|  |
| --- |
| PMF Cirilica.wmf |
| Институт за математику и информатику  Природно-математички факултет  Универзитет у Крагујевцу |
| Семинарски рад |
| **Практикум из Објектно-орјентисаног програмирања**  **Пројекат ОгласИМИ** |
| **Ментор Студенти**  **др Марина Свичевић Данило Новаковић 101/18**  **Иван Миљковић 100/18**  **Лука Марковић 97/18**  **Лука Јевтић 64/17** |
| Август 2021. |

**Садржај**

[Увод 3](#_Toc81153619)

[1. Опис структуре апликације 4](#_Toc81153620)

[1.1. База података 4](#_Toc81153621)

[1.2. Ентитетска класа 5](#_Toc81153622)

[1.3. Persistance слој 7](#_Toc81153623)

[1.4. Слој пословне логике 8](#_Toc81153624)

[1.5. Презентациони слој 10](#_Toc81153625)

[Литература 13](#_Toc81153626)

Увод

Тренд глобализације је донео много новина у области запошљавања, тако да више не постоје границе у пласирању понуда радних места у било које делове света. Међутим, питање је колико можемо бити сигурни у фирме које се оглашавају и исправност огласа. Додатно неповерење и конфузију међу онима који траже посао уносе и све агресивније маркетиншке кампање путем дигиталних медија

Циљ пројекта ОгласИМИ је развој интернет платформе за оглашавање и пријављивање на конкурсе за посао, која ће корисницима омогућити директну размену информација о понуди и потражњи радних места, без посредника и без могућности манипулације подацима. Мултиплатформска wеб апликација ће корисницима омогућити приступ са различитих врста рачунара и мобилних уређаја.

1. Опис структуре апликације

Web апликације направљене у Spring Boot-у треба да поштују Model-View-Controller архитектуру, тако да ће сваки слој архитектуре бити засебно обрађен. Апликација треба кориснику ,у зависности од тога који је тип корисника у питању, да омогући да прегледа списак огласа, да их претражује, да се пријави на оглас ако је радник или да постави сопствени оглас ако је послодавац. Ми ћемо овде проћи само кроз цео ток реализације функционалности огласа ради сажетости, остале функционалности су реализоване на веома сличан начин.

* 1. База података

Да би Spring Boot апликација могла да чува податке и прати стање потребно је направити базу података. У овом случају, користићемо MySQL релациону базу података. Подаци који су потребни за адекватно чување једног огласа су:

Радна позиција,

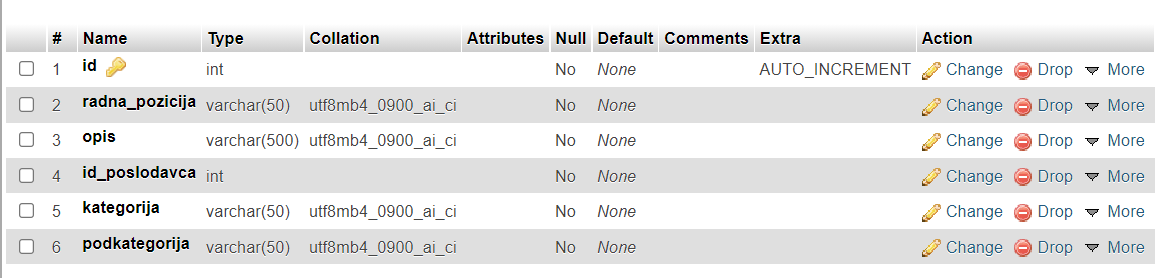
Опис радног места,

ID послодавца који је поставио оглас,

Категорија занимања,

Подкатегорија занимања,

Потребно је додати и ID који ће јединствено референцирати на сваки оглас у бази података и који ће се аутоматски повећавати за сваки унос.



* 1. Ентитетска класа

Ентитетска класа је класа која за задатак има да направи објекте које ће касније Spring Boot претворити у зрна. Spring објекте ентитетске класе претвара у зрна да би могао лакше да повлачи податке из базе података и смешта их у објекте класе. Ентитетска класа треба да буде POJO класа (Plain old Java class) што значи да не сме да имплементира или наслеђује друге интерфејсе и класе, већ сме само да има поља, конструкторе, get и set методе за сва поља, као и toString методу.

Да би Spring Boot-у нагласили да је класа ентитетска користимо анотацију @Entity. Да би објекти класе могли да добијају податке из базе података потребно је да ентитетску класу повежемо са табелом у којој чувамо те податке. Повезивање ентитетске класе и табеле из базе података радимо анотацијом @Table, у коју уписујемо назив табеле са којом желимо да повежемо класу. Да би могли да користимо ове анотације морамо да позовемо библиотеку “javax.persistence”.

Да би поља класе повезали са правим колонама у бази користимо анотацију @Column, у коју уносимо назив колоне у табели са којој желимо да повежемо поље. Ако нам је потребно да чувамо поље ID које је једнозначно за сваку колону поред анотације @Column и назива поља које смо поставили као примарни кључ користимо и анотацију @Id. Ако желимо да нам се вредност примарног кључа аутоматски генерише и увећава морамо да користимо анотацију @GeneratedValue и унутар њега подесимо начин генерисања примарног кључа.

Све наведено приказано је на примеру огласа. Креирана је POJO класа “Оглас” са свим пољима које смо раније дефинисали, get и set методама за сва поља, конструкторима, toString методом. Анотацијом постављамо ову класу на ентитетску класу, и повезујемо је са табелом из базе података. Сва поља класе су повезана са правим колонама, а поље id је означено анотацијом @Id, а @GeneratedValue је постављен тако да база сама генерише кључеве (GenerationType.IDENTITY).

package com.Oglasi.OglasiIMI.entity;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "oglas")  
public class Oglas {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id")  
 private int id;  
  
 @Column(name = "radna\_pozicija")  
 private String radnaPozicija;  
  
 @Column(name = "opis")  
 private String opis;  
  
 @Column(name = "id\_poslodavca")  
 private int idPoslodavca;  
  
 @Column(name = "kategorija")  
 private String kategorija;  
  
 @Column(name = "podkategorija")  
 private String podkategorija;  
  
 public Oglas(){}  
  
 public Oglas(int id, String radnaPozicija, String opis, int idPoslodavca, String kategorija, String podkategorija) {  
 this.id = id;  
 this.radnaPozicija = radnaPozicija;  
 this.opis = opis;  
 this.idPoslodavca = idPoslodavca;  
 this.kategorija = kategorija;  
 this.podkategorija = podkategorija;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String getRadnaPozicija() {  
 return radnaPozicija;  
 }  
  
 public void setRadnaPozicija(String radnaPozicija) {  
 this.radnaPozicija = radnaPozicija;  
 }  
  
 public String getOpis() {  
 return opis;  
 }  
  
 public void setOpis(String opis) {  
 this.opis = opis;  
 }  
  
 public int getIdPoslodavca() {  
 return idPoslodavca;  
 }  
  
 public void setIdPoslodavca(int idPoslodavca) {  
 this.idPoslodavca = idPoslodavca;  
 }  
  
 public String getKategorija() {  
 return kategorija;  
 }  
  
 public void setKategorija(String kategorija) {  
 this.kategorija = kategorija;  
 }  
  
 public String getPodkategorija() {  
 return podkategorija;  
 }  
  
 public void setPodkategorija(String podkategorija) {  
 this.podkategorija = podkategorija;  
 }  
}

* 1. Persistance слој

У овом слоју MVC архитектуре Spring Boot апликације потребно је декларисати све методе које су нам потребне за повлачење података из базе података. Да би наш интерфејс могао да комуницира са базом података, и тако постане репозиторијум, потребно је да наследи методе апстрактне класе JpaRepository. Да би интерфејс који имплементира ову апстрактну класу могао да комуницира са базом потребно је да му проследимо назив ентитетске класе као и тип података примарног кључа базе. Неке од основних метода наслеђених из JpaRepository класе су за проналажење свих података из одређене табеле у бази, проналажење податка по вредности његовог ID-a, брисање и чување података. Да би интерфејс који смо креирали као persistence слој апликације могао да проналази податке које не може да нађе преко метода апстрактне класе JpaRepository потребно је да га означимо анотацијом @Repository. Тиме интерфејс добија могућност претраживања података из базе по различитим критеријумима, као што су одређене колоне, вредности које су мање или веће од одређене вредности итд.

Након креираног интерфејса “OglasRepo“ који проширује апстрактну класу JpaRepository, којој се прослеђује ентитетска класа “Оглас”, и означавања интерфејса анотацијом @Repository, потребно је имплементирати методе које нису имплементиране. Анотација @Repository означава да је класа репозиторијум. Репозиторијум представља механизам за складиштење, повлачење и претрагу података који се налазе у бази података која је везана за пројекат.

package com.Oglasi.OglasiIMI.repo;  
  
import com.Oglasi.OglasiIMI.entity.Oglas;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import java.util.List;  
  
@Repository  
public interface OglasRepo extends JpaRepository<Oglas, Integer> {  
  
 public List<Oglas> findAllById(int id);  
  
 public List<Oglas> findAllByIdPoslodavca(int idPoslodavca);  
  
 public List<Oglas> findAll();  
  
 public List<Oglas> findAllByKategorija(String kategorija);  
  
 public List<Oglas> findAllByRadnaPozicijaIsLike(String radnaPozicija);  
  
  
}

* 1. Слој пословне логике

У овом слоју MVC архитектуре обављају се операције које су потребне да би се подаци добијени из базе правилно обрадили и тако обрађени проследили кориснику. Друга улога овог слоја је рад са наредбама добијеним од корисника. Баш из тог разлога, овај слој најчешће називамо слој сервиса, јер опслужује корисника.

Да би овај слој могао да извршава све потребне операције које може да извршава сервисни слој апликације потребно је да класу означимо анотацијом @Service. Да би сервисни слој могао да добија податке из репозиторијума потребно је да спојимо објекте класе репозиторијума и класе сервиса. Пошто су ово класе Spring Boot апликације, које производе објекте типа Bean (зрно), потребно је њихово спајање у самој апликацији. Два објекта спајамо тако што објекат који доставља податке “ожичимо” са објектом коме се достављају подаци. То се ради тако што се поменути објекат, креиран у класи објекта коме се достваљамо податке, означи анотацијом @Autowired.

На примеру нашег пројекта можемо да видимо како изгледа једна сервисна класа. Пошто се водимо логиком “програмирање према интерфејсу”, не би ли смањили зависност сервисне класе од метода које користи, креирамо интерфејс “OglasService”. Овај интерфејс има за задатак да дефинише методе које ће сервисна класа да користи.

package com.Oglasi.OglasiIMI.service;  
  
import com.Oglasi.OglasiIMI.entity.Oglas;  
  
import java.util.List;  
  
public interface OglasService {  
  
 public List<Oglas> findAllById(int id);  
  
 public List<Oglas> findAllByIdPoslodavca(int idPoslodavca);  
  
 public List<Oglas> findAll();  
  
 public List<Oglas> findAllByKategorija(String kategorija);  
  
 public List<Oglas> findAllByRadnaPozicijaIsLike(String radnaPozicija);  
  
}

Након тога, креирамо класу “OglasServiceImpl”, коју означавамо анотацијом @Service. Затим ова класа имплементира горе наведени интерфејс да би могла да користи методе које смо дефинисали у њему. Да би могли да добијемо податке из репозиторијума, потребно је да унутар ове класе креирамо објекат репозиторијумске класе (“private OglasRepo oglasRepo”) и ожичимо га. Тако смо сервисној класи омогућили да користи методе репозиторијума

package com.Oglasi.OglasiIMI.service;  
  
import com.Oglasi.OglasiIMI.entity.Oglas;  
import com.Oglasi.OglasiIMI.repo.OglasRepo;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class OglasServiceImpl implements OglasService{  
  
 @Autowired  
 private OglasRepo oglasRepo;  
  
 @Override  
 public List<Oglas> findAllById(int id) {  
 return oglasRepo.findAllById(id);  
 }  
  
 @Override  
 public List<Oglas> findAllByIdPoslodavca(int idPoslodavca){  
 return oglasRepo.findAllByIdPoslodavca(idPoslodavca);  
 }  
  
 @Override  
 public List<Oglas> findAll() {  
 return oglasRepo.findAll();  
 }  
  
 @Override  
 public List<Oglas> findAllByKategorija(String kategorija) {  
 return oglasRepo.findAllByKategorija(kategorija);  
 }  
  
 @Override  
 public List<Oglas> findAllByRadnaPozicijaIsLike(String radnaPozicija) {  
 radnaPozicija = "%" + radnaPozicija + "%";  
 return oglasRepo.findAllByRadnaPozicijaIsLike(radnaPozicija);  
 }  
}

* 1. Презентациони слој

У овом слоју Spring Boot апликације кориснику се достављају обрађени подаци и прослеђују захтеви корисника за податке и друге операције. Да би то могло да се изврши овај слој мора да комуницира са корисником преко НТТР захтева. Да би једна класа могла то да ради, потребно је да је означимо анотацијом @RestController. Ова анотација спој је двеју анотација, @Controller и @ResponseBody. Анотација @Controller има задатак да пронађе све методе које су потребне за рад наведене класе, док @ResponseBody треба да омогући рад са HTTP захтевима. Да би могли да извршимо методе које се налазе унутар контролерске класе преко веб претраживача, морамо да их иницијализујемо путем линкова заједно са одговарајућим HTTP захтевом.

@RestController  
public class OglasController {  
  
 @Autowired  
 private OglasService oglasService;  
  
 @GetMapping("/sviOglasi")  
 public List<Oglas> findAll(){  
 return oglasService.findAll();  
 }

Скоро све функционалности наше веб апликације се обављају уз помоћ AJAX-a, конкретно у овом примеру повлачимо категорије огласа из базе и приказујемо само огласе одговарајуће одабране категорије.

var ***kategorijeSave*** = [];  
  
function ucitajKategorije()  
{  
  
 var url = "http://localhost:8080/sviOglasi";  
 var xhr = new ***XMLHttpRequest***()  
 xhr.open('GET', url, true)  
 xhr.onload = function () {  
 var kat = ***JSON***.parse(xhr.responseText);  
 if (xhr.readyState == 4 && xhr.status == "200") {  
  
 var kategorije=[];  
 for(var i = 0; i < kat.length ; i++)  
 {  
 kategorije[i] = kat[i].kategorija;  
 }  
  
 kategorije = [...new ***Set***(kategorije)];  
  
 ***kategorijeSave*** = kategorije;  
  
 var opcije="<option>Sve Kategorije</option>";  
  
 for(var i = 0 ; i < kategorije.length ; i++)  
 {  
 opcije += "<option value='"  
 opcije += kategorije[i]+"'>";  
 opcije += kategorije[i];  
 opcije += "</option>";  
 }  
  
 ***document***.getElementById("dropDown").innerHTML = opcije;  
  
 } else {  
 ***console***.error(kat);  
 }  
 }  
 xhr.send(null);  
}  
  
function filterKategorije()  
{  
 var kategorija = ***document***.getElementById("dropDown").value;  
  
 for(var i = 0 ; i < ***kategorijeSave***.length ; i++)  
 {  
  
 var objekti = ***document***.getElementsByClassName(***kategorijeSave***[i]);  
  
 for(var j = 0 ; j < objekti.length ; j++)  
 {  
 objekti[j].style.display = "block";  
 }  
  
 }  
  
 if(kategorija == "Sve Kategorije")  
 return;  
 else  
 {  
 for(var i = 0 ; i < ***kategorijeSave***.length ; i++)  
 {  
 if(***kategorijeSave***[i] != kategorija)  
 {  
 var objekti = ***document***.getElementsByClassName(***kategorijeSave***[i]);  
  
 for(var j = 0 ; j < objekti.length ; j++)  
 {  
 objekti[j].style.display = "none";  
 }  
 }  
 }

Са друге стране , за просто приказивање података из базе, коришћен је класичан контролер у коме су додати модели на одређене странице и за њихов приказ на HTML страницу је коришћен je “template engine” Thymeleaf.

//Svi Oglasi  
@RequestMapping("/oglasi")  
public String prikaziSveOglase(Model model)  
{  
 model.addAttribute("oglasi",oglasService.findAll());  
 return "/oglasi";  
}

<div id="sviOglasi">  
 <th:block th:each="oglas:${oglasi}">  
 <div th:class="${oglas.kategorija}">  
 <br>  
 <hr>  
 <h4 th:text = "${oglas.radnaPozicija}"></h4>  
 <div th:text = "${oglas.opis}"></div>  
 <div th:text = "${oglas.kategorija}" th:value></div>  
 <div th:text = "${oglas.podkategorija}"></div>  
 <button th:value="${oglas.id}" th:id="'oglas'+${oglas.id}" th:class="'oglasPrijavaDugme'" onclick="prijaviMe(this)">Prijavi Me</button>  
 <hr>  
 <br>  
 </div>  
 </th:block>  
</div>

Литература

[1] Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Boston, MA: Prentice Hall. ISBN: 978-0-13-449416-6

[2] Oracle документација Java програмског језика доступно на:

[Java Platform Standard Edition 8 Documentation (oracle.com)](https://docs.oracle.com/javase/8/docs/) (посећено 20.05.2021)

[3] Документација Spring framework-a, доступно на:

[Spring Framework Documentation](https://docs.spring.io/spring-framework/docs/current/reference/html/) (посећено 20.05.2021)

[4] Walls, C. (2016). Spring Boot in action. Manning Publications.

Универзитет у Крагујевцу

Природно-математички факултет

Институт за математику и информатику

Завршни рад под називом

ОгласИМИ

одбрањен је \_\_\_\_\_\_\_\_\_\_\_\_.

МЕНТОР:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

др Име Презиме, звање, Институција

ЧЛАНОВИ КОМИСИЈЕ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

др Име Презиме, звање, ПМФ Крагујевац

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

др Име Презиме, звање, Институција

Завршни рад је оцењен оценом \_\_\_\_\_\_\_\_\_\_\_\_.