Google Firebase priručnik kroz vlastiti primjer

Popović, Marin

Sveučilište u Rijeci – Odjel za informatiku

Sveučilišni prijediplomski studij Informatika

Marin Popović

Google Firebase priručnik kroz vlastiti primjer

Završni rad

Mentor: doc. dr. sc. Danijela Jakšić

Rijeka, 17.7.2023

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 14.6.2022.

Zadatak za završni rad

Pristupnik: Marin Popović

Naziv završnog rada: Google Firebase priručnik kroz vlastiti primjer

Naziv završnog rada na eng. jeziku: Google Firebase tutorial

Sadržaj zadatka: Cilj ovog diplomskog rada je napraviti pregled alata Google Firebase, za rad s NoSQL bazama podataka. Potrebno je opisati Google Firebase platformu te kroz praktične primjere prikazati njene načine konfiguracije, značajke, arhitekturu, funkcionalnosti, upitni jezik, sigurnosne mogućnosti i podatkovne alate.

Mentor

doc. dr. sc. Danijela Jakšić

Voditelj za završne radove

doc. dr. sc. Miran Pobar

Zadatak preuzet: 14.6.2022.

Popović

(potpis pristupnika)

SADRŽAJ

KLJUČNE RIJEČI

SAŽETAK

Ovaj rad bavi se pregledom Google Firebase-a, platforme za razvoj mobilnih i web we koja nudi mnoge usluge kao što su pohrana u oblaku, autentifikacija, Google Analytics i razne integracije. Opisat će se povijesti, funkcionalnosti i mogućnosti Firebase-a, s naglaskom na korištene alate, uključujući Django i biblioteku Pyrebase. Objasnit će se proces stvaranja projekta u Firebase-u, zajedno s ulogama datoteka u Djangu i strukturom baze podataka. Osim toga, projekt implementiran u Firebase-u obuhvatit će značajke za upravljanje informacijama o knjigama, interakcijama korisnika i administrativnim zadacima. Koristit će se i strukturirani i nestrukturirani tipovi podataka, kao što su tekstovi, datumi, brojevi i slike. Tijekom rada, relevantni dijelovi koda bit će popraćeni slikom i kratkim ili detaljnim objašnjenjem.

1. UVOD

Firebase je platforma za izradu mobilnih i web aplikacija, podružnica je Google-a. Platforma je skup backend cloud usluga i platforma za razvoj aplikacija. U njoj se nalaze baze podataka, usluge, autentifikacije i integracije za razne aplikacije (Wikipedia, Firebase, 2023). Programski jezici koje Firebase podržava su: Java, Python, Node.js, Go, PHP, C#, and Ruby (Firebase, Cloud Firestore Library and framework integrations, 2023). Uz to, podržane su i integracije third-party biblioteke (AngularFire, FirebaseUI, React Native Firebase) (Firebase, Cloud Firestore Library and framework integrations, 2023).

Kompaniju su 2011. osnovali Andrew Lee i James Tamplin, a kupio ju je Google 2014. Tim se nalazi u San Franciscu i Mountain View-u u Kaliforniji. Kompanija se u početku zvala Envolve, prijašnji start-up osnivača firme. Envolve je pružala programerima API koji je omogućavao integraciju funkcionalnosti online chat-a na njihove web stranice. Kasnije, su Lee i Tamplin odlučili odvojiti sustav za chat i real-time arhitekturu koja ga je pokretala, osnovavši Firebase kao zasebnu tvrtku (travanj, 2012) (Wikipedia, Firebase, 2023).

Podaci se pohranjuju i sinkroniziraju u NoSQL bazi podataka u JSON formatu (Firebase, Store and sync data in real time, 2023). Firebase nudi dvije vrste baze podataka, Realtime Database i Cloud Firestore (Firebase, Choose a Database: Cloud Firestore or Realtime Database, 2023). Realtime Database sprema podatke kao JSON stablo i pri dodavanju novog podatka stvara se novi čvor u JSON strukturi s pripadajućim ključem, ključ može biti stvoren od strane korisnika ili može biti automatski generiran. Prilikom dohvaćanja podatak iz stabla na pojedinoj lokaciji, podaci iz svih podčvorova tog čvora budu vraćeni, što može dovesti do dugog čekanja kod duboko ugniježđenog stabla (Firebase, Getting started with the Firebase Realtime Database on the web, 2022). Cloud Firestore je novija baza podataka, pruža fleksibilne, hijerarhijske strukture podataka. Podaci se sastoje od dokumenata i zbirka. Dokumenti se spremaju unutar zbiraka. Obije baze podataka podržavaju mnogo različitih vrsta podataka, na primjer, string, brojevi, ugniježđeni objekti, itd (Firebase, Cloud Firestore, 2023).

Firebase nudi autentifikaciju putem emaila kao i drugih pružatelja identiteta, kao što su Google, Facebook, Twitter i Apple. Osim toga, korisnici mogu koristiti svoj telefonski broj kao način prijave. Anonimna autentifikacija je još jedan način autentifikacije korisnika, ona korisniku daje privremeni račun (Firebase, Firebase Authentication, 2023).

"Storage" (spremište) se koristi za spremanje slikovnih, audio ili video datoteka, ovaj sadržaj se izravno učitava iz web preglednika (Firebase, Getting started with Firebase Storage on the web, 2022).

Platforma nudi ekstenzije koje pomažu programeru implementiranje funkcionalnosti za aplikacije. Ekstenzije su, u svojoj osnovi, kod koji izvršava zadatak kad god se dogodi specifično definiran događaj u aplikaciji (Firebase, Firebase Extensions, 2023).

Neke od Firebase-ovih ekstenzija su: "Delete User Data" (brisanje korisničkih podataka), "Shorten URLs in Firestore" (skraćivanje URL-a), "Trigger Email from Firestore" (sastavljanje i slanje email-a), "Resize Images" (mijenjanje veličina slika) i "Translate Text in Firestore" (prevođenje teksta) (Firebase, Firebase Extensions Hub, 2023).

Google Analitycs je usluga koja se prati i izvješćuje o prometu na web stranicama i mobilnim aplikacija (Wikipedia, Google Analytics, 2023). Usluga integrira značajke Firebasea i pruža neograničeno izvještavanje do 500 različitih događaja koje programer može definirati pomoću Firebase SDK-a (Firebase, Google Analytics, 2023).

U ovome radu napravljena je knjižnica kao web stranica. Korisnik može pretraživati knjige, posuđivati ili ih dodati na popis želja, dok administrator dodaje, ažurira i briše podatke. Kao okvir (framework) korišten je Django, te je korištena biblioteka Pyrebase za povezivanja Djanga s Firebase bazom podataka. Korišteni programski jezici u ovome projektu su Python, JavaScript i HTML/CSS/Bootstrap.

Opisat ćemo sam proces stvaranja projekta u Firebase-u i Djangu, i strukturu baze podataka. Unutar projekta opisujemo: biblioteku Pyrebase i njene funkcije, prikaz svih i individualnih podataka (knjiga), filtriranje podataka, autentifikaciju, obrasce, posudbu/dodavanje na popis želja, administratorsku stranicu, te promjenu korisničkih podataka. Na kraju kroz nekoliko slika prikazujemo aplikaciju.

2. STVARANJE PROJEKTA

Za izradu novog Firebase projekta korisnik najprije mora kliknuti "Add project" na početnoj stranici (console.firebase.google.com), zatim treba dati naziv projektu i kliknuti gumb "Create project", po izboru se može uključiti Google Analytics (Firebase, Firebase console, 2023).

Nakon što je projekt kreiran korisnik mora izraditi aplikaciju, dane su četiri opcije: iOS, Android, web ili Unity aplikacija. Projekt izrađen u ovom radu je web aplikacija. Kao i kod stvaranja projekta, aplikaciji moramo dodijeliti ime.

Ovdje se također može omogućiti "Firebase hosting", hosting usluga za statički i dinamički sadržaj, kao i mikrousluga (na primjer, pružanje Firebase poddomena) (Firebase, What can you do with Firebase Hosting?, 2023).

Nakon registracije aplikacije, osiguravaju se SDK (Software development kit) i konfiguracijski objekt. Konfiguracijski objekt koristit će se za povezivanje aplikacije s resursima Firebase projekta, a SDK naredba: "npm install firebase" koristi se za instalaciju Firebase-a u uređivaču koda (Firebase, Add Firebase to your JavaScript project, 2023). Naredbu nećemo upotrijebiti s obzirom na to da se u ovom radu koristi Pyrebase biblioteka.

Kod autentifikacije uključit ćemo "Email/Password" kao pružatelj usluga za prijavu novih korisnika, a "Realtime Database" će biti korištena za pohranu podataka.

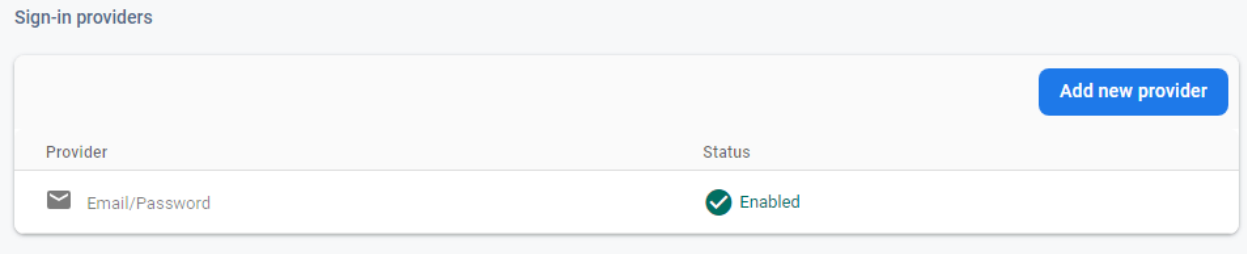

Figure 1 - Prijava putem email-a/lozinke omogućena u Firebase autentifikaciji

Radi lakšega rada s aplikacijom, sigurnosna pravila o čitanju i pisanju su postavljena na "true", što znači da svako tko ima pristup aplikaciji može čitati i pisati podatke (Firebase, Understand Firebase Realtime Database Security Rules, 2023). U idućoj cjelini će biti objašnjeno kako ćemo spriječiti upisivanje podatke u bazu podataka od strane običnog korisnika.

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Figure 2 - Pravila o čitanju i pisanju unutar baze podataka

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

Figure 3 - Pravila o čitanju i pisanju unutar spremišta

Za inicijalizaciju pohrane (storage) moramo otići u odjeljak "Storage" i kliknuti "Get started". Dane su nam dvije opcije sigurnosnih pravila (privatni ili javni podaci). Nakon klika na "Next" moramo odabrati lokaciju za pohranjivanje podataka, koje se kasnije ne može promijeniti (Firebase, Getting started with Firebase Storage on the web, 2022). Kao i kod "Realtime Database", čitanje i pisanje je dozvoljeno za svakoga tko ima pristup aplikaciji.

Kao što je već navedeno u uvodu, koristit ćemo Django kao okvir. Datoteke unutar Djanga koje će nama trebati su urls.py, views.py, forms.py, kao i mapa "templates" u kojoj ćemo spremati sve predloške. JavaScript funkcije spremljene su u "static" mapi.

Putanja direktorija "templates" se mora dodati u settings.py datoteku, u varijablu "DIRS" unutar "TEMPLATES" liste (Django, How to override templates, 2023).

U urls.py ćemo spremati sve URL-ove koji će se koristiti u ovome radu. Za dizajn URL-a za aplikaciju Django koristi modul "URLconf", to je preslikavanje između izraza URL putanje u Python funkcija, odnosno pogleda u views.py (Django, URL dispatcher, 2023).

Pogled u Djangu je Python funkcija koja preuzima web zahtjev i vraća web odgovor. Odgovor može biti web stranica, preusmjeravanje, pogreška, dokument, slika i druge stvari. Pogled sadrži logiku koja je neophodna za vraćanje tog odgovora. U Djangu se datoteka views.py koristi za pisanje pogleda (Django, Writing views, 2023). Ova datoteka će sadržavati konfiguracijski objekt koji dodijeljen prilikom registracije aplikacije.

Prilikom izrade obrasca u Djangu, programer mora dodati datoteku forms.py. U ovoj datoteci, imenovana klasa je mapa koja sadrži polja za taj obrazac, polja mogu biti različitih vrsta (int, float, boolean, slika, datoteka, datum, izbor, ...) . Nakon deklariranja obrasca u forms.py, on se mora pozvati u pogledu, a oznaka <form> mora se koristiti u HTML datoteci (Django, Working with forms, 2023; Django, Form fields, 2023).

2.1. STRUKTURA (STABLO) BAZE PODATAKA

Baza podataka je JSON oblika i sastoji se od 7 čvorova: "Knjige", "Izdavači", "Autori", "Korisnici", "Administratori", "Obrisani korisnici" i "Potvrda vraćanja knjige".

Čvor "Knjige" sadrži podatke o knjizi (ID, naslov, ...). Čvorovi "Izdavači" i "Autori" sadrže podatke o izdavaču i autori. Oni u sebi imaju ID brojeve koji su vanjski ključevi u čvoru "Knjige".

Čvorovi "Korisnici" i "Administratori" sadrže u sebi podatke o korisniku i administratoru (email, korisničko ime, ...). Autentificirane korisnike je trebalo odvojiti jer će administratori imati neke privilegije koje obični korisnici neće imati (na primjer, dodavanje novih podataka u bazu podataka).

"Obrisani korisnici" je čvor koji u sebi sadrži email adrese svih korisnika koje je administrator obrisao.

"Potvrda vraćanja knjige" je čvor koji sadrži privremene podatke, ID knjige i e-mail korisnika. Administrator mora odobriti povrat knjige nakon što korisnik klikne gumb "vrati knjigu" (u 3. poglavlju će ovaj postupak biti detaljnije objašnjen.).

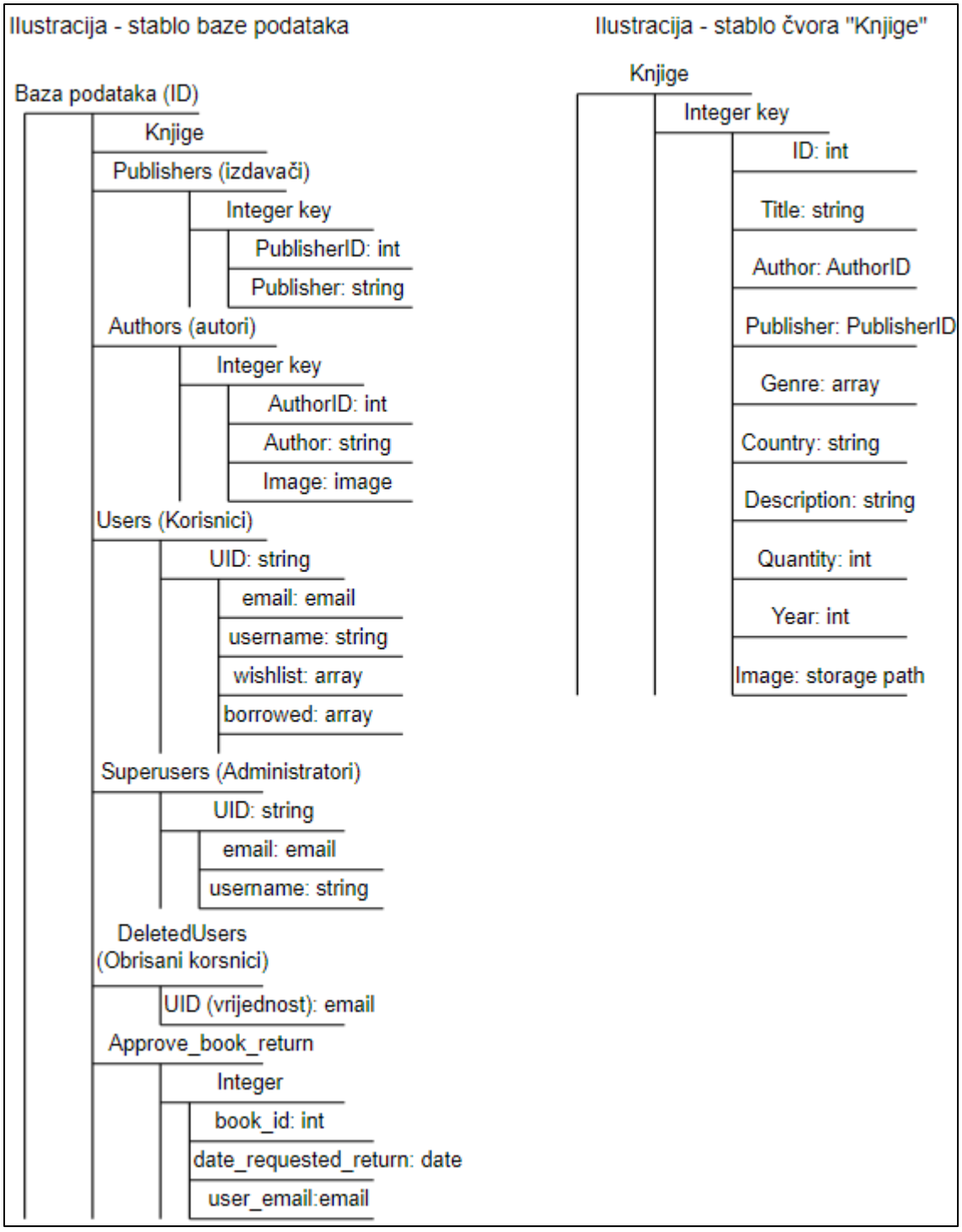

Figure 4 - stablo baze podataka i čvora "Knjige"

3. PROJEKT

3.1. BIBLIOTEKA PYREBASE

Budući da Firebase nije podržan u Djangu, moramo instalirati biblioteku "pyrebase" koristeći naredbu "pip install pyrebase" (Childs-Maidment, Pyrebase (Installation), 2020). Koristit ćemo i "firebase_admin" biblioteku kod jedne funkcije. Pyrebase je biblioteka otvorenog koda za Firebase-ov API, napisana je za python 3 (Childs-Maidment, Pyrebase (Pyrebase), 2020). Da bismo dodali biblioteku našoj aplikaciji, prvo moramo upisati konfiguracijski objekt, a zatim ga inicijalizirati (Childs-Maidment, Pyrebase (Add Pyrebase to your application), 2020). Cijeli postupak radimo u views.py datoteci.

Kako bi mogli koristiti autentifikaciju, bazu podataka i spremište (storage) moramo svako uslugu također inicijalizirati (Childs-Maidment, Pyrebase (Use Services), 2020).

```
import pyrebase
import firebase_admin
```

Figure 5 - Uvoz biblioteke pyrebase i firebase_admin

```
config={
  "apiKey": "AIzaSyAb3kFcqaffBQLvmPrAYJ3bEJ6aGj2mv1I",
  "authDomain": "zavrsnirad-d6160.firebaseapp.com",
  "databaseURL": "https://zavrsnirad-d6160-default-rtdb.firebaseio.com",
  "projectId": "zavrsnirad-d6160",
  "storageBucket": "zavrsnirad-d6160.appspot.com",
  "messagingSenderId": "805104473425",
  "appId": "1:805104473425:web:54db2c6cb5499e12fd2f70"
}

firebase=pyrebase.initialize_app(config)
authe=firebase.auth()
database=firebase.database()
storage = firebase.storage()
```

Figure 6 - Konfiguracija i inicijalizacija autentifikacije, baze podataka i spremišta

Na kraju, moramo i objasniti metode koje su korištene iz Pyrebase biblioteke u projektu.

Autentifikacija:

- sign_in_with_email_and_password() - vraća korisničke podatke, uključujući token,
- create_user_with_email_and_password() - stvaranje novog korisnika (Childs-Maidment, Pyrebase (Authentication), 2020).

Baza podataka:

- child() - izgrađivanje putanje do korisničkih podataka u bazi podataka,
- push() - spremanje podatka s jedinstvenim, automatski generiranim ključem koji se, temelji na vremenskoj oznaci
- set() - stvaranje vlastitog ključa,
- update() - ažuriranje podatka,
- remove() - brisanje podatka (Childs-Maidment, Pyrebase (Database), 2020).

Vraćanje podataka:

- val() - vraćanje podatka za objekt nad kojim je napravljen upit,
- key() - vraćanje ključa za podatke nad kojim je napravljen upit,
- each() - vraćanje popisa objekata za svaki koji se može pozvati metode val() i key(),
- get() - vraćanje podatka iz određene putanje,
- shallow() - vraćanje samo onih ključeva nad određenom putanjom (Childs-Maidment, Pyrebase (Retrieve data), 2020).

Spremište:

- child() - izgrađuje putanju do korisničkih podataka u spremištu,
- put() - uzimanje putanje do lokalne datoteke,
- get_url() - dohvaća putanju u kojoj je dokument spremljen i vraća URL od spremišta. (Childs-Maidment, Pyrebase (Storage), 2020).

3.2. POČETNA STRANICA I PRIKAZ INDIVIDUALNE KNJIGE

Početna stranica (nazvana "index.html") sadrži tablicu koja povlači podatke iz čvora "Knjige". Atributi koji se prikazuju u tablici su: "ID", "naslov", "autor", "izdavač", "žanr(ovi)", "država" i "godina".

ID i naslov su poveznice koje vode korisnika na zasebnu stranicu (nazvana "book_show") koja prikazuje sve podatke za tu knjigu.

ID	Title	Author	Publisher	Genre	Country	Year
0	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Pottermore Publishing	Fantasy , Magic , Young Adult	United Kingdom	1997
1	Harry Potter and the Chamber of Secrets	J.K. Rowling	Pottermore Publishing	Fantasy , Magic , Young Adult	United Kingdom	1998
2	Harry Potter and the Prisoner of Azkaban	J.K. Rowling	Pottermore Publishing	Fantasy , Magic , Young Adult	United Kingdom	1999
3	The Art of War	Sun Tzu	Chiron Academic Press	Military art , Philosophy	China	N/A
4	A Game of Thrones (A Song of Ice and Fire, Book 1)	George R. R. Martin	Bantam	Fantasy , Adult , Adventure	United States	1996
5	A Clash of Kings (A Song of Ice and Fire, Book 2)	George R. R. Martin	Bantam	Fantasy , Adult , Adventure	United States	1998

Figure 7 - Prikaz podataka na početnoj stranici

Klikom na poveznice "autor" ili "izdavač", korisnik je preusmjeren na zasebnu stranicu koja prikazuje sve knjige koje je napisao taj autor ili sve knjige izdavača tog izdavača. Stupac "žanr" sadrži jedan ili više žanrova za svaku knjigu. Svaki žanr sadrži svoj link. Kada korisnik klikne poveznicu, odvest će ga na zasebnu stranicu koja prikazuje sve knjige dodijeljene tom žanru.

Budući da prikazujemo sve knjige iz čvora, moramo proći kroz sve ključeve (cijeli broj dodijeljen svakoj knjizi) da dohvatimo podatke, koristeći for petlju. Prvo dohvaćamo sve ključeva iz čvora "Knjige" koristeći shallow() metodu (Childs-Maidment, Pyrebase (Retrieve data), 2020), zatim stvaramo praznu listu (nazvana "books" u funkciji), nakon toga prolazimo kroz sve ključeve i dodajemo ključeve u listu.
Lista "books" mora biti upisana u "context" varijablu.
Kontekst u Djangu je rječnik koji preslikava nazive varijabli u vrijednosti varijabli (Django, The Django template language: for Python programmers, 2023). Deklarirani predložak u ovoj funkciji je "index.html".

Predlošci se upisuju u render(), metoda kombinira dani predložak s danim rječnikom konteksta i vraća objekt "HttpResponse" s tim prikazanim tekstom (Django, Django shortcut functions, 2023). Ova metoda jer korištena u svim pogledima.

```
context = {"books": books, "username": userna
return render(request, 'index.html', context)
```

Figure 8 - Kontekstna varijabla i metoda render

Također, ID izdavača i autora napisani su u čvoru "Knjige", međutim želimo prikazati imena dva atributa u tablici. Atribute "PublisherID" i "AuthorID" zamjenjujemo s "Publisher" i "Author".

```
def index(request):
    # Get a list of keys in the Knjige node
    keys=database.child('Knjige').shallow().get().val()

    # Create an empty list to store the data
    books=[]

    # Loop through the keys and retrieve the data from each attribute in 'Knjige'
    for key in keys:
        book=database.child('Knjige').child(key).get().val()

        # Retrieve the publisher data from the "Publishers" node
        publisher_id = book.get('PublisherID')
        publisher = database.child('Publishers').child(publisher_id).child('Publisher').get().val()

        # Replace the "PublisherID" with the retrieved "Publisher" value
        book['Publisher'] = publisher

        # Retrieve the author data from the "Authors" node
        author_id = book.get('AuthorID')
        author = database.child('Authors').child(author_id).child('Author').get().val()

        # Replace the "AuthorsID" with the retrieved "Authors" value
        book['Author'] = author

        books.append(book)
```

Figure 9 - Pogled "index"

Na stranici "book_show.html" prikazani su podaci pojedinačne knjige.

Pošto dohvaćamo podatke od samo jednog objekta potreban nam je ID tog objekta. ID (nazvan "knjiga_id") upisujemo URL putanju na ovaj način: 'book_show/<knjiga_id>/' i upisujemo ga kao parametar funkcije: def book_show(request, knjiga_id). Varijabla "knjiga_id" je zapravo cijeli broj dodijeljen svakoj knjizi u bazi podataka. Varijabla i ID atribut su iste vrijednosti (dakle, knjiga_id=0 odgovara ID=0, knjiga_id=1 odgovara ID=1 i tako dalje) (Django, URL dispatcher, 2023).

U pogledu deklariramo varijablu koja vraća vrijednost za jednu knjigu.

```
book=database.child('Knjige').child(knjiga_id).get().val()
```

Figure 10 - Deklariranje varijabla za vraćanje vrijednosti jedne knjige

Također, i ovdje zamjenjujemo ID autora i izdavača s njihovim imenima.

```python
def book_show(request, knjiga_id):
    # Retrieve the data for the specified book
    book=database.child('Knjige').child(knjiga_id).get().val()

    # Retrieve the publisher ID from the book data
    publisher_id = book.get('PublisherID')

    # Retrieve the publisher data from the "Publishers" node
    publisher_data = database.child('Publishers').child(publisher_id).get().val()

    # Extract the publisher name from the fetched data
    publisher_name = publisher_data.get('Publisher')

    # Update the book dictionary with the publisher name
    book['Publisher'] = publisher_name

    author_id = book.get('AuthorID')
    author_data = database.child('Authors').child(author_id).get().val()
    author_name = author_data.get('Author')
    book['Author'] = author_name

    book['Genre'] = ', '.join(book['Genre'])

    # Retrieve the image path of the author from the "Authors" node
    image_path = author_data.get('Image')

    # Get the download URL for the author image from Firebase Storage
    image_url = storage.child(image_path).get_url(None)
```

Figure 11- Pogled "book_show" za prikaz individualne knjige

1984 (1949),
ID: 13

"1984" is a dystopian novel set in a totalitarian society ruled by the Party led by Big Brother. The story follows Winston Smith, a low-ranking member of the Party who begins to rebel

Author: George Orwell
Country: United Kingdom

Publisher: Signet

Genre: Dystopian, Political

Print length: 328, **Quantity:** 4

Update Delete

Back to homepage

Figure 12 - Prikaz individualne knjige

3.3. POVEZNICE ZA AUTORA/IZDAVAČA/ŽANR I TRAKA ZA PRETRAŽIVANJE

U prijašnjem poglavlju navedeno je da će poveznice za autora, izdavača i žanr odvesti korisnika na zasebne stranice koje prikazuju knjige od tog autora i izdavača, te žanr dodijeljen toj knjizi. Sada ćemo objasniti taj proces.

Deklarirane su tri URL putanje koje u svome nastavku sadrže parametre "author", "publisher" i "genre".

```
path('books_by_author/<author>/', views.books_by_author, name='books-by-author'),
path('books_by_publisher/<publisher>/', views.books_by_publisher, name='books-by-publisher'),
path('books_by_genre/<str:genre>/', views.books_by_genre, name='books-by-genre'),
```

Figure 13 - URL putanje

Nakon što korisnik klikne poveznicu, URL dohvaća vrijednost parametra, odvojenom kosom crtom "/". Uz to, parametri su upisani unutar znakova "<" i ">", inače bi se URL putanja tretirala kao doslovan niz (string) i ne bi uhvatio nijedan dinamički parametar. Dakle, pri izostavljanju znakova "<" i ">", definirani pogled bi uvijek primao niz "author", "publisher" ili "genre" kao vrijednost tog parametra umjesto stvarne vrijednosti navedene u URL-u (Django, URL dispatcher (How Django processes a request), 2023; Django, URL dispatcher (Example), 2023).

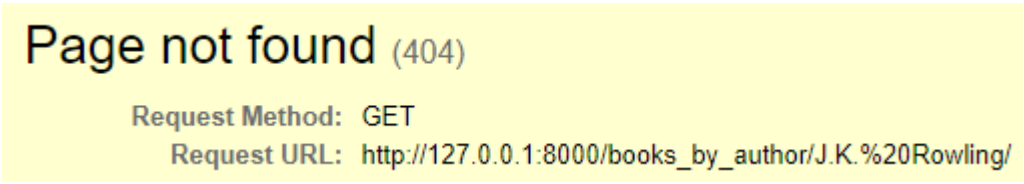

Figure 14 - Greška pri pogrešnom upisu parametra

Funkcija pogleda prima zahtjev i parametar. Kao i u index-u dohvaćamo ključeve knjiga, stvaramo praznu listu i dohvaćamo podatke koristeći for petlju (pritom i ovdje moramo zamijeniti ID autora i izdavača njihovim imenima).

Na kraju, spremamo parametar i listu unutar kontekstne varijable, te pozivamo metodu render() koja u sebi ima naziv nove stranice koja prikazuje podatke po zadanome parametru te "context" varijablu. Parametar spremamo u kontekstnu varijablu jer ćemo prikazati njenu vrijednost u novoj stranici (na primjer, kod prikaza knjiga po autoru, prikazujemo autorovo ime).

U "index.html", unutar "href", parametar je izjednačen s atributom unutar baze podataka. Na primjer, parametar "author" je izjednačen s vrijednosti koja je upisana unutar čvora "Autori" (podsjetnik, ID od autora je zamijenjen s imenom autora koji je string). Isti postupak radimo kod izdavača i žanra.

```
<td><a href="{% url 'main:books-by-publisher' publisher=book.Publisher %}" style="color: inherit;">{{ book.Publisher }}</a></td>
```

Figure 15 - Izjednačavanje parametra "publisher" s vrijednosti upisanom unutar čvora

U navigacijskoj traci nalazi se traka za pretraživanje. Korisniku su ponuđene tri opcije, pretraživanje po naslovu knjige, autoru ili izdavaču. Unosom pojma u polje i klikom na gumb "Search" korisnik je preusmjeren na novu stranicu koja prikazuje filtrirane podatke koje sadrže unesen pojam u svom imenu.

U pogledu i predlošku moramo koristiti GET metodu. GET je HTTP zahtjev koji se koristi za zahtjev podataka iz određenog izvora (W3Schools, 2023).

U pogledu su deklarirane dvije varijable "search_term" i "search_type". "search_type" referenca je na izbornik u traci i može se odnositi na 3 navedene opcije. Varijabla "search_term" referenca je na korisnički unos koju moramo dodijeliti kontekstnoj varijabli i moramo ju navesti u predlošku unutar <input> pod pojmom "name".

Radi lakše pretrage, URL pojma pretraživanja je modificiran tako da mu je u produžetku dodana varijabla pretrage ("search_term") koristeći metodu reverse().
Sposobnost metode je spremanje URL vrijednosti (Django, django.urls utility functions, 2023).

```
search_term = request.GET['search_term']
search_type = request.GET['search_type']
```

Figure 17 - Deklariranje varijabli "search_term" i "search_type"

```
if search_type == 'title':
    url = reverse('main:book-search') + '?title=' + search_term
    context = {'books': books, 'search_term': search_term, 'uid': uid, 'url': url, 'is_superuser': is_superuser}
    return render(request, 'search/book_search.html', context)
elif search_type == 'author':
    url = reverse('main:book-author-search') + '?author=' + search_term
    context = {'books': books, 'search_term': search_term, 'uid': uid, 'url': url, 'is_superuser': is_superuser}
    return render(request, 'search/book_author_search.html', context)
elif search_type == 'publisher':
    url = reverse('main:book-publisher-search') + '?publisher=' + search_term
    context = {'books': books, 'search_term': search_term, 'uid': uid, 'url': url, 'is_superuser': is_superuser}
    return render(request, 'search/book_publisher_search.html', context)
```

Figure 16 - Tri opcije za izbornik i modifikacija URL-a

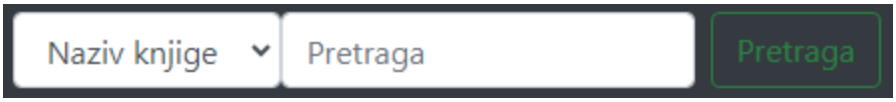

Figure 19 - Traka za pretraživanje

J.K. Rowling's books

ID	Title	Author	Publisher	Genre	Country	Year
0	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Pottermore Publishing	Fantasy, Magic, Young Adult	United Kingdom	1997
1	Harry Potter and the Chamber of Secrets	J.K. Rowling	Pottermore Publishing	Fantasy, Magic, Young Adult	United Kingdom	1998
2	Harry Potter and the Prisoner of Azkaban	J.K. Rowling	Pottermore Publishing	Fantasy, Magic, Young Adult	United Kingdom	1999

Figure 18 - Iispis svih knjiga po određenom autoru

Rezultat pretraživanja za pojam: "stone"

ID	Title	Author	Publisher	Genre	Country	Year
0	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Pottermore Publishing	Fantasy, Magic, Young Adult	United Kingdom	1997

Figure 20 - Razultat pretraživanja za određeni pojam

3.4. AUTENTIFIKACIJA

Kako bi mogli razlikovati običnog korisnika i administratora morat ćemo spremati podatke za različite korisnike u dva zasebna čvora, "Users" i "Superusers".

Napravljena su četiri pogleda: registracija korisnika, registracija admina, login i logout.

Ovdje, i svaki idući put kada budemo slali podatke, koristit ćemo POST metodu. POST metoda se koristi za slanje podataka na server za stvaranje ili ažuriranje podataka, dakle suprotno GET metode (W3Schools, 2023).

Kod registracije spremamo korisničko ime, email i lozinku. Čvor "Users" će imati i dvije prazne varijable "wishlist" i "borrowed", u njih ćemo spremati podatke o knjigama dodanih u popis želja i posuđenim knjigama.

Pri stvaranju novog korisnika radimo provjeru koristeći "try/except" naredbe. "Try/except" funkcionira na ovaj način: najprije se izvršava "try", ako nema izuzetaka "except" je preskočen. Ako se tijekom izvođenja "try" dogodi iznimka, ostatak klauzule se preskače. Zatim, ako njegov tip odgovara iznimci unutar "except", "except" se izvršava, a zatim se izvršavanje koda nastavlja nakon bloka "try/except". Naredba "try" može imati više od jedne iznimke (Python, 2023).

Prilikom registracije, ako korisnik pokuša unijeti postojeću email adresu ispisuje se poruka "Email adresa već postoji." Provjeru radimo tako da uspoređujemo unesen email sa svim email adresama unutar čvora "Users"/"Superusers". Metoda get() je korištena za dohvat putanje u čvoru, a za iteraciju svakog korisnika u čvoru koristimo each(), unutar for petlje vrijednost unesenog email-a označavamo s val() (Childs-Maidment, Pyrebase (Retrieve data), 2020).

```
try:
    # Create the user using Firebase authentication
    user = authe.create_user_with_email_and_password(email, password)
    uid=user['localId']
except:
    # Check if the email already exists in the Users node
    users = database.child("Users").get()
    for user in users.each():
        if user.val()["email"] == email:
            messages.error(request, "Email adresa već postoji.")
            return render(request, 'user/register.html')
```

Figure 21 - Registracija novog korisnika i provjera email-a

Ako nema greške stvaramo korisnika metodom "create_user_with_email_and_password ()" (Childs-Maidment, Pyrebase (Authentication), 2020). Stvara se jedinstveni korisnički ID (nazvan UID) unutar kojeg su spremljeni korisnički podaci. U kontekstnoj varijabli "data" spremamo sve varijable (osim lozinke).

```
data = {"username": username, "email": email, "wishlist": ["Empty"], "borrowed": ["Empty"]}
database.child("Users").child(uid).set(data)
```

Figure 22 - Spremanja podataka u kontekstnu varijablu

Pogled za stvaranje administratora identičan je kao i pogled za stvaranje običnog korisnika, jedina razlika je da nećemo imati varijable "wishlist" i "borrowed", a podaci će se spremati u "Superusers" čvor.

Nakon uspješne registracije, korisnik ili administrator je preusmjeren na login stranicu. Unosi se email adresu i lozinka nakon čega se poziva metoda "sign_in_with_email_and_password()" koju spremamo u varijablu "user" (Childs-Maidment, Pyrebase (Authentication), 2020). Koristimo "try/except" naredbe za provjeru unesene lozinke i email-a. Ako je lozinka ili email pogrešna ispisuje se poruka o greški. Uz to, ako je unesena email adresa od korisnika koji je izbrisan iz baze podataka, također se ispisuje greška. Nakon uspješnog login-a korisnik je preusmjeren na početnu stranicu.

Prilikom login-a moramo stvoriti login sesiju. Login sesija je razdoblje aktivnosti između korisnikove prijave i odjave iz sustava (Wikipedia, Login session, 2023). Pozivamo Djangov "request.session" čiji je ključ definirani UID (Django, How to use sessions, 2023). Vrijednost sesije spremamo u varijablu "user".

```
try:
    # Authenticate the user using Firebase authentication
    user = authe.sign_in_with_email_and_password(email, password)
except:
    messages.error(request, 'Netočna email adresa ili loznika. Molimo pokušajte ponovno.')
    return render(request, 'user/login.html')

# Save the user ID as a session variable
request.session['uid'] = user['localId']

# Redirect to the index page after successful login
return redirect('main:index')
```

Figure 23 - Login i stvaranje sesije

Ako se korisnik želi odjaviti poziva se pogled "logout" koji briše login sesiju i preusmjerava korisnika na početnu stranicu.

```python
def logout(request):
    # Remove the user ID from the session
    del request.session['uid']

    # Redirect to the login page after successful logout
    return redirect('main:index')
```

Figure 24 – Metoda provjere da li je korisnik obrisan

```python
# Check if the user's email exists in the "DeletedUsers" node
deleted_users = database.child('DeletedUsers').get()
for user in deleted_users.each():
    if user.val() == email:
        messages.error(request, 'Ovaj korisnik je obrisan.')
        return render(request, 'user/login.html')
```

Figure 25 - Pogled logout

3.5. OBRASCI ZA UNOS, AŽURIRANJE I BRISANJE PODATAKA

Administrator ima mogućnost unosa novih podataka, kao i ažuriranje i brisanje podataka.

Kako bi mogli raditi s obrascima moramo najprije stvoriti datoteku "forms.py". U njoj imenujemo klase. Klase sadrže polja različitih vrsta.
U HTML datoteci obrasce označavamo s <form>. U pogledima i predlošcima koristimo isključivo POST metodu jer šaljemo podatke u našu bazu (Django, Working with forms, 2023).

Klasa "BookForm" predstavlja obrazac unosa nove knjige. Svi atributi osim ID-a knjige unutar čvora "Knjige" upisani su kao polja u našoj klasi. ID knjige se automatski generira i unosi (počevši od 0 pa nadalje).

```
class BookForm(forms.Form):
    Title = forms.CharField(max_length=70, widget=forms.TextInput(attrs={'placeholder':'Book title'}))
    Author = forms.ChoiceField(choices=[], label='Author')
    Publisher = forms.ChoiceField(choices=[], label='Publisher')
    Genre = forms.CharField(max_length=100, widget=forms.TextInput(attrs={
        'placeholder': 'Zanr 1, Zanr 2, Zanr 3, ...', 'style': 'width: 200px'}), )
    Country = forms.CharField(max_length=30, required=False, widget=forms.TextInput(attrs={'placeholder':'Country'}))
    Print_length = forms.IntegerField(min_value=0)
    Year = forms.IntegerField(min_value=0, required=False, widget=forms.NumberInput(attrs={'placeholder': 'Year'}))
    Quantity = forms.IntegerField(min_value=0, widget=forms.NumberInput(attrs={'placeholder': 'Quantity'}))
    Description = forms.CharField(required=False, widget=forms.Textarea(attrs={'style': 'height: 50px'}))
```

Figure 26 - Klasa "BookForm"

Polje ChoiceField() polje je tipa string koje koristimo za odabir ponuđenih izbora s popisa (Arora, 2020). Koristimo ga za dodjelu autora i izdavača knjizi. Unos autora i izdavača radimo u posebnim obrascima. Deklarirane klase za ova dva čvora su "AuthorForm" i "PublisherForm".

```
class PublisherForm(forms.Form):
    Publisher = forms.CharField(max_length=50)

class AuthorForm(forms.Form):
    Author = forms.CharField(max_length=70)
    Image = forms.ImageField(required=False)
```

Figure 27 - Klase "PublisherForm" i "AuthorForm"

```
from .forms import *
```

Figure 28 - Uvoz stvorenih forma u views.py.

U stvorenim pogledima moramo dodati ograničenje tako da samo administrator ima pravo upravljati podacima. Ovo ograničenje se koristiti mnogo puta u projektu (administratorska stranica, gumbovi za ažuriranje i brisanje knjige, i tako dalje).

```
uid = request.session.get('uid')
is_superuser = database.child('Superusers').child(uid).get().val()
if not is_superuser:
    return HttpResponse(status=403)
```

Figure 29 - Ograničenje

Definirana polja u formi upisujemo unutar varijable "data". Pozivamo metodu "form.is_valid " koja provjerava jesu li podaci koje je korisnik unio važeći (Django, The Forms API, 2023). Svim poljima koja su izborna dodjeljujemo NULL vrijednost, Firebase ne dozvoljava unos praznih objekata ili lista (Richardson, 2013).

```
publishers = database.child('Publishers').get().val()
authors = database.child('Authors').get().val()

if request.method == 'POST':
    form = BookForm(request.POST, publishers=publishers, authors=authors)
    if form.is_valid():
        data = {
            'Title': form.cleaned_data['Title'],
            'AuthorID': form.cleaned_data['Author'],
            'PublisherID': form.cleaned_data['Publisher'],
            'Genre': form.cleaned_data['Genre'].split(', '),
            'Country': form.cleaned_data['Country'],
            'Print_length': form.cleaned_data['Print_length'],
            'Year': form.cleaned_data['Year'],
            'Quantity': form.cleaned_data['Quantity'],
            'Description': form.cleaned_data['Description'],
        }

        # Assign "NULL" to fields that are not required
        if data['Country'] == '':
            data['Country'] = 'NULL'
        if data['Year'] is None:
            data['Year'] = 'NULL'
        if data['Description'] == '':
            data['Description'] = 'NULL'
```

Figure 30 - Funkcija za izradu obrasca za unos nove knjige

Novi ID knjige stvaramo prebrojavanjem svih postojećih ID-eva, te novom ID-u dodamo izračunatu dužinu i povećamo ga za jedan. Unutar čvora "Knjige" najprije spremamo cijeli broj (koji je isti kao i ID knjige) unutar kojega spremamo podatke. Postupak je isti kod unosa autora i izdavača.

```
books_count = database.child('Knjige').get().each()
next_id = len(books_count) if books_count else 0
data['ID'] = next_id

database.child('Knjige').child(str(next_id)).set(data)
```

Figure 31 - Generiranje i upis novog ID-a

U pogledu za unosa autora moramo deklarirati putanju u kojoj ćemo spremiti sliku. Sliku spremamo u spremište koristeći metodu put() (Childs-Maidment, Pyrebase (Storage), 2020). Putanja se sprema u varijablu "Image" koja se sprema u čvor "Authors.

```
image = request.FILES['Image']

image_path = f"authors/{image.name}" #Get image path

# Upload image to storage
try:
    storage.child(image_path).put(image)
    messages.success(request, "Upload successful")
except:
    messages.error(request, "Upload unsuccessful")
    return redirect(reverse('upload-author'))

# Update author data with image path
author_data['Image'] = image_path

# Store author data in the database
database.child('Authors').child(str(next_id)).set(author_data)
```

Figure 32 - Spremanje slike u spremište

Kod brisanja objekta iz baze podataka dovoljno je definirati ID tog objekta u formi i unijeti ga u obrazac. Moramo paziti da pri brisanju autora i izdavača ne obrišemo samo polje unutar knjige, u slučaju da je obrisani autora/izdavača dodijeljen nekoj knjizi obrisanu vrijednost zamjenjujemo s NULL vrijednosti. Pri brisanju objekta koristimo metodu remove() (Childs-Maidment, Pyrebase (Database), 2020).

```
class DeleteAuthor(forms.Form):
    AuthorID = forms.IntegerField(min_value=0, label='Author ID')
```

Figure 33 - Klasa "DeleteAuthor"

```
# Check if the publisher exists
publisher_exists = database.child('Publishers').child(str(publisher_id)).get().val()

if publisher_exists:
    # Delete the publisher
    database.child('Publishers').child(str(publisher_id)).remove()

    # Update the books assigned to the deleted publisher
    books = database.child('Knjige').get().each()  # Retrieve books as a list

    for book in books:
        book_key = book.key()
        book_value = book.val()

        if book_value.get('PublisherID') == publisher_id:
            # Update the PublisherID value to 2 for books assigned to the deleted publisher
            database.child('Knjige').child(book_key).update({'PublisherID': 2})
```

Figure 34 - Ažuriranje ID-a izdavača ako je knjizi dodjeljen taj izdavač

Kod ažuriranja podataka o autoru i izdavača novi podatak zamjenjujemo starim tako da koristimo novu varijablu koja će predstavljati novi unos, cijeli postupak je isti kao i kod unosa podataka. Ovdje moramo koristiti metodu update() (Childs-Maidment, Pyrebase (Database), 2020). Administrator ažurira knjige tako da klikne na gumb "Update" koji ga preusmjerava na novu stranicu s obrascem koji sadržava sve podatke o toj knjizi.

```
if form.is_valid():
    publisher_id = form.cleaned_data['Publisher']
    new_publisher_name = form.cleaned_data['NewPublisherName']

    # Update the publisher data
    publisher_data = {
        'Publisher': new_publisher_name,
    }
    database.child('Publishers').child(str(publisher_id)).update(publisher_data)
```

Figure 35 - Ažuriranje izdavača

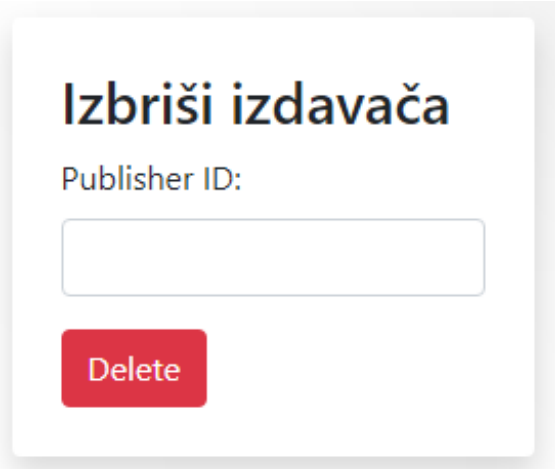

Figure 36 - Brisanje izdavača (obrazac)

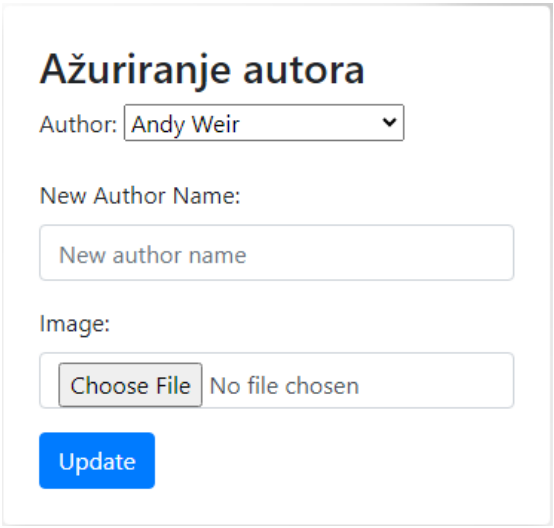

Figure 37 - Ažuriranje autora (obrazac)

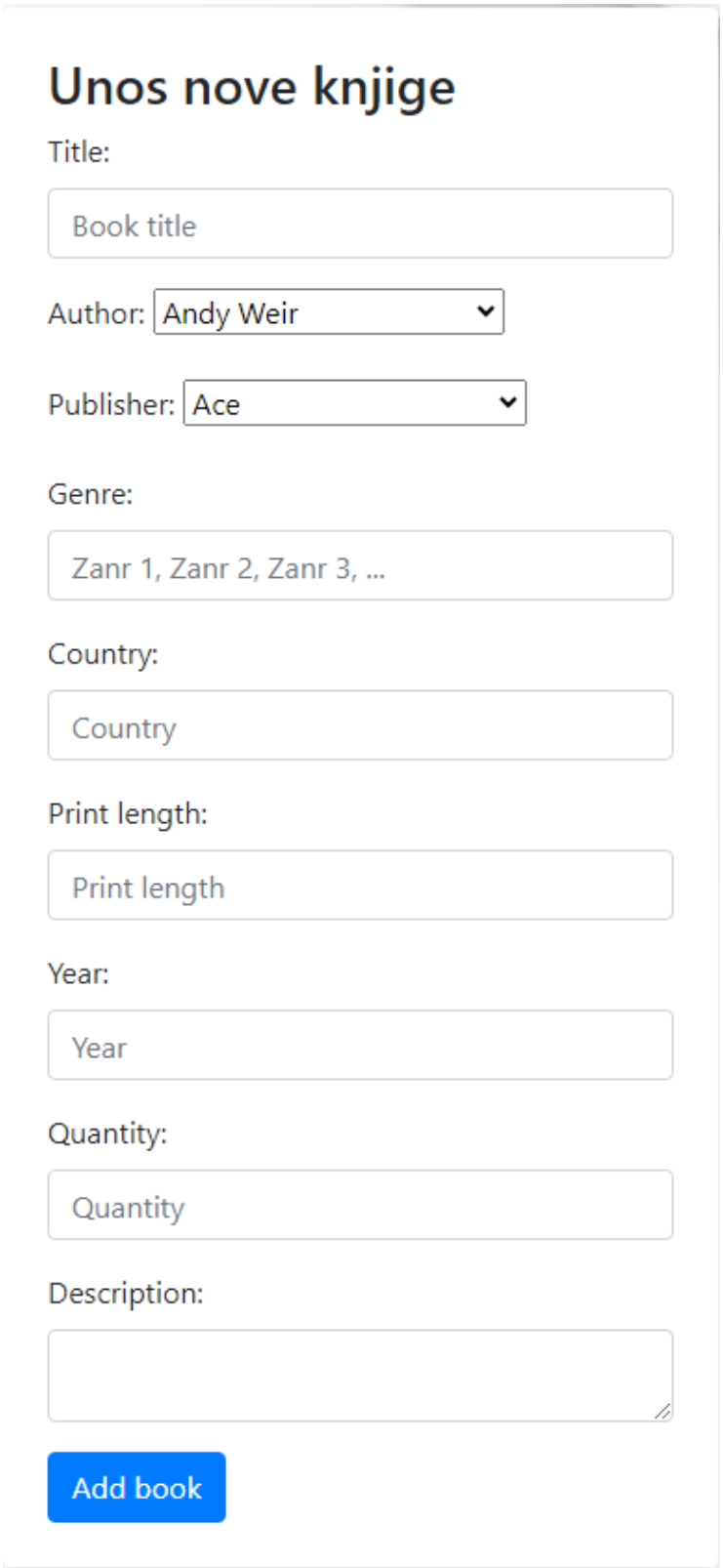

Figure 38 - Unos nove knjige (obrazac)

3.6. WISHLIST, POSUDBA KNJIGE I KORISNIČKA STRANICA

Na stranici za prikaz individualne knjige (book_show.html) korisnik može posuditi knjigu ili je dodati na popis želja.

Pogled "add_to_wishlist" dodaje ID knjige u listu "wishlist" koji se nalazi u korisničkom u čvor ("Users"). Prije dodavanja ID-a u moramo provjeriti da li je lista prazna, objekt "Empty" predstavlja prazan popis. Nakon dodavanja knjige uklanjamo objekt "Empty". Moramo i provjeriti ako knjiga već postoji u popisu, odnosno podudara li se ID kojeg korisnik pokušava dodati s ID-em na popisu.

Metoda set() korištena je kod stvaranja ključa pri dodavanju ID-a u listu (Childs-Maidment, Pyrebase (Database), 2020). Metode get() i val() su korištene pri vraćanju trenutne liste i pri vraćanju ID-a određene knjige (Childs-Maidment, Pyrebase (Retrieve data), 2020).

```python
def add_to_wishlist(request, knjiga_id):
    uid = request.session.get('uid')

    # Get the current wishlist for the user
    wishlist = database.child('Users').child(uid).child('wishlist').get().val()

    # If the wishlist is empty, remove the "Empty" item
    if wishlist == ['Empty']:
        database.child('Users').child(uid).child('wishlist').remove()
        wishlist = []

    # Check if the book is already in the wishlist
    if knjiga_id in wishlist:
        # Book already in wishlist, so don't add it again
        return redirect('main:user')

    # Retrieve the data for the specified book
    book = database.child('Knjige').child(knjiga_id).get().val()

    # Add the book to the wishlist
    wishlist.append(knjiga_id)
    database.child('Users').child(uid).child('wishlist').set(wishlist)

    # Redirect to the user page
    return redirect('main:user')
```

Figure 39 - Pogled "add_to_wishlist"

Metoda za posudbu knjige radi po istome principu. Nakon posudbe knjige moramo smanjiti količinu (atribut "Quantity") za jedan.

Ako želimo ukloniti knjigu s popisa želja pozivamo pogled "remove_from_wishlist". Najprije vraćamo popis kao listu (varijabla "wishlist"), nakon toga vraćamo ID knjige koristeći metodu POST i spremamo u varijablu "book_id". If petljom provjeravamo ako se ID nalazi na popisu. Metoda remove() briše ID iz liste (Childs-Maidment, Pyrebase (Database), 2020). Ako je lista opet prazna vraćamo privremeni objekt "Empty".

```
def remove_from_wishlist(request):
    if request.method == 'POST':
        uid = request.session.get('uid')

        # Retrieve the user data from the Firebase's database
        user_data = database.child('Users').child(uid).get().val()

        # Retrieve the wishlist of the user
        wishlist = user_data.get('wishlist', [])

        # Remove the book ID from the wishlist
        book_id = request.POST.get('book_id')
        if book_id in wishlist:
            wishlist.remove(book_id)

        # Check if the wishlist is now empty
        if not wishlist:
            # Add the "Empty" item back to the borrowed list
            wishlist.append('Empty')

        # Update the wishlist in the Firebase's database
        database.child('Users').child(uid).child('wishlist').set(wishlist)

        # Redirect to the user page
        return redirect('main:user')
```

Figure 40 - Pogled "remove_from_wishlist"

Na korisničkoj stranici (user_page.html) nalaze se popisi posuđenih knjiga i knjige dodane na popis želja. U pogledu najprije moramo stvoriti varijable koje vraćaju dvije liste iz čvora. Imamo dvije for petlje, u prvoj vraćamo podatke o knjigama u popisu želja, u drugoj vraćamo podatke o posuđenim knjigama. Varijable spremamo u kontekstnu varijablu.

```
# Retrieve the data for the books in the wishlist
books = []
for book_id in wishlist:
    book_data = database.child('Knjige').child(book_id).get().val()
    books.append(book_data)

# Retrieve the data for the books in the borrowed list
books_2 = []
for book_id in borrowed_list:
    book_data = database.child('Knjige').child(book_id).get().val()
    books_2.append(book_data)
```

Figure 41 - Dvije for petlje za vraćanje podataka iz lista

```
# Retrieve the user data from the Firebase's database
user_data = database.child('Users').child(uid).get().val()

# Retrieve the wishlist of the user
wishlist = user_data.get('wishlist', [])

# Retrieve the borrowed book list of the user
borrowed_list = user_data.get('borrowed', [])
```

Figure 43 - Deklariranje lista za vraćanje podataka

Proces prikaza podatka u predlošku je isti kao i kod prikaza knjiga na početnoj stranici i na stranici za prikaz individualne knjige. For petlje iteriramo po dvjema listama koje smo koristili za spremanje podataka (liste "books" i "books_2").

```
{% for book in books %}
  <tr>
    <td><a href="{% url 'main:book-show' book.ID %}" style="color: inherit;">{{ book.ID }}</a></td>
    <td><a href="{% url 'main:book-show' book.ID %}" style="color: inherit;">{{ book.Title }}</a></td>
```

Figure 42 - For petlja u predlošku

Proces vraćanja posuđene knjige bit će detaljnije objašnjen u idućem poglavlju.

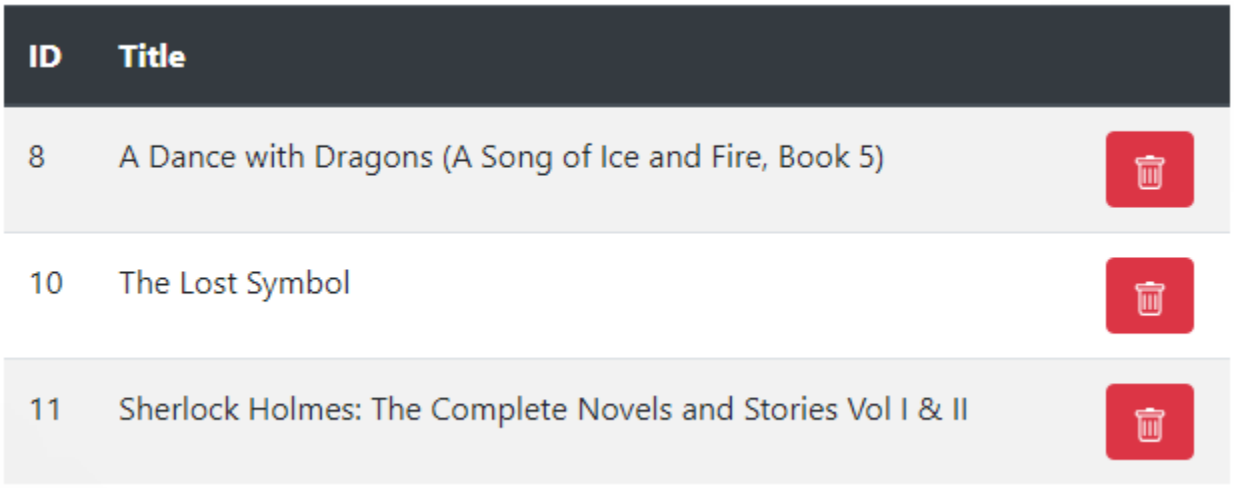

Figure 44 - Popis knjiga na listi želja na korisničkoj stranici

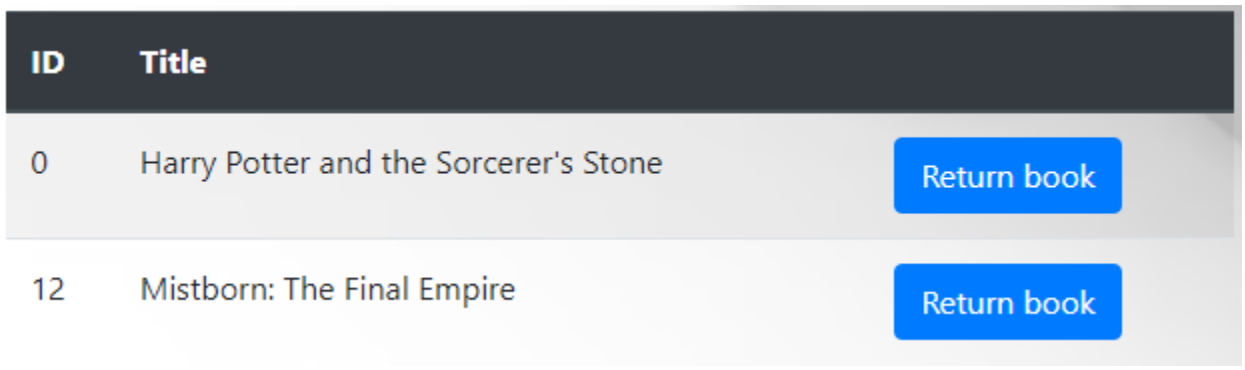

Figure 45 - Popis posuđenih knjiga na korisničkoj stranici

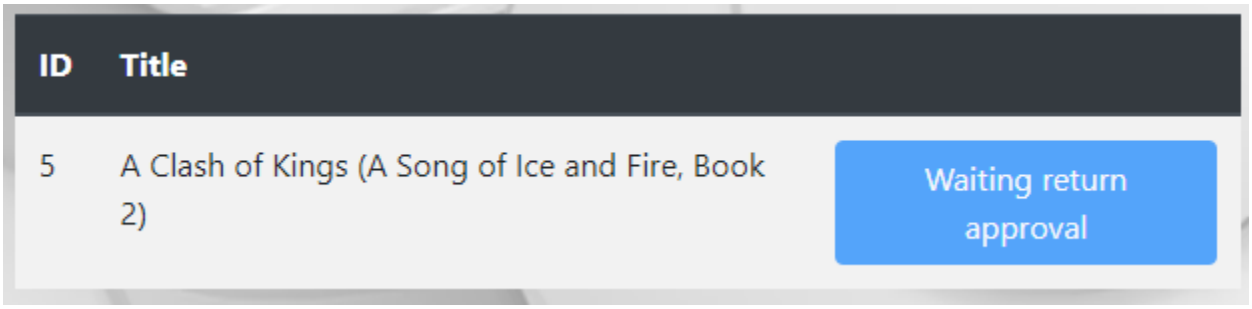

Figure 46 - Posuđene knjige koje su na popisu za čekanje odobrenja od administratora na korisničkoj stranici

3.7. PROCES VRAĆANJA KNJIGE

Korisnik vraća posuđenu knjigu klikom gumb "Return book", međutim ID knjige se ne briše odmah s popisa posuđenih knjiga. Administrator mora odobriti korisnikov zahtjev za povrat knjige.

ID knjige i korisnička email adresa upisuju se u zaseban čvor (koji se zove "potvrda vraćanja knjige"). Popis svih zahtjeva za povrat knjiga prikazan je na administratorskoj stranici. Email se mora upisati u čvor jer moramo znati koji korisnik želi vratiti knjigu (inače bi imali dva identična ID-a knjige, a da ne znamo kojem korisniku pripada).

Nakon što korisnik klikne gumb, poziva se pogled "return_book". Kao i u pogledu "remove_from_wishlist", moramo provjeriti ako je lista sada prazna, ako je vraćamo privremeni objekt "Empty". Atribut "Količina" povećavamo za jedan, te upisujemo ID knjige i email adresu u čvor.

Ovdje koristimo novu metodu, push(), kojom spremamo podatke s jedinstvenim, automatski generiranim ključem (Childs-Maidment, Pyrebase (Database), 2020). Ključ će biti jedinstveni string u kojega ćemo spremiti ID knjige i korisnikovu email adresu.

```
# Retrieve the user data from the Firebase's database
user_data = database.child('Users').child(uid).get().val()

# Retrieve the wishlist of the user
borrowed_books = user_data.get('borrowed', [])

# Remove the book ID from the wishlist
book_id = request.POST.get('book_id')
if book_id in borrowed_books:
    borrowed_books.remove(book_id)

    book = database.child('Knjige').child(book_id).get().val()
    quantity = book.get('Quantity')
    quantity += 1
    database.child('Knjige').child(book_id).child('Quantity').set(quantity)

# Check if the borrowed list is now empty
if not borrowed_books:
    # Add the "Empty" item back to the borrowed list
    borrowed_books.append('Empty')

# Update the borrowed list in the Firebase's database
database.child('Users').child(uid).child('borrowed').set(borrowed_books)

# Add the book ID and user email to the "Approve_book_return" node
approve_data = {'book_id': book_id, 'user_email': user_data['email'],
                'date_requested_return': datetime.datetime.now().strftime('%d-%m-%Y')}
database.child('Approve_book_return').push(approve_data)
```

Figure 47 - Pogled "return_book"

3.8. ADMINISTRATORSKA STRANICA I POSTAVKE

Administrator ima vlastitu stranicu, na njoj su prikazani svi korisnici i zahtjevi za potvrdu vraćanja knjige.

Administrator može pretraživati korisnika po email adresi i može obrisati korisnika.

Gumb "Delete user" poziva pogled "delete_user_account" koji briše korisnika iz baze podataka, ovdje nam je dovoljan korisnikov mail kao identifikator. Treba nam i parametar UID u kojem je spremljena email adresa. Prije brisanja korisnika, email spremamo u posebni čvor, "DeletedUsers", koristimo push() metodu za stvaranje jednistvenog ključa kojem će biti dodijeljen email (Childs-Maidment, Pyrebase (Database), 2020). Nakon toga brišemo korisnika iz baze podataka.

Figure 49 - Čvor "DeletedUsers"

```
def delete_user_account(request, uid):
    if request.method == 'POST':
        # Get the email of the user to be deleted
        user_email = database.child('Users').child(uid).child('email').get().val()

        # Store the email in the 'DeletedUsers' node
        database.child('DeletedUsers').push(user_email)

        # Delete the user from the 'Users' node
        database.child('Users').child(uid).remove()
```

Figure 48 - Pogled "delete_user_account"

Pokraj popisa svih korisnika nalazi se tablica koja sadrži sve zahtjeve o potvrdi vraćanja knjige. Svaki red sadrži ID i naslov knjige, korisnički email, te datum kada je zahtjev napravljen.

Gumb "Approve book return" poziva pogled "return_book" koji uklanja podatke o posudbi iz čvora "Potvrda vraćanja knjige". Ovdje kao parametar moramo uzeti ključ u kojem smo spremili podatke.

```
def approve_book_return(request, random_string):
    database.child('Approve_book_return').child(random_string).remove()
    return redirect('main:admin-page')
```

Figure 50 - Pogled "approve_book_return"

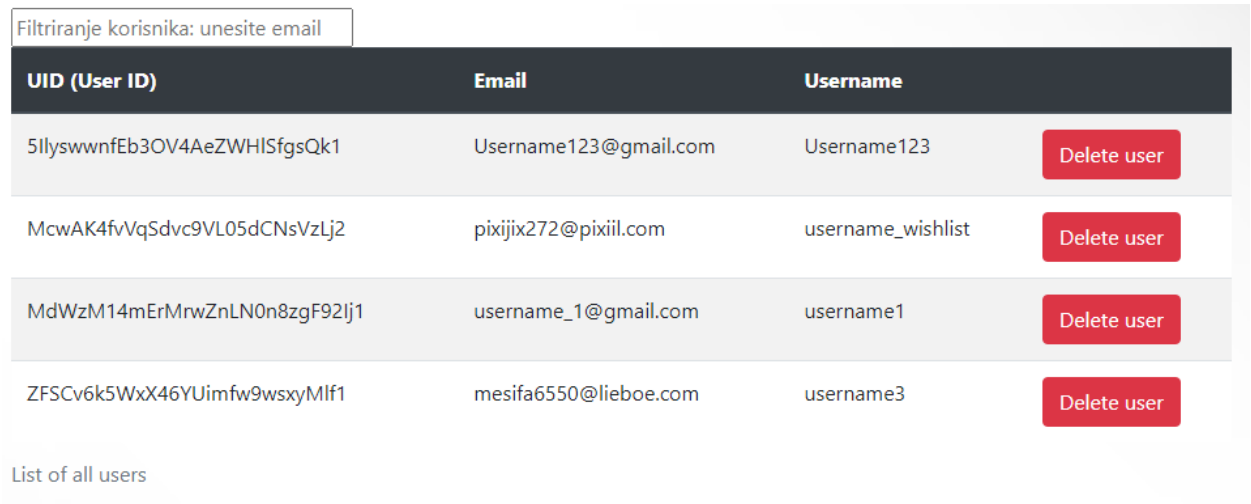

Figure 51 - popis svih korisnika

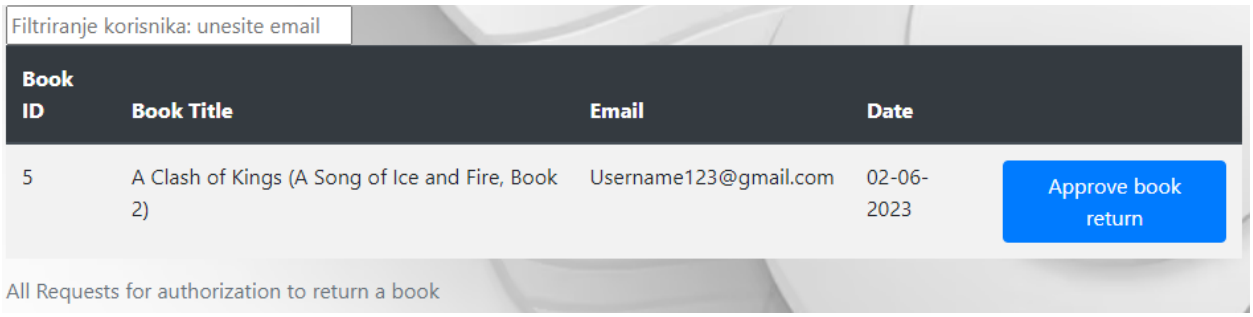

Figure 52 - popis svih zahtjeva za autorizacije vraćanja knjige

U projektu su napravljene i postavke u kojima korisnici i administratori mogu promijeniti svoje korisničko ime, email ili lozinku. Na glavnoj stranici "settings_main_page.html" najprije moramo provjeriti da li se radi o korisniku ili administratoru, nakon toga imamo tri poveznice koje vode na posebne stranice za navedene promjene.

```
def settings_main_page(request):
    uid = request.session.get('uid')

    is_superuser = database.child('Superusers').child(uid).get().val()

    if is_superuser:
        user_data = database.child("Superusers").child(uid).get().val()
    else:
        user_data = database.child("Users").child(uid).get().val()
```

Figure 53 - Provjera u postavkama

Kod promjene korisničkog imena i emaila deklariramo nove varijable, "new_username" i "new_email" koje korisnik unosi i upisujemo ih u bazu podataka. Kao i na glavnoj stranici, najprije provjeravamo da li je riječ o korisniku ili administratoru, dohvaćamo email, nakon toga koristimo metodu sign_in_with_email_and_password() za ponovnu autentifikaciju korisnika, te ažuriramo ime/email.
Novu varijablu spremamo u korisnički/administratorski čvor koristeći metodu update() (Childs-Maidment, Pyrebase (Database), 2020).

```
# Update the email in the Firebase's database
database.child(user_type).child(uid).update({"email": new_email})
```

Figure 54 - Ažuriranje email-a

Radnje koje su osjetljive na sigurnost, kao što je promjena korisničkih podataka, zahtijevaju ponovnu autentifikaciju korisnika. Time osiguravamo da je osoba koja je izvršila promjenu ista kao osoba koja se prvotno prijavila (Firebase, Manage Users in Firebase, 2023).

```
# Re-authenticate the user using Firebase authentication
user = authe.sign_in_with_email_and_password(email, password)
```

Figure 55 - Ponovna autentifikaciju korisnika

Promjena lozinke zahtjeva od korisnika unos stare i nove lozinke. Ostatak postupka je isti kao i u promjeni korisničkog imena/email-a.

```python
def change_password(request):
    if request.method == 'POST':
        current_password = request.POST.get('current_password')
        new_password = request.POST.get('new_password')

        # Get the user's UID from the session variable
        uid = request.session.get('uid')

        try:
            is_superuser = database.child('Superusers').child(uid).get().val()
            if is_superuser:
                user_data = database.child("Superusers").child(uid).get().val()
            else:
                user_data = database.child("Users").child(uid).get().val()

            email = user_data['email']

            # Authenticate the user using Firebase authentication
            user = authe.sign_in_with_email_and_password(email, current_password)

            # Change the password in Firebase Authentication
            auth.update_user(uid, password=new_password)

            # Redirect to the settings page with a success message
            messages.success(request, "The password has been successfully changed.")
            return render(request, 'settings/change_password.html', {'uid': uid})
```

Figure 56 - Pogled za promjenu lozinke

Kako bi email adresu i lozinku mogli ažurirati u Firebase-ovoj autentifikaciji koristimo biblioteku "firebase_admin" (Firebase, firebase_admin module, 2022). Za implementaciju ove funkcionalnosti potreban nam je Firebase-ov "admin SDK ključ". Ključ generiramo u postavkama projekta u odjeljku "Service accounts". Generirana je datoteka JSON formata koju spremamo u našem Django projektu. (Firebase, Add the Firebase Admin SDK to your server, 2023). Slika ispod prikazuje inicijalizaciju Firebase-ovog admin SDK

```python
cred = credentials.Certificate("./main/zavrsnirad-d6160-firebase-adminsdk-c68do-afe21305d5.json")
firebase_admin.initialize_app(cred)
```

Figure 57 - Inicijalizacija Firebase-ovog admin SDK

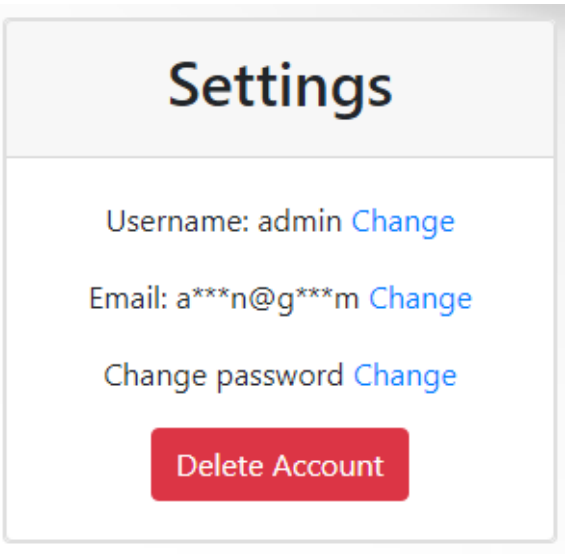

Figure 58 - Postavke

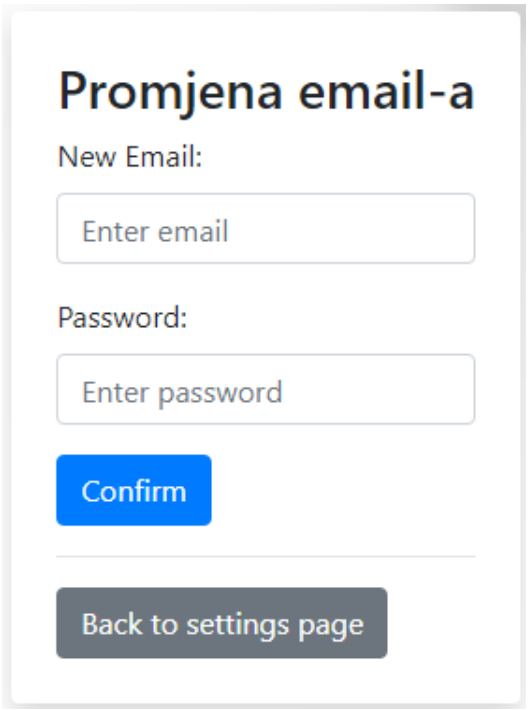

Figure 59 - Promjena email-a

4. PRIKAZ APLIKACIJE

U ovome poglavlju prikazat ćemo aplikaciju kroz slike i pojašnjenje svake slike (najprije je prikazana slika, a ispod nje tekstualno objašnjenje).

Aplikacija je spremljena na GitHub-u na ovoj web adresi:
https://github.com/MarinPopovic/ZavrsniRad

Kako bi se ova web aplikacija mogla koristiti potrebno je instalirati biblioteke "pyrebase" i "firebase_admin" komandama: "pip install pyrebase" i "pip install firebase_admin". Projekt se u Djangu pokreće naredbom "./manage.py runserver".

Unaprijed su stvoreni i korisnički računi za običnog korisnika i administratora:

Admin:
Email: admin@gmail.com
Lozinka: admin123

Korisnik:
Email: Username123@gmail.com
Lozinka: Username123

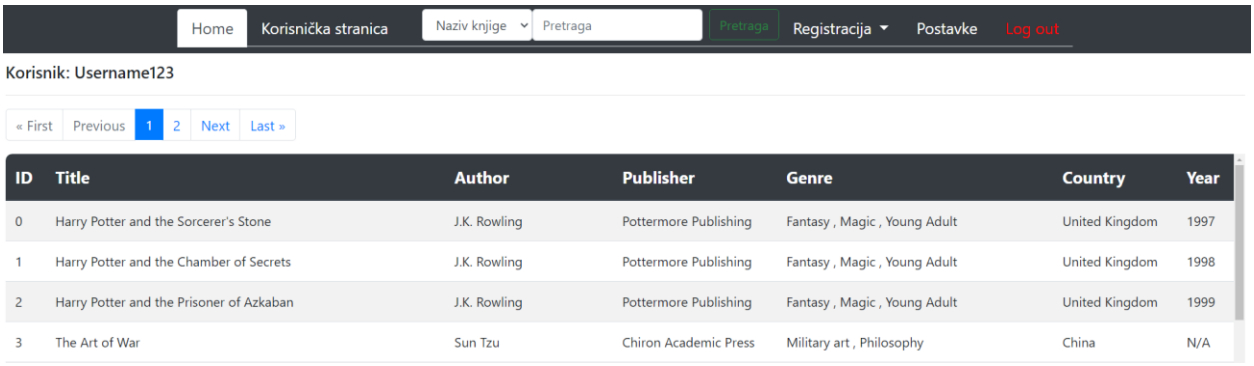

Figure 60 - Početna stranica

Na početnoj stranici nalazi se tablica koja prikazuje popis svih knjiga. Poveznice za "ID" i "Naslov" vode korisnika na stranicu za prikaz individualne knjige. Poveznice "Autor" i "Izdavač" vode korisnika na stranicu koju ispisuju sve knjige od dog autora/izdavača. U tražilici su ponuđene tri opcije za traženje knjige. Osoba ne mora biti autentificirana kako bi imala pristup početnoj stranici, mogućnosti pretraživanja, prikaz knjige po autori/izdavaču

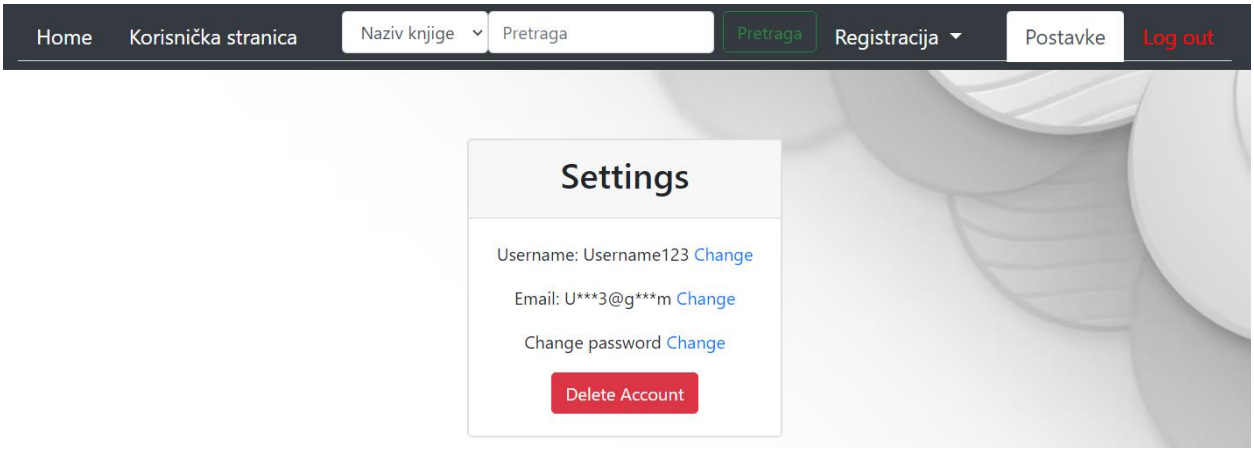

Figure 61 - Postavke

U postavkama korisnik i administrator može promijeniti korisničke podatke unosom novog podatka i lozinke (u slučaju promjene lozinke unosi se stara i nova lozinka).

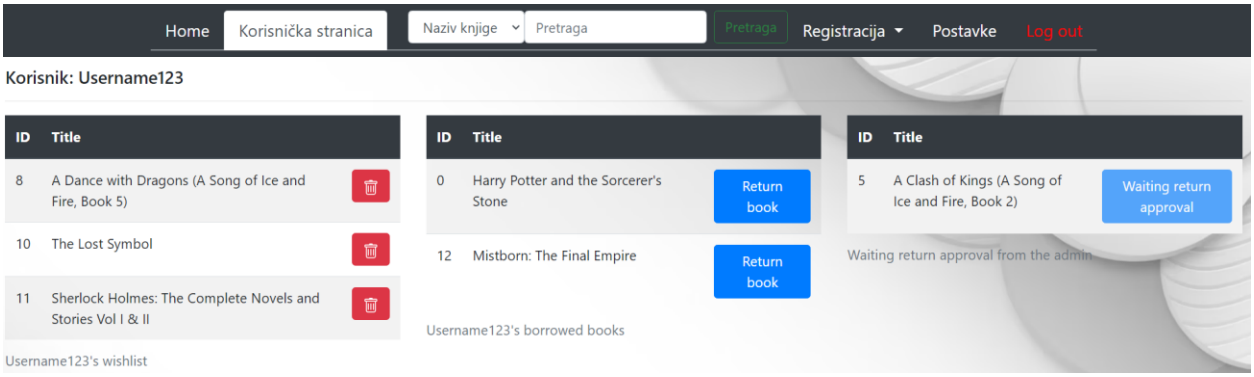

Figure 62 - Korisnička stranica

U korisničkoj stranici su prikazane knjige dodane na popis želja, posuđene knjige i knjige koje su na listi čekanja za odobrenje povratka.

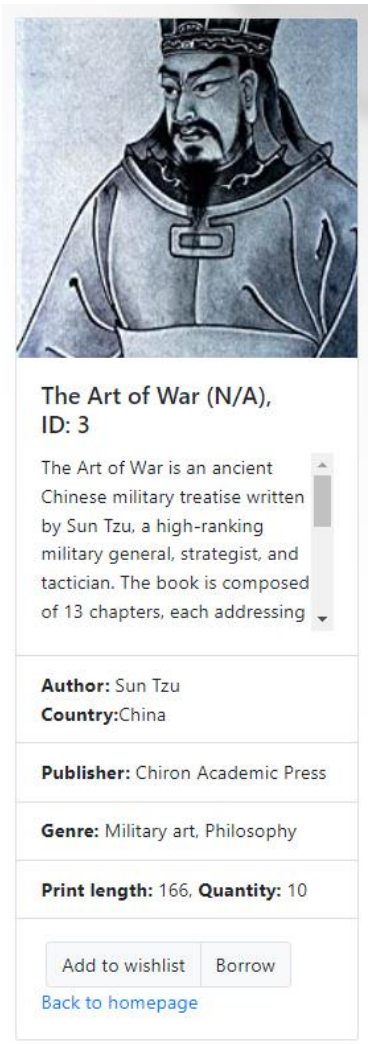

Figure 63 - Prikaz individualne knjige (iz korisničke perspektive)

Figure 64 - Prikaz individualne knjige (iz admin perspektive)

Kod prikaza individualne nalaza se svi podaci o toj knjizi. Običan korisnik može posuditi knjigu i dodati je na listu želja, a administrator ju može obrisati ili ažurirati podatke.

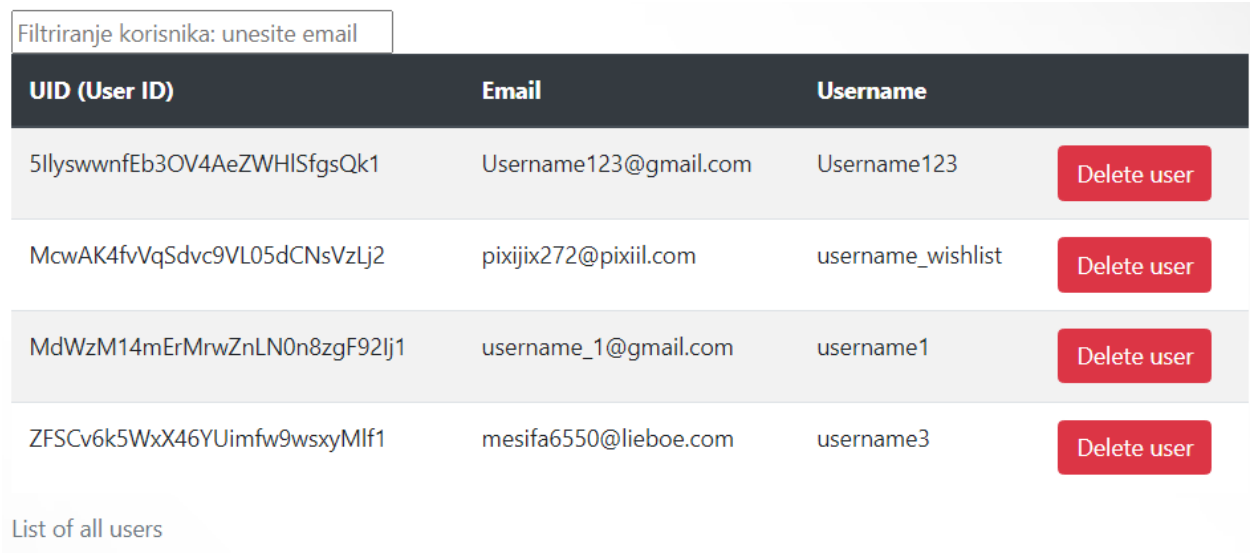

Figure 65 - Admin stranica (popis svih korisnika)

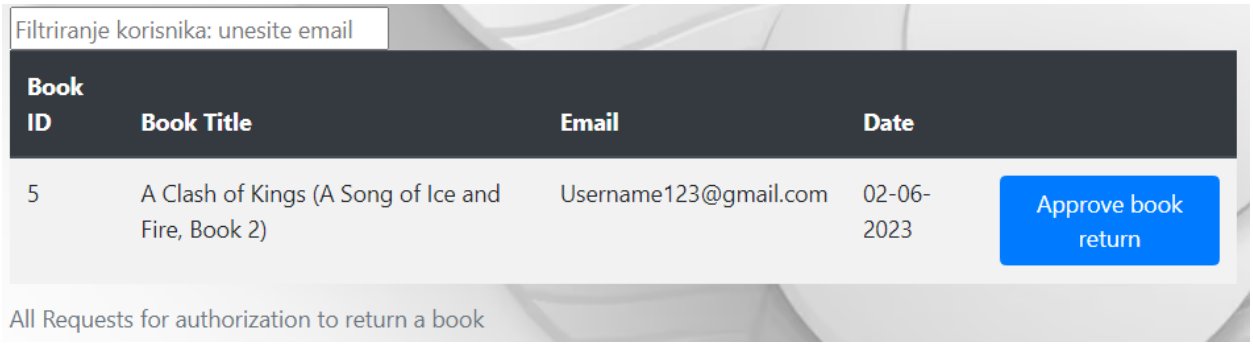

Figure 66 - Admin stranica (popis zahtjeva za odobrenje povratka knjige)

U admin stranici nalaze se dvije liste, popis svih korisnika i popis zahtjeva za odobrenje vraćanje knjige.

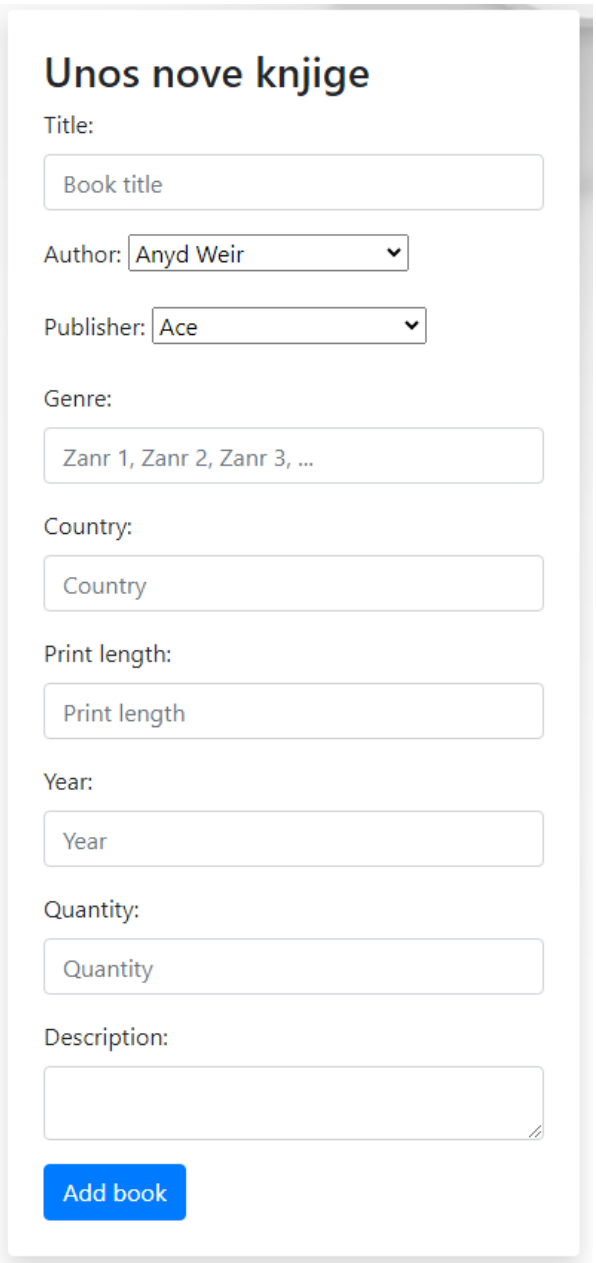

Figure 67 - unos nove knjige

Unos nove knjige se vrši na posebnoj stranici. Autori i izdavači se biraju u padajućem izborniku. Polja "Država", "Godina" i "Opis" nisu obavezna.

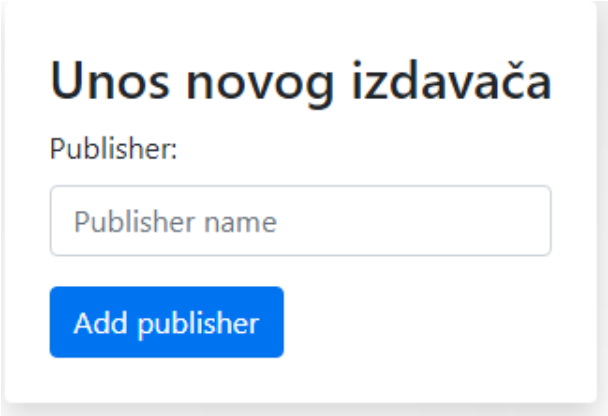

Figure 68 - Unos novog izdavača

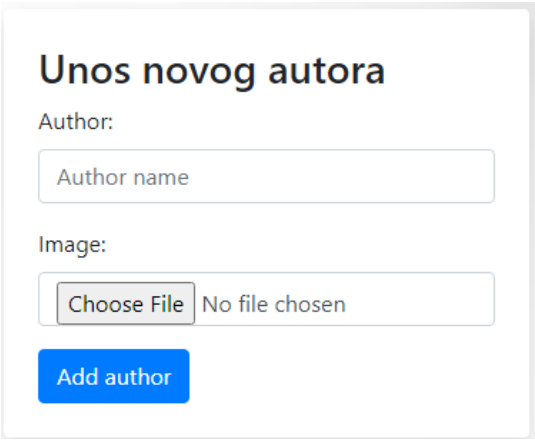

Figure 69 - Unos novog autora

U obrascu za unos izdavača moramo unijeti ime, a kod unosa autora možemo unijeti i sliku.

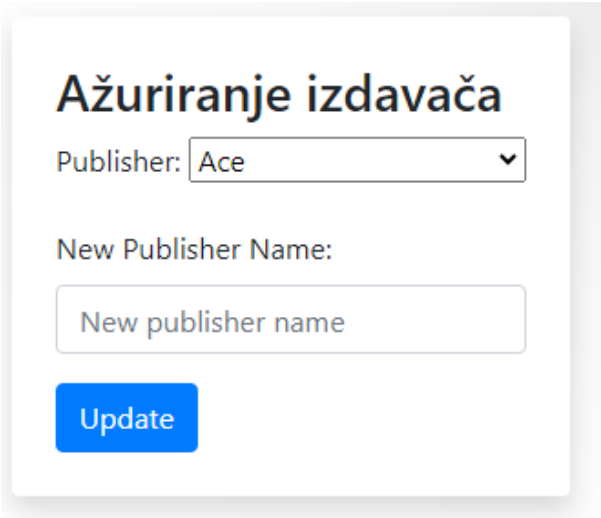

Figure 70 - Ažuriranje izdavača

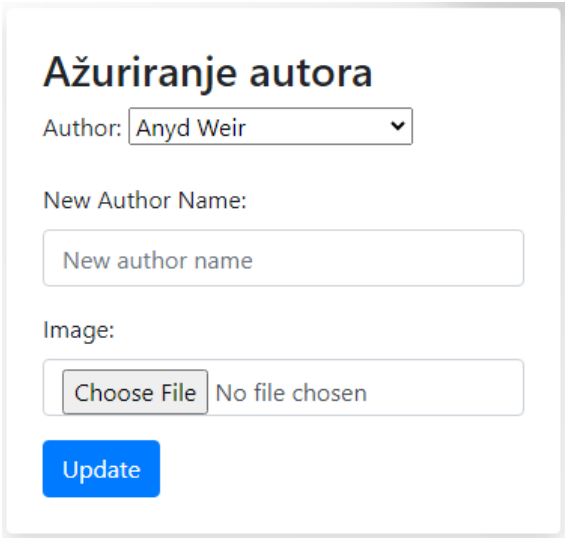

Figure 71 - Ažuriranje autora

U obrascima za ažuriranje najprije odaberemo autora/izdavača iz padajućeg izbornika, potom unesemo novo ime, kod autora možemo unijeti i novu sliku koja nije obavezno polje.

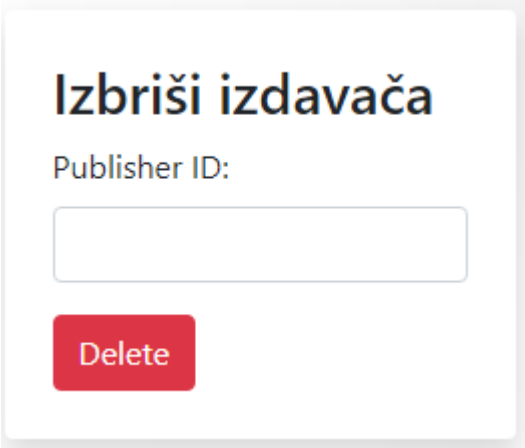

Figure 72 - Brisanje izdavača

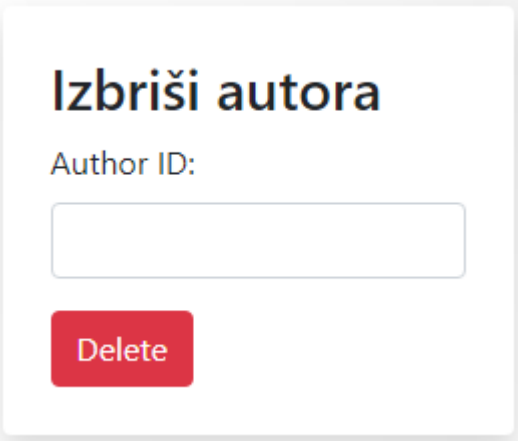

Figure 73 - Brisanje autora

Pri brisanju autora i izdavača unosimo njihove ID vrijednosti.

5. ZAKLJUČAK

U ovom smo radu predstavili osnove Google Firebase-a za razvoja aplikacija. Svi podaci u Firebase-u se spremaju u oblak, a baza podataka je tipa NoSQL (Firebase, Store and sync data in real time, 2023). Ovaj način spremanja podataka je postao jako popularan u zadnjih 10 godina, 2021. godine veličina tržišne vrijednosti pohrane u oblaku bila je procijenjena na 70 milijardi dolara, a do 2029. očekuje se porast na 376 milijarde dolara (Taylor, 2022).
Programer je naučio kako upravljati autentifikacijom, bazom podataka i pohranom. Definiranjem dvije vrste korisnika, običnog korisnika i administratora, pokazali smo njihove mogućnosti. Prikazali smo filtriranje podataka putem tražilice i klikom na poveznice. Pokazali smo administratorske ovlasti za brisanje korisnika iz baze, te načine učitavanja, ažuriranja i brisanja podataka putem obrazaca. Preko obrasca u postavkama smo demonstrirali ažuriranje korisničkih ili administratorskih podataka.

Platforma nudi raznolike proizvode i mogućnosti kao što su Google Analytics za uvid u korištenje aplikacije i promet korisnika, Cloud Messaging, alat za slanje poruka na više platformi, postavljanje aplikacije na poslužitelja, i mnoge druge mogućnosti (Firebase, Firebase Products, 2023).

Platforma Google Firebase nudi mnogo izbora za stvaranje novog korisnika (Google, Twitter, Facebook, telefonski broj, anoniman sign-in, ...). Uz to ponuđene su nam i razne ekstenzije, koje su zapravo gotov kod koje programer može uvesti u svoj rad. Nude se i dvije vrste baze podataka, "Realtime Database" i "Firestore Database". "Realtime Database" omogućuje ugnježđivanje podataka do 32 razine dubine (Firebase, Structure Your Database, 2023).

Kao negativne strane platforme najprije moramo reći da će aplikacija s bazom podataka s 32 nivoa dubokim stablom, morati potencijalno morati doseći 32. razinu (jer pri dohvaćanju podatak iz čvora vraćaju se i svi podčvorovi toga čvora). Budući da je ova usluga u vlasništvu Google-a, samo je Gmail dopušten za prijavu. Također, ne postoji službena podrška za Django.

Prema mom osobnom iskustvu, nisam nailazio na prepreke u korištenju usluge, Firebase-ova web stranica je dobro strukturirana, s korisnim alatima koji su mi pomogli u izradi aplikacije, uz opsežnu dokumentaciju i online vodičima.

Na kraju možemo zaključiti da Google Firebase ima mnogo funkcionalnosti i značajka, te nudi mnogo mogućnosti za izradu aplikacija, međutim treba napomenuti da kod duboko ugniježđenih baza podataka aplikacija se može dugo učitavati pa programer treba pažljivo planirati strukturu baze podataka.

POPIS LITERATURE

Arora, N. (2020, February 13). *ChoiceField – Django Forms*. Retrieved March 10, 2023, from GeeksforGeeks: https://www.geeksforgeeks.org/choicefield-django-forms/

Childs-Maidment, J. (2020, February 18). *Pyrebase (Add Pyrebase to your application)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#add-pyrebase-to-your-application

Childs-Maidment, J. (2020, February 18). *Pyrebase (Authentication)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#authentication

Childs-Maidment, J. (2020, February 18). *Pyrebase (Database)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#database

Childs-Maidment, J. (2020, February 18). *Pyrebase (Installation)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#installation

Childs-Maidment, J. (2020, February 18). *Pyrebase (Pyrebase)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#pyrebase

Childs-Maidment, J. (2020, February 18). *Pyrebase (Retrieve data)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#retrieve-data

Childs-Maidment, J. (2020, February 18). *Pyrebase (Storage)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#storage

Childs-Maidment, J. (2020, February 18). *Pyrebase (Use Services)*. Retrieved March 1, 2023, from GitHub: https://github.com/thisbejim/Pyrebase#use-services

Django. (2023). *Django shortcut functions*. Retrieved March 1, 2023, from Django: https://docs.djangoproject.com/en/4.2/topics/http/shortcuts

Django. (2023). *django.urls utility functions*. Retrieved March 7, 2023, from Django: https://docs.djangoproject.com/en/4.2/ref/urlresolvers

Django. (2023). *Form fields*. Retrieved 1 March, 2023, from Django: https://docs.djangoproject.com/en/4.2/ref/forms/fields/

Django. (2023). *How to override templates*. Retrieved March 1, 2023, from Django: https://docs.djangoproject.com/en/4.2/howto/overriding-templates/

Django. (2023). *How to use sessions*. Retrieved March 22, 2023, from Django: https://docs.djangoproject.com/en/4.2/topics/http/sessions/

Django. (2023). *The Django template language: for Python programmers*. Retrieved March 1, 2023, from Django: https://docs.djangoproject.com/en/4.2/ref/templates/api/

Django. (2023). *The Forms API*. Retrieved March 10, 2023, from Django: https://docs.djangoproject.com/en/4.2/ref/forms/api/

Django. (2023). *URL dispatcher*. Retrieved March 1, 2023, from Django: https://docs.djangoproject.com/en/4.2/topics/http/urls

Django. (2023). *URL dispatcher (Example)*. Retrieved March 3, 2023, from Django: https://docs.djangoproject.com/en/4.2/topics/http/urls/#example

Django. (2023). *URL dispatcher (How Django processes a request)*. Retrieved March 3, 2023, from Django: https://docs.djangoproject.com/en/4.2/topics/http/urls/#how-django-processes-a-request

Django. (2023). *Working with forms*. Retrieved 1 March, 2023, from Django: https://docs.djangoproject.com/en/4.2/topics/forms

Django. (2023). *Writing views*. Retrieved March 1, 2023, from Django: https://docs.djangoproject.com/en/4.2/topics/http/views/

Firebase. (2022, August 5). *firebase_admin module*. Retrieved April 9, 2023, from Firebase: https://firebase.google.com/docs/reference/admin/python/firebase_admin

Firebase. (2022, February 17). *Getting started with Firebase Storage on the web*. Retrieved March 1, 2023, from Youtube: https://www.youtube.com/watch?v=-IFRVMEhZDc

Firebase. (2022, March 17). *Getting started with the Firebase Realtime Database on the web*. Retrieved March 1, 2023, from Youtube: https://www.youtube.com/watch?v=pP7quzFmWBY

Firebase. (2023, June 16). *Add Firebase to your JavaScript project*. Retrieved March 1, 2023, from Firebase: https://firebase.google.com/docs/web/setup

Firebase. (2023, June 21). *Add the Firebase Admin SDK to your server*. Retrieved April 7, 2023, from Firebase: https://firebase.google.com/docs/admin/setup

Firebase. (2023, June 16). *Choose a Database: Cloud Firestore or Realtime Database*. Retrieved March 1, 2023, from Firebase: https://firebase.google.com/docs/database/rtdb-vs-firestore

Firebase. (2023, June 16). *Cloud Firestore*. Retrieved March 1, 2023, from Firebase: https://firebase.google.com/docs/firestore

Firebase. (2023, June 16). *Cloud Firestore Library and framework integrations*. Retrieved March 1, 2023, from Firebase: https://firebase.google.com/docs/firestore/library-integrations#firebaseui

Firebase. (2023, June 16). *Firebase Authentication*. Retrieved March 1, 2023, from Firebase: https://firebase.google.com/docs/auth

Firebase. (2023). *Firebase console*. Retrieved 1 March, 2023, from Firebase: https://console.firebase.google.com

Firebase. (2023, June 16). *Firebase Extensions*. Retrieved March 1, 2023, from Firebase: https://firebase.google.com/docs/extensions#how_does_it_work

Firebase. (2023). *Firebase Extensions Hub*. Retrieved March 1, 2023, from Firebase: https://extensions.dev/extensions

Firebase. (2023, June 26). *Firebase Hosting*. Retrieved July 1, 2023, from Firebase: https://firebase.google.com/docs/hosting

Firebase. (2023). *Firebase Products*. Retrieved July 1, 2023, from Firebase: https://firebase.google.com/products-release

Firebase. (2023, June 16). *Google Analytics*. Retrieved 1 March, 2023, from Firebase: https://firebase.google.com/docs/analytics

Firebase. (2023, June 21). *Manage Users in Firebase*. Retrieved April 15, 2023, from Firebase: https://firebase.google.com/docs/auth/web/manage-users

Firebase. (2023, June 16). *SDKs and client libraries*. Retrieved March 2023, 1, from Firebase: https://firebase.google.com/docs/firestore/client/libraries#google_cloud_client_libraries

Firebase. (2023). *Store and sync data in real time*. Retrieved March 1, 2023, from Firebase: https://firebase.google.com/products/realtime-database

Firebase. (2023, June 26). *Structure Your Database*. Retrieved July 2023, 1, from Firebase: https://firebase.google.com/docs/database/web/structure-data

Firebase. (2023, June 16). *Understand Firebase Realtime Database Security Rules*. Retrieved 1 March, 2023, from Firebase: https://firebase.google.com/docs/database/security

Firebase. (2023, June 16). *What can you do with Firebase Hosting?* Retrieved March 1, 2023, from Firebase: https://firebase.google.com/docs/hosting/use-cases

Python. (2023, June 19). *8. Errors and Exceptions*. Retrieved March 7, 2023, from Python: https://docs.python.org/3/tutorial/errors.html

Richardson, K. (2013, April 9). *Create an empty child record in Firebase*. Retrieved March 10, 2023, from Stack Overflow: stackoverflow.com/questions/15911165/create-an-empty-child-record-in-firebase

Taylor, P. (2022, September 20). *Size of the cloud storage market worldwide from 2021 to 2029*. Retrieved July 1, 2023, from Statista: https://www.statista.com/statistics/1322710/global-cloud-storage-market-size/

W3Schools. (2023). *HTTP Request Methods*. Retrieved March 3, 2023, from W3Schools: https://www.w3schools.com/tags/ref_httpmethods.asp

Wikipedia. (2023, June 14). *Firebase*. Retrieved March 1, 2023, from Wikipedia: https://en.wikipedia.org/wiki/Firebase

Wikipedia. (2023, June 14). *Google Analytics*. Retrieved March 1, 2023, from Wikipedia: https://en.wikipedia.org/wiki/Google_Analytics

Wikipedia. (2023, May 5). *Login session*. Retrieved March 7, 2023, from Wikipedia: https://en.wikipedia.org/wiki/Login_session

POPIS SLIKA

POPIS IZVORA SLIKA I DOKUMENATA KORIŠTENIH U WEB STRANICI PROJEKTA

Slike autora, https://www.goodreads.com

Pozadinska slika na web stranici projekta, https://www.freepik.com/free-vector/realistic-white-monochrome-background_15364616.htm#query=website%20white%20background&position=42&from_view=search&track=ais

Bootstrap ikona "Trash", https://icons.getbootstrap.com/icons/trash/