Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/10BE47Vasiljevic

SPRINGEXTRACTOR APLIKACIJA ZA AUTOMATIZACIJU PROCESA PREGLEDANJA RADOVA

SPRINGEXTRACTOR APPLICATION FOR AUTOMATING THE PROCESS OF REVIEWING STUDENT PAPERS

Aleksandar Vasiljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je opisano idejno rešenje problema automatizacije procesa pregledanja studentskih radova. U tu svrhu implementirana je aplikacija Spring-Extractor – rešenje koje se oslanja na radni okvir Spring, relacionu bazu podataka MySQL i Angular frejmvork. Uz to, dat je uporedni prikaz prednosti i mana napredne u odnosu na inicijalnu verziju, a izložene su i potencijalne buduće tačke proširenja i unapređenja.*

Ključne reči: *automatizacija, aplikacija, Java, Spring, MySQL, Angular, proces pregledanja radova*

Abstract – *The paper describes the conceptual solution to the problem of automating the process of reviewing student papers. For this purpose, the SpringExtractor application was implemented – a solution that relies on the Spring framework, relational MySQL database and Angular framework. In additional, a comparative overview of the advantages and disadvantages of the advanced version is given, and potential future points of expansion and improvement are presented.*

Keywords: *automatization, application, Java, Spring, MySQL, Angular, the process of reviewing student papers*

1. UVOD

Inicijalno su računari bili prilagođeni rešavanju relativno jednostavnih problema budući da je njihova resursna moć bila ograničena. Kako su se problemi usložnjavali, javila se potreba za nadogradnjom računarske inteligencije, što je posledično označilo podizanje nivoa apstrakcije, a samim tim i mogućnost rešavanja kompleksnijih problema šireg spektra.

Integracija sve moćnijih računara, novorazvijenih programskih jezika i potreba za podizanjem nivoa apstrakcije stvorila je osnovu za kreiranje sofisticiranih softverskih rešenja prilagođenih optimizaciji i automatizaciji problema koji se rešavaju.

Neke od dobrobiti procesa automatizacije softvera su: ušteda vremena, smanjenje napora koji je potrebno da korisnik softvera uloži, nije neophodno domensko poznavanje problema koji se rešava i sl. Automatizacija danas ima važnu ulogu u savremenim modelima nastave jer pozitivno utiče na poboljšanje njenog kvaliteta.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Popov, vanr. prof.

2. IDEJNI PUT RAZVOJA APLIKACIJE SPRINGEXTRACTOR

Ideja o razvoju aplikacije koja bi vršila automatizaciju procesa pregledanja radova nastala je usled problema sa vremenom: prevelik broj studenata, a nedovoljno nastavnog osoblja do sada je značilo dugotrajno čekanje na rezultate održanog testa/ispita. Da bi se taj problem rešio bilo bi dobro imati softverski alat koji bi mogao da pripremi studentske radove za ocenjivanje, kao i da popuni tabelu za unos poena neophodnim podacima (podaci o studentu i skala za ocenjivanje).

O problemu automatizacije diskutovalo se u nekolicini naučnih radova. Jedan od njih jeste „*Artificial intelligence, automation and work*“ [1]. U ovom radu autori su pažnju posvetili veštačkoj inteligenciji koja je iskorišćena za obavljanje poslova koji su automatizovani, a koji su prvobitno bili povereni ljudima.

„*The Benefits and Generic Procedure of Automating and Academic Student System in Primary and Secondary Schooles as an Impetus for Educational Technology*“ [2] primer je naučnog rada koji fokus stavlja na opis softvera razvijenog sa ciljem beleženja razlika između uspeha ostvarenog u osnovnim i srednjim školama, kao i razlike u uspehu učenika privatnih i državnih škola. Automatizacija je izvršena u domenu preuzimanja predatih učeničkih radova i njihovoj pripremi za ocenjivanje (nastavnik se prijavljuje na sistem i za preuzimanje mu postaju dostupni radovi koje treba da pregleda).

Glavna teza koja je potvrđena prilikom implementacije softvera razvijenih u svrhu detaljne analize izložene u prethodno navedenim radovima jeste da je ostavljen prostor za automatizaciju pojedinih kritičnih tački sistema, ali uz izvesnu dozu opreza kako se ne bi došlo u situaciju da prekomerna upotreba automatizacije izazove neželjene efekte.

3. SERVERSKA STRANA: RADNI OKVIR SPRING

Aplikacije pisane u programskom jeziku *Java* mogu se razvijati uz oslonac na radni okvir *Spring* čija je glavna karakteristika koncept primene inverzije kontrole (eng. *dependency injection*). Prednost ovog radnog okvira je to što enkapsulira module pomoću kojih je moguće izgenerisati inicijalni *Spring* projekat sa svim osnovnim podešavanjima neophodnim za dalji razvoj [1].

Tehnologija čijoj upotrebi danas pribegava velik broj programera koji rade na implementaciji veb orijentisanih softverskih rešenja, koju odlikuje jednostavnost u pogledu podešavanja konfiguracije, mogućnost rada sa lokalnim serverom i brzo dobijanje osnovne projektne strukture

buduće veb aplikacije, između ostalog ostavlja i prostor za manipulaciju zavisnostima, upotrebu predefinisanih ali i sopstvenih anotacija, asinhrono izvršavanje zadataka (`@Async`), kao i automatsko generisanje inicijalnih konfiguracionih fajlova.

3.1. Arhitektura SpringExtractor aplikacije za automatizaciju procesa pregledanja radova

Izgradnja ***SpringExtractor***-a oslanjala se na radni okvir *Spring* u kombinaciji sa *Maven* alatom koji pruža mogućnost jednostavnog uvezivanja sa već postojećim bibliotekama klasa.
Konvencija *Spring*-a nalaže podelu sastavnih delova *Spring* projekta po slojevima:

- sloj modela - sačinjen je od model klasa pomoću kojih je apstrahovan deo realnog sveta koji predstavlja predmet interesovanja aplikacije u razvoju, anotacija `@Entity`
- repozitorijum – interfejsi namenjeni pojednostavljenom procesu komunikacije sa skladištem podataka kojima aplikacija manipuliše, anotacija `@Repository`
- servisni sloj - predstavlja vezu između baze podataka i podržanih *backend* funkcionalnosti (interfejsi i klase koje ih implementiraju), anotacija `@Service`
- sloj kontrolera - sadrži aplikativnu (biznis) logiku sistema koji se implementira. Često se upotrebljava u kombinaciji sa *REST* arhitektonskim šablonom zasnovanim na principu postojanja jedinstvene putanje na kojoj se nalazi svaki od resursa, anotacija `@Controller`
- *DTO* – opcioni sloj koji predstavlja posrednika u komunikaciji između klijentske i serverske strane, sa ciljem izbegavanja direktne komunikacije

3.2 Uporedni prikaz strukture aplikacija PHPExtractor i SpringExtractor

Suštinska razlika između ***PHPExtractor*** i ***SpringExtractor*** aplikacija je u tome što je prvobitna verzija u okviru jednog projekta enkapsulirala i prezentacioni i upravljački sloj. Naredna verzija predstavljala je pokušaj relaksacije gradivnih komponenti aplikacije koje su raspoređene u dve nezavisne projektne celine – kreiran je zaseban projekat za prezentacioni sloj, kao i poseban projekat koji sadrži biznis logiku.

Glavni problem na koji se naišlo prilikom razvoja ***PHPExtractor***-a jeste složen postupak kreiranja komponenti *MVC* arhitekture. Naime, nedostatak *user-friendly* okruženja iziskivao je stalnu potrebu za radom sa terminalom. Da bi se napravio svaki od slojeva aplikacije, bilo je potrebno ručno kucati komande kojima se kreiraju klase određene namene. Nasuprot ovakvom načinu rada nalazi se integrisano razvojno okruženje *Eclipse* koje ostavlja prostor da se prilikom implementacije ***SpringExtractor***-a jednostavnim klikom na željenu opciju izgeneriše ono što je potrebno.

Nedostatak radnog okvira *Spring* sadržan je u činjenici da je *CRUD* operacije neophodno ručno napisati, dok je *Laravel* koncipiran tako da programeru olakšava posao tako što od njega zahteva samo da pozove odgovarajuću metodu. Problem bezbednosti *Laravel* uspešno prevazilazi zahvaljujući postojanju ugrađenog posrednika (eng. *middleware*) koji filtrira pristigle *HTTP* zahteve, dok *Spring* zahteva da zaštita veb aplikacije u razvoju bude ručno implementirana upotrebom *Spring security*-a. Najvažnija funkcionalnost – preuzimanje arhive studentskih radova dobijene od administratora i njeno raspakivanje – uspešno je implementirana u obe verzije aplikacije. Prva verzija aplikacije je u tu svrhu koristila eksterne dodatke (eng. *plugin*) koji zahtevaju posebnu instalaciju. *Spring* to izbegava zahvaljujući postojanju *maven* zavisnosti (navode se u konfiguracionom *pom.xml* fajlu). Nedostatak koji se pojavio prilikom rada sa *Spring* radnim okvirom odnosi se na to da se unapred mora znati tip podataka koji se prosleđuju putem mreže. Sa ovim problemom prethodna verzija aplikacije nije imala dodira zato što je postojala univerzalna promenljiva `$request` iz koje su preuzimani potrebni podaci i koji su zatim mapirani na odgovarajuće elemente.

4. BAZA PODATAKA

Pojam baze podataka objašnjava se kao prostor za skladištenje, organizaciju, čuvanje i pristup podacima. Najvažnija karakteristika svake baze podataka jeste očuvanje integriteta, konzistentnosti i neredundantnosti uskladištenih podataka.

4.1. MySQL relaciona baza podataka

MySQL je relaciona baza podataka koja obezbeđuje tabelarni prikaz podataka koji se u njoj nalaze. Prednost *MySQL*-a jeste direktno mapiranje iz objektnog u relacioni model podataka (uz posredstvo *Hibernate ORM*-a) [2]. Objašnjenje zašto *MySQL* ima široku primenu među veb aplikacijama sadržano je u činjenici da je reč o multiplatformskoj bazi podataka, što može da sarađuje sa velikim brojem programskih jezika, što može da radi sa ogromnom količinom podataka, što je besplatna, otvorenog koda i što za izražavanje upita koristi intuitivnost i jasnoću *SQL*-a.

4.2. MySQL baza podataka u kombinaciji sa PHPExtractor-om i SpringExtractor-om

U prvoj verziji aplikacije prilikom kreiranja svake klase modela trebalo je odmah kreirati i korespondirajuću šemu. Da bi šema bila kompletna trebalo je ručno popuniti polja koja će se naći u odgovarajućoj tabeli. Pored zadavanja imena poljima i dodeljivanja njihovih tipova, eksplicitno se morao navesti i naziv same šeme. Na kraju je potrebno bilo odraditi još i migraciju.

Naredna verzija aplikacije je optimalnija u pogledu rada sa bazom podataka jer je većina problema na koje se naišlo u *Laravel*-u prevaziđena upotrebom *Spring*-ovih anotacija. Pomoću `@Entity` anotacije u bazi podataka izgenerisaće se odgovarajuća tabela čiji naziv je naziv klase. Samo atributi klase modela ispred kojih je navedena `@Column` anotacija biće vidljivi u bazi. Sloj repozitorijuma (`@Repository`) predstavlja direktnu vezu između sloja modela i baze podataka. `JpaRepository` predstavlja *Spring*-ovu adaptaciju za rad sa bazom podataka – realizovan je tako što odabranu metodu transformiše u predefinisane / parametrizovane *SQL* upite.

Po pitanju baze podataka ispostavilo se da su oba pristupa ravnopravna u pogledu načina rukovanja sa podacima, transformacije iz objektnog u relacioni model i po pitanju održavanja konzistentnog stanja uskladištenih podataka.

5. KLIJENTSKA STRANA

SpringExtractor razvijen je uz oslonac na *Angular* frejmvork.

Razlog zbog kog je odabran baš taj radni okvir jeste to što sadrži velik broj biblioteka koje omogućavaju automatsko podešavanje i generisanje inicijalnih komponenti koje potom treba nadograditi u skladu sa zahtevima aplikacije.

5.1. Angular frejmvork

Angular je *TypeScript* jezik i predstavlja rešenje kom se pribegava kada se radi na razvoju modernih aplikacija složenih korisničkih zahteva.
Neke od pogodnosti koje nudi ovaj alat su: brzina kompajliranja, uvođenje *CLI*-ja, u osnovi se koristi samo jedna stranica koja se prikazuje korisniku na kojoj se smenjuju komponente od kojih je izgrađena (tzv. arhitektura zasnovana na komponentama), redukcija veličine koda (minimalan kod – maksimalan efekat) i informativnija obaveštenja o greškama [3].
Mane *Angular*-a su: nekompatibilnost pojedinih verzija koja uzrokuje potrebu za ponovnim pisanjem koda, zahtevan je za učenje zbog prevelike kompleksnosti njegovih sastavnih delova, nije pogodan za aplikacije koje zahtevaju pretragu veće količine podataka zato što u osnovi koristi *JavaScript* meta-tagove pomoću kojih se prikazuje sadržaj na stranici, teško je preći sa starije na noviju verziju (obrnuto ne važi) itd. [4].

5.2. Korisnički interfejs SpringExtractor-a

Za potrebe razvoja aplikacije ***SpringExtractor*** implementirane su sledeće komponente:

first-page (omogućava odabir uloge koju će korisnik koji pristupa sistemu imati u daljem radu: profesor/student),

home (registrovani korisnik u ulozi profesora ima mogućnost otpremanja arhive studentskih radova i bodovne skale za realizovani test),

finish (omogućava preuzimanje raspakovanih studentskih radova sa delimično popunjenom tabelom za unos ostvarenih bodova, otpremanje u međuvremenu potpuno popunjene *.xls* tabele i iniciranje njene konverzije u *.pdf* format),

student-home (komponenta koja služi za prikaz rezultata održanih testova u *.pdf* formatu),

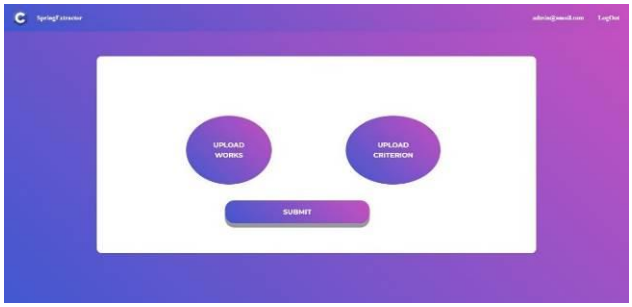
registration (izlistava formu koju je potrebno popuniti podacima neophodnim za registraciju korisnika) i

login (namenjena prethodno registrovanim korisnicima – očekuje unos korisničkih kredencijala za pristup naprednim funkcionalnostima). Na slici 1 predstavljen je korisnički interfejs ***SpringExtractor***-a.

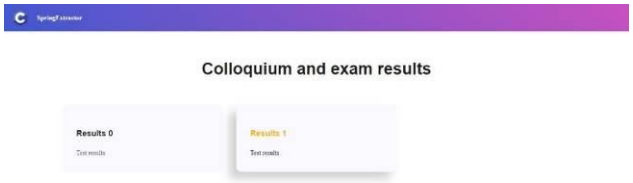
Na slici 2 može se videti izgled stranice za prikaz fajlova sa rezultatima testa.

Sastavni deo ***SpringExtractor*** aplikacije su: `home-service`, `finish-service`, `student-home-service`, `registration-service` i `login-service`.

Svaki od navedenih servisa implementira poslovnu logiku, tj. funkcionalnosti koje treba da se dese kao odgovor na akciju korisnika.



Slika 1. *Korisnički interfejs: home-page*



Slika 2. *Korisnički interfejs: student-home*

5.3. Komparacija prezentacionog sloja PHPExtractor-a i SpringExtractor-a

Konstatacija do koje se došlo nakon što su implementirane aplikacije ***PHPExtractor*** i

SpringExtractor jeste da je tehnologije korišćene pri razvoju ***PHPExtractor***-a optimalnije upotrebiti kada se radi na manjim aplikacijama koje nemaju potrebu da često dobavljaju podatke iz baze. Ako su zahtevi aplikacije takvi da podrazumevaju učestalo manipulisanje podacima uskladištenim u bazi, pametnije je odabrati koncept frejmvorka i razdvajanja klijentske i serverske strane (pristup koji je korišćen u implementaciji ***SpringExtractor***-a).

6. NAPREDNE FUNKCIONALNOSTI SPRINGEXTRACTOR-A

Napredna verzija aplikacije za automatizaciju procesa pregledanja radova enkapsulirala je implementaciju pojedinih mehanizama zaštite (validacija podataka, uvođenje uloga i oporavak u slučaju zaboravljene lozinke) i naprednih funkcionalnosti (registracija korisnika; prijava na sistem i odjava sa sistema; automatsko popunjavanje zaglavlja tabele adaptivnom bodovnom skalom; preuzimanje, parsiranje i ponovno arhiviranje kolekcije studentskih radova; konverzija izvornog *.xls* formata fajla sa rezultatima u *.pdf* format; preuzimanje konvertovanog fajla i mogućnost njegove dalje distribucije).

6.1. Registracija

Kako bi se novi korisnik registrovao na sistem potrebno je da popuni formu traženim podacima, nakon čega se upućuje zahtev ka serverskoj strani (poziva se metoda `registration`). Uspešnoj registraciji prethodi validacija korisničkog unosa i provera da li u bazi već postoji korisnik sa unetom *email* adresom.

6.2. Prijava na sistem i odjava sa sistema

Nakon što je korisnik izvršio registraciju na sistem i aktivirao svoj korisnički nalog posetom aktivacionog linka iz *mail*-a, omogućena mu je prijava za šta je neophodan unos korisničkih kredencijala. Klikom na dugme `Login` poziva se istoimena *backend* metoda.
Odjava sa sistema vrši se odabirom opcije `LogOut`. Na serverskoj strani ova metoda implementirana je tako da ima zadatak da ukloni aktivnog korisnika iz sesije.

6.3. Zaboravljena lozinka

U slučaju da korisnik dođe u situaciju da je zaboravio lozinku za pristup sistemu, može zahtevati da mu se automatski izgeneriše nova. Ova funkcionalnost radi na sledećem principu: u bazi se prvo pronađe korisnik sa unetom *email* adresom (validira se korisnički unos na klijentskoj strani), potom se pozove metoda koja generiše nasumičan string dužine deset karaktera sastavljen od alfanumerika (u tu svrhu iskorišćena je `RandomStringUtils` klasa uz prethodno dodavanje odgovarajuće *maven* zavisnosti u *pom.xml* fajl), da bi se zatim dobijeni string postavio kao vrednost nove lozinke. Kako bi korisnik znao sa kojom se lozinkom nadalje može prijaviti, sistem mu automatski šalje *email* sa tom informacijom.

6.4. Adaptivno bodovanje

Zapakovana kolekcija studentskih radova u sebi sadrži podarhive u kojima su smešteni radovi. Kada se ova arhiva otpremi, potrebno je odabrati još i kriterijum po kom će biti izvršeno bodovanje nakon čega je dozvoljeno kliknuti na dugme `submit`. ***SpringExtractor*** omogućio je dinamičko unošenje kriterijuma bodovanja: zaglavlje tabele popunjava se isparsiranim sadržajem fajla koji predstavlja bodovne stavke.

6.5. Preuzimanje i konverzija popunjene .xls tabele u .pdf format

Kada se korisnik nađe na *finish* stranici može odabrati opciju `download works` čiji je zadatak preuzimanje prethodno programski zapakovanog foldera koji sadrži raspakovane studentske radove i tabelu sa popunjenim zaglavljem i podacima o studentima. Proširena verzija aplikacije podržava mogućnost konverzije formata fajla sa rezultatima. Da bi bila pozvana metoda koja vrši tu funkciju korisnik treba da otpremi željeni fajl u *.xls* formatu i da potom klikne na dugme `convert excel to pdf`.

7. ZAKLJUČAK

U radu je predstavljen pokušaj implementacije softverskog rešenja koje bi automatizovalo proces pregledanja studentskih radova.

SpringExtractor uspešno je automatizovao prvu fazu procesa pregledanja radova – raspakivanje otpremljene arhive sa radovima i pripremu tabele za ocenjivanje. Nedostatak prve verzije aplikacije koji se odnosi na to da kandidati moraju ispoštovati određenu formu prilikom unosa ličnih podataka prevaziđen je tako što je implementirano parsiranje fajla sa podacima o svim studentima dobijenog od administratora.

Rešeno je ograničenje koje se ticalo vrste testa i bodovne skale za isti – dozvoljeno je da korisnik u ulozi profesora otpremi adaptivnu skalu za bodovanje. Uvedena su prava pristupa tako da samo registrovani korisnici mogu obavljati sve operacije kojima aplikacija raspolaže. Neregistrovani korisnici imaju samo *readonly* pravo pristupa nad dokumentom sa rezultatima testa.
Kada se bude radilo na implementaciji naredne verzije rešenja akcenat će biti stavljen na automatizaciju druge faze, tj. na sam proces pregledanja radova. Plan je da se pokuša napraviti skripta koja bi prolazila kroz priloženo rešenje svakog studenta i analizirala programski kod na osnovu kog bi generisala predlog ostvarenih bodova za odgovarajuću bodovnu stavku.
Ako bi se došlo do stadijuma da je polazna problematika u potpunosti uspešno automatizovana, to bi značilo da su stvoreni preduslovi za proširenje oblasti primene softverskih rešenja čiji je akcenat na procesu automatizacije.

8. LITERATURA

[1] Spring Framework, https://spring.io/projects/spring-framework, pristupljeno: 06.09.2020.
[2] Baza podataka – definicija, vrste i modeli, https://kompjuteras.com/baze-podataka/?script=lat, pristupljeno: 06.09.2020.
[3] Angular: Best Use Cases and Reasones To Opt For This Tool, https://yalantis.com/blog/when-to-use-angular/, pristupljeno: 07.09.2020.
[4] Angular tutorials point, https://www.tutorialspoint.com/angular7/angular7_tutorial.pdf, pristupljeno: 07.09.2020.

Kratka biografija:

Aleksandar Vasiljević rođen je 02.02.1996. u Prizrenu. Završio je osnovnu školu „Mirko Jovanović“ u Kragujevcu. Nakon završene osnovne škole upisuje srednju tehničku školu „Prva tehnička škola“ u Kragujevcu. 2015. godine upisuje „Fakultet tehničkih nauka“ u Novom Sadu, smer *Računarstvo i automatika*. Školske 2017/18. godine opredeljuje se za usmerenje *Primenjene računarske nauke i informatika*. Školske 2018/19. godine upisuje modul *Internet i elektronsko poslovanje*. Zvanje *diplomirani inženjer elektrotehnike i računarstva* dobija 30.08.2019. Školske 2019/20. godine upisuje master akademske studije, smer *Primenjene računarske nauke i informatika – Elektronsko poslovanje*. Zvanje *saradnik u nastavi* na Departmanu za računarstvo i automatiku na Fakultetu tehničkih nauka u Novom Sadu dobija 28.10.2019. kontakt: alexva02@uns.ac.rs