Mobilna aplikacija za upravljanje resursima trgovine

Jursik, Daniel

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:483088

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

MOBILNA APLIKACIJA ZA UPRAVLJANJE RESURSIMA TRGOVINE

Završni rad

Daniel Jursik

Osijek, 2020.

1. UVOD 1
1.1 Zadatak završnog rada 1
2. OPIS KORIŠTENIH TEHNOLOGIJA 2
2.1 Programski jezik Java 2
2.2 Opisni jezik XML 2
2.3 Google-ova platforma Firebase 3
2.4 PayPal sustav za internetsko plaćanje 4
3. DEFINIRANJE KORISNIČKIH ZAHTJEVA 5
4. SPECIFIKACIJE APLIKACIJE 13
4.1 Baza podataka 13
4.2 Exchange Rates API 14
4.3 Plaćanje putem PayPal-a 14
5. RAZVOJ APLIKACIJE 16
5.1 Sučelje aplikacije 16
5.2 Administratorski pristup 17
5.3 Korisnički pristup 21
6. ZAKLJUČAK 29
LITERATURA 30
SAŽETAK 31
ABSTRACT 31
ŽIVOTOPIS 32

1. UVOD

U današnje vrijeme, mobilni uređaji vrlo su moćni i pristupačni i velik broj ljudi posjeduje, i na dnevnoj bazi koristi, navedene uređaje. Aplikacija za upravljanje resursima trgovine ne samo da može znatno olakšati održavanje trgovine, već i znatno privući potencijalne kupce, koji kupovinu mogu obaviti uz svega nekoliko klikova i to iz udobnosti svog doma i kada god im to odgovara, neovisno o vremenu rada trgovine. Izrada ove aplikacije se i zasniva na ovakvoj ideji.

Ova mobilna aplikacija pisana je u Java programskom jeziku, a za njen daljnji razvoj korištene su razne tehnologije, odnosno knjižnice (eng. *Library*). Neke od njih su: *Retrofit2* (služi za spajanje aplikacije sa API), *PayPal* (omogućava plaćanje unutar aplikacije), *ButterKnife* (služi za povezivanje programskog koda sa dizajnom i elementima), *Glide* (služi za učitavanje slika), *Firebase* (*Google*-ova platforma koja pruža usluge poput ovjere korisnika, baza podataka te spremanja korisnički generiranih sadržaja)

1.1 Zadatak završnog rada

U teorijskom dijelu ovog završnog rada potrebno je proučiti i opisati tehnologije za izradu mobilnih aplikacija na Android platformi. U praktičnom dijelu potrebno je izraditi funkcionalnu mobilnu aplikaciju za upravljanje resursima klasične trgovine.

Mobilna aplikacija treba omogućiti dodavanje, uklanjanje, uređivanje i pretragu artikala u online bazi podataka, spremanje povijesti svih operacija na njima, dobivanje statistike po određenim vremenskim periodima i višekorisnički rad. Aplikacija također treba imati mogućnost dodavanja proizvoda u košaricu te plaćanja, odnosno kupovine.

2. OPIS KORIŠTENIH TEHNOLOGIJA

Za izvedbu ovog projekta korištene su sljedeće tehnologije: programski jezik *Java*, opisni jezik *XML*, Google-ova platforma *Firebase* te *PayPal* sustav za internetsko plaćanje. Cjelokupna logika aplikacije izvedena je u programskom jeziku *Java*, sučelje u opisnom jeziku *XML*, baze podataka aplikacije u Google-ovoj platformi *Firebase*, dok je sustav plaćanja izveden preko *PayPal*-a.

2.1 Programski jezik Java

Java spada u skupinu viših programskih jezika i jedan je od glavnih programskih jezika za razvoj Android mobilnih aplikacija. Dok drugi jezici, poput *C++*, imaju mogućnost pokretanja samo unutar okruženja u kojemu su izvedeni, Java radi po filozofiji *„write once, run anywhere“*, odnosno *„napiši jednom, pokreni bilo gdje“* te je isti *Java* kôd moguće pokrenuti na bilo kojem stroju koji ima instaliran *Java Virtual Machine*, odnosno JVM. JVM sadrži po nekoliko implementacija za sva računala. Jednom izveden, Java kôd se pretvara u *JVM bytecode* te se pokreće prema naredbama JVM-a. Konačno, JVM predaje obrađeni kôd procesoru u njegovom jeziku [1]. Drugim riječima, Java kôd se prvo mora prevesti iz višeg programskog jezika u niži, odnosno u strojni jezik, što je ujedno i glavni nedostatak ovog programskog jezika. U prednosti viših programskih jezika spada jednostavnije programiranje, kraće pisanje kôda te je samim time kôd jednostavniji za čitanje i održavanje.

Java također spada u statički izvedene jezike te koristi primitivne tipove podataka poput *integer, long, float, double, boolean* kao i korisnički kreirane tipove podataka koji se nazivaju objekti. Kada je jezik statički izveden, to znači da se njegov kôd u potpunosti prevodi u strojni jezik prije izvršavanja programa. Dinamički izvedeni jezici prevode jezik u strojni za vrijeme samog izvršavanja programa [2], [3], [4].

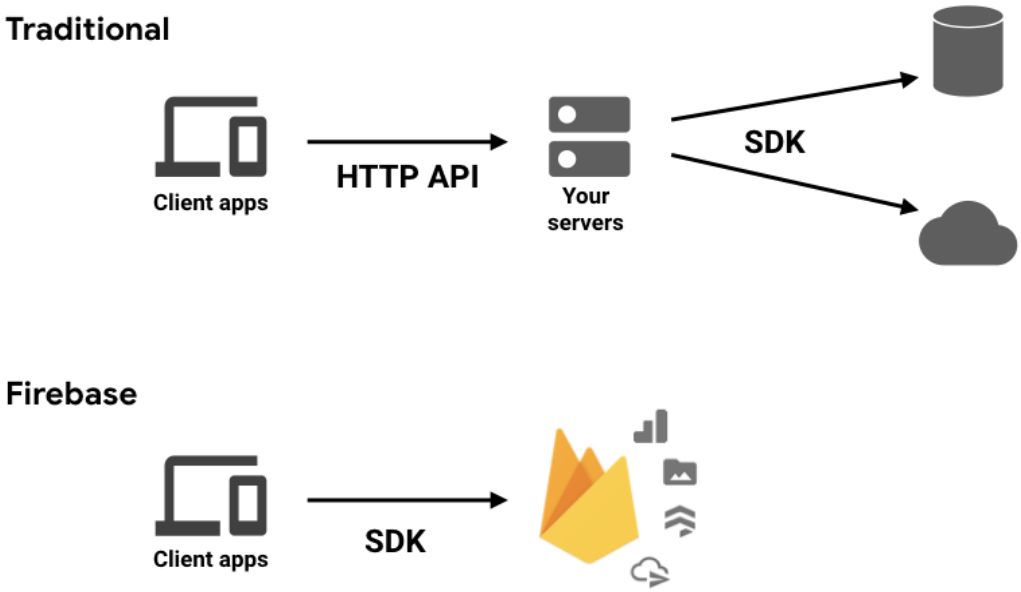
2.2 Opisni jezik XML

XML (engl. *Extensible Markup Language*) je jezik za označavanje podataka. Ideja je bila stvoriti jedan jezik koji će biti jednostavno čitljiv i ljudima i računalnim programima. Princip realizacije je vrlo jednostavan: odgovarajući sadržaj treba uokviriti odgovarajućim oznakama koje ga opisuju i imaju poznato, ili shvatljivo značenje. Format oznaka u XML-u vrlo je sličan formatu oznaka u npr. HTML-u. Pisanje XML kôda vrši se unutar oznaka (engl. *Tags*), odnosno njegove početne i krajnje oznake. XML je proširiv (engl. *Extensible*), što ga i razlikuje od

HTML-a. To znači da XML nema predefiniran jezik za označavanje kao što HTML ima, već dozvoljava korisnicima da naprave svoje oznake za opisivanje sadržaja [5].

2.3 Google-ova platforma Firebase

Firebase omogućava stvarno-vremenski pristup bazama podataka (*Firebase Database* ili *Firebase Firestore*), ovjeri korisnika (*Firebase Auth*), pohrani korisnički generiranih sadržaja (*Firebase Storage*) te brojne testne i analitičke usluge, uključujući i strojno učenje (engl. *Machine Learning*). Kako se svi podaci nalaze na platformi, za korištenje bilo koju od usluga, potreban je pristup internetu. *Firebase* platforma potpuno je besplatna za implementaciju i aplikacije koje nemaju puno prometa i korisnika. *Firebase* također pruža i potpunu *back-end* podršku, što znači da ne moramo voditi brigu o zaštiti bitnih podataka (poput lozinke) niti trošiti vrijeme na potpunu izvedbu sustava kako je bio slučaj sa klasičnim bazama podataka (Slika 2.1) [6].



Slika 2.1 Tradicionalni protiv Firebase pristupa [6]

Firebase Auth je sustav za registraciju i prijavu korisnika u aplikaciju. Osim prijave putem elektroničke pošte i lozinke, omogućava i implementaciju prijave putem *Google*, Facebook, Twitter, Yahoo te ostalih računa [7]. *Firebase Firestore* je tzv. *NoSQL* baza podataka. Radi na principu dokumenata i kolekcija koje sadrže navedene dokumente. Prednosti ovakve baze podataka su jednostavna implementacija, jednostavan i razumljiv prikaz podataka

te jednostavnija organizacija podataka u odnosu na klasične baze podataka. *Firebase Firestore* također omogućava i vlastitu izgradnju liste podataka koja se ažurira u stvarnom vremenu što ga čini idealnim rješenjem za prikaz liste podataka koje ovise o stvarnom vremenu [8]. *Firebase Storage* je sustav za pohranu multimedijskih podataka stvorenih od strane korisnika, npr. glazbe, slika ili video zapisa, koji se pohranjuju na *Google Cloud Storage*. Pri podizanju podatka na *Cloud* generira se link kojim je moguće pristupiti navedenom podatku. Podržava i mogućnost kontrole pristupa putem *Firebase Auth* [9].

2.4 PayPal sustav za internetsko plaćanje

PayPal je internetski orijentirana tvrtka koja omogućuje obavljanje uplata i novčanih prijenosa preko interneta. Zbog svoje „klijent-klijentu“ usluge (engl. *peer-to-peer*), način plaćanja omogućen je bilo kome tko ima e-poštu za slanje novca nekom drugom tko također ima e-poštu. Pokretač transakcije *PayPal-a* se prvo mora registrirati na *PayPal* stranici, prebaciti određenu svotu novca na svoj korisnički račun putem banke ili kreditne kartice [10].

3. DEFINIRANJE KORISNIČKIH ZAHTJEVA

Za razvoj aplikacije, potrebno je razumjeti i definirati zahtjeve administratora i korisnika. Aplikacija treba imati mogućnost registracije i prijave korisnika, pristup namijenjen isključivo administratoru trgovine te pristup namijenjen svim ostalim korisnicima (kupcima). U tablici 3.1. naveden je popis zahtjeva, dok ih tablice u nastavku detaljnije opisuju.

Tablica 3.1 Pregled korisničkih zahtjeva

ID	Naziv
1	Registracija korisnika
2	Prijava administratora ili korisnika
3	Dodavanje proizvoda
4	Ažuriranje proizvoda
5	Brisanje proizvoda
6	Prikaz povijesti
7	Prikaz statistike
8	Dodavanje proizvoda u košaricu
9	Kupovina proizvoda
10	Pregled kupljenih proizvoda

Tablica 3.2 detaljnije prikazuje zahtjev za registraciju korisnika. Registracija se vrši upisivanjem korisničkih podataka e-pošte i lozinke te pritiskom na gumb za registraciju. Jedini preduvjet registracije je da već ne postoji registrirana e-pošta kojom se korisnik želi registrirati. Aktivnost za registraciju korisniku nudi tri polja za unos (e-pošta, lozinka i ponovljena lozinka), a pritiskom na gumb za registraciju, podaci se preko *FirebaseAuth* provjeravaju, uspoređuju s postojećim korisnicima te, ukoliko korisnik s navedenom e-poštom ne postoji, podaci se spremaju u bazu. U protivnom će sustav u obliku *Toast* poruke ispisati grešku.

Tablica 3.2 Prikaz zahtjeva "Registracija korisnika"

ID korisničkog zahtjeva	**1**
Naziv korisničkog zahtjeva	Registracija korisnika
Opis korisničkog zahtjeva	Registracija korisnika elektronskom poštom i lozinkom
Preduvjet	Nepostojanje registrirane e-pošte
Glavni scenarij	Korisnik: 1. Odabire opciju registraciju 2. Unosi podatke – e-poštu i lozinku 3. Potvrđuje unos *4.* *Firebase Auth* sprema korisnika u bazu
Alternativni scenarij	1. Odabirom gumba za povratak • Sustav vraća korisnika na stranicu za prijavu 2. Nema pristup internetu • *Toast* poruka sa pogreškom 3. Korisnik već postoji • *Toast* poruka sa pogreškom

Tablica 3.3 detaljnije prikazuje korisnički zahtjev za prijavu u aplikaciju. Glavni preduvjet za podnošenje zahtjeva za prijavu je da korisnik, koji se pokušava prijaviti, već ima prethodno registriran korisnički račun, čije podatke na početnom zaslonu unosi u polja za prijavu. Pritiskom na gumb, korisnik ili administrator pokreće zahtjev za prijavu te u slučaju valjanih podataka, omogućuje mu se prijava. U protivnom dolazi do ispisivanja pogreške u obliku *Toast* poruke. Alternativni scenarij omogućava korisniku da umjesto prijave, pritiskom gumba za registraciju, odlazi na stranicu za registraciju. U slučaju zaboravljene lozinke, korisnik ili administrator ima mogućnost i resetiranja lozinke pritiskom na gumb za zaboravljenu lozinku, upisom svoje e-pošte u predviđeno polje za unos te pritiskom na gumb. Korisnik tada u svoj sandučić elektroničke pošte dobiva poveznicu, koja ga vodi na stranicu unosa nove lozinke. Unosom nove lozinke te pritiskom gumba, navedena se lozinka odmah postavlja kao važeća.

Tablica 3.3 Prikaz zahtjeva "Prijava korisnika"

ID korisničkog zahtjeva	**2**
Naziv korisničkog zahtjeva	Prijava administratora ili korisnika
Opis korisničkog zahtjeva	Prijava korisnika elektronskom poštom i lozinkom
Preduvjet	Registrirani korisnički račun
Glavni scenarij	1. Korisnik unosi podatke za prijavu 2. Pritiskom gumba, korisnik podnosi zahtjev za prijavu 3. Sustav provjerava podatke te omogućava prijavu ukoliko korisnik postoji
Alternativni scenarij	1. Administrator ili korisnik je zaboravio svoju lozinku • Otvara poveznicu *Password forgotten?* • Unosi e-poštu registriranog računa u polje • Na svojoj e-pošti otvara link za resetiranje lozinke • Upisuje novu lozinku • Prijavljuje se upotrebom nove lozinke

Tablica 3.4 prikazuje način dodavanje proizvoda u trgovinu te ažuriranje već postojećih proizvoda. Pritiskom na gumb za dodavanje, otvara se aktivnost za dodavanje i ažuriranje proizvoda. Administrator upisuje detalje proizvoda kao što su naziv, opis, cijena pojedinačnog proizvoda, ukupna količina dostupnih proizvoda te učitava sliku proizvoda, nakon čega odabirom gumba za spremanje, proizvod postavlja na trgovinu. Ažuriranje proizvoda pokreće se odabirom proizvoda iz pregleda trgovine, nakon čega se ponovno otvara navedena aktivnost. Ukoliko prilikom dodavanja ili ažuriranja proizvod pod unesenim imenom već postoji, ukoliko je sigurnosni mehanizam uključen (označen), ispisat će se odgovarajuća *Toast* poruka i operacija dodavanja proizvoda će biti obustavljena, a administrator tada može izmijeniti detalje proizvoda ili ukloniti oznaku sigurnosnog mehanizma. Uklanjanjem oznake, sigurnosni mehanizam se gasi, a dodavanjem proizvoda pod već postojećim imenom, on će se ažurirati.

Tablica 3.4 Prikaz zahtjeva "Dodavanje proizvoda"

ID korisničkog zahtjeva	**3 i 4**
Naziv korisničkog zahtjeva	Dodavanje proizvoda i ažuriranje proizvoda
Opis korisničkog zahtjeva	Administrator pritiskom na gumb za dodavanje proizvoda ili odabirom proizvoda u stranu dodaje ili ažurira proizvod
Preduvjet	Nema
Glavni scenarij	1. Pritiskom gumba, administrator otvara stranicu za dodavanje proizvoda 2. Administrator upisuje podatke proizvoda u polja 3. Administrator odabire i učitava sliku proizvoda 4. Pritiskom gumba, administrator dodaje proizvod na trgovinu 5. Sustav vraća administratora na pregled trgovine
Alternativni scenarij	1. Administrator poništava unos • Sustav vraća administratora na pregled trgovine

Tablicom 3.5 opisano je brisanje proizvoda iz trgovine. Glavni preduvjet za uklanjanjem proizvoda sa trgovine je postojanje proizvoda na trgovini. Administrator pokreće uklanjanje na način da karticu odabranog proizvoda pritisne i povuče u lijevu ili u desnu stranu. Otpuštanjem kartice proizvoda, on se uklanja s popisa. U slučaju da pristup internetu nije moguć, operacija uklanjanja neće biti izvršena, već će se kartica proizvoda vratiti u svoj prvotni položaj i ispisati odgovarajuća *Toast* poruka.

Tablicom 3.6 opisano je prikazivanje povijesti i statistike proizvoda. Izvršavanjem bilo koje od operacija dodavanja, ažuriranja ili brisanja stvaraju se unosi povijesti i statistike koji su vremenski organizirani. U povijesti se stvara kartica s nazivom proizvoda, e-poštom administratora koji je izvršio zahtjev, operacijom koja je na navedenom proizvodu izvršena, prethodnom cijenom proizvoda te datumom i vremenom izvršavanja operacije. U statistici se stvara kartica sa datumom te sumom svih operacija dodavanja, ažuriranja i uklanjanja izvršenih na taj datum. Podatke unutar statistike moguće je filtrirati i pretraživati.

Tablica 3.5 Prikaz zahtjeva "Brisanje proizvoda"

ID korisničkog zahtjeva	**5**
Naziv korisničkog zahtjeva	Brisanje proizvoda
Opis korisničkog zahtjeva	Administrator povlačenjem proizvoda u stranu briše navedeni proizvod
Preduvjet	Postojanje proizvoda
Glavni scenarij	1. Administrator povlači karticu proizvoda u stranu • Proizvod i slika proizvoda uklanjaju se iz baze
Alternativni scenarij	1. Administrator nema omogućen internet i povlači karticu proizvoda u stranu • Kartica se vraća na svoje početno mjesto

Tablica 3.6 Prikaz zahtjeva "Prikaz povijesti" i “Prikaz statistike”

ID korisničkog zahtjeva	**6 i 7**
Naziv korisničkog zahtjeva	Prikaz povijesti i statistike
Opis korisničkog zahtjeva	Prikazuje vremenski organiziranu listu koja se popunjava dodavanjem, ažuriranjem i brisanjem proizvoda
Preduvjet	Akcija dodavanja, ažuriranja ili brisanja proizvoda
Glavni scenarij	1. Administrator izvršava operaciju na proizvodu 2. U povijesti se stvara kartica s imenom proizvoda, e-poštom administratora, prethodnom cijenom, datumom te izvršenom operacijom 3. U statistici se stvara kartica s trenutnim datumom te ukupnim brojem svih operacija izvršenih tog datuma 4. Pokretanjem odgovarajućih stranica otvaraju se liste
Alternativni scenarij	nema

Tablica 3.7 prikazuje korisnički zahtjev za dodavanje željenih proizvoda u košaricu. Preduvjet je postojanje proizvoda kojeg će korisnik dodati. Dodavanje proizvoda korisnik vrši na način da prvo odabere proizvod te željenu količinu. Pritiskom na gumb za dodavanje u košaricu, proizvodi se dodaju. Pokretanjem aktivnosti, prikazuje se košarica sa svim proizvodima koje je korisnik dodao. Aktivnost, osim količine, prikazuje i cijenu pojedinačnog proizvoda, kao i ukupnu cijenu proizvoda s obzirom na količinu, dok se ispod popisa nalazi ukupna cijena svih proizvoda iz košarice. Odabirom proizvoda iz košarice, korisnik ponovno može otvoriti detalje proizvoda te promijeniti željenu količinu, dok se uklanjanje proizvoda vrši pritiskom na karticu proizvoda te njenim povlačenjem u lijevu ili desnu stranu.

Tablica 3.7 Prikaz zahtjeva "Dodavanje proizvoda u košaricu"

ID korisničkog zahtjeva	**8**
Naziv korisničkog zahtjeva	Dodavanje proizvoda u košaricu
Opis korisničkog zahtjeva	Dodavanje odabranih proizvoda u košaricu
Preduvjet	Postojanje proizvoda
Glavni scenarij	1. Korisnik odabire proizvod 2. Otvara se stranica sa detaljima proizvoda 3. Odabirom količine te pritiskom na gumb, korisnik dodaje proizvod u košaricu 4. Sustav vraća korisnika na pregled trgovine
Alternativni scenarij	nema

Tablica 3.8 pobliže opisuje kupovinu proizvoda. Nakon dodavanja jednog ili više proizvoda u košaricu, pritiskom na gumb za opcije plaćanja, sustav pokreće *PayPal* aktivnost za odabir metode plaćanja. Prva opcija nudi opciju plaćanja putem *PayPal* računa, dok drugom opcijom korisnik može platiti upisom podataka svoje kreditne kartice. Za plaćanje putem *PayPal* računa, potreban je *PayPal* račun te se od korisnika zahtijeva prijava valjanim *PayPal* podacima. Ukoliko korisnik ne posjeduje račun, može isti napraviti pritiskom na link za registraciju. U tom slučaju, sustav preusmjerava korisnika na web preglednik sa otvorenom stranicom za prijavu na *PayPal*. Odabirom opcije za plaćanje karticom, *PayPal* sustav nudi korisniku mogućnost slikanja kartice ili upisa podataka preko tipkovnice.

Tablica 3.8 Prikaz zahtjeva "Kupovina proizvoda"

ID korisničkog zahtjeva	**9**
Naziv korisničkog zahtjeva	Kupovina proizvoda
Opis korisničkog zahtjeva	Kupovina proizvoda prethodno dodanih u košaricu
Preduvjet	Postojanje proizvoda u košarici
Glavni scenarij	1. Pritiskom na gumb, korisnik podnosi zahtjev za kupovinom 2. Sustav preusmjerava korisnika na stranicu za plaćanje 3. Korisnik odabire metodu plaćanja putem *PayPal*-a 4. Korisnik unosi svoje *PayPal* podatke i prijavljuje se 5. Pritiskom na gumb za plaćanje, korisnik izvršava plaćanje 6. Sustav vraća korisnika na pregled trgovine
Alternativni scenarij	1. Korisnik odabire metodu plaćanja karticom 2. Sustav traži korisnika dozvolu za upotrebu kamere 3. Korisnik dozvoljava sustavu upotrebu kamere • Korisnik slika podatke kartice za plaćanje 4. Korisnik ne dozvoljava sustavu upotrebu kamere • Sustav prikazuje polja za unos podataka kartice za plaćanje 5. Ako korisnik nema *PayPal* račun • Odabire opciju za registraciju novog računa • Sustav preusmjerava korisnika na PayPal stranicu za registraciju putem web preglednika • Korisnik unosi podatke i registrira račun

Tablicom 3.9 detaljnije je predočen pregled kupljenih proizvoda. Kada korisnik izvrši kupovinu, u aktivnosti u administratorskom dijelu prikazuju se relevantni podaci kupovine, poput e-pošte kupca, količine kupljenih proizvoda, imena kupljenih proizvoda te njegove

ukupne cijene. Administrator ima mogućnost uklanjanja proizvoda s popisa pomicanjem njegove kartice u lijevu ili desnu stranu.

Tablica 3.9 Prikaz zahtjeva "Pregled prodanih proizvoda"

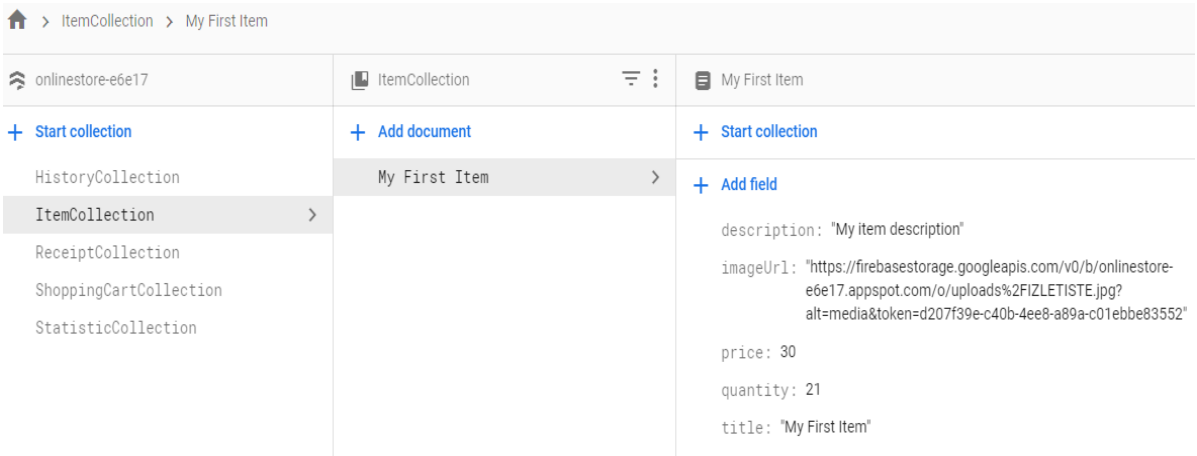
ID korisničkog zahtjeva	**10**
Naziv korisničkog zahtjeva	Pregled prodanih proizvoda
Opis korisničkog zahtjeva	Izvršavanjem kupovine, sučelje prikazuje popis prodanih proizvoda
Preduvjet	Kupovina proizvoda
Glavni scenarij	1. Korisnik izvršava kupovinu 2. Prodani proizvodi, e-pošta kupca te količina i cijena, dodaju se u popis 3. Administrator pokretanjem aktivnosti ima uvid u obavljene prodaje proizvoda te brisanje sa popisa
Alternativni scenarij	nema

4. SPECIFIKACIJE APLIKACIJE

Specifikacijama aplikacija definirane su baze podataka, Exchange Rates API te plaćanje putem PayPal-a.

4.1 Baza podataka

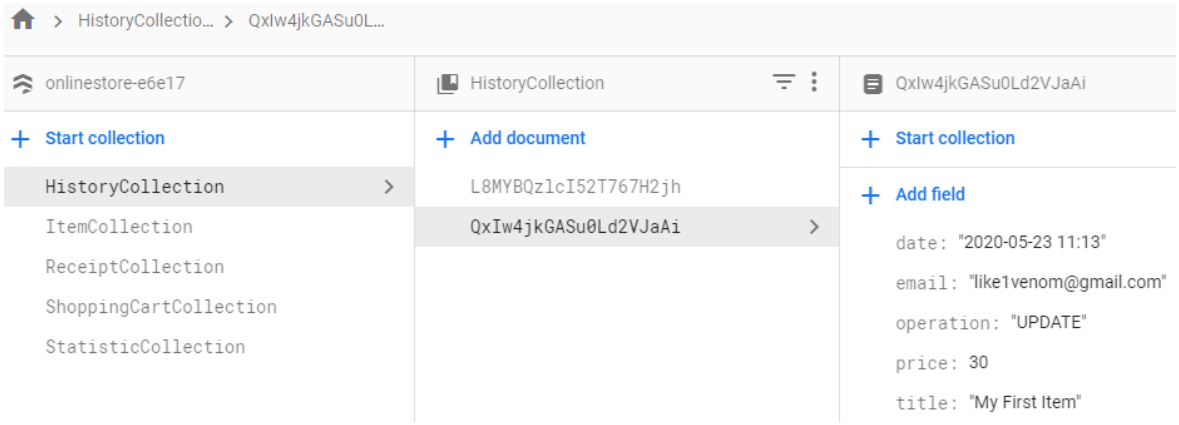
Baza podataka korištena za razvoj ove aplikacije je *Google*-ov *Firebase Firestore*. Podaci unutar baze strukturirani su u obliku kolekcija (engl. *collection*) i dokumenata (engl. *document*), koji su vrlo slični JSON-u, gdje kolekcija sadrži dokumente, a dokument se zapisuje kao objekt modela (klase) unutar aplikacije ili vodi do druge kolekcije, pod-kolekcije (engl. *sub-collection*). *Firebase Firestore* spada u tzv. *NoSQL* baze podataka te ne sadrži klasične tablice. Daljnje prednosti ove baze su izvršavanje, odnosno pamćenja upita čak i ako internet nije dostupan. U tom slučaju, upit će se izvršiti čim se uređaj ponovno spoji na internet. Dodavanjem zapisa, dokument se sprema unutar definirane kolekcije. Ukoliko ona ne postoji, automatski se generira nova kolekcija pod definiranim imenom. Kod dohvaćanja zapisa, dohvaća se cijeli dokument definirane kolekcije ili pod-kolekcije i nije moguće dohvatiti samo jedan dio podatka, bez dohvaćanja cijelog dokumenta.



Slika 4.1 Primjer baze podataka korištene u aplikaciji

Iz slike 4.1 možemo vidjeti kako su podaci unutar naše baze podataka strukturirani. Unutar naše baze podataka trenutno imamo pet kolekcija. Unutar svake kolekcije spremljeni su dokumenti sa svojim obilježjima. Naziv, odnosno ID dokumenta, može biti dodijeljen od strane programera (kao u slici 4.1) ili automatski dodijeljen u obliku *hash*-a, kako je prikazano na slici 4.2. Svaki dokument mora imati svoj (jedinstveni) ID. Ukoliko pokušamo postaviti model

dokumenta pod ID koji već postoji, zamijenit će se svi podaci iz dokumenta pod tim ID podacima iz modela.



Slika 4.2 Prikaz kolekcije *"HistoryCollection"* i automatski generiranog ID dokumenta

4.2 Exchange Rates API

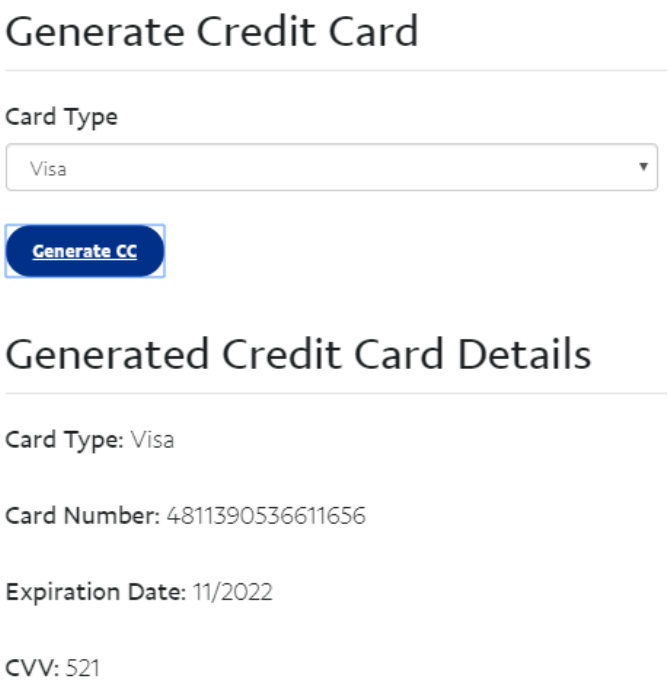
Kako ne bismo morali ručno dodjeljivati valute i njihove vrijednosti, što bi ujedno stvorilo dodatan problem ako bi u bilo kojem trenutku došlo do njihovih značajnijih promjena, za postavke, prikaz i promjenu valute, aplikacija koristi besplatni web servis pod nazivom *Exchange Rates API*. *Exchange Rates API* sadrži listu valuta korištenih u svijetu te njihove vrijednosti, a njihovo ažuriranje vrši se na dnevnoj bazi. API također omogućava i postavljanje osnovne valute za koju će vršiti računanje vrijednosti ostalih. Podaci na API strukturirani su u JSON formatu.

Dohvaćanje vrijednosti vrši se dodavanjem krajnjih točaka (engl. *Endpoints*) na osnovni URL, odnosno na glavnu hipervezu. Kako bismo dohvatili najnovije vrijednosti, kao *endpoint* dodajemo *latest.* Kako bismo postavili osnovnu valutu prema kojoj želimo dobiti vrijednosti ostalih valuta, moramo napraviti upit (engl. *Query*). Upiti se na hipervezama ostvaruju dodavanjem simbola upitnik (?), navođenjem vrste upita iza kojeg slijedi znak jednakosti (=) te dodjeljivanjem važeće vrijednosti. Na primjer, kako bismo dohvatili najnovije vrijednosti za Euro (EUR) potrebno je izvršiti sljedeći upit: *https://api.exchangeratesapi.io/latest?base=EUR*

4.3 Plaćanje putem PayPal-a

Kako je ranije spomenuto, *PayPal* je internetski orijentirana tvrtka koja u potpunosti omogućuje obavljanje uplata i novčanih prijenosa preko interneta [10]. Budući da *PayPal* podržava implementaciju unutar Android aplikacije te smo na taj način izveli plaćanje

proizvoda kupljenih putem naše aplikacije. Za primanje transakcija, potreban nam je poslovni *PayPal* račun (engl. *Business Account*), koji smo kreirali na *https://sandbox.paypal.com* stranici. U postavkama, potrebni su nam podaci koji sadrže jedinstveni ključ preko kojega spajamo naš *PayPal* račun sa aplikacijom, odnosno ključ na čiji račun ćemo vršiti transakcije. Da bismo dohvatili navedene podatke, potrebno se je prijaviti na *https://developer.paypal.com* te pristupiti *dashboard-u*, unutar kojega imamo opciju za kreirati aplikaciju, pritiskom na gumb *Create App*. Podatak koji tražimo nalazi se unutar kreirane aplikacije, ispod naziva *Client ID*. Za uspješnu simulaciju transakcije, otvaramo još jedan račun, ovaj put korisnički (engl. *Personal Account*) na *sandbox.paypal.com*. *Sandbox PayPal* nam također omogućava kreiranje i lažne kreditne kartice kako bismo mogli provjeriti plaćanja.



Slika 4.3 Primjer lažne kreditne kartice na *sandbox.paypal*

5. RAZVOJ APLIKACIJE

Organizacija razvoja aplikacije podijeljena je u tri dijela: sučelje aplikacije, administratorski pristup te korisnički pristup.

5.1 Sučelje aplikacije

Kako bi aplikacija bila smislena i prilagođena korisnicima, potrebno je izraditi sučelje. Naša aplikacija sadrži 13 aktivnosti podijeljenih na aktivnosti za ovjeru korisnika ili administratora (Tablica 5.1), aktivnosti korisničkog pristupa (Tablica 5.2) i aktivnosti administratorskog pristupa (Tablica 5.3).

Tablica 5.1 Popis aktivnosti za autentikaciju korisnika

NAZIV	OPIS
LoginActivity	Sučelje za prijavu korisnika. Sadrži dva polja za unos, gumbe za prijavu i registraciju te tekst koji vodi na *ResetPasswordActivity*.
RegistrationActivity	Sučelje za registraciju korisnika. Sadrži tri polja za unos teksta i gumb za registraciju.
ResetPasswordActivity	Sučelje za slanje link za resetiranje lozinke. Sadrži jedno polje za unos teksta i gumb za slanje uputa na unesenu poštu.

Tablica 5.2 Popis aktivnosti korisničkog pristupa

NAZIV	OPIS
UserStoreActivity	Korisničko sučelje koje sadrži popis svih proizvoda u trgovini.
ItemDetailsActivity	Korisničko sučelje koje prikazuje detalje za odabrani proizvod. Sadrži polje za unos brojeva, dva gumba za povećavanje ili umanjivanje količine i gumb za dodavanje proizvoda u košaricu.
ShoppingCartActivity	Korisničko sučelje koje prikazuje sve proizvode koje je korisnik dodao u košaricu. Sadrži gumb koji vodi na *PayPal* plaćanje.
ReceiptActivity	Korisničko sučelje koje prikazuje popis svih izvršenih kupovina i njihovih trošarina.

Tablica 5.3 Popis aktivnosti administratorskog pristupa

NAZIV	OPIS
MainActivity	Administratorsko sučelje koje sadrži glavni izbornik sa tri gumba koji vode na *StoreActivity*, *HistoryActivity* i *StatisticActivity*.
StoreActivity	Administratorsko sučelje koje sadrži listu predmeta na trgovini te gumb koji vodi na *AddEditActivity*.
HistoryActivity	Administratorsko sučelje koje sadrži povijest operacija izvršenih od administratora.
StatisticActivity	Administratorsko sučelje koje sadrži statistiku operacija izvršenih od administratora u nekim vremenskim periodima.
AddEditActivity	Administratorsko sučelje namijenjeno dodavanju i ažuriranju proizvoda na trgovini. Sadrži četiri polja za unos teksta, gumb za odabir slike iz telefonske memorije, gumb za učitavanje slike na online skladište, traku za prikazivanje napretka učitavanja te gumb za pohranjivanje unesenih podataka.
SoldItemsActivity	Administratorsko sučelje koje sadrži popis svih prodanih proizvoda, cijenu te e-poštu kupca.

5.2 Administratorski pristup

U administratorskom pristupu, dodavanje i preuređivanje proizvoda vrši se preko iste aktivnosti, što je i prikazano programskim kôdom 5.1. Kako bismo odijelili jednu operaciju od druge, pokrenuli smo aktivnost za rezultat pomoću *startActivityForResult()* naredbe. Na gumb za dodavanje te na odabir proizvoda predali smo različite *request* kôdove. Nakon izvršavanja aktivnosti koja je pokrenuta preko *startActivityForResult()*, *onActivityResult()* metoda dohvatit će *request* kôdove ako je rezultat aktivnosti uspio te ovisno o svakom od njih izvršiti određeni dio kôda.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == MyConstants.REQUEST_ADD_ITEM && resultCode == RESULT_OK
&& data != null) {
        actionAddItem(data);
        createAddHistoryEntry(storeItem);
        addToStatistic();

    } else if (requestCode == MyConstants.REQUEST_EDIT_ITEM && resultCode ==
RESULT_OK && data != null) {
        actionUpdateItem(data);
        createUpdateHistoryEntry(storeItem);
        updateToStatistic();
    }
}
```

Programski kôd 5.1 Prikaz grananja zahtjeva

Za *request* dodavanja proizvoda, naredbom *actionAddItem()* dodali smo proizvod u našu bazu podataka, naredbom *createAddHistoryEntry()* zapisali smo ga u povijest i pridijelili mu naziv operacije dodavanja (*ADD*), dok smo naredbom *addToStatistic()* uvećali broj operacije dodavanja za jedan te ga tako zapisali. Isti postupak primijenjen je i za ažuriranje.

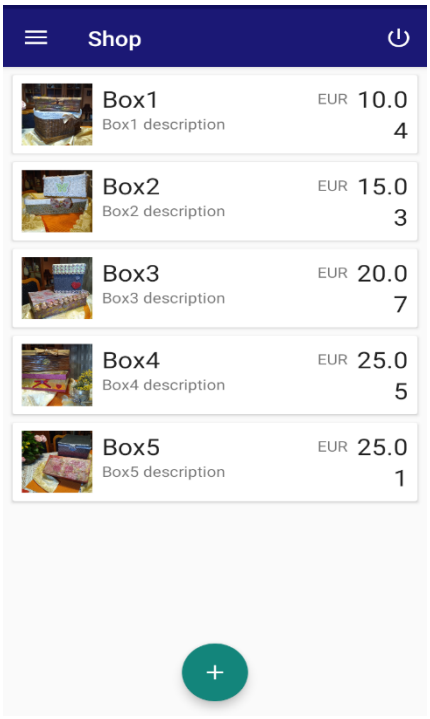
Za brisanje proizvoda sa trgovine, implementirali smo „*swipe to delete*“ funkcionalnost. Kao što i sam naziv govori, povlačenjem kartice sa proizvodom u stranu možemo ga izbrisati. *Firebase Firestore* ima mogućnost izvršavanja, tj. „pamćenja“ zahtjeva ako aplikacija nema pristup internetu. Na taj način korisnik može vršiti promjene nad proizvodima, a zahtjevi će se obraditi čim se uređaj spoji na mrežu. *Firebase Storage*, koji nam služi za pohranu slika proizvoda, nažalost nema tu mogućnost te moramo brinuti o situaciji što će se dogoditi ako internet nije dostupan. U slučaju kada bismo obrisali proizvod bez pristupa internetu, naša aplikacija bi prestala raditi. U programskom kôdu 5.2 prikazana je implementacija *swipe to delete* funkcionalnosti te je jedno od rješenja ovog problema. Povlačenje kartice omogućeno je metodom *SimpleCallback(),* unutar klase *ItemTouchHelper* te dodjeljivanjem prikladnih argumenata definiranih unutar klase (*ItemTouchHelper.LEFT, ItemTouchHelper.RIGHT*). Metoda *onMove()* se pokreće kada pritisnemo na karticu i povlačimo ju u tako specificiranu stranu. U našem slučaju ta metoda nema dodatnih implementiranih funkcionalnosti. Metoda *onSwiped()* izvršava se kada *swipe*-amo karticu proizvoda, odnosno kada karticu povučemo do kraja zaslona te je otpustimo. Ta će se kartica automatski ukloniti sa zaslona bez ikakvog dodatnog kôda, međutim, za uklanjanje podataka iz baze podataka, potrebno je to dodatno razraditi.

```
new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT
| ItemTouchHelper.RIGHT) {
    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull
RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int
direction) {
        if (isNetworkConnected()) {
            deleteStoreItem(viewHolder);
        } else {
            adapter.notifyItemChanged(viewHolder.getAdapterPosition());
        }
    }
}).attachToRecyclerView(recyclerView);
```

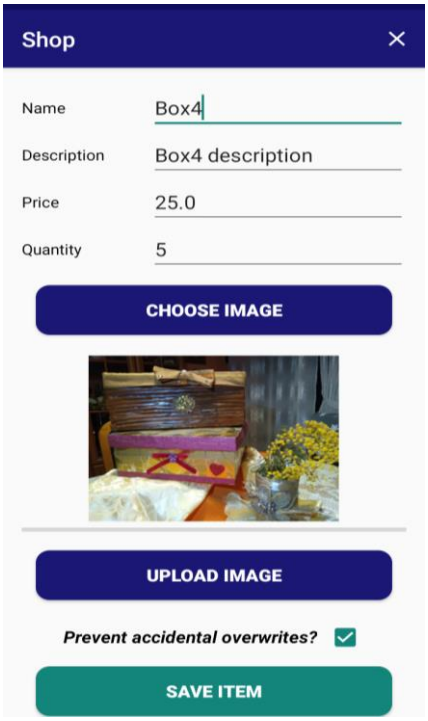
Programski kôd 5.2 Prikaz *"swipe to delete"* funkcionalnosti

Boolean metoda *isNetworkConnected()* dohvaća trenutno stanje mreže te ovisno o pristupu internetu vraća istinu ili laž. U slučaju istine, predmet se briše zajedno sa slikom, u protivnom se izvršava *notifyItemChanged()* metoda koja karticu vraća u njen prvotni položaj i onemogućuje brisanje. Na slici 5.1 prikazan je pregled trgovine unutar *StoreActivity*, dok je na slici 5.2 prikazan primjer dodavanja proizvoda unutar *AddEditAcivity*. *Prevent accidental overwrites* kućica, ukoliko označena, provjerava, obavještava i sprječava kreiranje predmeta pod imenom koje već postoji kako administrator ne bi nehotice preuredio neki drugi predmet. Uklanjanjem oznake kućice, mehanizam se gasi.

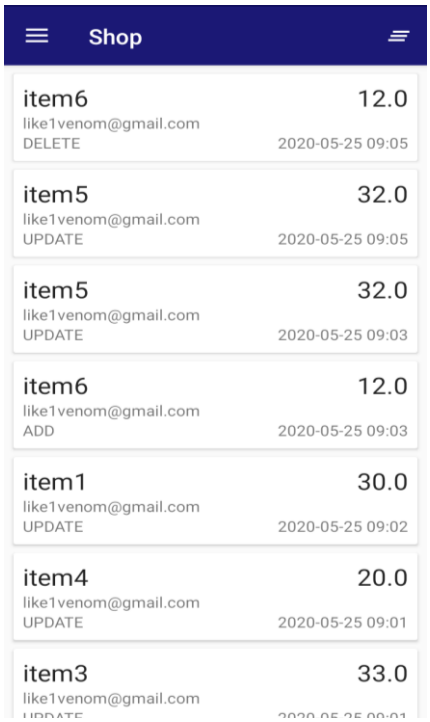
Kreiranje, odnosno dodavanje novog proizvoda, kao i njegovo ažuriranje ili uklanjanje, dodaje se u povijest (*HistoryActivity*) i statistiku (*StatisticActivity*). Kartica povijesti sadrži naziv predmeta, naziv operacije nad proizvodom, e-poštu administratora koji je izvršio operaciju na proizvodu, prethodnu cijenu proizvoda i datum izvršavanja operacije (Slika 5.3). Aktivnost statistike sadrži vremenski organiziranu listu kartica sa datumima izvršavanja operacija. Ispod svakog datuma nalazi se ukupan broj svake od operacija izvršenih na taj dan. Aktivnost također sadrži i mogućnosti pretraživanja podataka. Implementirane postavke filtriranja podataka uključuju: pretraživanje prema trenutnom mjesecu, pretraživanje po odabranom mjesecu, pretraživanje po datumu, pretraživanje broja najsvježijih podataka te prikaz svih podataka (Slika 5.4).



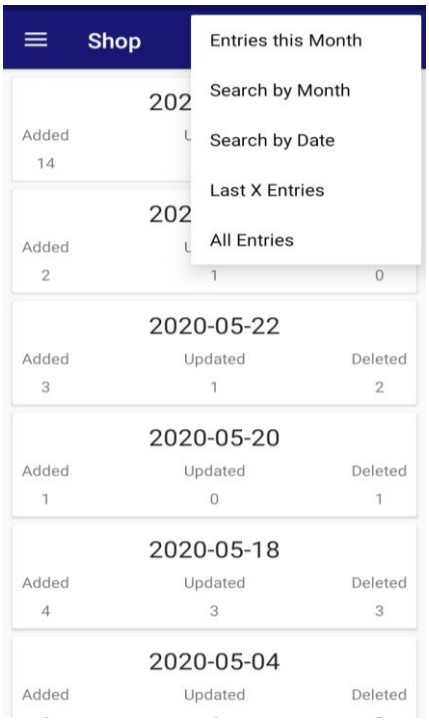
Slika 5.1 Popis proizvoda u *StoreActivity*



Slika 5.2 Dodavanje predmeta u *AddEditActivity*

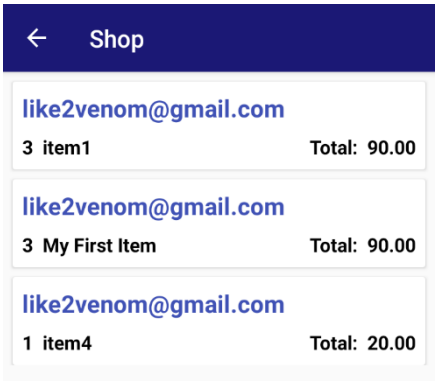


Slika 5.3 Prikaz povijesti u *HistoryActivity*



Slika 5.4 *StatisticActitiy* sa popisom postavki pretraživanja

Uspješnim izvršavanjem kupovine (i plaćanjem), kupljeni proizvodi, sa njihovom količinom, cijenom i e-poštom kupca, dodaju se u posebnu aktivnost. Aktivnost također omogućava brisanje pojedinih proizvoda sa popisa povlačenjem kartice proizvoda u lijevu ili u desnu stranu (Slika 5.5).

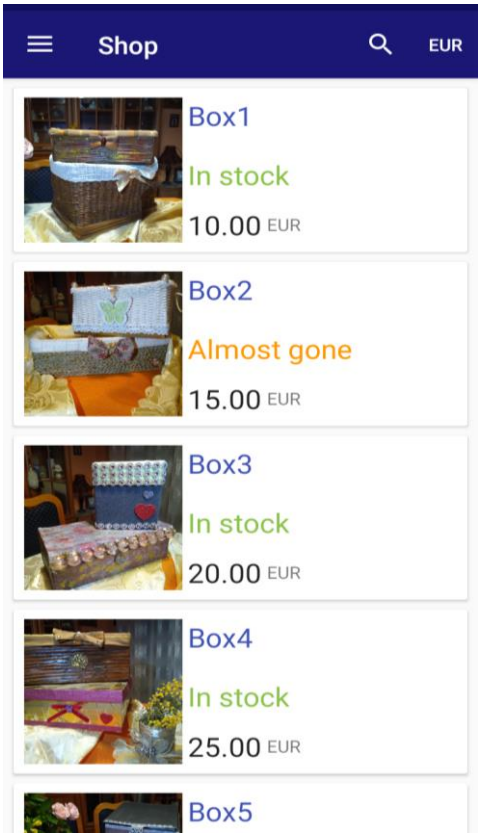


Slika 5.5 Prikaz popisa kupljenih proizvoda u *ItemsSoldActivity*

5.3 Korisnički pristup

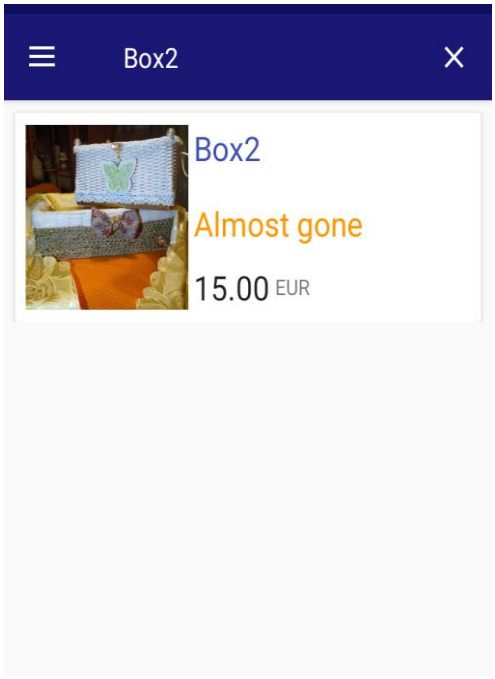
U korisničkom dijelu aplikacije potrebno je izraditi sučelje za korisnika, odnosno kupca, preko kojega bi on mogao imati pristup svim relevantnim informacijama za uspješno izvršavanje kupovine. Ti zahtjevi uključuju izradu sučelja za prikaz popisa ponuđenih predmeta na trgovini, njihovo sortiranje i pretraživanje, košaricu za dodavanje željenih proizvoda i količine, preračunavanje cijene proizvoda, kupovinu te prikazivanje svih računa koje je izvršio s pripadajućim *PayPalID* transakcije ukoliko bi došlo do poteškoća. Pritiskom na karticu računa kopira se njegov *PayPalID* transakcije u među-spremnik kako bi njegov prijenos bio što jednostavniji.

Proizvodi na slici 5.6 mijenjaju boju i tekst u ovisnosti o broju preostalih raspoloživih proizvoda. Za više od 3 proizvoda, tekst će prikazivati zelenom bojom da je proizvod dostupan. Za 3 ili manje preostalih proizvoda na skladištu, tekst će narančastom bojom prikazivati da je uskoro rasprodano. Za proizvod čija preostala količina iznosi 0, tekst će crvenom bojom prikazivati da je proizvod rasprodan. Odabirom proizvoda iz slike 5.6 otvara se aktivnost sa relevantnim detaljima proizvoda kao na slici 5.7, preko koje proizvod možemo (i njegovu količinu) dodati u našu košaricu. *UserStoreActivity* također ima mogućnost pretraživanja proizvoda, njihovo sortiranje, kao i promjenu valute. Na slici 5.8 prikazano je pretraživanje, dok je na slici 5.9 prikazan primjer promjene valute.

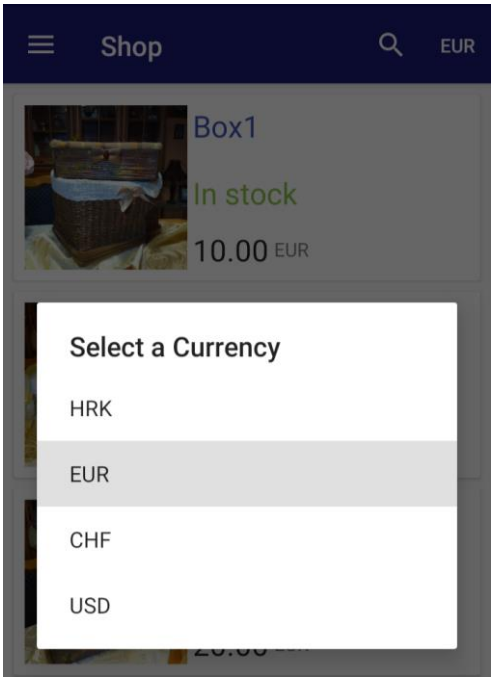


Slika 5.6 Prikaz proizvoda u *UserStoreActivity*

Slika 5.7 Prikaz detalja proizvoda u *ItemDetailsActivity*



Slika 5.8 Primjer pretraživanja proizvoda



Slika 5.9 Primjer promjene valute

Programski kôd 5.3 prikazuje implementaciju sustava za pretraživanje proizvoda - *SearchView*, koji za upis svakog slova poziva svoju *onQueryTextChange()* metodu (Programski kôd 5.4), kreira upit i prosljeđuje ga. Pozivanje navedene metode vrši se postavljanjem osluškivača (engl. *listener*). Pri poništavanju pretraživanja pokreće se *onClose()* metoda, koja ponovno učitava popis svih proizvoda trgovine.

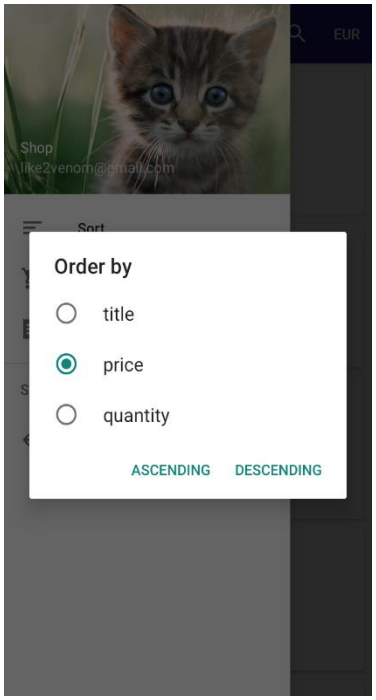
```
MenuItem searchItem = menu.findItem(R.id.menu_user_store_search_view);
SearchView searchView = (SearchView) searchItem.getActionView();
searchView.setOnQueryTextListener(this);
searchView.setOnCloseListener(new SearchView.OnCloseListener() {
    @Override
    public boolean onClose() {
        setUpRecyclerView();
        return false;
    }
});
```

Programski kôd 5.3 Prikaz sustava za pretraživanje - *SearchView*

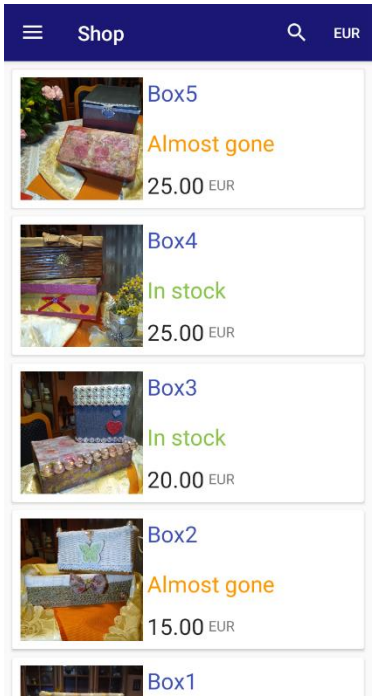
```
@Override
public boolean onQueryTextChange(String q) {
    Query query = storeRef
            .orderBy(MyConstants.ITEM_NAME_ORDER_BY,
Query.Direction.DESCENDING)
            .whereEqualTo(MyConstants.ITEM_NAME, q);

    buildStoreItemRecycler(query);
    adapter.updateOptions(options);
    return false;
}
```

Programski kôd 5.4 Prikaz *onQueryTextChange()* metode

Na slici 5.10 prikazan je dijalog sa opcijama za sortiranje. Dijalog nudi mogućnosti sortiranja prema nazivu, cijeni i količini preostalih proizvoda na trgovini u uzlaznom ili silaznom redoslijedu. Odabiranjem opcije sortiranja prema cijeni (engl. *price*) te pritiskom na gumb za silazni redoslijed (engl. *descending*), dobivamo rezultat sortiranja prikazan na slici 5.11. Programski kôd 5.5 prikazuje implementaciju dijaloga za sortiranje, koji smo kreirali nasljeđivanjem klase *DialogFragment*. Zadaća navedenog dijaloga je dohvatiti odabrane rezultate te ih primijeniti na upit (*Query*), odnosno izgradnju nove liste preko navedenog upita. Odabiranjem opcije pokreće se *onClick()* metoda, koja sprema vrijednost iz polja za slaganje prema nazivu, cijeni ili količini u *String*, koji će predati kao argument za slaganje upita (*Query*). *Positive* gumb i *negative* gumb smo preuredili tako da svaki od njih šalje različit smjer upita (*Query.Direction*) – *Positive* gumb šalje silazni smjer, dok *negative* gumb šalje uzlazni.



Slika 5.10 Dijalog za sortiranje proizvoda [11], [3]



Slika 5.11 Rezultat sortiranja proizvoda prema cijeni

```
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    builder.setTitle(R.string.dialog_sort_title);
    builder.setSingleChoiceItems(sortList, 0, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            selectedItemIndex = which; }
    });
    builder.setPositiveButton(R.string.dialog_descending, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            String order = sortList[selectedItemIndex];
            Query query = itemRef.orderBy(order, Query.Direction.DESCENDING);
            updateQuery(query); }
    });
    builder.setNegativeButton(R.string.dialog_ascending, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            String order = sortList[selectedItemIndex];
            Query query = itemRef.orderBy(order, Query.Direction.ASCENDING);
            updateQuery(query); }
    });
```

Programski kôd 5.5 Prikaz dijaloga za sortiranje - *SortDialog*

Dohvaćanje vrijednosti valuta izveden je preko knjižnice *Retrofit2*, koja ima mogućnost dohvaćanja podataka u JSON obliku putem željenog API. Kako je navedeno ranije, korišteni API je *Exchange Rates API*. Poziv preko *Retrofit* knjižnice obavlja se sastavljanjem hiperveze. Na osnovni dio hiperveze (*baseURL*) dodaju se krajnje točke (engl. *endpoints*) te upiti (engl. *query*). Kako je spomenuto u prethodnom poglavlju, osnovni dio hiperveze predstavlja *https://api.exchangeratesapi.io/*, na koji se dodaje krajnja točka (engl. *Endpoint*) *latest*, a vrsta upita (engl. *Query*) slijedi iza simbola upitnika (?). U našem slučaju, za postavljanje osnovne valute koristimo *base*, čiju vrijednost definiramo iza znaka jednakosti (=) te potpuna hiperveza izgleda ovako: *https://api.exchangeratesapi.io/latest?base=EUR*. Programski kôd 5.6 prikazuje ostvarivanje poziva.

```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(converterInterface.baseURL)
        .addConverterFactory(GsonConverterFactory.create())
        .build();
converterInterface = retrofit.create(ConverterInterface.class);
Call<Converter> call =
converterInterface.getRates(converterInterface.baseCurrencyName);
call.enqueue(new Callback<Converter>() {
    @Override
    public void onResponse(Call<Converter> call, Response<Converter> response){
        Converter converter = response.body();
        createCurrencyMap(converter);
        setUpRecyclerView();
    }
    @Override
    public void onFailure(Call<Converter> call, Throwable t) {
        Toast.makeText(getApplicationContext(), t.getMessage(),
      Toast.LENGTH_SHORT).show();
    }
});
```

Programski kôd 5.6 Ostvarivanje poziva preko *Retrofit*-a

Budući da *sandbox.paypal* ne omogućava plaćanje u hrvatskim kunama (HRK), već dozvoljava plaćanje samo određenim valutama (poput eura), za glavni račun i prikaz cijene u košarici sa odabranim proizvodima, unutar aktivnosti *ShoppingCartActivity*, koristili smo valutu euro. Slika 5.12 pobliže prikazuje aktivnost *ShoppingCartActivity*. Sa lijeve strane kartice ispod naslova svakog proizvoda nalazi se njegova pojedinačna cijena, u sredini se nalazi odabrana količina proizvoda, dok se sa desne strane nalazi umnožak pojedinačne cijene i količine, odnosno ukupna cijena proizvoda. Proizvode iz košarice na ovoj aktivnosti možemo ukloniti povlačenjem njihovih kartica u lijevu ili u desnu stranu, a pritiskom na njih pokrećemo aktivnost za prikazivanje detalja proizvoda (*ItemDetailsActivity*), u kojoj možemo promijeniti

željenu količinu proizvoda. Pritiskom na gumb *„To Payment Options“* izvršava se programski kôd 5.7, odnosno pokreće se *PayPal* servis i otvara *PayPal* stranica za odabir metode plaćanja.

Slika 5.12 Prikaz odabranih proizvoda u *ShoppingCartActivity*

```
PayPalConfiguration config = new PayPalConfiguration()
        .environment(PayPalConfiguration.ENVIRONMENT_SANDBOX)
        .merchantName(MyConstants.PAYPAL_MERCHANT_NAME)
        .clientId(MyConstants.PAYPAL_CLIENT_ID);

Intent paypalIntent = new Intent(this, PayPalService.class);
paypalIntent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION, config);
startService(paypalIntent);

paymentAmount = tvTotalValue.getText().toString();
payment = new PayPalPayment(new BigDecimal(String.valueOf(paymentAmount)),
MyConstants.BASE_CURRENCY_NAME, getString(R.string.paypal_total),
PayPalPayment.PAYMENT_INTENT_SALE);

Intent intent = new Intent(this, PaymentActivity.class);
intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION, config);
intent.putExtra(PaymentActivity.EXTRA_PAYMENT, payment);
startActivityForResult(intent, MyConstants.PAYPAL_REQUEST_CODE);
```

Programski kôd 5.7 Pokretanje *PayPal* servisa i aktivnosti

Kreiranjem objekta *config*, iz *PaypalConfiguration* klase, definiramo postavke poput okruženja (*environment*), imena trgovca (*merchantName*) te identifikacijske oznake trgovca (*clientId*). Kako smo u prethodnom poglavlju naveli, koristimo *paypal.sandbox* za našu aplikaciju, a *clientId* je identifikacijska oznaka koju smo preuzeli u postavkama računa preko *developer.paypal* stranice. Pokretanjem aktivnosti za rezultat, nakon izvršavanja transakcije,

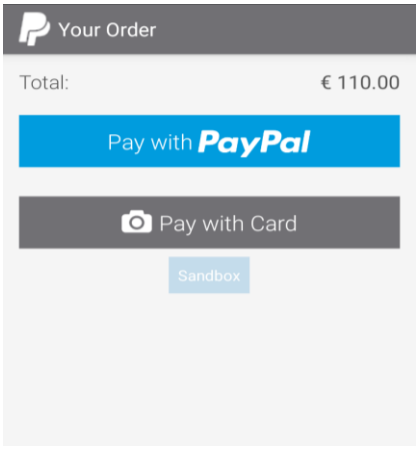
vrši se spremanje računa u *ReceiptActivity*, broj preostalih proizvoda na skladištu trgovine umanjuje se za broj kupljenih, šalje se popis kupljenih proizvoda administratoru te se vrši pražnjenje košarice i povratak na pregled trgovine, što je prikazano u programskom kôdu 5.8.

Slika 5.13 prikazuje pokretanje *PayPal* servisa sa ponuđenim metodama plaćanja. Gumb "*Pay with PayPal"*, otvara stranicu za prijavu na *PayPal* račun. Od kupca se zahtijeva unos podataka njegovog *PayPal* računa, a uspješnom prijavom na *PayPal* račun, otvara se aktivnost za potvrdu plaćanja sa ukupnim iznosom cijene (Slika 5.14). Pritiskom na gumb za plaćanje ("*Pay"*), *PayPal* pokreće transakciju. Gumb "*Pay with Card"* otvara aktivnost za slikanje kreditne kartice, ali nudi mogućnost unosa podataka preko tipkovnice pritiskom na gumb "*Keyboard"*. Unošenjem ispravnih podataka te pritiskom gumba "*Done"*, otvara se stranica za potvrdu plaćanja. Pokretanje transakcije vrši se pritiskom na gumb "*Charge Card"*, a po završetku transakcije, korisnika (kupca) se vraća na pregled trgovine.

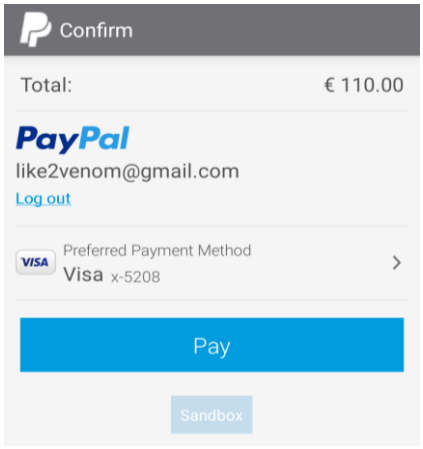
```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
 Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == MyConstants.PAYPAL_REQUEST_CODE) {
        if (resultCode == Activity.RESULT_OK) {
            PaymentConfirmation confirm =
data.getParcelableExtra(PaymentActivity.EXTRA_RESULT_CONFIRMATION);
if (confirm != null) {
   createReceipt(confirm);
   updateStorageQuantity();
   createSoldItemEntry();
   clearShoppingCartItems();
   startActivity(new Intent(getApplicationContext(), UserStoreActivity.class));
            }
        }
    }
}
```

Programski kôd 5.8 Postupak izvršavanja naredbi nakon potvrđene uplate

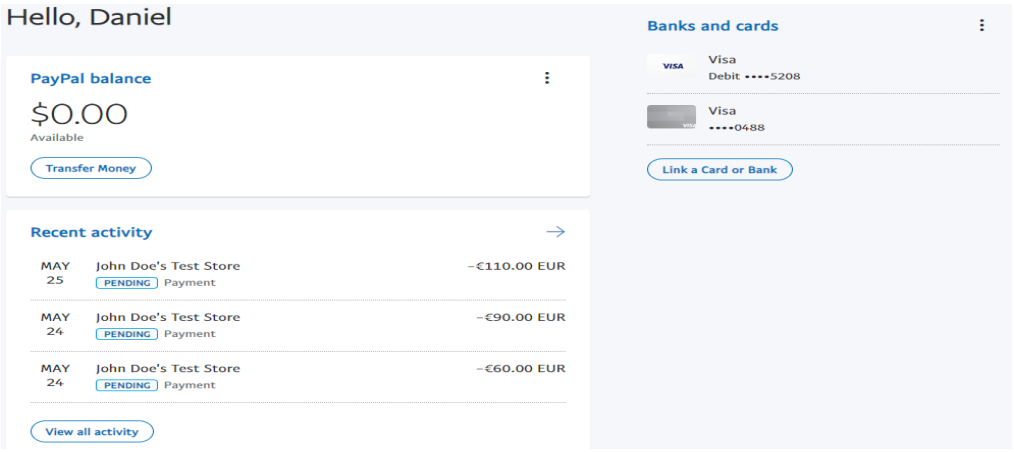
Po uspješno obavljenoj kupovini, proizvodi se knjiže i prikazuju u popisu u administratorskom dijelu, a korisnik pokretanjem aktivnosti *ReceiptActivity* dobiva popis njegovih izvršenih kupovina, sa svojim pripadajućim identifikatorima transakcije. Aktivnost, radi jednostavnijeg prijenosa i predočenja identifikatora kupovine, pritiskom na karticu, automatski kopira identifikator u među-spremnik. Izvan aplikacije, korisnik također može provjeriti transakcije na svom *PayPal* računu, kao i stanje računa, pregled povezanih kreditnih kartica ili banki, kako je prikazano na slici 5.15.



Slika 5.13 Metode plaćanja



Slika 5.14 Potvrda plaćanja putem *PayPal-a*



Slika 5.15 Prikaz obavljenih transakcija na *sandbox.paypal* korisničkom računu

6. ZAKLJUČAK

Osim dobrog poznavanja Java programskog jezika, izrada ovog završnog rada zahtijevala je dodatno istraživanje i proučavanje korištenih tehnologija. Prije same izrade aplikacije, potrebno je bilo detaljno isplanirati aplikaciju te zahtjeve za administratora i korisnika. U fazi proučavanja, bilo je potrebno detaljno razumijevanje rada *Google*-ove platforme *Firebase,* kao i dobro poznavanje rada sa *Retrofit* i *PayPal* knjižnicama. Ova mobilna aplikacija osmišljena je za lakše upravljanje trgovinom, vođenja povijesti dodavanja, izmjene ili brisanja proizvoda trgovine te prikazivanje organizirane statistike navedenih operacija izvršenih u nekom vremenskom periodu. Aplikacija također simulira i rad trgovine te odvijanje procesa prilikom postavljanja, pregleda i pretraživanja proizvoda, dodavanja željenih proizvoda u košaricu pa sve do zaključivanja kupovine implementiranim oblicima internetskog plaćanja putem *PayPal* usluge.

LITERATURA

[1] Wikipedia, Java (programming language), https://en.wikipedia.org/wiki/Java_(programming_language), svibanj 2020

[2] StackOverflow, Dynamically compiled language vs statically compiled language, https://stackoverflow.com/questions/12600296/dynamically-compiled-language-vs-statically-compiled-language, svibanj 2020

[3] K. Sierra, B. Bates, Head First Java 2nd Edition, O'Reilly Media, Sebastopol CA, 2005

[4] D. Griffiths, D. Griffiths, Head First Android Development: A Brain Friendly Guide 2nd Edition, O'Reilly Media, Sebastopol CA, 2005Wikipedia, XML,

[5] https://hr.wikipedia.org/wiki/XML, svibanj 2020

[6] Medium, What is Firebase? The complete story, abridged, https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0, svibanj 2020

[7] Firebase, Firebase Authentication, https://firebase.google.com/docs/auth, svibanj 2020

[8] Firebase, Firebase Firestore, https://firebase.google.com/docs/firestore, svibanj 2020

[9] Firebase, Firebase Storage, https://firebase.google.com/docs/storage, svibanj 2020

[10] Wikipedia, PayPal, https://hr.wikipedia.org/wiki/PayPal, svibanj 2020

[11] Head First Android Development, https://dogriffiths.github.io/HeadFirstAndroid/#/resources, svibanj 2020

SAŽETAK

U ovom završnom radu razvijena je Android aplikacija za upravljanje resursima trgovine. Aplikacija je pisana u programskom jeziku Java uz prisustvo drugih knjižnica kao što su *Retrofit*, *ButterKnife*, *Glide*, *PayPal* i *Firebase*. Aplikacija je podijeljena na dva dijela: administratorski dio i korisnički dio. Administrator ima mogućnost postavljanja, ažuriranja i brisanja predmeta trgovine, kao i uvid u povijest i statistiku dodavanja ili ažuriranja predmeta, te popis kupljenih predmeta. Korisnički dio ima mogućnost pregleda trgovine, dodavanja ili uklanjanja predmeta iz košarice te kupovinu predmeta prethodno dodanih u košaricu.

Ključne riječi: Firebase, kupovina, PayPal, trgovina, upravljanje

ABSTRACT

In this bachelor's thesis, an Android application for store resource management has been developed. The application was written in Java programming language with addition of other libraries like *Retrofit*, *ButterKnife*, *Glide*, *PayPal* and *Firebase*. The application is split into two parts: Admin Access and User Access. The administrator has the possibility of adding, updating and deleting store items but also an access to item's addition or update history and statistics, as well as listings of bought items. The user access has the possibility of viewing the store, adding or removing items from the shopping cart as well as buying items previously added to the cart.

Keywords: Firebase, management, PayPal, shop, shopping

ŽIVOTOPIS

Daniel Jursik rođen je 1. travnja 1992. godine u Zagrebu. 2007. godine završava osnovnu školu Braće Radića u Pakracu. 2011. godine završava opću gimnaziju u srednjoj školi Pakrac uz polaganje ispita državne mature. 2012. godine upisuje Elektrotehnički fakultet na Sveučilištu Josipa Jurja Strossmayera u Osijeku, stručni studij elektrotehnike, smjer informatika.