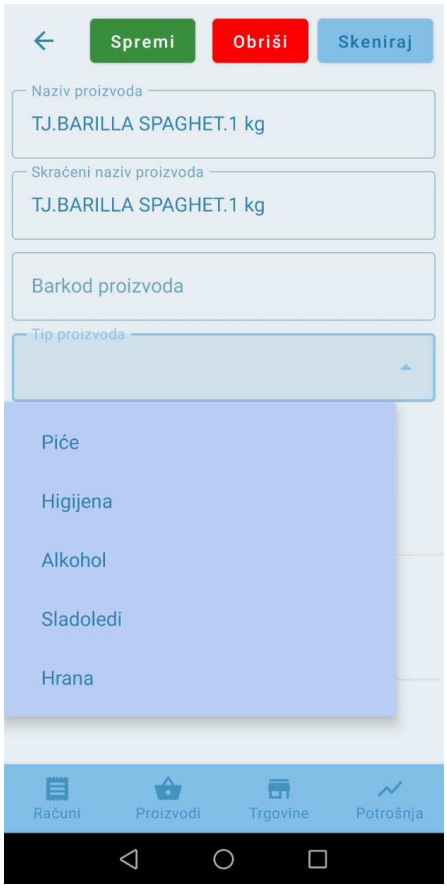


Slika 6.24: Uređivanje informacija o proizvodu

Informacije o proizvodu mogu se urediti dodirom na gumb "Uredi" pri vrhu zaslona, nakon čega se podatci u poljima mogu mijenjati (slika 6.24). Također se i pojavljuju 3 nova gumba, jedan za spremanje, jedan za brisanje i jedan za skeniranje barkoda s proizvoda. Kod uređivanja podataka o proizvodu moguće mu je dodijeliti tip proizvoda pomoću odgovarajućeg padajućeg izbornika kao što je prikazano na slici 6.25. Uređeni podatci se spremaju dodirom na zeleni gumb "Spremi", a proizvod se briše iz sustava dodirom na crveni gumb "Obriši".

Korisnik proizvodu može i dodijeliti sliku dodirom na gumb "Nova slika" pored ikonice slike. Dodirom na taj gumb otvara se zaslon s prikazom kamere identičan

onome na slici 6.11. Nakon okidanja slike proizvoda slika se sprema u sustav i prikazuje na odgovarajućim mjestima u aplikaciji kao na slici 6.26. Ako proizvod ima sliku, također se prikazuje i gumb za brisanje slike ispod gumba za novu sliku. Dodirom na taj gumb slika se briše iz sustava.



Slika 6.25: Padajući izbornik za biranje tipa proizvoda

Slika 6.26: Prikaz slike nakon spremanja

Kupnje proizvoda nije moguće uređivati na ovom zaslonu, već je to omogućeno na zaslonu s podatcima o odgovarajućem računu.

6.5.1. Skeniranje barkoda

Kod uređivanja podataka o proizvodu moguće je dodirnuti gumb "Skeniraj" nakon čega se otvara zaslon s prikazom kamere. Korisnik može skenirati barkod s proizvoda koji se očitava i šalje na daljnju obradu (slika 6.27). Skenirani barkod se upisuje u odgovarajuće polje, a obavlja se i upit na vanjski sustav pretrage proizvoda po barkodovima te se pokušaju preuzeti ime i slika proizvoda. Ako je pronalazak proizvoda bio uspješan, ti podatci će se automatski ažurirati, no ukoliko proizvod s tim barkodom nije

pronađen u vanjskom sustavu korisniku se prikazuje poruka "Nije pronađen proizvod s barkodom", primjer na slici 6.28.

Slika 6.27: Primjer skeniranja barkoda

Slika 6.28: Prikaz unosa skeniranog barkoda

Uspješni pronalazak proizvoda sa skeniranim barkodom prikazan je na slici 6.29. Aplikacija automatski popuni naziv proizvoda preuzet iz sustava barkodova, nekad taj naziv može biti i na stranom jeziku jer je sustav međunarodni, pa korisnik može proizvoljno dalje promijeniti sam naziv. Preuzima se i slika proizvoda iz sustava, a sliku je moguće povećano pregledati dodirom na umanjenu ikonu slike. Zaslon s prikazom uvećane slike proizvoda prikazan je na slici 6.30.

Slika 6.29: Primjer uspješnog pronalaska proizvoda sa skeniranim barkodom

Slika 6.30: Prikaz preuzete slike proizvoda

Skeniranje barkoda s proizvoda moguće je i kod uređivanja podataka o proizvodu prilikom unosa novog računa te funkcionira na isti način kao i ovdje.

6.6. Dodavanje novih računa

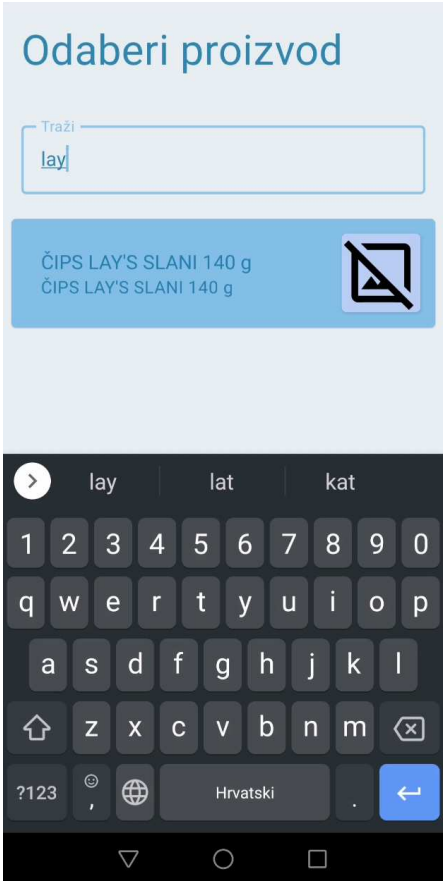
Nakon skeniranja nekoliko računa te kad skupimo malo veći broj proizvoda u sustavu, dodavanje novih računa s proizvodima koji su već u sustavu postaje brže i jednostavnije. Kad skenirani račun sadrži proizvode koji već jesu u sustavu, aplikacija pokuša dodati kupnju već postojećeg proizvod i tako bilježi kupnje za taj proizvod. Ovaj proces nije uvijek točan, no dovoljno je dobar da olakša unos postojećih proizvoda. Na slici 6.31 je primjer skeniranja novog računa koji sadrži kupnje proizvoda koji već jesu u sustavu. Na slici 6.32 prikazuju se dodani proizvodi iz sustava koji već imaju korisnički zadan naziv.

Slika 6.31: Primjer skeniranja računa

Slika 6.32: Prikaz postojećih proizvoda koji su dodani

Zbog toga što ovaj proces nije stopostotan, moguće je dodavanje već postojećih proizvoda dodirom na gumb "Dodaj postojeći", nakon čega se otvara zaslon za odabir proizvoda (slika 6.33). Proizvode je moguće pretražiti pomoću polja za pretraživanje, a proizvod se dodaje dodirom na njegovu karticu. Na slici 6.34 je prikazano dodavanje proizvoda "ČIPS LAY'S SLANI 140 g". Proizvod je zatim dodijeljen računu i sustav pri spremanju podataka računa sprema i kupnju tog proizvoda.

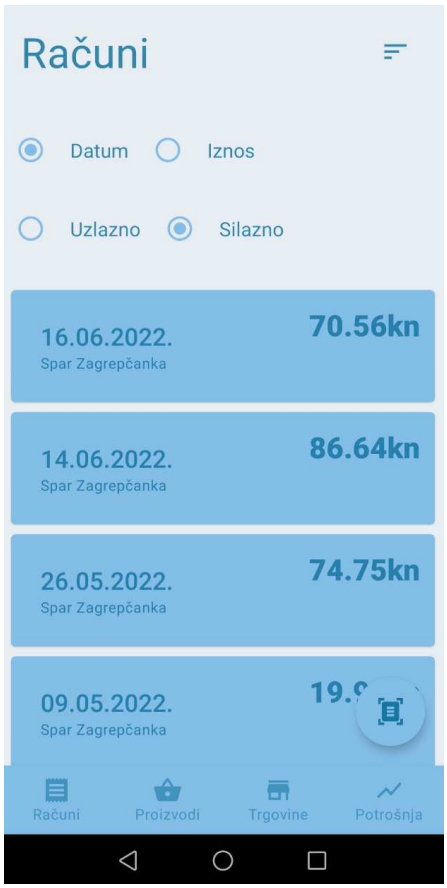
Slika 6.33: Zaslon za dodavanje postojećeg proizvoda



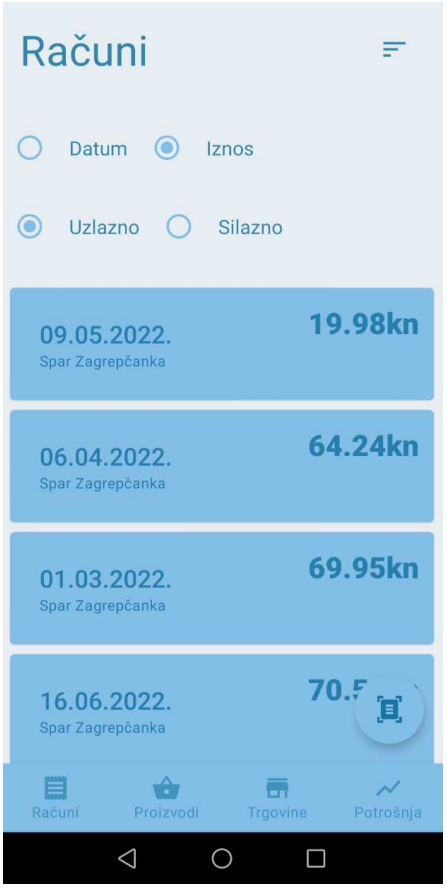
Slika 6.34: Pretraživanje proizvoda za dodavanje

6.7. Prikaz informacija o računu

Početni zaslon "Računi" je sad popunjen s desetak računa. U gornjem desnom uglu tog zaslona postoji gumb koji omogućava sortiranje prikaza liste računa. Dodirom na taj gumb otvara se izbornik za načine sortiranja prikazan na slici 6.35. Sortiranje prikaza liste računa omogućeno je silazno i uzlazno po datumu i po ukupnom iznosu računa. Zadan prikaz liste računa je silazno po datumu, a primjer drugačijeg sortiranja je na slici 6.36



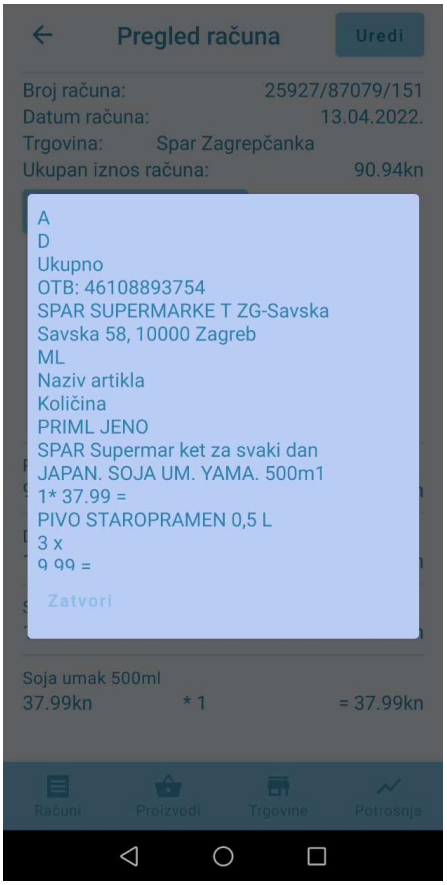
Slika 6.35: Zaslon s računima uz sortiranje



Slika 6.36: Sortiranje liste računa uzlazno po iznosu

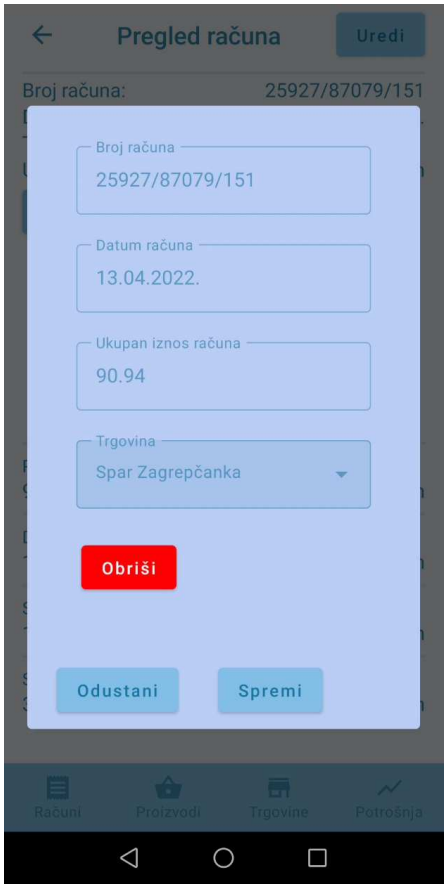
Dodirom na karticu računa otvara se zaslon s prikazom informacija o samom računu, prikazan na slici 6.37. Na tom zaslonu prikazane su informacije o broju računa, datumu računa, trgovini iz koje je račun i ukupnom iznosu računa. Dodirom na gumb "Očitani tekst računa" otvara se skočni prozor s tekstom računa prikazan na slici 6.38.

Slika 6.37: Zaslon za pregled informacija o računu



Slika 6.38: Pregled očitanog teksta računa

Uređivanja informacija o računu moguće je obaviti dodirom na gumb "Uredi" u gornjem desnom uglu zaslona nakon čega se otvara skočni prozor s poljima za uređivanje informacija koji je prikazan na slici 6.39. Mogu se promijeniti broj računa, datum računa, ukupan iznos računa i trgovina. Na ovom skočnom prozoru nalazi se i gumb za brisanje računa iz sustava. Promjene se spremaju dodirom na gumb "Spremi", a odbacuju dodirom na gumb "Odustani" ili dodirom izvan skočnog prozora.



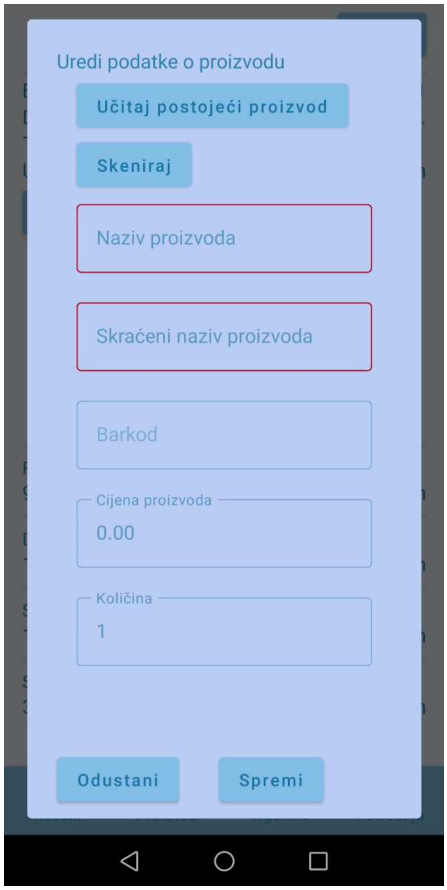
Slika 6.39: Skočni prozor za uređivanje računa

Slika 6.40: Pregled slike računa

Na pregledu računa također se nalazi i slika računa koju je moguće bolje pregledati dodirom na smanjenu sliku nakon čega se prelazi na zaslon s većom slikom (slika 6.40).

Na zaslonu računa također se nalazi lista kupljenih proizvoda, a za svaki proizvod se prikazuje njegov naziv, cijena, količina i ukupan iznos kupnje. Novu kupnju proizvoda moguće je dodati dodirom na gumb "+" nakon čega se otvara skočni prozor prozor za upisivanje informacija o proizvodu prikazan na slici 6.41. Na tom prozoru se nalazi gumb za učitavanje postojećeg proizvoda dodirom na kojeg se otvara zaslon za odabir proizvoda iz sustava identičan onome na slici 6.33.

Uređivanje kupnje proizvoda moguće je i dodirom na stavku kupnje na zaslonu računa nakon čega se otvara skočni prozor za uređivanje prikazan na slici 6.42. Brisanje kupnje proizvoda s računa moguće je dodirom na gumb "Obriši" na tom skočnom prozoru.

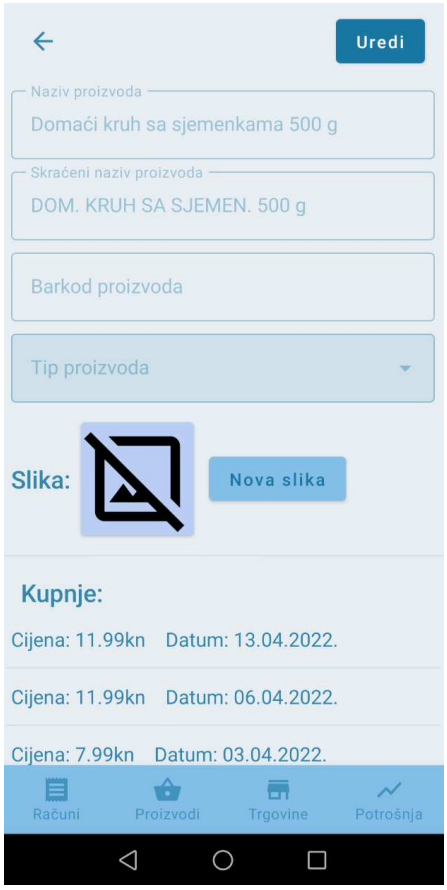


Slika 6.41: Skočni prozor za dodavanje nove kupnje proizvoda

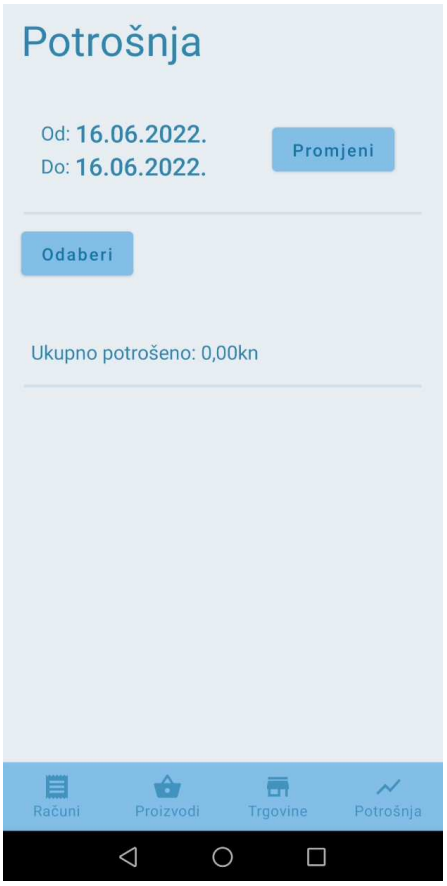
Slika 6.42: Skočni prozor za uređivanje informacija o kupnji proizvoda

6.8. Pregled potrošnje

Korisnik može pregledati svoje kupnje i potrošnju na nekoliko načina. Na zaslonu informacija o proizvodu nalaze se sve kupnje tog proizvoda, a prikazan je datum kupnje i cijena za koju je proizvod kupljen (slika 6.43).



Slika 6.43: Pregled kupnji na zaslonu proizvoda



Slika 6.44: Zaslon za pregled potrošnje

Dodirom na zaslon "Potrošnja" na donjem izborniku za navigaciju prikazuje se zaslon gdje korisnik može pregledati svoju potrošnju po danom vremenskom periodu i po danoj stavci (slika 6.44). Na tom zaslonu prikazan je vremenski raspon u datumima od kojeg do kojeg se prikazuje potrošnja korisnika. Postoji i gumb "Odaberi" dodirom na kojeg se prelazi na zaslon za odabir stavke pregleda potrošnje. Moguće je odabrati između tri različitih vrsti stavki.

Dodirom na gumb "Promijeni" otvara se prikaz za odabir raspona datuma za praćenje potrošnje (slika 6.45). Korisnik može odabrati dva datuma na prikazu kalendara ili ručno upisati datume nakon dodira na gumb za uređivanje u obliku olovke.

Slika 6.45: Odabir vremenskog perioda za pregled potrošnje

Slika 6.46: Biranje stavke proizvoda za pregled potrošnje

Na zaslonu za odabir stavke za pregled potrošnje prva vrsta su proizvodi. Prikazuju se svi proizvodi u sustavu sortirani abecednim redom, a korisnik odabire stavku dodirom na nju (slika 6.46). Nakon odabira prikaz se vraća na zaslon potrošnje gdje se zatim prikazuje ime odabrane stavke te sve obavljene kupnje koje ulaze u odabran vremenski period prikazan na slici 6.47. Za svaku kupnju piše datum i cijena proizvoda u vrijeme kupnje.

Slika 6.47: Prikaz potrošnje proizvoda

Slika 6.48: Biranje trgovine za pregled potrošnje

Osim odabira stavke proizvoda za pregled potrošnje je moguće odabrati i trgovinu i tip proizvoda. Na zaslonu za odabir stavke postoje gumbi "Proizvodi", "Trgovine" i "Tipovi proizvoda" pomoću kojih korisnik može navigirati između lista tih stavki. Na slici 6.48 prikazan je zaslon za odabir trgovine, a na slici 6.50 odabir tipa proizvoda. Na slici 6.49 prikazana je potrošnja u odabranoj trgovini. Za trgovinu se ispisuju računi koji su iz njih i koji spadaju pod odabran vremenski period.

Slika 6.49: Pregled potrošnje u trgovini

Slika 6.50: Biranje tipa proizvoda za pregled potrošnje

Osim liste kupnji ili računa prikazuje se i brojčana vrijednost ukupne potrošnje stavke u odabranom vremenskom periodu.

7. Zaključak

Praćenje osobne potrošnje bio je i ostaje mukotrpan proces svih ljudi koji se moraju držati nekog budžeta. U doba pametnih uređaja uzdigla se prilika za digitalno praćenje potrošnje u obliku mobilnih aplikacija ili udaljenih sustava povezanih s bankarstvom.

Uz ovaj rad opisana je i izrađena jedna takva aplikacija koja pomaže korisniku pratiti osobnu potrošnju u trgovinama opće namjene. Aplikacija olakšava taj proces tako što omogućava skeniranje računa iz trgovine s ciljem izvlačenja informacija bitnih o potrošnji iz njega. To korisniku minimizira vrijeme potrebno za unos podataka. Aplikacija također pruža uvid u potrošnju po danom vremenskom razdoblju, po trgovinama, po proizvodima te po korisnički stvorenim kategorijama proizvoda. Takva statistika pomaže osvijestiti korisnika o svojoj potrošnji te ga može navesti na smanjenje potrošnje na nepotrebne proizvode.

Iako aplikacija funkcionira dovoljno dobro da ubrza unos potrošnje, moguća su daljnja unaprijeđenja koja bi poboljšala točnost očitavanja računa čime bi se dodatno ubrzao unos podataka.

LITERATURA

[1] Android developers, . URL https://developer.android.com.

[2] Android studio, . URL https://en.wikipedia.org/wiki/Android_Studio.

[3] Barcode lookup. URL https://www.barcodelookup.com/.

[4] Receipt scanner: Easy expense. URL https://easy-expense.com/.

[5] Google cloud vision api. URL https://cloud.google.com/vision/docs/drag-and-drop.

[6] Microblink. URL https://microblink.com.

[7] Nanonets. URL https://nanonets.com/.

[8] Struktura prepoznatog teksta - slika. URL https://developers.google.com/ml-kit/vision/text-recognition/v2.

[9] Retrofit. URL https://square.github.io/retrofit/.

[10] Tesseract. URL https://github.com/tesseract-ocr/tesseract.

[11] Veryfi receipts ocr & expenses. URL https://www.veryfi.com/.

[12] Anh Duc Le, Dung Van Pham, i Tuan Anh Nguyen. Deep learning approach for receipt recognition, 2019. URL https://arxiv.org/abs/1905.12817.

[13] Xiaojing Liu, Feiyu Gao, Qiong Zhang, i Huasha Zhao. Graph convolution for multimodal information extraction from visually rich documents, 2019. URL https://arxiv.org/abs/1903.11279.

[14] Robert C. Martin. The clean architecture. 2012. URL https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.

[15] Xiaohui Zhao, Endi Niu, Zhuo Wu, i Xiaoguang Wang. Cutie: Learning to understand documents with convolutional universal text information extractor, 2019. URL https://arxiv.org/abs/1903.12363.

POPIS SLIKA

Analiza osobne potrošnje skeniranje računa

Sažetak

Ovaj rad opisuje razvoj aplikacije za praćenje potrošnje koja koristi skeniranje računa za brži unos informacija. Prvo je provedena analiza postojećih rješenja iz ovog područja, zatim su specificirani zahtjevi aplikacije. Provedena je i analiza postojećih usluga i tehnologija za skeniranje računa. Nakon toga se detaljno opisuje arhitektura rješenja te se prolazi kroz sve funkcionalnosti aplikacije počevši od prvog korištenja. Aplikacija omogućava skeniranja računa iz trgovine te izvlačenje bitnih informacija iz njega i tako pojednostavljuje praćenje osobne potrošnje korisnika. Omogućena je normalizacija proizvoda po barkodovima te praćenje statistike potrošnje po vremenskim periodima, proizvodima, trgovinama i kategorijama proizvoda.

Ključne riječi: Kotlin, Android, OCR, mobilna aplikacija, potrošnja, statistika potrošnje, skeniranje računa

Personal spending analysis by invoice scanning

Abstract

This paper describes the development of a spending analysis application that uses receipt scanning for faster data entry. First, an analysis of existing solutions in this area was conducted, then application requirements were specified. An analysis of existing receipt scanning services and technologies was also conducted. After that, the architecture of the solution is described in detail and we go through all the functionalities of the application starting from the first use. The application enables the scanning of invoices from the store and the extraction of important information from it, thus simplifying the tracking of users' personal spending. It is possible to normalize products by barcodes and monitor spending statistics by time periods, products, stores and product categories.

Keywords: Kotlin, Android, OCR, mobile application, spending, spending statistics, receipt scanning