Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/15BE39Zeljkovic

PROŠIRIVI SISTEM ZA PROVERU PLAGIJARIZMA BAZIRAN NA KOMPONENTAMA

EXTENSIBLE COMPONENT BASED PLAGIARISM DETECTION SYSTEM

Nikola Zeljković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO I INFORMACIONE TEHNOLOGIJE

Kratak sadržaj – *U radu je opisan proširiv sistem za proveru plagijarizma na skupovima dokumenata različitih tipova. Sistem pruža mogućnost veoma lakog dodavanja komponenti za: proveru plagijarizma, pristup referentnom skupu dokumenata i predprocesiranje dokumenata. Prethodna podela komponenti daje mogućnost provere plagijarizma širokog spektra različitih tipova dokumenata.*

Ključne reči: *Detekcija plagijarizma, obrada dokumenata, proširiva arhitektura, plugin*

Abstract – *The goal of this paper is to present an extensible component based system for plagiarism detection. The system can be easily extended using components for: plagiarism detection, accessing reference sets and document processing. Using these components allow for performing plagiarism detection on a wide range of document types.*

Keywords: *Plagiarism detection, document processing, extensible arhitecture, plugin*

1. UVOD

Pojam plagijarizma se definiše kao upotreba i predstavljanje tuđih tekstova, umetničkih radova, izvornih kodova pisanim u različitim programskim jezicima, itd. kao ličnu intelektualnu svojinu. Ovde spada falsifikovanje tuđeg dela i rada čime se krši niz autorskih prava. Pored različitih ljudskih delatnosti veliku pažnju privlači plagijarizam u naučnoj i obrazovnoj oblasti [1].

Do velike rasprostranjenosti plagijarizma doveo je ubrzan razvoj tehnologije i mogućnosti koje ona pruža. U današnje vreme ljudi imaju mogućnost da na veoma jednostavan i brz način, upotrebom interneta dođu do velikog broja različitih izvora informacija. Postoji veliki broj softverskih rešenja za proveru plagijarizma koji daju mogućnost analize naučnih radova i drugih tekstualnih dokumenata.

U naučnom domenu postoji određeni broj tipova dokumenata za koje je poželjno vršiti proveru plagijarizma. U ovom radu je predstavljen sistem koji za cilj ima da pruži mogućnost provere plagijarizma za sve tipove dokumenata koji imaju značaj u određenim domenima gde pojam intelektualne svojine igra bitnu ulogu.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivanović, red. prof.

Posmatranjem domena u kojem se stvaraju tekstualni dokumenti i softverska rešenja predstavljena pomoću izvornih kodova, poželjno je vršiti proveru plagijarizma za date tipove dokumenata. U slučaju da se u ovom domenu pojavi potreba za podržavanjem trećeg tipa dokumenta, bilo bi poželjno da postoji mogućnost lakog proširenja postojećeg sistema za detekciju plagijarizma da podrži novi tip dokumenta. U ovakvom sistemu ne postoji potreba da se implementira ili konfiguriše potpuno novi sistem radi podržavanja novog tipa dokumenta, već je dovoljno samo dodati komponente koje će rukovati sa novim tipom dokumenta.

2. TEORIJSKE OSNOVE

Za implementaciju lako proširivog sistema potrebno je razmotriti sam proces koji se automatizuje upotrebom datog sistema. Da bi se izvršila automatizovana provera plagijarizma između jednog dokumenta (eng. Candidate document) i skupa referentnih dokumenata (eng. Reference documents) potrebno je izvršiti predprocesiranje datih dokumenata, nakon čega algoritam za proveru plagijarizma pristupa datim dokumentima radi utvrđivanja potencijalnog plagijarizma.

2.1. Dokument

U ovom radu pod pojmom dokumenta podrazumeva se pojam digitalnog dokumenta. Digitalni dokument predstavlja računarski obrađenu informaciju kojom se rukuje kao osnovnom jedinicom obrade.

U ovom radu se spominju sledeći tipovi dokumenata:

- Tekstulani dokumenti u PDF formatu
- Izvorni kodovi u programskom jeziku Python

2.2. Metapodaci dokumenata

Metapodaci dokumenata predstavljaju podatke o dokumentima. Osnovni metapodaci koji su bitni za ovaj sistem su naziv, tip i putanja do reprezentacije dokumenta koja će se koristiti prilikom izvršavanja provere plagijarizma. Dokument sam po sebi ne sadrži sve potrebne informacije za skladištenje, uspostavljanje veze sa drugim entitetima i proveru plagijarizma datog dokumenta. Ove informacije su rezultat predprocesiranja u čuvaju se u skupu metapodataka.

2.3. Predprocesiranje dokumenata

Predprocesiranje dokumenta je proces u kojem se izvršava niz operacija nad datim dokumentom. Svaka operacija proizvodi određenu reprezentaciju datog dokumenta i/ili metapodatak koji se skladišti u skupu metapodataka datog dokumenta. Određene operacije se mogu izvršiti nad dokumentom samo nakon što je predprocesiranje datog dokumenta završeno.

2.4. Skladištenje dokumenata i metapodataka

U okviru jednog sistema koji se delom bavi skladištenjem dokumenata, metapodaci dokumenata mogu se čuvati na različite načine. Originalna reprezentacija, kao i ostale reprezentacije datog dokumenta se mogu čuvati na fajl sistemu ili uz pomoć određenog eksternog rešenja za skladištenje dokumenata. Kompanije koje nude usluge u računarskom oblaku imaju servise za čuvanje masivnih količina podataka. Jedan od najpopularnijih primera za ovakvu vrstu servisa je Amazonov servis S3 [2]. Lokacije različitih reprezentacija određenog dokumenta se čuvaju u metapodacima datog dokumenta.
Skup dokumenata sa kojima se može vršiti provera plagijarizma može da postoji i van specijalizovanog sistema za skladištenje dokumenata, npr. dokumenti u određenom direktorijumu jednog računara.

2.5. Detekcija plagijata

Postoji veliki broj tipova dokumenata koji mogu biti kandidati za proveru plagijarizma. Pojedinačni pristupi za proveru plagijarizma su prilagođeni tipovima dokumenata za koje su implementirani. Postoji značajna razlika u pristupu prilikom utvrđivanja plagijarizma teksta, umetničkog dela ili izvornog koda određenog programskog jezika. Postoji razlika čak i ako posmatramo dva izvorna koda pisana različitim programskim jezicima. Za svaki pristup čija upotreba se razmatra, treba utvrditi koji tipovi dokumenata kao i koje reprezentacije dokumenta su potrebne za primenu datog pristupa.

2.6. Arhitektura bazirana na komponentama

Arhitektura bazirana na komponentama nam pruža mogućnost lakog dodavanja novih komponenti u sistem [3] radi podržavanja različitih skladišta dokumenata kao i različite algoritme za detekciju plagijarizma. Poželjno je omogućiti dodavanje novih komponenti bez privremenog gašenja sistema da bi vreme za koje sistem nije dostupan sveli na minimum.

2.6.1. Visual Studio Code

Visual Studio Code (VSCode u nastavku) je univerzalni editor izvornih kodova proizveden od strane Microsoft-a [4], i odličan primer softvera čija arhitektura je zasnovana na komponentama. VSCode pruža mogućnost instalacije ogromnog broja različitih ekstenzija za podršku novih programskih jezika, *debug* alata kao i drugih alata koji olakšavaju proces programskog razvoja. Ovom arhitekturom VSCode ima sposobnost da se konfiguriše u idealno razvojno okruženje nezavisno od programskog jezika, i samim tim uklanja potrebu instalacije drugih softverskih alata za pisanje izvornih kodova.

3. SPECIFIKACIJA SISTEMA

U ovom poglavlju je opisana specifikacija sistema za detekciju plagijarizma. Sistem pruža mogućnost lakog dodavanja komponenti koje imaju mogućnost pristupanja različitim skupovima podataka kao i izvršavanja različitih algoritama detekcije plagijarizma na datim skupovima dokumenata.

3.1. Servis za autentifikaciju i autorizaciju

Servis za autentifikaciju i autorizaciju obezbeđuje skup funkcionalnosti sa ciljem da određenim delovima sistema pristup imaju samo registrovani korisnici. Ovaj servis treba da obezbedi sledeće funkcionalnosti:

- Registracija korisnika u sistem
- Skladištenje informacija o korisnicima u sistemu
- Autentifikacija registrovanog korisnika
    - Izdavanje ključa za pristup određenim resursima sistema

Ostali servisi u sistemu pružaju određene funkcionalnosti koje se mogu koristiti samo uz upotrebu ključa za pristup datim funkcionalnostima.

3.2. Servis za detekciju plagijarizma

Servis za detekciju plagijarizma pruža mogućnost učitavanja sledećih komponenti prilikom izvršavanja: komponente za izvršavanje provere plagijarizma i komponente za pristup dokumentima.

Pored toga, ovaj servis nudi niz funkcionalnosti za rukovanje zadacima za proveru plagijarizma. Prilikom kreiranja zadatka za proveru plagijarizma, korisnik zadaje identifikator dokumenta koji se upoređuje sa referentnim dokumentima, identifikator konkretnog pristupa za proveru plagijarizma i identifikator referentnog skupa podataka. Zadaci se izvršavaju u asinhronom režimu. Korisnik u svakom trenutku može da proveri status zadataka koje je on kreirao. U slučaju da je neki zadatak izvršen, korisnik može da pregleda rezultate datog zadatka.

3.2.1. Komponente za detekciju plagijarizma

Komponente za detekciju plagijarizma moraju specificirati sa kojim tipovima dokumenata su kompatibilni kao i koji metapodaci su potrebni za upotrebu određene komponente za detekciju plagijarizma.

3.2.2. Komponente za pristup dokumentima

Svaka komponenta ovog tipa mora specificirati koji metapodaci su dostupni na svim dokumentima kojima pristupamo ovom komponentom. Ovo nam daje mogućnost da unapred znamo da li je određena komponenta za pristup dokumentima kompatibilna sa određenom komponentom za detekciju plagijarizma.

3.3. Servis za obradu dokumenata

Servis za obradu dokumenata pruža mogućnost otpremanja i skladištenja dokumenata i metadpodataka o datim dokumentima, kao i obradu datih dokumenata. Osnovna funkcionalnost ovog servisa je inicijalna obrada dokumenata prilikom koje se kreiraju osnovni metapodaci za dati dokument. Pored toga, ovaj servis daje mogućnost za učitavanje komponenti za dodatnu obradu svih dokumenata.

3.3.1. Komponente za obradu dokumenata

Komponente za obradu su zadužene da izvrše dodatnu obradu dokumenata i kao rezultat imaju podatke potrebne za proces detekcije plagijarizma.

3.4. Korisnički interfejs

Korisnički interfejs treba da omogući kontrole za registraciju i prijavu u sistem, kao i kontrole za rukovanje određenim funkcionalnostima sistema. U ove funkcionalnosti spadaju: pregled instaliranih komponenti za detekciju plagijarizma, pregled skupova dokumenata

odnosno komponenti za pristup skupovima dokumenata i rukovanje zadacima. Korisnici sistema mogu da kreiraju zadatke i da pregledaju rezultate datih zadataka upotrebom korisničkog interfejsa.

4. IMPLEMENTACIJA SISTEMA

U ovom poglavlju su predstavljeni implementacioni detalji svih delova i komponenti sistema. Opis delova i komponenti sistema podrazumeva pregled konceptualnih rešenja i upotrebljenih tehnologija.

Autentifikacija i autorizacija, servis za detekciju plagijarizma i servis za obradu dokumenata su implementirani pomoću Java 1.8 programskog jezika u formi veb aplikacija u Spring Boot radnom okviru.

4.1. Autentifikacija i autorizacija

Autentifikacija i autorizacija je implementirana pomoću Spring Security dodatka za Spring Boot radni okvir. Sistem ima dve različite uloge: administrator i nastavnik. Nastavnik ima mogućnost da otpremi i proveri da li je dokument plagijat dok administrator pored ovoga ima opciju da dodaje i briše dodatne komponente.

4.1.1. Model korisnika

Model korisnika obuhvata sledeće informacije: identfikator korisnika predstavljen pomoću UUID-a (eng. Universally Unique Identifier), email adresa, lozinka, ime, prezime i skup uloga koje korisnik ima u sistemu.

4.1.2. Registracija korisnika

Prilikom registracije, korisnik prosleđuje svoju email adresu, lozinku, ime i prezime servisu za autentifikaciju i autorizaciju, nakon čega servis dodeljuje odgovarajući identifikator datom korisniku i skladišti sve informacije ukoliko korisnik sa datom email adresom već ne postoji u sistemu.

Bitna bezbednosna stavka kod ovih sistema je da se lozinka nikako ne sme čuvati u izvornom obliku već je obavezno primeniti algoritme za heširanje i čuvati samo heširani oblik lozinke [5].

4.1.3. Prijava na sistem

Prilikom prijave na sistem, korisnik je dužan da prosledi svoju email adresu kao i lozinku koju je izabrao prilikom registracije na sistem. Nakon uspešne prijave korisniku se izdaje JWT token [6] (eng. JSON Web Token) sa kojim dobija mogućnost da pristupi ostalim resursima u sistemu što uključuje ovaj i ostale servise.

4.2. Dodavanje novih komponenti u runtime-u

Javin mehanizam za učitavanje klasa je deo izvršnog okruženja Jave, i ima mogućnost dinamičkog učitavanja Java klasa u Java virtuelnu mašinu [7].

Java biblioteke su tipično upakovane u JAR datoteke koje mogu da sadrže objekte različitih vrsta. Najbitnija vrsta objekta u JAR datoteci je Java klasa. Bitno je napomenuti da Javin mehanizam za učitavanje klasa ima mogućnost učitavanja klase sa određenim imenom samo jednom u toku jednog izvršavanja.

Kada se Javina virtuelna mašina pokrene tri klase za učitavanje klasa se koriste: *bootstrap class loader*, *extensions class loader* i *systems class loader*. *Bootstrap class loader* je zadužen za učitavanje Javinih ugrađenih biblioteka (eng. Core libraries) koji se nalaze u *<JAVA_HOME>/jre/lib* direktorijumu. *Extensions class loader* učitava dodatne biblioteke koje se nalaze u *<JAVA_HOME>/jre/lib/ext* direktorijumu. *System class loader* učitava klase koje se nalaze na classpath-u. Navedeni mehanizmi su implementirani u Javi i samim tim korisnici jezika mogu implementirati svoje mehanizma za učitavanje klasa. Jedan od tih mehanizama je *java.net.URLClassLoader* koji pruža mogućnost učitavanja klasa iz fajl sistema kao i JAR fajlova.

Za implementaciju učitavanje komponenti korišćen je navedeni *URLClassLoader*. U sistemu su definisani interfejsi koji omogućuju komponentama da se prilagode sistemu. Interfejsi su dati na listinzima 4.2.1., 4.2.2., 4.2.3 i 4.2.4.

```
public interface Plugin {
  PluginMetadata getPluginMetadata();
}
```

Listing 4.2.1. *Osnovni interfejs koji mora biti ispoštovan od strane svih komponenti.*

Klasa *PluginMetadata* sadrži sledeća polja: identifikator komponente, naziv komponente, opis komponente, listu naziva metapodataka koji su potrebni da bi data komponenta rukovala sa datim dokumentom, lista identifikatora komponenti koji se moraju učitati pre ove komponente.

```
public interface DetectorPluginFactory extends Plugin {
  DetectorPlugin createInstance();
}
```

Listing 4.2.2. *Interfejs za komponente koje vrše detekciju plagijarizma*

```
public interface DataAccessPlugin extends Plugin {
  Dataset loadDataset(DocumentMetadata candidateDocument);
}
```

Listing 4.2.3. *Interfejs za komponente koje vrše pristup dokumentima*

```
public interface ProcessingPlugin extends Plugin {
  String getResultFieldName();
  void process(DocumentMetadata documentMetadata,
Consumer<Object> storeValueForDocument);
}
```

Listing 4.2.4. *Interfejs za komponente koje vrše dodatnu obradu dokumenata*

4.3. Servis za detekciju plagijarizma

Servis za detekciju plagijarizma je veb aplikacija koja koordiniše između korisnika, komponenti za detekciju plagijarizma i komponenti za pristup dokumentima.

Pored toga sistem daje mogućnost korisnicima da dobiju osnovne informacije o svim komponentama koje su učitane u sistem. Prilikom zahteva za izlistavanje komponenti, sistem proverava da li postoje nove komponente u direktorijumu gde se nalaze instalirane komponente. U slučaju da postoje, sistem ih učitava i uključuje u rezultate zahteva.

Na osnovu dobijenih informacija korisnik može da napravi i pošalje zahtev za kreiranje zadatka, nakon čeka sistem skladišti dati zadatak i šalje odgovor korisniku da je zadatak uspešno kreiran. Sistem periodično proverava da li postoje zadaci koji trebaju da se izvrše, pokreće njihovo izvršavanje i skladišti rezultate.

4.3.1. Komponente za detekciju plagijarizma

Postojeća rešenja za detekciju plagijarizma se mogu adaptirati radi priključivanja u sistem. Ova implementacija se sastoji od dve adapter komponente koje adaptiraju dva postojeća rešenja za detekciju plagijarizma. Jedna komponenta pruža detekciju plagijarizma među tekstualnim dokumentima [8] dok druga radi sa Python izvornim kodovima [9].

4.3.2. Komponente za pristup dokumentima

Ova implementacija trenutno nudi jednu komponentu koja pruža pristup dokumentima koji se nalaze na servisu za obradu dokumenata.

4.4. Servis za obradu dokumenata

Servis za obradu dokumenata je implementiran u obliku veb aplikacije. Reprezentacije dokumenata su skladištene u fajl sistemu. Naziv sačuvanih datoteka je predstavljen jednim UUID-om. Metapodaci su skladišteni u MongoDB nerelacionoj bazi podataka. Osnovni metapodaci koji se čuvaju prilikom skladištenja dokumenta su originalni naziv dokumenta, skladišteni naziv dokumenta, tip datoteke, ekstenzija naziva datoteke. Dodatni metapodaci se generišu pomoću dodatnih komponenti za obradu.

4.4.1. Komponente za obradu dokumenata

Postojeća rešenja za obradu dokumenata se mogu adaptirati radi priključivanja u sistem. Ova implementacija se sastoji od dve adapter komponente koje adaptiraju dve postojeće implementacije obrade dokumenta. Jedna komponenta pruža ekstrakciju teksta iz PDF datoteka dok druga generiše MD5 *checksum*. Naziv datoteke u kojoj se nalazi ekstraktovani tekst, kao i MD5 checksum originalnog dokumenta se skladišti u skupu metapodataka datog dokumenta.

4.5. Korisnički interfejs

Korisnički interfejs je implementiran pomoću Jave u Vaadin 21 radnom okviru.

5. ZAKLJUČAK

Broj različitih tipova dokumenata raste velikom brzinom u različitim domenima u kojima intelektualna svojina igra bitnu ulogu. Cilj ovog rada je predstavljanje sistema koji ne ograničava korisnike na određeni predefinisani skup tipova dokumenata. U slučaju da se pojavi potreba za podršku novog tipa dokumenta, sistem se može lako proširiti adaptacijom postojećih rešenja za dati tip dokumenta ili implementacijom novog rešenja.

5.1. Dalji razvoj sistema

Sistemi ovog tipa imaju generalan problem performansi usled povećanog broja digitalnih dokumenata sa kojima oni raspolažu. Pretpostavimo da provera plagijarizma traje N sekundi po dokumentu u referentnom skupu. Za M dokumenata potrebno nam je $T = MN$ sekundi. Tradicionalno horizontalno skaliranje u ovom slučaju ne bi donelo do poboljšanja performansi prilikom provere individualnih dokumenata, već bi doprinelo slučajevima gde ima dosta zahteva od različitih korisnika. Da bismo skaliranjem dobili bolje performanse za individualne dokumente, skaliranje se mora implementirati tako da se referentni skup podeli u particije i da se ulazni dokument proverava sa svakom od particija.

6. LITERATURA

[1] East, J., 2006. The problem of plagiarism in academic culture. International Journal for Educational Integrity, 2(2).

[2] Jung, M., Mallering, S., Dalbhanjan, P., Chapman, P. and Kassen, C., 2016. Microservices on AWS. Amazon Web Services, Inc., New York, NY, USA, Tech. Rep.

[3] Belguidoum, M. and Dagnat, F., 2007. Dependency management in software component deployment. Electronic Notes in theoretical computer science, 182, pp.17-32.

[4] Microsoft (2016). Visual Studio Code. [online] Visualstudio.com. Dostupno na: https://code. visualstudio.com/. [Pristupljeno 6. 8. 2021]

[5] Sriramya, P. and Karthika, R.A., 2015. Providing password security by salted password hashing using bcrypt algorithm. ARPN journal of engineering and applied sciences, 10(13), pp.5551-5556.

[6] Jones, M., Campbell, B. and Mortimore, C., 2015. JSON Web Token (JWT) profile for OAuth 2.0 client authentication and authorization Grants. May-2015. [online]. Dostupno na: https://tools.ietf .org/html/rfc7523. [Pristupljeno 6. 8. 2021]

[7] Liang, S. and Bracha, G., 1998. Dynamic class loading in the Java virtual machine. Acm sigplan notices, 33(10), pp.36-44.

[8] GitHub. (n.d.). GitHub - JonathanReeve/text-matcher: A simple text reuse detection CLI tool. [online] Dostupno na: https://github.com/JonathanReeve/ text-matcher. [Pristupljeno 6. 8. 2021]

[9] GitHub. (n.d.). GitHub - fyrestone/pycode_similar: A simple plagiarism detection tool for python code. [online] Dostupno na: https://github.com/fyres tone/pycode_similar. [Pristupljeno 6. 8. 2021]

Kratka biografija:

Nikola Zeljković rođen je 30. januara 1996. godine u Novom Sadu, Republika Srbija. Godine 2015. upisao je Fakultet tehničkih nauka u Novom Sadu smer Softversko inženjerstvo i informacione tehnologije. 2019. godine diplomirao je na osnovnim studijama na Fakultetu tehničkih nauka, iste godine potom upisuje master studije na smeru Softversko inženjerstvo i informacione tehnologije. Kontakt: nikzeljkovic@gmail.com